



Panduan Pengguna

AWS CloudHSM



AWS CloudHSM: Panduan Pengguna

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan antara para pelanggan, atau dengan cara apa pun yang menghina atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Apa itu AWS CloudHSM?	1
Kasus penggunaan	2
Cara kerjanya	4
klaster	5
Pengguna di AWS CloudHSM	6
Kunci di AWS CloudHSM	6
Klien SDKs	7
Pencadangan	8
Wilayah yang Didukung untuk AWS CloudHSM	10
Harga untuk AWS CloudHSM	10
Mulai menggunakan	11
Buat administrator IAM	11
Buat grup pengguna dan administrator IAM	12
Buat VPC	14
Buat klaster	15
Tinjau grup keamanan klaster	19
Luncurkan klien EC2	20
Konfigurasikan grup keamanan instans EC2	23
Langkah 1. Mengubah grup keamanan default	23
Langkah 2. Hubungkan instans Amazon EC2 ke cluster AWS CloudHSM	24
Buat HSM	25
Verifikasi identitas HSM (opsional)	27
Langkah 1. Dapatkan sertifikat dari HSM	29
Langkah 2. Dapatkan sertifikat root	32
Langkah 3. Verifikasi rantai sertifikat	32
Langkah 4. Ekstrak dan bandingkan kunci publik	34
Inisialisasi cluster	34
Ikhtisar	35
Langkah 1. Dapatkan CSR cluster	36
Langkah 2. Buat kunci pribadi untuk Root CA Anda	38
Langkah 3. Tanda tangani CSR	39
Langkah 4. Inisialisasi cluster	49
Instal CloudHSM CLI	51
Aktifkan cluster	55

Pengaturan mTL (disarankan)	59
Langkah 1. Buat dan daftarkan jangkar kepercayaan ke HSM	59
Langkah 2. Aktifkan mTL untuk AWS CloudHSM	63
Langkah 3. Tetapkan penegakan mTLS untuk AWS CloudHSM	68
Buat dan gunakan kunci di AWS CloudHSM	70
Praktik terbaik	72
Manajemen klaster	72
Skala klaster Anda untuk menangani lalu lintas puncak	72
Arsitek cluster Anda untuk ketersediaan tinggi	72
Memiliki setidaknya tiga HSMs untuk memastikan daya tahan untuk kunci yang baru dihasilkan	73
Akses aman ke klaster Anda	73
Kurangi biaya dengan menskalakan kebutuhan Anda	73
Manajemen pengguna	74
Lindungi kredensi pengguna HSM Anda	74
Memiliki setidaknya dua admin untuk mencegah penguncian	74
Aktifkan kuorum untuk semua operasi manajemen pengguna	75
Buat beberapa pengguna crypto, masing-masing dengan izin terbatas	75
Manajemen kunci	75
Pilih jenis tombol yang tepat	75
Kelola batas penyimpanan utama	75
Mengelola dan mengamankan pembungkus kunci	76
Integrasi aplikasi	77
Bootstrap SDK Klien Anda	77
Otentikasi untuk melakukan operasi	77
Mengelola kunci secara efektif dalam aplikasi Anda	78
Gunakan multi-threading	79
Menangani kesalahan pelambatan	79
Integrasikan percobaan ulang pada operasi cluster	79
Menerapkan strategi pemulihan bencana	80
Terhuyung-huyung penerapan aplikasi Anda	80
Memantau	81
Pantau log klien	81
Pantau log audit	81
Monitor AWS CloudTrail	81
Pantau CloudWatch metrik Amazon	82

klaster	83
Arsitektur cluster	83
Sinkronisasi cluster	85
Ketersediaan klaster tinggi dan penyeimbangan beban	86
Mode cluster	87
Jenis HSM	88
Menghubungkan ke cluster	90
Tempatkan sertifikat penerbitan pada setiap instans EC2	90
Tentukan lokasi sertifikat penerbitan	90
Bootstrap Klien SDK	93
Penskalaan HSMs	96
Menambahkan HSM	97
Hapus HSM	99
Menghapus klaster	100
Membuat cluster dari backup	101
Buat cluster dari cadangan (konsol)	101
Buat cluster dari backup (AWS CLI)	102
Buat cluster dari backup (API)AWS CloudHSM	103
Migrasi tipe cluster HSM	103
Migrasi dari hsm1.medium ke hsm2m.medium	104
Pengguna HSM	112
Manajemen pengguna dengan CloudHSM CLI	112
Prasyarat	113
Jenis pengguna	115
Tabel izin	116
Buat admin	118
Buat CUs	119
Daftar semua pengguna	120
Ubah kata sandi	120
Hapus pengguna	122
Kelola MFA pengguna	123
Mengelola otentikasi kuorum (M atau N)	138
Manajemen pengguna dengan CMU	164
Prasyarat	164
Jenis pengguna	168
Tabel izin	170

Buat pengguna	172
Daftar semua pengguna	174
Ubah kata sandi	176
Hapus pengguna	179
Kelola pengguna 2FA	181
Menggunakan CMU untuk mengelola otentikasi kuorum	190
Kunci	210
Sinkronisasi kunci dan daya tahan	210
Konsep	211
Memahami sinkronisasi kunci	212
Ubah pengaturan daya tahan kunci klien	213
Menyinkronkan kunci di seluruh cluster kloning	219
Pembungkus kunci AES	219
Algoritme yang didukung	219
Menggunakan pembungkus kunci AES AWS CloudHSM	221
Kunci tepercaya	223
Memahami kunci tepercaya	223
Atribut kunci tepercaya	223
Cara menggunakan kunci tepercaya untuk membungkus kunci data	224
Cara membuka kunci data dengan kunci tepercaya	227
Manajemen kunci dengan CloudHSM CLI	228
Hasilkan kunci	228
Hapus kunci	235
Bagikan dan batalkan berbagi kunci	237
Filter dengan kunci	245
Tandai kunci sebagai tepercaya	252
Mengelola otentikasi kuorum (M atau N)	254
Manajemen kunci dengan KMU	275
Hasilkan kunci	275
Kunci impor	277
Kunci ekspor	279
Hapus kunci	282
Bagikan dan batalkan berbagi kunci	282
Tandai kunci sebagai tepercaya	283
Cadangan cluster	284
Menggunakan cadangan	284

Menghapus kunci kedaluwarsa atau pengguna yang tidak aktif	285
Mempertimbangkan pemulihan bencana	285
Hapus cadangan	285
Kembalikan cadangan	287
Konfigurasi retensi cadangan	288
Retensi cadangan terkelola	289
Menyalin cadangan di seluruh Wilayah	292
Salin cadangan ke Wilayah yang berbeda (konsol)	293
Salin cadangan ke Wilayah yang berbeda (AWS CLI)	293
Salin cadangan ke Wilayah yang berbeda (API)AWS CloudHSM	294
Bekerja dengan cadangan bersama	294
Prasyarat untuk berbagi cadangan	295
Berbagi cadangan	295
Membatalkan berbagi cadangan bersama	299
Mengidentifikasi cadangan bersama	299
Izin untuk cadangan bersama	300
Tagihan dan pengukuran	300
Cluster kloning	301
Topik terkait	301
Memberi tanda pada sumber daya	302
Tambahkan atau perbarui tag	302
Daftar tag	304
Hapus tag	304
Alat baris perintah	306
Alat konfigurasi	307
Alat konfigurasi SDK 5 klien	308
Alat konfigurasi SDK 3 klien	340
CloudHSM CLI	349
Platform yang didukung	350
Memulai	351
Mode perintah	358
Atribut kunci	360
Konfigurasi lanjutan	368
Referensi	374
AWS CloudHSM Utilitas Manajemen	628
Platform yang didukung	629

Memulai	630
Instal klien (Linux)	634
Instal klien (Windows)	637
Referensi	639
Utilitas Manajemen Kunci	699
Memulai	700
Instal klien (Linux)	704
Instal klien (Windows)	707
Referensi	708
Klien SDKs	836
Periksa versi Anda	837
Bandingkan dukungan komponen	839
Pustaka PKCS #11	839
Utilitas Manajemen CloudHSM (CMU)	840
Utilitas Manajemen Kunci (KMU)	840
Penyedia JCE	840
OpenSSL Dynamic Engine	840
Penyedia penyimpanan kunci (KSP)	841
Migrasi ke SDK terbaru	841
Migrasi pustaka PKCS #11	842
Migrasi Mesin Dinamis OpenSSL	846
Migrasi Penyedia Penyimpanan Kunci (KSP)	850
Migrasikan penyedia JCE	857
SDK Klien 5	870
Manfaat SDK terbaru	871
Platform yang didukung	872
Pustaka PKCS #11	874
OpenSSL Dynamic Engine	926
Penyedia OpenSSL	933
Penyedia penyimpanan kunci (KSP)	940
Penyedia JCE	971
Versi sebelumnya	1012
Tingkatkan SDK Klien 3	1013
Platform yang didukung	1022
Pustaka PKCS #11	1024
OpenSSL Dynamic Engine	1065

Penyedia JCE	1069
Penyedia KSP dan CNG	1100
Mengintegrasikan aplikasi pihak ketiga	1114
SSL/TLS pembongkaran	1114
Cara kerjanya	1115
Offload di Linux dengan OpenSSL Engine	1117
Offload di Linux dengan Penyedia OpenSSL	1161
Offload di Linux dengan JSSE	1191
Offload pada Windows	1202
Tambahkan penyeimbang beban (opsional)	1217
Windows Server CA	1224
Klien SDK 5 dengan Windows Server CA	1224
Klien SDK 3 dengan Windows Server CA	1230
Enkripsi basis data Oracle	1235
Mengatur prasyarat	1236
Langkah 2: Perbarui konfigurasi database Oracle	1237
Langkah 3: Hasilkan kunci enkripsi master Oracle TDE	1238
Microsoft SignTool	1240
Klien SDK 5 dengan Microsoft SignTool	1241
Klien SDK 3 dengan Microsoft SignTool	1245
Java Keytool dan Jarsigner	1250
Klien SDK 5 dengan Java Keytool dan Jarsigner	1250
Klien SDK 3 dengan Java Keytool dan Jarsigner	1262
Alat Pembuatan dan Pengeditan Microsoft Manifest	1278
Langkah 1: Siapkan prasyarat	1279
Langkah 2: Buat sertifikat penandatanganan	1280
Langkah 3: Tanda tangani file	1281
Integrasi vendor pihak ketiga lainnya	1282
Memantau	1284
Log SDK Klien	1284
Pencatatan SDK 5 klien	1285
Pencatatan SDK 3 klien	1286
AWS CloudTrail	1288
AWS CloudHSM informasi di CloudTrail	1289
Memahami entri file AWS CloudHSM log	1290
Log Audit	1291

Cara kerja logging	1292
Melihat log	1293
Menafsirkan log	1295
Referensi log	1310
CloudWatch metrik	1313
Performa	1315
Data kinerja	1315
Pelambatan HSM	1316
Keamanan	1317
Kontrol akses API dengan kebijakan IAM	1318
Tingkatkan kebijakan IAM ke IPv6	1318
Perlindungan data	1321
Enkripsi saat diam	1322
Enkripsi saat bergerak	1323
End-to-end enkripsi	1323
Cadangan cluster	1324
Manajemen identitas dan akses	1325
Memberikan izin menggunakan kebijakan IAM	1326
Tindakan API untuk AWS CloudHSM	1327
Kunci kondisi untuk AWS CloudHSM	1328
Kebijakan terkelola AWS yang telah ditentukan sebelumnya untuk AWS CloudHSM	1328
Kebijakan yang dikelola pelanggan untuk AWS CloudHSM	1328
Peran terkait layanan	1332
Validasi kepatuhan	1334
PCI-PIN FAQs	1336
penghentian	1337
Ketahanan	1339
Keamanan infrastruktur	1339
Isolasi jaringan	1339
Otorisasi pengguna	1340
Titik akhir VPC (AWS PrivateLink)	1340
Pertimbangan untuk titik akhir AWS CloudHSM VPC	1340
Membuat titik akhir VPC antarmuka untuk AWS CloudHSM	1341
Membuat kebijakan titik akhir VPC untuk AWS CloudHSM	1341
Manajemen pembaruan	1342
Pemecahan masalah	1343

AWS CloudHSM masalah yang diketahui	1343
Masalah yang diketahui untuk semua instance HSM	1344
Masalah yang diketahui untuk hsm1.medium	1350
Masalah yang diketahui untuk hsm2m.medium	1350
Masalah yang diketahui untuk pustaka PKCS #11	1354
Masalah yang diketahui untuk JCE SDK	1360
Masalah yang diketahui untuk OpenSSL Dynamic Engine	1365
Masalah yang diketahui untuk Penyedia Penyimpanan Kunci (KSP)	1368
Masalah yang diketahui untuk Penyedia OpenSSL	1370
Masalah yang diketahui untuk instans Amazon EC2 yang menjalankan Amazon Linux 2 ...	1370
Masalah yang diketahui untuk mengintegrasikan aplikasi pihak ketiga	1371
Masalah yang diketahui untuk modifikasi cluster	1371
Masalah kegagalan operasi yang diketahui menggunakan AWS CloudHSM klien versi 5.12.0 di hsm2.medium	1372
Kegagalan sinkronisasi kunci SDK 3 klien	1373
Klien SDK 3 memverifikasi kinerja	1374
Rekomendasi tes	1376
Opsi yang dapat dikonfigurasi untuk alat pkpspeed	1376
Tes yang dapat dijalankan dengan alat pkpspeed	1376
Contoh	1377
Pengguna SDK 5 klien berisi nilai yang tidak konsisten	1381
Kegagalan replikasi pengguna SDK 5 klien	1388
Masalah: Pengguna yang dipilih tidak disinkronkan di seluruh cluster	1389
Masalah: Pengguna ada di cluster tujuan dengan atribut yang berbeda	1390
Masalah: Replikasi pengguna dari hsm2m.medium ke hsm1.medium gagal	1390
Kegagalan replikasi kunci SDK 5 klien	1391
Masalah: Kunci yang dipilih tidak disinkronkan di seluruh cluster	1391
Masalah: Kunci dengan referensi yang sama ada di cluster tujuan dengan informasi atau atribut yang berbeda	1393
AWS CloudHSM kesalahan terlihat selama pemeriksaan ketersediaan kunci	1394
Mengekstrak kunci menggunakan JCE	1395
getEncoded,, atau GETS mengembalikan getPrivateExponent null	1395
getEncoded getPrivateExponent,, atau GETS mengembalikan byte kunci di luar HSM	1395
Pelambatan HSM	1395
Resolusi	1396
Jaga agar pengguna HSM tetap sinkron	1397

Koneksi hilang	1397
Log AWS CloudHSM audit tidak ada di CloudWatch	1400
Pembungkus kunci AES yang tidak sesuai	1401
Tentukan apakah kode Anda menghasilkan kunci terbungkus yang tidak dapat dipulihkan	1401
Tindakan yang harus Anda ambil jika kode Anda menghasilkan kunci terbungkus yang tidak dapat dipulihkan	1403
Menyelesaikan AWS CloudHSM kegagalan pembuatan cluster	1404
Tambahkan izin yang hilang	1404
Buat peran terkait layanan secara manual	1405
Gunakan pengguna non-federasi	1405
Mengambil log konfigurasi AWS CloudHSM klien	1406
Alat dukungan SDK 5 klien	1406
Alat dukungan SDK 3 klien	1408
Kuota	1410
Unduh	1412
Rilis terbaru	1412
Rilis Klien SDK 5: Versi 5.17.1	1413
Rilis sebelumnya	1419
Rilis usang	1463
Rilis Client SDK 5 yang tidak digunakan lagi	1463
Rilis Client SDK 3 yang tidak digunakan lagi	1478
End-of-life rilis	1488
Riwayat dokumen	1489
Pembaruan terkini	1489
Pembaruan lebih awal	1495
.....	mcdxcvii

Apa itu AWS CloudHSM?

AWS CloudHSM menggabungkan manfaat AWS cloud dengan keamanan modul keamanan perangkat keras (HSMs). Modul keamanan perangkat keras (HSM) adalah perangkat komputasi yang memproses operasi kriptografi dan menyediakan penyimpanan aman untuk kunci kriptografi. Dengan AWS CloudHSM, Anda memiliki kendali penuh atas ketersediaan tinggi HSMs yang ada di AWS Cloud, memiliki akses latensi rendah, dan akar kepercayaan yang aman yang mengotomatiskan manajemen HSM (termasuk pencadangan, penyediaan, konfigurasi, dan pemeliharaan).

AWS CloudHSM menawarkan pelanggan berbagai manfaat:

Akses ke kluster FIPS dan non-FIPS

AWS CloudHSM menawarkan cluster dalam dua mode: FIPS dan non-FIPS. Dalam mode FIPS, hanya kunci dan algoritma yang divalidasi Federal Information Processing Standard (FIPS) yang dapat digunakan. Mode non-FIPS menawarkan semua kunci dan algoritme yang didukung oleh AWS CloudHSM, terlepas dari persetujuan FIPS. Untuk informasi selengkapnya, lihat [AWS CloudHSM mode cluster](#).

HSMs adalah tujuan umum, penyewa tunggal, dan FIPS 140-2 level-3 atau FIPS 140-3 level-3 divalidasi untuk cluster dalam mode FIPS

AWS CloudHSM menggunakan tujuan umum HSMs yang memberikan lebih banyak fleksibilitas jika dibandingkan dengan layanan AWS yang dikelola sepenuhnya yang memiliki algoritme dan panjang kunci yang telah ditentukan sebelumnya untuk aplikasi Anda. Kami menawarkan HSMs yang memenuhi standar, penyewa tunggal, dan FIPS 140-2 level-3 atau FIPS 140-3 level-3 yang divalidasi untuk cluster dalam mode FIPS. Untuk pelanggan dengan kasus penggunaan di luar batasan validasi FIPS 140-2 atau FIPS 140-3 level-3, juga menawarkan cluster dalam mode non-FIPS. AWS CloudHSM Untuk informasi selengkapnya, lihat [AWS CloudHSM kluster](#).

Enkripsi E2E tidak terlihat oleh AWS

Karena bidang data Anda end-to-end (E2E) dienkripsi dan tidak terlihat oleh AWS, Anda mengontrol manajemen pengguna Anda sendiri (di luar peran IAM). Trade off untuk kontrol ini adalah Anda memiliki tanggung jawab lebih besar daripada jika Anda menggunakan layanan AWS terkelola.

Kontrol penuh atas kunci, algoritme, dan pengembangan aplikasi

AWS CloudHSM memberi Anda kontrol penuh atas algoritma dan kunci yang Anda gunakan. Anda dapat membuat, menyimpan, mengimpor, mengeksport, mengelola, dan menggunakan

kunci kriptografi (termasuk, kunci sesi, kunci token, kunci simetris, dan pasangan kunci asimetris). Selain itu, AWS CloudHSM SDKs memberi Anda kontrol penuh atas pengembangan aplikasi, bahasa aplikasi, threading, dan di mana aplikasi Anda ada secara fisik.

Migrasikan beban kerja kriptografi Anda ke cloud

Pelanggan yang memigrasikan infrastruktur kunci publik yang menggunakan Standar Kriptografi Kunci Publik #11 (PKCS #11), Ekstensi Kriptografi Java (JCE), API Kriptografi: Generasi Berikutnya (CNG), atau Penyedia Penyimpanan Kunci (KSP) dapat bermigrasi ke dengan sedikit perubahan pada aplikasi mereka. AWS CloudHSM

Untuk mempelajari lebih lanjut tentang apa yang dapat Anda lakukan AWS CloudHSM, lihat topik berikut. Ketika Anda siap untuk memulai AWS CloudHSM, lihat [Mulai menggunakan](#).

Note

Jika Anda menginginkan layanan terkelola untuk membuat dan mengendalikan kunci enkripsi Anda tetapi Anda tidak ingin atau perlu mengoperasikannya sendiri HSMs, pertimbangkan untuk menggunakannya [AWS Key Management Service](#).

Jika Anda mencari layanan elastis yang mengelola pembayaran HSMs dan kunci untuk aplikasi pemrosesan pembayaran di cloud, pertimbangkan untuk menggunakan [AWS Payment Cryptography](#).

Konten

- [AWS CloudHSM kasus penggunaan](#)
- [Bagaimana cara AWS CloudHSM kerja](#)
- [Harga untuk AWS CloudHSM](#)

AWS CloudHSM kasus penggunaan

AWS CloudHSM dapat digunakan untuk mencapai berbagai tujuan. Konten dalam topik ini memberikan gambaran umum tentang apa yang dapat Anda lakukan AWS CloudHSM.

Mencapai kepatuhan terhadap peraturan

Bisnis yang perlu selaras dengan standar keamanan perusahaan dapat digunakan AWS CloudHSM untuk mengelola kunci pribadi yang melindungi data yang sangat rahasia. Yang HSMs

AWS CloudHSM disediakan oleh FIPS 140-2 level 3 bersertifikat dan sesuai dengan PCI DSS. Selain AWS CloudHSM itu, sesuai dengan PIN PCI dan sesuai dengan PCI-3DS. Untuk informasi selengkapnya, lihat [Validasi kepatuhan untuk AWS CloudHSM](#).

Mengenkripsi dan mendekripsi data

Gunakan AWS CloudHSM untuk mengelola kunci pribadi yang melindungi data yang sangat rahasia, enkripsi dalam perjalanan, dan enkripsi saat istirahat. Selain itu, AWS CloudHSM menawarkan integrasi yang sesuai standar dengan beberapa kriptografi. SDKs

Menandatangani dan memverifikasi dokumen dengan kunci pribadi dan publik

Dalam kriptografi, menggunakan kunci pribadi untuk menandatangani dokumen memungkinkan penerima untuk menggunakan kunci publik untuk memverifikasi bahwa Anda (dan bukan orang lain) benar-benar mengirim dokumen. Gunakan AWS CloudHSM untuk membuat pasangan kunci publik dan pribadi asimetris yang dirancang khusus untuk tujuan ini.

Mengautentikasi pesan menggunakan HMACs dan CMACs

Dalam kriptografi, Kode Otentikasi Pesan Cipher (CMACs) dan Kode Otentikasi Pesan berbasis Hash (HMACs) digunakan untuk mengautentikasi dan memastikan integritas pesan yang dikirim melalui jaringan yang tidak aman. Dengan AWS CloudHSM, Anda dapat dengan aman membuat dan mengelola kunci simetris yang mendukung HMACs dan CMACs.

Memanfaatkan manfaat AWS CloudHSM dan AWS Key Management Service

Pelanggan dapat menggabungkan AWS CloudHSM dan [AWS KMS](#) menyimpan materi utama dalam lingkungan penyewa tunggal sambil juga mendapatkan manfaat manajemen kunci, penskalaan, dan integrasi cloud. AWS KMS Untuk detail tentang cara melakukannya, lihat [toko AWS CloudHSM utama](#) di Panduan AWS Key Management Service Pengembang.

SSL/TLS Pemrosesan offload untuk server web

Untuk mengirim data dengan aman melalui internet, server web menggunakan pasangan kunci publik-pribadi dan sertifikat kunci SSL/TLS publik untuk membuat sesi HTTPS. Proses ini melibatkan banyak perhitungan untuk server web, tetapi Anda dapat mengurangi beban komputasi sambil memberikan keamanan ekstra dengan membongkar beberapa ini ke cluster

Anda. AWS CloudHSM Untuk informasi tentang pengaturan SSL/TLS offload dengan AWS CloudHSM, lihat [SSL/TLS pembongkaran](#).

Aktifkan enkripsi data transparan (TDE)

Transparent Data Encryption (TDE) digunakan untuk mengenkripsi file database. Menggunakan TDE, perangkat lunak database mengenkripsi data sebelum menyimpannya di disk. Anda dapat mencapai keamanan yang lebih besar dengan menyimpan kunci enkripsi master TDE di HSMs dalam. AWS CloudHSM Untuk informasi tentang pengaturan Oracle TDE dengan AWS CloudHSM, lihat. [Enkripsi basis data Oracle](#)

Mengelola kunci pribadi dari otoritas sertifikat penerbit (CA)

Otoritas sertifikat (CA) adalah entitas tepercaya yang mengeluarkan sertifikat digital yang mengikat kunci publik ke identitas (seseorang atau organisasi). Untuk mengoperasikan CA, Anda harus menjaga kepercayaan dengan melindungi kunci pribadi yang menandatangani sertifikat yang dikeluarkan oleh CA Anda. Anda dapat menyimpan kunci pribadi tersebut di AWS CloudHSM cluster Anda dan kemudian menggunakannya HSMs untuk melakukan operasi penandatanganan kriptografi.

Hasilkan angka acak

Menghasilkan angka acak untuk membuat kunci enkripsi adalah inti dari keamanan online. AWS CloudHSM dapat digunakan untuk menghasilkan angka acak dengan aman di kontrol HSMs Anda dan hanya terlihat oleh Anda.

Bagaimana cara AWS CloudHSM kerja

Topik ini memberikan gambaran umum tentang konsep dasar dan arsitektur yang Anda gunakan untuk mengenkripsi data dengan aman dan melakukan operasi kriptografi. HSMs AWS CloudHSM beroperasi di Amazon Virtual Private Cloud (VPC) Anda sendiri. Sebelum Anda dapat menggunakan AWS CloudHSM, Anda terlebih dahulu membuat cluster, HSMs menambahkannya, membuat pengguna dan kunci, dan kemudian menggunakan Klien SDKs untuk mengintegrasikan Anda HSMs dengan aplikasi Anda. Setelah ini selesai, Anda menggunakan log SDK Klien AWS CloudTrail, log audit, dan Amazon CloudWatch untuk [memantau AWS CloudHSM](#).

Pelajari AWS CloudHSM konsep dasar dan bagaimana mereka bekerja sama untuk membantu melindungi data Anda.

Topik

- [AWS CloudHSM kluster](#)
- [Pengguna di AWS CloudHSM](#)
- [Kunci di AWS CloudHSM](#)
- [Klien SDKs untuk AWS CloudHSM](#)
- [AWS CloudHSM cadangan kluster](#)
- [Wilayah yang Didukung untuk AWS CloudHSM](#)

AWS CloudHSM kluster

Membuat HSMs kerja individu bersama dengan cara yang disinkronkan, berlebihan, dan sangat tersedia bisa jadi sulit, tetapi AWS CloudHSM melakukan pekerjaan berat untuk Anda dengan menyediakan modul keamanan perangkat keras () dalam kelompok. HSMs Cluster adalah kumpulan individu HSMs yang AWS CloudHSM tetap sinkron. Ketika Anda melakukan tugas atau operasi pada satu HSM dalam sebuah cluster, yang lain HSMs di cluster itu secara otomatis tetap up to date.

AWS CloudHSM menawarkan cluster dalam dua mode: FIPS dan non-FIPS. Dalam mode FIPS, hanya kunci dan algoritma yang divalidasi Federal Information Processing Standard (FIPS) yang dapat digunakan. Mode non-FIPS menawarkan semua kunci dan algoritme yang didukung oleh AWS CloudHSM, terlepas dari persetujuan FIPS. AWS CloudHSM juga menawarkan dua jenis HSMs: hsm1.medium dan hsm2m.medium. Untuk detail tentang perbedaan antara setiap tipe HSM dan mode cluster, lihat [AWS CloudHSM mode cluster](#). Tipe HSM hsm1.medium mencapai akhir dukungan sehingga cluster baru tidak dapat dibuat dengan tipe ini. Untuk informasi selengkapnya, lihat [Pemberitahuan penghentian](#) untuk detailnya.

Untuk memenuhi sasaran ketersediaan, daya tahan, dan skalabilitas, Anda menetapkan jumlah HSMs di kluster Anda di beberapa zona ketersediaan. Anda dapat membuat cluster yang memiliki 1 hingga 28 HSMs ([batas default](#) adalah 6 HSMs per AWS akun per [AWS Wilayah](#)). Anda dapat menempatkan HSMs di [Availability Zone](#) yang berbeda di suatu AWS wilayah. Menambahkan lebih banyak HSMs ke cluster memberikan kinerja yang lebih tinggi. Menyebarkan kluster di seluruh Availability Zone menyediakan redundansi dan ketersediaan tinggi.

Untuk informasi selengkapnya tentang kluster, lihat [Cluster di AWS CloudHSM](#).

Untuk membuat klaster, lihat [Mulai menggunakan](#).

Pengguna di AWS CloudHSM

Tidak seperti kebanyakan AWS layanan dan sumber daya, Anda tidak menggunakan pengguna AWS Identity and Access Management (IAM) atau kebijakan IAM untuk mengakses sumber daya dalam klaster Anda AWS CloudHSM. Sebagai gantinya, Anda menggunakan pengguna HSM langsung HSMs di AWS CloudHSM cluster Anda.

Pengguna HSM berbeda dari pengguna IAM. Pengguna IAM yang memiliki kredensial yang benar dapat membuat HSMs dengan berinteraksi dengan sumber daya melalui AWS API. Karena enkripsi E2E tidak terlihat oleh AWS, Anda harus menggunakan kredensial pengguna HSM untuk mengautentikasi operasi pada HSM karena kredensialnya berlangsung langsung di HSM. HSM mengautentikasi setiap pengguna HSM dengan sarana kredensial yang Anda tentukan dan kelola. Setiap pengguna HSM memiliki tipe yang menentukan operasi mana yang dapat dilakukan pengguna pada HSM. [Setiap HSM mengautentikasi setiap pengguna HSM melalui kredensial yang Anda tentukan menggunakan CloudHSM CLI](#).

Jika Anda menggunakan [seri versi SDK sebelumnya](#), maka Anda akan menggunakan [CloudHSM Management](#) Utility (CMU).

Kunci di AWS CloudHSM

AWS CloudHSM memungkinkan Anda membuat, menyimpan, dan mengelola kunci enkripsi dengan aman dalam penyewa tunggal HSMs yang ada di cluster Anda. AWS CloudHSM Kunci dapat simetris atau asimetris, dapat berupa kunci sesi (kunci singkat) untuk sesi tunggal, kunci token (kunci persisten) untuk penggunaan jangka panjang, dan dapat diekspor dari dan diimpor ke AWS CloudHSM Kunci juga dapat digunakan untuk menyelesaikan tugas dan fungsi kriptografi umum:

- Lakukan penandatanganan data kriptografi dan verifikasi tanda tangan dengan algoritma enkripsi simetris dan asimetris.
- Bekerja dengan fungsi hash untuk menghitung intisari pesan dan kode otentikasi pesan berbasis hash (). HMACs
- Bungkus dan lindungi kunci lainnya.
- Akses data acak yang aman secara kriptografis.

Tombol maksimum yang dapat dimiliki cluster tergantung pada jenis HSMs yang ada di cluster. Misalnya, hsm2m.medium menyimpan lebih banyak kunci daripada hsm1, medium. Untuk perbandingan, lihat [AWS CloudHSM kuota](#).

Selain itu, AWS CloudHSM ikuti beberapa prinsip dasar untuk penggunaan dan manajemen utama:

Banyak jenis kunci dan algoritma untuk dipilih

Untuk memungkinkan Anda menyesuaikan solusi Anda sendiri, AWS CloudHSM sediakan banyak jenis dan algoritme utama untuk dipilih dari algoritme yang mendukung berbagai ukuran kunci. Untuk informasi lebih lanjut, lihat atribut dan halaman mekanisme masing-masing [Operasi pembongkaran dengan Klien AWS CloudHSM SDKs](#).

Bagaimana Anda mengelola kunci

AWS CloudHSM kunci dikelola melalui SDKs dan alat baris perintah. Untuk informasi tentang cara menggunakan alat ini untuk mengelola kunci, lihat [Kunci di AWS CloudHSM](#) dan [Praktik terbaik untuk AWS CloudHSM](#).

Siapa yang memiliki kunci

Pada tahun AWS CloudHSM, pengguna krypto (CU) yang membuat kunci memilikinya. Pemilik dapat menggunakan key unshare perintah key share dan untuk berbagi dan membatalkan pembagian kunci dengan yang lain CUs. Untuk informasi selengkapnya, lihat [Bagikan dan batalkan berbagi kunci menggunakan CloudHSM CLI](#).

Akses dan penggunaan dapat dikontrol dengan enkripsi berbasis atribut

AWS CloudHSM memungkinkan Anda menggunakan enkripsi berbasis atribut, bentuk enkripsi yang memungkinkan Anda menggunakan atribut kunci untuk mengontrol siapa yang dapat mendekripsi data berdasarkan kebijakan.

Klien SDKs untuk AWS CloudHSM

Saat menggunakan AWS CloudHSM, Anda melakukan operasi kriptografi dengan [Kit Pengembangan Perangkat Lunak AWS CloudHSM Klien \(\) SDKs](#). AWS CloudHSM Klien SDKs meliputi:

- Standar Kriptografi Kunci Publik #11 (PKCS #11)

- Penyedia JCE
- OpenSSL Dynamic Engine
- Penyedia Penyimpanan Kunci (KSP) untuk Microsoft Windows

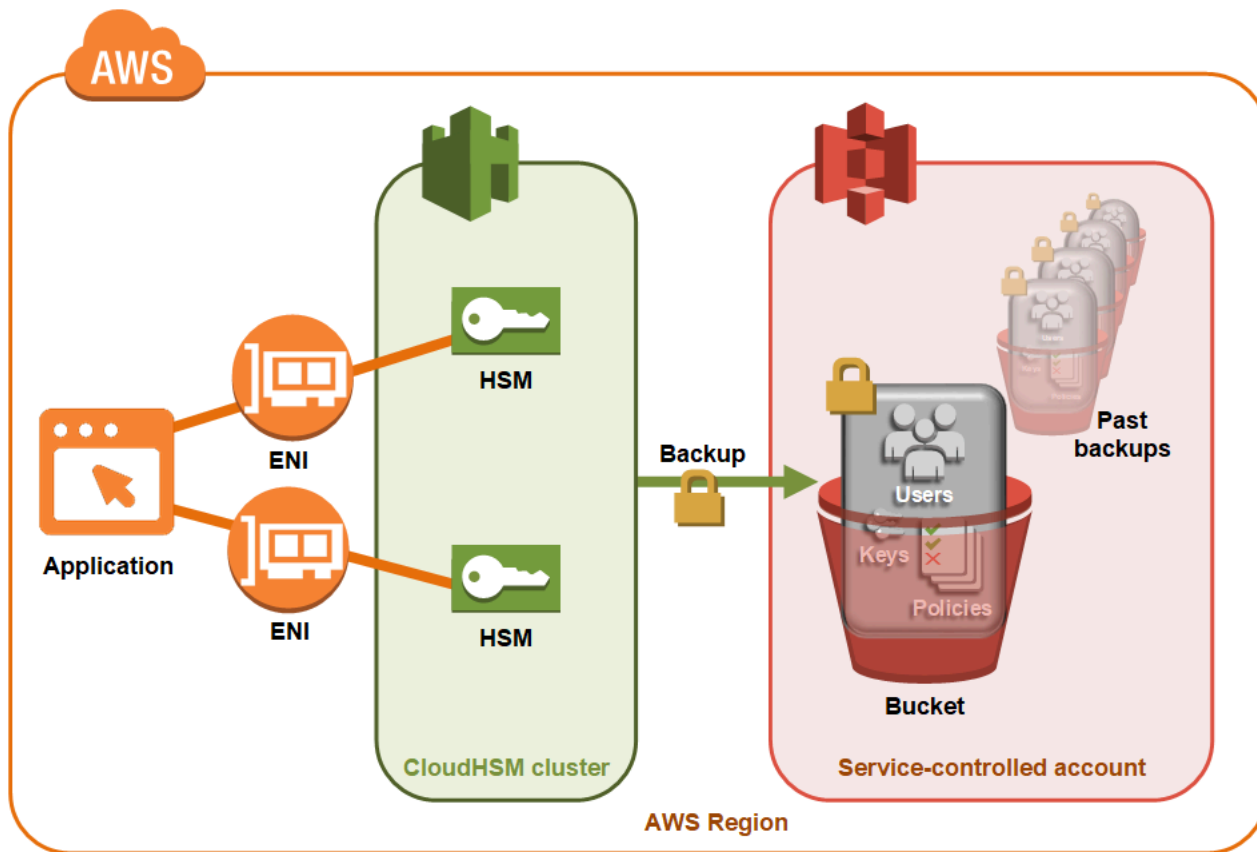
Anda dapat menggunakan salah satu atau semua SDK ini di AWS CloudHSM cluster Anda. Tulis kode aplikasi Anda untuk menggunakannya SDKs untuk melakukan operasi kriptografi di situs Anda HSMs. Untuk melihat platform dan tipe HSM apa yang mendukung setiap SDK, lihat [AWS CloudHSM Platform yang didukung SDK Klien 5](#)

Alat utilitas dan baris perintah diperlukan tidak hanya untuk digunakan SDKs tetapi juga untuk mengkonfigurasi kredensial, kebijakan, dan pengaturan aplikasi Anda. Untuk informasi lebih lanjut, lihat [AWS CloudHSM alat baris perintah](#).

Untuk informasi lebih lanjut tentang menginstal dan menggunakan SDK Klien atau keamanan sambungan klien, lihat [Klien SDKs](#) dan [End-to-end enkripsi](#).

AWS CloudHSM cadangan kluster

AWS CloudHSM membuat backup berkala dari pengguna, kunci, dan kebijakan di cluster. Cadangan aman, tahan lama, dan diperbarui pada jadwal yang dapat diprediksi. Ilustrasi berikut menunjukkan hubungan backup Anda dengan cluster.



Untuk informasi lebih lanjut tentang bekerja dengan pencadangan, lihat [Cadangan cluster](#).

Keamanan

Ketika AWS CloudHSM membuat cadangan dari HSM, HSM mengenkripsi semua datanya sebelum mengirimnya ke. AWS CloudHSM Data tidak pernah meninggalkan HSM dalam bentuk plaintext. Selain itu, backup tidak dapat didekripsi oleh AWS karena AWS tidak memiliki akses ke kunci yang digunakan untuk mendekripsi backup. Untuk informasi selengkapnya, lihat [Keamanan cadangan kluster](#)

Daya tahan

AWS CloudHSM menyimpan backup di bucket Amazon Simple Storage Service (Amazon S3) yang dikontrol layanan di wilayah yang sama dengan kluster Anda. Cadangan memiliki 99,999999999% tingkat daya tahan, sama seperti setiap objek yang disimpan di Amazon S3.

Wilayah yang Didukung untuk AWS CloudHSM

Untuk informasi tentang Wilayah yang didukung AWS CloudHSM, lihat [AWS CloudHSM Wilayah dan Titik Akhir](#) di Referensi Umum AWS, atau [Tabel Wilayah](#).

AWS CloudHSM mungkin tidak tersedia di semua Availability Zone di Wilayah tertentu. Namun, ini seharusnya tidak memengaruhi kinerja, karena AWS CloudHSM secara otomatis memuat saldo di semua HSMs dalam cluster.

Seperti kebanyakan AWS sumber daya, cluster dan HSMs sumber daya regional. Anda tidak dapat menggunakan kembali atau memperpanjang klaster di Wilayah. Anda harus melakukan semua langkah yang diperlukan yang tercantum dalam [Memulai dengan AWS CloudHSM](#) untuk membuat sebuah klaster di Wilayah baru.

Untuk tujuan pemulihan bencana, AWS CloudHSM Anda dapat menyalin cadangan AWS CloudHSM Cluster Anda dari satu wilayah ke wilayah lain. Lihat informasi yang lebih lengkap di [AWS CloudHSM cadangan klaster](#).

Harga untuk AWS CloudHSM

Dengan AWS CloudHSM, Anda membayar per jam tanpa komitmen jangka panjang atau pembayaran di muka. Untuk informasi lebih lanjut, lihat [AWS CloudHSM Harga](#) di AWS situs web.

Memulai dengan AWS CloudHSM

Topik berikut membantu Anda membuat, menginisialisasi, dan mengaktifkan klaster. AWS CloudHSM Setelah menyelesaikan prosedur ini, Anda akan siap mengelola pengguna, mengelola klaster, dan menggunakan pustaka perangkat lunak yang disertakan untuk melakukan operasi kriptografi. Untuk pengalaman terbaik, ikuti topik dalam urutan yang tercantum.

Konten

- [Buat grup administratif IAM untuk AWS CloudHSM](#)
- [Buat virtual private cloud \(VPC\) untuk AWS CloudHSM](#)
- [Buat cluster di AWS CloudHSM](#)
- [Tinjau grup keamanan untuk klaster Anda di AWS CloudHSM](#)
- [Luncurkan instans klien Amazon EC2 untuk berinteraksi AWS CloudHSM](#)
- [Konfigurasi grup keamanan instans Amazon EC2 Klien untuk AWS CloudHSM](#)
- [Buat HSM di AWS CloudHSM](#)
- [Verifikasi identitas dan keaslian HSM cluster Anda di AWS CloudHSM \(opsional\)](#)
- [Inisialisasi cluster di AWS CloudHSM](#)
- [Instal dan konfigurasi CloudHSM CLI](#)
- [Aktifkan cluster di AWS CloudHSM](#)
- [Siapkan TLS timbal balik antara klien dan AWS CloudHSM \(disarankan\)](#)
- [Buat dan gunakan kunci di AWS CloudHSM](#)

Buat grup administratif IAM untuk AWS CloudHSM

Langkah pertama untuk memulai AWS CloudHSM adalah mengatur izin IAM.

Sebagai [praktik terbaik](#), jangan gunakan Anda Pengguna root akun AWS untuk berinteraksi AWS, termasuk AWS CloudHSM. Sebagai gantinya, gunakan AWS Identity and Access Management (IAM) untuk membuat pengguna IAM, peran IAM, atau pengguna federasi. Ikuti langkah-langkah di bagian [Buat grup pengguna dan administrator IAM](#) untuk membuat grup administrator dan melampirkan AdministratorAccesskebijakan ke dalamnya. Kemudian, buat pengguna administrator baru dan tambahkan pengguna ke grup. Tambahkan pengguna tambahan ke grup sesuai kebutuhan. Setiap pengguna yang Anda tambahkan mewarisi AdministratorAccesskebijakan dari grup.

Praktik terbaik lainnya adalah membuat grup AWS CloudHSM administrator yang hanya memiliki izin yang diperlukan untuk dijalankan AWS CloudHSM. Tambahkan pengguna individu ke grup ini sesuai kebutuhan. Setiap pengguna mewarisi izin terbatas yang dilampirkan ke grup daripada akses AWS penuh. [Kebijakan yang dikelola pelanggan untuk AWS CloudHSM](#) Bagian berikut berisi kebijakan yang harus Anda lampirkan ke grup AWS CloudHSM administrator Anda.

AWS CloudHSM mendefinisikan [peran terkait layanan untuk akun](#) Anda. AWS Peran terkait layanan saat ini menentukan izin yang memungkinkan akun Anda mencatat peristiwa. AWS CloudHSM Peran dapat dibuat secara otomatis oleh AWS CloudHSM atau secara manual oleh Anda. Anda tidak dapat mengedit peran, tetapi Anda dapat menghapusnya. Untuk informasi lebih lanjut, lihat [Peran terkait layanan untuk AWS CloudHSM](#).

Buat grup pengguna dan administrator IAM

Mulai dengan membuat pengguna IAM bersama dengan grup administrator untuk pengguna tersebut.

Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka <https://portal.aws.amazon.com/billing/pendaftaran>.
2. Ikuti petunjuk online.

Bagian dari prosedur pendaftaran melibatkan menerima panggilan telepon atau pesan teks dan memasukkan kode verifikasi pada keypad telepon.

Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWS dibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan [tugas yang memerlukan akses pengguna root](#).

AWS mengirimkan email konfirmasi setelah proses pendaftaran selesai. Kapan saja, Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan masuk <https://aws.amazon.com/ke/> dan memilih Akun Saya.

Buat pengguna dengan akses administratif

Setelah Anda mendaftarkan Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan Anda Pengguna root akun AWS

1. Masuk ke [Konsol Manajemen AWS](#) sebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

Untuk bantuan masuk dengan menggunakan pengguna root, lihat [Masuk sebagai pengguna root](#) di AWS Sign-In Panduan Pengguna.

2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

Untuk petunjuk, lihat [Mengaktifkan perangkat MFA virtual untuk pengguna Akun AWS root \(konsol\) Anda](#) di Panduan Pengguna IAM.

Buat pengguna dengan akses administratif

1. Aktifkan Pusat Identitas IAM.

Untuk mendapatkan petunjuk, silakan lihat [Mengaktifkan AWS IAM Identity Center](#) di Panduan Pengguna AWS IAM Identity Center .

2. Di Pusat Identitas IAM, berikan akses administratif ke pengguna.

Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat [Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM](#) di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna dengan akses administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email saat Anda membuat pengguna Pusat Identitas IAM.

Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat [Masuk ke portal AWS akses](#) di Panduan AWS Sign-In Pengguna.

Tetapkan akses ke pengguna tambahan

1. Di Pusat Identitas IAM, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

Untuk petunjuk, lihat [Membuat set izin](#) di Panduan AWS IAM Identity Center Pengguna.

2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

Untuk petunjuk, lihat [Menambahkan grup](#) di Panduan AWS IAM Identity Center Pengguna.

Misalnya kebijakan AWS CloudHSM yang dapat Anda lampirkan ke grup pengguna IAM, lihat [Identitas dan manajemen akses untuk AWS CloudHSM](#).

Buat virtual private cloud (VPC) untuk AWS CloudHSM

Anda memerlukan virtual private cloud (VPC) untuk cluster Anda. AWS CloudHSM Jika Anda belum memilikinya, ikuti langkah-langkah dalam topik ini untuk membuat VPC.

Note

Mengikuti langkah-langkah ini akan menghasilkan penciptaan subnet publik dan swasta.

Untuk membuat VPC

1. Buka konsol Amazon VPC di <https://console.aws.amazon.com/vpc/>
2. Pada bilah navigasi, gunakan pemilih wilayah untuk memilih salah satu [AWS Wilayah yang saat AWS CloudHSM ini didukung](#).
3. Pilih tombol Buat VPC.
4. Agar Sumber Daya dapat dibuat, pilih VPC dan lainnya.
5. Untuk pembuatan otomatis tag Nama, ketikkan nama yang dapat diidentifikasi seperti **CloudHSM**
6. Untuk blok IPv6 CIDR, pilih blok IPv6 CIDR yang disediakan Amazon untuk menggunakan IPv6 konektivitas untuk Anda HSMs dan telah AWS mengalokasikan blok IPv6 CIDR untuk cluster Anda. Pengaturan ini mendukung Tipe Jaringan dual-stack. Pertahankan pengaturan default jika Anda tidak memerlukan IPv6 konektivitas.

7. Biarkan semua pilihan lain diatur ke defaultnya.
8. Pilih Buat VPC.
9. Setelah VPC dibuat, pilih Lihat VPC untuk melihat VPC yang baru saja Anda buat.

Buat cluster di AWS CloudHSM

Cluster adalah kumpulan modul keamanan perangkat keras individu (HSMs). AWS CloudHSM menyinkronkan HSMs di setiap cluster sehingga mereka berfungsi sebagai unit logis. AWS CloudHSM menawarkan dua jenis HSMs: hsm1.medium dan hsm2m.medium. Saat Anda membuat cluster, Anda memilih mana dari keduanya akan berada di cluster Anda. Untuk detail tentang perbedaan antara setiap tipe HSM dan mode cluster, lihat [AWS CloudHSM mode cluster](#).

Saat Anda membuat klaster, AWS CloudHSM buat grup keamanan untuk klaster atas nama Anda. Grup keamanan ini mengontrol akses jaringan ke HSMs dalam cluster. Ini memungkinkan koneksi masuk hanya dari instans Amazon Elastic Compute Cloud EC2 (Amazon) yang ada di grup keamanan. Secara default, grup keamanan tidak berisi instans apapun. Kemudian, Anda [meluncurkan instans klien](#) dan [mengatur konfigurasi grup keamanan klaster](#) untuk mengizinkan komunikasi dan koneksi dengan HSM.

Pertimbangan-pertimbangan

- Berikut ini adalah beberapa pertimbangan saat membuat cluster di AWS CloudHSM:
 - Saat Anda membuat klaster, AWS CloudHSM buat [peran terkait layanan](#) bernama AWSService RoleForCloud HSM. Jika AWS CloudHSM tidak dapat membuat peran atau peran belum ada, Anda mungkin tidak dapat membuat klaster. Untuk informasi selengkapnya, lihat [Menyelesaikan AWS CloudHSM kegagalan pembuatan cluster](#). Untuk informasi lebih lanjut tentang peran terkait layanan, lihat [Peran terkait layanan untuk AWS CloudHSM](#).
 - Jika Anda menggunakan [titik akhir AWS CloudHSM dual-stack](#) (yaitu, cloudhsmv2.
<region>.api.aws), pastikan bahwa kebijakan IAM Anda diperbarui untuk ditangani. IPv6 Untuk informasi selengkapnya, lihat bagian [Upgrade kebijakan IAM ke IPv6 bagian Keamanan](#).

Anda dapat membuat klaster dari [konsol AWS CloudHSM](#), [AWS Command Line Interface \(AWS CLI\)](#), atau API AWS CloudHSM .

Untuk detail tentang argumen cluster dan APIs, lihat [create-cluster](#) di AWS CLI Command Reference.

Console

Untuk membuat klaster (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Pada bilah navigasi, gunakan pemilih wilayah untuk memilih salah satu dari [Wilayah AWS tempat AWS CloudHSM saat ini didukung](#).
3. Pilih Buat klaster.
4. Di bagian Konfigurasi klaster, lakukan hal berikut:
 - a. Untuk VPC, pilih VPC yang Anda buat. [Buat virtual private cloud \(VPC\) untuk AWS CloudHSM](#)
 - b. Untuk Availability Zone (s), di samping setiap Availability Zone, pilih subnet pribadi yang Anda buat.

Note

Meskipun tidak AWS CloudHSM didukung di Availability Zone tertentu, performa tidak akan terpengaruh, karena AWS CloudHSM secara otomatis memuat saldo HSMs di semua klaster. Lihat [AWS CloudHSM Wilayah dan Titik Akhir](#) di bagian Referensi Umum AWS untuk melihat dukungan Availability Zone. AWS CloudHSM

- c. Untuk tipe HSM, pilih tipe HSM yang dapat dibuat di cluster Anda bersama dengan mode cluster yang diinginkan. Untuk melihat jenis HSM apa yang didukung di setiap wilayah, lihat [kalkulator AWS CloudHSM harga](#).

Important

Setelah cluster dibuat, mode cluster tidak dapat diubah. Untuk informasi tentang jenis dan mode yang tepat untuk kasus penggunaan Anda, lihat [AWS CloudHSM mode cluster](#).

- d. Untuk Jenis Jaringan, pilih protokol alamat IP untuk mengakses Anda. HSMs IPv4 membatasi komunikasi antara aplikasi Anda dan IPv4 hanya HSMs untuk. Ini adalah pilihan default. Dual-stack memungkinkan keduanya IPv4 dan IPv6 komunikasi. Untuk menggunakan dual-stack, tambahkan keduanya IPv4 dan IPv6 CIDRs ke konfigurasi VPC dan subnet Anda. Tipe Jaringan sulit diubah setelah pengaturan awal. Untuk

memodifikasinya, buat cadangan cluster yang ada dan pulihkan cluster baru dengan Jenis Jaringan yang diinginkan. Untuk informasi selengkapnya, lihat [Membuat AWS CloudHSM cluster dari backup](#)

- e. Untuk sumber Cluster, tentukan apakah Anda ingin membuat klaster baru atau memulihkannya dari cadangan yang ada.
 - Cadangan cluster dalam mode non-FIPS hanya dapat digunakan untuk mengembalikan cluster yang berada dalam mode non-FIPS.
 - Cadangan cluster dalam mode FIPS hanya dapat digunakan untuk mengembalikan cluster yang berada dalam mode FIPS.
5. Pilih Berikutnya.
6. Tentukan berapa lama layanan harus mempertahankan cadangan.
 - Terima periode retensi default 90 hari atau ketik nilai baru antara 7 dan 379 hari. Layanan akan secara otomatis menghapus cadangan di klaster ini yang lebih tua dari nilai yang Anda tentukan di sini. Anda dapat mengubah ini nanti. Untuk informasi lebih lanjut, lihat [Konfigurasi retensi cadangan](#).
7. Pilih Selanjutnya.
8. (Opsional) Ketik kunci tanda dan nilai tanda opsional. Untuk menambahkan lebih dari satu tanda ke klaster, pilih Tambahkan tanda.
9. Pilih Tinjau.
10. Tinjau konfigurasi klaster Anda, dan kemudian pilih Buat klaster.

Jika upaya Anda untuk membuat klaster gagal, itu mungkin terkait dengan masalah dengan peran AWS CloudHSM terkait layanan. Untuk bantuan menyelesaikan kegagalan, lihat [Menyelesaikan AWS CloudHSM kegagalan pembuatan cluster](#).

AWS CLI

Untuk membuat klaster ([AWS CLI](#))

- Pada jendela perintah, jalankan perintah [create-cluster](#). Tentukan jenis instans HSM, periode retensi cadangan, dan subnet subnet IDs tempat Anda berencana untuk membuat HSMs. Gunakan subnet IDs dari subnet pribadi yang Anda buat. Tentukan hanya satu subnet per Availability Zone.

```
$ aws cloudhsmv2 create-cluster --hsm-type hsm2m.medium \
```

```

--backup-retention-policy Type=DAYS,Value=<number of days> \
--subnet-ids <subnet ID> \
--mode <FIPS> \
--network-type <IPV4>

{
  "Cluster": {
    "BackupPolicy": "DEFAULT",
    "BackupRetentionPolicy": {
      "Type": "DAYS",
      "Value": 90
    },
    "VpcId": "vpc-50ae0636",
    "SubnetMapping": {
      "us-west-2b": "subnet-49a1bc00",
      "us-west-2c": "subnet-6f950334",
      "us-west-2a": "subnet-fd54af9b"
    },
    "SecurityGroup": "sg-6cb2c216",
    "HsmType": "hsm2m.medium",
    "NetworkType": "IPV4",
    "Certificates": {},
    "State": "CREATE_IN_PROGRESS",
    "Hsms": [],
    "ClusterId": "cluster-igklspoyj5v",
    "ClusterMode": "FIPS",
    "CreateTimestamp": 1502423370.069
  }
}

```

 Note

ClusterMode adalah parameter yang diperlukan untuk semua jenis hsm kecuali hsm1.medium. --mode:

```

$ aws cloudhsmv2 create-cluster --hsm-type hsm2m.medium \
  --backup-retention-policy Type=DAYS,Value=<number of days> \
  --subnet-ids <subnet ID> \
  --mode NON_FIPS

```

Jika upaya Anda untuk membuat klaster gagal, itu mungkin terkait dengan masalah dengan peran AWS CloudHSM terkait layanan. Untuk bantuan menyelesaikan kegagalan, lihat [Menyelesaikan AWS CloudHSM kegagalan pembuatan cluster](#).

AWS CloudHSM API

Untuk membuat cluster (AWS CloudHSM API)

- Kirim permintaan [CreateCluster](#). Tentukan jenis instans HSM, kebijakan penyimpanan cadangan, dan subnet subnet IDs tempat Anda berencana untuk membuat. HSMs Gunakan subnet IDs dari subnet pribadi yang Anda buat. Tentukan hanya satu subnet per Availability Zone.

Jika upaya Anda untuk membuat klaster gagal, itu mungkin terkait dengan masalah dengan peran AWS CloudHSM terkait layanan. Untuk bantuan menyelesaikan kegagalan, lihat [Menyelesaikan AWS CloudHSM kegagalan pembuatan cluster](#).

Tinjau grup keamanan untuk klaster Anda di AWS CloudHSM

Saat Anda membuat klaster atau menambahkan HSM ke klaster, AWS CloudHSM buat grup keamanan dengan nama `cloudhsm-cluster-<clusterID>-sg` jika belum ada. Grup keamanan ini berisi aturan TCP yang telah dikonfigurasi yang memungkinkan komunikasi masuk dan keluar dalam grup keamanan klaster melalui port 2223-2225. SG ini memungkinkan instans EC2 Anda menggunakan VPC Anda untuk diajak bicara di cluster Anda. HSMs

Warning

- Jangan menghapus atau mengubah aturan TCP yang telah dikonfigurasi, yang diisikan dalam grup keamanan klaster. Aturan ini dapat mencegah masalah konektivitas dan akses tidak sah ke Anda HSMs.
- Grup keamanan klaster mencegah akses tidak sah ke Anda HSMs. Siapa pun yang dapat mengakses instans di grup keamanan dapat mengakses instans Anda HSMs. Sebagian besar operasi mewajibkan pengguna untuk login ke HSM. Namun, dimungkinkan untuk memusatkan perhatian HSMs tanpa otentikasi, yang menghancurkan materi utama, sertifikat, dan data lainnya. Jika hal ini terjadi, data yang dibuat atau dimodifikasi setelah cadangan terbaru akan hilang dan tidak dapat dipulihkan. Untuk mencegah akses yang

tidak sah, pastikan bahwa hanya administrator tepercaya yang dapat memodifikasi atau mengakses instans dalam grup keamanan default.

- Cluster hsm2m.medium memperkenalkan fitur mTLS untuk membatasi pengguna yang tidak sah terhubung ke cluster. Pengguna yang tidak sah akan memerlukan kredensial mTLS yang valid agar berhasil terhubung ke cluster sebelum mencoba zeroisasi.

Pada langkah berikutnya, Anda dapat [meluncurkan instans Amazon EC2](#) dan menghubungkannya HSMs dengan [melampirkan grup keamanan klaster ke](#) dalamnya.

Luncurkan instans klien Amazon EC2 untuk berinteraksi AWS CloudHSM

Untuk berinteraksi dengan dan mengelola instans AWS CloudHSM cluster dan HSM Anda, Anda harus dapat berkomunikasi dengan antarmuka jaringan elastis HSM Anda. Cara termudah untuk melakukannya adalah dengan menggunakan instans EC2 dalam VPC yang sama sebagai klaster Anda. Anda juga dapat menggunakan sumber daya AWS untuk menyambung ke klaster Anda:

- [Pengintipan VPC Amazon](#)
- [Direct Connect](#)
- [Koneksi VPN](#)

Note

Panduan ini memberikan contoh sederhana tentang cara menghubungkan instans EC2 ke AWS CloudHSM cluster Anda. Untuk praktik terbaik seputar konfigurasi jaringan aman, lihat [Akses aman ke klaster Anda](#)

AWS CloudHSM Dokumentasi biasanya mengasumsikan bahwa Anda menggunakan instans EC2 di VPC dan Availability Zone (AZ) yang sama di mana Anda membuat cluster Anda.

Untuk membuat instans EC2

1. Buka Dasbor EC2 di <https://console.aws.amazon.com/ec2/>.
2. Pilih Launch instance. Dari menu tarik-turun, pilih Launch instance.

3. Di bidang Nama, masukkan nama untuk instans EC2 Anda.
4. Di bagian Aplikasi dan Gambar OS (Amazon Machine Image), pilih Amazon Machine Image (AMI) yang sesuai dengan platform yang didukung CloudHSM. Untuk informasi selengkapnya, lihat [AWS CloudHSM Platform yang didukung SDK Klien 5](#).
5. Di bagian Jenis Instance, pilih jenis instance.
6. Di bagian Key pair, gunakan key pair yang ada atau pilih Create new key pair dan selesaikan langkah-langkah berikut:
 - a. Untuk nama Key pair, masukkan nama untuk key pair.
 - b. Untuk tipe Key pair, pilih tipe key pair.
 - c. Untuk format file kunci pribadi, pilih format file kunci pribadi.
 - d. Pilih Create key pair.
 - e. Unduh dan simpan file kunci pribadi.

 Important

Ini adalah satu-satunya kesempatan Anda untuk menyimpan file kunci pribadi. Unduh dan simpan file di tempat yang aman. Anda harus menyediakan nama pasangan kunci saat meluncurkan sebuah instans. Selain itu, Anda harus memberikan kunci pribadi yang sesuai setiap kali Anda terhubung ke instance dan memilih key pair yang Anda buat saat mengatur.

7. Di Pengaturan jaringan, pilih Edit.
8. Untuk VPC, pilih VPC yang sebelumnya Anda buat untuk klaster Anda.
9. Untuk Subnet, pilih subnet publik yang telah Anda buat untuk VPC.
10. Untuk Tetapkan Otomatis IP Publik, pilih Aktifkan.
11. Untuk Auto-assign IPv6 IP, pilih Aktifkan untuk menggunakan konektivitas IPv6 dengan cluster Anda dan Dual-stack. NetworkType Jika Anda mengaktifkan opsi ini, perbarui aturan grup keamanan instans Amazon EC2, tabel rute VPC dan subnet, serta jaringan ACLs untuk mengizinkan lalu lintas IPv6 keluar dari instans ke. HSMs
12. Pilih Pilih grup keamanan yang ada.
13. Di grup keamanan umum, pilih grup keamanan default dari menu tarik-turun.
14. Di Configure Storage, gunakan menu drop-down untuk memilih konfigurasi penyimpanan.
15. Di jendela Ringkasan, pilih Launch instance.

 Note

Menyelesaikan langkah ini akan memulai proses untuk membuat instans EC2 Anda.

Untuk informasi selengkapnya tentang membuat klien Amazon EC2 Linux, lihat [Memulai dengan Instans Amazon EC2 Linux](#). Untuk informasi tentang menghubungkan dengan klien yang sedang berjalan, lihat topik berikut ini:

- [Menghubungkan ke Instance Linux Anda Menggunakan SSH](#)
- [Menghubungkan ke Instance Linux Anda dari Windows Menggunakan PutTY](#)

Panduan pengguna Amazon EC2 berisi petunjuk detail untuk menyiapkan dan menggunakan instans Amazon EC2 Anda. Daftar berikut ini memberikan gambaran tentang dokumentasi yang tersedia untuk klien Amazon EC2 Linux dan Windows:

- Untuk membuat klien Amazon EC2 Linux, lihat [Memulai dengan Instans Amazon EC2 Linux](#).

Untuk informasi tentang menghubungkan dengan klien yang sedang berjalan, lihat topik berikut ini:

- [Menghubungkan ke Instans Linux Anda Menggunakan SSH](#)
- [Menghubungkan ke Instance Linux Anda dari Windows Menggunakan PutTY](#)
- Untuk membuat klien Amazon EC2 Windows, lihat [Memulai dengan instans Amazon EC2 Windows](#). Untuk informasi selengkapnya tentang menghubungkan ke klien Windows Anda, lihat [Hubungkan ke Instans Windows Anda](#).

 Note

Instans EC2 Anda dapat menjalankan semua AWS CLI perintah yang terkandung dalam panduan ini. Jika AWS CLI tidak terpasang, Anda bisa mengunduhnya dari [AWS Command Line Interface](#). Jika Anda menggunakan Windows, Anda dapat mengunduh dan menjalankan penginstal Windows 64-bit atau 32-bit. Jika Anda menggunakan Linux atau macOS, Anda dapat menginstal CLI menggunakan pip.

Konfigurasi grup keamanan instans Amazon EC2 Klien untuk AWS CloudHSM

Saat meluncurkan instans Amazon EC2 untuk kluster AWS CloudHSM, Anda mengaitkannya dengan grup keamanan Amazon VPC default. Topik ini menjelaskan cara mengaitkan grup keamanan kluster dengan instans EC2. Asosiasi ini memungkinkan AWS CloudHSM klien yang berjalan pada instans EC2 Anda untuk berkomunikasi dengan HSM Anda. Untuk menghubungkan instans EC2 ke AWS CloudHSM cluster, Anda harus mengonfigurasi grup keamanan default VPC dengan benar dan mengaitkan grup keamanan kluster dengan instans tersebut.

Gunakan langkah-langkah berikut untuk menyelesaikan perubahan konfigurasi.

Topik

- [Langkah 1. Mengubah grup keamanan default](#)
- [Langkah 2. Hubungkan instans Amazon EC2 ke cluster AWS CloudHSM](#)

Langkah 1. Mengubah grup keamanan default

Anda perlu memodifikasi grup keamanan default untuk mengizinkan koneksi SSH atau RDP, sehingga Anda dapat mengunduh dan menginstal perangkat lunak klien, dan berinteraksi dengan HSM Anda.

Untuk memodifikasi grup keamanan default

1. Buka Dasbor EC2 di <https://console.aws.amazon.com/ec2/>.
2. Pilih Instans (berjalan) dan kemudian pilih kotak centang di sebelah instans EC2 yang ingin Anda instal klien. AWS CloudHSM
3. Di bawah tab Keamanan, pilih grup keamanan bernama Default.
4. Di bagian atas halaman, pilih Tindakan, dan kemudian Edit Aturan Masuk.
5. Pilih Tambahkan Aturan.
6. Untuk Jenis, lakukan salah satu hal berikut:
 - Untuk instans Amazon EC2 Windows Server, pilih RDP. Port secara otomatis 3389 terisi.
 - Untuk instans Amazon EC2 Linux, pilih SSH. Rentang port 22 secara otomatis diisi.
7. Untuk salah satu opsi, setel Sumber ke IP Saya agar Anda dapat berkomunikasi dengan instans Amazon EC2 Anda.

 Important

Jangan tentukan 0.0.0.0/0 sebagai rentang CIDR untuk menghindari mengizinkan siapa pun mengakses instance Anda.

8. Pilih Simpan.

Langkah 2. Hubungkan instans Amazon EC2 ke cluster AWS CloudHSM

Anda harus melampirkan grup keamanan cluster ke instans EC2 sehingga instans EC2 dapat berkomunikasi dengan HSMs di cluster Anda. Grup keamanan klaster berisi aturan yang telah dikonfigurasi yang memungkinkan komunikasi masuk melalui port 2223-2225.

Untuk menghubungkan instans EC2 ke cluster AWS CloudHSM

1. Buka Dasbor EC2 di <https://console.aws.amazon.com/ec2/>.
2. Pilih Instans (berjalan) dan kemudian pilih kotak centang untuk instans EC2 di mana Anda ingin menginstal klien. AWS CloudHSM
3. Di bagian atas halaman, pilih Tindakan, Keamanan, dan kemudian Ubah Grup Keamanan.
4. Pilih grup keamanan dengan nama grup yang cocok dengan ID klaster Anda, seperti `cloudhsm-cluster-<clusterID>-sg`.
5. Pilih Tambahkan Grup Keamanan.
6. Pilih Simpan.

 Note

Anda dapat menetapkan maksimum lima grup keamanan untuk instans Amazon EC2. Jika Anda telah mencapai batas maksimum, Anda harus mengubah grup keamanan default instans Amazon EC2 dan grup keamanan klaster:

Di grup keamanan default, lakukan hal berikut:

- Tambahkan aturan masuk untuk mengizinkan lalu lintas menggunakan protokol TCP melalui port 2223-2225 dari grup keamanan klaster.

Di grup keamanan klaster, lakukan hal berikut:

- Tambahkan aturan masuk untuk mengizinkan lalu lintas menggunakan protokol TCP melalui port 2223-2225 dari grup keamanan default.

Buat HSM di AWS CloudHSM

Setelah Anda membuat cluster di AWS CloudHSM, Anda dapat membuat modul keamanan perangkat keras (HSM). Namun, sebelum Anda dapat membuat HSM di klaster Anda, klaster harus dalam keadaan tidak diinisialisasi. Untuk menentukan status klaster, lihat [halaman cluster di AWS CloudHSM konsol](#), gunakan perintah AWS CLI untuk menjalankan [describe-clusters](#) perintah, atau mengirim [DescribeClusters](#) permintaan di AWS CloudHSM API. Anda dapat membuat HSM dari [konsol AWS CloudHSM](#), [AWS CLI](#), atau API AWS CloudHSM .

Important

Hanya buat satu HSM saat cluster Anda dalam keadaan tidak diinisialisasi.

Console

Untuk membuat HSM (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Pilih tombol radio di sebelah ID cluster yang ingin Anda buat HSM.
3. Pilih Tindakan. Dari menu tarik-turun, pilih Inisialisasi.
4. Pilih Availability Zone (AZ) untuk HSM yang Anda buat.
5. Pilih Buat.

Setelah Anda membuat klaster dan HSM, Anda dapat secara opsional [memverifikasi identitas HSM](#), atau langsung lanjutkan ke [Inisialisasi cluster](#).

AWS CLI

Untuk membuat HSM ([AWS CLI](#))

- Pada jendela perintah, jalankan perintah [create-hsm](#). Tentukan ID klaster dari klaster yang Anda buat sebelumnya dan Availability Zone untuk HSM. Tentukan Availability Zone dalam bentuk us-west-2a, us-west-2b, dll.

```
$ aws cloudhsmv2 create-hsm --cluster-id <cluster ID> --availability-  
zone <Availability Zone>  
  
{  
  "Hsm": {  
    "HsmId": "hsm-ted36yp5b2x",  
    "EniIp": "10.0.1.12",  
    "EniIpV6": "2600:113f:404:be09:310e:ed34:3412:f733",  
    "AvailabilityZone": "us-west-2a",  
    "ClusterId": "cluster-igklspoyj5v",  
    "EniId": "eni-5d7ade72",  
    "SubnetId": "subnet-fd54af9b",  
    "State": "CREATE_IN_PROGRESS"  
  }  
}
```

Setelah Anda membuat klaster dan HSM, Anda dapat secara opsional [memverifikasi identitas HSM](#), atau langsung lanjutkan ke [Inisialisasi cluster](#).

AWS CloudHSM API

Untuk membuat HSM (AWS CloudHSM API)

- Kirim permintaan [CreateHsm](#). Tentukan ID klaster dari klaster yang Anda buat sebelumnya dan Availability Zone untuk HSM.

Setelah Anda membuat klaster dan HSM, Anda dapat secara opsional [memverifikasi identitas HSM](#), atau langsung lanjutkan ke [Inisialisasi cluster](#).

Verifikasi identitas dan keaslian HSM cluster Anda di AWS CloudHSM (opsional)

Untuk menginisialisasi klaster AWS CloudHSM, Anda menandatangani permintaan penandatanganan sertifikat (CSR) yang dihasilkan oleh modul keamanan perangkat keras (HSM) pertama klaster. Sebelum Anda melakukan ini, Anda mungkin perlu memverifikasi identitas dan keaslian HSM.

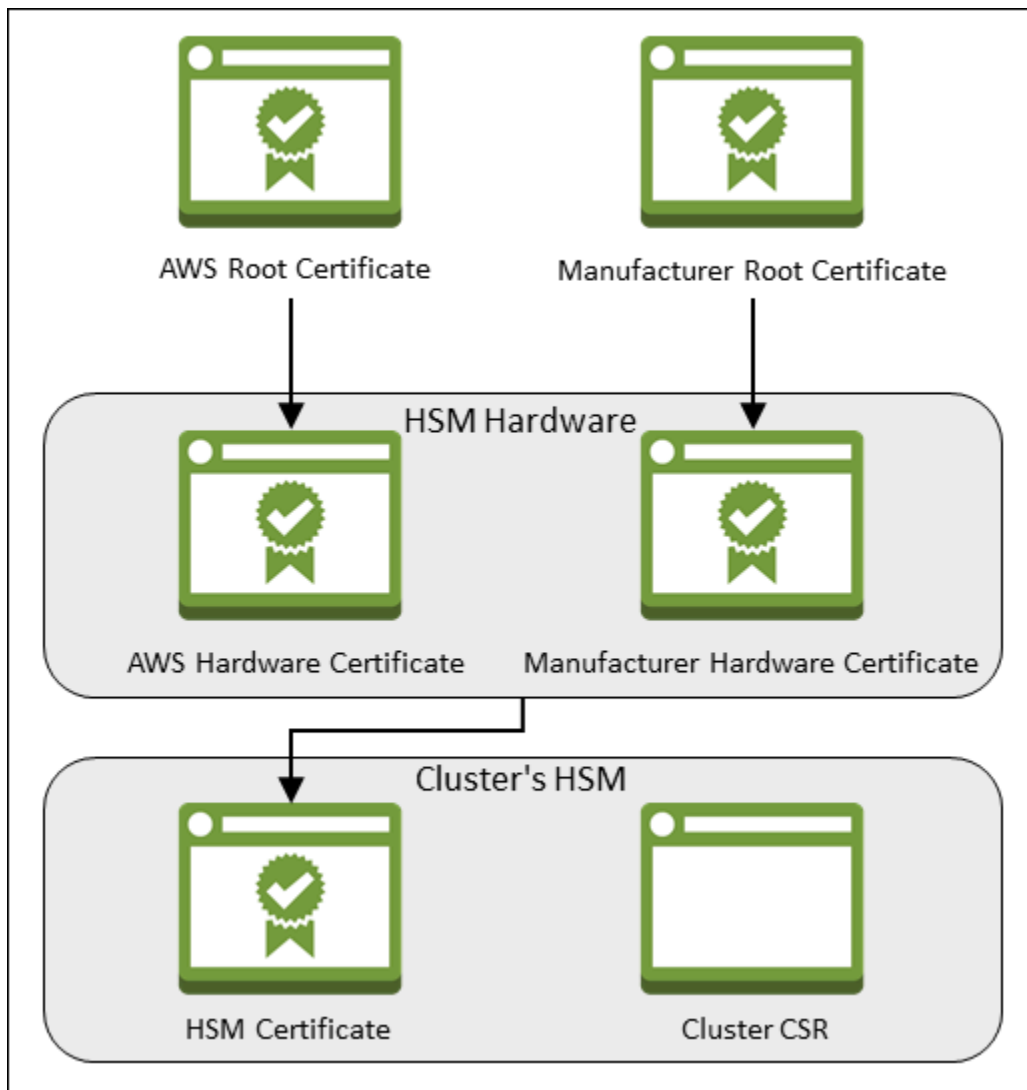
Note

Proses ini opsional. Namun, ini bekerja hanya sampai sebuah klaster diinisialisasi. Setelah cluster diinisialisasi, Anda tidak dapat menggunakan proses ini untuk mendapatkan sertifikat atau memverifikasi HSMs

Untuk memverifikasi identitas HSM klaster Anda, selesaikan langkah-langkah berikut:

1. [Dapatkan sertifikat dan CSR](#) — Pada langkah ini, Anda mendapatkan tiga sertifikat dan CSR dari HSM. Anda juga mendapatkan dua sertifikat root, satu dari AWS CloudHSM dan satu dari produsen perangkat keras HSM.
2. [Verifikasi rantai sertifikat](#) — Pada langkah ini, Anda membuat dua rantai sertifikat, satu ke sertifikat AWS CloudHSM root dan satu ke sertifikat root pabrikan. Kemudian Anda memverifikasi sertifikat HSM dengan rantai sertifikat ini untuk menentukan itu AWS CloudHSM dan produsen perangkat keras membuktikan identitas dan keaslian HSM.
3. [Bandingkan kunci publik](#) — Pada langkah ini, Anda mengekstraksi dan membandingkan kunci publik dalam sertifikat HSM dan klaster CSR, untuk memastikan bahwa mereka adalah sama. Hal ini akan memberi Anda keyakinan bahwa CSR dihasilkan oleh HSM yang autentik, tepercaya .

Diagram berikut menunjukkan CSR, sertifikat, dan hubungan mereka satu sama lain. Daftar berikutnya mendefinisikan setiap sertifikat.



AWS Sertifikat Root

Ini AWS CloudHSM adalah sertifikat root.

Sertifikat Root Produsen

Ini adalah sertifikat root pabrik perangkat keras.

AWS Sertifikat Perangkat Keras

AWS CloudHSM membuat sertifikat ini ketika perangkat keras HSM ditambahkan ke armada. Sertifikat ini menegaskan bahwa AWS CloudHSM memiliki perangkat keras.

Sertifikat Perangkat Keras Produsen

Produsen perangkat keras HSM membuat sertifikat ini ketika memproduksi perangkat keras HSM. Sertifikat ini menegaskan bahwa produsen membuat perangkat keras.

Sertifikat HSM

Sertifikat HSM dihasilkan oleh perangkat keras divalidasi FIPS ketika Anda membuat HSM pertama di klaster. Sertifikat ini menegaskan bahwa perangkat keras HSM membuat HSM.

CSR Klaster

HSM pertama membuat CSR klaster. Saat Anda [tandatangani CSR klaster](#), Anda mengklaim klaster. Kemudian, Anda dapat menggunakan CSR yang ditandatangani untuk [menginisialisasi klaster](#).

Langkah 1. Dapatkan sertifikat dari HSM

Untuk memverifikasi identitas dan keaslian HSM Anda, mulailah dengan mendapatkan CSR dan lima sertifikat. Anda mendapatkan tiga sertifikat dari HSM, yang dapat Anda lakukan dengan [AWS CloudHSM konsol](#), [AWS Command Line Interface \(AWS CLI\)](#), atau AWS CloudHSM API.

Console

Untuk mendapatkan CSR dan sertifikat HSM (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Pilih tombol radio di sebelah ID cluster dengan HSM yang ingin Anda verifikasi.
3. Pilih Tindakan. Dari menu tarik-turun, pilih Inisialisasi.
4. Jika Anda tidak menyelesaikan [langkah sebelumnya](#) untuk membuat HSM, pilih Availability Zone (AZ) untuk HSM yang Anda buat. Kemudian pilih Buat.
5. Saat sertifikat dan CSR siap, Anda akan melihat tautan untuk mengunduhnya.

Certificate signing request

To initialize the cluster, you must download a certificate signing request (CSR) and then [sign it](#) .

 **Cluster CSR**

Cluster verification certificate

Optionally, you may wish to download the HSM certificate below which generated this Cluster CSR and [verify its authenticity](#) .

 **HSM certificate**

6. Pilih setiap tautan untuk mengunduh dan menyimpan CSR dan sertifikat. Untuk menyederhanakan langkah-langkah berikutnya, simpan semua file ke direktori yang sama dan menggunakan nama file default.

AWS CLI

Untuk mendapatkan CSR dan sertifikat HSM ([AWS CLI](#))

- Pada prompt perintah, jalankan perintah [describe-clusters](#) empat kali, mengekstraksi CSR dan sertifikat yang berbeda setiap kalinya dan menyimpannya ke file.

- a. Keluarkan perintah berikut untuk mengekstraksi CSR klaster. Ganti *<cluster ID>* dengan ID cluster yang Anda buat sebelumnya.

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                     --output text \
                                     --query
'Clusters[].Certificates.ClusterCsr' \
                                     > <cluster ID>_ClusterCsr.csr
```

- b. Keluarkan perintah berikut untuk mengekstraksi sertifikat HSM. Ganti *<cluster ID>* dengan ID cluster yang Anda buat sebelumnya.

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                     --output text \
                                     --query
'Clusters[].Certificates.HsmCertificate' \
                                     > <cluster ID>_HsmCertificate.crt
```

- c. Keluarkan perintah berikut untuk mengekstrak sertifikat AWS perangkat keras. Ganti *<cluster ID>* dengan ID cluster yang Anda buat sebelumnya.

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                     --output text \
                                     --query
'Clusters[].Certificates.AwsHardwareCertificate' \
                                     > <cluster ID>_AwsHardwareCertificate.crt
```

- d. Keluarkan perintah berikut untuk mengekstrak sertifikat perangkat keras produsen. Ganti *<cluster ID>* dengan ID cluster yang Anda buat sebelumnya.

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                     --output text \
                                     --query
'Clusters[].Certificates.ManufacturerHardwareCertificate' \
                                     > <cluster
ID>_ManufacturerHardwareCertificate.crt
```

AWS CloudHSM API

Untuk mendapatkan sertifikat CSR dan HSM (API)AWS CloudHSM

- Kirim permintaan [DescribeClusters](#), lalu ekstrak dan simpan CSR dan sertifikat dari respons.

Langkah 2. Dapatkan sertifikat root

Ikuti langkah-langkah ini untuk mendapatkan sertifikat root untuk AWS CloudHSM dan pabrikan. Simpan berkas sertifikat root ke direktori yang berisi file CSR dan sertifikat HSM.

Untuk mendapatkan sertifikat root AWS CloudHSM dan produsen

1. Unduh sertifikat AWS CloudHSM root: [AWS_CloudHSM_Root-G1.zip](#)
2. Unduh sertifikat root pabrikan yang tepat untuk jenis HSM Anda:
 - [hsm1.medium sertifikat root produsen: liquid_security_certificate.zip](#)
 - [hsm2m.sertifikat root produsen medium: liquid_security_certificate.zip](#)

Note

Untuk mengunduh setiap sertifikat dari halaman arahnya, gunakan tautan berikut:

- [Halaman arahan untuk sertifikat root pabrikan hsm1.medium](#)
- [Halaman arahan untuk sertifikat root pabrikan hsm2m.medium](#)

Anda mungkin harus mengeklik kanan Unduh Sertifikat dan kemudian pilih Simpan Tautan Sebagai... untuk menyimpan file sertifikat.

3. Setelah Anda mengunduh file, ekstrak (unzip) konten.

Langkah 3. Verifikasi rantai sertifikat

Pada langkah ini, Anda membangun dua rantai sertifikat, satu ke sertifikat AWS CloudHSM root dan satu ke sertifikat root pabrikan. Kemudian, gunakan OpenSSL untuk memverifikasi sertifikat HSM dengan setiap rantai sertifikat.

Untuk membuat rantai sertifikat, buka shell Linux. Anda membutuhkan OpenSSL, yang tersedia di sebagian besar shell Linux, dan Anda memerlukan [sertifikat root](#) dan [file sertifikat HSM](#) yang Anda unduh. Namun, Anda tidak perlu AWS CLI untuk langkah ini, dan shell tidak perlu dikaitkan dengan AWS akun Anda.

Untuk memverifikasi sertifikat HSM dengan sertifikat AWS CloudHSM root

1. Navigasi ke direktori tempat Anda menyimpan [Sertifikat root](#) dan [file sertifikat HSM](#) yang Anda unduh. Perintah berikut menganggap bahwa semua sertifikat dalam direktori saat ini dan menggunakan nama file default.

Gunakan perintah berikut untuk membuat rantai sertifikat yang mencakup sertifikat AWS perangkat keras dan sertifikat AWS CloudHSM root, dalam urutan itu. Ganti *<cluster ID>* dengan ID cluster yang Anda buat sebelumnya.

```
$ cat <cluster ID>_AwsHardwareCertificate.crt \  
    AWS_CloudHSM_Root-G1.crt \  
    > <cluster ID>_AWS_chain.crt
```

2. Gunakan perintah OpenSSL berikut ini untuk memverifikasi sertifikat HSM dengan rantai sertifikat AWS . Ganti *<cluster ID>* dengan ID cluster yang Anda buat sebelumnya.

```
$ openssl verify -CAfile <cluster ID>_AWS_chain.crt <cluster ID>_HsmCertificate.crt  
<cluster ID>_HsmCertificate.crt: OK
```

Untuk memverifikasi sertifikat HSM dengan sertifikat root produsen

1. Gunakan perintah berikut untuk membuat rantai sertifikat yang mencakup sertifikat perangkat keras produsen dan sertifikat root produsen, dalam urutan itu. Ganti *<cluster ID>* dengan ID cluster yang Anda buat sebelumnya.

```
$ cat <cluster ID>_ManufacturerHardwareCertificate.crt \  
    liquid_security_certificate.crt \  
    > <cluster ID>_manufacturer_chain.crt
```

2. Gunakan perintah OpenSSL berikut ini untuk memverifikasi sertifikat HSM dengan rantai sertifikat . Ganti *<cluster ID>* dengan ID cluster yang Anda buat sebelumnya.

```
$ openssl verify -CAfile <cluster ID>_manufacturer_chain.crt <cluster ID>_HsmCertificate.crt
<cluster ID>_HsmCertificate.crt: OK
```

Langkah 4. Ekstrak dan bandingkan kunci publik

Gunakan OpenSSL untuk mengekstraksi dan membandingkan kunci publik dalam sertifikat HSM dan CSR klaster, untuk memastikan bahwa mereka adalah sama.

Untuk membandingkan kunci publik, gunakan shell Linux Anda. Anda memerlukan OpenSSL, yang tersedia di sebagian besar shell Linux, tetapi Anda tidak perlu untuk langkah AWS CLI ini. Shell tidak perlu dikaitkan dengan AWS akun Anda.

Untuk mengekstraksi dan membandingkan kunci publik

1. Gunakan perintah berikut untuk mengekstraksi kunci publik dari sertifikat HSM.

```
$ openssl x509 -in <cluster ID>_HsmCertificate.crt -pubkey -noout > <cluster ID>_HsmCertificate.pub
```

2. Gunakan perintah berikut untuk mengekstraksi kunci publik dari klaster CSR.

```
$ openssl req -in <cluster ID>_ClusterCsr.csr -pubkey -noout > <cluster ID>_ClusterCsr.pub
```

3. Gunakan perintah berikut untuk membandingkan kunci publik. Jika kunci publik identik, perintah berikut tidak menghasilkan output.

```
$ diff <cluster ID>_HsmCertificate.pub <cluster ID>_ClusterCsr.pub
```

Setelah Anda memverifikasi identitas dan keaslian HSM, lanjutkan ke [Inisialisasi cluster](#).

Inisialisasi cluster di AWS CloudHSM

Setelah Anda membuat cluster dan menambahkan modul keamanan perangkat keras (HSM) Anda AWS CloudHSM, Anda dapat menginisialisasi cluster. Selesaikan langkah-langkah dalam topik berikut untuk menginisialisasi klaster .

 Note

Sebelum Anda menginisialisasi cluster, tinjau proses di mana Anda dapat [memverifikasi identitas dan keaslian klaster. HSMs](#) Proses ini opsional dan bekerja hanya sampai klaster diinisialisasi. Setelah cluster diinisialisasi, Anda tidak dapat menggunakan proses ini untuk mendapatkan sertifikat atau memverifikasi. HSMs

Topik

- [Ikhtisar](#)
- [Langkah 1. Dapatkan CSR cluster](#)
- [Langkah 2. Buat kunci pribadi untuk Root CA Anda](#)
- [Langkah 3. Tanda tangani CSR](#)
- [Langkah 4. Inisialisasi cluster](#)

Ikhtisar

Proses inisialisasi cluster menetapkan kepemilikan dan kontrol Anda atas cluster dan Anda HSMs melalui sistem otentikasi berbasis sertifikat. Proses ini secara kriptografis membuktikan bahwa Anda adalah pemilik tunggal HSMs di cluster Anda dan menciptakan fondasi kepercayaan yang akan diperlukan untuk semua koneksi masa depan ke Anda. HSMs

Halaman ini akan menunjukkan kepada Anda bagaimana melakukan hal berikut:

- Ambil permintaan penandatanganan sertifikat (CSR) klaster Anda.
- Buat dan gunakan kunci pribadi untuk membuat sertifikat root yang ditandatangani sendiri atau rantai sertifikat.
- Tanda tangani CSR klaster Anda untuk menghasilkan sertifikat HSM yang ditandatangani.
- Inisialisasi klaster Anda menggunakan sertifikat HSM yang ditandatangani dan sertifikat yang ditandatangani sendiri atau rantai sertifikat.

Saat Anda siap memulai, buka [Langkah 1. Dapatkan CSR cluster](#).

Langkah 1. Dapatkan CSR cluster

Sebelum Anda dapat menginisialisasi kluster, Anda harus mengunduh dan menandatangani permintaan penandatanganan sertifikat (CSR) yang dihasilkan oleh kluster pertama HSM. Jika Anda mengikuti langkah-langkah untuk [memverifikasi identitas kluster HSM Anda](#), Anda sudah memiliki CSR dan Anda dapat menandatangani. Jika tidak, dapatkan CSR sekarang dengan menggunakan [AWS CloudHSM konsol](#), [AWS Command Line Interface \(AWS CLI\)](#), atau AWS CloudHSM API.

Console

Untuk mendapatkan CSR (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Pilih tombol radio di sebelah ID cluster dengan HSM yang ingin Anda verifikasi.
3. Pilih Tindakan. Dari menu tarik-turun, pilih Inisialisasi.
4. Jika Anda tidak menyelesaikan [langkah sebelumnya](#) untuk membuat HSM, pilih Availability Zone (AZ) untuk HSM yang Anda buat. Kemudian pilih Buat.
5. Saat CSR sudah siap, Anda akan melihat tautan untuk mengunduhnya.

Certificate signing request

To initialize the cluster, you must download a certificate signing request (CSR) and then [sign it](#).

 **Cluster CSR**

Cluster verification certificate

Optionally, you may wish to download the HSM certificate below which generated this Cluster CSR and [verify its authenticity](#).

 **HSM certificate**

6. Pilih CSR Klaster untuk mengunduh dan menyimpan CSR.

AWS CLI

Untuk mendapatkan CSR ([AWS CLI](#))

- Pada prompt perintah, jalankan perintah [describe-clusters](#), yang mengekstraksi CSR dan menyimpannya ke file. Ganti *<cluster ID>* dengan ID cluster yang Anda [buat sebelumnya](#).

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                     --output text \
                                     --query 'Clusters[].Certificates.ClusterCsr'
\
```

```
> <cluster ID>_ClusterCsr.csr
```

AWS CloudHSM API

Untuk mendapatkan CSR (AWS CloudHSM API)

1. Kirim permintaan [DescribeClusters](#).
2. Ekstrak dan simpan CSR dari respons.

Langkah 2. Buat kunci pribadi untuk Root CA Anda

Note

Untuk kluster produksi, kunci yang akan Anda buat harus dibuat dengan cara yang aman menggunakan sumber keacakan tepercaya. Kami menyarankan Anda menggunakan HSM offsite dan offline yang aman atau setara. Simpan kunci dengan aman. Kunci menetapkan identitas cluster dan satu-satunya kendali Anda atas yang HSMs dikandungnya.

Untuk pengembangan dan pengujian, Anda dapat menggunakan alat apa pun yang nyaman (seperti OpenSSL) untuk membuat dan menandatangani sertifikat cluster. Contoh berikut menunjukkan kepada Anda cara membuat kunci. Setelah Anda menggunakan kunci untuk membuat sertifikat yang ditandatangani sendiri (lihat di bawah), Anda harus menyimpannya dengan cara yang aman. Untuk masuk ke AWS CloudHSM instans Anda, sertifikat harus ada, tetapi kunci pribadi tidak.

Tabel di bawah ini menguraikan algoritme, ukuran kunci, dan kurva yang didukung untuk pembuatan sertifikat.

Algoritma	Ukuran/Kurva
RSA PKCSv1 .5	2048, 3072, 4096
RSA-PSS	2048, 3072, 4096
ECDSA	prime256v1, secp384r1, secp521r1
Intisari	SHA-224, SHA-256, SHA-384, dan SHA-512

Gunakan perintah contoh berikut untuk membuat kunci pribadi untuk Root CA yang ditandatangani sendiri.

```
$ openssl genrsa -aes256 -out customerRootCA.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
Enter pass phrase for customerRootCA.key:
Verifying - Enter pass phrase for customerRootCA.key:
```

Langkah 3. Tanda tangani CSR

Pada langkah sebelumnya, Anda mengambil CSR klaster Anda dan membuat kunci pribadi untuk CA root Anda. Pada langkah ini, Anda akan menggunakan kunci pribadi Anda untuk menghasilkan sertifikat penandatanganan untuk menandatangani CSR klaster Anda. Topik di bawah ini akan memandu Anda melalui proses pembuatan sertifikat tunggal yang ditandatangani sendiri, atau rantai sertifikat, menggunakan OpenSSL. Anda tidak perlu AWS CLI untuk langkah ini, dan shell tidak perlu dikaitkan dengan AWS akun Anda.

Important

Untuk menginisialisasi klaster Anda, jangkar kepercayaan Anda harus mematuhi [RFC 5280](#) dan memenuhi persyaratan berikut:

- Jika menggunakan ekstensi X509v3, ekstensi X509v3 Basic Constraints harus ada.
- Jangkar kepercayaan harus berupa sertifikat yang ditandatangani sendiri.
- Nilai ekstensi tidak boleh bertentangan satu sama lain.

Pilih salah satu pendekatan berikut untuk menandatangani CSR klaster Anda:

Pilih pendekatan sertifikat Anda

Anda harus memilih salah satu dari dua pendekatan berikut. Jangan menyelesaikan kedua pendekatan tersebut.

Opsi A: Sertifikat tunggal yang ditandatangani sendiri

Buat satu sertifikat root yang ditandatangani sendiri untuk menandatangani CSR klaster Anda. Ini adalah metode langsung yang paling sederhana untuk membangun kepercayaan.

Direkomendasikan untuk:

- Lingkungan di mana PKI eksternal tidak diperlukan
- Lingkungan pengujian dan pengembangan di mana kesederhanaan lebih disukai

Pergi ke: [Buat satu sertifikat yang ditandatangani sendiri](#)

Opsi B: Rantai sertifikat dengan CA menengah

Buat rantai sertifikat menggunakan otoritas sertifikat perantara. Rantai sertifikat perantara memberikan peningkatan keamanan, skalabilitas, dan fleksibilitas dengan memungkinkan root Certificate Authority (CAs) untuk tetap offline saat mendelegasikan penerbitan sertifikat ke perantara CAs, sehingga mengurangi risiko membahayakan root CA.

Direkomendasikan untuk:

- Lingkungan di mana PKI eksternal diperlukan
- Integrasi dengan AWS Private Certificate Authority (PCA)

Contoh Integrasi AWS PCA: Anda dapat menggunakan AWS Private Certificate Authority untuk membuat dan mengelola sertifikat CA perantara Anda. Ini menyediakan manajemen siklus hidup sertifikat otomatis, termasuk pembaruan dan pencabutan, sambil mempertahankan manfaat keamanan menjaga root CA Anda tetap offline. Untuk informasi selengkapnya tentang AWS PCA, lihat [Panduan Pengguna AWS Private Certificate Authority](#).

Pergi ke: [Buat rantai otoritas sertifikat menengah \(ICA\)](#)

Buat satu sertifikat yang ditandatangani sendiri

Perangkat keras tepercaya yang Anda gunakan untuk membuat kunci privat untuk klaster produksi Anda juga harus menyediakan perangkat lunak untuk menghasilkan sertifikat yang ditandatangani sendiri menggunakan kunci tersebut. Contoh berikut menggunakan OpenSSL dan kunci pribadi yang Anda buat pada langkah sebelumnya untuk membuat sertifikat penandatanganan CA root yang ditandatangani sendiri. Sertifikat ini berlaku selama 10 tahun (3652 hari). Baca petunjuk di layar dan ikuti prompt-nya.

```
$ openssl req -new -x509 -days 3652 -key customerRootCA.key -out customerRootCA.crt
```



```

Enter pass phrase for customerRootCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:

```

Perintah ini membuat file sertifikat bernama `customerRootCA.crt`. Letakkan sertifikat ini di setiap host tempat Anda akan terhubung ke AWS CloudHSM cluster Anda. Jika Anda memberikan file nama yang berbeda atau menyimpannya di jalan selain root host Anda, Anda harus mengedit file konfigurasi klien Anda agar sesuai. Gunakan sertifikat dan kunci privat yang baru saja Anda buat untuk menandatangani permintaan penandatanganan sertifikat klaster (CSR) di langkah berikutnya.

Tanda tangani CSR klaster dengan Root CA yang ditandatangani sendiri

Perangkat keras tepercaya yang Anda gunakan untuk membuat kunci privat Anda untuk klaster produksi Anda juga harus menyediakan alat untuk menandatangani CSR menggunakan kunci tersebut. Contoh berikut menggunakan OpenSSL untuk menandatangani CSR klaster. Contoh perintah di bawah ini menandatangani CSR dengan tanda tangan sendiri `customerRootCA.crt`

```

$ openssl x509 -req -days 3652 -in <cluster ID>_ClusterCsr.csr \
  -CA <customerRootCA>.crt \
  -CAkey <customerRootCA>.key \
  -CAcreateserial \
  -out <cluster ID>_CustomerHsmCertificate.crt
Signature ok
subject=/C=US/ST=CA/O=Cavium/OU=N3FIPS/L=SanJose/CN=HSM:<HSM
  identifier>:PARTN:<partition number>, for FIPS mode
Getting CA Private Key
Enter pass phrase for <customerRootCA>.key:

```

Perintah ini membuat file sertifikat bernama `<cluster ID>_CustomerHsmCertificate.crt`. Gunakan file ini sebagai sertifikat yang ditandatangani ketika Anda menginisialisasi kluster.

Verifikasi sertifikat yang ditandatangani terhadap root CA (opsional):

```
$ openssl verify -purpose sslserver -CAfile customerRootCA.crt <cluster ID>_CustomerHsmCertificate.crt
<cluster ID>_CustomerHsmCertificate.crt: OK
```

Setelah memproduksi sertifikat HSM yang ditandatangani dengan Root CA yang ditandatangani sendiri, buka. [Langkah 4. Inisialisasi cluster](#)

Buat rantai otoritas sertifikat menengah (ICA)

Contoh berikut akan memandu Anda melalui pembuatan rantai sertifikat panjang 2, yang terdiri dari root Certificate Authority (CA) dan CA perantara. Pertama-tama Anda akan membuat sertifikat CA root yang ditandatangani sendiri, lalu menghasilkan CA perantara yang ditandatangani oleh root CA. Terakhir, Anda akan menggunakan CA perantara untuk menandatangani CSR kluster Anda, membuat rantai kepercayaan lengkap dari sertifikat HSM Anda kembali ke root CA. Pendekatan ini memberikan keamanan yang ditingkatkan dengan menjaga root CA tetap offline saat menggunakan CA perantara untuk operasi day-to-day sertifikat.

Important

Untuk menginisialisasi kluster Anda dengan rantai sertifikat, rantai Anda harus memenuhi persyaratan berikut:

- Rantai harus diurutkan, dimulai dengan CA Menengah yang menandatangani CSR cluster. Dalam urutan ini, ICA pertama harus memiliki penerbit yang cocok dengan subjek ICA berikutnya dalam rantai, dan seterusnya.
- Hanya Root CA yang harus ditandatangani sendiri, artinya penerbit dan subjeknya harus identik.
- Rantai harus terdiri dari tidak lebih dari 4 sertifikat (termasuk Root CA di akhir), dan ukuran total rantai tidak boleh melebihi 16 kb (kilobyte).
- Semua Otoritas Sertifikat (CAs) harus sesuai dengan pedoman [RFC 5280](#).

Bagian ini memberikan contoh untuk membuat rantai otoritas sertifikat perantara menggunakan dua pendekatan berbeda: OpenSSL untuk pembuatan sertifikat lokal, dan AWS Private Certificate

Authority (PCA) untuk layanan sertifikat terkelola. Pilih pendekatan yang paling sesuai dengan lingkungan dan persyaratan keamanan Anda.

Note

Contoh berikut adalah kasus penggunaan umum dan keduanya disederhanakan, menggunakan konfigurasi paling dasar. Untuk lingkungan produksi, tinjau opsi konfigurasi tambahan dan persyaratan keamanan khusus untuk kasus penggunaan Anda.

OpenSSL

Buat file konfigurasi OpenSSL dengan ekstensi v3 umum untuk CA:

```
$ cat > ca-extensions.conf <<EOF
[req]
distinguished_name = req_distinguished_name
[req_distinguished_name]
C = Country Name (2 letter code)
ST = State or Province Name (full name)
L = Locality Name (eg, city)
O = Organization Name (eg, company)
OU = Organizational Unit Name (eg, section)
CN = Common Name (e.g. server FQDN or YOUR name)
[v3_ca]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical,CA:true
keyUsage = critical, keyCertSign, cRLSign, digitalSignature
EOF
```

Hasilkan Root CA yang ditandatangani sendiri menggunakan OpenSSL:

```
$ openssl req -new -x509 -days 3652 -key customerRootCA.key -out customerRootCA.crt
-extentions v3_ca -config ca-extensions.conf
Enter pass phrase for customerRootCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
```

```

-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:

```

Hasilkan kunci CA Menengah:

```

$ openssl genrsa -aes256 -out intermediateCA.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
Enter pass phrase for intermediateCA.key:
Verifying - Enter pass phrase for intermediateCA.key:

```

Buat permintaan penandatanganan sertifikat CA Menengah (CSR):

```

$ openssl req -new -key intermediateCA.key -out intermediateCA.csr
Enter pass phrase for intermediateCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:

```

Menggunakan Root CA yang ditandatangani sendiri, buat sertifikat CA Menengah:

```
$ openssl x509 -req -in intermediateCA.csr \
  -CA customerRootCA.crt \
  -CAkey customerRootCA.key \
  -CAcreateserial \
  -days 3652 \
  -extensions v3_ca \
  -extfile ca-extensions.conf \
  -out intermediateCA.crt
```

Certificate request self-signature ok
 subject=C= , ST= , L= , O= , OU=

Gabungkan sertifikat ke dalam file rantai:

```
$ cat intermediateCA.crt customerRootCA.crt > chainCA.crt

-----BEGIN CERTIFICATE-----
[Intermediate CA]
-----END CERTIFICATE-----
...
...
-----BEGIN CERTIFICATE-----
[Root CA]
-----END CERTIFICATE-----
```

Tanda tangani CSR klaster dengan CA perantara Anda:

```
$ openssl x509 -req -days 3652 -in <cluster ID>_ClusterCsr.csr \
  -CA intermediateCA.crt \
  -CAkey intermediateCA.key \
  -CAcreateserial \
  -out <cluster ID>_CustomerHsmCertificate.crt
```

Signature ok

subject=/C=US/ST=CA/O=Cavium/OU=N3FIPS/L=SanJose/CN=HSM:<HSM
 identifier>:PARTN:<partition number>, for FIPS mode

Getting CA Private Key

Enter pass phrase for intermediateCA.key:

AWS PCA

Membuat dan mengaktifkan Root CA menggunakan AWS Private Certificate Authority:

```
$ # 1. Create Root CA
aws acm-pca create-certificate-authority \
  --certificate-authority-configuration \
    "KeyAlgorithm=RSA_4096,
    SigningAlgorithm=SHA256WITHRSA,

    Subject={Country=US,Organization=MyOrg,OrganizationalUnit=IT,CommonName=RootCA}" \
  --certificate-authority-type ROOT

# Store the Root CA Authority ARN from the previous output
ROOT_CA_AUTHORITY_ARN="arn:aws:acm-pca:<region>:<account-id>:certificate-
authority/<ca-authority-id>"

# 2. Generate Root CA CSR
aws acm-pca get-certificate-authority-csr \
  --certificate-authority-arn $ROOT_CA_AUTHORITY_ARN \
  --output text > customerRootCA.csr

# 3. Self-sign Root CA Certificate
aws acm-pca issue-certificate \
  --certificate-authority-arn $ROOT_CA_AUTHORITY_ARN \
  --csr fileb://customerRootCA.csr \
  --signing-algorithm SHA256WITHRSA \
  --template-arn arn:aws:acm-pca:::template/RootCACertificate/V1 \
  --validity Value=3652,Type=DAYS

# Store the Root CA certificate ARN from the previous output
ROOT_CA_ARN="arn:aws:acm-pca:<region>:<account-id>:certificate-authority/<ca-
authority-id>/certificate/<cert-id>"

# 4. Retrieve the Root CA certificate
aws acm-pca get-certificate \
  --certificate-authority-arn $ROOT_CA_AUTHORITY_ARN \
  --certificate-arn $ROOT_CA_ARN \
  --output text > customerRootCA.crt

# 5. Import the Root CA Certificate
aws acm-pca import-certificate-authority-certificate \
  --certificate-authority-arn $ROOT_CA_AUTHORITY_ARN \
  --certificate fileb://customerRootCA.crt
```

Buat dan aktifkan CA bawahan (juga dikenal sebagai CA perantara):

```
$ # 6. Create Subordinate CA
aws acm-pca create-certificate-authority \
  --certificate-authority-configuration \
    "KeyAlgorithm=RSA_4096,
    SigningAlgorithm=SHA256WITHRSA,

    Subject={Country=US,Organization=MyOrg,OrganizationalUnit=IT,CommonName=SubordinateCA}" \
  --certificate-authority-type SUBORDINATE

# Store the Subordinate CA Authority ARN from the previous output
SUB_CA_AUTHORITY_ARN="arn:aws:acm-pca:<region>:<account-id>:certificate-
authority/<sub-ca-authority-id>"

# 7. Generate Subordinate CA CSR
aws acm-pca get-certificate-authority-csr \
  --certificate-authority-arn $SUB_CA_AUTHORITY_ARN \
  --output text > intermediateCA.csr

# 8. Issue Subordinate CA Certificate using Root CA
aws acm-pca issue-certificate \
  --certificate-authority-arn $ROOT_CA_AUTHORITY_ARN \
  --csr fileb://intermediateCA.csr \
  --signing-algorithm SHA256WITHRSA \
  --template-arn arn:aws:acm-pca:::template/SubordinateCACertificate_PathLen0/V1 \
  --validity Value=3651,Type=DAYS

# Store the Subordinate CA certificate ARN from the previous output
SUB_CA_ARN="arn:aws:acm-pca:<region>:<account-id>:certificate-authority/<sub-ca-
authority-id>"

# 9. Retrieve Subordinate CA Certificate
aws acm-pca get-certificate \
  --certificate-authority-arn $ROOT_CA_AUTHORITY_ARN \
  --certificate-arn $SUB_CA_ARN \
  --query 'Certificate' \
  --output text > intermediateCA.crt

# 10. Import the Subordinate CA Certificate
```

```
aws acm-pca import-certificate-authority-certificate \
  --certificate-authority-arn $SUB_CA_AUTHORITY_ARN \
  --certificate fileb://intermediateCA.crt \
  --certificate-chain fileb://customerRootCA.crt
```

Gabungkan sertifikat ke dalam file rantai:

```
$ cat intermediateCA.crt customerRootCA.crt > chainCA.crt

-----BEGIN CERTIFICATE-----
[Intermediate CA]
-----END CERTIFICATE-----
...
...
-----BEGIN CERTIFICATE-----
[Root CA]
-----END CERTIFICATE-----
```

Tanda tangani CSR cluster menggunakan AWS PCA:

```
$ aws acm-pca issue-certificate \
  --certificate-authority-arn $SUB_CA_AUTHORITY_ARN \
  --csr fileb://<cluster ID>_ClusterCsr.csr \
  --signing-algorithm SHA256WITHRSA \
  --template-arn arn:aws:acm-pca::template/EndEntityCertificate/V1 \
  --validity Value=3650,Type=DAYS

# Store your cluster's cert ARN from the previous output
CLUSTER_CERT_ARN="arn:aws:acm-pca:<region>:<account-id>:certificate-
authority/<cluster-cert-arn>"
```

Unduh sertifikat cluster yang ditandatangani:

```
$ aws acm-pca get-certificate \
  --certificate-authority-arn $SUB_CA_AUTHORITY_ARN \
  --certificate-arn $CLUSTER_CERT_ARN \
  --output text --query Certificate > <cluster ID>_CustomerHsmCertificate.crt
```


Perintah ini membuat file sertifikat bernama `<cluster ID>_CustomerHsmCertificate.crt`. Gunakan file ini sebagai sertifikat yang ditandatangani ketika Anda menginisialisasi kluster.

Verifikasi sertifikat yang ditandatangani terhadap rantai sertifikat (opsional):

```
$ openssl verify -purpose sslserver -CAfile chainCA.crt <cluster ID>_CustomerHsmCertificate.crt
<cluster ID>_CustomerHsmCertificate.crt: OK
```

Setelah memproduksi sertifikat HSM yang ditandatangani dengan CA perantara Anda, buka. [Langkah 4. Inisialisasi cluster](#)

Langkah 4. Inisialisasi cluster

Gunakan sertifikat HSM yang ditandatangani dan sertifikat penandatanganan Anda untuk menginisialisasi kluster Anda. Anda dapat menggunakan [AWS CloudHSM konsol](#), the [AWS CLI](#), atau AWS CloudHSM API.

Console

Untuk menginisialisasi kluster (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Pilih tombol radio di sebelah ID cluster dengan HSM yang ingin Anda verifikasi.
3. Pilih Tindakan. Dari menu tarik-turun, pilih Inisialisasi.
4. Jika Anda tidak menyelesaikan [langkah sebelumnya](#) untuk membuat HSM, pilih Availability Zone (AZ) untuk HSM yang Anda buat. Kemudian pilih Buat.
5. Pada Unduh permintaan penandatanganan sertifikat, pilih Selanjutnya. Jika Selanjutnya tidak tersedia, pertama-tama pilih salah satu tautan CSR atau sertifikat. Lalu, pilih Selanjutnya.
6. Pada halaman Tanda tangani permintaan penandatanganan sertifikat (CSR), pilih Berikutnya.
7. Pada halaman Unggah sertifikat, lakukan hal berikut:
 - a. Di samping Sertifikat kluster, pilih Unggah file. Kemudian, cari dan pilih sertifikat HSM yang Anda tanda tangani sebelumnya. Jika Anda menyelesaikan langkah-langkah di bagian sebelumnya, pilih file bernama `<cluster ID>_CustomerHsmCertificate.crt`.
 - b. Di samping Menerbitkan sertifikat, pilih Unggah file. Kemudian pilih sertifikat penandatanganan Anda berdasarkan pendekatan yang Anda pilih:

- Jika Anda memilih Opsi A (sertifikat tunggal yang ditandatangani sendiri): Pilih file bernama *<customerRootCA>.cert*
 - Jika Anda memilih Opsi B (rantai sertifikat): Pilih file bernama *<chainCA>.cert*
- c. Pilih Unggah dan inisialisasi.

AWS CLI

Untuk menginisialisasi sebuah klaster ([AWS CLI](#))

- Pada prompt perintah, jalankan perintah [initialize-cluster](#). Berikan yang berikut ini:
 - ID klaster yang Anda buat sebelumnya.
 - Sertifikat HSM yang Anda tandatangi sebelumnya. Jika Anda menyelesaikan langkah-langkah di bagian sebelumnya, file tersimpan dengan nama *<cluster ID>_CustomerHsmCertificate.cert*.
 - Sertifikat penandatanganan Anda berdasarkan pendekatan yang Anda pilih:
 - Jika Anda memilih Opsi A (sertifikat tunggal yang ditandatangani sendiri): Gunakan file bernama *<customerRootCA>.cert*
 - Jika Anda memilih Opsi B (rantai sertifikat): Gunakan file bernama *<chainCA>.cert*

```
$ aws cloudhsmv2 initialize-cluster --cluster-id <cluster ID> \
                                     --signed-cert file://<cluster
                                     ID>_CustomerHsmCertificate.cert \
                                     --trust-anchor file://<customerRootCA.cert OR
                                     chainCA.cert>
{
  "State": "INITIALIZE_IN_PROGRESS",
  "StateMessage": "Cluster is initializing. State will change to INITIALIZED
upon completion."
}
```

AWS CloudHSM API

Untuk menginisialisasi cluster (AWS CloudHSM API)

- Kirim permintaan [InitializeCluster](#) dengan yang berikut ini:

- ID kluster yang Anda buat sebelumnya.
- Sertifikat HSM yang Anda tandatangani sebelumnya. Jika Anda menyelesaikan langkah-langkah di bagian sebelumnya, file tersimpan dengan nama `<cluster ID>_CustomerHsmCertificate.crt`.
- Sertifikat penandatanganan Anda berdasarkan pendekatan yang Anda pilih:
 - Jika Anda memilih Opsi A (sertifikat tunggal yang ditandatangani sendiri): Gunakan file bernama `<customerRootCA>.crt`
 - Jika Anda memilih Opsi B (rantai sertifikat): Gunakan file bernama `<chainCA>.crt`

Instal dan konfigurasikan CloudHSM CLI

Untuk berinteraksi dengan HSM di AWS CloudHSM cluster Anda, Anda memerlukan CloudHSM CLI.

Connect ke instance klien Anda dan jalankan perintah berikut untuk men-download dan menginstal alat baris AWS CloudHSM perintah. Untuk informasi selengkapnya, lihat [Luncurkan instans klien Amazon EC2 untuk berinteraksi AWS CloudHSM](#).

Amazon Linux 2023

Amazon Linux 2023 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-cli-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.amzn2023.x86_64.rpm
```

Amazon Linux 2023 tentang ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-cli-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.amzn2023.aarch64.rpm
```

Amazon Linux 2

Amazon Linux 2 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.x86_64.rpm
```

Amazon Linux 2 tentang ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.aarch64.rpm
```

RHEL 10 (10.0+)

RHEL 10 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/cloudhsm-cli-latest.el10.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el10.x86_64.rpm
```

RHEL 10 tentang ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/cloudhsm-cli-latest.el10.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el10.aarch64.rpm
```

RHEL 9 (9.2+)

RHEL 9 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-cli-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el9.x86_64.rpm
```

RHEL 9 tentang ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-cli-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el9.aarch64.rpm
```

RHEL 8 (8.3+)

RHEL 8 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-cli-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el8.x86_64.rpm
```

RHEL 8 tentang ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-cli-latest.el8.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el8.aarch64.rpm
```

Ubuntu 24.04 LTS

Ubuntu 24.04 LTS pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-cli_latest_u24.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u24.04_amd64.deb
```

Ubuntu 24.04 LTS pada ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-cli_latest_u24.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u24.04_arm64.deb
```

Ubuntu 22.04 LTS

Ubuntu 22.04 LTS pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-  
cli_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u22.04_amd64.deb
```

Ubuntu 22.04 LTS pada ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-  
cli_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u22.04_arm64.deb
```

Windows Server 2022

Untuk Windows Server 2022 pada arsitektur x86_64, buka PowerShell sebagai administrator dan jalankan perintah berikut:

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/  
AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msixexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /  
quiet /norestart /log C:\client-install.txt' -Wait
```

Windows Server 2019

Untuk Windows Server 2019 pada arsitektur x86_64, buka PowerShell sebagai administrator dan jalankan perintah berikut:

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/  
AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msixexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /  
quiet /norestart /log C:\client-install.txt' -Wait
```

Windows Server 2016

Untuk Windows Server 2016 pada arsitektur x86_64, buka PowerShell sebagai administrator dan jalankan perintah berikut:

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msixexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /quiet /norestart /log C:\client-install.txt' -Wait
```

Gunakan perintah berikut untuk mengkonfigurasi CloudHSM CLI.

Untuk mem-bootstrap EC2 instance Linux untuk Client SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda.

```
$ sudo /opt/cloudhsm/bin/configure-cli -a <The ENI IPv4 / IPv6 addresses of the HSMs>
```

Untuk mem-bootstrap EC2 instance Windows untuk Client SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" -a <The ENI IPv4 / IPv6 addresses of the HSMs>
```

Aktifkan cluster di AWS CloudHSM

Saat Anda mengaktifkan AWS CloudHSM klaster, status klaster berubah dari yang diinisialisasi menjadi aktif. Anda kemudian dapat [mengelola pengguna modul keamanan perangkat keras \(HSM\)](#) dan [menggunakan HSM](#).

Important

Sebelum Anda dapat mengaktifkan cluster, Anda harus terlebih dahulu menyalin sertifikat penerbitan ke lokasi default untuk platform pada setiap EC2 instance yang terhubung ke

cluster (Anda membuat sertifikat penerbitan saat Anda menginisialisasi cluster). Gunakan file sertifikat yang sesuai berdasarkan pendekatan yang Anda pilih selama inisialisasi cluster:

- Jika Anda memilih Opsi A (sertifikat tunggal yang ditandatangani sendiri): Salin `customerRootCA.crt`
- Jika Anda memilih Opsi B (rantai sertifikat): Salin `chainCA.crt`

Lokasi Linux:

```
/opt/cloudhsm/etc/<customerRootCA.crt OR chainCA.crt>
```

Lokasi Windows:

```
C:\ProgramData\Amazon\CloudHSM\<customerRootCA.crt OR chainCA.crt>
```

Setelah menyalin file sertifikat, edit `/opt/cloudhsm/etc/cloudhsm-cli.cfg` file untuk memastikan nama file sertifikat cocok dengan nama sertifikat CA yang Anda salin.

Setelah menempatkan sertifikat penerbitan, instal CloudHSM CLI dan jalankan [cluster activate](#) perintah pada HSM pertama Anda. Anda akan melihat akun admin pada HSM pertama di klaster Anda memiliki peran admin yang [tidak aktif](#). Ini peran sementara yang hanya ada sebelum aktivasi cluster. Saat Anda mengaktifkan klaster, peran admin yang tidak aktif berubah menjadi admin.

Untuk mengaktifkan sebuah klaster

1. Connect ke instance klien yang sebelumnya Anda luncurkan. Untuk informasi selengkapnya, lihat [Luncurkan instans klien Amazon EC2 untuk berinteraksi AWS CloudHSM](#). Anda dapat meluncurkan instans Linux atau Windows Server.
2. Jalankan CloudHSM CLI dalam mode interaktif.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```


Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

3. (Optional) Gunakan perintah `user list` untuk menampilkan pengguna yang ada.

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "unactivated-admin",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

4. Gunakan `cluster activate` perintah untuk mengatur kata sandi admin awal.

```
aws-cloudhsm > cluster activate
Enter
password:<NewPassword>
Confirm password:<NewPassword>
{
  "error_code": 0,
  "data": "Cluster activation successful"
}
```

Kami menyarankan Anda menuliskan kata sandi baru pada lembar kerja kata sandi. Jangan hilangkan lembar kerja tersebut. Kami merekomendasikan Anda untuk mencetak salinan kata sandi, menggunakannya untuk mencatat kata sandi HSM penting, dan menyimpannya di tempat yang aman. Kami juga merekomendasikan Anda untuk menyimpan salinan lembar kerja ini dalam penyimpanan luar lokasi yang aman.

5. (Opsional) Gunakan user list perintah untuk memverifikasi bahwa tipe pengguna berubah menjadi [admin/Co](#).

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

6. Gunakan quit perintah untuk menghentikan alat CloudHSM CLI.

```
aws-cloudhsm > quit
```

Untuk informasi selengkapnya tentang bekerja dengan CloudHSM CLI atau CMU, [lihat Memahami Pengguna HSM dan Memahami Manajemen Pengguna HSM dengan CMU](#).

Siapkan TLS timbal balik antara klien dan AWS CloudHSM (disarankan)

Topik berikut menjelaskan langkah-langkah yang harus Anda selesaikan untuk mengaktifkan Transport Layer Security (mTLS) bersama antara klien dan AWS CloudHSM.

Pertimbangan-pertimbangan

- Saat ini fitur ini tersedia secara eksklusif di hsm2m.medium. Untuk informasi lebih lanjut tentang jenis HSM, lihat [AWS CloudHSM mode cluster](#).
- mTL tidak didukung untuk toko AWS CloudHSM utama yang digunakan. AWS Key Management Service

Topik

- [Langkah 1. Buat dan daftarkan jangkar kepercayaan ke HSM](#)
- [Langkah 2. Aktifkan mTL untuk AWS CloudHSM](#)
- [Langkah 3. Tetapkan penegakan mTLS untuk AWS CloudHSM](#)

Langkah 1. Buat dan daftarkan jangkar kepercayaan ke HSM

Jangkar kepercayaan harus dibuat dan didaftarkan ke HSM sebelum mengaktifkan mTL. Ini adalah proses dua langkah:

Topik

- [Buat kunci pribadi dan sertifikat root yang ditandatangani sendiri](#)
- [Daftarkan jangkar kepercayaan ke HSM](#)

Buat kunci pribadi dan sertifikat root yang ditandatangani sendiri

Note

Untuk cluster produksi, kunci yang akan Anda buat harus dibuat dengan cara yang aman menggunakan sumber keacakan tepercaya. Kami menyarankan Anda menggunakan HSM offsite dan offline yang aman atau setara. Simpan kunci dengan aman.

Untuk pengembangan dan pengujian, Anda dapat menggunakan alat apa pun yang nyaman (seperti OpenSSL) untuk membuat kunci dan menandatangani sendiri sertifikat root. Anda akan memerlukan sertifikat kunci dan root untuk menandatangani sertifikat klien di [mTL aktifkan untuk AWS CloudHSM](#).

Contoh berikut menunjukkan cara membuat kunci pribadi dan sertifikat root yang ditandatangani sendiri dengan [OpenSSL](#).

Example— Membuat kunci privat dengan OpenSSL

Gunakan perintah berikut untuk membuat kunci RSA 4096-bit yang dienkripsi dengan algoritma AES-256. Untuk menggunakan contoh ini, ganti `<mtls_ca_root_1.key>` dengan nama file tempat Anda ingin menyimpan kunci.

```
$ openssl genrsa -out <mtls_ca_root_1.key> -aes256 4096
Generating RSA private key, 4096 bit long modulus
.....+++
.+++
e is 65537 (0x10001)
Enter pass phrase for mtlsc_a_root_1.key:
Verifying - Enter pass phrase for mtlsc_a_root_1.key:
```

Example— Buat sertifikat root yang ditandatangani sendiri dengan OpenSSL

Gunakan perintah berikut untuk membuat sertifikat root yang ditandatangani sendiri bernama `mtls_ca_root_1.crt` dari kunci pribadi yang baru saja Anda buat. Sertifikat ini berlaku selama 25 tahun (9130 hari). Baca petunjuk di layar dan ikuti prompt-nya.

```
$ openssl req -new -x509 -days 9130 -key mtlsc_a_root_1.key -out mtlsc_a_root_1.crt
Enter pass phrase for mtlsc_a_root_1.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
```

```
Locality Name (eg, city) []:  
Organization Name (eg, company) [Internet Widgits Pty Ltd]:  
Organizational Unit Name (eg, section) []:  
Common Name (e.g. server FQDN or YOUR name) []:  
Email Address []:
```

Daftarkan jangkar kepercayaan ke HSM

Setelah membuat sertifikat root yang ditandatangani sendiri, admin harus mendaftarkannya sebagai jangkar kepercayaan dengan cluster. AWS CloudHSM

Untuk mendaftarkan jangkar kepercayaan dengan HSM

1. Gunakan perintah berikut untuk memulai CloudHSM CLI:

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Menggunakan CloudHSM CLI, masuk sebagai admin.

```
aws-cloudhsm > login --username <admin> --role admin  
Enter password:  
{  
  "error_code": 0,  
  "data": {  
    "username": "<admin>",  
    "role": "admin"  
  }  
}
```

3. Gunakan [Daftarkan jangkar kepercayaan dengan CloudHSM CLI](#) perintah untuk mendaftarkan jangkar kepercayaan. Untuk informasi selengkapnya, lihat contoh berikut atau gunakan perintah `help cluster mtls register-trust-anchor`.

Example— Daftarkan jangkar kepercayaan dengan cluster AWS CloudHSM

Contoh berikut menunjukkan cara menggunakan cluster `mtls register-trust-anchor` perintah di CloudHSM CLI untuk mendaftarkan jangkar kepercayaan ke HSM. Untuk menggunakan perintah ini, admin harus login ke HSM. Ganti nilai-nilai ini dengan nilai Anda sendiri:

```
aws-cloudhsm > cluster mtls register-trust-anchor --path </path/mtls_ca_root_1.crt>
{
  "error_code": 0,
  "data": {
    "trust_anchor": {
      "certificate-reference": "0x01",
      "certificate": "<PEM Encoded Certificate>",
      "cluster-coverage": "full"
    }
  }
}
```

Note

AWS CloudHSM mendukung pendaftaran sertifikat perantara sebagai jangkar kepercayaan. Dalam kasus seperti itu, seluruh file rantai sertifikat yang dikodekan PEM perlu didaftarkan ke HSM, dengan sertifikat dalam urutan hierarkis. AWS CloudHSM mendukung rantai sertifikat 6980 byte.

Setelah berhasil mendaftarkan jangkar kepercayaan, Anda dapat menjalankan `cluster mtls list-trust-anchors` perintah untuk memeriksa jangkar kepercayaan terdaftar saat ini, seperti yang ditunjukkan di bawah ini:

```
aws-cloudhsm > cluster mtls list-trust-anchors
{
  "error_code": 0,
  "data": {
    "trust_anchors": [
      {
        "certificate-reference": "0x01",
        "certificate": "<PEM Encoded Certificate>",
        "cluster-coverage": "full"
      }
    ]
  }
}
```

```
}  
}
```

Note

Jumlah maksimum jangka kepercayaan dapat didaftarkan ke hsm2m.medium adalah dua (2).

Langkah 2. Aktifkan mTL untuk AWS CloudHSM

Untuk mengaktifkan mTL AWS CloudHSM, Anda perlu membuat kunci pribadi dan sertifikat klien yang ditandatangani oleh sertifikat root yang kami [buat di Buat dan daftarkan jangka kepercayaan ke HSM, lalu gunakan salah satu](#) alat konfigurasi Client SDK 5 untuk mengatur jalur kunci pribadi dan jalur rantai sertifikat klien.

Topik

- [Buat kunci pribadi dan rantai sertifikat klien](#)
- [Konfigurasi mTL untuk Klien SDK 5](#)

Buat kunci pribadi dan rantai sertifikat klien

Example— Membuat kunci privat dengan OpenSSL

Gunakan perintah berikut untuk membuat kunci RSA 4096-bit. Untuk menggunakan contoh ini, ganti `<ssl-client.key>` dengan nama file tempat Anda ingin menyimpan kunci.

```
$ openssl genrsa -out <ssl-client.key> 4096  
Generating RSA private key, 4096 bit long modulus  
.....+++  
.+++  
e is 65537 (0x10001)
```

Example— Buat permintaan penandatanganan sertifikat (CSR) dengan OpenSSL

Gunakan perintah berikut untuk menghasilkan permintaan penandatanganan sertifikat (CSR) dari kunci pribadi yang baru saja Anda buat. Baca petunjuk di layar dan ikuti prompt-nya.

```
$ openssl req -new -key <ssl-client.key> -out <ssl-client.csr>
```

You are about to be asked to enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

```
-----
```

Country Name (2 letter code) [AU]:

State or Province Name (full name) [Some-State]:

Locality Name (eg, city) []:

Organization Name (eg, company) [Internet Widgits Pty Ltd]:

Organizational Unit Name (eg, section) []:

Common Name (e.g. server FQDN or YOUR name) []:

Email Address []:

Example— Tanda tangani CSR dengan sertifikat root

Gunakan perintah berikut untuk menandatangani CSR dengan sertifikat root yang kami buat dan daftarkan di [Buat dan daftarkan jangkar kepercayaan ke HSM](#) dan buat sertifikat klien bernama. `ssl-client.crt` Sertifikat ini berlaku selama 5 tahun (1826 hari).

```
$ openssl x509 -req -days 1826 -in <ssl-client.csr> -CA <mtls_ca_root_1.crt> -CAkey <mtls_ca_root_1.key> -CAcreateserial -out <ssl-client.crt>
```

Example— Buat rantai sertifikat klien

Gunakan perintah berikut untuk menggabungkan sertifikat klien dan sertifikat root yang kami buat dan daftarkan di [Buat dan daftarkan jangkar kepercayaan ke HSM](#) dan buat rantai sertifikat klien bernama `ssl-client.pem`, yang akan digunakan untuk mengkonfigurasi pada langkah berikutnya.

```
$ cat <ssl-client.crt> <mtls_ca_root_1.crt> > <ssl-client.pem>
```

Note

Jika Anda [mendaftarkan sertifikat perantara di Buat dan daftarkan jangkar kepercayaan ke HSM](#) sebagai jangkar kepercayaan, pastikan untuk menggabungkan sertifikat klien dengan seluruh rantai sertifikat untuk membuat rantai sertifikat klien.

Konfigurasi mTL untuk Klien SDK 5

Gunakan salah satu alat konfigurasi Client SDK 5 untuk mengaktifkan TLS bersama dengan menyediakan jalur kunci klien dan jalur rantai sertifikat klien yang tepat. Untuk informasi selengkapnya tentang alat konfigurasi untuk Client SDK 5, lihat [???](#).

PKCS #11 library

Untuk menggunakan sertifikat dan kunci khusus untuk otentikasi timbal balik klien TLS-HSM dengan Client SDK 5 di Linux

1. Salin kunci dan sertifikat Anda ke direktori yang sesuai.

```
$ sudo cp ssl-client.pem </opt/cloudhsm/etc>
$ sudo cp ssl-client.key </opt/cloudhsm/etc>
```

2. Gunakan alat konfigurasi untuk menentukan `ssl-client.pem` dan `ssl-client.key`.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 \
    --client-cert-hsm-tls-file </opt/cloudhsm/etc/ssl-client.pem> \
    --client-key-hsm-tls-file </opt/cloudhsm/etc/ssl-client.key>
```

Untuk menggunakan sertifikat kustom dan kunci untuk otentikasi timbal balik klien TLS-HSM dengan Klien SDK 5 pada Windows

1. Salin kunci dan sertifikat Anda ke direktori yang sesuai.

```
cp ssl-client.pem <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem>
cp ssl-client.key <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

2. Dengan PowerShell penerjemah, gunakan alat konfigurasi untuk menentukan `ssl-client.pem` dan `ssl-client.key`.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" `
    --client-cert-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.pem> `
    --client-key-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.key>
```

OpenSSL Dynamic Engine

Untuk menggunakan sertifikat dan kunci khusus untuk otentikasi timbal balik klien TLS-HSM dengan Client SDK 5 di Linux

1. Salin kunci dan sertifikat Anda ke direktori yang sesuai.

```
$ sudo cp ssl-client.pem </opt/cloudhsm/etc>
sudo cp ssl-client.key </opt/cloudhsm/etc>
```

2. Gunakan alat konfigurasi untuk menentukan `ssl-client.pem` dan `ssl-client.key`.

```
$ sudo /opt/cloudhsm/bin/configure-dyn \
    --client-cert-hsm-tls-file </opt/cloudhsm/etc/ssl-client.pem> \
    --client-key-hsm-tls-file </opt/cloudhsm/etc/ssl-client.key>
```

Key Storage Provider (KSP)

Untuk menggunakan sertifikat kustom dan kunci untuk otentikasi timbal balik klien TLS-HSM dengan Klien SDK 5 pada Windows

1. Salin kunci dan sertifikat Anda ke direktori yang sesuai.

```
cp ssl-client.pem <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem>
cp ssl-client.key <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

2. Dengan PowerShell penerjemah, gunakan alat konfigurasi untuk menentukan `ssl-client.pem` dan `ssl-client.key`.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" `
    --client-cert-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.pem> `
    --client-key-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.key>
```

JCE provider

Untuk menggunakan sertifikat dan kunci khusus untuk otentikasi timbal balik klien TLS-HSM dengan Client SDK 5 di Linux

1. Salin kunci dan sertifikat Anda ke direktori yang sesuai.

```
$ sudo cp ssl-client.pem </opt/cloudhsm/etc>
sudo cp ssl-client.key </opt/cloudhsm/etc>
```

2. Gunakan alat konfigurasi untuk menentukan `ssl-client.pem` dan `ssl-client.key`.

```
$ sudo /opt/cloudhsm/bin/configure-jce \
    --client-cert-hsm-tls-file </opt/cloudhsm/etc/ssl-client.pem> \
    --client-key-hsm-tls-file </opt/cloudhsm/etc/ssl-client.key>
```

Untuk menggunakan sertifikat kustom dan kunci untuk otentikasi timbal balik klien TLS-HSM dengan Klien SDK 5 pada Windows

1. Salin kunci dan sertifikat Anda ke direktori yang sesuai.

```
cp ssl-client.pem <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem>
cp ssl-client.key <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

2. Dengan PowerShell penerjemah, gunakan alat konfigurasi untuk menentukan `ssl-client.pem` dan `ssl-client.key`.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" `
    --client-cert-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.pem> `
    --client-key-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.key>
```

CloudHSM CLI

Untuk menggunakan sertifikat dan kunci khusus untuk otentikasi timbal balik klien TLS-HSM dengan Client SDK 5 di Linux

1. Salin kunci dan sertifikat Anda ke direktori yang sesuai.

```
$ sudo cp ssl-client.pem </opt/cloudhsm/etc>
sudo cp ssl-client.key </opt/cloudhsm/etc>
```

- Gunakan alat konfigurasi untuk menentukan `ssl-client.pem` dan `ssl-client.key`.

```
$ sudo /opt/cloudhsm/bin/configure-cli \
    --client-cert-hsm-tls-file </opt/cloudhsm/etc/ssl-client.pem> \
    --client-key-hsm-tls-file </opt/cloudhsm/etc/ssl-client.key>
```

Untuk menggunakan sertifikat kustom dan kunci untuk otentikasi timbal balik klien TLS-HSM dengan Klien SDK 5 pada Windows

- Salin kunci dan sertifikat Anda ke direktori yang sesuai.

```
cp ssl-client.pem <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem>
cp ssl-client.key <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

- Dengan PowerShell penerjemah, gunakan alat konfigurasi untuk menentukan `ssl-client.pem` dan `ssl-client.key`.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" `
    --client-cert-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.pem> `
    --client-key-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.key>
```

Langkah 3. Tetapkan penegakan mTLS untuk AWS CloudHSM

Setelah mengkonfigurasi dengan salah satu alat konfigurasi Client SDK 5, koneksi antara klien dan AWS CloudHSM akan menjadi TLS bersama di cluster. Namun, menghapus jalur kunci pribadi dan jalur rantai sertifikat klien dari file konfigurasi akan mengubah koneksi menjadi TLS biasa lagi. Anda dapat menggunakan CloudHSM CLI untuk mengatur penegakan mtls di cluster dengan menyelesaikan langkah-langkah berikut:

- Gunakan perintah berikut untuk memulai CloudHSM CLI:

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Menggunakan CloudHSM CLI, masuk sebagai admin.

```
aws-cloudhsm > login --username <admin> --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<admin>",
    "role": "admin"
  }
}
```

Note

1. Pastikan Anda telah mengonfigurasi CloudHSM CLI dan memulai CloudHSM CLI di bawah koneksi mTLS.
2. Anda harus masuk sebagai pengguna admin default dengan nama pengguna sebagai admin sebelum menetapkan penegakan mTLS.

3. Gunakan [Atur tingkat penegakan mTLS dengan CloudHSM CLI](#) perintah untuk mengatur penegakan hukum. Untuk informasi selengkapnya, lihat contoh berikut atau gunakan perintah `help cluster mtls set-enforcement`.

Example— Tetapkan penegakan mTLS dengan cluster AWS CloudHSM

Contoh berikut menunjukkan cara menggunakan `cluster mtls set-enforcement` perintah di CloudHSM CLI untuk mengatur penegakan mTLS dengan HSM. Untuk menggunakan perintah ini, admin dengan nama pengguna sebagai admin harus login ke HSM.

```
aws-cloudhsm > cluster mtls set-enforcement --level cluster
{
```

```
"error_code": 0,  
"data": {  
  "message": "Mtls enforcement level set to Cluster successfully"  
}  
}
```

 Warning

Setelah Anda menerapkan penggunaan mTLS di cluster, semua koneksi non-mTLS yang ada akan dihapus dan Anda hanya dapat terhubung ke cluster dengan sertifikat mTLS.

Buat dan gunakan kunci di AWS CloudHSM

Sebelum Anda dapat membuat dan menggunakan kunci di klaster baru, buat pengguna modul keamanan perangkat keras (HSM) dengan AWS CloudHSM CLI Untuk informasi selengkapnya, lihat [Memahami Tugas Manajemen Pengguna HSM](#), [Memulai AWS CloudHSM Command Line Interface \(CLI\)](#), dan [Cara Mengelola PenggunaHSM](#).

 Note

Jika menggunakan Client SDK 3, gunakan [CloudHSM Management Utility \(CMU\)](#) alih-alih [CloudHSM CLI](#).

Setelah Anda membuat pengguna HSM, Anda dapat masuk ke HSM dan mengelola kunci menggunakan salah satu opsi ini:

- Gunakan [utilitas manajemen kunci, alat baris perintah](#)
- Bangun aplikasi C menggunakan [pustaka PKCS #11](#)
- Bangun aplikasi Java menggunakan [Penyedia JCE](#)
- Gunakan [Mesin Dinamis OpenSSL langsung dari baris perintah](#)
- Gunakan Mesin Dinamis OpenSSL untuk pengosongan TLS dengan [server web NGINX dan Apache](#)
- Gunakan Penyedia Penyimpanan Kunci (KSP) untuk AWS CloudHSM dengan [Microsoft Windows Server Certificate Authority \(CA\)](#)
- Gunakan Penyedia Penyimpanan Kunci (KSP) untuk AWS CloudHSM [Microsoft Sign Tool](#)

- Gunakan Penyedia Penyimpanan Kunci (KSP) untuk TLS offload dengan server [web Internet Information Server \(IIS\)](#)

Praktik terbaik untuk AWS CloudHSM

Lakukan praktik terbaik dalam topik ini untuk digunakan secara efektif AWS CloudHSM.

Daftar Isi

- [AWS CloudHSM praktik terbaik manajemen klaster](#)
- [AWS CloudHSM praktik terbaik manajemen pengguna](#)
- [AWS CloudHSM praktik terbaik manajemen kunci](#)
- [AWS CloudHSM praktik terbaik integrasi aplikasi](#)
- [AWS CloudHSM memantau praktik terbaik](#)

AWS CloudHSM praktik terbaik manajemen klaster

Ikuti praktik terbaik di bagian ini saat membuat, mengakses, dan mengelola AWS CloudHSM klaster Anda.

Skala klaster Anda untuk menangani lalu lintas puncak

Beberapa faktor dapat memengaruhi throughput maksimum yang dapat ditangani klaster Anda, termasuk ukuran instans klien, ukuran cluster, topografi jaringan, dan operasi kriptografi yang Anda perlukan untuk kasus penggunaan Anda.

Sebagai titik awal, lihat topik [AWS CloudHSM informasi kinerja](#) untuk perkiraan kinerja pada ukuran dan konfigurasi cluster umum. Kami menyarankan Anda menguji beban klaster Anda dengan beban puncak yang Anda antisipasi untuk menentukan apakah arsitektur Anda saat ini tangguh dan pada skala yang tepat.

Arsitek cluster Anda untuk ketersediaan tinggi

Tambahkan redundansi ke akun untuk pemeliharaan: AWS dapat mengganti HSM Anda untuk pemeliharaan terjadwal atau jika mendeteksi masalah. Sebagai aturan umum, ukuran cluster Anda harus memiliki setidaknya +1 redundansi. Misalnya, jika Anda memerlukan dua HSMs untuk layanan Anda untuk beroperasi pada waktu puncak, ukuran cluster ideal Anda akan menjadi tiga. Jika Anda mengikuti praktik terbaik terkait ketersediaan, penggantian HSM ini seharusnya tidak memengaruhi layanan Anda. Namun, operasi yang sedang berlangsung pada HSM yang diganti mungkin gagal dan harus dicoba ulang.

Sebarikan ke banyak Availability Zone: Pertimbangkan bagaimana layanan Anda akan dapat beroperasi selama pemadaman Availability Zone. HSMs AWS merekomendasikan agar Anda HSMs menyebarkan Zona Ketersediaan sebanyak mungkin. Untuk cluster dengan tiga HSMs, Anda harus HSMs tersebar di tiga Availability Zone. Tergantung pada sistem Anda, Anda mungkin memerlukan redundansi tambahan.

Memiliki setidaknya tiga HSMs untuk memastikan daya tahan untuk kunci yang baru dihasilkan

Untuk aplikasi yang membutuhkan daya tahan kunci yang baru dibuat, sebaiknya sedikitnya tiga HSMs tersebar di Availability Zone yang berbeda di suatu wilayah.

Akses aman ke klaster Anda

Gunakan subnet pribadi untuk membatasi akses ke instans Anda: Luncurkan instance Anda HSMs dan klien di subnet pribadi VPC Anda. Ini membatasi akses Anda HSMs dari dunia luar.

Gunakan titik akhir VPC untuk mengakses APIs: Pesawat AWS CloudHSM data dirancang untuk beroperasi tanpa memerlukan akses ke internet atau AWS. APIs Jika instance klien Anda memerlukan akses ke AWS CloudHSM API, Anda dapat menggunakan titik akhir VPC untuk mengakses API tanpa memerlukan akses internet pada instance klien Anda. Untuk informasi selengkapnya, lihat [AWS CloudHSM dan titik akhir VPC](#).

Kurangi biaya dengan menskalakan kebutuhan Anda

Tidak ada biaya dimuka untuk digunakan AWS CloudHSM. Anda membayar biaya per jam untuk setiap HSM yang Anda luncurkan sampai Anda mengakhiri HSM. Jika layanan Anda tidak memerlukan penggunaan terus-menerus AWS CloudHSM, Anda dapat mengurangi biaya dengan mengurangi (menghapus) Anda HSMs ke nol ketika mereka tidak diperlukan. Ketika HSMs diperlukan lagi, Anda dapat memulihkan HSMs dari cadangan. Jika, misalnya, Anda memiliki beban kerja yang mengharuskan Anda menandatangani kode sebulan sekali, khususnya pada hari terakhir bulan itu, Anda dapat meningkatkan skala klaster Anda sebelumnya, menurunkannya dengan menghapus HSMs setelah pekerjaan selesai, dan kemudian memulihkan klaster Anda untuk melakukan operasi penandatanganan lagi di akhir bulan berikutnya.

AWS CloudHSM secara otomatis membuat cadangan berkala dari HSMs dalam cluster. Saat menambahkan HSM baru di kemudian hari, AWS CloudHSM akan mengembalikan cadangan terbaru ke HSM baru sehingga Anda dapat melanjutkan penggunaan dari tempat yang sama dengan Anda

meninggalkannya. Untuk menghitung biaya AWS CloudHSM arsitektur Anda, lihat [AWS CloudHSM Harga](#).

Sumber daya terkait:

- [Gambaran umum cadangan](#)
- [Kebijakan retensi cadangan](#)
- [Menyalin cadangan AWS CloudHSM klaster di seluruh Wilayah AWS](#)

AWS CloudHSM praktik terbaik manajemen pengguna

Ikuti praktik terbaik di bagian ini untuk mengelola pengguna secara efektif di AWS CloudHSM klaster Anda. Pengguna HSM berbeda dari pengguna IAM. Pengguna dan entitas IAM yang memiliki kebijakan berbasis identitas dengan izin yang sesuai dapat membuat dengan berinteraksi dengan sumber daya HSMs melalui AWS API. Setelah HSM dibuat, Anda harus menggunakan kredensial pengguna HSM untuk mengautentikasi operasi pada HSM. Untuk panduan rinci tentang pengguna HSM, lihat [Pengguna HSM di AWS CloudHSM](#).

Lindungi kredensi pengguna HSM Anda

Sangat penting untuk menjaga kredensial pengguna HSM Anda terlindungi dengan aman karena pengguna HSM adalah entitas yang dapat mengakses dan melakukan operasi kriptografi dan manajemen pada HSM Anda. AWS CloudHSM tidak memiliki akses ke kredensial pengguna HSM Anda, dan tidak akan dapat membantu Anda jika Anda kehilangan akses ke mereka.

Memiliki setidaknya dua admin untuk mencegah penguncian

Untuk menghindari terkunci dari klaster Anda, kami sarankan Anda memiliki setidaknya dua admin jika satu kata sandi admin hilang. Jika ini terjadi, Anda dapat menggunakan admin lain untuk mengatur ulang kata sandi.

Note

Admin di Client SDK 5 identik dengan crypto officer (COs) di Client SDK 3.

Aktifkan kuorum untuk semua operasi manajemen pengguna

Kuorum memungkinkan Anda untuk menetapkan jumlah min admin yang harus menyetujui operasi manajemen pengguna sebelum operasi itu dapat dilakukan. Karena hak istimewa yang dimiliki admin, kami menyarankan Anda mengaktifkan kuorum untuk semua operasi manajemen pengguna. Ini dapat membatasi potensi dampak jika salah satu kata sandi admin Anda disusupi. Untuk informasi selengkapnya, lihat [Mengelola Kuorum](#).

Buat beberapa pengguna crypto, masing-masing dengan izin terbatas

Dengan memisahkan tanggung jawab pengguna crypto, tidak ada satu pengguna yang memiliki kendali penuh atas seluruh sistem. Untuk alasan ini, kami sarankan Anda membuat beberapa pengguna krypto dan membatasi izin masing-masing. Biasanya, ini dilakukan dengan memberi pengguna crypto yang berbeda tanggung jawab dan tindakan yang berbeda yang mereka lakukan (misalnya, memiliki satu pengguna crypto yang bertanggung jawab untuk membuat dan berbagi kunci dengan pengguna krypto lain yang kemudian menggunakannya dalam aplikasi Anda).

Sumber daya terkait:

- [Bagikan kunci menggunakan CloudHSM CLI](#)
- [Hapus berbagi kunci menggunakan CloudHSM CLI](#)

AWS CloudHSM praktik terbaik manajemen kunci

Ikuti praktik terbaik di bagian ini saat mengelola kunci AWS CloudHSM.

Pilih jenis tombol yang tepat

Saat menggunakan kunci sesi, transaksi per detik (TPS) Anda akan dibatasi pada satu HSM di mana kunci tersebut ada. Ekstra HSMs di cluster Anda tidak akan meningkatkan throughput permintaan untuk kunci itu. Jika Anda menggunakan kunci token untuk aplikasi yang sama, permintaan Anda akan diseimbangkan beban di semua yang tersedia HSMs di klaster Anda. Untuk informasi selengkapnya, lihat [Sinkronisasi kunci dan pengaturan daya tahan di AWS CloudHSM](#).

Kelola batas penyimpanan utama

HSMs memiliki batasan jumlah maksimum token dan kunci sesi yang dapat disimpan pada HSM pada satu waktu. Untuk informasi tentang batas penyimpanan utama, lihat [AWS CloudHSM kuota](#).

Jika aplikasi Anda membutuhkan lebih dari batas, Anda dapat menggunakan satu atau beberapa strategi berikut untuk mengelola kunci secara efektif:

Gunakan pembungkus tepercaya untuk menyimpan kunci Anda di penyimpanan data eksternal: Menggunakan pembungkus kunci tepercaya, Anda dapat mengatasi batas penyimpanan kunci dengan menyimpan semua kunci Anda yang dibungkus di dalam penyimpanan data eksternal. Ketika Anda diminta untuk menggunakan kunci ini, Anda dapat membuka kunci ke HSM sebagai kunci sesi, menggunakan kunci untuk operasi yang Anda butuhkan, dan kemudian membuang kunci sesi. Data kunci asli tetap disimpan dengan aman di penyimpanan data Anda untuk digunakan kapan pun Anda membutuhkannya. Menggunakan kunci tepercaya untuk melakukan ini memaksimalkan perlindungan Anda.

Mendistribusikan kunci di seluruh cluster: Strategi lain untuk mengatasi batas penyimpanan kunci adalah menyimpan kunci Anda dalam beberapa cluster. Dalam pendekatan ini, Anda mempertahankan pemetaan kunci yang disimpan di setiap cluster. Gunakan pemetaan ini untuk merutekan permintaan klien Anda ke cluster dengan kunci yang diperlukan. Untuk informasi tentang cara menyambung ke beberapa cluster dari aplikasi klien yang sama, lihat topik berikut:

- [Menghubungkan ke beberapa AWS CloudHSM cluster dengan penyedia JCE](#)
- [Beberapa konfigurasi slot dengan pustaka PKCS #11 untuk AWS CloudHSM](#)

Mengelola dan mengamankan pembungkus kunci

Kunci dapat ditandai baik diekstraksi atau tidak dapat diekstraksi melalui atribut. `EXTRACTABLE` Secara default, kunci HSM ditandai sebagai dapat diekstraksi.

Kunci yang dapat diekstraksi adalah kunci yang diizinkan untuk diekspor dari HSM melalui pembungkus kunci. Kunci yang dibungkus dienkripsi, dan harus dibuka menggunakan kunci pembungkus yang sama sebelum dapat digunakan. Kunci yang tidak dapat diekstraksi tidak boleh diekspor dari HSM dalam keadaan apa pun. Tidak ada cara untuk membuat kunci yang tidak dapat diekstraksi dapat diekstraksi. Untuk alasan ini, penting untuk mempertimbangkan apakah Anda memerlukan kunci Anda untuk diekstraksi atau tidak dan untuk mengatur atribut kunci yang sesuai.

Jika Anda memerlukan pembungkus kunci dalam aplikasi Anda, Anda harus menggunakan pembungkus kunci tepercaya untuk membatasi kemampuan pengguna HSM Anda hanya untuk wrap/unwrap kunci yang telah secara eksplisit ditandai sebagai dipercaya oleh admin. Untuk informasi selengkapnya, lihat topik tentang pembungkus kunci tepercaya. [Kunci di AWS CloudHSM](#)

Sumber daya terkait

- [Fungsi Bungkus dan Buka Bungkus](#)
- [Fungsi cipher untuk JCE](#)
- [Atribut kunci Java yang didukung untuk AWS CloudHSM Klien SDK 5](#)
- [Atribut kunci untuk CloudHSM CLI](#)

AWS CloudHSM praktik terbaik integrasi aplikasi

Ikuti praktik terbaik di bagian ini untuk mengoptimalkan cara aplikasi Anda terintegrasi dengan AWS CloudHSM klaster Anda.

Bootstrap SDK Klien Anda

Sebelum SDK klien Anda dapat terhubung ke cluster Anda, itu harus di-bootstrap. Saat mem-bootstrapping alamat IP ke cluster Anda, sebaiknya gunakan `--cluster-id` parameter jika memungkinkan. Metode ini mengisi konfigurasi Anda dengan semua alamat IP HSM di cluster Anda tanpa perlu melacak setiap alamat individual. Melakukan hal ini menambah ketahanan ekstra pada inisialisasi aplikasi Anda jika HSM sedang menjalani pemeliharaan atau selama pemadaman Zona Ketersediaan. Untuk detail selengkapnya, lihat [Bootstrap Klien SDK](#).

Otentikasi untuk melakukan operasi

Di AWS CloudHSM, Anda harus mengautentikasi ke cluster Anda sebelum Anda dapat melakukan sebagian besar operasi seperti operasi kriptografi.

Otentikasi dengan CloudHSM CLI : [Anda dapat mengautentikasi dengan CloudHSM CLI menggunakan mode perintah tunggal atau mode interaktif](#). Gunakan [Masuk ke HSM menggunakan CloudHSM CLI](#) perintah untuk mengautentikasi dalam mode interaktif. Untuk mengautentikasi dalam mode perintah tunggal, Anda harus mengatur variabel lingkungan `CLOUDHSM_ROLE` dan `CLOUDHSM_PIN`. Untuk detail tentang melakukan ini, lihat [Mode Perintah Tunggal](#). AWS CloudHSM merekomendasikan untuk menyimpan kredensial HSM Anda dengan aman saat tidak digunakan oleh aplikasi Anda.

Otentikasi dengan PKCS #11: Di PKCS #11, Anda masuk menggunakan `C_Login` API setelah membuka sesi menggunakan `C_`. `OpenSession` Anda hanya perlu melakukan satu `C_Login` per slot (cluster). Setelah Anda berhasil masuk, Anda dapat membuka sesi tambahan menggunakan `C_` `OpenSession` tanpa perlu melakukan operasi login tambahan. Untuk contoh tentang autentikasi ke PKCS #11, lihat. [Contoh kode untuk pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5](#)

Otentikasi dengan JCE: Penyedia AWS CloudHSM JCE mendukung login implisit dan eksplisit. Metode yang bekerja untuk Anda tergantung pada kasus penggunaan Anda. Jika memungkinkan, sebaiknya gunakan Implicit Login karena SDK akan secara otomatis menangani otentikasi jika aplikasi Anda terputus dari cluster Anda dan perlu diautentikasi ulang. Menggunakan login implisit juga memungkinkan Anda untuk memberikan kredensial untuk aplikasi Anda ketika menggunakan integrasi yang tidak memungkinkan Anda untuk memiliki kontrol atas kode aplikasi Anda. Untuk selengkapnya tentang metode login, lihat [Langkah 2: Berikan kredensial ke penyedia JCE](#).

Otentikasi dengan OpenSSL: Dengan OpenSSL Dynamic Engine, Anda memberikan kredensial melalui variabel lingkungan. AWS CloudHSM merekomendasikan untuk menyimpan kredensial HSM Anda dengan aman saat tidak digunakan oleh aplikasi Anda. Jika memungkinkan, Anda harus mengonfigurasi lingkungan Anda untuk mengambil dan mengatur variabel lingkungan ini secara sistematis tanpa entri manual. Untuk detail tentang otentikasi dengan OpenSSL, lihat. [Instal OpenSSL Dynamic Engine AWS CloudHSM untuk Client SDK 5](#)

Otentikasi dengan KSP: Anda dapat mengautentikasi dengan Penyedia Penyimpanan Kunci (KSP) menggunakan pengelola kredensial Windows atau variabel lingkungan, lihat. [Instal penyedia penyimpanan kunci \(KSP\) untuk AWS CloudHSM Client SDK 5](#)

Mengelola kunci secara efektif dalam aplikasi Anda

Gunakan atribut kunci untuk mengontrol apa yang dapat dilakukan kunci: Saat membuat kunci, gunakan atribut kunci untuk menentukan serangkaian izin yang akan mengizinkan atau menolak jenis operasi tertentu untuk kunci tersebut. Kami menyarankan agar kunci dibuat dengan jumlah atribut paling sedikit yang diperlukan untuk menyelesaikan tugasnya. Misalnya, kunci AES yang digunakan untuk enkripsi juga tidak boleh diizinkan untuk membungkus kunci dari HSM. Untuk informasi selengkapnya, lihat halaman atribut kami untuk Klien berikut SDKs:

- [Atribut kunci PKCS #11](#)
- [Atribut kunci JCE](#)

Jika memungkinkan, cache objek kunci untuk meminimalkan latensi: Operasi pencarian kunci akan menanyakan setiap HSM di cluster Anda. Operasi ini mahal dan tidak berskala dengan jumlah HSM di cluster Anda.

- Dengan PKCS #11, Anda menemukan kunci menggunakan API. `C_FindObjects`
- Dengan JCE, Anda menemukan kunci menggunakan. `KeyStore`

Untuk kinerja yang optimal, AWS merekomendasikan agar Anda menggunakan perintah pencarian kunci (seperti [Cari AWS CloudHSM kunci berdasarkan atribut menggunakan KMU](#) dan [Daftar kunci untuk pengguna dengan CloudHSM CLI](#)) hanya sekali selama aplikasi Anda start-up dan cache objek kunci yang dikembalikan dalam memori aplikasi. Jika Anda memerlukan objek kunci ini nanti, Anda harus mengambil objek dari cache Anda alih-alih menanyakan objek ini untuk setiap operasi yang akan menambahkan overhead kinerja yang signifikan.

Gunakan multi-threading

AWS CloudHSM mendukung aplikasi multi-threaded, tetapi ada hal-hal tertentu yang perlu diingat dengan aplikasi multi-threaded.

Dengan PKCS #11, Anda harus menginisialisasi pustaka PKCS #11 (memanggil) hanya sekali. `C_Initialize` Setiap thread harus diberi session sendiri (`C_OpenSession`). Menggunakan sesi yang sama di beberapa utas tidak disarankan.

Dengan JCE, AWS CloudHSM penyedia harus diinisialisasi hanya sekali. Jangan berbagi contoh objek SPI di seluruh utas. Misalnya, Cipher, Signature, Digest, Mac, KeyFactory atau KeyGenerator objek hanya boleh digunakan dalam konteks thread mereka sendiri.

Menangani kesalahan pelambatan

Anda mungkin mengalami kesalahan pelambatan HSM dalam keadaan berikut:

- Cluster Anda tidak diskalakan dengan benar untuk mengelola lalu lintas puncak.
- Cluster Anda tidak berukuran dengan redundansi +1 selama acara pemeliharaan.
- Pemadaman Zona Ketersediaan mengakibatkan berkurangnya jumlah yang tersedia HSMs di kluster Anda.

Lihat [Pelambatan HSM](#) untuk informasi tentang cara terbaik menangani skenario ini.

Untuk memastikan cluster Anda berukuran cukup dan tidak akan dibatasi, AWS sarankan Anda memuat tes di lingkungan Anda dengan lalu lintas puncak yang Anda harapkan.

Integrasikan percobaan ulang pada operasi cluster

AWS dapat menggantikan HSM Anda untuk alasan operasional atau pemeliharaan. Agar aplikasi Anda tahan terhadap situasi seperti itu, AWS merekomendasikan agar Anda menerapkan logika coba

lagi sisi klien pada semua operasi yang dirutekan ke klaster Anda. Percobaan ulang berikutnya pada operasi yang gagal karena penggantian diharapkan berhasil.

Menerapkan strategi pemulihan bencana

Menanggapi suatu peristiwa, mungkin perlu mengalihkan lalu lintas Anda dari seluruh cluster atau wilayah. Bagian berikut menjelaskan beberapa strategi untuk melakukan ini.

Gunakan peering VPC untuk mengakses klaster Anda dari akun atau wilayah lain: Anda dapat memanfaatkan peering VPC untuk mengakses klaster Anda AWS CloudHSM dari akun atau wilayah lain. Untuk informasi tentang cara mengaturnya, lihat [Apa itu VPC peering?](#) dalam Panduan Peering VPC. Setelah Anda membuat koneksi peering dan mengonfigurasi grup keamanan Anda dengan tepat, Anda dapat berkomunikasi dengan alamat IP HSM dengan cara yang sama seperti biasanya.

Connect ke beberapa cluster dari aplikasi yang sama: Penyedia JCE, pustaka PKCS #11, dan CloudHSM CLI di Client SDK 5 mendukung koneksi ke beberapa cluster dari aplikasi yang sama. Misalnya, Anda dapat memiliki dua cluster aktif, masing-masing di wilayah yang berbeda, dan aplikasi Anda dapat terhubung ke keduanya sekaligus dan keseimbangan beban antara keduanya sebagai bagian dari operasi normal. Jika aplikasi Anda tidak menggunakan Client SDK 5 (SDK terbaru), maka Anda tidak dapat terhubung ke beberapa cluster dari aplikasi yang sama. Atau, Anda dapat menjaga cluster lain tetap aktif dan berjalan dan, jika ada pemadaman regional, alihkan lalu lintas Anda ke cluster lain untuk meminimalkan waktu henti. Lihat halaman masing-masing untuk detailnya:

- [Beberapa konfigurasi slot dengan pustaka PKCS #11 untuk AWS CloudHSM](#)
- [Menghubungkan ke beberapa AWS CloudHSM cluster dengan penyedia JCE](#)
- [Menghubungkan ke beberapa cluster dengan CloudHSM CLI](#)

Memulihkan klaster dari cadangan: Anda dapat membuat Cluster baru dari cadangan Cluster yang ada. Untuk informasi selengkapnya, lihat [Cadangan cluster di AWS CloudHSM](#).

Terhuyung-huyung penerapan aplikasi Anda

Ikuti strategi penerapan yang terhuyung-huyung seperti penerapan berbasis gelombang progresif, penerapan satu kotak, dan penerapan bergulir untuk penerapan dan restart aplikasi klien Anda. Pendekatan ini meminimalkan dampak potensial dari perubahan sementara juga memastikan ada kapasitas yang cukup untuk melayani lalu lintas produksi selama penyebaran.

AWS CloudHSM memantau praktik terbaik

Bagian ini menjelaskan beberapa mekanisme yang dapat Anda gunakan untuk memantau kluster dan aplikasi Anda. Untuk detail tambahan tentang pemantauan, lihat [Pemantauan AWS CloudHSM](#).

Pantau log klien

Setiap SDK Klien menulis log yang dapat Anda pantau. Untuk informasi tentang pencatatan klien, lihat [Bekerja dengan log SDK AWS CloudHSM klien](#).

Pada platform yang dirancang untuk bersifat fana, seperti Amazon ECS dan AWS Lambda, mengumpulkan log klien dari file bisa jadi sulit. Dalam situasi ini, merupakan praktik terbaik untuk mengonfigurasi logging SDK Klien Anda untuk menulis log ke konsol. Sebagian besar layanan akan secara otomatis mengumpulkan output ini dan mempublikasikannya ke CloudWatch log Amazon untuk Anda simpan dan lihat.

Jika Anda menggunakan integrasi pihak ketiga apa pun di atas SDK AWS CloudHSM Klien, Anda harus memastikan bahwa Anda mengonfigurasi paket perangkat lunak tersebut untuk mencatat outputnya ke konsol juga. Output dari SDK AWS CloudHSM Klien dapat ditangkap oleh paket ini dan ditulis ke file lognya sendiri jika tidak.

Lihat informasi tentang cara mengonfigurasi opsi logging di aplikasi Anda. [AWS CloudHSM Alat konfigurasi SDK 5 klien](#)

Pantau log audit

AWS CloudHSM menerbitkan log audit ke CloudWatch akun Amazon Anda. Log audit berasal dari HSM dan melacak operasi tertentu untuk tujuan audit.

Anda dapat menggunakan log audit untuk melacak perintah manajemen apa pun yang dipanggil di HSM Anda. Misalnya, Anda dapat memicu alarm ketika Anda melihat operasi manajemen yang tidak terduga sedang dilakukan.

Lihat [Cara kerja pencatatan audit HSM](#) untuk detail selengkapnya.

Monitor AWS CloudTrail

AWS CloudHSM terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di AWS CloudHSM. AWS CloudTrail

menangkap semua panggilan API untuk AWS CloudHSM sebagai peristiwa. Panggilan yang diambil termasuk panggilan dari AWS CloudHSM konsol dan panggilan kode ke operasi AWS CloudHSM API.

Anda dapat menggunakan AWS CloudTrail untuk mengaudit panggilan API apa pun yang dibuat ke bidang AWS CloudHSM kontrol untuk memastikan bahwa tidak ada aktivitas yang tidak diinginkan yang terjadi di akun Anda.

Lihat [Bekerja dengan AWS CloudTrail dan AWS CloudHSM](#) untuk detail.

Pantau CloudWatch metrik Amazon

Anda dapat menggunakan CloudWatch metrik Amazon untuk memantau AWS CloudHSM kluster Anda secara real time. Metrik dapat dikelompokkan berdasarkan wilayah, ID cluster, atau ID HSM dan ID cluster.

Dengan menggunakan CloudWatch metrik Amazon, Anda dapat mengonfigurasi CloudWatch alarm Amazon untuk memberi tahu Anda tentang potensi masalah yang mungkin timbul yang dapat memengaruhi layanan Anda. Sebaiknya konfigurasi alarm untuk memantau hal-hal berikut:

- Mendekati batas kunci Anda pada HSM
- Mendekati batas hitungan sesi HSM pada HSM
- Mendekati batas jumlah pengguna HSM pada HSM
- Perbedaan pengguna HSM atau jumlah kunci untuk mengidentifikasi masalah sinkronisasi
- Tidak sehat HSMs untuk meningkatkan skala cluster Anda hingga AWS CloudHSM dapat menyelesaikan masalah

Untuk detail selengkapnya, lihat [Bekerja dengan CloudWatch Log Amazon dan Log AWS CloudHSM Audit](#).

Cluster di AWS CloudHSM

Cluster adalah kumpulan modul keamanan perangkat keras individu (HSM) yang AWS CloudHSM tetap sinkron. Ketika Anda melakukan tugas atau operasi pada satu HSM dalam sebuah cluster, yang lain HSMs di cluster itu secara otomatis tetap up to date.

Anda dapat mengelola AWS CloudHSM cluster Anda dari [AWS CloudHSM konsol](#) atau salah satu [alat baris perintah AWS SDKs](#) atau. Untuk informasi lebih lanjut, lihat topik berikut.

Untuk membuat klaster, lihat [Mulai menggunakan](#).

Topik berikut memberikan informasi lebih lanjut tentang cluster.

Topik

- [AWS CloudHSM arsitektur cluster](#)
- [AWS CloudHSM sinkronisasi cluster](#)
- [AWS CloudHSM cluster ketersediaan tinggi dan load balancing](#)
- [AWS CloudHSM mode cluster](#)
- [Jenis HSM di AWS CloudHSM](#)
- [Connect SDK klien ke cluster AWS CloudHSM](#)
- [Penskalaan HSMs dalam sebuah cluster AWS CloudHSM](#)
- [Menghapus sebuah cluster AWS CloudHSM](#)
- [Membuat AWS CloudHSM cluster dari backup](#)
- [Migrasi tipe HSM cluster](#)

AWS CloudHSM arsitektur cluster

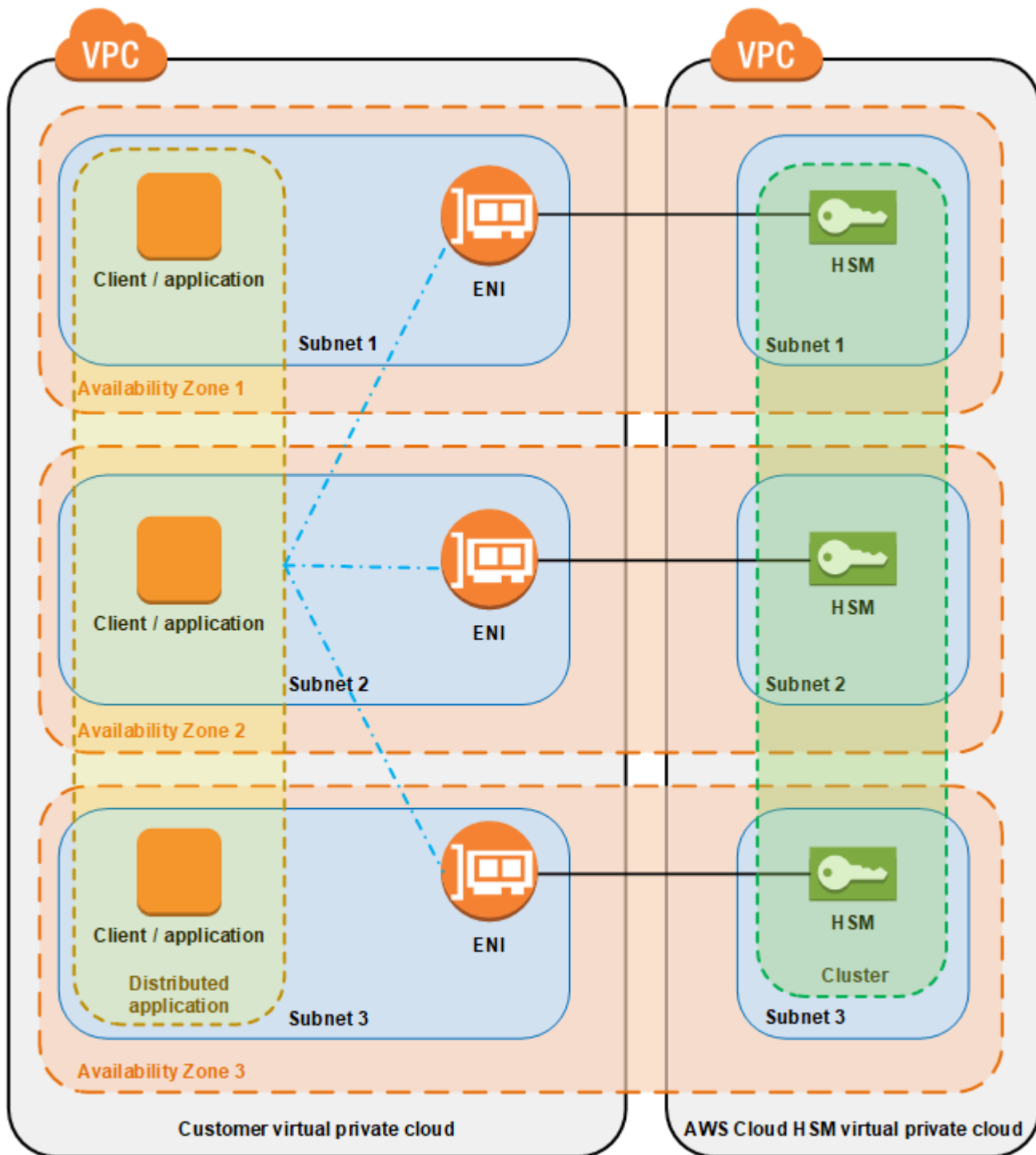
Saat membuat klaster, Anda menentukan Amazon Virtual Private Cloud (VPC) di AWS akun Anda dan satu atau beberapa subnet di VPC tersebut. Kami menyarankan Anda membuat satu subnet di setiap Availability Zone (AZ) di AWS Wilayah pilihan Anda. Anda dapat membuat subnet pribadi saat membuat VPC. Untuk mempelajari selengkapnya, lihat [Buat virtual private cloud \(VPC\) untuk AWS CloudHSM](#).

Setiap kali Anda membuat HSM, Anda menentukan klaster dan Availability Zone untuk HSM. Dengan menempatkan HSMs di Availability Zone yang berbeda, Anda mencapai redundansi dan ketersediaan tinggi jika satu Availability Zone tidak tersedia.

Saat Anda membuat HSM, AWS CloudHSM letakkan elastic network interface (ENI) di subnet yang ditentukan di akun Anda AWS . Antarmuka jaringan elastis adalah antarmuka untuk berinteraksi dengan HSM. HSM berada di VPC terpisah di AWS akun yang dimiliki oleh. AWS CloudHSM HSM dan antarmuka jaringan yang sesuai berada di Availability Zone yang sama.

Untuk berinteraksi dengan HSMs dalam cluster, Anda memerlukan perangkat lunak AWS CloudHSM klien. Biasanya Anda menginstal klien di instans Amazon EC2, yang dikenal sebagai instance klien, yang berada di VPC yang sama dengan HSM, seperti yang ditunjukkan pada gambar berikut ENIs. Itu tidak diperlukan secara teknis; Anda dapat menginstal klien pada komputer yang kompatibel, asalkan dapat terhubung ke ENIs HSM. Klien berkomunikasi dengan individu HSMs di cluster Anda melalui mereka ENIs.

Gambar berikut mewakili AWS CloudHSM cluster dengan tiga HSMs, masing-masing di Availability Zone yang berbeda di VPC.



AWS CloudHSM sinkronisasi cluster

Dalam sebuah AWS CloudHSM cluster, AWS CloudHSM jaga agar kunci HSMs pada individu tetap sinkron. Anda tidak perlu melakukan apa pun untuk menyinkronkan tombol pada Anda HSMs. Untuk menjaga agar pengguna dan kebijakan pada setiap HSM tetap sinkron, perbarui file konfigurasi AWS

CloudHSM klien sebelum Anda [mengelola pengguna HSM](#). Untuk informasi selengkapnya, lihat [Jaga agar pengguna HSM tetap sinkron](#).

Saat Anda menambahkan HSM baru ke kluster, AWS CloudHSM buat cadangan semua kunci, pengguna, dan kebijakan pada HSM yang ada. Ini kemudian mengembalikan cadangan ke HSM baru. Ini membuat keduanya tetap HSMs sinkron.

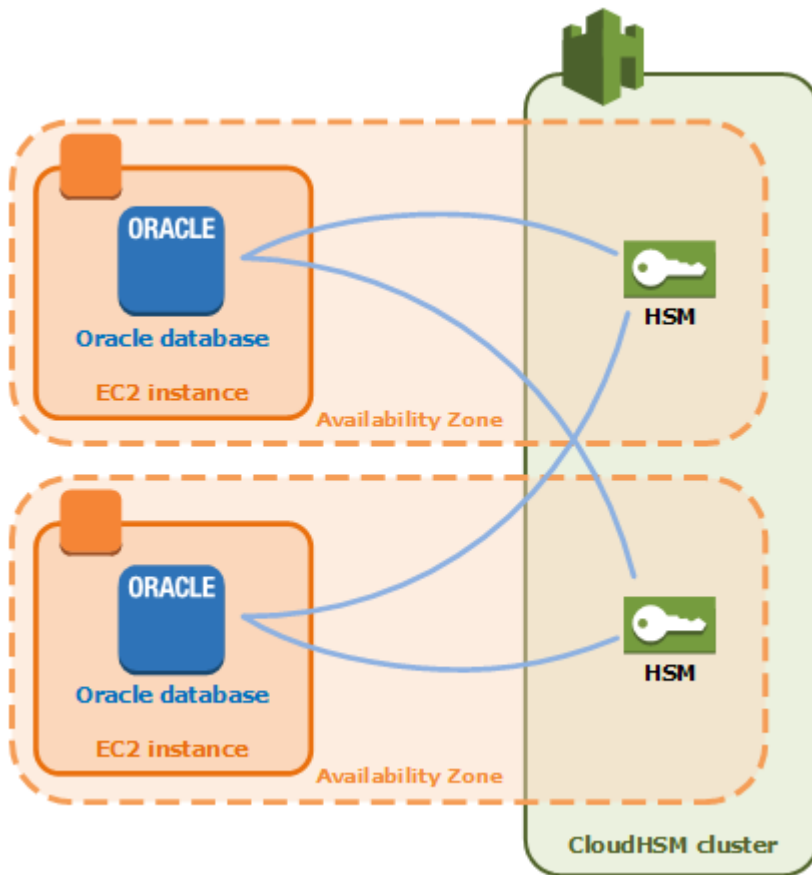
Jika HSMs dalam cluster jatuh dari sinkronisasi, AWS CloudHSM secara otomatis menyinkronkan ulang mereka. Untuk mengaktifkan ini, AWS CloudHSM gunakan kredensi pengguna [alat](#). Pengguna ini ada di semua HSMs yang disediakan oleh AWS CloudHSM dan memiliki izin terbatas. Hal ini bisa mendapatkan hash dari objek pada HSM dan dapat mengekstrak dan memasukkan objek tertutup (dienkripsi). AWS tidak dapat melihat atau mengubah pengguna atau kunci Anda dan tidak dapat melakukan operasi kriptografi menggunakan kunci tersebut.

AWS CloudHSM cluster ketersediaan tinggi dan load balancing

Saat Anda membuat AWS CloudHSM cluster dengan lebih dari satu HSM, Anda secara otomatis mendapatkan load balancing. Load balancing berarti bahwa [AWS CloudHSM klien](#) mendistribusikan operasi kriptografi HSMs di semua cluster berdasarkan kapasitas masing-masing HSM untuk pemrosesan tambahan.

Saat Anda membuat HSMs di AWS Availability Zone yang berbeda, Anda secara otomatis mendapatkan ketersediaan tinggi. Ketersediaan tinggi berarti Anda mendapatkan keandalan yang lebih tinggi karena tidak ada HSM individu yang merupakan satu titik kegagalan. Kami menyarankan Anda memiliki minimal dua HSMs di setiap cluster, dengan setiap HSM di Availability Zone yang berbeda dalam suatu AWS Wilayah.

Sebagai contoh, gambar berikut menunjukkan aplikasi basis data Oracle yang didistribusikan ke dua Availability Zone yang berbeda. Instance database menyimpan kunci master mereka dalam cluster yang menyertakan HSM di setiap Availability Zone. AWS CloudHSM secara otomatis menyinkronkan kunci ke keduanya HSMs sehingga mereka segera dapat diakses dan berlebihan.



AWS CloudHSM mode cluster

AWS CloudHSM menawarkan cluster dalam dua mode: FIPS dan non-FIPS. Dalam mode FIPS, hanya kunci dan algoritma yang divalidasi Federal Information Processing Standard (FIPS) yang dapat digunakan. Mode non-FIPS menawarkan semua kunci dan algoritma yang didukung oleh AWS CloudHSM, terlepas dari persetujuan FIPS.

Tinjau detail di halaman ini sebelum memutuskan mode cluster dan tipe HSM mana yang tepat untuk kebutuhan Anda.

Note

Semua cluster yang dibuat sebelum 10 Juni 2024 berada dalam mode FIPS dan memiliki tipe HSM `hsm1.medium`.

Untuk melihat mode cluster dan tipe HSM Anda, gunakan [perintah deskripsi-cluster](#).

Tabel berikut mencantumkan perbedaan utama antara setiap modus cluster:

Fitur pembeda	Modus FIPS	Mode non-FIPS
Kompatibilitas tipe HSM	Tersedia dengan hsm1.medium dan hsm2m.medium.	Tersedia dengan hsm2m.medium.
Kompatibilitas Backup	Hanya dapat digunakan untuk mencadangkan cluster pemulihan dalam mode FIPS.	Hanya dapat digunakan untuk mencadangkan cluster pemulihan dalam mode non-FIPS.
Pemilihan kunci	Mendukung pembuatan dan penggunaan kunci dengan mekanisme yang disetujui ¹ FIPS.	Mendukung pembuatan dan penggunaan kunci dengan semua mekanisme yang divalidasi FIPS, selain mekanisme non-validasi lainnya.
Algoritma	Mendukung AWS CloudHSM algoritma yang disetujui FIPS. ¹	Mendukung AWS CloudHSM algoritma yang disetujui FIPS dan tidak disetujui FIPS.

[1] Lihat [Pemberitahuan penghentian](#) untuk detailnya.

Sebelum memilih mode cluster, perhatikan bahwa mode cluster (FIPS atau non-FIPS) tidak dapat diubah setelah dibuat, jadi pastikan Anda memilih mode yang tepat untuk kebutuhan Anda.

Jenis HSM di AWS CloudHSM

AWS CloudHSM juga menawarkan dua jenis modul keamanan perangkat keras (HSM): hsm1.medium dan hsm2m.medium. Tinjau detail di halaman ini sebelum memutuskan jenis HSM mana yang tepat untuk kebutuhan Anda.

Selain mode cluster, AWS CloudHSM menawarkan dua jenis HSM: hsm1.medium dan hsm2m.medium. Setiap jenis HSM menggunakan perangkat keras yang berbeda, dan setiap cluster hanya dapat berisi satu jenis HSM. Tabel berikut mencantumkan perbedaan utama antara keduanya:

Fitur pembeda	hsm1.sedang	hsm2m.sedang
Kompatibilitas mode cluster	Tersedia untuk cluster dalam mode FIPS.	Tersedia untuk cluster dalam mode FIPS atau non-FIPS.
Kompatibilitas tipe jaringan	Tidak tersedia	Tersedia untuk cluster dalam mode FIPS atau non-FIPS.
Kompatibilitas Backup	Dapat digunakan untuk membuat cadangan dan mengembalikan ke klaster hsm1.medium dan hsm2m.medium dalam mode FIPS.	Hanya dapat digunakan untuk mencadangkan dan memulihkan klaster hsm2m.medium.
Kapasitas kunci	3.300 per cluster.	16.666 total kunci, dengan tombol asimetris memiliki maksimum 3.333 per cluster.
Klien SDKs	Mendukung semua Klien SDKs.	Mendukung semua Klien SDKs.
Versi SDK Klien	Kompatibel dengan SDK versi 3.1.0 dan yang lebih baru.	Kompatibel dengan Client SDK versi 5.9.0 dan yang lebih baru.
Ketersediaan wilayah	CloudHSM tidak lagi mendukung pembuatan cluster baru di Wilayah mana pun. AWS Untuk informasi selengkapnya, lihat Pemberitahuan penghentian untuk detailnya.	Tersedia di CloudHSM Wilayah AWS yang tersedia.
Performa	Untuk melihat kinerja setiap jenis HSM, lihat. AWS CloudHSM informasi kinerja	

Fitur pembeda	hsm1.sedang	hsm2m.sedang
Sertifikasi	FIPS 140-2, PCI DSS, PIN PCI,, dan sesuai dengan PCI-3DS. SOC2	FIPS 140-3, PCI DSS, PIN PCI, dan sesuai dengan PCI-3DS. SOC2

Connect SDK klien ke cluster AWS CloudHSM

Untuk menyambung ke klaster baik dengan Klien SDK 5 atau Klien SDK 3, Anda harus terlebih dahulu melakukan dua hal:

- Memiliki sertifikat penerbitan terpasang pada instans EC2
- Bootstrap Klien SDK ke klaster

Tempatkan sertifikat penerbitan pada setiap instans EC2

Anda membuat sertifikat penerbitan ketika Anda menginisialisasi klaster. Salin sertifikat penerbitan ke lokasi default untuk platform pada setiap instans EC2 yang menghubungkan ke klaster.

Linux

```
/opt/cloudhsm/etc/customerCA.crt
```

Windows

```
C:\ProgramData\Amazon\CloudHSM\customerCA.crt
```

Tentukan lokasi sertifikat penerbitan

Dengan Client SDK 5, Anda menggunakan alat konfigurasi untuk menentukan lokasi sertifikat penerbitan.

PKCS #11 library

Tempat penerbitan sertifikat pada Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --hsm-ca-cert <customerCA certificate file>
```

Tempat penerbitan sertifikat pada Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --hsm-ca-cert <customerCA certificate file>
```

OpenSSL Dynamic Engine

Tempat penerbitan sertifikat pada Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --hsm-ca-cert <customerCA certificate file>
```

OpenSSL Dynamic Engine Provider

Tempat penerbitan sertifikat pada Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --hsm-ca-cert <customerCA certificate file>
```

Key Storage Provider (KSP)

Tempat penerbitan sertifikat pada Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --hsm-ca-cert <customerCA certificate file>
```

JCE provider

Tempat penerbitan sertifikat pada Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
$ sudo /opt/cloudhsm/bin/configure-jce --hsm-ca-cert <customerCA certificate file>
```

Tempat penerbitan sertifikat pada Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --hsm-ca-cert <customerCA certificate file>
```

CloudHSM CLI

Tempat penerbitan sertifikat pada Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
$ sudo /opt/cloudhsm/bin/configure-cli --hsm-ca-cert <customerCA certificate file>
```

Tempat penerbitan sertifikat pada Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --hsm-ca-cert <customerCA certificate file>
```

Untuk informasi selengkapnya, lihat [Alat Konfigurasi](#).

Untuk informasi selengkapnya tentang menginisialisasi kluster atau membuat dan menandatangani sertifikat, lihat [Menginisialisasi](#) Cluster.

Bootstrap Klien SDK

Proses bootstrap berbeda tergantung pada versi Klien SDK yang Anda gunakan, tetapi Anda harus memiliki alamat IP dari salah satu modul keamanan perangkat keras (HSM) di kluster. Anda dapat menggunakan alamat IP dari setiap HSM yang terlampir pada kluster Anda. Setelah SDK Klien terhubung, ia mengambil alamat IP tambahan apa pun HSMs dan melakukan penyeimbangan beban dan operasi sinkronisasi kunci sisi klien.

Untuk mendapatkan alamat IP untuk kluster

Untuk mendapatkan alamat IP untuk HSM (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Untuk mengubah Wilayah AWS, gunakan pemilih Wilayah di sudut kanan atas halaman.
3. Untuk membuka halaman detail kluster, dalam tabel kluster, pilih ID kluster.
4. Untuk mendapatkan alamat IP, buka HSMs tab. Untuk IPv4 cluster, pilih alamat yang tercantum di bawah IPv4 alamat ENI. Untuk cluster dual-stack gunakan ENI IPv4 atau alamat ENI. IPv6

Untuk mendapatkan alamat IP untuk HSM (AWS CLI)

- Dapatkan alamat IP dari HSM dengan menggunakan perintah [describe-clusters](#) dari AWS CLI. Dalam output dari perintah, alamat IP dari HSMs adalah nilai-nilai `EniIp` dan `EniIPv6` (jika itu adalah cluster dual-stack).

```
$ aws cloudhsmv2 describe-clusters
{
  "Clusters": [
    { ... }
    "Hsms": [
      {
...
        "EniIp": "10.0.0.9",
...
      },
      {
...
        "EniIp": "10.0.1.6",
        "EniIPv6": "2600:113f:404:be09:310e:ed34:3412:f733",
...
      }
    ]
  }
}
```

Untuk informasi selengkapnya tentang bootstrapping, lihat [Alat Konfigurasi](#).

Untuk melakukan bootstrap Klien SDK 5

PKCS #11 library

Untuk melakukan bootstrap instans EC2 Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 -a <HSM IP addresses>
```

Untuk melakukan bootstrap instans EC2 Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" -a <HSM IP addresses>
```

OpenSSL Dynamic Engine

Untuk melakukan bootstrap instans EC2 Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda.

```
$ sudo /opt/cloudhsm/bin/configure-dyn -a <HSM IP addresses>
```

OpenSSL Dynamic Engine Provider

Untuk melakukan bootstrap instans EC2 Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda.

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider -a <HSM IP addresses>
```

Key Storage Provider (KSP)

Untuk melakukan bootstrap instans EC2 Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" -a <HSM IP addresses>
```

JCE provider

Untuk melakukan bootstrap instans EC2 Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda.

```
$ sudo /opt/cloudhsm/bin/configure-jce -a <HSM IP addresses>
```

Untuk melakukan bootstrap instans EC2 Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" -a <HSM IP addresses>
```

CloudHSM CLI

Untuk melakukan bootstrap instans EC2 Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM (s) di cluster Anda.

```
$ sudo /opt/cloudhsm/bin/configure-cli -a <The ENI IPv4 / IPv6 addresses of the HSMs>
```

Untuk melakukan bootstrap instans EC2 Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM (s) di cluster Anda.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" -a <The ENI  
IPv4 / IPv6 addresses of the HSMs>
```

Note

Anda dapat menggunakan `--cluster-id` parameter sebagai pengganti-a `<HSM_IP_ADDRESSES>`. Untuk melihat persyaratan penggunaan `--cluster-id`, lihat [AWS CloudHSM Alat konfigurasi SDK 5 klien](#).

Untuk melakukan bootstrap Klien SDK 3

Untuk melakukan bootstrap instans EC2 Linux untuk Klien SDK 3

- Gunakan `configure` untuk menentukan alamat IP HSM di cluster Anda.

```
sudo /opt/cloudhsm/bin/configure -a <IP address>
```

Untuk melakukan bootstrap instans EC2 Windows untuk Klien SDK 3

- Gunakan `configure` untuk menentukan alamat IP HSM di cluster Anda.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" -a <HSM IP address>
```

Untuk informasi lebih lanjut tentang konfigurasi, lihat [???](#).

Penskalaan HSMs dalam sebuah cluster AWS CloudHSM

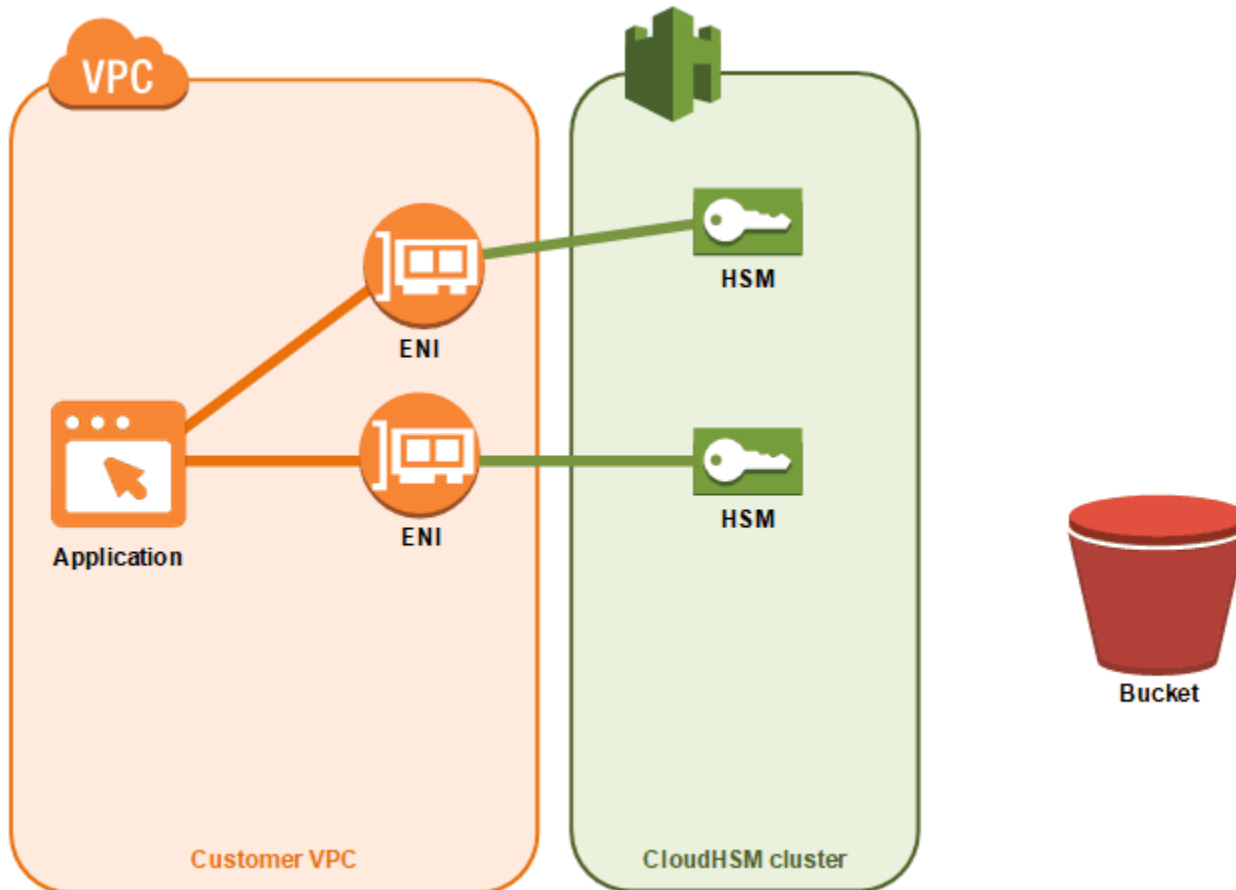
Untuk meningkatkan atau menurunkan AWS CloudHSM klaster Anda, tambahkan atau hapus HSMs dengan menggunakan [AWS CloudHSM konsol](#) atau salah satu [alat baris perintah AWS SDKs atau](#). Kami merekomendasikan pengujian beban klaster Anda untuk menentukan beban puncak yang harus Anda antisipasi, dan kemudian menambahkan satu HSM lagi untuk memastikan ketersediaan tinggi.

Topik

- [Menambahkan HSM ke cluster AWS CloudHSM](#)
- [Menghapus HSM dari sebuah cluster AWS CloudHSM](#)

Menambahkan HSM ke cluster AWS CloudHSM

Gambar berikut menggambarkan peristiwa yang terjadi ketika Anda menambahkan HSM ke klaster.



1. Anda menambahkan HSM baru ke sebuah klaster. Prosedur berikut menjelaskan cara melakukannya dari [konsol AWS CloudHSM](#), [AWS Command Line Interface \(AWS CLI\)](#), dan [API AWS CloudHSM](#).

Ini adalah satu-satunya tindakan yang Anda ambil. Peristiwa yang tersisa terjadi secara otomatis.

2. AWS CloudHSM membuat salinan cadangan dari HSM yang ada di cluster. Untuk informasi selengkapnya, lihat [Pencadangan](#).
3. AWS CloudHSM mengembalikan cadangan ke HSM baru. Hal ini memastikan bahwa HSM sinkron dengan yang lain di klaster.
4. Yang ada HSMs di cluster memberi tahu AWS CloudHSM klien bahwa ada HSM baru di cluster.

5. Klien menetapkan koneksi ke HSM baru.

Untuk menambahkan HSM (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Pilih sebuah klaster untuk HSM yang Anda tambahkan.
3. Pada HSMstab, pilih Buat HSM.
4. Pilih Availability Zone (AZ) untuk HSM yang Anda buat. Lalu, pilih Buat.

Untuk menambahkan HSM (AWS CLI)

- Pada jendela perintah, keluarkan perintah [create-hsm](#), menentukan ID klaster dan Availability Zone untuk HSM yang Anda buat. Jika Anda tidak tahu ID klaster dari klaster pilihan Anda, keluarkan perintah [describe-clusters](#). Tentukan Availability Zone dalam bentuk us-east-2a, us-east-2b, dll.

```
$ aws cloudhsmv2 create-hsm --cluster-id <cluster ID> --availability-  
zone <Availability Zone>  
{  
  "Hsm": {  
    "State": "CREATE_IN_PROGRESS",  
    "ClusterId": "cluster-5a73d5qzrdh",  
    "HsmId": "hsm-lgavqitns2a",  
    "SubnetId": "subnet-0e358c43",  
    "AvailabilityZone": "us-east-2c",  
    "EniId": "eni-bab18892",  
    "EniIp": "10.0.3.10",  
    "EniIpV6": "2600:113f:404:be09:310e:ed34:3412:f733"  
  }  
}
```

Untuk menambahkan HSM (AWS CloudHSM API)

- Kirim permintaan [CreateHsm](#), menentukan ID klaster dan Availability Zone untuk HSM yang Anda buat.

Menghapus HSM dari sebuah cluster AWS CloudHSM

Anda dapat menghapus HSM dengan menggunakan [AWS CloudHSM konsol](#), the [AWS CLI](#), atau AWS CloudHSM API.

Untuk menghapus HSM (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Pilih klaster yang berisi HSM yang Anda hapus.
3. Pada HSMstab, pilih HSM yang Anda hapus. Kemudian, pilih Hapus HSM.
4. Konfirmasi bahwa Anda ingin menghapus HSM. Kemudian, pilih Hapus.

Untuk menghapus HSM (AWS CLI)

- Pada jendela perintah, keluarkan perintah [delete-hsm](#). Lewatkan ID klaster yang berisi HSM yang Anda hapus dan salah satu pengenalan HSM berikut:
 - ID HSM (`--hsm-id`)
 - Alamat IP HSM (`--eni-ip`)
 - ID antarmuka jaringan elastis HSM (`--eni-id`)

Jika Anda tidak tahu nilai untuk pengenalan ini, keluarkan perintah [describe-clusters](#).

```
$ aws cloudhsmv2 delete-hsm --cluster-id <cluster ID> --eni-ip <HSM IP address>
{
  "HsmId": "hsm-lgavqitns2a"
}
```

Untuk menghapus HSM (AWS CloudHSM API)

- Kirim permintaan [DeleteHsm](#), menentukan ID klaster dan pengenalan untuk HSM yang Anda hapus.

Menghapus sebuah cluster AWS CloudHSM

Sebelum Anda dapat menghapus cluster, Anda harus menghapus semua HSMs dari cluster. Untuk informasi selengkapnya, lihat [Menghapus HSM dari sebuah cluster AWS CloudHSM](#).

Setelah menghapus semua HSMs, Anda dapat menghapus klaster dengan menggunakan [AWS CloudHSM konsol](#), [AWS Command Line Interface \(AWS CLI\)](#), atau AWS CloudHSM API.

Untuk menghapus sebuah klaster (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Pilih klaster yang Anda hapus. Kemudian pilih Hapus klaster.
3. Konfirmasi bahwa Anda ingin menghapus klaster, lalu pilih Hapus.

Untuk menghapus klaster (AWS CLI)

- Pada prompt perintah, keluarkan perintah [delete-cluster](#), melewati ID klaster yang Anda hapus. Jika Anda tidak tahu ID klaster, keluarkan perintah [describe-clusters](#).

```
$ aws cloudhsmv2 delete-cluster --cluster-id <cluster ID>
{
  "Cluster": {
    "Certificates": {
      "ClusterCertificate": "<certificate string>"
    },
    "SourceBackupId": "backup-rtq2dwi2gq6",
    "SecurityGroup": "sg-40399d28",
    "CreateTimestamp": 1504903546.035,
    "SubnetMapping": {
      "us-east-2a": "subnet-f1d6e798",
      "us-east-2c": "subnet-0e358c43",
      "us-east-2b": "subnet-40ed9d3b"
    },
    "ClusterId": "cluster-kdmrayrc7gi",
    "VpcId": "vpc-641d3c0d",
    "State": "DELETE_IN_PROGRESS",
    "HsmType": "hsm1.medium",
    "StateMessage": "The cluster is being deleted.",
    "Hsms": [],
    "BackupPolicy": "DEFAULT"
  }
}
```

```
}
```

Untuk menghapus klaster (AWS CloudHSM API)

- Kirim permintaan [DeleteCluster](#), menentukan ID klaster yang Anda hapus.

Membuat AWS CloudHSM cluster dari backup

Untuk memulihkan AWS CloudHSM cluster dari cadangan, ikuti langkah-langkah dalam topik ini. Cluster Anda akan berisi pengguna, materi kunci, sertifikat, konfigurasi, dan kebijakan yang sama yang ada di cadangan. Untuk informasi lebih lanjut tentang mengelola cadangan, lihat [Cadangan cluster](#).

Buat cluster dari cadangan (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Pilih Buat klaster.
3. Di bagian Konfigurasi klaster, lakukan hal berikut:
 - a. Untuk VPC, pilih VPC untuk klaster yang Anda buat.
 - b. Untuk AZ, pilih subnet privat untuk setiap Availability Zone yang Anda tambahkan ke klaster.
 - c. Untuk jenis Jaringan, pilih protokol IP yang HSMs akan Anda gunakan untuk koneksi.
4. Di bagian Sumber klaster, lakukan hal berikut:
 - a. Pilih Kembalikan klaster dari cadangan yang ada.
 - b. Pilih cadangan yang Anda pulihkan.
5. Pilih Selanjutnya: Tinjau.
6. Tinjau konfigurasi klaster Anda, lalu pilih Buat klaster.
7. Tentukan berapa lama layanan harus mempertahankan cadangan.

Terima periode retensi default 90 hari atau ketik nilai baru antara 7 dan 379 hari. Layanan akan secara otomatis menghapus cadangan di klaster ini yang lebih tua dari nilai yang Anda tentukan di sini. Anda dapat mengubah ini nanti. Untuk informasi lebih lanjut, lihat [Konfigurasi retensi cadangan](#).

8. Pilih Selanjutnya.

9. (Opsional) Ketik kunci tanda dan nilai tanda opsional. Untuk menambahkan lebih dari satu tanda ke klaster, pilih Tambahkan tanda.
10. Pilih Tinjau.
11. Tinjau konfigurasi klaster Anda, dan kemudian pilih Buat klaster.

Tip

Untuk membuat HSM di klaster ini yang berisi pengguna, materi kunci, sertifikat, konfigurasi, dan kebijakan yang sama yang ada di cadangan yang Anda pulihkan, [tambahkan HSM](#) ke cluster.

Buat cluster dari backup (AWS CLI)

Untuk menentukan ID cadangan, keluarkan perintah [describe-backups](#).

- Pada jendela perintah, keluarkan perintah [create-cluster](#). Tentukan jenis instans HSM, subnet IDs dari subnet tempat Anda berencana untuk membuat HSMs, dan ID cadangan cadangan yang Anda pulihkan.

```
$ aws cloudhsmv2 create-cluster --hsm-type hsm2m.medium \
                                --subnet-ids <subnet ID 1> <subnet ID 2> <subnet ID
N> \
                                --source-backup-id <backup ID>
                                --mode <FIPS> \
                                --network-type <IPV4>
{
  "Cluster": {
    "HsmType": "hsm2m.medium",
    "VpcId": "vpc-641d3c0d",
    "Hsms": [],
    "State": "CREATE_IN_PROGRESS",
    "SourceBackupId": "backup-rtq2dwi2gq6",
    "BackupPolicy": "DEFAULT",
    "BackupRetentionPolicy": {
      "Type": "DAYS",
      "Value": 90
    },
    "NetworkType": "IPV4",
    "SecurityGroup": "sg-640fab0c",
```

```

    "CreateTimestamp": 1504907311.112,
    "SubnetMapping": {
      "us-east-2c": "subnet-0e358c43",
      "us-east-2a": "subnet-f1d6e798",
      "us-east-2b": "subnet-40ed9d3b"
    },
    "Certificates": {
      "ClusterCertificate": "<certificate string>"
    },
    "ClusterId": "cluster-jxhlf7644ne"
  }
}

```

Buat cluster dari backup (API)AWS CloudHSM

Lihat topik berikut untuk mempelajari cara membuat klaster dari cadangan dengan menggunakan API.

- [CreateCluster](#)

Migrasi tipe HSM cluster

AWS CloudHSM menawarkan kemampuan untuk mengubah tipe HSM dari cluster yang ada. Tinjau tabel di halaman ini untuk menentukan apakah modifikasi tipe HSM diperbolehkan.

Untuk informasi lebih lanjut tentang jenis yang HSMs didukung dan fitur-fiturnya, silakan merujuk ke [Jenis HSM di AWS CloudHSM](#).

Note

Anda tidak dapat mengubah mode FIPS cluster selama operasi ini.

Dari	Ke	Komentar
hsm1.sedang	hsm2m.sedang	Diizinkan
hsm2m.sedang	hsm1.sedang	Bersyarat. Anda dapat memutar kembali dari

Dari	Ke	Komentar
		hsm2m.medium ke hsm1.medium dalam waktu 24 jam sejak dimulainya migrasi.

Topik

- [Migrasi dari hsm1.medium ke hsm2m.medium](#)

Migrasi dari hsm1.medium ke hsm2m.medium

Anda dapat memigrasikan AWS CloudHSM cluster Anda dari hsm1.medium ke hsm2m.medium. Topik ini menjelaskan prasyarat, proses migrasi, dan prosedur rollback.

Sebelum memulai migrasi, pastikan aplikasi Anda mengikuti rekomendasi di [Arsitek cluster Anda untuk ketersediaan tinggi](#). Ini membantu menghindari downtime selama proses berlangsung.

Note

Migrasi otomatis ke hsm2m.medium akan dimulai pada 20 Januari 2026.

Ikhtisar proses migrasi hsm1.medium ke hsm2m.medium

Anda dapat memulai migrasi menggunakan AWS CloudHSM Console, the AWS CLI, atau AWS CloudHSM API. Di mana pun Anda memulainya, migrasi AWS CloudHSM cluster menggunakan titik akhir `modify-cluster` API. Atau, AWS CloudHSM akan secara otomatis memigrasikan klaster atas nama Anda. Setelah migrasi dimulai, seluruh klaster Anda memasuki mode penulisan terbatas. Untuk informasi selengkapnya, lihat [Mode penulisan terbatas klaster](#).

Untuk meminimalkan dampak, AWS CloudHSM perubahan HSMs dari hsm1.medium ke hsm2m.medium satu per satu. Pengganti HSMs mempertahankan alamat IP yang sama, sehingga tidak memerlukan perubahan konfigurasi selama atau setelah migrasi.

Begini cara kerja migrasi:

1. Sebelum memigrasikan HSM pertama, AWS CloudHSM buat cadangan lengkap dari seluruh cluster.

2. Menggunakan cadangan ini, AWS CloudHSM membuat HSM baru dari jenis yang diminta (hsm2m.medium) untuk menggantikan HSM pertama.
3. Sebelum memigrasi setiap HSM berikutnya, AWS CloudHSM buat cadangan lengkap baru dari seluruh cluster.
4. AWS CloudHSM mengulangi langkah 3 dan 4 untuk setiap HSM di cluster, memigrasikan satu HSM pada satu waktu.
5. Setiap migrasi HSM individu membutuhkan waktu sekitar 30 menit.

AWS CloudHSM memantau kesehatan klaster dan melakukan validasi selama proses migrasi. Jika AWS CloudHSM mendeteksi peningkatan kesalahan atau pemeriksaan validasi gagal, maka secara otomatis akan menghentikan migrasi dan mengembalikan cluster ke tipe HSM aslinya. Anda juga dapat memutar kembali secara manual hingga 24 jam setelah memulai migrasi. Sebelum memutar kembali, lihat pertimbangan [rollback tipe HSM](#).

Prasyarat untuk bermigrasi ke hsm2m.medium

AWS CloudHSM Cluster Anda yang ada harus memenuhi persyaratan ini untuk bermigrasi ke hsm2m.medium. Jika kondisi apa pun tidak terpenuhi selama pemeriksaan validasi, AWS CloudHSM secara otomatis mengembalikan cluster ke tipe HSM aslinya.

Untuk daftar masalah migrasi yang diketahui, lihat [???](#)

- Dalam 7 hari terakhir:
 - Semua koneksi klien telah menggunakan SDK 5.9 atau lebih tinggi.
 - Jika melakukan Verifikasi ECDSA, semua koneksi klien telah menggunakan SDK 5.13 atau lebih tinggi.
 - AWS CloudHSM instance hanya menggunakan fungsionalitas yang didukung (dan tidak ada yang tidak digunakan lagi). Lihat [Pemberitahuan penghentian](#) untuk detailnya.
 - Anda harus telah menggunakan SDK untuk terhubung dengan setidaknya satu HSM di cluster dalam 7 hari terakhir.
- Cluster berada dalam keadaan AKTIF.
- Cluster memiliki 27 HSMs atau kurang.
- Tingkat kesalahan untuk operasi HSM tidak meningkat selama migrasi.

 Note

Pembatasan sebelumnya yang mencegah pelanggan dengan beban kerja kunci token dari migrasi telah dihapus.

Mode tulis terbatas cluster

Saat klaster Anda memulai migrasi, klaster memasuki mode penulisan terbatas. Operasi yang dapat mengubah status HSM ditolak. Semua operasi baca tetap tidak terpengaruh.

Selama migrasi, aplikasi Anda menerima kesalahan dari HSM saat mencoba operasi ini:

- Pembuatan dan penghapusan kunci token (beban kerja kunci sesi terus beroperasi).
- Semua pembuatan, penghapusan, atau modifikasi pengguna.
- Operasi kuorum.
- Modifikasi kunci dalam HSM, seperti mengubah atribut kunci.
- Pendaftaran mTLS.

AWS CloudHSM juga menempatkan klaster Anda dalam `MODIFY_IN_PROGRESS` status selama migrasi. Selama waktu ini, Anda tidak dapat menambah atau menghapus HSMs dari cluster.

Memulai migrasi

Proses migrasi klaster menggantikan individu HSMs di klaster Anda satu per satu. Durasi tergantung pada jumlah HSMs di cluster Anda. Rata-rata, proses ini memakan waktu sekitar 30 menit per HSM. Anda dapat melacak kemajuan dengan memantau tipe individu HSM HSMs di cluster untuk melihat berapa banyak yang telah dimigrasi ke tipe baru.

Console

Untuk mengubah tipe HSM (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Pilih tombol radio di sebelah ID cluster yang ingin Anda ubah
3. Dari menu Tindakan, pilih `Modify HSM Type` dan pilih jenis HSM yang diinginkan

Prosedur ini menempatkan cluster Anda ke dalam MODIFY_IN_PROGRESS status. Setelah migrasi, klaster Anda kembali ke ACTIVE status.

AWS CLI

Untuk mengubah tipe HSM () [AWS CLI](#)

- Pada jendela perintah, jalankan perintah [modify-cluster](#). Tentukan ID cluster dan tipe HSM yang diinginkan.

```
$ aws cloudhsmv2 modify-cluster --cluster-id <cluster ID> --hsm-type <HSM Type>

{
  "Cluster": {
    "BackupPolicy": "DEFAULT",
    "BackupRetentionPolicy": {
      "Type": "DAYS",
      "Value": 90
    },
    "VpcId": "vpc-50ae0636",
    "SubnetMapping": {
      "us-west-2b": "subnet-49a1bc00",
      "us-west-2c": "subnet-6f950334",
      "us-west-2a": "subnet-fd54af9b"
    },
    "SecurityGroup": "sg-6cb2c216",
    "HsmType": "hsm2m.medium",
    "HsmTypeRollbackExpiration": 1730383180.000,
    "Certificates": {},
    "State": "MODIFY_IN_PROGRESS",
    "Hsms": [],
    "ClusterId": "cluster-igklspoyj5v",
    "ClusterMode": "FIPS",
    "CreateTimestamp": 1502423370.069
  }
}
```

Prosedur ini menempatkan cluster Anda ke dalam MODIFY_IN_PROGRESS status. Setelah migrasi, klaster Anda kembali ke ACTIVE status.

AWS CloudHSM API

Untuk mengubah tipe HSM (AWS CloudHSM API)

- Kirim permintaan [ModifyCluster](#). Tentukan ID cluster dan tipe HSM yang diinginkan untuk cluster.

Prosedur ini menempatkan cluster Anda ke dalam MODIFY_IN_PROGRESS status. Setelah migrasi, klaster Anda kembali ke ACTIVE status.

Mengembalikan migrasi

AWS CloudHSM memantau tingkat kesalahan yang meningkat dan melakukan pemeriksaan validasi berkelanjutan selama migrasi. Jika AWS CloudHSM mendeteksi penurunan kualitas layanan atau kegagalan validasi apa pun, secara otomatis akan memulai rollback ke tipe HSM asli cluster Anda. Selama rollback, untuk setiap HSM di cluster:

- AWS CloudHSM menggunakan cadangan yang diambil pada awal migrasi HSM itu.
- Ini menggantikan satu HSM pada satu waktu sampai semua HSMs dikembalikan ke tipe aslinya.
- Cluster Anda tetap dalam mode penulisan terbatas selama proses berlangsung.

Anda dapat memutar kembali migrasi dalam waktu 24 jam setelah memulainya. Untuk memeriksa tenggat waktu rollback:

1. Jalankan [perintah deskripsi-cluster](#).
2. Cari `HsmTypeRollbackExpiration` nilainya. Stempel waktu ini adalah batas waktu rollback Anda.

Jika Anda memutuskan untuk memutar kembali, lakukan sebelum batas waktu ini. Rollback menggunakan cadangan terbaru dari tipe HSM asli Anda.

Warning

Berhati-hatilah untuk mundur setelah migrasi selesai. Jika Anda menyelesaikan migrasi dan kemudian digunakan AWS CloudHSM untuk membuat kunci atau pengguna baru, memutar

kembali dapat mengakibatkan hilangnya data. Lihat [Menyinkronkan Data Setelah Rollback](#) untuk mempelajari cara mengurangi kehilangan data setelah rollback.

Console

Untuk memutar kembali tipe HSM Anda (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Pilih ID cluster yang ingin Anda putar kembali.
3. Dari menu Actions, pilih Modify HSM Type dan pilih tipe HSM asli

Prosedur ini menempatkan cluster Anda ke dalam ROLLBACK_IN_PROGRESS status. Setelah rollback, cluster Anda kembali ke status. ACTIVE

AWS CLI

Untuk memutar kembali tipe HSM Anda () [AWS CLI](#)

- Pada jendela perintah, jalankan perintah [modify-cluster](#). Tentukan ID cluster dan tipe HSM asli.

```
$ aws cloudhsmv2 modify-cluster --cluster-id <cluster ID> --hsm-type <HSM Type>

{
  "Cluster": {
    "BackupPolicy": "DEFAULT",
    "BackupRetentionPolicy": {
      "Type": "DAYS",
      "Value": 90
    },
    "VpcId": "vpc-50ae0636",
    "SubnetMapping": {
      "us-west-2b": "subnet-49a1bc00",
      "us-west-2c": "subnet-6f950334",
      "us-west-2a": "subnet-fd54af9b"
    },
    "SecurityGroup": "sg-6cb2c216",
    "HsmType": "hsm1.medium",
    "HsmTypeRollbackExpiration": 1730383180.000,
    "Certificates": {},
  }
}
```

```
"State": "ROLLBACK_IN_PROGRESS",
"Hsms": [],
"ClusterId": "cluster-igklspoyj5v",
"ClusterMode": "FIPS",
"CreateTimestamp": 1502423370.069
}
}
```

Prosedur ini menempatkan cluster Anda ke dalam ROLLBACK_IN_PROGRESS status. Setelah rollback, cluster Anda kembali ke status. ACTIVE

AWS CloudHSM API

Untuk memutar kembali tipe HSM Anda (AWS CloudHSM API)

- Kirim permintaan [ModifyCluster](#). Tentukan ID cluster dan tipe HSM asli untuk cluster.

Prosedur ini menempatkan cluster Anda ke dalam ROLLBACK_IN_PROGRESS status. Setelah rollback, cluster Anda kembali ke status. ACTIVE

Menyinkronkan data setelah rollback

Selama migrasi, HSMs berada dalam mode tulis terbatas, mencegah perubahan status HSM. Jika Anda memutar kembali selama waktu ini (saat cluster berada MODIFY_IN_PROGRESS), itu menghasilkan cluster dengan konten yang identik dengan cluster asli.

Setelah cluster Anda kembali ke ACTIVE status, mode penulisan terbatas diangkat. Jika Anda membuat kunci atau pengguna saat dalam ACTIVE status dan kemudian memutar kembali, kunci atau pengguna itu tidak akan ada di cluster yang digulung kembali.

Untuk mengatasi sinkronisasi pengguna, gunakan [manajemen pengguna](#) dengan CloudHSM CLI untuk membuat ulang pengguna yang hilang di cluster mundur Anda. Pengguna harus dibuat ulang secara manual karena `user replicate` perintah tidak mendukung sinkronisasi pengguna dari `hsm2m.medium` ke `hsm1.medium`. Lihat pengguna mereplikasi [masalah yang diketahui](#).

Untuk mengatasi sinkronisasi kunci, gunakan perintah [replika kunci untuk mereplikasi](#) kunci antara dua cluster. Jika Anda belum instal CloudHSM CLI, lihat instruksi di. [???](#)

Untuk menyinkronkan kunci setelah rollback

Ikuti langkah-langkah ini setelah menyelesaikan rollback. Kami akan menggunakan istilah-istilah ini:

- “cluster-1”: Cluster Anda yang digulung kembali (sekarang hsm1.medium)
- “cluster-2”: Cluster hsm2m.medium sementara baru yang akan Anda buat

1. Buat cluster hsm2m.medium baru (cluster-2) menggunakan cadangan hsm2m.medium terbaru dari cluster-1:

```
aws cloudhsmv2 create-cluster --hsm-type hsm2m.medium \
                               --subnet-ids <subnet ID 1> <subnet ID 2> <subnet ID
N> \
                               --source-backup-id <backup ID>
                               --mode <FIPS>
```

2. Buat HSM di cluster-2:

```
aws cloudhsmv2 create-hsm --cluster-id <cluster-2 ID>
```

3. Daftar kunci di cluster-2 yang membutuhkan replikasi:

```
cloudhsm-cli key list --cluster-id <cluster-2 ID>
```

4. Replikasi setiap kunci dari cluster-2 ke cluster-1:

```
cloudhsm-cli key replicate --source-cluster-id <cluster-2 ID> \
                           --destination-cluster-id <cluster-1 ID> \
                           --filter attr.label=<key ID>
```

5. Ulangi langkah 4 untuk setiap kunci yang perlu disalin.
6. Hapus HSM di cluster-2:

```
aws cloudhsmv2 delete-hsm --cluster-id <cluster-2 ID> --hsm-id <HSM ID>
```

7. Hapus cluster-2:

```
aws cloudhsmv2 delete-cluster --cluster-id <cluster-2 ID>
```

Pengguna HSM di AWS CloudHSM

Sebelum Anda dapat menggunakan AWS CloudHSM cluster Anda untuk cryptoprocessing, Anda harus membuat pengguna dan [kunci](#) pada modul keamanan perangkat keras (HSM) di cluster Anda.

Note

Pengguna HSM berbeda dari pengguna IAM. Pengguna IAM yang memiliki kredensial yang benar dapat membuat HSMs dengan berinteraksi dengan sumber daya melalui AWS API. Setelah HSM dibuat, Anda harus menggunakan kredensial pengguna HSM untuk mengotentikasi operasi pada HSM.

Di AWS CloudHSM, Anda harus menggunakan alat baris perintah [CloudHSM CLI](#) atau [CloudHSM Management Utility](#) (CMU) untuk membuat dan mengelola pengguna di HSM Anda. [CloudHSM CLI dirancang untuk digunakan dengan seri versi SDK terbaru, sedangkan CMU dirancang untuk digunakan dengan seri versi SDK sebelumnya.](#)

Lihat topik berikut untuk informasi selengkapnya tentang mengelola pengguna HSM di AWS CloudHSM. Anda juga dapat mempelajari cara menggunakan autentikasi kuorum (juga dikenal sebagai M dari N kontrol akses).

Topik

- [Manajemen pengguna HSM dengan CloudHSM CLI](#)
- [Manajemen pengguna HSM dengan CloudHSM Management Utility \(CMU\)](#)

Manajemen pengguna HSM dengan CloudHSM CLI

[Untuk mengelola pengguna modul keamanan perangkat keras \(HSM\) AWS CloudHSM, Anda harus masuk ke HSM dengan nama pengguna dan kata sandi admin.](#) Hanya admin yang dapat mengelola pengguna. HSM berisi admin default bernama admin. Anda mengatur kata sandi untuk admin ketika Anda [mengaktifkan klaster](#).

Topik ini memberikan step-by-step instruksi dan detail tentang mengelola pengguna HSM dengan CloudHSM CLI.

Topik

- [Prasyarat untuk manajemen pengguna di CloudHSM CLI](#)
- [Jenis pengguna HSM untuk CloudHSM CLI](#)
- [Tabel izin pengguna HSM untuk CloudHSM CLI](#)
- [Buat admin pengguna HSM menggunakan CloudHSM CLI](#)
- [Buat pengguna kriptografi HSM menggunakan CloudHSM CLI](#)
- [Daftar semua pengguna HSM di cluster menggunakan CloudHSM CLI](#)
- [Ubah kata sandi pengguna HSM menggunakan CloudHSM CLI](#)
- [Hapus pengguna HSM menggunakan CloudHSM CLI](#)
- [Kelola MFA untuk pengguna HSM menggunakan CloudHSM CLI](#)
- [Kelola otentikasi kuorum \(M of N access control\) menggunakan CloudHSM CLI](#)

Prasyarat untuk manajemen pengguna di CloudHSM CLI

Sebelum Anda menggunakan CloudHSM CLI untuk mengelola AWS CloudHSM pengguna modul keamanan perangkat keras (HSM), Anda harus menyelesaikan prasyarat ini. Topik berikut menjelaskan memulai dengan CloudHSM CLI.

Topik

- [Dapatkan alamat IP HSM di AWS CloudHSM](#)
- [Unduh CloudHSM CLI](#)

Dapatkan alamat IP HSM di AWS CloudHSM

Untuk menggunakan CloudHSM CLI, Anda harus menggunakan alat konfigurasi untuk memperbarui konfigurasi lokal. Untuk petunjuk tentang menjalankan alat konfigurasi dengan CloudHSM CLI, lihat [Memulai dengan AWS CloudHSM Command Line Interface \(CLI\)](#). Parameter `-a` mengharuskan Anda untuk menambahkan alamat IP HSM di kluster Anda. Jika Anda memiliki beberapa HSMs, Anda dapat menggunakan alamat IP apa pun. Ini memastikan CloudHSM CLI dapat menyebarkan perubahan apa pun yang Anda buat di seluruh cluster. Ingat bahwa CloudHSM CLI menggunakan file lokalnya untuk melacak informasi cluster. Jika kluster telah berubah sejak terakhir kali Anda menggunakan CloudHSM CLI dari host tertentu, Anda harus menambahkan perubahan tersebut ke file konfigurasi lokal yang disimpan di host tersebut. Jangan pernah menghapus HSM saat Anda menggunakan CloudHSM CLI.

Untuk mendapatkan alamat IP untuk HSM (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Untuk mengubah Wilayah AWS, gunakan pemilih Wilayah di sudut kanan atas halaman.
3. Untuk membuka halaman detail klaster, dalam tabel klaster, pilih ID klaster.
4. Untuk mendapatkan alamat IP, buka HSMs tab. Untuk IPv4 cluster, pilih alamat yang tercantum di bawah IPv4 alamat ENI. Untuk cluster dual-stack gunakan ENI IPv4 atau alamat ENI. IPv6

Untuk mendapatkan alamat IP untuk HSM (AWS CLI)

- Dapatkan alamat IP dari HSM dengan menggunakan perintah [describe-clusters](#) dari AWS CLI. Dalam output dari perintah, alamat IP dari HSMs adalah nilai-nilai `EniIp` dan `EniIPv6` (jika itu adalah cluster dual-stack).

```
$ aws cloudhsmv2 describe-clusters
{
  "Clusters": [
    { ... }
    "Hsms": [
      {
...
        "EniIp": "10.0.0.9",
...
      },
      {
...
        "EniIp": "10.0.1.6",
        "EniIPv6": "2600:113f:404:be09:310e:ed34:3412:f733",
...
      }
    ]
  }
}
```

Unduh CloudHSM CLI

Versi terbaru CloudHSM CLI tersedia untuk tugas manajemen pengguna HSM untuk Client SDK

5. Untuk mengunduh dan menginstal CloudHSM CLI, ikuti petunjuk di Instal [dan](#) konfigurasi CloudHSM CLI.

Jenis pengguna HSM untuk CloudHSM CLI

Sebagian besar operasi yang Anda lakukan pada modul keamanan perangkat keras (HSM) memerlukan kredensial pengguna HSM. AWS CloudHSM HSM mengautentikasi setiap pengguna HSM dan setiap pengguna HSM memiliki jenis yang menentukan operasi yang dapat Anda lakukan pada HSM sebagai pengguna tersebut.

Note

Pengguna HSM berbeda dari pengguna IAM. Pengguna IAM yang memiliki kredensial yang benar dapat membuat HSMs dengan berinteraksi dengan sumber daya melalui AWS API. Setelah HSM dibuat, Anda harus menggunakan kredensi pengguna HSM untuk mengautentikasi operasi pada HSM.

Jenis pengguna

- [Admin yang tidak aktif](#)
- [Admin](#)
- [Pengguna kriptografi \(CU\)](#)
- [Pengguna alat \(AU\)](#)

Admin yang tidak aktif

Di CloudHSM CLI, admin yang tidak aktif adalah pengguna sementara yang hanya ada pada HSM AWS CloudHSM pertama di cluster yang belum pernah diaktifkan. Untuk [mengaktifkan cluster](#), jalankan `cluster activate` perintah di CloudHSM CLI. Setelah menjalankan perintah ini, admin yang tidak aktif diminta untuk mengubah kata sandi. Setelah mengubah kata sandi, admin yang tidak aktif menjadi admin.

Admin

Di CloudHSM CLI, admin dapat melakukan operasi manajemen pengguna. Misalnya, mereka dapat membuat dan menghapus pengguna dan mengubah kata sandi pengguna. Untuk informasi selengkapnya tentang admin, lihat. [Tabel izin pengguna HSM untuk CloudHSM CLI](#)

Pengguna kriptografi (CU)

Pengguna kriptografi (CU) dapat melakukan manajemen kunci dan operasi kriptografi berikut.

- Manajemen kunci— Buat, hapus, bagikan, impor, dan ekspor kunci kriptografi.
- Operasi kriptografi— Gunakan kunci kriptografi untuk enkripsi, dekripsi, penandatanganan, verifikasi, dan lainnya.

Untuk informasi selengkapnya, lihat [Tabel izin pengguna HSM untuk CloudHSM CLI](#).

Pengguna alat (AU)

Pengguna alat (AU) dapat melakukan operasi kloning dan sinkronisasi pada kluster Anda. HSMs AWS CloudHSM menggunakan AU untuk menyinkronkan HSMs dalam sebuah AWS CloudHSM cluster. AU ada pada semua yang HSMs disediakan oleh AWS CloudHSM, dan memiliki izin terbatas. Untuk informasi selengkapnya, lihat [Tabel izin pengguna HSM untuk CloudHSM CLI](#).

AWS tidak dapat melakukan operasi apa pun pada Anda HSMs . AWS tidak dapat melihat atau memodifikasi pengguna atau kunci Anda dan tidak dapat melakukan operasi kriptografi apa pun menggunakan kunci tersebut.

Tabel izin pengguna HSM untuk CloudHSM CLI

Tabel berikut mencantumkan operasi modul keamanan perangkat keras (HSM) yang diurutkan berdasarkan jenis pengguna HSM atau sesi yang dapat melakukan operasi di. AWS CloudHSM

	Admin	Pengguna Kripto (CU)	Pengguna Peralatan (AU)	Sesi Tidak Diautentikasi	
Dapatkan informasi kluster dasar ¹		Y 	Y 	Y 	Ya
Ubah kata sandi sendiri		Y 	Y 	Y	Tidak berlaku
Ubah kata sandi pengguna		Y 	T 	T 	Tidak

	Admin	Pengguna Kripto (CU)	Pengguna Peralatan (AU)	Sesi Tidak Diautentikasi	
Tambah, hapus pengguna		Y	T		Tidak
Dapatkan status sinkronis ²		Y	Y		Tidak
Ekstrak, masukkan benda bertopeng ³		Y	Y		Tidak
Fungsi manajemen kunci		T	Y		Tidak
Enkripsi, dekripsi		T	Y		Tidak
Tanda tangan, verifikasi		T	Y		Tidak
Hasilkan intisari dan HMACs		T	Y		Tidak

- [1] Informasi cluster dasar mencakup jumlah HSMs di cluster dan setiap alamat IP HSM, model, nomor seri, ID perangkat, ID firmware, dll.
- [2] Pengguna bisa mendapatkan satu set intisari (hash) yang sesuai dengan kunci pada HSM. Aplikasi dapat membandingkan kumpulan intisari ini untuk memahami status sinkronisasi HSMs dalam sebuah cluster.

- [3] Objek bertopeng adalah kunci yang dienkripsi sebelum meninggalkan HSM. Objek tidak dapat didekripsi di luar HSM. Objek hanya didekripsi setelah dimasukkan ke dalam HSM yang ada di kluster yang sama dengan HSM asalnya diekstraksi. Aplikasi dapat mengekstrak dan menyisipkan objek bertopeng untuk menyinkronkan HSMs dalam sebuah cluster.
- [4] Fungsi manajemen kunci termasuk membuat, menghapus, membungkus, membuka bungkus, dan memodifikasi atribut kunci.

Buat admin pengguna HSM menggunakan CloudHSM CLI

Ikuti langkah-langkah ini untuk membuat pengguna admin modul keamanan perangkat keras (HSM) menggunakan CloudHSM CLI.

1. Gunakan perintah berikut untuk memulai mode interaktif CloudHSM CLI.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Gunakan login perintah dan masuk ke cluster sebagai admin.

```
aws-cloudhsm > login --username <username> --role admin
```

3. Sistem meminta kata sandi Anda. Masukkan kata sandi, dan output menunjukkan bahwa perintah berhasil.

```
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<username>",
    "role": "admin"
  }
}
```

4. Masukkan perintah berikut untuk membuat admin:

```
aws-cloudhsm > user create --username <username> --role admin
```

5. Masukkan kata sandi untuk pengguna baru.
6. Masukkan kembali kata sandi untuk mengonfirmasi kata sandi yang Anda masukkan sudah benar.

Buat pengguna krypto HSM menggunakan CloudHSM CLI

Ikuti langkah-langkah ini untuk membuat pengguna krypto (CU) modul keamanan perangkat keras (HSM) menggunakan CloudHSM CLI.

1. Gunakan perintah berikut untuk memulai mode interaktif CloudHSM CLI.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Gunakan login perintah dan masuk ke cluster sebagai admin.

```
aws-cloudhsm > login --username <username> --role admin
```

3. Sistem meminta kata sandi Anda. Masukkan kata sandi, dan output menunjukkan bahwa perintah berhasil.

```
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<USERNAME>",
    "role": "admin"
  }
}
```

4. Masukkan perintah berikut untuk membuat pengguna krypto:

```
aws-cloudhsm > user create --username <username> --role crypto-user
```

5. Masukkan kata sandi untuk pengguna krypto baru.
6. Masukkan kembali kata sandi untuk mengonfirmasi kata sandi yang Anda masukkan sudah benar.

Daftar semua pengguna HSM di cluster menggunakan CloudHSM CLI

Gunakan `user list` perintah di CloudHSM CLI untuk mencantumkan semua pengguna di cluster. AWS CloudHSM Anda tidak perlu masuk untuk menjalankan `user list`. Semua tipe pengguna dapat mencantumkan pengguna.

Ikuti langkah-langkah ini untuk membuat daftar semua pengguna di cluster

1. Gunakan perintah berikut untuk memulai mode interaktif CloudHSM CLI.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Masukkan perintah berikut untuk mencantumkan semua pengguna di cluster:

```
aws-cloudhsm > user list
```

Untuk informasi selengkapnya `user list`, lihat [daftar pengguna](#).

Ubah kata sandi pengguna HSM menggunakan CloudHSM CLI

Gunakan `user change-password` perintah di CloudHSM CLI untuk mengubah kata sandi pengguna modul keamanan perangkat keras (HSM).

Jenis pengguna dan kata sandi adalah sensitif huruf besar-kecil, tetapi nama pengguna tidak sensitif huruf besar-kecil.

Admin, pengguna kriptografi (CU), dan pengguna alat (AU) dapat mengubah kata sandi mereka sendiri. Untuk mengubah kata sandi pengguna lain, Anda harus masuk sebagai admin. Anda tidak dapat mengubah kata sandi pengguna yang saat ini login ke klien atau `key_mgmt_util`.

Untuk mengubah kata sandi Anda sendiri

1. Gunakan perintah berikut untuk memulai mode interaktif CloudHSM CLI.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Gunakan login perintah dan masuk sebagai pengguna dengan kata sandi yang ingin Anda ubah.

```
aws-cloudhsm > login --username <username> --role <role>
```

3. Masukkan kata sandi pengguna.

```
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<username>",
    "role": "<role>"
  }
}
```

4. Masukkan user change-password perintah.

```
aws-cloudhsm > user change-password --username <username> --role <role>
```

5. Masukkan kata sandi baru.
6. Masukkan kembali kata sandi baru.

: Ubah Kata Sandi Pengguna Lain

1. Gunakan perintah berikut untuk memulai mode interaktif CloudHSM CLI.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Menggunakan CloudHSM CLI, masuk sebagai admin.

```
aws-cloudhsm > login --username <admin> --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<admin>",
    "role": "admin"
  }
}
```

3. Masukkan user change-password perintah bersama dengan nama pengguna pengguna yang kata sandinya ingin Anda ubah.

```
aws-cloudhsm > user change-password --username <username> --role <role>
```

4. Masukkan kata sandi baru.
5. Masukkan kembali kata sandi baru.

Untuk informasi selengkapnya user change-password, lihat [perubahan kata sandi pengguna](#).

Hapus pengguna HSM menggunakan CloudHSM CLI

Gunakan user delete di CloudHSM CLI untuk menghapus pengguna modul keamanan perangkat keras (HSM). Anda harus masuk sebagai admin untuk menghapus pengguna lain.

Tip

Anda tidak dapat menghapus pengguna krypto (CU) yang memiliki kunci.

Untuk menghapus klaster

1. Gunakan perintah berikut untuk memulai mode interaktif CloudHSM CLI.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Gunakan login perintah dan masuk ke cluster sebagai admin.

```
aws-cloudhsm > login --username <username> --role admin
```

3. Sistem meminta kata sandi Anda. Masukkan kata sandi, dan output menunjukkan bahwa perintah berhasil.

```
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<username>",
    "role": "admin"
  }
}
```

4. Gunakan user delete perintah untuk menghapus pengguna.

```
aws-cloudhsm > user delete --username <username> --role <role>
```

Untuk informasi selengkapnya tentang user delete, lihat [deleteUser](#).

Kelola MFA untuk pengguna HSM menggunakan CloudHSM CLI

Untuk meningkatkan keamanan, Anda dapat mengonfigurasi otentikasi multi-faktor (MFA) bagi pengguna untuk membantu melindungi klaster. AWS CloudHSM

Saat Anda masuk ke kluster dengan akun pengguna modul keamanan perangkat keras (HSM) yang diaktifkan MFA, Anda memberikan CloudHSM CLI dengan kata sandi Anda—faktor pertama, apa yang Anda tahu—dan CloudHSM CLI memberi Anda token dan meminta Anda untuk menandatangani token.

Untuk menyediakan faktor kedua—apa yang Anda miliki—Anda tandatangani token dengan kunci privat dari pasangan kunci yang telah Anda buat dan terkait dengan pengguna HSM. Untuk mengakses cluster, Anda memberikan token yang ditandatangani ke CloudHSM CLI.

Untuk informasi selengkapnya tentang pengaturan MFA bagi pengguna, lihat [Mengatur MFA untuk CloudHSM CLI](#)

Topik berikut memberikan informasi lebih lanjut tentang bekerja dengan otentikasi kuorum di. AWS CloudHSM

Topik

- [Otentikasi kuorum dan MFA dalam cluster menggunakan CloudHSM CLI AWS CloudHSM](#)
- [Persyaratan key pair MFA untuk menggunakan AWS CloudHSM CloudHSM CLI](#)
- [Mengatur MFA untuk CloudHSM CLI](#)
- [Buat pengguna dengan MFA diaktifkan untuk CloudHSM CLI](#)
- [Masuk pengguna dengan MFA diaktifkan untuk CloudHSM CLI](#)
- [Putar tombol untuk pengguna dengan MFA diaktifkan untuk CloudHSM CLI](#)
- [Membatalkan pendaftaran kunci publik MFA menggunakan CloudHSM CLI](#)
- [Referensi file token untuk MFA dengan CloudHSM CLI](#)

Otentikasi kuorum dan MFA dalam cluster menggunakan CloudHSM CLI AWS CloudHSM

AWS CloudHSM Cluster menggunakan kunci yang sama untuk otentikasi kuorum dan untuk otentikasi multi-faktor (MFA). Ini berarti pengguna dengan MFA diaktifkan secara efektif terdaftar untuk mOfN atau kontrol akses kuorum. Agar berhasil menggunakan otentikasi MFA dan kuorum untuk pengguna HSM yang sama, pertimbangkan hal-hal berikut:

- Jika Anda menggunakan otentikasi kuorum untuk pengguna hari ini, Anda harus menggunakan key pair yang sama yang Anda buat untuk pengguna kuorum untuk mengaktifkan MFA bagi pengguna.

- Jika Anda menambahkan persyaratan MFA untuk pengguna non-MFA yang bukan pengguna otentikasi kuorum, maka Anda mendaftarkan pengguna tersebut sebagai pengguna terdaftar kuorum (MoFN) dengan otentikasi MFA.
- Jika Anda menghapus persyaratan MFA atau mengubah kata sandi untuk pengguna MFA yang juga merupakan pengguna otentikasi kuorum terdaftar, Anda juga akan menghapus pendaftaran pengguna sebagai pengguna kuorum (MoFN).
- Jika Anda menghapus persyaratan MFA atau mengubah kata sandi untuk pengguna MFA yang juga pengguna otentikasi kuorum, tetapi Anda masih ingin pengguna tersebut berpartisipasi dalam otentikasi kuorum, maka Anda harus mendaftarkan pengguna itu lagi sebagai pengguna Kuorum (MoFN).

Untuk informasi selengkapnya tentang autentikasi kuorum, lihat [Mengelola otentikasi kuorum \(M atau N\)](#).

Persyaratan key pair MFA untuk menggunakan AWS CloudHSM CloudHSM CLI

Untuk mengaktifkan otentikasi multi-faktor (MFA) bagi pengguna modul keamanan perangkat keras (HSM) AWS CloudHSM, Anda dapat membuat key pair baru atau menggunakan kunci yang ada yang memenuhi persyaratan berikut:

- Tipe kunci: Asimetris
- Penggunaan kunci: Tanda tangani dan verifikasi
- Spesifikasi kunci: RSA_2048
- Algoritma penandatanganan meliputi: SHA256with RSAEncryption

Note

Jika Anda menggunakan otentikasi kuorum atau berencana menggunakan otentikasi kuorum, lihat [Otentikasi kuorum dan MFA dalam cluster menggunakan CloudHSM CLI AWS CloudHSM](#)

Anda dapat menggunakan CloudHSM CLI dan key pair untuk membuat pengguna admin baru dengan MFA diaktifkan.

Mengatur MFA untuk CloudHSM CLI

Ikuti langkah-langkah ini untuk menyiapkan otentikasi multi-faktor (MFA) untuk CloudHSM CLI.

1. Untuk mengatur MFA menggunakan Strategi Tanda Token, Anda harus terlebih dahulu membuat kunci pribadi RSA 2048 bit dan kunci publik terkait.

```
$ openssl genrsa -out officer1.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)

$ openssl rsa -in officer1.key -outform PEM -pubout -out officer1.pub
writing RSA key
```

2. Gunakan perintah berikut untuk memulai CLI dalam mode interaktif.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

3. Menggunakan CloudHSM CLI, masuk ke akun pengguna Anda.

```
aws-cloudhsm > login --username <admin> --role <admin> --cluster-id <cluster ID>
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<admin>",
    "role": "<admin>"
  }
}
```

4. Selanjutnya, jalankan perintah untuk mengubah strategi MFA Anda. Anda harus memberikan parameter `--token`. Parameter ini menentukan file yang akan memiliki token yang tidak ditandatangani ditulis untuk itu.

```
aws-cloudhsm > user change-mfa token-sign --token unsigned-tokens.json --
username <username> --role crypto-user --change-quorum
Enter password:
Confirm password:
```

- Anda sekarang memiliki file dengan token yang tidak ditandatangani yang perlu ditandatangani: `unsigned-tokens.json`. Jumlah token dalam file ini tergantung pada jumlah HSMs di cluster Anda. Setiap token mewakili satu HSM. File ini diformat JSON dan berisi token yang perlu ditandatangani untuk membuktikan bahwa Anda memiliki kunci pribadi.

```
$ cat unsigned-tokens.json
{
  "version": "2.0",
  "tokens": [
    {
      {
        "unsigned": "Vtf/9Q0FY45v/E1osvpEMr59JsnP/hLDm4It002vqL8=",
        "signed": ""
      },
      {
        "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
        "signed": ""
      },
      {
        "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
        "signed": ""
      }
    ]
  }
}
```

- Langkah selanjutnya adalah menandatangani token ini dengan kunci pribadi yang dibuat pada langkah 1. Tempatkan tanda tangan kembali di file. Pertama, Anda harus mengekstrak dan memecahkan kode token yang dikodekan base64.

```
$ echo "Vtf/9Q0FY45v/E1osvpEMr59JsnP/hLDm4It002vqL8=" > token1.b64
$ echo "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=" > token2.b64
$ echo "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=" > token3.b64
$ base64 -d token1.b64 > token1.bin
$ base64 -d token2.b64 > token2.bin
$ base64 -d token3.b64 > token3.bin
```

7. Sekarang, Anda memiliki token biner yang dapat Anda tandatangani menggunakan kunci pribadi RSA yang dibuat pada langkah 1.

```
$ openssl pkeyutl -sign \
    -inkey officer1.key \
    -pkeyopt digest:sha256 \
    -keyform PEM \
    -in token1.bin \
    -out token1.sig.bin
$ openssl pkeyutl -sign \
    -inkey officer1.key \
    -pkeyopt digest:sha256 \
    -keyform PEM \
    -in token2.bin \
    -out token2.sig.bin
$ openssl pkeyutl -sign \
    -inkey officer1.key \
    -pkeyopt digest:sha256 \
    -keyform PEM \
    -in token3.bin \
    -out token3.sig.bin
```

8. Sekarang, Anda memiliki tanda tangan biner token. Anda harus menyandikannya menggunakan base64, dan kemudian menempatkannya kembali di file token Anda.

```
$ base64 -w0 token1.sig.bin > token1.sig.b64
$ base64 -w0 token2.sig.bin > token2.sig.b64
$ base64 -w0 token3.sig.bin > token3.sig.b64
```

9. Terakhir, Anda dapat menyalin dan menempelkan nilai base64 kembali ke file token Anda:

```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwx9bJ0UUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "e1w3fZeCKIY50C4zPeg9Rt90M1Q1q3W1Jh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2t1F9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGB0j4yB9recp0BB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
  ],
}
```



```
{
  "unsigned": "LMMFc34ASpNvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
  "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+Hjmr0530QpZaJ/VXGIgApD0myuu/
ZGNKQTCskkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWr13JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAx0RTLlmwyU0YvPY0vUhc
+s83hx36QpGwGcD7RA0bPT50rTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
},
{
  "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93Sb0UyZRjH8=",
  "signed": "VgQPvrTsvGljVBFxHnsduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWRdvS0uGHdkFYp1apHgJZ7PDVmgCtkIXVD21FYppcgN1SzkY1ftr5E0jqS9ZjYEggGuB4g//
MxaBaRbJai/6B1cE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96Qok1Nb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+Vhmn1nFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCi0YDzUCd43GftGq2LfxH3qSD51oFHg1HQV0Y0jyVzz1Avub5HQdt00"
}
]
}
```

10. Sekarang file token Anda memiliki semua tanda tangan yang diperlukan, Anda dapat melanjutkan. Masukkan nama file yang berisi token yang ditandatangani dan tekan tombol enter. Terakhir, masukkan jalur kunci publik Anda.

```
Enter signed token file path (press enter if same as the unsigned token file):
Enter public key PEM file path:officer1.pub
```

```
{
  "error_code": 0,
  "data": {
    "username": "<username>",
    "role": "crypto-user"
  }
}
```

Sekarang Anda telah mengatur pengguna Anda dengan MFA.

```
{
  "username": "<username>",
  "role": "crypto-user",
  "locked": "false",
  "mfa": [
    {
      "strategy": "token-sign",
```

```
    "status": "enabled"
  }
],
"cluster-coverage": "full"
},
```

Buat pengguna dengan MFA diaktifkan untuk CloudHSM CLI

Ikuti langkah-langkah ini untuk membuat AWS CloudHSM pengguna dengan otentikasi multi-faktor (MFA) diaktifkan.

1. Gunakan CloudHSM CLI untuk masuk ke HSM sebagai admin.
2. Gunakan [user create](#) perintah untuk membuat pengguna pilihan Anda. Kemudian ikuti langkah-langkah [Mengatur MFA untuk CloudHSM CLI](#) untuk mengatur MFA untuk pengguna.

Masuk pengguna dengan MFA diaktifkan untuk CloudHSM CLI

Ikuti langkah-langkah ini untuk login AWS CloudHSM pengguna dengan otentikasi multi-faktor (MFA) diaktifkan.

1. Gunakan [login mfa-token-sign](#) perintah di CloudHSM CLI untuk memulai proses login dengan MFA untuk pengguna yang mengaktifkan MFA.

```
aws-cloudhsm > login --username <username> --role <role> mfa-token-sign --
token <unsigned-tokens.json>
Enter password:
```

2. Masukkan kata sandi Anda. Anda kemudian akan diminta untuk memasukkan jalur ke file token yang berisi pasangan token, di mana unsigned/signed token yang ditandatangani adalah yang dihasilkan dengan menggunakan kunci pribadi Anda.

```
aws-cloudhsm > login --username <username> --role <role> mfa-token-sign --
token <unsigned-tokens.json>
Enter password:
Enter signed token file path (press enter if same as the unsigned token file):
```

3. Saat diminta untuk memasukkan jalur file token yang ditandatangani, Anda dapat memeriksa file token yang tidak ditandatangani di terminal terpisah. Identifikasi file dengan token yang tidak ditandatangani yang perlu ditandatangani: *<unsigned-tokens.json>*. Jumlah token dalam

file ini tergantung pada jumlah HSMs di cluster Anda. Setiap token mewakili satu HSM. File ini diformat JSON dan berisi token yang perlu ditandatangani untuk membuktikan bahwa Anda memiliki kunci pribadi.

```
$ cat <unsigned-tokens.json>
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "Vtf/9Q0FY45v/E1osvpEMr59JsnP/hLDm4It002vqL8=",
      "signed": ""
    },
    {
      "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
      "signed": ""
    },
    {
      "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
      "signed": ""
    }
  ]
}
```

4. Tanda tangani token yang tidak ditandatangani dengan kunci pribadi yang dibuat pada langkah 2. Pertama, Anda harus mengekstrak dan memecahkan kode token yang dikodekan base64.

```
$ echo "Vtf/9Q0FY45v/E1osvpEMr59JsnP/hLDm4It002vqL8=" > token1.b64
$ echo "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=" > token2.b64
$ echo "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=" > token3.b64
$ base64 -d token1.b64 > token1.bin
$ base64 -d token2.b64 > token2.bin
$ base64 -d token3.b64 > token3.bin
```

5. Anda sekarang memiliki token biner. Tanda tangani mereka menggunakan kunci pribadi RSA yang sebelumnya Anda buat di [langkah 1 pengaturan MFA](#).

```
$ openssl pkeyutl -sign \
  -inkey officer1.key \
  -pkeyopt digest:sha256 \
  -keyform PEM \
  -in token1.bin \
  -out token1.sig.bin
```

```
$ openssl pkeyutl -sign \
    -inkey officer1.key \
    -pkeyopt digest:sha256 \
    -keyform PEM \
    -in token2.bin \
    -out token2.sig.bin
$ openssl pkeyutl -sign \
    -inkey officer1.key \
    -pkeyopt digest:sha256 \
    -keyform PEM \
    -in token3.bin \
    -out token3.sig.bin
```

6. Anda sekarang memiliki tanda tangan biner token. Encode mereka menggunakan base64, dan menempatkan mereka kembali dalam file token Anda.

```
$ base64 -w0 token1.sig.bin > token1.sig.b64
$ base64 -w0 token2.sig.bin > token2.sig.b64
$ base64 -w0 token3.sig.bin > token3.sig.b64
```

7. Terakhir, salin dan tempel nilai base64 kembali ke file token Anda:

```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwx9bJ0UUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3W1Jh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2t1F9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGB0j4yB9recp0BB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASpNvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+Hjmr0530QpZaJ/VXGIgApD0myuu/
ZGNKQTCskkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIrd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAx0RTLlmwyU0YvPY0vUhC
+s83hx36QpGwGcD7RA0bPT50rTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    }
  ]
}
```

```

    "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93Sb0UyZRjH8=",
    "signed": "VgQPvrTsvGljVBFxHnsduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkY1ftr5E0jqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCi0YDzUCd43GftGq2LfxH3qSD51oFHglHQV0Y0jyVzzlAvub5HQdt00
  }
]
}

```

8. Sekarang file token Anda memiliki semua tanda tangan yang diperlukan, Anda dapat melanjutkan. Masukkan nama file yang berisi token yang ditandatangani dan tekan tombol enter. Anda sekarang harus berhasil masuk.

```

aws-cloudhsm > login --username <username> --role <role> mfa-token-sign --
token <unsigned-tokens.json>
Enter password:
Enter signed token file path (press enter if same as the unsigned token file):
{
  "error_code": 0,
  "data": {
    "username": "<username>",
    "role": "<role>"
  }
}

```

Putar tombol untuk pengguna dengan MFA diaktifkan untuk CloudHSM CLI

Ikuti langkah-langkah ini untuk memutar tombol bagi AWS CloudHSM pengguna dengan otentikasi multi-faktor (MFA) diaktifkan.

<result>

Anda telah menandatangani file token berformat JSON yang dihasilkan dengan kunci pribadi Anda dan mendaftarkan kunci publik MFA baru.

</result>

1. Gunakan CloudHSM CLI untuk masuk ke HSM sebagai admin atau sebagai pengguna tertentu yang mengaktifkan MFA ([lihat Masuk](#) pengguna dengan MFA diaktifkan untuk detailnya).

2. Selanjutnya, jalankan perintah untuk mengubah strategi MFA Anda. Anda harus memberikan parameter `--token`. Parameter ini menentukan file yang akan memiliki token yang tidak ditandatangani ditulis untuk itu.

```
aws-cloudhsm > user change-mfa token-sign --token unsigned-tokens.json --
username <username> --role crypto-user --change-quorum
Enter password:
Confirm password:
```

3. Identifikasi file dengan token yang tidak ditandatangani yang perlu ditandatangani: `unsigned-tokens.json`. Jumlah token dalam file ini tergantung pada jumlah HSMs di cluster Anda. Setiap token mewakili satu HSM. File ini diformat JSON dan berisi token yang perlu ditandatangani untuk membuktikan bahwa Anda memiliki kunci pribadi. Ini akan menjadi kunci pribadi baru dari public/private key pair RSA baru yang ingin Anda gunakan untuk memutar kunci publik yang saat ini terdaftar.

```
$ cat unsigned-tokens.json
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "Vtf/9Q0FY45v/E1osvpEMr59JsnP/hLDm4It002vqL8=",
      "signed": ""
    },
    {
      "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
      "signed": ""
    },
    {
      "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
      "signed": ""
    }
  ]
}
```

4. Tanda tangani token ini dengan kunci pribadi yang sebelumnya Anda buat selama penyiapan. Pertama kita harus mengekstrak dan memecahkan kode token yang dikodekan base64.

```
$ echo "Vtf/9Q0FY45v/E1osvpEMr59JsnP/hLDm4It002vqL8=" > token1.b64
$ echo "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=" > token2.b64
$ echo "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=" > token3.b64
```

```
$ base64 -d token1.b64 > token1.bin
$ base64 -d token2.b64 > token2.bin
$ base64 -d token3.b64 > token3.bin
```

5. Anda sekarang memiliki token biner. Tanda tangani mereka menggunakan kunci pribadi RSA yang sebelumnya Anda buat selama penyiapan.

```
$ openssl pkeyutl -sign \
    -inkey officer1.key \
    -pkeyopt digest:sha256 \
    -keyform PEM \
    -in token1.bin \
    -out token1.sig.bin
$ openssl pkeyutl -sign \
    -inkey officer1.key \
    -pkeyopt digest:sha256 \
    -keyform PEM \
    -in token2.bin \
    -out token2.sig.bin
$ openssl pkeyutl -sign \
    -inkey officer1.key \
    -pkeyopt digest:sha256 \
    -keyform PEM \
    -in token3.bin \
    -out token3.sig.bin
```

6. Anda sekarang memiliki tanda tangan biner token. Encode mereka menggunakan base64, dan menempatkan mereka kembali dalam file token Anda.

```
$ base64 -w0 token1.sig.bin > token1.sig.b64
$ base64 -w0 token2.sig.bin > token2.sig.b64
$ base64 -w0 token3.sig.bin > token3.sig.b64
```

7. Terakhir, salin dan tempel nilai base64 kembali ke file token Anda:

```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwx9bJ0UUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Q1q3W1Jh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2t1F9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGB0j4yB9recp0BB9K8QFSpJZALSEdDgUc/
```

```

mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
  },
  {
    "unsigned": "LMMFc34ASpNvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
    "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+Hjmr0530QpZaJ/VXGIgApD0myuu/
ZGNKQTCskkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWr13JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAx0RTLlmwyU0YvPY0vUhc
+s83hx36QpGwGcD7RA0bPT50rTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
  },
  {
    "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93Sb0UyZRjH8=",
    "signed": "VgQPvrTsvGljVBFxHnsduq16x8ZrxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD21FYppcgN1SzkY1ftr5E0jqS9ZjYEggGuB4g//
MxaBaRbJai/6B1cE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96Qok1Nb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCi0YDzUCd43GftGq2LfxH3qSD51oFHg1HQV0Y0jyVzz1Avub5HQdt0Q
  }
]
}

```

8. Sekarang file token Anda memiliki semua tanda tangan yang diperlukan, Anda dapat melanjutkan. Masukkan nama file yang berisi token yang ditandatangani dan tekan tombol enter. Terakhir, masukkan jalur kunci publik baru Anda. Sekarang Anda akan melihat yang berikut ini sebagai bagian dari output [daftar pengguna](#).

```

Enter signed token file path (press enter if same as the unsigned token file):
Enter public key PEM file path:officer1.pub
{
  "error_code": 0,
  "data": {
    "username": "<username>",
    "role": "crypto-user"
  }
}

```

Sekarang kami telah mengatur pengguna kami dengan MFA.

```

{
  "username": "<username>",

```



```

    "role": "crypto-user",
    "locked": "false",
    "mfa": [
      {
        "strategy": "token-sign",
        "status": "enabled"
      }
    ],
    "cluster-coverage": "full"
  },

```

Membatalkan pendaftaran kunci publik MFA menggunakan CloudHSM CLI

Ikuti langkah-langkah ini untuk membatalkan pendaftaran kunci publik multi-faktor otentikasi (MFA) untuk AWS CloudHSM pengguna admin saat kunci publik MFA terdaftar.

1. Gunakan CloudHSM CLI untuk masuk ke HSM sebagai admin dengan MFA diaktifkan.
2. Gunakan user change-mfa token-sign perintah untuk menghapus MFA untuk pengguna.

```

aws-cloudhsm > user change-mfa token-sign --username <username> --role admin --
deregister --change-quorum
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "<username>",
    "role": "admin"
  }
}

```

Referensi file token untuk MFA dengan CloudHSM CLI

File token yang dihasilkan saat mendaftarkan kunci publik multi-faktor otentikasi (MFA) atau ketika mencoba masuk ke CloudHSM CLI menggunakan MFA terdiri dari yang berikut:

- Token: Sebuah array base64 dikodekan pasangan unsigned/signed token dalam bentuk literal objek JSON.
- Unsigned: Token base64 yang dikodekan dan di-hash. SHA256

- Ditandatangani: Token bertanda tangan (tanda tangan) yang disandikan base64 dari token yang tidak ditandatangani, menggunakan kunci pribadi RSA 2048-bit.

```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwx9bJ0UUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2t1F9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/TK0PVaxLN42X
+pebV4juwMhN4mK4CzdFAJgM+UGB0j4yB9recp0BB9K8QFSpJZALSEdDgUc/mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37+j/
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASpvnNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+Hjmr0530QpZaJ/VXGIgApD0myuu/
ZGNKQTCskkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAx0RTLlmwyU0YvPY0vUhc
+s83hx36QpGwGcD7RA0bPT50rTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93Sb0UyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnsduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmgCtkIXVD2lFYppcgNlSzkYlfttr5E0jqS9ZjYEggGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCi0YDzUCd43GftGq2LfxH3qSD51oFHg1HQV0Y0jyVzzlAvub5HQdt0QdErI
    }
  ]
}
```

Kelola otentikasi kuorum (M of N access control) menggunakan CloudHSM CLI

AWS CloudHSM cluster mendukung otentikasi kuorum, juga dikenal sebagai M of N access control. Fitur ini mengharuskan pengguna HSM untuk bekerja sama untuk operasi tertentu, menambahkan lapisan perlindungan ekstra.

Dengan otentikasi kuorum, tidak ada satu pengguna pun di HSM yang dapat melakukan operasi yang dikendalikan kuorum pada HSM. Sebaliknya, sejumlah minimum pengguna HSM (paling sedikit 2) harus bekerja sama untuk melakukan operasi ini.

Autentikasi kuorum dapat mengontrol operasi berikut:

- Manajemen pengguna HSM oleh [admin](#): Membuat dan menghapus pengguna HSM atau mengubah kata sandi pengguna HSM yang berbeda. Untuk informasi selengkapnya, lihat [Manajemen pengguna dengan otentikasi kuorum diaktifkan untuk menggunakan AWS CloudHSM CloudHSM CLI](#).

Poin penting tentang otentikasi kuorum di AWS CloudHSM

- Pengguna HSM dapat menandatangani token kuorum mereka sendiri—yaitu, memberikan salah satu persetujuan yang diperlukan untuk otentikasi kuorum.
- Anda memilih jumlah minimum pemberi persetujuan kuorum, yang berkisar dari dua (2) hingga delapan (8).
- HSMs dapat menyimpan hingga 1024 token kuorum. Ketika batas ini tercapai, HSM membersihkan token kedaluwarsa untuk membuat yang baru.
- Token kedaluwarsa sepuluh menit setelah pembuatan secara default.
- Untuk cluster dengan MFA diaktifkan, kunci yang sama digunakan untuk otentikasi kuorum dan otentikasi multi-faktor (MFA). Lihat [Menggunakan CloudHSM CLI untuk mengelola MFA untuk informasi selengkapnya](#).
- Setiap HSM dapat berisi satu token per layanan Admin dan beberapa token per layanan Pengguna Crypto.

Topik berikut ini menyediakan informasi lebih lanjut tentang autentikasi kuorum di AWS CloudHSM.

Topik

- [Proses otentikasi kuorum untuk CloudHSM CLI](#)
- [Nama dan jenis AWS CloudHSM layanan yang didukung untuk otentikasi kuorum dengan CloudHSM CLI](#)
- [Mengatur otentikasi kuorum untuk AWS CloudHSM admin menggunakan CloudHSM CLI](#)
- [Manajemen pengguna dengan otentikasi kuorum diaktifkan untuk menggunakan AWS CloudHSM CloudHSM CLI](#)

- [Ubah nilai minimum kuorum untuk menggunakan AWS CloudHSM CloudHSM CLI](#)

Proses otentikasi kuorum untuk CloudHSM CLI

Langkah-langkah berikut merangkum proses otentikasi kuorum untuk CloudHSM CLI. Untuk langkah-langkah dan alat tertentu, lihat [Manajemen pengguna dengan otentikasi kuorum diaktifkan untuk menggunakan AWS CloudHSM CloudHSM CLI](#).

1. Setiap pengguna modul keamanan perangkat keras (HSM) membuat kunci asimetris untuk penandatanganan. Pengguna melakukan ini di luar HSM, berhati-hati untuk melindungi kunci dengan tepat.
2. Setiap pengguna HSM masuk ke HSM dan mendaftarkan bagian publik dari kunci penandatanganan mereka (kunci publik) dengan HSM.
3. Ketika pengguna HSM ingin melakukan operasi yang dikendalikan kuorum, pengguna yang sama masuk ke HSM dan mendapatkan token kuorum.
4. Pengguna HSM memberikan token kuorum ke satu atau lebih pengguna HSM lainnya dan meminta persetujuan mereka.
5. Pengguna HSM lainnya menyetujui dengan menggunakan kunci mereka untuk secara kriptografi menandatangani token kuorum. Hal ini terjadi di luar HSM.
6. Ketika pengguna HSM memiliki jumlah persetujuan yang diperlukan, pengguna yang sama masuk ke HSM dan menjalankan operasi yang dikendalikan kuorum dengan --approval argumen, menyediakan file token kuorum yang ditandatangani, yang berisi semua persetujuan yang diperlukan (tanda tangan).
7. HSM menggunakan kunci publik terdaftar dari setiap penandatanganan untuk memverifikasi tanda tangan. Jika tanda tangan valid, HSM menyetujui token dan operasi yang dikendalikan kuorum dilakukan.

Nama dan jenis AWS CloudHSM layanan yang didukung untuk otentikasi kuorum dengan CloudHSM CLI

Layanan Admin: Otentikasi kuorum digunakan untuk layanan istimewa admin seperti membuat pengguna, menghapus pengguna, mengubah kata sandi pengguna, mengatur nilai kuorum, dan menonaktifkan kemampuan kuorum dan MFA.

Layanan Pengguna Crypto: Otentikasi kuorum digunakan untuk layanan istimewa pengguna kriptografi yang terkait dengan kunci tertentu seperti menandatangani dengan kunci, kunci, sharing/unsharing

kunci, dan wrapping/unwrapping pengaturan atribut kunci. Nilai kuorum kunci terkait dikonfigurasi saat kunci dihasilkan, diimpor, atau dibuka. Nilai kuorum harus sama dengan atau kurang dari jumlah pengguna yang dikaitkan dengan kunci, yang mencakup pengguna yang kunci dibagikan dan pemilik kunci.

Setiap jenis layanan selanjutnya dipecah menjadi nama layanan yang memenuhi syarat, yang berisi serangkaian operasi layanan yang didukung kuorum tertentu yang dapat dilakukan.

Nama layanan	Jenis Layanan	Operasi layanan
user	Admin	• pengguna membuat • menghapus pengguna • perubahan kata sandi pengguna • perubahan pengguna-mfa
kuorum	Admin	• tanda token kuorum set-quorum-value
kluster ¹	Admin	• kluster mtls register-trust-anchor • kluster mtls deregister-trust-anchor • cluster mtls set-penegakan
manajemen kunci	Pengguna Crypto	• bungkus kunci • buka kunci • pembagian kunci • kunci unshare • kunci set-attribut
penggunaan kunci	Pengguna Crypto	• tanda kunci

[1] Layanan cluster tersedia secara eksklusif di hsm2m.medium

Mengatur otentikasi kuorum untuk AWS CloudHSM admin menggunakan CloudHSM CLI

Topik berikut menjelaskan langkah-langkah yang harus Anda selesaikan untuk mengonfigurasi modul keamanan perangkat keras (HSM) Anda sehingga AWS CloudHSM [admin](#) dapat menggunakan otentikasi kuorum. Anda perlu melakukan langkah-langkah ini hanya sekali ketika Anda pertama kali mengonfigurasi otentikasi kuorum untuk admin. Setelah Anda menyelesaikan langkah ini, lihat [Manajemen pengguna dengan otentikasi kuorum diaktifkan untuk menggunakan AWS CloudHSM CloudHSM CLI](#).

Topik

- [Prasyarat](#)
- [Langkah 1. Buat dan daftarkan kunci untuk penandatanganan](#)
- [Langkah 2. Tetapkan nilai minimum kuorum pada HSM](#)
- [Nilai minimum kuorum](#)

Prasyarat

Untuk memahami contoh ini, Anda harus terbiasa dengan [CloudHSM CLI](#).

Langkah 1. Buat dan daftarkan kunci untuk penandatanganan

Untuk menggunakan otentikasi kuorum, setiap admin harus menyelesaikan semua langkah berikut:

Topik

- [Buat key pair RSA](#)
- [Membuat dan menandatangani token pendaftaran](#)
- [Daftarkan kunci publik dengan HSM](#)

Buat key pair RSA

Ada banyak cara berbeda untuk membuat dan melindungi pasangan kunci. Contoh berikut menunjukkan cara melakukannya dengan [OpenSSL](#).

Example— Membuat kunci privat dengan OpenSSL

Contoh berikut menunjukkan bagaimana menggunakan OpenSSL untuk membuat kunci RSA 2048-bit. Untuk menggunakan contoh ini, ganti `<admin.key>` dengan nama file tempat Anda ingin menyimpan kunci.

```
$ openssl genrsa -out <admin.key>
Generating RSA private key, 2048 bit long modulus
.....+++
.+++
e is 65537 (0x10001)
```

Berikutnya, hasilkan kunci publik menggunakan kunci privat yang baru saja Anda buat.

Example— Membuat kunci privat dengan OpenSSL

Contoh berikut menunjukkan bagaimana menggunakan OpenSSL untuk membuat kunci publik dari kunci privat yang baru saja Anda buat.

```
$ openssl rsa -in admin.key -outform PEM -pubout -out admin1.pub
writing RSA key
```

Membuat dan menandatangani token pendaftaran

Anda membuat token dan menandatangani dengan kunci privat yang baru saja Anda hasilkan pada langkah sebelumnya.

Example— Buat token pendaftaran

1. Gunakan perintah berikut untuk memulai CloudHSM CLI:

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Buat token registrasi dengan menjalankan perintah [quorum token-sign](#) generate:

```
aws-cloudhsm > quorum token-sign generate --service registration --token /path/
tokenfile
{
  "error_code": 0,
  "data": {
    "path": "/path/tokenfile"
  }
}
```

3. Perintah [quorum token-sign generate menghasilkan](#) token registrasi di jalur file yang ditentukan. Periksa file token:

```
$ cat /path/tokenfile
{
  "version": "2.0",
  "tokens": [
    {
      "approval_data": <approval data in base64 encoding>,
      "unsigned": <unsigned token in base64 encoding>,
      "signed": ""
    }
  ]
}
```

File token terdiri dari yang berikut:

- approval_data: Token data acak yang dikodekan base64 yang data mentahnya tidak melebihi maksimum 245 byte.
- unsigned: Token base64 yang dikodekan dan SHA256 di-hash dari approval_data.
- ditandatangani: Token bertanda tangan (tanda tangan) yang disandikan base64 dari token yang tidak ditandatangani, menggunakan kunci pribadi RSA 2048-bit yang sebelumnya dibuat dengan OpenSSL.

Anda menandatangani token yang tidak ditandatangani dengan kunci pribadi untuk menunjukkan bahwa Anda memiliki akses ke kunci pribadi. Anda akan memerlukan file token pendaftaran yang diisi penuh dengan tanda tangan dan kunci publik untuk mendaftarkan admin sebagai pengguna quorum dengan cluster. AWS CloudHSM

Example— Tanda tangani token pendaftaran yang tidak ditandatangani

1. Dekode token unsigned base64 yang dikodekan dan letakkan ke dalam file biner:

```
$ echo -n '6BMUj6mUjjko6ZLCEdzG1WpR5sILhFJfqhW1ej30q1g=' | base64 -d > admin.bin
```

2. Gunakan OpenSSL dan kunci pribadi untuk menandatangani token pendaftaran biner yang sekarang tidak ditandatangani dan membuat file tanda tangan biner:

```
$ openssl pkeyutl -sign \
-inkey admin.key \
-pkeyopt digest:sha256 \
-keyform PEM \
-in admin.bin \
-out admin.sig.bin
```

3. Mengkodekan tanda tangan biner ke base64:

```
$ base64 -w0 admin.sig.bin > admin.sig.b64
```

4. Salin dan tempel tanda tangan yang dikodekan base64 ke dalam file token:

```
{
  "version": "2.0",
  "tokens": [
    {
      "approval_data": <approval data in base64 encoding>,
      "unsigned": <unsigned token in base64 encoding>,
      "signed": <signed token in base64 encoding>
    }
  ]
}
```

Daftarkan kunci publik dengan HSM

Setelah membuat kunci, admin harus mendaftarkan kunci publik dengan AWS CloudHSM cluster.

Untuk mendaftarkan kunci publik dengan HSM

1. Gunakan perintah berikut untuk memulai CloudHSM CLI:

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Menggunakan CloudHSM CLI, masuk sebagai admin.

```
aws-cloudhsm > login --username <admin> --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<admin>",
    "role": "admin"
  }
}
```

3. Gunakan perintah [Daftarkan strategi kuorum tanda token pengguna menggunakan CloudHSM CLI](#) untuk mendaftarkan kunci publik. Untuk informasi selengkapnya, lihat contoh berikut atau gunakan perintah `help user change-quorum token-sign register`.

Example— Daftarkan kunci publik dengan AWS CloudHSM cluster

Contoh berikut menunjukkan cara menggunakan `user change-quorum token-sign register` perintah di CloudHSM CLI untuk mendaftarkan kunci publik admin dengan HSM. Untuk menggunakan perintah ini, admin harus login ke HSM. Ganti nilai-nilai ini dengan nilai Anda sendiri:

```
aws-cloudhsm > user change-quorum token-sign register --public-key </path/admin.pub> --
signed-token </path/tokenfile>
{
  "error_code": 0,
  "data": {
    "username": "admin",
    "role": "admin"
  }
}
```

 Note

/path/admin.pub: Jalur file ke file PEM kunci publik

Wajib: Ya

/path/tokenfile: Filepath dengan token yang ditandatangani oleh kunci pribadi pengguna

Wajib: Ya

Setelah semua admin mendaftarkan kunci publik mereka, output dari user list perintah menunjukkan ini di bidang quorum, yang menyatakan strategi quorum yang diaktifkan yang digunakan, seperti yang ditunjukkan di bawah ini:

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "admin2",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

```

    },
    {
      "username": "admin3",
      "role": "admin",
      "locked": "false",
      "mfa": [],
      "quorum": [
        {
          "strategy": "token-sign",
          "status": "enabled"
        }
      ],
      "cluster-coverage": "full"
    },
    {
      "username": "admin4",
      "role": "admin",
      "locked": "false",
      "mfa": [],
      "quorum": [
        {
          "strategy": "token-sign",
          "status": "enabled"
        }
      ],
      "cluster-coverage": "full"
    },
    {
      "username": "app_user",
      "role": "internal(APPLIANCE_USER)",
      "locked": "false",
      "mfa": [],
      "quorum": [],
      "cluster-coverage": "full"
    }
  ]
}

```

Dalam contoh ini, AWS CloudHSM cluster memiliki dua HSMs, masing-masing dengan admin yang sama, seperti yang ditunjukkan pada output berikut dari perintah `user list`. Untuk informasi selengkapnya tentang membuat pengguna, lihat [Manajemen pengguna dengan CloudHSM CLI](#)

Langkah 2. Tetapkan nilai minimum kuorum pada HSM

Untuk menggunakan otentikasi kuorum, admin harus masuk ke HSM dan kemudian menetapkan nilai minimum kuorum. Ini adalah jumlah minimum persetujuan admin yang diperlukan untuk melakukan operasi manajemen pengguna HSM. Setiap admin di HSM dapat menetapkan nilai minimum kuorum, termasuk admin yang belum mendaftarkan kunci untuk penandatanganan. Anda dapat mengubah nilai minimum kuorum kapan saja. Untuk informasi selengkapnya, lihat [Ubah nilai minimum](#).

Untuk menetapkan nilai minimum kuorum pada HSM

1. Gunakan perintah berikut untuk memulai CloudHSM CLI:

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Menggunakan CloudHSM CLI, masuk sebagai admin.

```
aws-cloudhsm > login --username <admin> --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<admin>",
    "role": "admin"
  }
}
```

3. Gunakan perintah [Perbarui nilai kuorum menggunakan CloudHSM CLI](#) untuk menetapkan nilai minimum kuorum. `--serviceBendera` mengidentifikasi layanan HSM yang Anda setel nilainya. Lihat contoh berikut atau gunakan `help quorum token-sign set-quorum-value` perintah untuk informasi lebih lanjut.

Example— Tetapkan nilai minimum kuorum pada HSM

Contoh ini menggunakan nilai minimum kuorum dua (2). Anda dapat memilih nilai dari dua (2) hingga delapan (8), hingga jumlah total admin di HSM. Dalam contoh ini, HSM memiliki empat (4) admin, sehingga nilai maksimum yang mungkin adalah empat (4).

Untuk menggunakan perintah contoh berikut, ganti angka akhir (**<2>**) dengan nilai minimum kuorum yang disukai.

```
aws-cloudhsm > quorum token-sign set-quorum-value --service user --value <2>
{
  "error_code": 0,
  "data": "Set quorum value successful"
}
```

Dalam contoh ini, [Tampilkan nilai kuorum menggunakan CloudHSM CLI](#) perintah mencantumkan jenis layanan HSM, nama, dan deskripsi yang disertakan dalam layanan.

Nilai minimum kuorum

Untuk mendapatkan nilai minimum kuorum untuk layanan, gunakan perintah: `quorum token-sign list-quorum-values`

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 2,
    "quorum": 1
  }
}
```

Output dari `quorum token-sign list-quorum-values` perintah sebelumnya menunjukkan bahwa nilai minimum kuorum untuk layanan pengguna HSM, yang bertanggung jawab untuk operasi manajemen pengguna, sekarang dua (2). Setelah Anda menyelesaikan langkah ini, lihat [Manajemen pengguna dengan kuorum \(M dari N\)](#).

Layanan Admin: Otentikasi kuorum digunakan untuk layanan istimewa admin seperti membuat pengguna, menghapus pengguna, mengubah kata sandi pengguna, mengatur nilai kuorum, dan menonaktifkan kemampuan kuorum dan MFA.

Layanan Pengguna Crypto: Otentikasi kuorum digunakan untuk layanan istimewa pengguna kripto yang terkait dengan kunci tertentu seperti menandatangani dengan kunci, kunci, sharing/unsharing kunci, dan wrapping/unwrapping pengaturan atribut kunci. Nilai kuorum kunci terkait dikonfigurasi saat kunci dihasilkan, diimpor, atau dibuka. Nilai kuorum harus sama dengan atau kurang dari jumlah pengguna yang dikaitkan dengan kunci, yang mencakup pengguna yang kunci dibagikan dan pemilik kunci.

Setiap jenis layanan selanjutnya dipecah menjadi nama layanan yang memenuhi syarat, yang berisi serangkaian operasi layanan yang didukung kuorum tertentu yang dapat dilakukan.

Nama layanan	Jenis Layanan	Operasi layanan
user	Admin	• pengguna membuat • menghapus pengguna • perubahan kata sandi pengguna • perubahan pengguna-mfa
kuorum	Admin	• tanda token kuorum set-quorum-value
kluster ¹	Admin	• kluster mtls register-trust-anchor • kluster mtls deregister-trust-anchor • cluster mtls set-penegakan
manajemen kunci	Pengguna Crypto	• bungkus kunci • buka kunci • pembagian kunci • kunci unshare • kunci set-attribut
penggunaan kunci	Pengguna Crypto	• tanda kunci

[1] Layanan cluster tersedia secara eksklusif di hsm2m.medium

Manajemen pengguna dengan otentikasi kuorum diaktifkan untuk menggunakan AWS CloudHSM CloudHSM CLI

AWS CloudHSM [Admin](#) pada modul keamanan perangkat keras (HSM) dapat mengonfigurasi otentikasi kuorum untuk operasi berikut di cluster: AWS CloudHSM

- [Buat AWS CloudHSM pengguna dengan CloudHSM CLI](#)
- [Hapus AWS CloudHSM pengguna dengan CloudHSM CLI](#)
- [Ubah kata sandi pengguna dengan CloudHSM CLI](#)
- [Kategori perubahan pengguna-mfa di CloudHSM CLI](#)

Setelah AWS CloudHSM cluster dikonfigurasi untuk otentikasi kuorum, admin tidak dapat melakukan operasi manajemen pengguna HSM sendiri. Contoh berikut menunjukkan output ketika admin mencoba untuk membuat pengguna baru di HSM. Perintah gagal dengan kesalahan, yang menyatakan bahwa otentikasi kuorum diperlukan.

```
aws-cloudhsm > user create --username user1 --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 1,
  "data": "Quorum approval is required for this operation"
}
```

Untuk melakukan operasi manajemen pengguna HSM, admin harus menyelesaikan tugas-tugas berikut:

Topik

- [Langkah 1. Dapatkan token kuorum](#)
- [Langkah 2. Dapatkan tanda tangan dari admin yang menyetujui](#)
- [Langkah 3. Menyetujui token pada AWS CloudHSM cluster dan menjalankan operasi manajemen pengguna](#)

Langkah 1. Dapatkan token kuorum

Pertama, admin harus menggunakan CloudHSM CLI untuk meminta token kuorum.

Dapatkan token kuorum.

1. Gunakan perintah berikut untuk memulai CloudHSM CLI.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Menggunakan CloudHSM CLI, masuk sebagai admin.

```
aws-cloudhsm > login --username <admin> --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<admin>",
    "role": "admin"
  }
}
```

3. Gunakan quorum token-sign generate perintah untuk menghasilkan token kuorum. Untuk informasi selengkapnya, lihat contoh berikut atau gunakan perintah help quorum token-sign generate.

Example— Hasilkan token kuorum

Contoh ini mendapatkan token kuorum untuk admin dengan nama pengguna admin dan menyimpan token ke file bernama. admin.token Untuk menggunakan perintah contoh berikut, ganti nilai ini dengan nilai Anda sendiri:

- **<admin>**— Nama admin yang mendapatkan token. Ini harus admin yang sama yang masuk ke HSM dan menjalankan perintah ini.
- **<admin.token>**— Nama file yang akan digunakan untuk menyimpan token kuorum.

Dalam perintah berikut, user identifikasi nama layanan yang dapat Anda gunakan token yang Anda hasilkan. Dalam hal ini, token adalah untuk operasi manajemen pengguna HSM (userlayanan).

```
aws-cloudhsm > login --username <admin> --role admin --password <password>
{
  "error_code": 0,
  "data": {
    "username": "<admin>",
    "role": "admin"
  }
}
```

```
aws-cloudhsm > quorum token-sign generate --service user --token </path/admin.token>
{
  "error_code": 0,
  "data": {
    "path": "/home/tfile"
  }
}
```

quorum token-sign generatePerintah menghasilkan token kuorum layanan pengguna di jalur file yang ditentukan. File token dapat diperiksa:

```
$ cat </path/admin.token>
{
  "version": "2.0",
  "service": "user-management",
  "approval_data": "AAEAAwAAABgAAAAAAAAAAJ9eFkfcP3mNzJA1fK
+0WbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAAQADAAAFQAAAAAAAAAAW/
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAAAABAAMAAAAUA
+b23gAAAAAAAAAA",
  "token": "012LZkmAHZyAc1hPhyck0oVW33aGrgG77qmDHWQ3CJ8=",
  "signatures": []
}
```

File token terdiri dari yang berikut:

- layanan: Pengenal untuk layanan kuorum yang dikaitkan dengan token.
- approval_data: Token data mentah yang dikodekan base64 yang dihasilkan oleh HSM.
- token: Token base64 yang dikodekan dan SHA-256 yang di-hash dari approval_data

- tanda tangan: Array token bertanda tangan yang disandikan base64 (tanda tangan) dari token yang tidak ditandatangani, di mana setiap tanda tangan pemberi persetujuan dalam bentuk objek JSON literal:

```
{
  "username": "<APPROVER_USERNAME>",
  "role": "<APPROVER_ROLE>",
  "signature": "<APPROVER_RSA2048_BIT_SIGNATURE>"
}
```

Setiap tanda tangan dibuat dari hasil pemberi persetujuan menggunakan kunci pribadi RSA 2048-bit yang sesuai yang kunci publiknya terdaftar di HSM.

Token kuorum layanan pengguna yang dihasilkan dapat dikonfirmasi ada di kluster CloudHSM dengan menjalankan perintah: `quorum token-sign list`

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": [
      {
        "username": "admin",
        "service": "user",
        "approvals-required": {
          "value": 2
        },
        "number-of-approvals": {
          "value": 0
        },
        "token-timeout-seconds": {
          "value": 597
        },
        "cluster-coverage": "full"
      }
    ]
  }
}
```

`token-timeout-seconds` Waktu menunjukkan periode batas waktu dalam hitungan detik agar token yang dihasilkan disetujui sebelum kedaluwarsa.

Langkah 2. Dapatkan tanda tangan dari admin yang menyetujui

Admin yang memiliki token kuorum harus mendapatkan token yang disetujui oleh admin lain. Untuk memberikan persetujuan mereka, admin lain menggunakan kunci penandatanganan mereka untuk menandatangani token secara kriptografis. Mereka melakukan ini di luar HSM.

Ada banyak cara yang berbeda untuk menandatangani token. Contoh berikut menunjukkan cara melakukannya dengan [OpenSSL](#). Untuk menggunakan alat penandatanganan yang berbeda, pastikan alat tersebut menggunakan kunci pribadi admin (kunci penandatanganan) untuk menandatangani intisari token SHA-256.

Example— Dapatkan tanda tangan dari admin yang menyetujui

Dalam contoh ini, admin yang memiliki token (`admin`) membutuhkan setidaknya dua (2) persetujuan. Contoh perintah berikut menunjukkan bagaimana dua (2) admin dapat menggunakan OpenSSL untuk menandatangani token secara kriptografis.

1. Dekode token unsigned base64 yang dikodekan dan letakkan ke dalam file biner:

```
$ echo -n '012LZkmAHZyAc1hPhyck0oVW33aGrgG77qmDHWQ3CJ8=' | base64 -d > admin.bin
```

2. Gunakan OpenSSL dan kunci pribadi masing-masing dari (`admin3`) pemberi persetujuan untuk menandatangani token kuorum biner yang sekarang tidak ditandatangani untuk layanan pengguna dan membuat file tanda tangan biner:

```
$ openssl pkeyutl -sign \  
-inkey admin3.key \  
-pkeyopt digest:sha256 \  
-keyform PEM \  
-in admin.bin \  
-out admin.sig.bin
```

3. Mengkodekan tanda tangan biner ke base64:

```
$ base64 -w0 admin.sig.bin > admin.sig.b64
```

4. Terakhir, salin dan tempel tanda tangan yang dikodekan base64 ke dalam file token, mengikuti format literal objek JSON yang ditentukan sebelumnya untuk tanda tangan pemberi persetujuan:

```
{  
  "version": "2.0",
```

```

    "approval_data": "AAEAAwAAABgAAAAAAAAAAJ9eFkfcP3mNzJA1fK
+0WbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAQADAAAFQAAAAAAAAAAW
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAAAABAAMAA
+b23gAAAAAAAA",
    "token": "012LZkmAHZyAc1hPhyck0oVW33aGrgG77qmDHWQ3CJ8=",
    "signatures": [
      {
        "username": "admin2",
        "role": "admin",
        "signature": "06qx7/mUaVkYYVr1PW7l8JJko+Kh3e8zBIqdk3tAiNy+1rW
+0sDtvYujhEU4a0FVLcrUFmyB/CX90QmgJLgx/pyK+ZPEH+GoJGqk9YZ7X1n0XwZRP9g7hKV
+7XCtg9TuDFtHYWDpBfz2jWiu2fXfX4/
jTs4f2xIfFPIDKcSP8fhxjQ63xEcCf1jzGha6rDQMu4xUWwdtDgft7um7EJ9dXNoHqLB7cTzphaubNaEFbFPXQ1siGm
ssktwyrUGFLpXs1n0tJ0EglGhx2qbYTs+omKWZd0R15WIWEXW3IXw/
Dg5vV0brNpvG0eZK08nSMc27+cyPySc+ZbNw=="
      },
      {
        "username": "admin3",
        "role": "admin",
        "signature": "06qx7/mUaVkYYVr1PW7l8JJko+Kh3e8zBIqdk3tAiNy+1rW
+0sDtvYujhEU4a0FVLcrUFmyB/CX90QmgJLgx/pyK+ZPEH+GoJGqk9YZ7X1n0XwZRP9g7hKV
+7XCtg9TuDFtHYWDpBfz2jWiu2fXfX4/
jTs4f2xIfFPIDKcSP8fhxjQ63xEcCf1jzGha6rDQMu4xUWwdtDgft7um7EJ9dXNoHqLB7cTzphaubNaEFbFPXQ1siGm
ssktwyrUGFLpXs1n0tJ0EglGhx2qbYTs+omKWZd0R15WIWEXW3IXw/
Dg5vV0brNpvG0eZK08nSMc27+cyPySc+ZbNw=="
      }
    ]
  }
}

```

Langkah 3. Menyetujui token pada AWS CloudHSM cluster dan menjalankan operasi manajemen pengguna

Setelah admin memiliki persetujuan/tanda tangan yang diperlukan, seperti yang dijelaskan di bagian sebelumnya, admin dapat menyediakan token tersebut ke AWS CloudHSM cluster bersama dengan salah satu operasi manajemen pengguna berikut:

- [buat](#)
- [hapus](#)
- [ubah-kata sandi](#)
- [user change-mfa](#)

Untuk informasi selengkapnya tentang menggunakan perintah ini, lihat [Manajemen pengguna dengan CloudHSM CLI](#).

Selama transaksi, token akan disetujui dalam AWS CloudHSM cluster dan menjalankan operasi manajemen pengguna yang diminta. Keberhasilan operasi manajemen pengguna bergantung pada token kuorum yang disetujui yang valid dan operasi manajemen pengguna yang valid.

Admin dapat menggunakan token hanya untuk satu operasi. Ketika operasi itu berhasil, token tidak lagi berlaku. Untuk melakukan operasi manajemen pengguna HSM lainnya, admin harus mengulangi proses yang diuraikan di atas. Artinya, admin harus membuat token kuorum baru, mendapatkan tanda tangan baru dari pemberi persetujuan, dan kemudian menyetujui dan menggunakan token baru di HSM dengan operasi manajemen pengguna yang diminta.

Note

Token kuorum hanya berlaku selama sesi login Anda saat ini terbuka. Jika Anda keluar dari CloudHSM CLI atau jika jaringan terputus, token tidak lagi valid. Demikian pula, token resmi hanya dapat digunakan dalam CloudHSM CLI. Itu tidak dapat digunakan untuk mengautentikasi dalam aplikasi yang berbeda.

Example Membuat pengguna baru sebagai admin

Dalam contoh berikut, admin yang masuk membuat pengguna baru di HSM:

```
aws-cloudhsm > user create --username user1 --role crypto-user --approval /path/
admin.token
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "user1",
    "role": "crypto-user"
  }
}
```

Admin kemudian memasukkan user list perintah untuk mengonfirmasi pembuatan pengguna baru:

```
aws-cloudhsm > user list
{
```

```
"error_code": 0,
"data": {
  "users": [
    {
      "username": "admin",
      "role": "admin",
      "locked": "false",
      "mfa": [],
      "quorum": [
        {
          "strategy": "token-sign",
          "status": "enabled"
        }
      ],
      "cluster-coverage": "full"
    },
    {
      "username": "admin2",
      "role": "admin",
      "locked": "false",
      "mfa": [],
      "quorum": [
        {
          "strategy": "token-sign",
          "status": "enabled"
        }
      ],
      "cluster-coverage": "full"
    },
    {
      "username": "admin3",
      "role": "admin",
      "locked": "false",
      "mfa": [],
      "quorum": [
        {
          "strategy": "token-sign",
          "status": "enabled"
        }
      ],
      "cluster-coverage": "full"
    },
    {
      "username": "admin4",
```

```

    "role": "admin",
    "locked": "false",
    "mfa": [],
    "quorum": [
      {
        "strategy": "token-sign",
        "status": "enabled"
      }
    ],
    "cluster-coverage": "full"
  },
  {
    "username": "user1",
    "role": "crypto-user",
    "locked": "false",
    "mfa": [],
    "quorum": [],
    "cluster-coverage": "full"
  },
  {
    "username": "app_user",
    "role": "internal(APPLIANCE_USER)",
    "locked": "false",
    "mfa": [],
    "quorum": [],
    "cluster-coverage": "full"
  }
]
}

```

Jika admin mencoba melakukan operasi manajemen pengguna HSM lain, itu gagal dengan kesalahan otentikasi kuorum:

```

aws-cloudhsm > user delete --username user1 --role crypto-user
{
  "error_code": 1,
  "data": "Quorum approval is required for this operation"
}

```

Seperti yang ditunjukkan di bawah ini, quorum token-sign list perintah menunjukkan bahwa admin tidak memiliki token yang disetujui. Untuk melakukan operasi manajemen pengguna HSM

lainnya, admin harus membuat token kuorum baru, mendapatkan tanda tangan baru dari pemberi persetujuan, dan menjalankan operasi manajemen pengguna yang diinginkan dengan argumen `--approval` untuk menyediakan token kuorum yang akan disetujui dan digunakan selama eksekusi operasi manajemen pengguna.

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": []
  }
}
```

Ubah nilai minimum kuorum untuk menggunakan AWS CloudHSM CloudHSM CLI

Setelah [menyetel nilai minimum kuorum](#) untuk [admin CloudHSM](#), Anda mungkin perlu menyesuaikan [nilai minimum kuorum](#). HSM memungkinkan perubahan pada nilai minimum kuorum hanya ketika jumlah pemberi persetujuan memenuhi atau melebihi nilai saat ini. Misalnya, dengan nilai minimum kuorum dua (2), setidaknya dua (2) admin harus menyetujui perubahan apa pun.

Note

Nilai kuorum layanan pengguna harus selalu kurang dari atau sama dengan nilai kuorum layanan kuorum. Untuk informasi tentang nama layanan, lihat [Nama dan jenis AWS CloudHSM layanan yang didukung untuk otentikasi kuorum dengan CloudHSM CLI](#).

Untuk mendapatkan persetujuan kuorum untuk mengubah nilai minimum kuorum, Anda memerlukan token kuorum untuk menggunakan perintah `quorum service quorum token-sign set-quorum-value`. Untuk menghasilkan token kuorum untuk untuk quorum service menggunakan `quorum token-sign set-quorum-value` perintah, layanan kuorum harus lebih tinggi dari satu (1). Ini berarti bahwa sebelum Anda dapat mengubah nilai minimum kuorum untuk layanan pengguna, Anda mungkin perlu mengubah nilai minimum kuorum untuk layanan kuorum.

Langkah-langkah untuk mengubah nilai minimum kuorum untuk admin

1. Mulai mode interaktif CloudHSM CLI.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Menggunakan CloudHSM CLI, masuk sebagai admin.

```
aws-cloudhsm > login --username <admin> --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "<admin>",
    "role": "admin"
  }
}
```

3. Periksa nilai minimum kuorum saat ini:

```
aws-cloudhsm > quorum token-sign list-quorum-values
```

4. Jika nilai minimum kuorum untuk layanan kuorum lebih rendah dari nilai untuk layanan pengguna, ubah nilai layanan kuorum:

```
aws-cloudhsm > quorum token-sign set-quorum-value --service quorum --value <3>
```

5. [Hasilkan token kuorum untuk layanan](#) kuorum.
6. [Dapatkan persetujuan \(tanda tangan\) dari](#) admin lain.
7. [Menyetujui token pada klaster CloudHSM dan jalankan operasi manajemen pengguna.](#)
8. Ubah nilai minimum kuorum untuk layanan pengguna:

```
aws-cloudhsm > quorum token-sign set-quorum-value
```

Example Menyesuaikan nilai minimum layanan kuorum

1. Periksa nilai saat ini. Contoh menunjukkan bahwa nilai minimum kuorum untuk layanan pengguna saat ini dua (2).

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 2,
    "quorum": 1
  }
}
```

2. Ubah nilai layanan kuorum. Tetapkan nilai minimum kuorum untuk layanan kuorum ke nilai yang sama atau lebih tinggi dari nilai untuk layanan pengguna. Contoh ini menetapkan nilai minimum kuorum untuk layanan kuorum menjadi dua (2), nilai yang sama yang ditetapkan untuk layanan pengguna pada contoh sebelumnya.

```
aws-cloudhsm > quorum token-sign set-quorum-value --service quorum --value 2
{
  "error_code": 0,
  "data": "Set quorum value successful"
}
```

3. Verifikasi perubahan. Contoh ini menunjukkan bahwa nilai minimum kuorum sekarang dua (2) untuk layanan pengguna dan layanan kuorum.

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 2,
    "quorum": 2
  }
}
```

Manajemen pengguna HSM dengan CloudHSM Management Utility (CMU)

Untuk mengelola pengguna modul keamanan perangkat keras (HSM) AWS CloudHSM, Anda harus masuk ke HSM dengan nama pengguna dan kata sandi [petugas kriptografi](#) (CO). Hanya COs dapat mengelola pengguna. HSM berisi CO default bernama admin. Anda mengatur kata sandi untuk admin ketika Anda [mengaktifkan klaster](#).

Topik ini memberikan step-by-step instruksi dan detail tentang mengelola pengguna HSM dengan AWS CloudHSM Management Utility (CMU).

Topik

- [Prasyarat untuk manajemen pengguna di Utilitas Manajemen AWS CloudHSM](#)
- [Jenis pengguna HSM untuk Utilitas AWS CloudHSM Manajemen](#)
- [Tabel izin pengguna HSM untuk AWS CloudHSM Utilitas Manajemen](#)
- [Buat pengguna HSM menggunakan Utilitas AWS CloudHSM Manajemen](#)
- [Daftar semua pengguna HSM di cluster menggunakan AWS CloudHSM Management Utility](#)
- [Ubah kata sandi pengguna HSM menggunakan Utilitas AWS CloudHSM Manajemen](#)
- [Hapus pengguna HSM menggunakan Utilitas AWS CloudHSM Manajemen](#)
- [Kelola 2FA untuk pengguna yang menggunakan AWS CloudHSM Management Utility](#)
- [Menggunakan CloudHSM Management Utility \(CMU\) untuk mengelola otentikasi kuorum \(M of N access control\)](#)

Prasyarat untuk manajemen pengguna di Utilitas Manajemen AWS CloudHSM

Sebelum Anda menggunakan AWS CloudHSM Management Utility (CMU) untuk mengelola pengguna modul keamanan perangkat keras (HSM) AWS CloudHSM, Anda harus menyelesaikan prasyarat ini. Topik berikut menjelaskan memulai dengan CMU.

Bagian-bagian

- [Dapatkan alamat IP HSM di AWS CloudHSM](#)
- [Menggunakan CMU dengan Client SDK 3.2.1 dan sebelumnya](#)
- [Unduh Utilitas Manajemen CloudHSM](#)

Dapatkan alamat IP HSM di AWS CloudHSM

Untuk menggunakan CMU, Anda harus menggunakan alat konfigurasi untuk memperbarui konfigurasi lokal. CMU membuat koneksi sendiri ke klaster dan koneksi ini tidak sadar klaster. Untuk melacak informasi klaster, CMU mempertahankan file konfigurasi lokal. Ini berarti bahwa setiap kali Anda menggunakan CMU, Anda harus terlebih dahulu memperbarui file konfigurasi dengan menjalankan [konfigurasi](#) alat baris perintah dengan parameter `--cmu`. Jika Anda menggunakan klien SDK 3.2.1 atau sebelumnya, Anda harus menggunakan parameter yang berbeda dari `--cmu`. Untuk informasi lebih lanjut, lihat [the section called “Menggunakan CMU dengan Client SDK 3.2.1 dan sebelumnya”](#).

Parameter `--cmu` mengharuskan Anda untuk menambahkan alamat IP HSM di klaster Anda. Jika Anda memiliki beberapa HSMs, Anda dapat menggunakan alamat IP apa pun. Hal ini memastikan CMU dapat menyebarkan perubahan yang Anda buat di seluruh klaster. Ingat bahwa CMU menggunakan file lokal untuk melacak informasi klaster. Jika klaster telah berubah sejak terakhir kali Anda menggunakan CMU dari host tertentu, Anda harus menambahkan perubahan tersebut ke file konfigurasi lokal yang disimpan di host tersebut. Jangan pernah menambahkan atau menghapus HSM saat Anda menggunakan CMU.

Untuk mendapatkan alamat IP untuk HSM (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Untuk mengubah Wilayah AWS, gunakan pemilih Wilayah di sudut kanan atas halaman.
3. Untuk membuka halaman detail klaster, dalam tabel klaster, pilih ID klaster.
4. Untuk mendapatkan alamat IP, buka HSMs tab. Untuk IPv4 cluster, pilih alamat yang tercantum di bawah IPv4 alamat ENI. Untuk cluster dual-stack gunakan ENI IPv4 atau alamat ENI. IPv6

Untuk mendapatkan alamat IP untuk HSM (AWS CLI)

- Dapatkan alamat IP dari HSM dengan menggunakan perintah [describe-clusters](#) dari AWS CLI. Dalam output dari perintah, alamat IP dari HSMs adalah nilai-nilai `EniIp` dan `EniIPv6` (jika itu adalah cluster dual-stack).

```
$ aws cloudhsmv2 describe-clusters
{
  "Clusters": [
    { ... }
  ],
  "Hsms": [
```

```

    {
    ...
        "EniIp": "10.0.0.9",
    ...
    },
    {
    ...
        "EniIp": "10.0.1.6",
        "EniIpV6": "2600:113f:404:be09:310e:ed34:3412:f733",
    ...
    }

```

Menggunakan CMU dengan Client SDK 3.2.1 dan sebelumnya

Dengan Client SDK 3.3.0, AWS CloudHSM menambahkan dukungan untuk `--cmu` parameter, yang menyederhanakan proses memperbarui file konfigurasi untuk CMU. Jika Anda menggunakan versi CMU dari SDK Klien 3.2.1 atau sebelumnya, Anda harus terus menggunakan parameter `-a` dan `-m` untuk memperbarui file konfigurasi. Untuk informasi selengkapnya tentang parameter ini, lihat [Alat konfigurasi](#).

Unduh Utilitas Manajemen CloudHSM

Versi terbaru CMU tersedia untuk tugas manajemen pengguna HSM apakah Anda menggunakan SDK Klien 5 dan SDK Klien 3.

Untuk mengunduh dan menginstal CMU

- Unduh dan instal CMU.

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-mgmt-util-latest.el6.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-mgmt-util-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-mgmt-util-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-mgmt-util-latest.el7.x86_64.rpm
```

CentOS 7.8+

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-mgmt-util-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-mgmt-util-latest.el7.x86_64.rpm
```

CentOS 8.3+

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-mgmt-util-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-mgmt-util-latest.el8.x86_64.rpm
```

RHEL 7 (7.8+)

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-mgmt-util-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-mgmt-util-latest.el7.x86_64.rpm
```

RHEL 8 (8.3+)

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-mgmt-util-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-mgmt-util-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-mgmt-util_latest_amd64.deb
```

```
$ sudo apt install ./cloudhsm-mgmt-util_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-mgmt-util_latest_u18.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-mgmt-util_latest_u18.04_amd64.deb
```

Windows Server 2012

1. Unduh [Utilitas Manajemen CloudHSM](#)
2. Jalankan penginstal CMU (AWSCloudHSMMManagementUtil-latest.msi) dengan hak administratif Windows.

Windows Server 2012 R2

1. Unduh [Utilitas Manajemen CloudHSM](#)
2. Jalankan penginstal CMU (AWSCloudHSMMManagementUtil-latest.msi) dengan hak administratif Windows.

Windows Server 2016

1. Unduh [Utilitas Manajemen CloudHSM](#)
2. Jalankan penginstal CMU (AWSCloudHSMMManagementUtil-latest.msi) dengan hak administratif Windows.

Jenis pengguna HSM untuk Utilitas AWS CloudHSM Manajemen

Sebagian besar operasi yang Anda lakukan pada modul keamanan perangkat keras (HSM) memerlukan kredensial pengguna HSM. AWS CloudHSM HSM mengautentikasi setiap pengguna HSM dan setiap pengguna HSM memiliki jenis yang menentukan operasi yang dapat Anda lakukan pada HSM sebagai pengguna tersebut.

 Note

Pengguna HSM berbeda dari pengguna IAM. Pengguna IAM yang memiliki kredensial yang benar dapat membuat HSMs dengan berinteraksi dengan sumber daya melalui AWS API. Setelah HSM dibuat, Anda harus menggunakan kredensi pengguna HSM untuk mengautentikasi operasi pada HSM.

Jenis pengguna

- [Petugas prakripto \(PRECO\)](#)
- [Petugas kripto \(CO\)](#)
- [Pengguna kripto \(CU\)](#)
- [Pengguna alat \(AU\)](#)

Petugas prakripto (PRECO)

Baik dalam utilitas manajemen cloud (CMU) dan utilitas manajemen kunci (KMU), PRECO adalah pengguna sementara yang hanya ada pada HSM pertama dalam sebuah cluster. AWS CloudHSM HSM pertama di cluster baru berisi pengguna PRECO yang menunjukkan bahwa cluster ini tidak pernah diaktifkan. Untuk [mengaktifkan cluster](#), Anda menjalankan cloudhsm-cli dan menjalankan perintah `cluster activate` Masuk ke HSM dan ubah kata sandi PRECO. Saat Anda mengubah kata sandi, pengguna ini menjadi petugas kripto (CO).

Petugas kripto (CO)

Baik dalam utilitas manajemen cloud (CMU) dan utilitas manajemen kunci (KMU), petugas kripto (CO) dapat melakukan operasi manajemen pengguna. Misalnya, mereka dapat membuat dan menghapus pengguna dan mengubah kata sandi pengguna. Untuk informasi lebih lanjut tentang pengguna root, lihat [Tabel izin pengguna HSM untuk AWS CloudHSM Utilitas Manajemen](#). Saat Anda mengaktifkan cluster baru, pengguna berubah dari [Precto Officer](#) (PRECO) menjadi crypto officer (CO).

Pengguna kripto (CU)

Pengguna kripto (CU) dapat melakukan manajemen kunci dan operasi kriptografi berikut.

- Manajemen kunci— Buat, hapus, bagikan, impor, dan ekspor kunci kriptografi.

- Operasi kriptografi— Gunakan kunci kriptografi untuk enkripsi, dekripsi, penandatanganan, verifikasi, dan lainnya.

Untuk informasi selengkapnya, lihat [Tabel izin pengguna HSM untuk AWS CloudHSM Utilitas Manajemen](#).

Pengguna alat (AU)

Pengguna alat (AU) dapat melakukan operasi kloning dan sinkronisasi pada kluster Anda. HSMs AWS CloudHSM menggunakan AU untuk menyinkronkan HSMs dalam sebuah AWS CloudHSM cluster. AU ada pada semua yang HSMs disediakan oleh AWS CloudHSM, dan memiliki izin terbatas. Untuk informasi selengkapnya, lihat [Tabel izin pengguna HSM untuk AWS CloudHSM Utilitas Manajemen](#).

AWS tidak dapat melakukan operasi apa pun pada Anda HSMs . AWS tidak dapat melihat atau memodifikasi pengguna atau kunci Anda dan tidak dapat melakukan operasi kriptografi apa pun menggunakan kunci tersebut.

Tabel izin pengguna HSM untuk AWS CloudHSM Utilitas Manajemen

Tabel berikut mencantumkan modul keamanan perangkat keras (HSM (operasi diurutkan berdasarkan jenis pengguna HSM atau sesi yang dapat melakukan operasi di. AWS CloudHSM

	Petugas kriptografi (CO)	Pengguna Kriptografi (CU)	Pengguna Peralatan (AU)	Sesi Tidak Diautentikasi	
Dapatkan informasi kluster dasar ¹		Y 	Y 	Y 	Ya
Ubah kata sandi sendiri		Y 	Y 	Y	Tidak berlaku
Ubah kata sandi pengguna		T 	T 	T 	Tidak

	Petugas krypto (CO)	Pengguna Kripto (CU)	Pengguna Peralatan (AU)	Sesi Tidak Diautentikasi	
Tambah, hapus pengguna		Y	T		Tidak
Dapatkan status sinkronis ²		Y	Y		Tidak
Ekstrak, masukkan benda bertopeng ³		Y	Y		Tidak
Fungsi manajemen kunci		T	Y		Tidak
Enkripsi, dekripsi		T	Y		Tidak
Tanda tangan, verifikasi		T	Y		Tidak
Hasilkan intisari dan HMACs		T	Y		Tidak

- [1] Informasi cluster dasar mencakup jumlah HSMs di cluster dan setiap alamat IP HSM, model, nomor seri, ID perangkat, ID firmware, dll.
- [2] Pengguna bisa mendapatkan satu set intisari (hash) yang sesuai dengan kunci pada HSM. Aplikasi dapat membandingkan kumpulan intisari ini untuk memahami status sinkronisasi HSMs dalam sebuah cluster.

- [3] Objek bertopeng adalah kunci yang dienkripsi sebelum meninggalkan HSM. Objek tidak dapat didekripsi di luar HSM. Objek hanya didekripsi setelah dimasukkan ke dalam HSM yang ada di kluster yang sama dengan HSM asalnya diekstraksi. Aplikasi dapat mengekstrak dan menyisipkan objek bertopeng untuk menyinkronkan HSMs dalam sebuah cluster.
- [4] Fungsi manajemen kunci termasuk membuat, menghapus, membungkus, membuka bungkus, dan memodifikasi atribut kunci.

Buat pengguna HSM menggunakan Utilitas AWS CloudHSM Manajemen

Gunakan `createUser` di AWS CloudHSM Management Utility (CMU) untuk membuat pengguna baru pada modul keamanan perangkat keras (HSM). Anda harus masuk sebagai CO untuk membuat pengguna.

Untuk membuat pengguna baru

1. Gunakan alat konfigurasi untuk memperbarui konfigurasi CMU.

Linux

```
$ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" --cmu <IP address>
```

2. Mulai CMU.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\cloudhsm_mgmt_util.exe" C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_util.cfg
```

3. Masuk ke HSM sebagai pengguna CO.

```
aws-cloudhsm > loginHSM CO admin co12345
```

Pastikan jumlah koneksi daftar CMU cocok dengan jumlah HSMs di cluster. Jika tidak, keluar dan mulai dari awal.

- Gunakan `createUser` untuk membuat pengguna CO bernama **example_officer** dengan kata sandi **password1**.

```
aws-cloudhsm > createUser CO example_officer password1
```

CMU menanyakan kepada Anda tentang operasi membuat pengguna.

```
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****

Do you want to continue(y/n)?
```

- Ketik **y**.

Untuk membuat pengguna baru

- Gunakan alat konfigurasi untuk memperbarui konfigurasi CMU.

Linux

```
$ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" --cmu <IP address>
```

- Mulai CMU.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\cloudhsm_mgmt_util.exe" C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_util.cfg
```

3. Masuk ke HSM sebagai pengguna CO.

```
aws-cloudhsm > loginHSM CO admin co12345
```

Pastikan jumlah koneksi daftar CMU cocok dengan jumlah HSMs di cluster. Jika tidak, keluar dan mulai dari awal.

4. Gunakan `createUser` untuk membuat pengguna CU bernama **example_user** dengan kata sandi **password1**.

```
aws-cloudhsm > createUser CU example_user password1
```

CMU menanyakan kepada Anda tentang operasi membuat pengguna.

```
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****
Do you want to continue(y/n)?
```

5. Ketik **y**.

Untuk informasi selengkapnya tentang `createUser`, lihat [createUser](#).

Daftar semua pengguna HSM di cluster menggunakan AWS CloudHSM Management Utility

Gunakan `listUsers` perintah di AWS CloudHSM Management Utility (CMU) untuk daftar semua pengguna di AWS CloudHSM cluster. Anda tidak perlu masuk untuk menjalankan `listUsers` dan semua jenis pengguna dapat mendaftarkan pengguna.

Untuk mendaftar semua pengguna di klaster

1. Gunakan alat konfigurasi untuk memperbarui konfigurasi CMU.

Linux

```
$ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" --cmu <IP address>
```

2. Mulai CMU.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\cloudhsm_mgmt_util.exe" C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_util.cfg
```

3. Gunakan listUsers untuk mencantumkan semua pengguna di klaster.

```
aws-cloudhsm > listUsers
```

CMU mendaftar semua pengguna di klaster.

Users on server 0(10.0.2.9):

Number of users found:4

User Id	User Type	User Name	
MofnPubKey	LoginFailureCnt	2FA	
1	AU	app_user	NO
0	NO		
2	CO	example_officer	NO
0	NO		
3	CU	example_user	NO
0	NO		

Users on server 1(10.0.3.11):

```
Number of users found:4
```

User Id	User Type	User Name	
MofnPubKey	LoginFailureCnt	2FA	
1	AU	app_user	N0
	0		N0
2	CO	example_officer	N0
	0		N0
3	CU	example_user	N0
	0		N0

```
Users on server 2(10.0.1.12):
```

```
Number of users found:4
```

User Id	User Type	User Name	
MofnPubKey	LoginFailureCnt	2FA	
1	AU	app_user	N0
	0		N0
2	CO	example_officer	N0
	0		N0
3	CU	example_user	N0
	0		N0

Untuk informasi lebih lanjut tentang listUsers, lihat [listUsers](#).

Ubah kata sandi pengguna HSM menggunakan Utilitas AWS CloudHSM Manajemen

Gunakan changePswd di AWS CloudHSM Management Utility (CMU) untuk mengubah kata sandi pengguna modul keamanan perangkat keras (HSM).

Jenis pengguna dan kata sandi adalah sensitif huruf besar-kecil, tetapi nama pengguna tidak sensitif huruf besar-kecil.

CO, Pengguna kriptografi (CU), dan pengguna peralatan (AU) dapat mengubah kata sandi mereka sendiri. Untuk mengubah kata sandi pengguna lain, Anda harus masuk sebagai CO. Anda tidak dapat mengubah kata sandi pengguna yang saat ini login ke klien atau key_mgmt_util.

Untuk mengubah kata sandi Anda sendiri

1. Gunakan alat konfigurasi untuk memperbarui konfigurasi CMU.

Linux

```
$ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" --cmu <IP address>
```

2. Mulai CMU.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\cloudhsm_mgmt_util.exe" C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_util.cfg
```

3. : Masuk ke HSM

```
aws-cloudhsm > loginHSM C0 admin co12345
```

Pastikan jumlah koneksi daftar CMU cocok dengan jumlah HSMs di cluster. Jika tidak, keluar dan mulai dari awal.

4. Gunakan changePswd untuk mengubah kata sandi Anda sendiri.

```
aws-cloudhsm > changePswd C0 example_officer <new password>
```

CMU menanyakan kepada Anda tentang operasi ubah kata sandi.

```
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****
```

```
Do you want to continue(y/n)?
```

5. Ketik **y**.

CMU menanyakan kepada Anda tentang operasi ubah kata sandi.

```
Changing password for example_officer(C0) on 3 nodes
```

: Ubah Kata Sandi Pengguna Lain

1. Gunakan alat konfigurasi untuk memperbarui konfigurasi CMU.

Linux

```
$ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" --cmu <IP address>
```

2. Mulai CMU.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\cloudhsm_mgmt_util.exe" C:\  
ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_util.cfg
```

3. Masuk ke HSM sebagai pengguna CO.

```
aws-cloudhsm > loginHSM C0 admin co12345
```

Pastikan jumlah koneksi daftar CMU cocok dengan jumlah HSMs di cluster. Jika tidak, keluar dan mulai dari awal.

4. Gunakan changePswd untuk mengubah kata sandi pengguna lain.

```
aws-cloudhsm > changePswd CU example_user <new password>
```

CMU menanyakan kepada Anda tentang operasi ubah kata sandi.

```
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****

Do you want to continue(y/n)?
```

5. Ketik **y**.

CMU menanyakan kepada Anda tentang operasi ubah kata sandi.

```
Changing password for example_user(CU) on 3 nodes
```

Untuk informasi lebih lanjut tentang `changePswd`, lihat [changePswd](#).

Hapus pengguna HSM menggunakan Utilitas AWS CloudHSM Manajemen

Gunakan `deleteUser` di AWS CloudHSM Management Utility (CMU) untuk menghapus pengguna modul keamanan perangkat keras (HSM). Anda harus masuk sebagai CO untuk menghapus pengguna lain.

Tip

Anda tidak dapat menghapus pengguna krypto (CU) yang memiliki kunci.

Untuk menghapus kluster

1. Gunakan alat konfigurasi untuk memperbarui konfigurasi CMU.

Linux

```
$ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" --cmu <IP address>
```

2. Mulai CMU.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\cloudhsm_mgmt_util.exe" C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_util.cfg
```

3. Masuk ke HSM sebagai pengguna CO.

```
aws-cloudhsm > loginHSM C0 admin co12345
```

Pastikan jumlah koneksi daftar CMU cocok dengan jumlah HSMs di cluster. Jika tidak, keluar dan mulai dari awal.

4. Gunakan deleteUser untuk menghapus pengguna.

```
aws-cloudhsm > deleteUser C0 example_officer
```

CMU menghapus pengguna.

```
Deleting user example_officer(C0) on 3 nodes
deleteUser success on server 0(10.0.2.9)
deleteUser success on server 1(10.0.3.11)
deleteUser success on server 2(10.0.1.12)
```

Untuk informasi selengkapnya tentang deleteUser, lihat [deleteUser](#).

Kelola 2FA untuk pengguna yang menggunakan AWS CloudHSM Management Utility

Untuk meningkatkan keamanan, Anda dapat mengonfigurasi otentikasi dua faktor (2FA) untuk membantu melindungi klaster. AWS CloudHSM Anda hanya dapat mengaktifkan 2FA untuk petugas kriptografi (CO).

Saat Anda masuk ke klaster dengan akun modul layanan perangkat keras (HSM) yang mendukung 2FA, Anda menyediakan `cloudhsm_mgmt_util` (CMU) dengan kata sandi Anda—faktor pertama, apa yang Anda tahu—dan CMU memberi Anda token dan meminta Anda untuk menandatangani token. Untuk menyediakan faktor kedua—apa yang Anda miliki—Anda tandatangani token dengan kunci privat dari pasangan kunci yang telah Anda buat dan terkait dengan pengguna HSM. Untuk mengakses klaster, Anda memberikan token ditandatangani untuk CMU.

Note

Anda tidak dapat mengaktifkan 2FA untuk pengguna kriptografi (CU) atau aplikasi. Autentikasi dua faktor (2FA) hanya untuk pengguna CO.

Topik

- [Otentikasi kuorum dan 2FA dalam AWS CloudHSM cluster menggunakan Management Utility AWS CloudHSM](#)
- [Persyaratan 2FA key pair untuk AWS CloudHSM menggunakan AWS CloudHSM Management Utility](#)
- [Buat pengguna dengan 2FA diaktifkan untuk pengguna AWS CloudHSM Management Utility](#)
- [Kelola 2FA untuk pengguna HSM menggunakan AWS CloudHSM Management Utility](#)
- [Nonaktifkan 2FA untuk pengguna HSM menggunakan Utilitas Manajemen AWS CloudHSM](#)
- [Referensi konfigurasi untuk 2FA dengan Utilitas AWS CloudHSM Manajemen](#)

Otentikasi kuorum dan 2FA dalam AWS CloudHSM cluster menggunakan Management Utility AWS CloudHSM

Cluster menggunakan kunci yang sama untuk otentikasi kuorum dan untuk otentikasi dua faktor (2FA). Ini berarti pengguna dengan 2FA diaktifkan secara efektif terdaftar untuk M-of-N-access -control

(mOFN). Untuk berhasil menggunakan autentikasi 2FA dan kuorum untuk pengguna HSM yang sama, pertimbangkan hal-hal berikut:

- Jika Anda menggunakan autentikasi kuorum untuk pengguna saat ini, Anda harus menggunakan pasangan kunci yang sama yang Anda buat untuk pengguna kuorum untuk mengaktifkan 2FA untuk pengguna tersebut.
- Jika Anda menambahkan persyaratan 2FA untuk pengguna non-2FA yang bukan pengguna autentikasi kuorum, maka Anda mendaftarkan pengguna itu sebagai pengguna MofN dengan autentikasi 2FA.
- Jika Anda menghapus persyaratan 2FA atau mengubah kata sandi untuk pengguna 2FA yang juga pengguna autentikasi kuorum, Anda juga akan menghapus pendaftaran pengguna kuorum sebagai pengguna MofN.
- Jika Anda menghapus persyaratan 2FA atau mengubah kata sandi untuk pengguna 2FA yang juga pengguna autentikasi kuorum, tetapi Anda masih ingin pengguna tersebut berpartisipasi dalam autentikasi kuorum, maka Anda harus mendaftarkan pengguna tersebut lagi sebagai pengguna MofN.

Untuk informasi selengkapnya tentang autentikasi kuorum, lihat [Menggunakan CMU untuk mengelola otentikasi kuorum](#).

Persyaratan 2FA key pair untuk AWS CloudHSM menggunakan AWS CloudHSM Management Utility

Untuk mengaktifkan otentikasi dua faktor (2FA) untuk pengguna modul keamanan AWS CloudHSM perangkat keras (HSM), gunakan kunci yang memenuhi persyaratan berikut.

Anda dapat membuat pasangan kunci baru atau menggunakan kunci yang sudah ada yang memenuhi persyaratan berikut.

- Tipe kunci: Asimetris
- Penggunaan kunci: Tanda Tangan dan Verifikasi
- Spesifikasi kunci: RSA_2048
- Algoritme penandatanganan meliputi:
 - sha256WithRSAEncryption

Note

Jika Anda menggunakan autentikasi kuorum atau berencana untuk menggunakan autentikasi kuorum, lihat [the section called “Otentikasi kuorum”](#).

Buat pengguna dengan 2FA diaktifkan untuk pengguna AWS CloudHSM Management Utility

Gunakan AWS CloudHSM Management Utility CMU (CMU) dan key pair untuk membuat pengguna kantor kriptografi (CO) baru dengan autentikasi dua faktor (2FA) diaktifkan.

Untuk membuat pengguna CO dengan 2FA diaktifkan

1. Dalam satu terminal, lakukan langkah-langkah berikut:
 - a. Akses HSM Anda dan masuk ke utilitas Manajemen CloudHSM:

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

- b. Masuk sebagai CO dan gunakan perintah berikut untuk membuat MFA pengguna baru dengan 2FA:

```
aws-cloudhsm > createUser CO MFA <CO USER PASSWORD> -2fa /home/ec2-user/
authdata
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****

Do you want to continue(y/n)? y

Creating User exampleuser3(CO) on 1 nodesAuthentication data written to: "/
home/ec2-user/authdata"Generate Base64-encoded signatures for SHA256 digests in
the authentication datafile.
To generate the signatures, use the RSA private key, which is the second factor
ofauthentication for this user. Paste the signatures and the corresponding
public keyinto the authentication data file and provide
```

the file path below. Leave this field blank to use the path initially provided. Enter filename:

- c. Biarkan terminal di atas dalam keadaan ini. Jangan tekan enter atau masukkan nama file apa pun.

2. Di terminal lain, lakukan langkah-langkah berikut:

- a. Akses HSM Anda dan masuk ke utilitas Manajemen CloudHSM:

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

- b. Buat pasangan kunci publik-pribadi menggunakan perintah berikut:

```
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt  
rsa_keygen_bits:2048
```

```
openssl rsa -pubout -in private_key.pem -out public_key.pem
```

- c. Jalankan perintah berikut untuk menginstal fitur kueri json untuk mengekstrak Digest dari file authdata:

```
sudo yum install jq
```

- d. Untuk mengekstrak nilai intisari, pertama-tama temukan data berikut di file authdata:

```
{  
  "Version": "1.0",  
  "PublicKey": "",  
  "Data": [  
    {  
      "HsmId": <"HSM ID">,  
      "Digest": <"DIGEST">,  
      "Signature": ""  
    }  
  ]  
}
```


 Note

Digest yang diperoleh dikodekan base64, namun untuk menandatangani intisari, Anda perlu file yang akan diterjemahkan terlebih dahulu dan kemudian ditandatangani. Perintah berikut akan memecahkan kode intisari dan menyimpan konten yang diterjemahkan dalam 'digest1.bin'

```
cat authdata | jq '.Data[0].Digest' | cut -c2- | rev | cut -c2- | rev |
base64 -d > digest1.bin
```

- e. Konversikan konten kunci publik, tambahkan "\n" dan hapus spasi seperti yang ditunjukkan di sini:

```
-----BEGIN PUBLIC KEY-----\n<PUBLIC KEY>\n-----END PUBLIC KEY-----
```

 Important

Perintah di atas menunjukkan bagaimana "\n" ditambahkan segera setelah BEGIN PUBLIC KEY-----, spasi antara "\n" dan karakter pertama dari kunci publik dihapus, "\n" ditambahkan sebelumnya-----END PUBLIC KEY, dan spasi dihapus antara "\n" dan akhir kunci publik.

Ini adalah format PEM untuk kunci publik yang diterima dalam file authdata.

- f. Rekatkan konten format pem kunci publik di bagian kunci publik di file authdata.

```
vi authdata
```

```
{
  "Version": "1.0",
  "PublicKey": "-----BEGIN PUBLIC KEY-----\n<PUBLIC KEY>\n-----END PUBLIC
KEY-----",
  "Data": [
    {
      "HsmId": "<HSM ID>",
      "Digest": "<DIGEST>",
      "Signature": ""
```

```
}
]
}
```

- g. Tanda tangani file token menggunakan perintah berikut:

```
openssl pkeyutl -sign -in digest1.bin -inkey private_key.pem -pkeyopt
digest:sha256 | base64
Output Expected:
<"THE SIGNATURE">
```

 Note

Seperti yang ditunjukkan pada perintah di atas, gunakan openssl pkeyutl bukan openssl dgst untuk penandatanganan.

- h. Tambahkan intisari yang ditandatangani di File Authdata di bidang "Tanda Tangan".

```
vi authdata
```

```
{
  "Version": "1.0",
  "PublicKey": "-----BEGIN PUBLIC KEY----- ... -----END PUBLIC KEY-----",
  "Data": [
    {
      "HsmId": <"HSM ID">,
      "Digest": <"DIGEST">,
      "Signature": <"Kkd1 ... rkrvJ6Q==">
    },
    {
      "HsmId": <"HSM ID">,
      "Digest": <"DIGEST">,
      "Signature": <"K1hxy ... Q261Q==">
    }
  ]
}
```

3. Kembali ke terminal pertama dan tekan **Enter**:

Generate Base64-encoded signatures for SHA256 digests in the authentication datafile. To generate the signatures, use the RSA private key,

which is the second factor of authentication for this user. Paste the signatures and the corresponding public key into the authentication data file and provide the file path below. Leave this field blank to use the path initially provided.

Enter filename: >>>> Press Enter here

```
createUser success on server 0(10.0.1.11)
```

Kelola 2FA untuk pengguna HSM menggunakan AWS CloudHSM Management Utility

Gunakan `changePswd` di AWS CloudHSM Management Utility (CMU) untuk memodifikasi otentikasi dua faktor (2FA) untuk pengguna. Setiap kali Anda mengaktifkan 2FA, Anda harus menyediakan kunci publik untuk login 2FA.

`changePswd` melakukan salah satu skenario berikut:

- Mengubah kata sandi untuk pengguna 2FA
- Mengubah kata sandi untuk pengguna non-2FA
- Tambahkan 2FA ke pengguna non-2FA
- Hapus 2FA dari pengguna 2FA
- Memutar kunci untuk pengguna 2FA

Anda juga dapat menggabungkan tugas. Misalnya, Anda dapat menghapus 2FA dari pengguna dan mengubah kata sandi pada saat yang sama, atau Anda mungkin memutar kunci 2FA dan mengubah kata sandi pengguna.

Untuk mengubah password atau memutar kunci untuk pengguna CO dengan 2FA diaktifkan

1. Gunakan CMU untuk masuk ke HSM sebagai CO dengan 2FA diaktifkan.
2. Gunakan `changePswd` untuk mengubah kata sandi atau memutar kunci dari pengguna CO dengan 2FA diaktifkan. Gunakan parameter `-2fa` dan sertakan lokasi dalam sistem file agar sistem menulis file `authdata`. File ini termasuk digest untuk setiap HSM di kluster.

```
aws-cloudhsm > changePswd CO example-user <new-password> -2fa /path/to/authdata
```

CMU meminta Anda untuk menggunakan kunci privat untuk menandatangani digest di file `authdata` dan mengembalikan tanda tangan dengan kunci publik.

- Gunakan kunci privat untuk menandatangani digest di file authdata, tambahkan tanda tangan dan kunci publik untuk file authdata berformat JSON dan kemudian memberikan CMU dengan lokasi file authdata. Untuk informasi lebih lanjut, lihat [the section called “Referensi konfigurasi”](#).

 Note

Klaster menggunakan kunci yang sama untuk autentikasi kuorum dan 2FA. Jika Anda menggunakan autentikasi kuorum atau berencana untuk menggunakan autentikasi kuorum, lihat [the section called “Otentikasi kuorum”](#).

Nonaktifkan 2FA untuk pengguna HSM menggunakan Utilitas Manajemen AWS CloudHSM

Gunakan AWS CloudHSM Management Utility (CMU) untuk menonaktifkan otentikasi dua faktor (2FA) untuk pengguna modul keamanan perangkat keras HSM) di. AWS CloudHSM

Untuk menonaktifkan 2FA untuk pengguna CO dengan 2FA diaktifkan

- Gunakan CMU untuk masuk ke HSM sebagai CO dengan 2FA diaktifkan.
- Gunakan `changePswd` untuk menghapus 2FA dari pengguna CO dengan 2FA diaktifkan.

```
aws-cloudhsm > changePswd CO example-user <new password>
```

CMU meminta Anda untuk mengonfirmasi operasi perubahan kata sandi.

 Note

Jika Anda menghapus persyaratan 2FA atau mengubah kata sandi untuk pengguna 2FA yang juga pengguna autentikasi kuorum, Anda juga akan menghapus pendaftaran pengguna kuorum sebagai pengguna MofN. Untuk informasi selengkapnya tentang pengguna kuorum dan 2FA, lihat [the section called “Otentikasi kuorum”](#).

- Ketik **y**.

CMU mengonfirmasi operasi perubahan kata sandi.

Referensi konfigurasi untuk 2FA dengan Utilitas AWS CloudHSM Manajemen

Berikut ini adalah contoh properti otentikasi dua faktor (2FA) dalam authdata file untuk permintaan yang dihasilkan AWS CloudHSM Management Utility (CMU) dan tanggapan Anda.

```
{
  "Version": "1.0",
  "PublicKey": "-----BEGIN PUBLIC KEY----- ... -----END PUBLIC KEY-----",
  "Data": [
    {
      "HsmId": "hsm-lgavqitns2a",
      "Digest": "k501p3f6foQRVQH7S8Rrjcau6h3TYqsSdr16A54+qG8=",
      "Signature": "Kkd1 ... rkrvJ6Q=="
    },
    {
      "HsmId": "hsm-lgavqitns2a",
      "Digest": "IyBcx4I5Vyx1jztwvXinCBQd9lDx8oQe7iRrWjBAi1w=",
      "Signature": "K1hxy ... Q261Q=="
    }
  ]
}
```

Data

Simpul tingkat atas. Berisi simpul bawahan untuk setiap HSM di klaster. Muncul dalam permintaan dan tanggapan untuk semua perintah 2FA.

Intisari

Ini adalah apa yang harus Anda tanda tangani untuk memberikan faktor kedua autentikasi. CMU dihasilkan dalam permintaan untuk semua perintah 2FA.

HsmId

ID dari HSM Anda. Muncul dalam permintaan dan tanggapan untuk semua perintah 2FA.

PublicKey

Bagian kunci publik dari pasangan kunci yang Anda buat dimasukkan sebagai string berformat PEM. Anda memasukkan ini dalam tanggapan untuk createUser dan changePswd.

Tanda tangan

Digest bertanda tangan dengan encode Base 64. Anda memasukkan ini dalam tanggapan untuk semua perintah 2FA.

Versi

Versi file data autentikasi berformat JSON. Muncul dalam permintaan dan tanggapan untuk semua perintah 2FA.

Menggunakan CloudHSM Management Utility (CMU) untuk mengelola otentikasi kuorum (M of N access control)

HSMs Dalam AWS CloudHSM klaster Anda mendukung otentikasi kuorum, yang juga dikenal sebagai M of N access control. Dengan autentikasi kuorum, tidak ada pengguna tunggal di HSM dapat melakukan operasi dikendalikan kuorum pada HSM. Sebaliknya, sejumlah minimum pengguna HSM (paling sedikit 2) harus bekerja sama untuk melakukan operasi ini. Dengan autentikasi kuorum, Anda dapat menambahkan lapisan perlindungan ekstra dengan meminta persetujuan dari lebih dari satu pengguna HSM.

Autentikasi kuorum dapat mengontrol operasi berikut:

- Manajemen pengguna HSM oleh [petugas krypto \(COs\)](#) — Membuat dan menghapus pengguna HSM, dan mengubah kata sandi pengguna HSM yang berbeda. Untuk informasi selengkapnya, lihat [Manajemen pengguna dengan otentikasi kuorum diaktifkan untuk Utilitas Manajemen AWS CloudHSM](#).

Perhatikan informasi tambahan berikut tentang menggunakan autentikasi kuorum di AWS CloudHSM.

- Pengguna HSM dapat menandatangani token kuorum mereka sendiri—yaitu, pengguna yang meminta dapat memberikan salah satu persetujuan yang diperlukan untuk otentikasi kuorum.
- Anda memilih jumlah minimum pemberi persetujuan kuorum untuk operasi yang dikontrol kuorum. Angka terkecil yang dapat Anda pilih adalah dua (2), dan angka terbesar yang dapat Anda pilih adalah delapan (8).
- HSM dapat menyimpan hingga 1024 token kuorum. Jika HSM sudah memiliki 1024 token ketika Anda mencoba untuk membuat yang baru, HSM membersihkan salah satu token kedaluwarsa. Secara default, token kedaluwarsa sepuluh menit setelah pembuatannya.
- Klaster menggunakan kunci yang sama untuk autentikasi kuorum dan autentikasi dua faktor (2FA). Untuk informasi selengkapnya tentang penggunaan autentikasi kuorum dan 2FA, lihat [Autentikasi Kuorum dan 2FA](#).

Topik berikut ini menyediakan informasi lebih lanjut tentang autentikasi kuorum di AWS CloudHSM.

Topik

- [Proses otentikasi kuorum untuk Utilitas Manajemen AWS CloudHSM](#)
- [Siapkan otentikasi kuorum untuk petugas krypto AWS CloudHSM](#)
- [Manajemen pengguna dengan otentikasi kuorum diaktifkan untuk Utilitas Manajemen AWS CloudHSM](#)
- [Ubah nilai minimum kuorum dengan AWS CloudHSM Management Utility](#)

Proses otentikasi kuorum untuk Utilitas Manajemen AWS CloudHSM

Langkah-langkah berikut merangkum proses autentikasi kuorum. Untuk langkah-langkah dan alat tertentu, lihat [Manajemen pengguna dengan otentikasi kuorum diaktifkan untuk Utilitas Manajemen AWS CloudHSM](#).

1. Setiap pengguna HSM membuat kunci asimetris untuk penandatanganan. Mereka melakukan ini di luar HSM, berhati-hati untuk melindungi kunci dengan tepat.
2. Setiap pengguna HSM masuk ke HSM dan mendaftarkan bagian publik dari kunci penandatanganan mereka (kunci publik) dengan HSM.
3. Ketika pengguna HSM ingin melakukan operasi yang dikendalikan kuorum, setiap pengguna masuk ke HSM dan mendapatkan token kuorum.
4. Pengguna HSM memberikan token kuorum ke satu atau lebih pengguna HSM lainnya dan meminta persetujuan mereka.
5. Pengguna HSM lainnya menyetujui dengan menggunakan kunci mereka untuk secara kriptografi menandatangani token kuorum. Hal ini terjadi di luar HSM.
6. Ketika pengguna HSM memiliki jumlah persetujuan yang diperlukan, pengguna yang sama masuk ke HSM dan memberikan token kuorum dan persetujuan (tanda tangan) ke HSM.
7. HSM menggunakan kunci publik terdaftar dari setiap penandatanganan untuk memverifikasi tanda tangan. Jika tanda tangan valid, HSM menyetujui token.
8. Pengguna HSM sekarang dapat melakukan operasi dikendalikan kuorum.

Siapkan otentikasi kuorum untuk petugas krypto AWS CloudHSM

Topik berikut menjelaskan langkah-langkah yang harus Anda selesaikan untuk mengonfigurasi modul keamanan perangkat keras (HSM) Anda sehingga [petugas AWS CloudHSM krypto \(COs\)](#)

dapat menggunakan otentikasi kuorum. Anda perlu melakukan langkah-langkah ini hanya sekali ketika Anda pertama kali mengonfigurasi otentikasi kuorum untuk COs. Setelah Anda menyelesaikan langkah ini, lihat [Manajemen pengguna dengan otentikasi kuorum diaktifkan untuk Utilitas Manajemen AWS CloudHSM](#).

Topik

- [Prasyarat](#)
- [Langkah 1. Buat dan daftarkan kunci untuk penandatanganan](#)
- [Langkah 2. Tetapkan nilai minimum kuorum pada HSM](#)

Prasyarat

Untuk memahami contoh ini, Anda harus familier dengan [cloudhsm_mgmt_util alat baris perintah \(CMU\)](#). Dalam contoh ini, AWS CloudHSM cluster memiliki dua HSMs, masing-masing dengan yang sama COs, seperti yang ditunjukkan pada output berikut dari listUsers perintah. Untuk informasi selengkapnya tentang membuat pengguna, lihat [Pengguna HSM](#).

```
aws-cloudhsm > listUsers
```

```
Users on server 0(10.0.2.14):
```

```
Number of users found:7
```

User Id	User Type	User Name	MofnPubKey
LoginFailureCnt	2FA		
1	PREC0	admin	N0
0	N0		
2	AU	app_user	N0
0	N0		
3	C0	officer1	N0
0	N0		
4	C0	officer2	N0
0	N0		
5	C0	officer3	N0
0	N0		
6	C0	officer4	N0
0	N0		
7	C0	officer5	N0
0	N0		

```
Users on server 1(10.0.1.4):
```

```
Number of users found:7
```


User Id	User Type	User Name	MofnPubKey
LoginFailureCnt	2FA		
1	PRECO	admin	N0
0	N0		
2	AU	app_user	N0
0	N0		
3	CO	officer1	N0
0	N0		
4	CO	officer2	N0
0	N0		
5	CO	officer3	N0
0	N0		
6	CO	officer4	N0
0	N0		
7	CO	officer5	N0
0	N0		

Langkah 1. Buat dan daftarkan kunci untuk penandatanganan

Untuk menggunakan autentikasi kuorum, setiap CO harus melakukan semualangkah berikut:

Topik

- [Buat key pair RSA](#)
- [Membuat dan menandatangani token pendaftaran](#)
- [Daftarkan kunci publik dengan HSM](#)

Buat key pair RSA

Ada banyak cara berbeda untuk membuat dan melindungi pasangan kunci. Contoh berikut menunjukkan cara melakukannya dengan [OpenSSL](#).

Example— Membuat kunci privat dengan OpenSSL

Contoh berikut menunjukkan bagaimana menggunakan OpenSSL untuk membuat kunci RSA 2048-bit yang dilindungi oleh frase sandi. Untuk menggunakan contoh ini, ganti *officer1.key* dengan nama file tempat Anda ingin menyimpan kunci.

```
$ openssl genrsa -out <officer1.key> -aes256 2048
Generating RSA private key, 2048 bit long modulus
.....+++
```

```
.+++  
e is 65537 (0x10001)  
Enter pass phrase for officer1.key:  
Verifying - Enter pass phrase for officer1.key:
```

Berikutnya, hasilkan kunci publik menggunakan kunci privat yang baru saja Anda buat.

Example— Membuat kunci privat dengan OpenSSL

Contoh berikut menunjukkan bagaimana menggunakan OpenSSL untuk membuat kunci publik dari kunci privat yang baru saja Anda buat.

```
$ openssl rsa -in officer1.key -outform PEM -pubout -out officer1.pub  
Enter pass phrase for officer1.key:  
writing RSA key
```

Membuat dan menandatangani token pendaftaran

Anda membuat token dan menandatangani dengan kunci privat yang baru saja Anda hasilkan pada langkah sebelumnya.

Example— Buat token

Token pendaftaran hanyalah sebuah file dengan data acak yang tidak melebihi ukuran maksimum 245 byte. Anda menandatangani token dengan kunci privat untuk menunjukkan bahwa Anda memiliki akses ke kunci privat. Perintah berikut menggunakan echo untuk mengalihkan string ke file.

```
$ echo <token to be signed> > officer1.token
```

Tanda tangani token dan simpan ke file tanda tangan. Anda akan membutuhkan token ditandatangani, token belum ditandatangani, dan kunci publik untuk mendaftarkan CO sebagai pengguna MofN dengan HSM.

Example— Tandatangani token

Gunakan OpenSSL dan kunci privat untuk menandatangani token pendaftaran dan membuat file tanda tangan.

```
$ openssl dgst -sha256 \
```

```
-sign officer1.key \
-out officer1.token.sig officer1.token
```

Daftarkan kunci publik dengan HSM

Setelah membuat kunci, CO harus mendaftarkan bagian publik dari kunci (kunci publik) dengan HSM.

Untuk mendaftarkan kunci publik dengan HSM

1. Gunakan perintah berikut untuk memulai alat baris perintah cloudhsm_mgmt_util.

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

2. Gunakan perintah loginHSM untuk masuk ke HSM sebagai pengguna CO. Untuk informasi selengkapnya, lihat [???](#).
3. Gunakan perintah [registerQuorumPubKey](#) untuk mendaftarkan kunci publik. Untuk informasi selengkapnya, lihat contoh berikut atau gunakan perintah help registerQuorumPubKey.

Example— Daftarkan kunci publik dengan HSM

Contoh berikut menunjukkan cara menggunakan perintah registerQuorumPubKey di alat baris perintah cloudhsm_mgmt_util untuk mendaftarkan kunci publik CO dengan HSM. Untuk menggunakan perintah ini, CO harus login ke HSM. Ganti nilai-nilai ini dengan nilai Anda sendiri:

```
aws-cloudhsm > registerQuorumPubKey
CO <officer1> <officer1.token> <officer1.token.sig> <officer1.pub>
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****

Do you want to continue(y/n)? y
registerQuorumPubKey success on server 0(10.0.2.14)
```

<officer1.token>

Jalur ke file yang berisi token pendaftaran belum ditandatangani. Dapat memiliki data acak ukuran file max 245 byte.

Wajib: Ya

<officer1.token.sig>

Path ke file yang berisi mekanisme SHA256 _PKCS ditandatangani hash dari token pendaftaran.

Wajib: Ya

<officer1.pub>

Jalur ke file yang berisi kunci publik dari pasangan kunci RSA-2048 asimetris. Gunakan kunci privat untuk menandatangani token pendaftaran.

Wajib: Ya

Setelah semua COs mendaftarkan kunci publik mereka, output dari listUsers perintah menunjukkan ini di MofnPubKey kolom, seperti yang ditunjukkan pada contoh berikut.

```
aws-cloudhsm > listUsers
```

```
Users on server 0(10.0.2.14):
```

```
Number of users found:7
```

User Id LoginFailureCnt	User Type 2FA	User Name	MofnPubKey
1	PRECO	admin	NO
0	NO		
2	AU	app_user	NO
0	NO		
3	CO	officer1	YES
0	NO		
4	CO	officer2	YES
0	NO		
5	CO	officer3	YES
0	NO		
6	CO	officer4	YES
0	NO		
7	CO	officer5	YES
0	NO		

```
Users on server 1(10.0.1.4):
```

```
Number of users found:7
```

User Id LoginFailureCnt	User Type 2FA	User Name	MofnPubKey

1	PRECO	admin	NO
0	NO		
2	AU	app_user	NO
0	NO		
3	CO	officer1	YES
0	NO		
4	CO	officer2	YES
0	NO		
5	CO	officer3	YES
0	NO		
6	CO	officer4	YES
0	NO		
7	CO	officer5	YES
0	NO		

Langkah 2. Tetapkan nilai minimum kuorum pada HSM

Untuk menggunakan otentikasi kuorum COs, CO harus masuk ke HSM dan kemudian menetapkan nilai minimum kuorum, juga dikenal sebagai nilai *m*. Ini adalah jumlah minimum persetujuan CO yang diperlukan untuk melakukan operasi manajemen pengguna HSM. Setiap CO pada HSM dapat menetapkan nilai minimum kuorum, termasuk COs yang belum mendaftarkan kunci untuk penandatanganan. Anda dapat mengubah nilai minimum kuorum kapan saja; untuk informasi lebih lanjut, lihat [Ubah nilai minimum](#).

Untuk menetapkan nilai minimum kuorum pada HSM

1. Gunakan perintah berikut untuk memulai alat baris perintah `cloudhsm_mgmt_util`.

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

2. Gunakan perintah `loginHSM` untuk masuk ke HSM sebagai pengguna CO. Untuk informasi selengkapnya, lihat [???](#).
3. Gunakan perintah `setMValue` untuk menetapkan nilai minimum kuorum. Untuk informasi selengkapnya, lihat contoh berikut atau gunakan perintah `help setMValue`.

Example— Tetapkan nilai minimum kuorum pada HSM

Contoh ini menggunakan nilai minimum kuorum dua. Anda dapat memilih nilai dari dua (2) hingga delapan (8), hingga jumlah total COs pada HSM. Dalam contoh ini, HSM memiliki enam COs, sehingga nilai maksimum yang mungkin adalah enam.

Untuk menggunakan perintah contoh berikut, ganti angka akhir (2) dengan nilai minimum kuorum yang disukai.

```
aws-cloudhsm > setMValue 3 <2>
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****

Do you want to continue(y/n)? y
Setting M Value(2) for 3 on 2 nodes
```

Dalam contoh sebelumnya, nomor pertama (3) mengidentifikasi layanan HSM yang nilai minimum kuorumnya Anda tetapkan.

Tabel berikut mencantumkan pengenalan layanan HSM bersama dengan nama, deskripsi, dan perintah yang disertakan dalam layanan.

Pengenalan Layanan	Nama Layanan	Deskripsi Layanan	Perintah HSM
3	USER_MGMT	Manajemen pengguna HSM	• createUser • deleteUser • changePswd (hanya berlaku saat mengganti kata sandi pengguna HSM yang berbeda)
4	MISC_CO	Layanan CO lainnya	• setMValue

Untuk mendapatkan nilai minimum kuorum untuk layanan, gunakan perintah getMValue, seperti dalam contoh berikut.

```
aws-cloudhsm > getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

Output dari perintah `getMValue` sebelumnya menunjukkan bahwa nilai minimum kuorum untuk operasi manajemen pengguna HSM (layanan 3) sekarang dua.

Setelah Anda menyelesaikan langkah ini, lihat [Manajemen pengguna dengan otentikasi kuorum diaktifkan untuk Utilitas Manajemen AWS CloudHSM](#).

Manajemen pengguna dengan otentikasi kuorum diaktifkan untuk Utilitas Manajemen AWS CloudHSM

[Petugas AWS CloudHSM krypto \(CO\)](#) pada modul keamanan perangkat keras (HSM) dapat mengonfigurasi otentikasi kuorum untuk operasi berikut di HSM:

- Membuat pengguna HSM
- Membuat pengguna HSM
- Mengubah kata sandi pengguna HSM lainnya

Setelah HSM dikonfigurasi untuk otentikasi kuorum, COs tidak dapat melakukan operasi manajemen pengguna HSM sendiri. Contoh berikut menunjukkan output ketika CO mencoba untuk membuat pengguna baru di HSM. Perintah gagal dengan kesalahan `RET_MXN_AUTH_FAILED`, yang menunjukkan bahwa autentikasi kuorum gagal.

```
aws-cloudhsm > createUser CU user1 password
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****

Do you want to continue(y/n)? y
Creating User user1(CU) on 2 nodes
createUser failed: RET_MXN_AUTH_FAILED
creating user on server 0(10.0.2.14) failed

Retry/Ignore/Abort?(R/I/A): A
```

Untuk melakukan operasi manajemen pengguna HSM, CO harus menyelesaikan tugas berikut:

1. [Dapatkan token kuorum](#).

2. [Dapatkan persetujuan \(tanda tangan\)](#) dari orang lain. COs
3. [Setujui token pada HSM.](#)
4. [Melakukan operasi manajemen pengguna HSM.](#)

Jika Anda belum mengonfigurasi HSM untuk otentikasi kuorum COs, lakukan itu sekarang. Untuk informasi selengkapnya, lihat [Pengaturan pertama kali](#).

Langkah 1. Dapatkan token kuorum

Pertama, CO harus menggunakan alat baris perintah `cloudhsm_mgmt_util` untuk meminta token kuorum.

Dapatkan token kuorum.

1. Gunakan perintah berikut untuk memulai alat baris perintah `cloudhsm_mgmt_util`.

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

2. Gunakan perintah `loginHSM` untuk masuk ke HSM sebagai pengguna CO. Untuk informasi selengkapnya, lihat [???](#).
3. Gunakan `getToken` untuk mendapatkan token kuorum. Untuk informasi selengkapnya, lihat contoh berikut atau gunakan perintah `help getToken`.

Example— Dapatkan token kuorum

Contoh ini mendapat token kuorum untuk CO dengan nama pengguna `officer1` dan menyimpan token ke file bernama `officer1.token`. Untuk menggunakan perintah contoh berikut, ganti nilai ini dengan nilai Anda sendiri:

- ***officer1***— Nama CO yang mendapatkan token. Ini harus CO yang sama yang login ke HSM dan menjalankan perintah ini.
- ***officer1.token***— Nama file yang akan digunakan untuk menyimpan token kuorum.

Dalam perintah berikut, 3 mengidentifikasi layanan yang Anda dapat gunakan dengan token yang Anda dapatkan. Dalam hal ini, token adalah untuk operasi manajemen pengguna HSM (layanan 3). Untuk informasi selengkapnya, lihat [Langkah 2. Tetapkan nilai minimum kuorum pada HSM](#).

```
aws-cloudhsm > getToken 3 officer1 officer1.token
```



```
getToken success on server 0(10.0.2.14)
Token:
Id:1
Service:3
Node:1
Key Handle:0
User:officer1
getToken success on server 1(10.0.1.4)
Token:
Id:1
Service:3
Node:0
Key Handle:0
User:officer1
```

Langkah 2. Dapatkan tanda tangan dari menyetujui COs

CO yang memiliki token kuorum harus mendapatkan token yang disetujui oleh orang lain. COs Untuk memberikan persetujuan mereka, yang lain COs menggunakan kunci penandatanganan mereka untuk menandatangani token secara kriptografis. Mereka melakukan ini di luar HSM.

Ada banyak cara yang berbeda untuk menandatangani token. Contoh berikut menunjukkan cara melakukannya dengan [OpenSSL](#). Untuk menggunakan alat penandatanganan yang berbeda, pastikan bahwa alat menggunakan kunci privat CO (kunci penandatanganan) untuk menandatangani SHA-256 digest token .

Example— Dapatkan tanda tangan dari menyetujui COs

Dalam contoh ini, CO yang memiliki token (officer1) membutuhkan setidaknya dua persetujuan. Contoh perintah berikut menunjukkan bagaimana dua COs dapat menggunakan OpenSSL untuk menandatangani token secara kriptografis.

Pada perintah pertama, officer1 menandatangani tokennya sendiri. Untuk menggunakan perintah contoh berikut, ganti nilai ini dengan nilai Anda sendiri:

- *officer1.key* dan *officer2.key* — Nama file yang berisi kunci penandatanganan CO.
- *officer1.token.sig1* dan *officer1.token.sig2* — Nama file yang akan digunakan untuk menyimpan tanda tangan. Pastikan untuk menyimpan setiap tanda tangan dalam file yang berbeda.
- *officer1.token* — Nama file yang berisi token yang ditandatangani CO.

```
$ openssl dgst -sha256 -sign officer1.key -out officer1.token.sig1 officer1.token
Enter pass phrase for officer1.key:
```

Pada perintah berikut, officer2 menandatangani token yang sama.

```
$ openssl dgst -sha256 -sign officer2.key -out officer1.token.sig2 officer1.token
Enter pass phrase for officer2.key:
```

Langkah 3. Menyetujui token yang ditandatangani di HSM

Setelah CO mendapatkan jumlah minimum persetujuan (tanda tangan) dari yang lain COs, ia harus menyetujui token yang ditandatangani pada HSM.

Untuk menyetujui token yang ditandatangani pada HSM

1. Buat file persetujuan token. Untuk informasi selengkapnya, lihat contoh berikut.
2. Gunakan perintah berikut untuk memulai alat baris perintah cloudhsm_mgmt_util.

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

3. Gunakan perintah loginHSM untuk masuk ke HSM sebagai pengguna CO. Untuk informasi selengkapnya, lihat [???](#).
4. Gunakan perintah approveToken untuk menyetujui token yang ditandatangani, melewati file persetujuan token. Untuk informasi selengkapnya, lihat contoh berikut.

Example— Buat file persetujuan token dan setuju token yang ditandatangani di HSM

File persetujuan token adalah file teks dalam format tertentu yang dibutuhkan HSM. File berisi informasi tentang token, pemberi persetujuan, dan tanda tangan pemberi persetujuan. Berikut ini adalah contoh file persetujuan token.

```
# For "Multi Token File Path", type the path to the file that contains
# the token. You can type the same value for "Token File Path", but
# that's not required. The "Token File Path" line is required in any
# case, regardless of whether you type a value.
Multi Token File Path = officer1.token;
Token File Path = ;

# Total number of approvals
```

```

Number of Approvals = 2;

# Approver 1
# Type the approver's type, name, and the path to the file that
# contains the approver's signature.
Approver Type = 2; # 2 for CO, 1 for CU
Approver Name = officer1;
Approval File = officer1.token.sig1;

# Approver 2
# Type the approver's type, name, and the path to the file that
# contains the approver's signature.
Approver Type = 2; # 2 for CO, 1 for CU
Approver Name = officer2;
Approval File = officer1.token.sig2;

```

Setelah membuat file persetujuan token, CO menggunakan alat baris perintah `cloudhsm_mgmt_util` untuk masuk ke HSM. CO kemudian menggunakan perintah `approveToken` untuk menyetujui token, seperti yang ditunjukkan dalam contoh berikut. Ganti *approval.txt* dengan nama file persetujuan token.

```

aws-cloudhsm > approveToken approval.txt
approveToken success on server 0(10.0.2.14)
approveToken success on server 1(10.0.1.4)

```

Ketika perintah ini berhasil, HSM telah menyetujui token kuorum. Untuk memeriksa status token, gunakan perintah `listTokens`, seperti yang ditunjukkan dalam contoh berikut. Output perintah menunjukkan bahwa token memiliki jumlah persetujuan yang diperlukan.

Waktu validitas token menunjukkan berapa lama token dijamin bertahan di HSM. Bahkan setelah waktu validitas token berlalu (nol detik), Anda masih dapat menggunakan token.

```

aws-cloudhsm > listTokens
=====
      Server 0(10.0.2.14)
=====
----- Token - 0 -----
Token:
Id:1
Service:3
Node:1
Key Handle:0

```

```
User:officer1
Token Validity: 506 sec
Required num of approvers : 2
Current num of approvals : 2
Approver-0: officer1
Approver-1: officer2
Num of tokens = 1

=====
      Server 1(10.0.1.4)
=====
----- Token - 0 -----
Token:
Id:1
Service:3
Node:0
Key Handle:0
User:officer1
Token Validity: 506 sec
Required num of approvers : 2
Current num of approvals : 2
Approver-0: officer1
Approver-1: officer2
Num of tokens = 1

listTokens success
```

Langkah 4. Gunakan token untuk operasi manajemen pengguna

Setelah CO memiliki token dengan jumlah persetujuan yang diperlukan, seperti yang ditunjukkan pada bagian sebelumnya, CO dapat melakukan salah satu operasi manajemen pengguna HSM berikut:

- Buat pengguna HSM dengan perintah [createUser](#)
- Hapus pengguna HSM dengan perintah `deleteUser`
- Ubah kata sandi pengguna HSM yang berbeda dengan perintah `changePswd`

Untuk informasi selengkapnya tentang menggunakan perintah ini, lihat [Pengguna HSM](#).

CO dapat menggunakan token hanya untuk satu operasi. Ketika operasi itu berhasil, token tidak lagi berlaku. Untuk melakukan operasi manajemen pengguna HSM lain, CO harus mendapatkan token

kuorum baru, mendapatkan tanda tangan baru dari pemberi persetujuan, dan menyetujui token baru di HSM.

Note

Token MofN hanya berlaku selama sesi login Anda saat ini terbuka. Jika Anda keluar dari cloudhsm_mgmt_util atau sambungan jaringan terputus, token tidak lagi valid. Demikian pula, token resmi hanya dapat digunakan dalam cloudhsm_mgmt_util, tidak dapat digunakan untuk mengautentikasi dalam aplikasi yang berbeda.

Pada contoh perintah berikut, CO membuat pengguna baru pada HSM.

```
aws-cloudhsm > createUser CU user1 <password>
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****

Do you want to continue(y/n)? y
Creating User user1(CU) on 2 nodes
```

Setelah perintah sebelumnya berhasil, perintah listUsers berikutnya menunjukkan pengguna baru.

```
aws-cloudhsm > listUsers
Users on server 0(10.0.2.14):
Number of users found:8
```

User Id	User Type	User Name	MofnPubKey
1	PCO	admin	NO
0	2FA		
2	AU	app_user	NO
0	NO		
3	CO	officer1	YES
0	NO		
4	CO	officer2	YES
0	NO		
5	CO	officer3	YES
0	NO		

```

6          CO          officer4          YES
0          NO
7          CO          officer5          YES
0          NO
8          CU          user1             NO
0          NO

```

Users on server 1(10.0.1.4):

Number of users found:8

User Id LoginFailureCnt	User Type 2FA	User Name	MofnPubKey
1	PCO	admin	NO
0	NO		
2	AU	app_user	NO
0	NO		
3	CO	officer1	YES
0	NO		
4	CO	officer2	YES
0	NO		
5	CO	officer3	YES
0	NO		
6	CO	officer4	YES
0	NO		
7	CO	officer5	YES
0	NO		
8	CU	user1	NO
0	NO		

Jika CO mencoba untuk melakukan operasi manajemen pengguna HSM lain, operasi akan gagal dengan kesalahan autentikasi kuorum, seperti yang ditunjukkan dalam contoh berikut.

```

aws-cloudhsm > deleteUser CU user1
Deleting user user1(CU) on 2 nodes
deleteUser failed: RET_MXN_AUTH_FAILED
deleteUser failed on server 0(10.0.2.14)

Retry/rollBack/Ignore?(R/B/I): I
deleteUser failed: RET_MXN_AUTH_FAILED
deleteUser failed on server 1(10.0.1.4)

Retry/rollBack/Ignore?(R/B/I): I

```

Perintah `listTokens` akan menampilkan bahwa CO tidak memiliki token yang disetujui, seperti yang ditunjukkan dalam contoh berikut. Untuk melakukan operasi manajemen pengguna HSM lain, CO harus mendapatkan token kuorum baru, mendapatkan tanda tangan baru dari pemberi persetujuan, dan menyetujui token baru pada HSM.

```
aws-cloudhsm > listTokens
=====
      Server 0(10.0.2.14)
=====
Num of tokens = 0

=====
      Server 1(10.0.1.4)
=====
Num of tokens = 0

listTokens success
```

Ubah nilai minimum kuorum dengan AWS CloudHSM Management Utility

Setelah Anda [menetapkan nilai minimum kuorum](#) sehingga [petugas AWS CloudHSM kriptografi \(COs\)](#) dapat menggunakan otentikasi kuorum, Anda mungkin ingin mengubah nilai minimum kuorum. HSM memungkinkan Anda mengubah nilai minimum kuorum hanya jika jumlah pemberi persetujuan sama atau lebih tinggi dari nilai minimum kuorum saat ini. Misalnya, jika nilai minimum kuorum adalah dua, setidaknya dua COs harus menyetujui untuk mengubah nilai minimum kuorum.

Untuk mendapatkan persetujuan kuorum untuk mengubah nilai minimum kuorum, Anda memerlukan token kuorum untuk perintah `setMValue` (layanan 4). Untuk mendapatkan token kuorum perintah `setMValue` (layanan 4), nilai minimum kuorum untuk layanan 4 harus lebih tinggi dari satu. Ini berarti bahwa sebelum Anda dapat mengubah nilai minimum kuorum untuk COs (layanan 3), Anda mungkin perlu mengubah nilai minimum kuorum untuk layanan 4.

Tabel berikut mencantumkan pengenalan layanan HSM bersama dengan nama, deskripsi, dan perintah yang disertakan dalam layanan.

Pengenalan Layanan	Nama Layanan	Deskripsi Layanan	Perintah HSM
3	USER_MGMT	Manajemen pengguna HSM	<code>createUser</code> <code>deleteUser</code>

Pengenal Layanan	Nama Layanan	Deskripsi Layanan	Perintah HSM
			changePswd (hanya berlaku saat mengganti kata sandi pengguna HSM yang berbeda)
4	MISC_CO	Layanan CO lainnya	setMValue

Untuk mengubah nilai minimum kuorum untuk petugas krypto

- Gunakan perintah berikut untuk memulai alat baris perintah cloudhsm_mgmt_util.

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

- Gunakan perintah loginHSM untuk masuk ke HSM sebagai pengguna CO. Untuk informasi selengkapnya, lihat [???](#).
- Gunakan perintah getMValue untuk mendapatkan nilai minimum kuorum untuk layanan 3. Untuk informasi selengkapnya, lihat contoh berikut.
- Gunakan perintah getMValue untuk mendapatkan nilai minimum kuorum untuk layanan 4. Untuk informasi selengkapnya, lihat contoh berikut.
- Jika nilai minimum kuorum untuk layanan 4 lebih rendah dari nilai untuk layanan 3, gunakan perintah setMValue untuk mengubah nilai untuk layanan 4. Ubah nilai untuk layanan 4 ke nilai yang sama atau lebih tinggi dari nilai untuk layanan 3. Untuk informasi selengkapnya, lihat contoh berikut.
- [Dapatkan token kuorum](#), berhati-hatilah untuk menentukan layanan 4 sebagai layanan yang dapat Anda gunakan dengan token.
- [Dapatkan persetujuan \(tanda tangan\)](#) dari orang lain. COs
- [Setujui token pada HSM](#).
- Gunakan setMValue perintah untuk mengubah nilai minimum kuorum untuk layanan 3 (operasi manajemen pengguna dilakukan oleh COs).

Example— Dapatkan nilai minimum kuorum dan ubah nilai untuk layanan 4

Contoh perintah berikut menunjukkan bahwa nilai minimum kuorum untuk layanan 3 saat ini dua.


```
aws-cloudhsm > getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

Contoh perintah berikut menunjukkan bahwa nilai minimum kuorum untuk layanan 4 saat ini satu.

```
aws-cloudhsm > getMValue 4
MValue of service 4[MISC_C0] on server 0 : [1]
MValue of service 4[MISC_C0] on server 1 : [1]
```

Untuk mengubah nilai minimum kuorum untuk layanan 4, gunakan perintah `setMValue`, menetapkan nilai yang sama atau lebih tinggi dari nilai untuk layanan 3. Contoh berikut menetapkan nilai minimum kuorum untuk layanan 4 menjadi dua (2), nilai yang sama yang ditetapkan untuk layanan 3.

```
aws-cloudhsm > setMValue 4 2
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****

Do you want to continue(y/n)? y
Setting M Value(2) for 4 on 2 nodes
```

Perintah berikut menunjukkan bahwa nilai minimum kuorum sekarang adalah dua untuk layanan 3 dan layanan 4.

```
aws-cloudhsm > getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

```
aws-cloudhsm > getMValue 4
MValue of service 4[MISC_C0] on server 0 : [2]
MValue of service 4[MISC_C0] on server 1 : [2]
```

Kunci di AWS CloudHSM

Sebelum Anda dapat menggunakan AWS CloudHSM cluster Anda untuk cryptoprocessing, Anda harus membuat [pengguna](#) dan kunci pada modul keamanan perangkat keras (HSM) di cluster Anda.

Di AWS CloudHSM, gunakan salah satu dari berikut ini untuk mengelola kunci HSMs di klaster Anda:

- Pustaka PKCS #11
- Penyedia JCE
- Penyedia CNG dan KSP
- CloudHSM CLI

Sebelum Anda dapat mengelola kunci, Anda harus login ke HSM dengan nama pengguna dan kata sandi pengguna kriptografi (CU). Hanya CU yang bisa membuat kunci. CU yang membuat kunci memiliki dan mengelola kunci itu.

Lihat topik berikut untuk informasi selengkapnya tentang mengelola kunci di AWS CloudHSM.

Topik

- [Sinkronisasi kunci dan pengaturan daya tahan di AWS CloudHSM](#)
- [Pembungkus kunci AES AWS CloudHSM](#)
- [Menggunakan kunci tepercaya di AWS CloudHSM](#)
- [Manajemen kunci dengan CloudHSM CLI](#)
- [Manajemen kunci dengan AWS CloudHSM KMU](#)

Sinkronisasi kunci dan pengaturan daya tahan di AWS CloudHSM

AWS CloudHSM menyinkronkan setiap kunci token yang Anda buat. Sinkronisasi kunci sebagian besar merupakan proses otomatis, tetapi Anda dapat menggunakan minimal dua modul keamanan perangkat keras (HSM) di cluster Anda untuk membuat kunci lebih tahan lama. Topik ini menjelaskan pengaturan sinkronisasi utama, masalah umum yang dihadapi pelanggan yang bekerja dengan kunci pada sebuah klaster, dan strategi untuk membuat kunci lebih tahan lama.

Topik ini menjelaskan pengaturan sinkronisasi utama di AWS CloudHSM, masalah umum yang dihadapi pelanggan saat bekerja dengan kunci pada klaster, dan strategi untuk membuat kunci lebih tahan lama.

Topik

- [AWS CloudHSM konsep kunci](#)
- [Memahami sinkronisasi AWS CloudHSM kunci](#)
- [Ubah pengaturan daya tahan kunci AWS CloudHSM klien](#)
- [Menyinkronkan kunci di seluruh cluster kloning AWS CloudHSM](#)

AWS CloudHSM konsep kunci

Berikut ini adalah konsep yang harus diperhatikan saat bekerja dengan kunci masuk AWS CloudHSM.

Tombol Token

Kunci persisten yang Anda buat selama operasi menghasilkan, mengimpor, atau membuka bungkus kunci. AWS CloudHSM menyinkronkan kunci token di seluruh cluster.

Kunci sesi

Kunci singkat yang hanya ada pada satu modul keamanan perangkat keras (HSM) di cluster. AWS CloudHSM tidak menyinkronkan kunci sesi di seluruh cluster.

Sinkronisasi tombol sisi klien

Proses sisi klien yang mengkloning kunci token yang Anda buat selama operasi menghasilkan, impor atau membuka kunci. Anda dapat membuat kunci token lebih tahan lama dengan menjalankan cluster dengan minimal dua HSMs.

Sinkronisasi tombol sisi server

Secara berkala klon kunci ke setiap HSM di klaster. Tidak memerlukan manajemen.

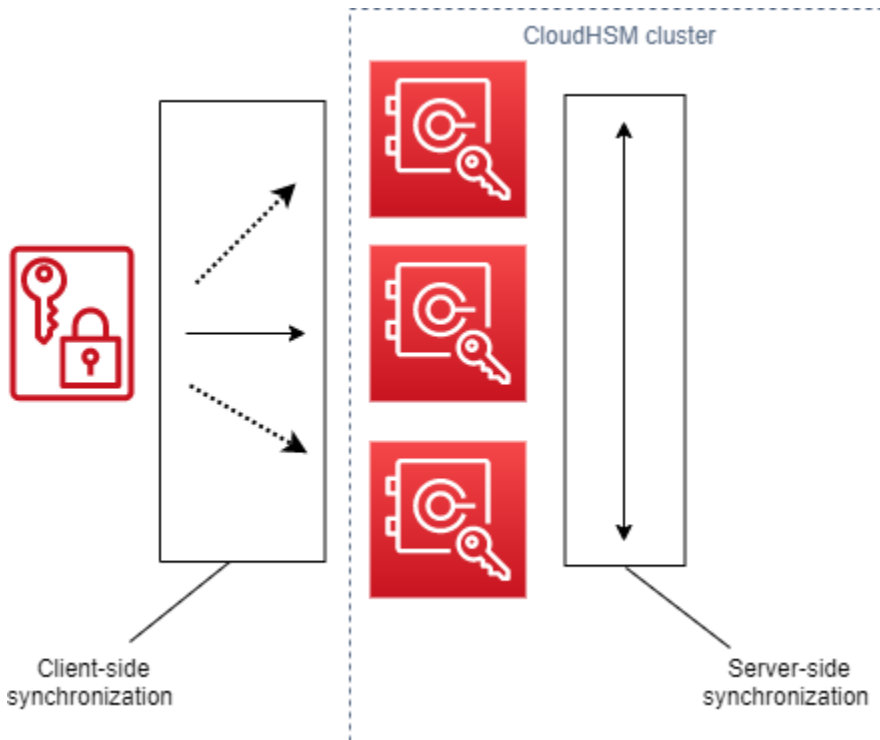
Pengaturan daya tahan kunci klien

Pengaturan yang Anda atur pada klien yang memengaruhi daya tahan kunci. Pengaturan ini bekerja secara berbeda di Client SDK 5 dan Client SDK 3.

- Di SDK Klien 5, gunakan pengaturan ini untuk menjalankan satu klaster HSM.
- Di Client SDK 3, gunakan pengaturan ini untuk menentukan jumlah yang HSMs diperlukan agar operasi pembuatan kunci berhasil.

Memahami sinkronisasi AWS CloudHSM kunci

AWS CloudHSM menggunakan sinkronisasi kunci untuk mengkloning kunci token di semua modul keamanan perangkat keras (HSM) dalam sebuah cluster. Anda membuat kunci token sebagai kunci persisten selama operasi pembuatan, impor, atau membuka kunci. Untuk mendistribusikan kunci ini di klaster, CloudHSM menawarkan sinkronisasi kunci sisi klien dan sisi server.



Tujuan dengan sinkronisasi kunci—baik sisi server maupun sisi klien—adalah mendistribusikan kunci baru di seluruh klaster secepat mungkin setelah Anda membuatnya. Hal ini penting karena panggilan berikutnya yang Anda buat untuk menggunakan kunci baru dapat dirutekan ke setiap HSM yang tersedia di klaster. Jika panggilan Anda membuat rute ke HSM tanpa kunci, maka panggilan gagal. Anda dapat mengurangi kegagalan jenis ini dengan menentukan bahwa panggilan coba lagi aplikasi Anda berikutnya dilakukan setelah operasi pembuatan kunci. Waktu yang diperlukan untuk menyinkronkan dapat bervariasi, tergantung pada beban kerja klaster Anda dan hal-hal tak berwujud lainnya. Gunakan CloudWatch metrik untuk menentukan waktu yang harus digunakan aplikasi Anda dalam situasi seperti ini. Untuk informasi selengkapnya, lihat [CloudWatch Metrik](#).

Tantangan dengan sinkronisasi kunci di lingkungan cloud adalah daya tahan kunci. Anda membuat kunci pada HSM tunggal dan sering mulai menggunakan kunci-kunci tersebut dengan segera. Jika HSM tempat Anda membuat kunci harus gagal sebelum kunci dikloning ke HSM lain di klaster, Anda kehilangan kunci dan akses ke apa pun yang dienkripsi oleh kunci. Untuk mengurangi risiko ini, kami menawarkan sinkronisasi sisi klien. Sinkronisasi sisi klien adalah proses sisi klien yang mengkloning

kunci yang Anda buat selama operasi menghasilkan, impor, atau membuka kunci. Kunci kloning saat Anda membuatnya membuat kunci lebih tahan lama. Tentu saja, Anda tidak dapat mengkloning kunci dalam sebuah klaster dengan satu HSM. Untuk membuat kunci lebih tahan lama, kami juga menyarankan Anda mengonfigurasi cluster Anda untuk menggunakan minimal dua HSMs. Dengan sinkronisasi sisi klien dan cluster dengan dua HSMs, Anda dapat memenuhi tantangan ketahanan kunci di lingkungan cloud.

Ubah pengaturan daya tahan kunci AWS CloudHSM klien

Sinkronisasi kunci sebagian besar merupakan proses otomatis, tetapi Anda dapat mengelola pengaturan daya tahan kunci sisi klien. Pengaturan daya tahan kunci sisi klien bekerja secara berbeda dalam SDK Klien 5 dan SDK Klien 3.

- Di Client SDK 5, kami memperkenalkan konsep kuorum ketersediaan kunci yang mengharuskan Anda menjalankan cluster dengan minimal dua. HSMs Anda dapat menggunakan pengaturan daya tahan kunci sisi klien untuk memilih keluar dari persyaratan dua HSM. Untuk informasi selengkapnya tentang kuorum, lihat [the section called “Konsep”](#).
- Di Client SDK 3, Anda menggunakan pengaturan daya tahan kunci sisi klien untuk menentukan jumlah HSMs pembuatan kunci mana yang harus berhasil agar keseluruhan operasi dianggap berhasil.

Pengaturan daya tahan kunci klien SDK 5 klien

Dalam SDK klien 5, sinkronisasi kunci adalah proses sepenuhnya otomatis. Dengan kuorum ketersediaan kunci, kunci yang baru dibuat harus ada pada dua HSMs di cluster sebelum aplikasi Anda dapat menggunakan kunci. Untuk menggunakan kuorum ketersediaan kunci, klaster Anda harus memiliki minimal dua. HSMs

Jika konfigurasi klaster Anda tidak memenuhi persyaratan daya tahan utama, setiap upaya untuk membuat atau menggunakan kunci token akan gagal dengan pesan galat berikut di log:

```
Key <key handle> does not meet the availability requirements - The key must be available on at least 2 HSMs before being used.
```

Anda dapat menggunakan pengaturan konfigurasi klien untuk memilih keluar dari kuorum ketersediaan kunci. Anda mungkin ingin memilih keluar untuk menjalankan klaster dengan HSM tunggal, misalnya.

Konsep SDK 5 klien

Kuorum Ketersediaan Utama

AWS CloudHSM menentukan jumlah HSMs dalam cluster di mana kunci harus ada sebelum aplikasi Anda dapat menggunakan kunci. Membutuhkan cluster dengan minimal dua HSMs.

Mengelola pengaturan daya tahan kunci klien

Untuk mengelola pengaturan daya tahan kunci klien, Anda harus menggunakan alat konfigurasi untuk SDK Klien 5.

PKCS #11 library

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Linux

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --disable-key-availability-check
```

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Windows

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --disable-key-availability-check
```

OpenSSL Dynamic Engine

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Linux

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --disable-key-availability-check
```

OpenSSL Dynamic Engine Provider

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Linux

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --disable-key-availability-check
```

Key Storage Provider (KSP)

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Windows

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --disable-key-availability-check
```

JCE provider

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Linux

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
$ sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check
```

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Windows

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-key-availability-check
```

CloudHSM CLI

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Linux

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
$ sudo /opt/cloudhsm/bin/configure-cli --disable-key-availability-check
```

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Windows

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --disable-key-availability-check
```

Pengaturan daya tahan kunci klien SDK 3 klien

Dalam SDK Klien 3, sinkronisasi kunci sebagian besar merupakan proses otomatis, tetapi Anda dapat menggunakan pengaturan daya tahan kunci klien untuk membuat kunci lebih tahan lama. Anda menentukan jumlah HSMs pembuatan kunci mana yang harus berhasil agar keseluruhan operasi dianggap sukses. Sinkronisasi sisi klien selalu melakukan upaya terbaik untuk mengkloning kunci ke setiap HSM di klaster, apa pun pengaturan yang Anda pilih. Pengaturan Anda memberlakukan pembuatan kunci pada nomor yang HSMs Anda tentukan. Jika Anda menentukan nilai dan sistem tidak dapat mereplikasi kunci ke jumlah itu HSMs, maka sistem secara otomatis membersihkan materi kunci yang tidak diinginkan dan Anda dapat mencoba lagi.

Important

Jika Anda tidak mengatur pengaturan daya tahan kunci klien (atau jika Anda menggunakan nilai default 1), kunci Anda rentan terhadap kehilangan. Jika HSM Anda saat ini harus gagal sebelum layanan sisi server mengkloning kunci itu ke HSM lain, Anda kehilangan materi kunci.

Untuk memaksimalkan daya tahan kunci, pertimbangkan untuk menentukan setidaknya dua HSMs untuk sinkronisasi sisi klien. Ingatlah bahwa tidak peduli berapa banyak yang HSMs Anda tentukan,

beban kerja di klaster Anda tetap sama. Sinkronisasi sisi klien selalu membuat upaya terbaik untuk mengkloning kunci ke setiap HSM di klaster.

Rekomendasi

- Minimum: Dua HSMs per cluster
- Maksimum: Satu kurang dari jumlah total HSMs di cluster Anda

Jika sinkronisasi sisi klien gagal, layanan klien membersihkan kunci yang tidak diinginkan yang mungkin telah dibuat dan sekarang tidak diinginkan. Pembersihan ini adalah respon upaya terbaik yang mungkin tidak selalu bekerja. Jika pembersihan gagal, Anda mungkin harus menghapus materi kunci yang tidak diinginkan. Untuk informasi selengkapnya, lihat [Kegagalan Sinkronisasi Kunci](#).

Menyiapkan file konfigurasi untuk daya tahan kunci klien

Untuk menentukan pengaturan daya tahan kunci klien, Anda harus mengedit `cloudhsm_client.cfg`.

Untuk mengedit konfigurasi klien

1. Buka `cloudhsm_client.cfg`.

Linux:

```
/opt/cloudhsm/etc/cloudhsm_client.cfg
```

Windows:

```
C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

2. Di `client` simpul file, tambahkan `create_object_minimum_nodes` dan tentukan nilai untuk jumlah HSMs minimum yang AWS CloudHSM harus berhasil membuat kunci agar operasi pembuatan kunci berhasil.

```
"create_object_minimum_nodes" : 2
```

Note

Alat baris perintah `key_mgmt_util` (KMU) memiliki pengaturan tambahan untuk daya tahan kunci klien. Untuk informasi selengkapnya, lihat [the section called “KMU dan sinkronisasi sisi klien”](#)

Referensi konfigurasi

Ini adalah properti sinkronisasi sisi klien, ditampilkan dalam kutipan `cloudhsm_client.cfg`:

```
{
  "client": {
    "create_object_minimum_nodes" : 2,
    ...
  },
  ...
}
```

`create_object_minimum_nodes`

Menentukan jumlah minimum yang HSMs diperlukan untuk menganggap pembuatan kunci, impor kunci, atau operasi pembukaan kunci sukses. Default diatur ke 1. Ini berarti bahwa untuk setiap operasi membuat kunci, layanan sisi klien mencoba untuk membuat kunci pada setiap HSM di kluster, tetapi untuk kembali sukses, hanya perlu untuk membuat satu kunci tunggal pada satu HSM di kluster.

KMU dan sinkronisasi sisi klien

Jika Anda membuat kunci dengan alat baris perintah `key_mgmt_util` (KMU), Anda menggunakan parameter baris perintah opsional (`-min_srv`) untuk membatasi jumlah tombol untuk mengkloning. HSMs Jika Anda menentukan parameter baris perintah dan nilai dalam file konfigurasi, AWS CloudHSM menghormati LARGER dari dua nilai.

Untuk informasi selengkapnya, lihat topik berikut:

- [DSAKeypasangan gen](#)
- [ECCKeypasangan gen](#)

- [RSAKeypasangan gen](#)
- [genSymKey](#)
- [importPrivateKey](#)
- [importPubKey](#)
- [imSymKey](#)
- [insertMaskedObject](#)
- [unWrapKey](#)

Menyinkronkan kunci di seluruh cluster kloning AWS CloudHSM

Sinkronisasi sisi klien dan sisi server hanya untuk menyinkronkan kunci dalam cluster yang sama. AWS CloudHSM Jika Anda menyalin cadangan klaster ke wilayah lain, gunakan perintah [replika kunci untuk mereplikasi](#) kunci di antara dua cluster. Anda mungkin menggunakan kloning klaster untuk redundansi lintas wilayah atau untuk menyederhanakan proses pemulihan bencana Anda. Jika Anda belum menginstal CloudHSM CLI, lihat instruksi di [???](#)

Pembungkus kunci AES AWS CloudHSM

Topik ini menjelaskan opsi untuk pembungkus kunci AES. AWS CloudHSM Pembungkus kunci AES menggunakan kunci AES (kunci pembungkus) untuk membungkus kunci lain dari jenis apa pun (kunci target). Anda menggunakan kunci pembungkus untuk melindungi kunci yang disimpan atau mengirimkan kunci melalui jaringan yang tidak aman.

Topik

- [Algoritme yang didukung](#)
- [Menggunakan pembungkus kunci AES AWS CloudHSM](#)

Algoritme yang didukung

AWS CloudHSM menawarkan tiga opsi untuk pembungkus kunci AES, masing-masing berdasarkan bagaimana kunci target dilapisi sebelum dibungkus. Bantalan dilakukan secara otomatis, sesuai dengan algoritme yang Anda gunakan, ketika Anda memanggil pembungkus kunci. Tabel berikut mencantumkan algoritme yang didukung dan detail terkait untuk membantu Anda memilih mekanisme pembungkus yang sesuai untuk aplikasi Anda.

Algoritme Bungkus Kunci AES	Spesifikasi	Jenis Kunci Target Didukung	Skema bantalan	AWS CloudHSM Ketersediaan Klien
Bungkus Kunci AES dengan Nol Bantalan	RFC 5649 dan SP 800-38F	Semua	Menambahkan nol setelah bit kunci, jika perlu, untuk memblokir penyelarasan	SDK 3.1 dan versi lebih baru
Bungkus Kunci AES dengan Tanpa Bantalan	RFC 3394 dan SP 800-38F	Kunci dengan penyelarasan diblokir seperti AES dan 3DES	Tidak ada	SDK 3.1 dan versi lebih baru
Bungkus Kunci AES dengan Bantalan PKCS #5	Tidak ada	Semua	Setidaknya 8 byte ditambahk an sesuai skema bantalan PKCS #5 untuk memblokir penyelarasan	Semua

Untuk mempelajari cara menggunakan algoritma pembungkus kunci AES dari tabel sebelumnya dalam aplikasi Anda, lihat [Menggunakan AES](#) Key Wrap in. AWS CloudHSM

Memahami vektor inisialisasi dalam bungkus kunci AES

Sebelum membungkus, CloudHSM menambahkan vektor inisialisasi (IV) ke kunci target untuk integritas data. Setiap algoritme pembungkus kunci memiliki batasan khusus pada jenis IV yang diperbolehkan. Untuk mengatur IV AWS CloudHSM, Anda memiliki dua opsi:

- Implisit: mengatur IV ke NULL dan CloudHSM menggunakan nilai default untuk algoritme tersebut untuk operasi membungkus dan membuka bungkus (disarankan)
- Explicit: mengatur IV dengan melewati nilai default IV ke fungsi pembungkus kunci

⚠ Important

Anda harus memahami IV apa yang Anda gunakan dalam aplikasi Anda. Untuk membuka kunci, Anda harus memberikan IV yang sama yang Anda gunakan untuk membungkus kunci. Jika Anda menggunakan IV implisit untuk membungkus, maka gunakan IV implisit untuk membuka. Dengan IV implisit, CloudHSM akan menggunakan nilai default untuk membuka.

Tabel berikut menjelaskan nilai yang diizinkan untuk IVs, yang ditentukan oleh algoritma pembungkus.

Algoritme Bungkus Kunci AES	IV implisit	IV eksplisit
Bungkus Kunci AES dengan Nol Bantalan	Wajib Nilai default: (IV dihitung secara internal berdasarkan spesifikasi)	Tidak diizinkan
Bungkus Kunci AES dengan Tanpa Bantalan	Diizinkan (disarankan) Nilai default: 0xA6A6A6A6A6A6A6A6	Diizinkan Hanya nilai ini yang diterima: 0xA6A6A6A6A6A6A6A6
Bungkus Kunci AES dengan Bantalan PKCS #5	Diizinkan (disarankan) Nilai default: 0xA6A6A6A6A6A6A6A6	Diizinkan Hanya nilai ini yang diterima: 0xA6A6A6A6A6A6A6A6

Menggunakan pembungkus kunci AES AWS CloudHSM

Anda membungkus dan membuka kunci sebagai berikut:

- Di [pustaka PKCS #11](#), pilih mekanisme yang sesuai untuk fungsi `C_WrapKey` dan `C_UnWrapKey` seperti yang ditunjukkan dalam tabel berikut.
- Di [penyedia JCE](#), pilih algoritme, kombinasi mode dan bantalan, menerapkan metode cipher `Cipher.WRAP_MODE` dan `Cipher.UNWRAP_MODE` seperti yang ditunjukkan dalam tabel berikut.

- Di [CloudHSM CLI](#), pilih algoritma yang sesuai dari daftar algoritma yang [Perintah buka kunci di CloudHSM CLI](#) didukung dan seperti [Perintah bungkus kunci di CloudHSM CLI](#) yang ditunjukkan pada tabel berikut.
- Di [key_mgmt_util \(KMU\)](#), gunakan perintah [Ekspor AWS CloudHSM kunci menggunakan KMU](#) dan [Buka AWS CloudHSM kunci menggunakan KMU](#) dengan nilai m yang sesuai seperti yang ditunjukkan dalam tabel berikut.

Algoritme Bungkus Kunci AES	Mekanisme PKCS #11	Metode Java	Sub Perintah CloudHSM CLI	Argumen Utilitas Manajemen Kunci (KMU)
Bungkus Kunci AES dengan Nol Bantalan	CKM_CLOUD_HSM_AES_KEY_WRAP_ZERO_PAD (Mekanisme Ditetapkan Vendor)	AESWrap/ECB/ZeroPadding	aes-zero-pad	m = 6
Bungkus Kunci AES dengan Tanpa Bantalan	CKM_CLOUD_HSM_AES_KEY_WRAP_NO_PAD (Mekanisme Ditetapkan Vendor)	AESWrap/ECB/NoPadding	aes-no-pad	m = 5
Bungkus Kunci AES dengan Bantalan PKCS #5	CKM_CLOUD_HSM_AES_KEY_WRAP_PKCS5_PAD (Mekanisme Ditetapkan Vendor)	AESWrap/ECB/PKCS5Padding	aes-pkcs5-pad	m = 4

Menggunakan kunci tepercaya di AWS CloudHSM

AWS CloudHSM mendukung pembungkus kunci tepercaya untuk melindungi kunci data dari ancaman orang dalam. Topik ini menjelaskan cara membuat kunci tepercaya untuk mengamankan data.

Topik

- [Memahami kunci tepercaya di AWS CloudHSM](#)
- [Atribut kunci tepercaya di AWS CloudHSM](#)
- [Cara menggunakan kunci tepercaya untuk membungkus kunci data AWS CloudHSM](#)
- [Cara membuka kunci data dengan kunci tepercaya untuk AWS CloudHSM](#)

Memahami kunci tepercaya di AWS CloudHSM

Kunci tepercaya adalah kunci yang digunakan untuk membungkus kunci lain dan admin dan petugas kriptografi (COs) secara khusus mengidentifikasi sebagai tepercaya menggunakan atribut. CKA_TRUSTED Selain itu, admin dan petugas kriptografi (COs) menggunakan CKA_UNWRAP_TEMPLATE dan atribut terkait untuk menentukan tindakan apa yang dapat dilakukan kunci data setelah dibuka oleh kunci tepercaya. Kunci data yang dibuka oleh kunci tepercaya juga harus berisi atribut ini agar operasi unwrap berhasil, yang membantu memastikan bahwa kunci data yang tidak dibungkus hanya diizinkan untuk penggunaan yang Anda inginkan.

Gunakan atribut CKA_WRAP_WITH_TRUSTED untuk mengidentifikasi semua kunci data yang ingin Anda bungkus dengan kunci tepercaya. Melakukan hal ini memungkinkan Anda untuk membatasi kunci data sehingga aplikasi hanya dapat menggunakan kunci tepercaya untuk membuka bungkusnya. Setelah Anda menetapkan atribut ini pada kunci data, atribut menjadi read-only dan Anda tidak dapat mengubahnya. Dengan atribut ini di tempat, aplikasi hanya dapat membuka kunci data Anda dengan kunci yang Anda percayai, dan unwraps selalu menghasilkan kunci data dengan atribut yang membatasi bagaimana kunci ini dapat digunakan.

Atribut kunci tepercaya di AWS CloudHSM

Atribut berikut memungkinkan Anda menandai AWS CloudHSM kunci sebagai tepercaya, menentukan kunci data hanya dapat dibungkus dan dibuka dengan kunci tepercaya, dan mengontrol apa yang dapat dilakukan kunci data setelah dibuka:

- **CKA_TRUSTED:** Terapkan atribut ini (selain **CKA_UNWRAP_TEMPLATE**) ke kunci yang akan membungkus kunci data untuk menentukan bahwa admin atau petugas kriptografi (CO) telah melakukan ketekunan yang diperlukan dan mempercayai kunci ini. Hanya admin atau CO yang dapat mengatur **CKA_TRUSTED**. Pengguna kriptografi (CU) memiliki kunci, tetapi hanya CO yang dapat mengatur **CKA_TRUSTED** atributnya.
- **CKA_WRAP_WITH_TRUSTED:** Terapkan atribut ini ke kunci data yang dapat diekspor untuk menentukan bahwa Anda hanya dapat membungkus kunci ini dengan kunci yang ditandai sebagai **CKA_TRUSTED**. Setelah Anda mengatur **CKA_WRAP_WITH_TRUSTED** menjadi **true** (benar), atribut menjadi hanya-baca dan Anda tidak dapat mengubah atau menghapus atribut.
- **CKA_UNWRAP_TEMPLATE:** Terapkan atribut ini ke kunci pembungkus (selain **CKA_TRUSTED**) untuk menentukan nama atribut dan nilai-nilai layanan harus secara otomatis berlaku untuk kunci data yang dibuka oleh layanan. Ketika sebuah aplikasi mengajukan kunci untuk pembukaan, aplikasi juga dapat memberikan templat buka sendiri. Jika Anda menentukan template unwrap dan aplikasi menyediakan template unwrap sendiri, HSM menggunakan kedua template untuk menerapkan nama atribut dan nilai ke kunci. Namun, jika nilai dalam **CKA_UNWRAP_TEMPLATE** untuk kunci pembungkus konflik dengan atribut yang disediakan oleh aplikasi selama permintaan buka, maka permintaan buka gagal.

Untuk informasi lebih lanjut tentang atribut, lihat topik berikut:

- [Atribut kunci PKCS #11](#)
- [Atribut kunci JCE](#)
- [Atribut kunci CloudHSM CLI](#)

Cara menggunakan kunci tepercaya untuk membungkus kunci data AWS CloudHSM

Untuk menggunakan kunci tepercaya untuk membungkus kunci data AWS CloudHSM, Anda harus menyelesaikan tiga langkah dasar:

1. Untuk kunci data yang Anda rencanakan untuk dibungkus dengan kunci tepercaya, atur **CKA_WRAP_WITH_TRUSTED** atributnya ke **true**.
2. Untuk kunci tepercaya yang Anda rencanakan untuk membungkus kunci data, atur **CKA_TRUSTED** atributnya ke **true**.
3. Gunakan kunci tepercaya untuk membungkus kunci data.

Langkah 1: Atur kunci data **CKA_WRAP_WITH_TRUSTED** ke true

Untuk kunci data yang ingin Anda bungkus, pilih salah satu opsi berikut untuk mengatur CKA_WRAP_WITH_TRUSTED atribut kunci ke true. Melakukan hal ini membatasi kunci data sehingga aplikasi hanya dapat menggunakan kunci tepercaya untuk membungkusnya.

Opsi 1: Jika menghasilkan kunci baru, atur **CKA_WRAP_WITH_TRUSTED** ke true

[Buat kunci menggunakan PKCS #11, JCE, atau CloudHSM CLI.](#) Lihat contoh berikut untuk lebih jelasnya.

PKCS #11

Untuk menghasilkan kunci dengan PKCS #11, Anda perlu menyetel CKA_WRAP_WITH_TRUSTED atribut kunci ke true. Seperti yang ditunjukkan dalam contoh berikut, lakukan ini dengan memasukkan atribut ini dalam kunci CK_ATTRIBUTE template dan kemudian mengatur atribut ke true:

```
CK_BYTE_PTR label = "test_key";
CK_ATTRIBUTE template[] = {
    {CKA_WRAP_WITH_TRUSTED, &true_val,      sizeof(CK_BBOOL)},
    {CKA_LABEL,             label,          strlen(label)},
    ...
};
```

Untuk informasi selengkapnya, lihat [sampel publik kami yang mendemonstrasikan pembuatan kunci dengan PKCS #11](#).

JCE

Untuk menghasilkan kunci dengan JCE, Anda perlu mengatur WRAP_WITH_TRUSTED atribut kunci ke true. Seperti yang ditunjukkan dalam contoh berikut, lakukan ini dengan memasukkan atribut ini dalam kunci KeyAttributesMap dan kemudian mengatur atribut ke true:

```
final String label = "test_key";
final KeyAttributesMap keySpec = new KeyAttributesMap();
keySpec.put(KeyAttribute.WRAP_WITH_TRUSTED, true);
keySpec.put(KeyAttribute.LABEL, label);
...
```

Untuk informasi lebih lanjut, lihat [sampel publik kami yang mendemonstrasikan pembuatan kunci dengan JCE](#).

CloudHSM CLI

Untuk menghasilkan kunci dengan CloudHSM CLI, Anda perlu mengatur atribut kunci ke true. `wrap-with-trusted` Lakukan ini dengan memasukkan `wrap-with-trusted=true` argumen yang sesuai untuk perintah pembuatan kunci:

- Untuk kunci simetris, `wrap-with-trusted` tambahkan `attributes` argumen.
- Untuk kunci publik, `wrap-with-trusted` tambahkan `public-attributes` argumen.
- Untuk kunci pribadi, `wrap-with-trusted` tambahkan `private-attributes` argumen.

Untuk informasi selengkapnya tentang pembuatan key pair, lihat [generate-asymmetric-pair](#) Kategori di CloudHSM CLI.

Untuk informasi lebih lanjut tentang pembuatan kunci simetris, lihat [Kategori generate-simetris](#) di CloudHSM CLI.

Opsi 2: Jika menggunakan kunci yang ada, gunakan CloudHSM CLI untuk menyetelnya ke true **CKA_WRAP_WITH_TRUSTED**

Untuk menyetel `CKA_WRAP_WITH_TRUSTED` atribut kunci yang ada ke true, ikuti langkah-langkah berikut:

1. Gunakan [Masuk ke HSM menggunakan CloudHSM CLI](#) perintah untuk masuk sebagai pengguna krypto (CU).
2. Gunakan [Mengatur atribut kunci dengan CloudHSM CLI](#) perintah untuk mengatur `wrap-with-trusted` atribut kunci ke true.

```
aws-cloudhsm > key set-attribute --filter attr.label=test_key --name wrap-with-trusted --value true
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}
```

Langkah 2: Atur kunci tepercaya **CKA_TRUSTED** ke true

Untuk membuat kunci menjadi kunci tepercaya, CKA_TRUSTED atributnya harus disetel ke true. Anda dapat menggunakan CloudHSM CLI atau CloudHSM Management Utility (CMU) untuk melakukan ini.

- Jika menggunakan CloudHSM CLI untuk menyetel atribut kunci, lihat. CKA_TRUSTED [Tandai kunci sebagai tepercaya menggunakan CloudHSM CLI](#)
- Jika menggunakan CMU untuk menyetel CKA_TRUSTED atribut kunci, lihat [Cara menandai kunci sebagai tepercaya dengan Utilitas AWS CloudHSM Manajemen](#).

Langkah 3. Gunakan kunci tepercaya untuk membungkus kunci data

Untuk membungkus kunci data yang direferensikan di Langkah 1 dengan kunci tepercaya yang Anda tetapkan di Langkah 2, lihat tautan berikut untuk contoh kode. Masing-masing menunjukkan cara membungkus kunci.

- [AWS CloudHSM Contoh PKCS #11](#)
- [AWS CloudHSM Contoh JCE](#)

Cara membuka kunci data dengan kunci tepercaya untuk AWS CloudHSM

Untuk membuka kunci data AWS CloudHSM, Anda memerlukan kunci tepercaya yang telah CKA_UNWRAP disetel ke true. Untuk menjadi kunci seperti itu, itu juga harus memenuhi kriteria berikut:

- CKA_TRUSTED atribut kunci harus disetel ke true.
- Kunci harus menggunakan CKA_UNWRAP_TEMPLATE dan atribut terkait untuk menentukan tindakan apa yang dapat dilakukan oleh kunci data setelah dibuka. Jika, misalnya, Anda ingin kunci yang tidak dibungkus menjadi tidak dapat diekspor, Anda menetapkan CKA_EXPORTABLE = FALSE sebagai bagian dari. CKA_UNWRAP_TEMPLATE

Note

CKA_UNWRAP_TEMPLATE hanya tersedia dengan PKCS #11.

Ketika aplikasi mengirimkan kunci untuk dibuka, aplikasi juga dapat menyediakan template unwrap sendiri. Jika Anda menentukan template unwrap dan aplikasi menyediakan template unwrap sendiri,

HSM menggunakan kedua template untuk menerapkan nama atribut dan nilai ke kunci. Namun, jika selama permintaan buka bungkus nilai dalam kunci tepercaya CKA_UNWRAP_TEMPLATE bertentangan dengan atribut yang disediakan oleh aplikasi, permintaan unwrap gagal.

Untuk melihat contoh membuka kunci data dengan kunci tepercaya, lihat contoh [PKCS #11 ini](#).

Manajemen kunci dengan CloudHSM CLI

Jika menggunakan [seri versi SDK terbaru](#), gunakan [CloudHSM CLI](#) untuk mengelola kunci di kluster Anda. AWS CloudHSM Untuk lebih jelasnya, lihat topik di bawah ini.

- [Menggunakan kunci tepercaya](#) menjelaskan cara menggunakan CloudHSM CLI untuk membuat kunci tepercaya untuk mengamankan data.
- [Menghasilkan kunci](#) mencakup instruksi untuk membuat kunci, termasuk kunci simetris, kunci RSA, dan kunci EC.
- [Menghapus kunci](#) menjelaskan bagaimana pemilik kunci menghapus kunci.
- [Kunci berbagi dan tidak berbagi](#) merinci cara pemilik kunci berbagi dan membatalkan pembagian kunci.
- [Tombol penyaringan](#) menawarkan panduan tentang cara menggunakan filter untuk menemukan kunci.
- [Kelola otentikasi kuorum kunci \(M dari N\)](#) menawarkan panduan tentang cara mengatur dan menggunakan otentikasi kuorum dengan kunci.

Hasilkan kunci dengan CloudHSM CLI

Sebelum Anda dapat membuat kunci, Anda harus memulai [CloudHSM CLI](#) dan masuk sebagai pengguna krypto (CU). Untuk menghasilkan kunci pada HSM, gunakan perintah yang sesuai dengan jenis kunci yang ingin Anda hasilkan.

Topik

- [Hasilkan kunci simetris dengan CloudHSM CLI](#)
- [Hasilkan kunci asimetris menggunakan CloudHSM CLI](#)
- [AWS CloudHSM topik terkait utama](#)

Hasilkan kunci simetris dengan CloudHSM CLI

Gunakan perintah yang tercantum dalam [Kategori generate-simetris di CloudHSM CLI](#) untuk menghasilkan kunci simetris untuk AWS CloudHSM. Untuk melihat semua pilihan yang tersedia, gunakan perintah `help key generate-symmetric`.

Hasilkan sebuah kunci AES

Gunakan `key generate-symmetric aes` perintah untuk menghasilkan kunci AES. Untuk melihat semua pilihan yang tersedia, gunakan perintah `help key generate-symmetric aes`.

Example

Contoh berikut menghasilkan kunci AES 32-byte.

```
aws-cloudhsm > key generate-symmetric aes \  
  --label aes-example \  
  --key-length-bytes 32
```

Pendapat

<LABEL>

Menentukan label yang ditentukan pengguna untuk kunci AES.

Wajib: Ya

<KEY-LENGTH-BYTES>

Menentukan panjang kunci dalam byte.

Nilai valid:

- 16, 24, dan 32

Wajib: Ya

<KEY_ATTRIBUTES>

Menentukan daftar spasi dipisahkan atribut kunci untuk mengatur kunci AES dihasilkan dalam bentuk `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (misalnya,) `sign=true`

Untuk daftar atribut AWS CloudHSM kunci yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#).

Wajib: Tidak

<SESSION>

Membuat kunci yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir. Gunakan parameter ini ketika Anda memerlukan kunci hanya sebentar, seperti kunci pembungkus yang mengenkripsi, dan kemudian dengan cepat mendekripsi, kunci lain. Jangan gunakan kunci sesi untuk mengenkripsi data yang mungkin perlu Anda dekripsi setelah sesi berakhir.

Untuk mengubah kunci sesi menjadi kunci persisten (token), gunakan [key set-attribute](#).

Secara default, ketika kunci dihasilkan mereka adalah persistent/token kunci. Menggunakan <SESSION>perubahan ini, memastikan kunci yang dihasilkan dengan argumen ini adalah sesi/fana

Wajib: Tidak

Hasilkan kunci rahasia generik

Gunakan key generate-symmetric generic-secret perintah untuk menghasilkan kunci rahasia generik. Untuk melihat semua pilihan yang tersedia, gunakan perintah help key generate-symmetric generic-secret.

Example

Contoh berikut menghasilkan kunci rahasia generik 32-byte.

```
aws-cloudhsm > key generate-symmetric generic-secret \  
  --label generic-secret-example \  
  --key-length-bytes 32
```

Pendapat

<LABEL>

Menentukan label yang ditentukan pengguna untuk kunci rahasia generik.

Wajib: Ya

<KEY-LENGTH-BYTES>

Menentukan panjang kunci dalam byte.

Nilai valid:

- 1 hingga 800

Wajib: Ya

<KEY_ATTRIBUTES>

Menentukan daftar spasi dipisahkan atribut kunci untuk mengatur untuk kunci rahasia generik yang dihasilkan dalam bentuk KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (misalnya,) `sign=true`

Untuk daftar atribut AWS CloudHSM kunci yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#).

Wajib: Tidak

<SESSION>

Membuat kunci yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir. Gunakan parameter ini ketika Anda memerlukan kunci hanya sebentar, seperti kunci pembungkus yang mengenkripsi, dan kemudian dengan cepat mendekripsi, kunci lain. Jangan gunakan kunci sesi untuk mengenkripsi data yang mungkin perlu Anda dekripsi setelah sesi berakhir.

Untuk mengubah kunci sesi menjadi kunci persisten (token), gunakan [key set-attribute](#).

Secara default, ketika kunci dihasilkan mereka adalah persistent/token kunci. Menggunakan <SESSION>perubahan ini, memastikan kunci yang dihasilkan dengan argumen ini adalah sesi/fana

Wajib: Tidak

Hasilkan kunci asimetris menggunakan CloudHSM CLI

Gunakan perintah yang tercantum dalam [generate-asymmetric-pair](#) **Kategori di CloudHSM CLI** untuk menghasilkan pasangan kunci asimetris untuk AWS CloudHSM cluster.

Menghasilkan kunci RSA

Gunakan `key generate-asymmetric-pair rsa` perintah untuk menghasilkan key pair RSA. Untuk melihat semua pilihan yang tersedia, gunakan perintah `help key generate-asymmetric-pair rsa`.

Example

Contoh berikut menghasilkan pasangan kunci RSA 2048-bit.

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
```

```
--public-exponent 65537 \  
--modulus-size-bits 2048 \  
--public-label rsa-public-example \  
--private-label rsa-private-example
```

Pendapat

<PUBLIC_LABEL>

Menentukan label yang ditentukan pengguna untuk kunci publik.

Wajib: Ya

<PRIVATE_LABEL>

Menentukan label yang ditentukan pengguna untuk kunci pribadi.

Wajib: Ya

<MODULUS_SIZE_BITS>

Menentukan panjang modulus dalam bit. Nilai minimum adalah 2048.

Wajib: Ya

<PUBLIC_EXPONENT>

Menentukan eksponen publik. Nilai harus angka ganjil yang lebih besar dari atau sama dengan 65537.

Wajib: Ya

<PUBLIC_KEY_ATTRIBUTES>

Menentukan daftar spasi dipisahkan atribut kunci untuk mengatur untuk kunci publik RSA dihasilkan dalam bentuk KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (misalnya,).
sign=true

Untuk daftar atribut AWS CloudHSM kunci yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#).

Wajib: Tidak

<SESSION>

Membuat kunci yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir. Gunakan parameter ini ketika Anda memerlukan kunci hanya sebentar, seperti kunci pembungkus

yang mengenkripsi, dan kemudian dengan cepat mendekripsi, kunci lain. Jangan gunakan kunci sesi untuk mengenkripsi data yang mungkin perlu Anda dekripsi setelah sesi berakhir.

Untuk mengubah kunci sesi menjadi kunci persisten (token), gunakan [key set-attribute](#).

Secara default, ketika kunci dihasilkan mereka adalah persistent/token kunci. Menggunakan <SESSION> perubahan ini, memastikan kunci yang dihasilkan dengan argumen ini adalah sesi/fana

Wajib: Tidak

Hasilkan pasangan kunci EC (elliptic curve cryptography)

Gunakan `key generate-asymmetric-pair ec` perintah untuk menghasilkan EC key pair. Untuk melihat semua opsi yang tersedia, termasuk daftar kurva elips yang didukung, gunakan perintah. `help key generate-asymmetric-pair ec`

Example

Contoh berikut menghasilkan key pair EC menggunakan kurva eliptik Secp384R1.

```
aws-cloudhsm > key generate-asymmetric-pair ec \  
  --curve secp384r1 \  
  --public-label ec-public-example \  
  --private-label ec-private-example
```

Pendapat

<PUBLIC_LABEL>

Menentukan label yang ditentukan pengguna untuk kunci publik. Ukuran maksimum yang diizinkan `label` adalah 127 karakter untuk Client SDK 5.11 dan setelahnya. Klien SDK 5.10 dan sebelumnya memiliki batas 126 karakter.

Wajib: Ya

<PRIVATE_LABEL>

Menentukan label yang ditentukan pengguna untuk kunci pribadi. Ukuran maksimum yang diizinkan `label` adalah 127 karakter untuk Client SDK 5.11 dan setelahnya. Klien SDK 5.10 dan sebelumnya memiliki batas 126 karakter.

Wajib: Ya

<CURVE>

Menentukan pengenalan untuk kurva elips.

Nilai valid:

- primer256v1
- secp256r1
- secp224r1
- secp384r1
- secp256k1
- secp521r1
- ed25519 (hanya didukung pada instance hsm2m.medium dalam mode non-FIPS)

Wajib: Ya

<PUBLIC_KEY_ATTRIBUTES>

Menentukan daftar atribut kunci yang dipisahkan spasi untuk ditetapkan untuk kunci publik EC yang dihasilkan dalam bentuk KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (misalnya,).
verify=true

Untuk daftar atribut AWS CloudHSM kunci yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#).

Wajib: Tidak

<PRIVATE_KEY_ATTRIBUTES>

Menentukan daftar atribut kunci yang dipisahkan spasi untuk mengatur kunci pribadi EC yang dihasilkan dalam bentuk KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (misalnya,).
sign=true

Untuk daftar atribut AWS CloudHSM kunci yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#).

Wajib: Tidak

<SESSION>

Membuat kunci yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir. Gunakan parameter ini ketika Anda memerlukan kunci hanya sebentar, seperti kunci pembungkus yang mengenkripsi, dan kemudian dengan cepat mendekripsi, kunci lain. Jangan gunakan kunci sesi untuk mengenkripsi data yang mungkin perlu Anda dekripsi setelah sesi berakhir.

Untuk mengubah kunci sesi menjadi kunci persisten (token), gunakan [key set-attribute](#).

Secara default, kunci yang dihasilkan adalah kunci persisten (token). Melewati `<SESSION>` perubahan ini, memastikan kunci yang dihasilkan dengan argumen ini adalah kunci sesi (singkat).

Wajib: Tidak

AWS CloudHSM topik terkait utama

Lihat bagian berikut untuk informasi tambahan tentang kunci di AWS CloudHSM.

- [Atribut kunci untuk CloudHSM CLI](#)
- [generate-asymmetric-pair](#) Kategori di CloudHSM CLI
- [Kategori generate-simetris di CloudHSM CLI](#)

Hapus kunci menggunakan CloudHSM CLI

Gunakan contoh dalam topik ini untuk menghapus kunci dengan [CloudHSM](#) CLI. Hanya pemilik kunci yang dapat menghapus kunci.

Topik

- [Contoh: Hapus kunci](#)
- [Topik terkait](#)

Contoh: Hapus kunci

1. Jalankan key list perintah untuk mengidentifikasi kunci yang ingin Anda hapus:

```
aws-cloudhsm > key list --filter attr.label="my_key_to_delete" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000000540011",
        "key-info": {
          "key-owners": [
            {
              "username": "my_crypto_user",
```

```

        "key-coverage": "full"
    },
    ],
    "shared-users": [],
    "key-quorum-values": {
        "manage-key-quorum-value": 0,
        "use-key-quorum-value": 0
    },
    "cluster-coverage": "full"
},
"attributes": {
    "key-type": "rsa",
    "label": "my_key_to_delete",
    "id": "",
    "check-value": "0x29bbd1",
    "class": "private-key",
    "encrypt": false,
    "decrypt": true,
    "token": true,
    "always-sensitive": true,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": true,
    "verify": false,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 1217,
    "public-exponent": "0x010001",
    "modulus":
"0x8b3a7c20618e8be08220ed8ab2c8550b65fc1aad8d4cf04fbf2be685f97eeb78fcbbad9b02cd91a3b15e990
    "modulus-size-bits": 2048
}
}
],
"total_key_count": 1,
"returned_key_count": 1

```

```
}
```

2. Setelah mengidentifikasi kunci, jalankan `label` atribut unik key delete dengan kunci untuk menghapus kunci:

```
aws-cloudhsm > key delete --filter attr.label="my_key_to_delete"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
}
```

3. Jalankan `key list` perintah dengan `label` atribut unik kunci dan konfirmasikan bahwa kunci telah dihapus. Seperti yang ditunjukkan pada contoh berikut, tidak ada kunci dengan label `my_key_to_delete` di cluster HSM:

```
aws-cloudhsm > key list --filter attr.label="my_key_to_delete"
{
  "error_code": 0,
  "data": {
    "matched_keys": [],
    "total_key_count": 0,
    "returned_key_count": 0
  }
}
```

Topik terkait

- [Atribut kunci untuk CloudHSM CLI](#)
- [Hapus kunci dengan CloudHSM CLI](#)

Bagikan dan batalkan berbagi kunci menggunakan CloudHSM CLI

Gunakan perintah dalam topik ini untuk berbagi dan membatalkan berbagi kunci di [CloudHSM](#) CLI. Pada tahun AWS CloudHSM, pengguna kriptografi (CU) yang membuat kunci memilikinya. Pemilik dapat menggunakan `key unshare` perintah `key share` dan untuk berbagi dan membatalkan pembagian kunci dengan yang lain CUs. Pengguna dengan siapa kunci dibagikan dapat menggunakan kunci

dalam operasi kriptografi, tetapi mereka tidak dapat mengekspor kunci, menghapus kunci, atau membagikannya dengan pengguna lain.

Sebelum Anda dapat membagikan kunci, Anda harus masuk ke HSM sebagai pengguna krypto (CU) yang memiliki kunci tersebut.

Topik

- [Contoh: Berbagi dan membatalkan berbagi kunci](#)
- [Topik terkait](#)

Contoh: Berbagi dan membatalkan berbagi kunci

Example

Contoh berikut menunjukkan cara berbagi dan membatalkan pembagian kunci dengan pengguna krypto (CU)alice. Seiring dengan key unshare perintah key share dan, perintah berbagi dan tidak berbagi juga memerlukan kunci tertentu menggunakan filter kunci [CloudHSM CLI dan nama pengguna spesifik pengguna yang kuncinya](#) akan dibagikan atau tidak dibagikan.

1. Mulailah dengan menjalankan key list perintah dengan filter untuk mengembalikan kunci tertentu dan melihat dengan siapa kunci tersebut sudah dibagikan.

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            }
          ]
        }
      }
    ]
  }
}
```

```
{
  "username": "cu1",
  "key-coverage": "full"
},
{
  "username": "cu4",
  "key-coverage": "full"
},
{
  "username": "cu5",
  "key-coverage": "full"
},
{
  "username": "cu6",
  "key-coverage": "full"
},
{
  "username": "cu7",
  "key-coverage": "full"
},
],
"key-quorum-values": {
  "manage-key-quorum-value": 0,
  "use-key-quorum-value": 0
},
"cluster-coverage": "full"
},
"attributes": {
  "key-type": "rsa",
  "label": "rsa_key_to_share",
  "id": "",
  "check-value": "0xae8ff0",
  "class": "private-key",
  "encrypt": false,
  "decrypt": true,
  "token": true,
  "always-sensitive": true,
  "derive": false,
  "destroyable": true,
  "extractable": true,
  "local": true,
  "modifiable": true,
  "never-extractable": false,
  "private": true,
```

```

        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": true,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1219,
        "public-exponent": "0x010001",
        "modulus":
"0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
        "modulus-size-bits": 2048
    }
}
],
"total_key_count": 1,
"returned_key_count": 1
}
}

```

2. Lihat `shared-users` output untuk mengidentifikasi dengan siapa kunci saat ini dibagikan.
3. Untuk membagikan kunci ini dengan pengguna krypto (CU) `alice`, masukkan perintah berikut:

```

aws-cloudhsm > key share --filter attr.label="rsa_key_to_share" attr.class=private-
key --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "message": "Key shared successfully"
  }
}

```

Perhatikan bahwa, bersama dengan `key share` perintah, perintah ini menggunakan label unik kunci dan nama pengguna yang kunci akan dibagikan.

4. Jalankan `key list` perintah untuk mengonfirmasi bahwa kunci telah dibagikan dengan `alice`:

```

aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {

```



```
"key-reference": "0x000000000001c0686",
"key-info": {
  "key-owners": [
    {
      "username": "cu3",
      "key-coverage": "full"
    }
  ],
  "shared-users": [
    {
      "username": "cu2",
      "key-coverage": "full"
    },
    {
      "username": "cu1",
      "key-coverage": "full"
    },
    {
      "username": "cu4",
      "key-coverage": "full"
    },
    {
      "username": "cu5",
      "key-coverage": "full"
    },
    {
      "username": "cu6",
      "key-coverage": "full"
    },
    {
      "username": "cu7",
      "key-coverage": "full"
    },
    {
      "username": "alice",
      "key-coverage": "full"
    }
  ],
  "key-quorum-values": {
    "manage-key-quorum-value": 0,
    "use-key-quorum-value": 0
  },
  "cluster-coverage": "full"
},
```

```

    "attributes": {
      "key-type": "rsa",
      "label": "rsa_key_to_share",
      "id": "",
      "check-value": "0xae8ff0",
      "class": "private-key",
      "encrypt": false,
      "decrypt": true,
      "token": true,
      "always-sensitive": true,
      "derive": false,
      "destroyable": true,
      "extractable": true,
      "local": true,
      "modifiable": true,
      "never-extractable": false,
      "private": true,
      "sensitive": true,
      "sign": true,
      "trusted": false,
      "unwrap": true,
      "verify": false,
      "wrap": false,
      "wrap-with-trusted": false,
      "key-length-bytes": 1219,
      "public-exponent": "0x010001",
      "modulus":
"0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
      "modulus-size-bits": 2048
    }
  ],
  "total_key_count": 1,
  "returned_key_count": 1
}
}

```

5. Untuk membatalkan berbagi kunci yang sama dengan `alice`, jalankan `unshare` perintah berikut:

```

aws-cloudhsm > key unshare --filter attr.label="rsa_key_to_share"
attr.class=private-key --username alice --role crypto-user
{
  "error_code": 0,
  "data": {

```

```

    "message": "Key unshared successfully"
  }
}

```

Perhatikan bahwa, bersama dengan key unshare perintah, perintah ini menggunakan label unik kunci dan nama pengguna yang kunci akan dibagikan.

6. Jalankan key list perintah lagi dan konfirmasikan bahwa kunci tidak dibagikan dengan pengguna alice krypto:

```

aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
            {
              "username": "cu4",
              "key-coverage": "full"
            },
            {
              "username": "cu5",
              "key-coverage": "full"
            },
            {
              "username": "cu6",

```

```

        "key-coverage": "full"
    },
    {
        "username": "cu7",
        "key-coverage": "full"
    },
],
"key-quorum-values": {
    "manage-key-quorum-value": 0,
    "use-key-quorum-value": 0
},
"cluster-coverage": "full"
},
"attributes": {
    "key-type": "rsa",
    "label": "rsa_key_to_share",
    "id": "",
    "check-value": "0xae8ff0",
    "class": "private-key",
    "encrypt": false,
    "decrypt": true,
    "token": true,
    "always-sensitive": true,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": true,
    "verify": false,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 1219,
    "public-exponent": "0x010001",
    "modulus":
"0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
    "modulus-size-bits": 2048
}
}

```

```
    ],  
    "total_key_count": 1,  
    "returned_key_count": 1  
  }  
}
```

Topik terkait

- [Atribut kunci untuk CloudHSM CLI](#)
- [Bagikan kunci menggunakan CloudHSM CLI](#)
- [Hapus berbagi kunci menggunakan CloudHSM CLI](#)
- [Tombol filter menggunakan CloudHSM CLI](#)

Tombol filter menggunakan CloudHSM CLI

[Gunakan perintah kunci berikut untuk memanfaatkan mekanisme filtrasi kunci standar untuk CloudHSM CLI.](#)

- key list
- key delete
- key share
- key unshare
- key set-attribute

Untuk memilih kunci and/or filter dengan CloudHSM CLI, perintah kunci menggunakan mekanisme filtrasi standar berdasarkan. [Atribut kunci untuk CloudHSM CLI](#) Kunci atau set kunci dapat ditentukan dalam perintah kunci dengan menggunakan satu atau lebih AWS CloudHSM atribut yang dapat mengidentifikasi satu kunci atau beberapa kunci. Mekanisme filtrasi kunci hanya beroperasi pada kunci yang dimiliki dan dibagikan oleh pengguna yang saat ini masuk, serta semua kunci publik di AWS CloudHSM cluster.

Topik

- [Persyaratan](#)
- [Memfilter untuk menemukan satu kunci](#)
- [Kesalahan Filtrasi](#)

- [Topik terkait](#)

Persyaratan

Untuk memfilter kunci, Anda harus masuk sebagai pengguna krypto (CUs).

Memfilter untuk menemukan satu kunci

Harap dicatat bahwa dalam contoh berikut, setiap atribut yang digunakan sebagai filter harus ditulis dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`. Misalnya, jika Anda ingin memfilter berdasarkan atribut label, Anda akan menulis `attr.label=my_label`.

Example Gunakan atribut tunggal untuk menemukan satu kunci

Contoh ini menunjukkan bagaimana untuk memfilter ke kunci unik tunggal menggunakan hanya atribut identifikasi tunggal.

```
aws-cloudhsm > key list --filter attr.label="my_unique_key_label" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "alice",
              "key-coverage": "full"
            }
          ],
          "key-quorum-values": {
            "manage-key-quorum-value": 0,
            "use-key-quorum-value": 0
          },
          "cluster-coverage": "full"
        }
      }
    ]
  }
}
```

```

    },
    "attributes": {
      "key-type": "rsa",
      "label": "my_unique_key_label",
      "id": "",
      "check-value": "0xae8ff0",
      "class": "private-key",
      "encrypt": false,
      "decrypt": true,
      "token": true,
      "always-sensitive": true,
      "derive": false,
      "destroyable": true,
      "extractable": true,
      "local": true,
      "modifiable": true,
      "never-extractable": false,
      "private": true,
      "sensitive": true,
      "sign": true,
      "trusted": false,
      "unwrap": true,
      "verify": false,
      "wrap": false,
      "wrap-with-trusted": false,
      "key-length-bytes": 1219,
      "public-exponent": "0x010001",
      "modulus":
"0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254c8f5
      "modulus-size-bits": 2048
    }
  ],
  "total_key_count": 1,
  "returned_key_count": 1
}
}

```

Example Gunakan beberapa atribut untuk menemukan satu kunci

Contoh berikut menunjukkan bagaimana menemukan kunci tunggal menggunakan beberapa atribut kunci.

```
aws-cloudhsm > key list --filter attr.key-type=rsa attr.class=private-key attr.check-  
value=0x29bbd1 --verbose  
{  
  "error_code": 0,  
  "data": {  
    "matched_keys": [  
      {  
        "key-reference": "0x00000000000540011",  
        "key-info": {  
          "key-owners": [  
            {  
              "username": "cu3",  
              "key-coverage": "full"  
            }  
          ],  
          "shared-users": [  
            {  
              "username": "cu2",  
              "key-coverage": "full"  
            }  
          ],  
          "key-quorum-values": {  
            "manage-key-quorum-value": 0,  
            "use-key-quorum-value": 0  
          },  
          "cluster-coverage": "full"  
        },  
        "attributes": {  
          "key-type": "rsa",  
          "label": "my_crypto_user",  
          "id": "",  
          "check-value": "0x29bbd1",  
          "class": "my_test_key",  
          "encrypt": false,  
          "decrypt": true,  
          "token": true,  
          "always-sensitive": true,  
          "derive": false,  
          "destroyable": true,  
          "extractable": true,  
          "local": true,  
          "modifiable": true,  
          "never-extractable": false,  
        }  
      }  
    ]  
  }  
}
```



```

        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": true,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1217,
        "public-exponent": "0x010001",
        "modulus":
"0x8b3a7c20618e8be08220ed8ab2c8550b65fc1aad8d4cf04fbf2be685f97eeb78fcbbad9b02cd91a3b15e990c2a7
        "modulus-size-bits": 2048
    }
  ],
  "total_key_count": 1,
  "returned_key_count": 1
}
}

```

Example Memfilter untuk menemukan satu set kunci

Contoh berikut menunjukkan cara memfilter untuk menemukan satu set kunci rsa pribadi.

```

aws-cloudhsm > key list --filter attr.key-type=rsa attr.class=private-key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "my_crypto_user",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            }
          ],

```

```

    {
      "username": "cu1",
      "key-coverage": "full"
    },
  ],
  "key-quorum-values": {
    "manage-key-quorum-value": 0,
    "use-key-quorum-value": 0
  },
  "cluster-coverage": "full"
},
"attributes": {
  "key-type": "rsa",
  "label": "rsa_key_to_share",
  "id": "",
  "check-value": "0xae8ff0",
  "class": "private-key",
  "encrypt": false,
  "decrypt": true,
  "token": true,
  "always-sensitive": true,
  "derive": false,
  "destroyable": true,
  "extractable": true,
  "local": true,
  "modifiable": true,
  "never-extractable": false,
  "private": true,
  "sensitive": true,
  "sign": true,
  "trusted": false,
  "unwrap": true,
  "verify": false,
  "wrap": false,
  "wrap-with-trusted": false,
  "key-length-bytes": 1219,
  "public-exponent": "0x010001",
  "modulus":
"0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254c8f5",
  "modulus-size-bits": 2048
}
},
{
  "key-reference": "0x00000000000540011",

```

```
"key-info": {
  "key-owners": [
    {
      "username": "my_crypto_user",
      "key-coverage": "full"
    }
  ],
  "shared-users": [
    {
      "username": "cu2",
      "key-coverage": "full"
    }
  ],
  "key-quorum-values": {
    "manage-key-quorum-value": 0,
    "use-key-quorum-value": 0
  },
  "cluster-coverage": "full"
},
"attributes": {
  "key-type": "rsa",
  "label": "my_test_key",
  "id": "",
  "check-value": "0x29bbd1",
  "class": "private-key",
  "encrypt": false,
  "decrypt": true,
  "token": true,
  "always-sensitive": true,
  "derive": false,
  "destroyable": true,
  "extractable": true,
  "local": true,
  "modifiable": true,
  "never-extractable": false,
  "private": true,
  "sensitive": true,
  "sign": true,
  "trusted": false,
  "unwrap": true,
  "verify": false,
  "wrap": false,
  "wrap-with-trusted": false,
  "key-length-bytes": 1217,
```

```

        "public-exponent": "0x010001",
        "modulus":
"0x8b3a7c20618e8be08220ed8ab2c8550b65fc1aad8d4cf04fbf2be685f97eeb78fcbbad9b02cd91a3b15e990c2a7
        "modulus-size-bits": 2048
    }
}
],
"total_key_count": 2,
"returned_key_count": 2
}
}

```

Kesalahan Filtrasi

Operasi kunci tertentu hanya dapat dilakukan pada satu kunci pada satu waktu. Untuk operasi ini, CloudHSM CLI akan memberikan kesalahan jika kriteria filtrasi tidak cukup disempurnakan dan beberapa kunci cocok dengan kriteria. Salah satu contohnya ditunjukkan di bawah ini dengan tombol hapus.

Example Kesalahan Filtrasi saat mencocokkan terlalu banyak tombol

```

aws-cloudhsm > key delete --filter attr.key-type=rsa
{
  "error_code": 1,
  "data": "Key selection criteria matched 48 keys. Refine selection criteria to select
a single key."
}

```

Topik terkait

- [Atribut kunci untuk CloudHSM CLI](#)

Tandai kunci sebagai tepercaya menggunakan CloudHSM CLI

Konten di bagian ini memberikan petunjuk tentang penggunaan CloudHSM CLI untuk menandai kunci sebagai tepercaya.

1. Menggunakan perintah [CloudHSM login CLI](#), masuk sebagai pengguna kripto (CU).

- Gunakan `key list` perintah untuk mengidentifikasi referensi kunci dari kunci yang ingin Anda tandai sebagai tepercaya. Contoh berikut mencantumkan kunci dengan `labeltest_aes_trusted`.

```
aws-cloudhsm > key list --filter attr.label=test_aes_trusted
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000000200333",
        "attributes": {
          "label": "test_aes_trusted"
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

- Menggunakan [Keluar dari HSM menggunakan CloudHSM CLI](#) perintah, keluar sebagai pengguna krypto (CU).
- Menggunakan [Masuk ke HSM menggunakan CloudHSM CLI](#) perintah, masuk sebagai admin.
- Menggunakan perintah [key set-attribute](#) dengan referensi kunci yang Anda identifikasi di langkah 2, atur nilai tepercaya kunci ke true:

```
aws-cloudhsm > key set-attribute --filter key-reference=<Key Reference> --name
trusted --value true
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}
```

Kelola otentikasi kuorum (M dari N kontrol akses) menggunakan CloudHSM CLI

Modul keamanan perangkat keras (HSMs) di AWS CloudHSM klaster Anda mendukung otentikasi kuorum, juga dikenal sebagai kontrol akses M of N. Dengan otentikasi kuorum, tidak ada satu pengguna pun di HSM yang dapat melakukan operasi yang dikendalikan kuorum. Sebaliknya, sejumlah minimum pengguna HSM (paling sedikit 2) harus bekerja sama untuk melakukan operasi ini. Otentikasi kuorum menambahkan lapisan perlindungan tambahan dengan memerlukan persetujuan dari beberapa pengguna HSM.

Autentikasi kuorum dapat mengontrol operasi berikut:

- Penggunaan dan pengelolaan kunci HSM oleh [pengguna krypto](#) — Membuat tanda tangan dengan kunci, atau membungkus, membuka bungkusan, berbagi, membatalkan berbagi, dan menyetel atribut kunci.

Pertimbangan penting

- Pengguna HSM dapat menandatangani token kuorum mereka sendiri—yaitu, pengguna yang meminta dapat memberikan salah satu persetujuan yang diperlukan untuk otentikasi kuorum.
- Anda memilih jumlah minimum pemberi persetujuan kuorum untuk operasi yang dikontrol kuorum. Angka terkecil yang dapat Anda pilih adalah dua (2), dan angka terbesar yang dapat Anda pilih adalah delapan (8).
- HSM dapat menyimpan hingga 1.024 token kuorum. Jika HSM sudah memiliki 1.024 token saat Anda mencoba membuat yang baru, HSM membersihkan salah satu token yang kedaluwarsa. Secara default, token kedaluwarsa sepuluh menit setelah pembuatannya.
- Jika otentikasi multi-faktor (MFA) diaktifkan, cluster menggunakan kunci yang sama untuk otentikasi kuorum dan untuk MFA. [Untuk informasi selengkapnya tentang menggunakan otentikasi kuorum dan MFA, lihat Menggunakan CloudHSM CLI untuk mengelola MFA.](#)
- Setiap HSM hanya dapat berisi satu token per layanan Admin pada satu waktu, tetapi beberapa token per layanan Pengguna Crypto.

Topik berikut ini menyediakan informasi lebih lanjut tentang autentikasi kuorum di AWS CloudHSM.

Topik

- [Proses otentikasi kuorum untuk CloudHSM CLI](#)

- [Nama dan jenis AWS CloudHSM layanan yang didukung untuk otentikasi kuorum dengan CloudHSM CLI](#)
- [Siapkan otentikasi kuorum untuk pengguna kriptografi menggunakan CloudHSM AWS CloudHSM CLI](#)
- [Manajemen kunci dan penggunaan dengan otentikasi kuorum diaktifkan untuk menggunakan AWS CloudHSM CloudHSM CLI](#)

Proses otentikasi kuorum untuk CloudHSM CLI

Langkah-langkah berikut merangkum proses otentikasi kuorum untuk CloudHSM CLI. Untuk langkah-langkah dan alat tertentu, lihat [Manajemen kunci dan penggunaan dengan otentikasi kuorum diaktifkan untuk menggunakan AWS CloudHSM CloudHSM CLI](#).

1. Setiap pengguna modul keamanan perangkat keras (HSM) membuat kunci asimetris untuk penandatanganan. Pengguna melakukan ini di luar HSM, berhati-hati untuk melindungi kunci dengan tepat.
2. Setiap pengguna HSM masuk ke HSM dan mendaftarkan bagian publik dari kunci penandatanganan mereka (kunci publik) dengan HSM.
3. Ketika pengguna HSM ingin melakukan operasi yang dikendalikan kuorum, pengguna yang sama masuk ke HSM dan mendapatkan token kuorum.
4. Pengguna HSM memberikan token kuorum ke satu atau lebih pengguna HSM lainnya dan meminta persetujuan mereka.
5. Pengguna HSM lainnya menyetujui dengan menggunakan kunci mereka untuk secara kriptografi menandatangani token kuorum. Hal ini terjadi di luar HSM.
6. Ketika pengguna HSM memiliki jumlah persetujuan yang diperlukan, pengguna yang sama masuk ke HSM dan menjalankan operasi yang dikendalikan kuorum dengan --approval argumen, menyediakan file token kuorum yang ditandatangani, yang berisi semua persetujuan yang diperlukan (tanda tangan).
7. HSM menggunakan kunci publik terdaftar dari setiap penandatanganan untuk memverifikasi tanda tangan. Jika tanda tangan valid, HSM menyetujui token dan operasi yang dikendalikan kuorum dilakukan.

Nama dan jenis AWS CloudHSM layanan yang didukung untuk otentikasi kuorum dengan CloudHSM CLI

Layanan Admin: Otentikasi kuorum digunakan untuk layanan istimewa admin seperti membuat pengguna, menghapus pengguna, mengubah kata sandi pengguna, mengatur nilai kuorum, dan menonaktifkan kemampuan kuorum dan MFA.

Layanan Pengguna Crypto: Otentikasi kuorum digunakan untuk layanan istimewa pengguna kriptografi yang terkait dengan kunci tertentu seperti menandatangani dengan kunci, kunci, sharing/unsharing kunci, dan wrapping/unwrapping pengaturan atribut kunci. Nilai kuorum kunci terkait dikonfigurasi saat kunci dihasilkan, diimpor, atau dibuka. Nilai kuorum harus sama dengan atau kurang dari jumlah pengguna yang dikaitkan dengan kunci, yang mencakup pengguna yang kunci dibagikan dan pemilik kunci.

Setiap jenis layanan selanjutnya dipecah menjadi nama layanan yang memenuhi syarat, yang berisi serangkaian operasi layanan yang didukung kuorum tertentu yang dapat dilakukan.

Nama layanan	Jenis Layanan	Operasi layanan
user	Admin	• pengguna membuat • menghapus pengguna • perubahan kata sandi pengguna • perubahan pengguna-mfa
kuorum	Admin	• tanda token kuorum set-quorum-value
kluster ¹	Admin	• kluster mtls register-trust-anchor • kluster mtls deregister-trust-anchor • cluster mtls set-penegakan
manajemen kunci	Pengguna Crypto	• bungkus kunci • buka kunci • pembagian kunci

Nama layanan	Jenis Layanan	Operasi layanan
		• kunci unshare • kunci set-attribut
penggunaan kunci	Pengguna Crypto	• tanda kunci

[1] Layanan cluster tersedia secara eksklusif di hsm2m.medium

Siapkan otentikasi kuorum untuk pengguna krypto menggunakan CloudHSM AWS CloudHSM CLI

[Topik ini menjelaskan cara mengonfigurasi CloudHSM Anda untuk otentikasi kuorum oleh pengguna krypto.](#) Lakukan langkah-langkah ini sekali selama pengaturan awal. Untuk manajemen dan penggunaan kunci selanjutnya, lihat [Manajemen kunci dan penggunaan dengan otentikasi kuorum diaktifkan untuk menggunakan AWS CloudHSM CloudHSM CLI](#).

Topik

- [Prasyarat](#)
- [Langkah 1. Buat dan daftarkan kunci untuk penandatanganan](#)
- [Langkah 2. Tetapkan nilai kuorum kunci selama pembuatan kunci](#)

Prasyarat

- Keakraban dengan [CloudHSM CLI](#)

Langkah 1. Buat dan daftarkan kunci untuk penandatanganan

Untuk menggunakan otentikasi kuorum, setiap pengguna krypto harus menyelesaikan semua langkah berikut:

Topik

- [Buat key pair RSA](#)
- [Buat token pendaftaran](#)
- [Tanda tangani token pendaftaran yang belum ditandatangani](#)
- [Daftarkan kunci publik dengan HSM](#)

Buat key pair RSA

Ada banyak cara berbeda untuk membuat dan melindungi pasangan kunci. Contoh berikut menunjukkan cara melakukannya dengan [OpenSSL](#).

Example— Membuat kunci privat dengan OpenSSL

Contoh berikut menunjukkan bagaimana menggunakan OpenSSL untuk membuat kunci RSA 2048-bit. Untuk menggunakan contoh ini, ganti `<crypto_user1.key>` dengan nama file tempat Anda ingin menyimpan kunci.

```
$ openssl genrsa -out <crypto_user1.key>
Generating RSA private key, 2048 bit long modulus
.....+++
.+++
e is 65537 (0x10001)
```

Berikutnya, hasilkan kunci publik menggunakan kunci privat yang baru saja Anda buat.

Example— Membuat kunci publik dengan OpenSSL

Contoh berikut menunjukkan bagaimana menggunakan OpenSSL untuk membuat kunci publik dari kunci privat yang baru saja Anda buat.

```
$ openssl rsa -in crypto_user1.key -outform PEM -pubout -out crypto_user1.pub
writing RSA key
```

Buat token pendaftaran

Anda membuat token dan menandatanganiya dengan kunci privat yang baru saja Anda hasilkan pada langkah sebelumnya.

Buat token pendaftaran

1. Gunakan perintah berikut untuk memulai CloudHSM CLI:

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Buat token registrasi dengan menjalankan perintah [quorum token-sign generate](#):

```
aws-cloudhsm > quorum token-sign generate --service registration --token /path/
tokenfile
{
  "error_code": 0,
  "data": {
    "path": "/path/tokenfile"
  }
}
```

3. Perintah [quorum token-sign generate menghasilkan](#) token registrasi di jalur file yang ditentukan. Periksa file token:

```
$ cat /path/tokenfile
{
  "version": "2.0",
  "tokens": [
    {
      "approval_data": <approval data in base64 encoding>,
      "unsigned": <unsigned token in base64 encoding>,
      "signed": ""
    }
  ]
}
```

File token terdiri dari yang berikut:

- approval_data: Token data acak yang dikodekan base64 yang data mentahnya tidak melebihi maksimum 245 byte.
- unsigned: Token base64 yang dikodekan dan SHA256 di-hash dari approval_data.
- ditandatangani: Token bertanda tangan (tanda tangan) yang disandikan base64 dari token yang tidak ditandatangani, menggunakan kunci pribadi RSA 2048-bit yang sebelumnya dibuat dengan OpenSSL.

Anda menandatangani token yang tidak ditandatangani dengan kunci pribadi untuk menunjukkan bahwa Anda memiliki akses ke kunci pribadi. Anda akan memerlukan file token pendaftaran yang diisi penuh dengan tanda tangan dan kunci publik untuk mendaftarkan pengguna kripto sebagai pengguna kuorum dengan cluster. AWS CloudHSM

Tanda tangani token pendaftaran yang belum ditandatangani

1. Dekode token unsigned base64 yang dikodekan dan letakkan ke dalam file biner:

```
$ echo -n '6BMUj6mUjjko6ZLCEdzG1WpR5sILhFJfqhW1ej30q1g=' | base64 -d >
crypto_user.bin
```

2. Gunakan OpenSSL dan kunci pribadi untuk menandatangani token pendaftaran biner yang sekarang tidak ditandatangani dan membuat file tanda tangan biner:

```
$ openssl pkeyutl -sign \
-inkey crypto_user1.key \
-pkeyopt digest:sha256 \
-keyform PEM \
-in crypto_user.bin \
-out crypto_user.sig.bin
```

3. Mengkodekan tanda tangan biner ke base64:

```
$ base64 -w0 crypto_user.sig.bin > crypto_user.sig.b64
```

4. Salin dan tempel tanda tangan yang dikodekan base64 ke dalam file token:

```
{
  "version": "2.0",
  "tokens": [
    {
      "approval_data": <approval data in base64 encoding>,
      "unsigned": <unsigned token in base64 encoding>,
      "signed": <signed token in base64 encoding>
    }
  ]
}
```

Daftarkan kunci publik dengan HSM

Setelah membuat kunci, pengguna krypto harus mendaftarkan kunci publik dengan cluster. AWS CloudHSM

1. Mulai CloudHSM CLI:

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Masuk sebagai pengguna krypto yang kunci publiknya ingin Anda daftarkan.

```
aws-cloudhsm > login --username crypto_user1 --role crypto-user
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "crypto_user1",
    "role": "crypto-user"
  }
}
```

3. Daftarkan kunci publik dengan [Daftarkan strategi kuorum tanda token pengguna menggunakan CloudHSM CLI](#). Untuk informasi selengkapnya, lihat contoh berikut atau gunakan perintah `help user change-quorum token-sign register`.

Example— Daftarkan kunci publik dengan AWS CloudHSM cluster

Contoh berikut menunjukkan cara menggunakan `user change-quorum token-sign register` perintah di CloudHSM CLI untuk mendaftarkan kunci publik `crypto-user` dengan HSM. Untuk menggunakan perintah ini, pengguna krypto harus masuk ke HSM. Ganti nilai-nilai ini dengan nilai Anda sendiri:

```
aws-cloudhsm > user change-quorum token-sign register --public-key </path/
crypto_user.pub> --signed-token </path/tokenfile>
{
```

```
"error_code": 0,  
"data": {  
  "username": "crypto_user1",  
  "role": "crypto-user"  
}  
}
```

Note

/path/crypto_user.pub: Jalur file ke file PEM kunci publik

Wajib: Ya

/path/token_file: Filepath dengan token yang ditandatangani oleh kunci pribadi pengguna

Wajib: Ya

4. Setelah semua pengguna krypto mendaftarkan kunci publik mereka, output dari user list perintah menunjukkan ini di bidang quorum, yang menyatakan strategi quorum yang diaktifkan yang digunakan.

Dalam contoh ini, AWS CloudHSM cluster memiliki dua HSMs, masing-masing dengan pengguna krypto yang sama, seperti yang ditunjukkan pada output berikut dari perintah. user list Untuk informasi selengkapnya tentang membuat pengguna, lihat [Manajemen pengguna dengan CloudHSM CLI](#).

```
aws-cloudhsm > user list  
{  
  "error_code": 0,  
  "data": {  
    "users": [  
      {  
        "username": "admin",  
        "role": "admin",  
        "locked": "false",  
        "mfa": [],  
        "quorum": [],  
        "cluster-coverage": "full"  
      },  
      {  
        "username": "crypto_user1",  
        "role": "crypto-user",  
        "locked": "false",  
        "mfa": [],
```

```

    "quorum": [
      {
        "strategy": "token-sign",
        "status": "enabled"
      }
    ],
    "cluster-coverage": "full"
  },
  {
    "username": "crypto_user2",
    "role": "crypto-user",
    "locked": "false",
    "mfa": [],
    "quorum": [
      {
        "strategy": "token-sign",
        "status": "enabled"
      }
    ],
    "cluster-coverage": "full"
  },
  {
    "username": "crypto_user3",
    "role": "crypto-user",
    "locked": "false",
    "mfa": [],
    "quorum": [
      {
        "strategy": "token-sign",
        "status": "enabled"
      }
    ],
    "cluster-coverage": "full"
  },
  {
    "username": "app_user",
    "role": "internal(APPLIANCE_USER)",
    "locked": "false",
    "mfa": [],
    "quorum": [],
    "cluster-coverage": "full"
  }
]
}

```

```
}
```

Langkah 2. Tetapkan nilai kuorum kunci selama pembuatan kunci

Untuk menggunakan otentikasi kuorum, pengguna krypto harus masuk ke HSM dan kemudian menetapkan nilai kuorum kunci terkait. Ini adalah jumlah minimum persetujuan pengguna krypto yang diperlukan untuk melakukan operasi kunci HSM. management/usage Untuk informasi selengkapnya tentang perintah kunci terkait yang terkait dengan manajemen kunci atau penggunaan kunci, lihat [Layanan dan tipe yang didukung](#).

Hasilkan key pair dengan nilai kuorum kunci yang ditetapkan

1. Gunakan perintah berikut untuk memulai CloudHSM CLI:

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Menggunakan CloudHSM CLI, masuk sebagai pengguna krypto.

```
aws-cloudhsm > login --username crypto_user1 --role crypto-user
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "crypto_user1",
    "role": "crypto-user"
  }
}
```

Contoh ini menghasilkan key pair RSA yang memiliki nilai kuorum kunci dua (2) yang ditetapkan untuk operasi manajemen kunci dan penggunaan kunci. Anda dapat memilih nilai dari nol (0) hingga delapan (8), hingga jumlah total pengguna krypto di HSM. Dalam contoh ini, HSM memiliki tiga (3) pengguna krypto, sehingga nilai maksimum yang mungkin adalah tiga (3). Perhatikan bahwa dalam

contoh ini kita berbagi kunci dengan *<crypto_user2>* selama pembuatan kunci. Perhatikan juga bahwa kunci publik tidak memiliki nilai kuorum.

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
--public-exponent 65537 \
--modulus-size-bits 2048 \
--public-label rsa-public-key-example \
--private-label rsa-private-key-example \
--public-attributes verify=true \
--private-attributes sign=true
--share-crypto-users crypto_user2 \
--manage-private-key-quorum-value 2 \
--use-private-key-quorum-value 2
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x00000000000640006",
      "key-info": {
        "key-owners": [
          {
            "username": "crypto_user",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-public-key-example",
        "id": "0x",
        "check-value": "0x218f50",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
```

```

    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": false,
    "sign": false,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 512,
    "public-exponent": "0x010001",
    "modulus":
"0xbdf471a3d2a869492f51c767bece8780730ae6479a9a75efffe7cea3594fb28ca518630e7b1d988b45d2fedc830
    "modulus-size-bits": 2048
  }
},
"private_key": {
  "key-reference": "0x00000000000640007",
  "key-info": {
    "key-owners": [
      {
        "username": "crypto_user",
        "key-coverage": "full"
      }
    ],
    "shared-users": [
      {
        "username": "crypto_user2",
        "key-coverage": "full"
      }
    ],
    "key-quorum-values": {
      "manage-key-quorum-value": 2,
      "use-key-quorum-value": 2
    },
    "cluster-coverage": "full"
  },
  "attributes": {
    "key-type": "rsa",
    "label": "rsa-private-key-example",

```

```

    "id": "0x",
    "check-value": "0x218f50",
    "class": "private-key",
    "encrypt": false,
    "decrypt": false,
    "token": true,
    "always-sensitive": true,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": false,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 1216,
    "public-exponent": "0x010001",
    "modulus":
      "0xbdf471a3d2a869492f51c767bece8780730ae6479a9a75efffe7cea3594fb28ca518630e7b1d988b45d2fedc830"
    "modulus-size-bits": 2048
  }
}
}
}

```

Saat membuat kunci dengan kontrol kuorum, kunci harus dikaitkan dengan jumlah minimum pengguna yang sama dengan nilai kuorum kunci terbesar. Pengguna terkait termasuk pemilik kunci dan Pengguna Crypto dengan siapa kunci tersebut dibagikan. Untuk menentukan jumlah pengguna minimum untuk berbagi kunci, dapatkan nilai kuorum terbesar antara nilai kuorum penggunaan kunci dan nilai kuorum manajemen kunci dan kurangi 1 untuk memperhitungkan pemilik kunci, yang secara default dikaitkan dengan kunci. Untuk berbagi kunci dengan lebih banyak pengguna, gunakan [Bagikan kunci menggunakan CloudHSM CLI](#) perintah.

Kegagalan untuk berbagi kunci dengan pengguna yang cukup pada pembuatan kunci akan mengakibatkan kegagalan, seperti yang ditunjukkan di bawah ini.

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
--public-exponent 65537 \
--modulus-size-bits 2048 \
--public-label rsa-public-key-example \
--private-label rsa-private-key-example \
--public-attributes verify=true \
--private-attributes sign=true
--share-crypto-users crypto_user2 crypto_user3 \
--manage-private-key-quorum-value 3 \
--use-private-key-quorum-value 4
{
  "error_code": 1,
  "data": "Invalid quorum value provided."
}
```

Manajemen kunci dan penggunaan dengan otentikasi kuorum diaktifkan untuk menggunakan AWS CloudHSM CloudHSM CLI

Setelah Anda mengonfigurasi otentikasi kuorum untuk AWS CloudHSM klaster Anda, pengguna kriptografi tidak dapat melakukan pengelolaan kunci HSM atau operasi penggunaan sendiri jika kunci mereka memiliki nilai kuorum yang terkait. Topik ini menjelaskan bagaimana pengguna kriptografi bisa mendapatkan token sementara untuk melakukan manajemen kunci HSM atau operasi penggunaan kunci.

Note

Setiap token kuorum berlaku untuk satu operasi. Ketika operasi itu berhasil, token tidak lagi valid dan pengguna kriptografi harus mendapatkan token baru. Token kuorum hanya berlaku selama sesi login Anda saat ini. Jika Anda keluar dari CloudHSM CLI atau jika jaringan terputus, token tidak lagi valid dan Anda perlu mendapatkan token baru. Anda hanya dapat menggunakan token CloudHSM dalam CloudHSM CLI. Anda tidak dapat menggunakannya untuk mengautentikasi di aplikasi yang berbeda.

Contoh berikut menunjukkan output ketika pengguna kriptografi mencoba membuat tanda tangan dengan kunci terkait kuorum pada HSM setelah otentikasi kuorum dikonfigurasi. Perintah gagal dengan Quorum Failed kesalahan, yang berarti otentikasi kuorum gagal:

```
aws-cloudhsm > crypto sign rsa-pkcs --key-filter attr.label=rsa-private-key-example --
hash-function sha256 --data YWJmMTIz
```

```
{
  "error_code": 1,
  "data": "Quorum Failed"
}
```

Pengguna krypto harus menyelesaikan tugas-tugas berikut untuk mendapatkan token sementara untuk melakukan manajemen kunci atau operasi penggunaan kunci pada HSM:

Langkah-langkah

- [Langkah 1. Dapatkan token kuorum](#)
- [Langkah 2. Dapatkan tanda tangan dari menyetujui pengguna krypto](#)
- [Langkah 3. Menyetujui token di CloudHSM; cluster dan jalankan operasi](#)

Langkah 1. Dapatkan token kuorum

1. Mulai CloudHSM CLI.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Masuk ke cluster sebagai pengguna krypto.

```
aws-cloudhsm > login --username <crypto_user1> --role crypto-user --
password password123
```

Contoh ini `crypto_user1` masuk ke CloudHSM CLI dengan peran tersebut. `crypto-user` Ganti nilai-nilai ini dengan nilai Anda sendiri.

```
{
  "error_code": 0,
  "data": {
    "username": "crypto_user1",
    "role": "crypto-user"
  }
}
```

```
}
```

3. Hasilkan token kuorum menggunakan perintah. `quorum token-sign generate`

Dalam perintah berikut, `key-usage` mengidentifikasi nama layanan di mana Anda akan menggunakan token yang Anda hasilkan. Dalam hal ini, token adalah untuk operasi penggunaan kunci (`key-usagelayanan`) Contoh ini menggunakan `--filter` bendera untuk mengaitkan token dengan kunci tertentu.

```
aws-cloudhsm > quorum token-sign generate --service key-usage --token </path/
crypto_user1.token> --filter attr.label=rsa-private-key-example
{
  "error_code": 0,
  "data": {
    "path": "/home/crypto_user1.token"
  }
}
```

Contoh ini mendapatkan token kuorum untuk pengguna krypto dengan nama pengguna `crypto_user1` dan menyimpan token ke file bernama. `crypto_user1.token` Untuk menggunakan perintah contoh berikut, ganti nilai ini dengan nilai Anda sendiri:

`quorum token-sign generate` Perintah menghasilkan token kuorum layanan penggunaan kunci di jalur file yang ditentukan. Anda dapat memeriksa file token:

```
$ cat </path/crypto_user1.token>
{
  "version": "2.0",
  "service": "key-usage",
  "key_reference": "0x000000000000680006",
  "approval_data":
  "AAIABQAAABkAAAAAGgABi5CDa9x9VyyRIaFbkSrHgJjcnlwdG9fdXNlcmgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
+GJj8gXo9lKuANGNyeXB0b191c2VyAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGc9AEsAAAAAAAAAAAAA==",
  "token": "5G1goW01QU4fw4QI1bxkPGZV0VoDugFGuSKE/k67ncM=",
  "signatures": []
}
```

File token terdiri dari yang berikut:

- `layanan`: Pengidentifikasi untuk layanan kuorum yang dikaitkan dengan token.
- `key_reference`: Pengidentifikasi untuk kunci yang dikaitkan dengan token kuorum ini.

- `approval_data`: Token data mentah yang dikodekan base64 yang dihasilkan oleh HSM.
- `token`: Token base64 yang dikodekan dan SHA-256 yang di-hash dari `approval_data`
- `tanda tangan`: Array token bertanda tangan yang disandikan base64 (tanda tangan) dari token yang tidak ditandatangani. Setiap tanda tangan pemberi persetujuan dalam bentuk literal objek JSON:

```
{
  "username": "<APPROVER_USERNAME>",
  "role": "<APPROVER_ROLE>",
  "signature": "<APPROVER_RSA2048_BIT_SIGNATURE>"
}
```

Setiap tanda tangan dibuat dari hasil pemberi persetujuan menggunakan kunci pribadi RSA 2048-bit yang sesuai yang kunci publiknya terdaftar di HSM.

4. Validasi token kuorum layanan pengguna baru. `quorum token-sign list` mengonfirmasi bahwa token ada di CloudHSM.

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": [
      {
        "username": "crypto_user",
        "service": "key-usage",
        "key-reference": "0x00000000000680006",
        "minimum-token-count": 2
      }
    ]
  }
}
```

Ini `minimum-token-count` menyajikan tampilan cluster agregat dari jumlah minimum token kunci yang dapat digunakan sesuai dengan nama pengguna, layanan, dan referensi kunci yang diambil dari satu HSM di cluster.

Misalnya, dengan asumsi cluster 2-HSM, jika kami menerima dua (2) token penggunaan kunci yang dihasilkan oleh pengguna `crypto_user1` untuk kunci dengan referensi `0x00000000000680006` dari HSM pertama di cluster dan kami menerima satu (1) token

penggunaan kunci yang dihasilkan oleh pengguna `crypto_user1` untuk kunci dengan referensi `0x0000000000680006` dari HSM lain di cluster, kami akan menampilkan. `"minimum-token-count": 1`

Langkah 2. Dapatkan tanda tangan dari menyetujui pengguna krypto

Pengguna krypto yang memiliki token kuorum harus mendapatkan token yang disetujui oleh pengguna krypto lainnya. Untuk memberikan persetujuan mereka, pengguna krypto =lainnya menggunakan kunci penandatanganan mereka untuk menandatangani token secara kriptografis di luar HSM.

Ada banyak cara yang berbeda untuk menandatangani token. Contoh berikut menunjukkan cara menandatangani token menggunakan [OpenSSL](#). Untuk menggunakan alat penandatanganan yang berbeda, pastikan alat tersebut menggunakan kunci pribadi (kunci penandatanganan) pengguna krypto untuk menandatangani intisari token SHA-256.

Dalam contoh ini, pengguna krypto yang memiliki token (`crypto-user`) membutuhkan setidaknya dua (2) persetujuan. Contoh perintah berikut menunjukkan bagaimana dua (2) pengguna krypto dapat menggunakan OpenSSL untuk menandatangani token secara kriptografis.

1. Dekode token unsigned base64 yang dikodekan dan letakkan ke dalam file biner:

```
$echo -n '5G1goW01QU4fw4QI1bXkPGZV0VoDugFGuSKE/k67ncM=' | base64 -d >
crypto_user1.bin
```

2. Gunakan OpenSSL dan kunci pribadi pemberi persetujuan untuk menandatangani token unsigned kuorum biner untuk layanan pengguna dan membuat file tanda tangan biner:

```
$openssl pkeyutl -sign \
-inkey crypto_user1.key \
-pkeyopt digest:sha256 \
-keyform PEM \
-in crypto_user1.bin \
-out crypto_user1.sig.bin
```

3. Encode tanda tangan biner ke base64:

```
$ base64 -w0 crypto_user1.sig.bin > crypto_user1.sig.b64
```


4. Salin dan tempel tanda tangan yang disandikan base64 ke dalam file token, menggunakan format literal objek JSON yang ditentukan sebelumnya untuk tanda tangan pemberi persetujuan:

```
{
  "version": "2.0",
  "service": "key-usage",
  "key_reference": "0x00000000000680006",
  "approval_data":
    "AAIABQAAABkAAAAAGgABi5CDa9x9VyyRIaFbkSrHgJjcnlwdG9fdXNlcmgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    +GJj8gXo9lKuANGNyeXB0b191c2VyAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGc9AEsAAAAAAAAAAAA==",
  "token": "5GlgoW0lQU4fw4QIlbXkPGZV0VoDugFGuSKE/k67ncM=",
  "signatures": [
    {
      "username": "crypto_user1",
      "role": "crypto-user",
      "signature": "wa7aPzmGwBjcEoZ6jAzYASp841Afg0vcI27Y/
      tGlCj1E9DawnFw5Uf0IJT2Ca7T5XD2ThVkUi0B+dhAomdqYNl6aUUFrJyH9GBJ
      +E0PmA5jNVm25tzeRWBjzneTg4/
      zTeE2reNqrHFHicWnttQLe9jS09J1znuDGWDe0HaBKWUaz2gUIInJRqmeXDsZYdSvZksrqUH5dci/
      RsaDE2+tGiS9g0RcIkFbsPW4HpGe2e5HVzGsqrV803PKlYQv6+fymfcNTTuoxKcHAK0jpl43QSuSIu2gVq7KI8mSmmW
      +oiukaNfLJr+MoDKzAvCGDg4cDArg=="
    },
    {
      "username": "crypto_user2",
      "role": "crypto-user",
      "signature": "wa7aPzmGwBjcEoZ6jAzYASp841Afg0vcI27Y/
      tGlCj1E9DawnFw5Uf0IJT2Ca7T5XD2ThVkUi0B+dhAomdqYNl6aUUFrJyH9GBJ
      +E0PmA5jNVm25tzeRWBjzneTg4/
      zTeE2reNqrHFHicWnttQLe9jS09J1znuDGWDe0HaBKWUaz2gUIInJRqmeXDsZYdSvZksrqUH5dci/
      RsaDE2+tGiS9g0RcIkFbsPW4HpGe2e5HVzGsqrV803PKlYQv6+fymfcNTTuoxKcHAK0jpl43QSuSIu2gVq7KI8mSmmW
      +oiukaNfLJr+MoDKzAvCGDg4cDArg=="
    }
  ]
}
```

Langkah 3. Menyetujui token di CloudHSM; cluster dan jalankan operasi

Setelah pengguna krypto memiliki persetujuan dan tanda tangan yang diperlukan, mereka dapat memasok token itu ke kluster CloudHSM bersama dengan manajemen kunci atau operasi penggunaan kunci.

Pastikan operasi kunci sesuai dengan layanan kuorum yang sesuai yang terkait dengan token kuorum. Untuk informasi lebih lanjut, lihat [Layanan dan tipe yang didukung](#) untuk informasi lebih lanjut.

Selama transaksi, token akan disetujui dalam AWS CloudHSM cluster dan menjalankan operasi kunci yang diminta. Keberhasilan operasi kunci bergantung pada token kuorum yang disetujui yang valid dan operasi kunci yang valid.

Example Hasilkan tanda tangan dengan mekanisme RSA-PKCS

Dalam contoh berikut, pengguna krypto yang masuk membuat tanda tangan dengan kunci pada HSM:

```
aws-cloudhsm > crypto sign rsa-pkcs --key-filter attr.label=rsa-private-key-example --hash-function sha256 --data YWJjMTIz --approval /path/crypto_user1.token
```

```
{
  "error_code": 0,
  "data": {
    "key-reference": "0x00000000000640007",
    "signature":
      "h6hMqXacBrT3x3MXV13RXHdQno0+IQ6iy0kVrGzo23+eoWT0ZZgrSpBCu5KcuP6IYYHw9goQ5CfPf4jI1n05m/
      IUJtF1A1lmcz0HjEy1CJ7ICXNReDRye0U8m43dkJzt00UdkbtkDJGAcxkbKHLZ02uWsGXaQ8b0KhoGwsRAHHF6nldTXquIC
      +pZmUS38ythybney94Wj6fzY0ER8v7VIY5ijQGa3LfxrjSG4aw6QijEEbno5LSf18ahEaVKmVEnDBL54tylCJBGvGsYSY9H
      TDd2wfvP4PaxbFRyyHaw=="
  }
}
```

Jika pengguna crypto mencoba melakukan operasi penggunaan kunci HSM lain dengan token yang sama, gagal:

```
aws-cloudhsm > crypto sign rsa-pkcs --key-filter attr.label=rsa-private-key-example --hash-function sha256 --data YWJjMTIz --approval /home/crypto_user1.token
{
  "error_code": 1,
  "data": "Quorum approval is required for this operation"
}
```

Untuk melakukan operasi kunci HSM lainnya, pengguna krypto harus membuat token kuorum baru, mendapatkan tanda tangan baru dari pemberi persetujuan, dan menjalankan operasi kunci yang diinginkan dengan argumen `--approval` untuk memasok token kuorum.

Gunakan quorum token-sign list untuk memeriksa token yang tersedia. Contoh ini menunjukkan bahwa pengguna krypto tidak memiliki token yang disetujui.

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": []
  }
}
```

Manajemen kunci dengan AWS CloudHSM KMU

Jika menggunakan [seri versi SDK terbaru](#), gunakan [CloudHSM CLI](#) untuk mengelola kunci di kluster Anda. AWS CloudHSM

Jika menggunakan [seri versi SDK sebelumnya](#), Anda dapat mengelola kunci pada modul keamanan perangkat keras (HSM) di AWS CloudHSM cluster Anda menggunakan alat baris perintah key_mgmt_util (KMU). Sebelum Anda dapat mengelola kunci, Anda harus memulai AWS CloudHSM klien, mulai key_mgmt_util, dan masuk ke file. HSMs Untuk informasi selengkapnya, lihat [Memulai dengan key_mgmt_util](#).

- [Menggunakan kunci tepercaya](#) menjelaskan cara menggunakan atribut library PKCS #11 dan CMU untuk membuat kunci tepercaya untuk mengamankan data.
- [Menghasilkan kunci](#) memiliki instruksi untuk menghasilkan kunci, termasuk kunci simetris, kunci RSA, dan kunci EC.
- [Kunci impor](#) memberikan detail tentang cara pemilik kunci mengimpor kunci.
- [Mengekspor kunci](#) memberikan rincian tentang bagaimana pemilik kunci mengekspor kunci.
- [Menghapus kunci](#) memberikan rincian tentang bagaimana pemilik kunci menghapus kunci.
- [Kunci berbagi dan tidak berbagi](#) merinci cara pemilik kunci berbagi dan membatalkan pembagian kunci.

Hasilkan kunci dengan AWS CloudHSM KMU

Untuk menghasilkan kunci pada modul keamanan perangkat keras (HSM), gunakan perintah di AWS CloudHSM key_mgmt_util (KMU) yang sesuai dengan jenis kunci yang ingin Anda hasilkan.

Topik

- [Hasilkan kunci simetris dengan KMU AWS CloudHSM](#)
- [Hasilkan pasangan kunci RSA dengan KMU AWS CloudHSM](#)
- [Hasilkan pasangan kunci ECC \(elliptic curve cryptography\) menggunakan KMU AWS CloudHSM](#)

Hasilkan kunci simetris dengan KMU AWS CloudHSM

Gunakan [genSymKey](#) perintah di AWS CloudHSM key_mgmt_util (KMU) untuk menghasilkan AES dan jenis kunci simetris lainnya. AWS CloudHSM Untuk melihat semua pilihan yang tersedia, gunakan perintah `genSymKey -h`.

Contoh berikut membuat kunci AES 256-bit.

```
Command: genSymKey -t 31 -s 32 -l aes256
Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

Symmetric Key Created.  Key Handle: 524295

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Hasilkan pasangan kunci RSA dengan KMU AWS CloudHSM

Untuk menghasilkan key pair RSA AWS CloudHSM, gunakan perintah [gen Pair di RSAKey AWS CloudHSM key_mgmt_util](#). Untuk melihat semua pilihan yang tersedia, gunakan perintah `genRSAKeyPair -h`.

Contoh berikut menghasilkan pasangan kunci RSA 2048-bit.

```
Command: genRSAKeyPair -m 2048 -e 65537 -l rsa2048
Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair:    public key handle: 524294    private key handle: 524296

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Hasilkan pasangan kunci ECC (elliptic curve cryptography) menggunakan KMU AWS CloudHSM

Untuk menghasilkan key pair ECC AWS CloudHSM, gunakan perintah [gen Pair di ECCKey AWS CloudHSM key_mgmt_util](#). Untuk melihat semua opsi yang tersedia, termasuk daftar kurva elips yang didukung, gunakan perintah `genECCKeyPair -h`.

Contoh berikut menghasilkan pasangan kunci ECC menggunakan kurva elips P-384 yang ditentukan dalam [Publikasi FIPS NIST 186-4](#).

```
Command: genECCKeyPair -i 14 -l ecc-p384
Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair:      public key handle: 524297      private key handle: 524298

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Kunci impor dengan AWS CloudHSM KMU

Untuk mengimpor kunci rahasia—yaitu, kunci simetris dan kunci pribadi asimetris—ke dalam modul keamanan perangkat keras (HSM) menggunakan AWS CloudHSM `key_mgmt_util`, Anda harus terlebih dahulu membuat kunci pembungkus pada HSM. Anda dapat mengimpor kunci publik secara langsung tanpa kunci pembungkus.

Topik

- [Impor kunci rahasia dengan AWS CloudHSM KMU](#)
- [Impor kunci publik dengan AWS CloudHSM KMU](#)

Impor kunci rahasia dengan AWS CloudHSM KMU

Selesaikan langkah-langkah berikut untuk mengimpor kunci rahasia ke dalam AWS CloudHSM menggunakan `key_mgmt_util` (KMU). Sebelum Anda mengimpor kunci rahasia, simpan ke file. Simpan kunci simetris sebagai byte mentah, dan kunci privat asimetris dalam format PEM.

Contoh ini menunjukkan cara mengimpor kunci rahasia teks terang dari file ke HSM. Untuk mengimpor kunci terenkripsi dari file ke HSM, gunakan perintah. [unWrapKey](#)

Untuk mengimpor kunci rahasia

1. Gunakan [genSymKey](#) perintah untuk membuat kunci pembungkus. Perintah berikut membuat kunci pembungkus AES 128-bit yang hanya berlaku untuk sesi saat ini. Anda dapat menggunakan kunci sesi atau kunci persisten sebagai kunci pembungkus.

```
Command: genSymKey -t 31 -s 16 -sess -l import-wrapping-key  
Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS  
  
Symmetric Key Created. Key Handle: 524299  
  
Cluster Error Status  
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

2. Gunakan salah satu dari perintah berikut ini, tergantung pada jenis kunci rahasia yang Anda impor.
 - Untuk mengimpor kunci simetris, gunakan perintah [imSymKey](#). Perintah berikut mengimpor kunci AES dari file bernama `aes256.key` menggunakan kunci pembungkus yang dibuat pada langkah sebelumnya. Untuk melihat semua pilihan yang tersedia, gunakan perintah `imSymKey -h`.

```
Command: imSymKey -f aes256.key -t 31 -l aes256-imported -w 524299  
Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS  
  
Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS  
  
Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS  
  
Symmetric Key Unwrapped. Key Handle: 524300  
  
Cluster Error Status  
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS  
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS  
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

- Untuk mengimpor kunci privat asimetris, gunakan perintah [importPrivateKey](#). Perintah berikut mengimpor kunci privat dari file bernama `rsa2048.key` menggunakan kunci pembungkus

yang dibuat pada langkah sebelumnya. Untuk melihat semua pilihan yang tersedia, gunakan perintah `importPrivateKey -h`.

```
Command: importPrivateKey -f rsa2048.key -l rsa2048-imported -w 524299  
BER encoded key length is 1216  
  
Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS  
  
Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS  
  
Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS  
  
Private Key Unwrapped. Key Handle: 524301  
  
Cluster Error Status  
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS  
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS  
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Impor kunci publik dengan AWS CloudHSM KMU

Gunakan [importPubKey](#) perintah di AWS CloudHSM key_mgmt_util (KMU) untuk mengimpor kunci publik. Untuk melihat semua pilihan yang tersedia, gunakan perintah `importPubKey -h`.

Contoh berikut mengimpor kunci publik RSA dari file bernama `rsa2048.pub`.

```
Command: importPubKey -f rsa2048.pub -l rsa2048-public-imported  
Cfm3CreatePublicKey returned: 0x00 : HSM Return: SUCCESS  
  
Public Key Handle: 524302  
  
Cluster Error Status  
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS  
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS  
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Ekspor kunci dengan AWS CloudHSM KMU

Untuk mengekspor kunci AWS CloudHSM rahasia—yaitu, kunci simetris dan kunci pribadi asimetris—dari modul keamanan perangkat keras (HSM) menggunakan AWS CloudHSM key_mgmt_util

(KMU), Anda harus terlebih dahulu membuat kunci pembungkus. Anda dapat mengekspor kunci publik secara langsung tanpa kunci pembungkus.

Hanya pemilik kunci yang dapat mengekspor kunci. Pengguna dengan siapa kunci dibagi dapat menggunakan kunci dalam operasi kriptografi, tetapi mereka tidak dapat mengekspornya. Saat menjalankan contoh ini, pastikan untuk mengekspor kunci yang Anda buat.

 Important

[exSymKey](#) Perintah menulis salinan plaintext (tidak terenkripsi) dari kunci rahasia ke file. Proses ekspor membutuhkan kunci pembungkus, tetapi kunci dalam file bukankunci dibungkus. Untuk mengekspor salinan kunci yang dibungkus (dienkripsi), gunakan perintah [wrapKey](#).

Topik

- [Ekspor kunci rahasia dengan AWS CloudHSM KMU](#)
- [Ekspor kunci publik dengan AWS CloudHSM KMU](#)

Ekspor kunci rahasia dengan AWS CloudHSM KMU

Selesaikan langkah-langkah berikut untuk mengekspor kunci rahasia dari AWS CloudHSM menggunakan `key_mgmt_util` (KMU).

Untuk mengekspor kunci rahasia

1. Gunakan [genSymKey](#) perintah untuk membuat kunci pembungkus. Perintah berikut membuat kunci pembungkus AES 128-bit yang hanya berlaku untuk sesi saat ini.

```
Command: genSymKey -t 31 -s 16 -sess -l export-wrapping-key
Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

Symmetric Key Created. Key Handle: 524304

Cluster Error Status
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

2. Gunakan salah satu dari perintah berikut ini, tergantung pada jenis kunci rahasia yang Anda ekspor.

- Untuk mengekspor kunci simetris, gunakan [exSymKey](#) perintah. Perintah berikut mengekspor kunci AES ke file bernama `aes256.key.exp`. Untuk melihat semua pilihan yang tersedia, gunakan perintah `exSymKey -h`.

```
Command: exSymKey -k 524295 -out aes256.key.exp -w 524304  
Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS  
  
Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS  
  
Wrapped Symmetric Key written to file "aes256.key.exp"
```

Note

Output perintah mengatakan bahwa “Kunci Simetris Dibungkus” ditulis ke file output. Namun, file output berisi kunci teks terang (tidak dibungkus). Untuk mengekspor kunci dibungkus (dienkripsi) ke file, gunakan perintah [wrapKey](#).

- Untuk mengekspor kunci privat, gunakan perintah `exportPrivateKey`. Perintah berikut mengekspor kunci privat ke file bernama `rsa2048.key.exp`. Untuk melihat semua pilihan yang tersedia, gunakan perintah `exportPrivateKey -h`.

```
Command: exportPrivateKey -k 524296 -out rsa2048.key.exp -w 524304  
Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS  
  
Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS  
  
PEM formatted private key is written to rsa2048.key.exp
```

Ekspor kunci publik dengan AWS CloudHSM KMU

Gunakan `exportPubKey` perintah di AWS CloudHSM `key_mgmt_util` (KMU) untuk mengekspor kunci publik. Untuk melihat semua pilihan yang tersedia, gunakan perintah `exportPubKey -h`.

Contoh berikut mengekspor kunci publik RSA ke file bernama `rsa2048.pub.exp`.

```
Command: exportPubKey -k 524294 -out rsa2048.pub.exp  
PEM formatted public key is written to rsa2048.pub.key
```

```
Cfm3ExportPubKey returned: 0x00 : HSM Return: SUCCESS
```

Hapus kunci dengan KMU dan CMU

Gunakan perintah [deleteKey](#) untuk menghapus kunci, seperti pada contoh berikut. Hanya pemilik kunci yang dapat menghapus kunci.

```
Command: deleteKey -k 524300  
Cfm3DeleteKey returned: 0x00 : HSM Return: SUCCESS  
  
Cluster Error Status  
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS  
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS  
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Bagikan dan batalkan berbagi kunci dengan KMU dan CMU

Pada tahun AWS CloudHSM, CU yang membuat kunci memilikinya. Pemilik mengelola kunci, dapat mengeksport dan menghapusnya, dan dapat menggunakan kunci dalam operasi kriptografi. Pemilik juga dapat berbagi kunci dengan pengguna CU lainnya. Pengguna dengan siapa kunci dibagi dapat menggunakan kunci dalam operasi kriptografi, tetapi mereka tidak dapat mengeksport atau menghapus kunci, atau berbagi dengan pengguna lain.

Anda dapat berbagi kunci dengan pengguna CU lainnya saat Anda membuat kunci, seperti dengan menggunakan `-u` parameter perintah [genSymKey](#) atau [gen RSAKey Pair](#). Untuk berbagi kunci yang ada dengan pengguna HSM yang berbeda, gunakan alat baris perintah [cloudhsm_mgmt_util](#). Ini berbeda dari sebagian besar tugas yang didokumentasikan di bagian ini, yang menggunakan alat baris perintah [key_mgmt_util](#).

Sebelum Anda dapat berbagi kunci, Anda harus memulai `cloudhsm_mgmt_util`, mengaktifkan enkripsi, dan masuk ke file. end-to-end HSMs Untuk berbagi kunci, masuk ke HSM sebagai pengguna kriptografi (CU) yang memiliki kunci. Hanya pemilik kunci yang dapat berbagi kunci.

Gunakan `shareKey` perintah untuk berbagi atau membatalkan pembagian kunci, menentukan pegangan kunci dan pengguna atau pengguna. IDs Untuk berbagi atau membatalkan pembagian dengan lebih dari satu pengguna, tentukan daftar pengguna yang dipisahkan koma. IDs Untuk berbagi kunci, gunakan 1 sebagai parameter terakhir perintah, seperti pada contoh berikut. Untuk membatalkan berbagi, gunakan 0.

```
aws-cloudhsm > shareKey 524295 4 1
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****

Do you want to continue(y/n)? y
shareKey success on server 0(10.0.2.9)
shareKey success on server 1(10.0.3.11)
shareKey success on server 2(10.0.1.12)
```

Berikut ini adalah sintaks untuk perintah shareKey.

```
aws-cloudhsm > shareKey <key handle> <user ID> <Boolean: 1 for share, 0 for unshare>
```

Cara menandai kunci sebagai tepercaya dengan Utilitas AWS CloudHSM Manajemen

Konten di bagian ini memberikan petunjuk tentang penggunaan Utilitas AWS CloudHSM manajemen (CMU) untuk menandai kunci sebagai tepercaya.

1. Menggunakan perintah [LoginHSM](#), masuk sebagai petugas krypto (CO).
2. Gunakan [Mengatur atribut AWS CloudHSM kunci menggunakan CMU](#) perintah dengan OBJ_ATTR_TRUSTED (value134) disetel ke true (1).

```
aws-cloudhsm > setAttribute <Key Handle> 134 1
```

Cadangan cluster di AWS CloudHSM

AWS CloudHSM membuat backup berkala dari cluster Anda setidaknya sekali setiap 24 jam. Setiap cadangan berisi salinan terenkripsi dari data berikut:

- Pengguna (COs, CUs, dan AUs)
- Materi kunci dan sertifikat
- Konfigurasi dan kebijakan modul keamanan perangkat keras (HSM)

Anda tidak dapat menginstruksikan layanan untuk membuat cadangan, tetapi Anda dapat melakukan tindakan tertentu yang memaksa layanan untuk membuat cadangan. Layanan membuat cadangan saat Anda melakukan salah satu tindakan berikut:

- Mengaktifkan klaster
- Menambahkan HSM ke klaster aktif
- Hapus HSM dari klaster aktif

AWS CloudHSM menghapus cadangan berdasarkan kebijakan penyimpanan cadangan yang Anda tetapkan saat membuat klaster. Untuk informasi tentang pengelolaan kebijakan retensi cadangan, lihat [Konfigurasi retensi cadangan](#).

Topik

- [Bekerja dengan cadangan AWS CloudHSM cluster](#)
- [Hapus cadangan AWS CloudHSM klaster](#)
- [Kembalikan AWS CloudHSM cadangan](#)
- [Konfigurasi kebijakan retensi AWS CloudHSM cadangan](#)
- [Menyalin cadangan AWS CloudHSM klaster di seluruh Wilayah AWS](#)
- [Bekerja dengan cadangan bersama di AWS CloudHSM](#)

Bekerja dengan cadangan AWS CloudHSM cluster

Saat Anda menambahkan modul keamanan perangkat keras (HSM) ke cluster AWS CloudHSM yang sebelumnya berisi satu atau lebih aktif HSMs, layanan mengembalikan cadangan terbaru ke HSM baru. Gunakan cadangan untuk mengelola yang jarang HSMs Anda gunakan. Bila Anda

tidak memerlukan HSM, hapus HSM tersebut untuk memicu cadangan. Kemudian, ketika Anda membutuhkan HSM, buat yang baru di cluster yang sama, dan tindakan ini akan mengembalikan cadangan yang sebelumnya Anda buat dengan operasi hapus HSM.

Menghapus kunci kedaluwarsa atau pengguna yang tidak aktif

Anda mungkin ingin menghapus materi kriptografi yang tidak diinginkan dari lingkungan Anda seperti kunci kedaluwarsa atau pengguna yang tidak aktif. Ini adalah proses dua langkah. Pertama, hapus materi ini dari HSM Anda. Selanjutnya, hapus semua cadangan yang ada. Mengikuti proses ini memastikan Anda tidak memulihkan informasi yang dihapus ketika menginisialisasi klaster baru dari cadangan. Untuk informasi lebih lanjut, lihat [the section called “Hapus cadangan”](#).

Mempertimbangkan pemulihan bencana

Anda dapat membuat klaster dari cadangan. Anda mungkin ingin melakukan ini untuk menetapkan titik pemulihan untuk klaster Anda. Menominasikan cadangan yang berisi semua pengguna, materi kunci, sertifikat yang Anda inginkan di titik pemulihan Anda, dan kemudian gunakan cadangan tersebut untuk membuat sebuah klaster baru. Untuk informasi selengkapnya tentang cara membuat klaster dari cadangan, lihat [Membuat cluster dari backup](#).

Anda juga dapat menyalin cadangan klaster ke wilayah yang berbeda, di mana Anda dapat membuat klaster baru sebagai tiruan dari aslinya. Anda mungkin ingin melakukan ini karena sejumlah alasan, termasuk penyederhanaan proses pemulihan bencana. Untuk informasi selengkapnya tentang menyalin cadangan ke wilayah, lihat [Menyalin cadangan di seluruh Wilayah](#).

Hapus cadangan AWS CloudHSM klaster

Setelah Anda menghapus cadangan AWS CloudHSM cluster, layanan menyimpan cadangan selama tujuh hari, selama waktu itu Anda dapat memulihkan cadangan. Setelah periode tujuh hari, Anda tidak bisa lagi memulihkan cadangan. Untuk informasi selengkapnya tentang mengelola cadangan, lihat [Cadangan cluster](#).

Tabel berikut menjelaskan cara menghapus cadangan.

Console

Untuk menghapus cadangan (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.

2. Untuk mengubah Wilayah AWS, gunakan pemilih Wilayah di sudut kanan atas halaman.
3. Di panel navigasi, pilih Cadangan.
4. Pilih cadangan untuk dihapus.
5. Untuk menghapus cadangan yang dipilih, pilih Tindakan, Hapus.

Kotak dialog Hapus cadangan akan muncul.

6. Pilih Hapus.

Status cadangan berubah menjadi `PENDING_DELETE`. Anda dapat memulihkan cadangan yaitu penghapusan tunda hingga 7 hari setelah Anda meminta penghapusan.

Untuk mencantumkan cadangan (AWS CLI)

- Untuk melihat daftar semua cadangan di kondisi `PENDING_DELETION`, jalankan perintah `describe-backups` dan sertakan `states=PENDING_DELETION` sebagai filter.

```
$ aws cloudhsmv2 describe-backups --filters states=PENDING_DELETION
{
  "Backups": [
    {
      "BackupId": "backup-ro5c4er4aac",
      "BackupState": "PENDING_DELETION",
      "ClusterId": "cluster-dygnwhmscg5",
      "CreateTimestamp": 1534461854.64,
      "DeleteTimestamp": 1536339805.522,
      "HsmType": "hsm2m.medium",
      "Mode": "NON_FIPS",
      "NeverExpires": false,
      "TagList": []
    }
  ]
}
```

AWS CLI

Periksa status cadangan atau temukan ID dengan menggunakan perintah [describe-backups](#) dari AWS CLI.

Untuk menghapus cadangan (AWS CLI)

- Pada prompt perintah, jalankan perintah [delete-backup](#), lewati ID cadangan yang akan dihapus.

```
$ aws cloudhsmv2 delete-backup --backup-id <backup ID>
{
  "Backup": {
    "CreateTimestamp": 1534461854.64,
    "ClusterId": "cluster-dygnwhmscg5",
    "BackupId": "backup-ro5c4er4aac",
    "BackupState": "PENDING_DELETION",
    "DeleteTimestamp": 1536339805.522,
    "HsmType": "hsm1.medium",
    "Mode": "FIPS"
  }
}
```

AWS CloudHSM API

Lihat [DeleteBackup](#) untuk mempelajari cara menghapus cadangan dengan menggunakan API.

Kembalikan AWS CloudHSM cadangan

AWS CloudHSM menahan cadangan yang dihapus selama tujuh hari, selama waktu itu Anda dapat memulihkan cadangan. Setelah periode tujuh hari, Anda tidak bisa lagi memulihkan cadangan. Untuk informasi selengkapnya tentang mengelola cadangan, lihat [Cadangan cluster](#).

Tabel berikut menjelaskan cara menghapus cadangan.

Console

Untuk memulihkan cadangan (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Untuk mengubah Wilayah AWS, gunakan pemilih Wilayah di sudut kanan atas halaman.
3. Di panel navigasi, pilih Cadangan.
4. Pilih cadangan di kondisi PENDING_DELETE untuk dipulihkan.
5. Untuk memulihkan cadangan yang dipilih, pilih Tindakan, Pulihkan.

AWS CLI

Untuk memulihkan cadangan (AWS CLI)

- Untuk memulihkan cadangan, keluarkan perintah [restore-backup](#), lewati ID cadangan yang ada di kondisi PENDING_DELETION.

```
$ aws cloudhsmv2 restore-backup --backup-id <backup ID>
{
  "Backup": {
    "ClusterId": "cluster-dygnwhmscg5",
    "CreateTimestamp": 1534461854.64,
    "BackupState": "READY",
    "BackupId": "backup-ro5c4er4aac"
  }
}
```

AWS CloudHSM API

Lihat [RestoreBackup](#) untuk mempelajari cara memulihkan cadangan dengan menggunakan API.

Konfigurasi kebijakan retensi AWS CloudHSM cadangan

AWS CloudHSM membersihkan cadangan berdasarkan kebijakan retensi cadangan yang Anda tetapkan saat membuat kluster. Kebijakan penyimpanan cadangan berlaku untuk kluster. Jika Anda memindahkan cadangan ke wilayah yang berbeda, cadangan tidak lagi terkait dengan kluster dan tidak lagi memiliki kebijakan penyimpanan cadangan. Anda harus menghapus cadangan apa pun yang tidak terkait dengan cluster secara manual. AWS CloudHSM tidak menghapus cadangan terakhir cluster. Untuk cluster yang dimigrasi dari hsm1.medium, AWS CloudHSM pertahankan cadangan hsm1.medium terbaru hingga hsm1.medium mencapai akhir masa pakai.

[AWS CloudTrail](#) melaporkan cadangan yang ditandai untuk penghapusan. Anda dapat memulihkan cadangan yang layanan bersihkan seperti Anda akan memulihkan [pencadangan yang dihapus secara manual](#). Untuk mencegah kondisi balapan, Anda harus mengubah kebijakan penyimpanan cadangan untuk kluster sebelum Anda memulihkan cadangan yang dihapus oleh layanan. Jika Anda ingin mempertahankan kebijakan penyimpanan yang sama dan mempertahankan cadangan pilihan, Anda dapat menentukan bahwa layanan [mengecualikan cadangan](#) dari kebijakan penyimpanan cadangan kluster.

Untuk informasi lebih lanjut tentang AWS CloudHSM harga, lihat [Kurangi biaya dengan menskalakan kebutuhan Anda](#).

Retensi cadangan terkelola

Cluster yang dibuat sebelum 18 November 2020 memiliki kebijakan retensi cadangan 90 hari ditambah usia cluster. Misalnya, jika Anda membuat klaster pada 18 November 2019, layanan akan menetapkan kebijakan retensi cadangan klaster Anda selama satu tahun ditambah 90 hari (455 hari). Anda dapat mengatur periode ini ke angka apa pun antara 7 dan 379 hari. AWS CloudHSM tidak menghapus cadangan terakhir cluster. Untuk informasi lebih lanjut tentang mengelola cadangan, lihat [Cadangan cluster](#).

Note

Anda dapat memilih keluar dari retensi cadangan terkelola sama sekali dengan menghubungi [AWS Dukungan](#).

Tabel berikut menjelaskan cara mengatur retensi cadangan.

Console

Untuk mengatur konfigurasi kebijakan retensi cadangan (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Untuk mengubah Wilayah AWS, gunakan pemilih Wilayah di sudut kanan atas halaman.
3. Klik ID klaster dari klaster dalam keadaan aktif untuk mengelola kebijakan penyimpanan cadangan untuk klaster tersebut.
4. Untuk mengubah kebijakan penyimpanan cadangan, pilih Tindakan, Ubah periode penyimpanan cadangan.

Kotak dialog Ubah periode penyimpanan cadangan akan muncul.

5. Di Periode penyimpanan cadangan (dalam hari), ketik nilai antara 7 dan 379 hari.
6. Pilih Ubah periode retensi cadangan.

Untuk mengecualikan atau menyertakan cadangan dari kebijakan penyimpanan cadangan (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Untuk melihat cadangan Anda, di panel navigasi pilih Cadangan.
3. Klik ID cadangan dalam keadaan Siap untuk mengecualikan atau menyertakan.
4. Pada Detail Cadangan, lakukan salah satu tindakan berikut.
 - Untuk mengecualikan cadangan dengan tanggal di Waktu kedaluwarsa, pilih Tindakan, Nonaktifkan kedaluwarsa.
 - Untuk menyertakan cadangan yang tidak kedaluwarsa, pilih Tindakan, Gunakan kebijakan penyimpanan klaster.

AWS CLI

Untuk mengatur konfigurasi kebijakan penyimpanan cadangan (AWS CLI)

- Pada prompt perintah, keluarkan perintah modify-cluster. Tentukan ID klaster dan kebijakan penyimpanan cadangan.

```
$ aws cloudhsmv2 modify-cluster --cluster-id <cluster ID> \
                                --backup-retention-policy
                                Type=DAYS,Value=<number of days to retain backups>
{
  "Cluster": {
    "BackupPolicy": "DEFAULT",
    "BackupRetentionPolicy": {
      "Type": "DAYS",
      "Value": 90
    },
  },
  "Certificates": {},
  "ClusterId": "cluster-kdmrayrc7gi",
  "CreateTimestamp": 1504903546.035,
  "Hsms": [],
  "HsmType": "hsm1.medium",
  "SecurityGroup": "sg-40399d28",
  "State": "ACTIVE",
  "SubnetMapping": {
    "us-east-2a": "subnet-f1d6e798",
    "us-east-2c": "subnet-0e358c43",
    "us-east-2b": "subnet-40ed9d3b"
  }
}
```

```

    },
    "TagList": [
      {
        "Key": "Cost Center",
        "Value": "12345"
      }
    ],
    "VpcId": "vpc-641d3c0d"
  }
}

```

Untuk mengecualikan cadangan dari kebijakan penyimpanan cadangan (AWS CLI)

- Pada prompt perintah, keluarkan perintah `modify-backup-attributes`. Tentukan ID cadangan dan tetapkan bendera tanpa kedaluwarsa untuk mempertahankan cadangan.

```

$ aws cloudhsmv2 modify-backup-attributes --backup-id <backup ID> \
                                           --never-expires
{
  "Backup": {
    "BackupId": "backup-ro5c4er4aac",
    "BackupState": "READY",
    "ClusterId": "cluster-dygnwhmscg5",
    "NeverExpires": true
  }
}

```

Untuk menyertakan cadangan dalam kebijakan penyimpanan cadangan (AWS CLI)

- Pada prompt perintah, keluarkan perintah `modify-backup-attributes`. Tentukan ID cadangan dan atur `no-never-expires` tanda untuk menyertakan cadangan dalam kebijakan penyimpanan cadangan, yang berarti layanan pada akhirnya akan menghapus cadangan.

```

$ aws cloudhsmv2 modify-backup-attributes --backup-id <backup ID> \
                                           --no-never-expires
{
  "Backup": {
    "BackupId": "backup-ro5c4er4aac",
    "BackupState": "READY",
    "ClusterId": "cluster-dygnwhmscg5",

```

```
    "NeverExpires": false
  }
}
```

AWS CloudHSM API

Lihat topik berikut untuk mempelajari cara mengelola penyimpanan cadangan dengan menggunakan API.

- [ModifyCluster](#)
- [ModifyBackupAttributes](#)

Menyalin cadangan AWS CloudHSM klaster di seluruh Wilayah AWS

[Anda dapat menyalin cadangan AWS CloudHSM klaster di seluruh Wilayah karena berbagai alasan, termasuk ketahanan lintas wilayah, beban kerja global, dan pemulihan bencana.](#) Setelah Anda menyalin cadangan, cadangan muncul di wilayah tujuan dengan status `CREATE_IN_PROGRESS`. Setelah berhasil menyelesaikan salinan, status cadangan berubah menjadi `READY`. Jika salinan gagal, status cadangan akan berubah jadi `DELETED`. Periksa parameter masukan Anda untuk kesalahan dan pastikan bahwa cadangan sumber tertentu tidak dalam kondisi `DELETED` sebelum menjalankan kembali operasi. Untuk informasi tentang cadangan atau cara membuat sebuah klaster dari cadangan, lihat [Cadangan cluster](#) atau [Membuat cluster dari backup](#).

Perhatikan hal-hal berikut:

- Untuk menyalin cadangan klaster ke wilayah tujuan, akun Anda harus memiliki izin kebijakan IAM yang tepat. Untuk menyalin cadangan ke wilayah yang berbeda, kebijakan IAM Anda harus mengizinkan akses ke wilayah sumber di mana cadangan terletak. Setelah disalin di seluruh wilayah, kebijakan IAM Anda harus mengizinkan akses ke wilayah tujuan untuk berinteraksi dengan cadangan yang disalin, yang mencakup penggunaan operasi [CreateCluster](#). Untuk informasi lebih lanjut, lihat [Buat administrator IAM](#).
- Klaster asli dan klaster yang mungkin dibangun dari cadangan di wilayah tujuan tidak terkait. Anda harus mengelola masing-masing klaster ini secara mandiri. Untuk informasi selengkapnya, lihat [klaster](#).

- Cadangan tidak dapat disalin antar partisi. AWS Cadangan dapat disalin antara wilayah AWS terbatas dan wilayah standar dalam partisi yang sama.

Salin cadangan ke Wilayah yang berbeda (konsol)

Untuk menyalin cadangan ke Wilayah yang berbeda (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Untuk mengubah Wilayah AWS, gunakan pemilih Wilayah di sudut kanan atas halaman.
3. Di panel navigasi, pilih Cadangan.
4. Pilih cadangan untuk disalin ke wilayah yang berbeda.
5. Untuk menyalin cadangan yang dipilih, pilih Tindakan, Salin cadangan ke wilayah lain.

Kotak dialog Salin cadangan ke wilayah lain akan muncul.

6. Di Wilayah tujuan, pilih wilayah dari Pilih wilayah.
7. (Opsional) Ketik kunci tanda dan nilai tanda opsional. Untuk menambahkan lebih dari satu tanda ke klaster, pilih Tambahkan tanda.
8. Pilih Salin cadangan.

Salin cadangan ke Wilayah yang berbeda (AWS CLI)

Untuk menentukan ID cadangan, jalankan [describe-backups](#) perintah.

Untuk menyalin cadangan ke berbagai wilayah (AWS CLI)

- Pada jendela perintah, jalankan perintah [copy-backup-to-region](#). Tentukan wilayah tujuan dan ID cadangan cadangan sumber. Jika Anda menentukan ID cadangan, cadangan terkait akan disalin.

```
$ aws cloudhsmv2 copy-backup-to-region --destination-region <destination region> \  
--backup-id <backup ID>
```

Salin cadangan ke Wilayah yang berbeda (API)AWS CloudHSM

Rujuk topik berikut untuk mempelajari cara menyalin cadangan ke berbagai wilayah dengan menggunakan API.

- [CopyBackupToRegion](#)

Bekerja dengan cadangan bersama di AWS CloudHSM

CloudHSM terintegrasi AWS Resource Access Manager dengan AWS RAM() untuk mengaktifkan berbagi sumber daya. AWS RAM adalah layanan yang memungkinkan Anda untuk berbagi beberapa sumber daya CloudHSM dengan Akun AWS yang lain atau melalui AWS Organizations Dengan AWS RAM, Anda berbagi sumber daya yang Anda miliki dengan membuat pembagian sumber daya. Pembagian sumber daya menentukan sumber daya yang akan dibagikan, dan konsumen yang akan dibagikan. Konsumen dapat mencakup:

- Khusus Akun AWS di dalam atau di luar organisasinya di AWS Organizations
- Unit organisasi di dalam organisasinya di AWS Organizations
- Seluruh organisasi di AWS Organizations

Untuk informasi selengkapnya AWS RAM, lihat [Panduan AWS RAM Pengguna](#).

Topik ini menjelaskan cara berbagi sumber daya yang Anda miliki, dan cara menggunakan sumber daya yang dibagikan dengan Anda.

Daftar Isi

- [Prasyarat untuk berbagi cadangan](#)
- [Berbagi cadangan](#)
- [Membatalkan berbagi cadangan bersama](#)
- [Mengidentifikasi cadangan bersama](#)
- [Izin untuk cadangan bersama](#)
- [Tagihan dan pengukuran](#)

Prasyarat untuk berbagi cadangan

- Untuk berbagi cadangan, Anda harus memilikinya di Akun AWS. Ini berarti bahwa sumber daya harus dialokasikan atau disediakan di akun Anda. Anda tidak dapat membagikan cadangan yang telah dibagikan dengan Anda.
- Untuk berbagi cadangan, itu harus dalam keadaan READY.
- Untuk berbagi cadangan dengan organisasi Anda atau unit organisasi di AWS Organizations, Anda harus mengaktifkan berbagi dengan AWS Organizations. Untuk informasi selengkapnya, lihat [Mengaktifkan Berbagi dengan AWS Organizations](#) di Panduan AWS RAM Pengguna.

Berbagi cadangan

Ketika Anda berbagi cadangan dengan yang lain Akun AWS, Anda memungkinkan mereka untuk memulihkan cluster dari cadangan yang berisi kunci dan pengguna yang disimpan dalam cadangan.

Untuk membagikan cadangan, Anda harus menambahkannya ke berbagi sumber daya. Pembagian sumber daya adalah AWS RAM sumber daya yang memungkinkan Anda berbagi sumber daya Akun AWS. Pembagian sumber daya menentukan sumber daya yang akan dibagikan, dan konsumen yang akan berbagi dengan mereka. Saat membagikan cadangan menggunakan konsol CloudHSM, Anda menambahkannya ke pembagian sumber daya yang ada. Untuk menambahkan cadangan ke pembagian sumber daya baru, Anda harus terlebih dahulu membuat pembagian sumber daya menggunakan [AWS RAM konsol](#).

Jika Anda adalah bagian dari organisasi AWS Organizations dan berbagi dalam organisasi Anda diaktifkan, konsumen di organisasi Anda secara otomatis diberikan akses ke cadangan bersama. Jika tidak, konsumen menerima undangan untuk bergabung dengan pembagian sumber daya dan diberikan akses ke cadangan bersama setelah menerima undangan.

Anda dapat membagikan cadangan yang Anda miliki menggunakan AWS RAM konsol atau AWS CLI.

Untuk berbagi cadangan yang Anda miliki menggunakan AWS RAM konsol

Lihat [Membuat Sumber Daya Bersama](#) di Panduan Pengguna AWS RAM .

Untuk berbagi cadangan yang Anda miliki (AWS RAM perintah)

Gunakan perintah [create-resource-share](#).

Untuk berbagi cadangan yang Anda miliki (perintah CloudHSM)

⚠ Important

Meskipun Anda dapat membagikan cadangan menggunakan operasi `PutResourcePolicy` CloudHSM, sebaiknya AWS Resource Access Manager gunakan AWS RAM() sebagai gantinya. Menggunakan AWS RAM memberikan banyak manfaat karena menciptakan kebijakan untuk Anda, memungkinkan beberapa sumber daya untuk dibagikan sekaligus, dan meningkatkan kemampuan menemukan sumber daya bersama. Jika Anda menggunakan `PutResourcePolicy` dan ingin konsumen dapat menjelaskan cadangan yang Anda bagikan dengan mereka, Anda harus mempromosikan cadangan ke Pembagian AWS RAM Sumber Daya standar menggunakan operasi AWS RAM `PromoteResourceShareCreatedFromPolicy` API.

Gunakan perintah [put-resource-policy](#).

1. Buat file bernama `policy.json` dan salin kebijakan berikut ke dalamnya.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": [
        "cloudhsm:CreateCluster",
        "cloudhsm:DescribeBackups"
      ],
      "Resource": "arn:aws:cloudhsm:us-west-2:111122223333:backup/backup-to-  
share"
    }
  ]
}
```

2. Perbarui `policy.json` dengan ARN cadangan dan pengidentifikasi untuk dibagikan. Contoh berikut memberikan akses read-only ke pengguna root untuk AWS akun yang diidentifikasi oleh 123456789012.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "123456789012"
        ]
      },
      "Action": [
        "cloudhsm:CreateCluster",
        "cloudhsm:DescribeBackups"
      ],
      "Resource": "arn:aws:cloudhsm:us-west-2:123456789012:backup/backup-123"
    }
  ]
}
```

 Important

Anda hanya dapat memberikan izin untuk DescribeBackups di tingkat akun. Ketika Anda berbagi cadangan dengan pelanggan lain, prinsipal apa pun yang memiliki DescribeBackups izin di akun itu dapat menjelaskan cadangan.

3. Jalankan perintah [put-resource-policy](#).

```
$ aws cloudhsmv2 put-resource-policy --resource-arn <resource-arn> --policy file://policy.json
```

 Note

Pada titik ini, konsumen dapat menggunakan cadangan tetapi tidak akan muncul dalam DescribeBackups respons dengan parameter bersama. Langkah selanjutnya

menjelaskan cara mempromosikan pembagian AWS RAM sumber daya agar cadangan dimasukkan dalam respons.

4. Dapatkan AWS RAM sumber daya berbagi ARN.

```
$ aws ram list-resources --resource-owner SELF --resource-arns <backup-arn>
```

Ini mengembalikan respons yang mirip dengan ini:

```
{
  "resources": [
    {
      "arn": "<project-arn>",
      "type": "<type>",
      "resourceShareArn": "<resource-share-arn>",
      "creationTime": "<creation-time>",
      "lastUpdatedTime": "<last-update-time>"
    }
  ]
}
```

Dari respons, salin **<resource-share-arn>** nilai yang akan digunakan pada langkah selanjutnya.

5. Jalankan perintah AWS RAM [promote-resource-share-created-from-policy](#).

```
$ aws ram promote-resource-share-created-from-policy --resource-share-arn <resource-share-arn>
```

6. Untuk memvalidasi bahwa pembagian sumber daya telah dipromosikan, Anda dapat menjalankan AWS RAM [get-resource-shares](#) perintah.

```
$ aws ram get-resource-shares --resource-owner SELF --resource-share-arns <resource-share-arn>
```

Ketika kebijakan telah dipromosikan, yang `featureSet` tercantum dalam tanggapan adalah `STANDARD`. Ini juga berarti cadangan dapat dijelaskan oleh akun baru dalam kebijakan.

Membatalkan berbagi cadangan bersama

Saat Anda membatalkan pembagian sumber daya, konsumen mungkin tidak lagi menggunakannya untuk memulihkan kluster. Konsumen masih dapat mengakses cluster apa pun yang mereka pulihkan dari cadangan bersama.

Untuk membatalkan pembagian cadangan bersama yang Anda miliki, Anda harus menghapusnya dari pembagian sumber daya. Anda dapat melakukan ini menggunakan AWS RAM konsol atau AWS CLI.

Untuk membatalkan pembagian cadangan bersama yang Anda miliki menggunakan konsol AWS RAM

Lihat [Memperbarui Sumber Daya Bersama](#) di Panduan Pengguna AWS RAM .

Untuk membatalkan pembagian cadangan bersama yang Anda miliki (AWS RAM perintah)

Gunakan perintah [disassociate-resource-share](#).

Untuk membatalkan berbagi cadangan bersama yang Anda miliki (perintah CloudHSM)

Gunakan perintah [delete-resource-policy](#).

```
$ aws cloudhsmv2 delete-resource-policy --resource-arn <resource-arn>
```

Mengidentifikasi cadangan bersama

Konsumen dapat mengidentifikasi cadangan yang dibagikan dengan mereka menggunakan konsol CloudHSM dan. AWS CLI

Untuk mengidentifikasi cadangan yang dibagikan dengan Anda menggunakan konsol CloudHSM

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Untuk mengubah AWS Region, gunakan pemilih Wilayah di sudut kanan atas halaman.
3. Di panel navigasi, pilih Cadangan.
4. Dalam tabel, pilih tab Cadangan bersama.

Untuk mengidentifikasi cadangan yang dibagikan dengan Anda menggunakan AWS CLI

Gunakan [perintah deskripsi-cadangan](#) dengan `--shared` parameter untuk mengembalikan cadangan yang dibagikan dengan Anda.

Izin untuk cadangan bersama

Izin untuk pemilik

Pemilik cadangan dapat mendeskripsikan dan mengelola cadangan bersama serta menggunakannya untuk memulihkan kluster.

Izin untuk konsumen

Konsumen cadangan tidak dapat memodifikasi cadangan bersama, tetapi mereka dapat mendeskripsikannya dan menggunakannya untuk memulihkan cluster.

Tagihan dan pengukuran

Tidak ada biaya tambahan untuk berbagi cadangan.

Cluster kloning di AWS CloudHSM

Gunakan CloudHSM CLI untuk menyinkronkan cluster di wilayah terpicil, jika cluster di wilayah itu awalnya dibuat dari cadangan cluster di wilayah lain. Katakanlah Anda menyalin sebuah klaster ke daerah lain (tujuan) dan kemudian Anda ingin menyinkronkan perubahan dari klaster asli (sumber). Dalam skenario seperti ini, Anda menggunakan perintah [replika kunci](#) dan [replikasi pengguna](#) untuk menyinkronkan cluster. Jika Anda belum menginstal CloudHSM CLI, lihat instruksi di [???](#)

Topik terkait

- [replikasi pengguna](#)
- [replikasi kunci](#)
- [Menyalin Cadangan Lintas Wilayah](#)

AWS CloudHSM Sumber daya tag

Tag adalah label yang Anda tetapkan ke AWS sumber daya. Anda dapat menetapkan tanda ke klaster AWS CloudHSM . Setiap tanda terdiri atas sebuah kunci tanda dan sebuah nilai tanda, yang keduanya Anda tentukan. Misalnya, kunci tanda mungkin Pusat Biaya dan nilai tag mungkin 12345. Kunci tanda harus unik untuk setiap klaster.

Anda dapat menggunakan tanda untuk berbagai tujuan. Salah satu penggunaan umum adalah untuk mengelompokkan dan melacak biaya AWS . Anda dapat menerapkan tanda yang mewakili kategori bisnis (seperti pusat biaya, nama aplikasi, atau pemilik) untuk mengatur biaya Anda di berbagai layanan. Saat menambahkan tag ke AWS sumber daya, buat AWS laporan alokasi biaya dengan penggunaan dan biaya yang dikumpulkan berdasarkan tag. Anda dapat menggunakan laporan ini untuk melihat AWS CloudHSM biaya Anda dalam hal proyek atau aplikasi, alih-alih melihat semua AWS CloudHSM biaya sebagai item baris tunggal.

Untuk informasi selengkapnya tentang penggunaan tanda untuk alokasi biaya, lihat [Menggunakan Tanda Alokasi Biaya](#) dalam Panduan Pengguna AWS Billing .

Anda dapat menggunakan [AWS CloudHSM konsol](#) atau salah satu [alat baris perintah AWS SDKs](#) [atau](#) untuk menambahkan, memperbarui, membuat daftar, dan menghapus tag.

Topik

- [Menambahkan atau memperbarui tag untuk AWS CloudHSM sumber daya](#)
- [Daftar tag untuk AWS CloudHSM sumber daya](#)
- [Hapus tag dari AWS CloudHSM sumber daya](#)

Menambahkan atau memperbarui tag untuk AWS CloudHSM sumber daya

Anda dapat menambahkan atau memperbarui tanda dari [konsol AWS CloudHSM](#), [AWS Command Line Interface \(AWS CLI\)](#), atau API AWS CloudHSM .

Untuk menambahkan atau memperbarui tanda (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Pilih klaster yang Anda tandai.

3. Pilih Tanda.
4. Untuk menambahkan tanda, lakukan hal berikut:
 - a. Pilih Edit Tanda, lalu pilih Tambahkan Tanda.
 - b. Untuk Kunci, ketikkan kunci untuk tanda tersebut.
 - c. (Opsional) Untuk Nilai, ketikkan nilai untuk tanda tersebut.
 - d. Pilih Simpan.
5. Untuk menambahkan tanda, lakukan hal berikut:
 - a. Pilih Edit Tanda.

 Note

Jika Anda memperbarui kunci tanda untuk tanda yang ada, konsol akan menghapus tanda yang ada dan membuat yang baru.

- b. Ketikkan nilai tanda baru.
- c. Pilih Simpan.

Untuk menambahkan atau memperbarui tanda (AWS CLI)

1. Pada prompt perintah, keluarkan perintah [tag-resource](#), menentukan tanda dan ID klaster yang Anda tandai. Jika Anda tidak tahu ID klaster, keluarkan perintah [describe-clusters](#).

```
$ aws cloudhsmv2 tag-resource --resource-id <cluster ID> \  
--tag-list Key="<tag key>",Value="<tag value>"
```

2. Untuk memperbarui tanda, gunakan perintah yang sama tetapi tentukan kunci tanda yang ada. Bila Anda menentukan nilai tanda baru untuk tanda yang ada, tanda ditimpa dengan nilai baru.

Untuk menambah atau memperbarui tag (AWS CloudHSM API)

- Kirim permintaan [TagResource](#). Tentukan tanda dan ID klaster yang Anda tandai.

Daftar tag untuk AWS CloudHSM sumber daya

Anda dapat mencantumkan tag untuk klaster dari [AWS CloudHSM konsol](#), [AWS CLI](#)file, atau AWS CloudHSM API.

Untuk membuat daftar tanda (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Pilih klaster yang tandanya Anda cantumkan.
3. PilihTanda.

Untuk daftar tanda (AWS CLI)

- Pada prompt perintah, keluarkan perintah [list-tags](#), menentukan ID klaster tanda yang Anda daftar. Jika Anda tidak tahu ID klaster, keluarkan perintah [describe-clusters](#).

```
$ aws cloudhsmv2 list-tags --resource-id <cluster ID>
{
  "TagList": [
    {
      "Key": "Cost Center",
      "Value": "12345"
    }
  ]
}
```

Untuk daftar tag (AWS CloudHSM API)

- Kirim permintaan [ListTags](#), menentukan ID dari klaster yang tandanya Anda daftar.

Hapus tag dari AWS CloudHSM sumber daya

Anda dapat menghapus tag dari AWS CloudHSM klaster dengan menggunakan [AWS CloudHSM konsol](#), file [AWS CLI](#), atau AWS CloudHSM API.

Untuk menghapus tanda (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.

2. Pilih klaster yang tandanya Anda hapus.
3. Pilih Tanda.
4. Pilih Edit Tanda lalu pilih Hapus tanda untuk tanda yang ingin Anda hapus.
5. Pilih Simpan.

Untuk menghapus tanda (AWS CLI)

- Pada prompt perintah, keluarkan perintah [untag-resource](#), menentukan kunci tanda dari tanda yang Anda hapus dan ID klaster yang tandanya Anda hapus. Bila Anda menggunakan AWS CLI untuk menghapus tag, tentukan hanya kunci tag, bukan nilai tag.

```
$ aws cloudhsmv2 untag-resource --resource-id <cluster ID> \  
                                --tag-key-list "<tag key>"
```

Untuk menghapus tag (AWS CloudHSM API)

- Kirim [UntagResource](#) permintaan di AWS CloudHSM API, tentukan ID cluster dan tag yang Anda hapus.

AWS CloudHSM alat baris perintah

Selain AWS Command Line Interface (AWS CLI) yang Anda gunakan untuk mengelola sumber daya AWS Anda, AWS CloudHSM menawarkan alat baris perintah untuk membuat dan mengelola pengguna dan kunci modul keamanan perangkat keras (HSM) pada Anda. HSMs Di AWS CloudHSM, Anda menggunakan CLI yang sudah dikenal untuk mengelola cluster Anda, dan alat baris perintah CloudHSM untuk mengelola HSM Anda.

Ini adalah berbagai alat baris perintah:

Untuk mengelola HSMs dan mengelompokkan

[HSMv2 Perintah cloud di AWS CLI](#) dan [HSM2 PowerShell cmdlet dalam modul Shell AWSPower](#)

- Alat-alat ini mendapatkan, membuat, menghapus, dan menandai AWS CloudHSM cluster dan HSMs:
- [Untuk menggunakan perintah dalam HSMv2 perintah Cloud di CLI, Anda perlu menginstal dan mengkonfigurasi.](#) AWS CLI
- [HSM2 PowerShell cmdlet dalam modul AWSPower Shell](#) tersedia dalam PowerShell modul Windows dan modul Core lintas platform PowerShell .

Untuk mengelola pengguna HSM

[CloudHSM CLI](#)

- Gunakan [CloudHSM](#) CLI untuk membuat pengguna, menghapus pengguna, mencantumkan pengguna, mengubah kata sandi pengguna, dan memperbarui otentikasi multi-faktor pengguna (MFA). Ini tidak termasuk dalam perangkat lunak klien AWS CloudHSM. Untuk panduan cara menginstal alat ini, lihat [Menginstal dan mengkonfigurasi CloudHSM CLI](#).

Alat Pembantu

Dua alat membantu Anda menggunakan AWS CloudHSM alat dan pustaka perangkat lunak:

- [Alat konfigurasi memperbarui file konfigurasi](#) klien CloudHSM Anda. Hal ini memungkinkan AWS CloudHSM untuk menyinkronkan HSMs dalam sebuah cluster.

AWS CloudHSM menawarkan dua versi utama, dan Client SDK 5 adalah yang terbaru. Ini menawarkan berbagai keunggulan dibandingkan Client SDK 3 (seri sebelumnya).

- [pkpspeed](#) mengukur performa perangkat keras HSM Anda secara independen dari pustaka perangkat lunak.

Alat untuk sebelumnya SDKs

Gunakan alat manajemen kunci (KMU) buat, hapus, impor, dan ekspor kunci simetris dan pasangan kunci asimetris:

- [key_mgmt_util](#). Alat ini termasuk dalam perangkat lunak klien AWS CloudHSM .

Gunakan alat manajemen CloudHSM (CMU) untuk membuat dan menghapus pengguna HSM, termasuk menerapkan otentikasi kuorum tugas manajemen pengguna

- [cloudhsm_mgmt_util](#). Alat ini termasuk dalam perangkat lunak klien AWS CloudHSM .

Topik berikut lebih lanjut menjelaskan alat baris perintah yang tersedia untuk mengelola dan menggunakan. AWS CloudHSM

Topik

- [AWS CloudHSM mengkonfigurasi alat](#)
- [AWS CloudHSM Antarmuka Baris Perintah \(CLI\)](#)
- [AWS CloudHSM Utilitas Manajemen \(CMU\)](#)
- [AWS CloudHSM Utilitas Manajemen Kunci \(KMU\)](#)

AWS CloudHSM mengkonfigurasi alat

AWS CloudHSM secara otomatis menyinkronkan data di antara semua modul keamanan perangkat keras (HSM) dalam sebuah cluster. Alat configure memperbarui data HSM dalam file konfigurasi yang digunakan oleh mekanisme sinkronisasi. Gunakan configure untuk menyegarkan data HSM sebelum Anda menggunakan alat baris perintah, terutama ketika HSMs di cluster telah berubah.

AWS CloudHSM mencakup dua versi SDK Klien utama:

- SDK Klien 5: Ini adalah SDK Klien terbaru dan default kami. Untuk informasi tentang manfaat dan keuntungan yang diberikannya, lihat [Manfaat SDK AWS CloudHSM Klien 5](#).
- SDK Klien 3: Ini adalah SDK Klien kami yang lebih lama. Ini mencakup satu set lengkap komponen untuk platform dan aplikasi berbasis bahasa kompatibilitas dan alat manajemen.

Untuk petunjuk tentang migrasi dari Client SDK 3 ke Client SDK 5, lihat. [Migrasi dari SDK AWS CloudHSM Klien 3 ke SDK Klien 5](#)

Topik

- [AWS CloudHSM Alat konfigurasi SDK 5 klien](#)
- [AWS CloudHSM Alat konfigurasi SDK 3 klien](#)

AWS CloudHSM Alat konfigurasi SDK 5 klien

Gunakan alat konfigurasi AWS CloudHSM Client SDK 5 untuk memperbarui file konfigurasi sisi klien.

Setiap komponen dalam Klien SDK 5 termasuk mengatur konfigurasi alat dengan designator komponen dalam nama file alat konfigurasi. Sebagai contoh, pustaka PKCS #11 untuk Klien SDK 5 termasuk alat konfigurasi bernama `configure-pkcs11` pada Linux atau `configure-pkcs11.exe` pada Windows.

Topik

- [AWS CloudHSM Sintaks konfigurasi SDK 5 klien](#)
- [AWS CloudHSM Parameter konfigurasi SDK 5 klien](#)
- [AWS CloudHSM Contoh konfigurasi SDK 5 klien](#)
- [Penyedia OpenSSL Bootstrap](#)
- [Konfigurasi lanjutan untuk alat konfigurasi Client SDK 5](#)
- [AWS CloudHSM Klien SDK 5 topik terkait](#)

AWS CloudHSM Sintaks konfigurasi SDK 5 klien

Tabel berikut menggambarkan sintaks untuk file AWS CloudHSM konfigurasi untuk Client SDK 5. Untuk informasi tentang parameter, lihat [the section called "Parameter"](#).

PKCS #11

```
Usage: configure-pkcs11[ .exe ] [OPTIONS]
```

Options:

```
--disable-certificate-storage
    Disables Certificate Storage
--enable-certificate-storage
    Enables Certificate Storage
-a <HSM ENI IP>...
    The address of the HSM instance
```

```

--cluster-id <CLUSTER ID>
    The id of the cluster containing the HSM instance(s)
--disable-key-availability-check
    Disables key availability check during key use
--enable-key-availability-check
    Enables key availability check during key use
--disable-validate-key-at-init
    Disables parameter validation during initialization of crypto operations
--enable-validate-key-at-init
    Enables parameter validation during initialization of crypto operations
--endpoint <ENDPOINT>
    Specify the AWS CloudHSM API Endpoint
--region <REGION>
    The region of the cluster
--hsm-ca-cert <HSM CA CERTIFICATE FILE>
    The HSM CA certificate file
--log-type <LOG TYPE>
    The log type [possible values: term, file]
--log-file <LOG FILE>
    The log file
--log-level <LOG LEVEL>
    The logging level [possible values: error, warn, info, debug, trace]
--log-rotation <LOG ROTATION>
    The log rotation interval [possible values: never, hourly, daily]
--default-retry-mode <RETRY MODE>
    The default method of retry to use for certain non-terminal failures
    [possible values: off, standard]
--client-cert-hsm-tls-file <CLIENT CERTIFICATE HSM TLS FILE>
    The client certificate used for TLS client-hsm mutual authentication
--client-key-hsm-tls-file <CLIENT KEY HSM TLS FILE>
    The client private key used for TLS client-hsm mutual authentication
-h, --help
    Print help

```

OpenSSL

Usage: configure-dyn[.exe] [OPTIONS]

Options:

```

-a <HSM ENI IP>...
    The address of the HSM instance
--cluster-id <CLUSTER ID>
    The id of the cluster containing the HSM instance(s)

```

```

--disable-key-availability-check
    Disables key availability check during key use
--enable-key-availability-check
    Enables key availability check during key use
--disable-validate-key-at-init
    Disables parameter validation during initialization of crypto operations
--enable-validate-key-at-init
    Enables parameter validation during initialization of crypto operations
--endpoint <ENDPOINT>
    Specify the AWS CloudHSM API Endpoint
--region <REGION>
    The region of the cluster
--hsm-ca-cert <HSM CA CERTIFICATE FILE>
    The HSM CA certificate file
--log-type <LOG TYPE>
    The log type [possible values: term, file]
--log-file <LOG FILE>
    The log file
--log-level <LOG LEVEL>
    The logging level [possible values: error, warn, info, debug, trace]
--log-rotation <LOG ROTATION>
    The log rotation interval [possible values: never, hourly, daily]
--default-retry-mode <RETRY MODE>
    The default method of retry to use for certain non-terminal failures
[possible values: off, standard]
--client-cert-hsm-tls-file <CLIENT CERTIFICATE HSM TLS FILE>
    The client certificate used for TLS client-hsm mutual authentication
--client-key-hsm-tls-file <CLIENT KEY HSM TLS FILE>
    The client private key used for TLS client-hsm mutual authentication
-h, --help
    Print help

```

KSP

Usage: configure-ksp.exe [OPTIONS]

Options:

```

-a <HSM ENI IP>...
    The address of the HSM instance
--server-client-cert-file <CLIENT CERTIFICATE FILE>
    The client certificate used for TLS client-server mutual authentication
--server-client-key-file <CLIENT KEY FILE>
    The client private key used for TLS client-server mutual authentication

```

```

--cluster-id <CLUSTER ID>
    The id of the cluster containing the HSM instance(s)
--disable-key-availability-check
    Disables key availability check during key use
--enable-key-availability-check
    Enables key availability check during key use
--disable-validate-key-at-init
    Disables parameter validation during initialization of crypto operations
--enable-validate-key-at-init
    Enables parameter validation during initialization of crypto operations
--endpoint <ENDPOINT>
    Specify the AWS CloudHSM API Endpoint
--region <REGION>
    The region of the cluster
--hsm-ca-cert <HSM CA CERTIFICATE FILE>
    The HSM CA certificate file
--log-type <LOG TYPE>
    The log type [possible values: term, file]
--log-file <LOG FILE>
    The log file
--log-level <LOG LEVEL>
    The logging level [possible values: error, warn, info, debug, trace]
--log-rotation <LOG ROTATION>
    The log rotation interval [possible values: never, hourly, daily]
--default-retry-mode <RETRY MODE>
    The default method of retry to use for certain non-terminal failures
[possible values: off, standard]
--client-cert-hsm-tls-file <CLIENT CERTIFICATE HSM TLS FILE>
    The client certificate used for TLS client-hsm mutual authentication
--client-key-hsm-tls-file <CLIENT KEY HSM TLS FILE>
    The client private key used for TLS client-hsm mutual authentication
--enable-sdk3-compatibility-mode
    Enables key file usage for KSP
--disable-sdk3-compatibility-mode
    Disables key file usage for KSP
-h, --help
    Print help

```

JCE

Usage: configure-jce[.exe] [OPTIONS]

Options:

```

-a <HSM ENI IP>...
    The address of the HSM instance
--cluster-id <CLUSTER ID>
    The id of the cluster containing the HSM instance(s)
--disable-key-availability-check
    Disables key availability check during key use
--enable-key-availability-check
    Enables key availability check during key use
--disable-validate-key-at-init
    Disables parameter validation during initialization of crypto operations
--enable-validate-key-at-init
    Enables parameter validation during initialization of crypto operations
--endpoint <ENDPOINT>
    Specify the AWS CloudHSM API Endpoint
--region <REGION>
    The region of the cluster
--hsm-ca-cert <HSM CA CERTIFICATE FILE>
    The HSM CA certificate file
--log-type <LOG TYPE>
    The log type [possible values: term, file]
--log-file <LOG FILE>
    The log file
--log-level <LOG LEVEL>
    The logging level [possible values: error, warn, info, debug, trace]
--log-rotation <LOG ROTATION>
    The log rotation interval [possible values: never, hourly, daily]
--default-retry-mode <RETRY MODE>
    The default method of retry to use for certain non-terminal failures
[possible values: off, standard]
--client-cert-hsm-tls-file <CLIENT CERTIFICATE HSM TLS FILE>
    The client certificate used for TLS client-hsm mutual authentication
--client-key-hsm-tls-file <CLIENT KEY HSM TLS FILE>
    The client private key used for TLS client-hsm mutual authentication
-h, --help
    Print help

```

CloudHSM CLI

Usage: configure-cli[.exe] [OPTIONS]

Options:

```

-a <HSM ENI IP>...
    The address of the HSM instance

```



```

--cluster-id <CLUSTER ID>
    The id of the cluster containing the HSM instance(s)
--disable-key-availability-check
    Disables key availability check during key use
--enable-key-availability-check
    Enables key availability check during key use
--disable-validate-key-at-init
    Disables parameter validation during initialization of crypto operations
--enable-validate-key-at-init
    Enables parameter validation during initialization of crypto operations
--endpoint <ENDPOINT>
    Specify the AWS CloudHSM API Endpoint
--region <REGION>
    The region of the cluster
--hsm-ca-cert <HSM CA CERTIFICATE FILE>
    The HSM CA certificate file
--log-type <LOG TYPE>
    The log type [possible values: term, file]
--log-file <LOG FILE>
    The log file
--log-level <LOG LEVEL>
    The logging level [possible values: error, warn, info, debug, trace]
--log-rotation <LOG ROTATION>
    The log rotation interval [possible values: never, hourly, daily]
--default-retry-mode <RETRY MODE>
    The default method of retry to use for certain non-terminal failures
[possible values: off, standard]
--client-cert-hsm-tls-file <CLIENT CERTIFICATE HSM TLS FILE>
    The client certificate used for TLS client-hsm mutual authentication
--client-key-hsm-tls-file <CLIENT KEY HSM TLS FILE>
    The client private key used for TLS client-hsm mutual authentication
-h, --help
    Print help

```

OpenSSL Provider

Usage: configure-openssl-provider[.exe] [OPTIONS]

Options:

```

-a <HSM ENI IP>...
    The address of the HSM instance
--cluster-id <CLUSTER ID>
    The id of the cluster containing the HSM instance(s)

```

```

--disable-key-availability-check
    Disables key availability check during key use
--enable-key-availability-check
    Enables key availability check during key use
--disable-validate-key-at-init
    Disables parameter validation during initialization of crypto operations
--enable-validate-key-at-init
    Enables parameter validation during initialization of crypto operations
--endpoint <ENDPOINT>
    Specify the AWS CloudHSM API Endpoint
--region <REGION>
    The region of the cluster
--hsm-ca-cert <HSM CA CERTIFICATE FILE>
    The HSM CA certificate file
--log-type <LOG TYPE>
    The log type [possible values: term, file]
--log-file <LOG FILE>
    The log file
--log-level <LOG LEVEL>
    The logging level [possible values: error, warn, info, debug, trace]
--log-rotation <LOG ROTATION>
    The log rotation interval [possible values: never, hourly, daily]
--default-retry-mode <RETRY MODE>
    The default method of retry to use for certain non-terminal failures
[possible values: off, standard]
--client-cert-hsm-tls-file <CLIENT CERTIFICATE HSM TLS FILE>
    The client certificate used for TLS client-hsm mutual authentication
--client-key-hsm-tls-file <CLIENT KEY HSM TLS FILE>
    The client private key used for TLS client-hsm mutual authentication
-h, --help
    Print help

```

AWS CloudHSM Parameter konfigurasi SDK 5 klien

Berikut ini adalah daftar parameter untuk mengkonfigurasi SDK AWS CloudHSM Klien 5.

-a *<ENI IP address>*

Menambahkan alamat IP yang ditentukan ke file konfigurasi Klien SDK 5. Masukkan alamat IP ENI dari HSM dari cluster. Untuk informasi lebih lanjut tentang cara menggunakan opsi ini, lihat [Bootstrap Klien SDK 5](#).

Wajib: Ya

--hsm-ca-cert **<customerCA certificate file path>**

Jalur ke direktori yang menyimpan sertifikat otoritas (CA) yang digunakan sertifikat untuk menghubungkan instans klien EC2 ke kluster. Anda membuat file ini ketika Anda menginisialisasi kluster. Secara default, sistem mencari file ini di lokasi berikut:

Linux

```
/opt/cloudhsm/etc/customerCA.crt
```

Windows

```
C:\ProgramData\Amazon\CloudHSM\customerCA.crt
```

Untuk informasi lebih lanjut tentang menginisialisasi kluster atau menempatkan sertifikat, lihat [???](#) dan [???](#).

Wajib: Tidak

--kluster-id **<cluster ID>**

Membuat panggilan `DescribeClusters` untuk menemukan semua alamat IP antarmuka jaringan elastis (ENI) HSM dalam kluster yang terkait dengan ID kluster. Sistem menambahkan alamat IP ENI ke file AWS CloudHSM konfigurasi.

 Note

Jika Anda menggunakan `--cluster-id` parameter dari instans EC2 dalam VPC yang tidak memiliki akses ke internet publik, maka Anda harus membuat antarmuka VPC endpoint untuk terhubung. AWS CloudHSM Untuk informasi lebih lanjut tentang VPC endpoint, lihat [???](#).

Wajib: Tidak

--titik akhir **<endpoint>**

Tentukan titik akhir AWS CloudHSM API yang digunakan untuk melakukan `DescribeClusters` panggilan. Anda harus menetapkan pilihan ini dalam kombinasi dengan `--cluster-id`.

Wajib: Tidak

--wilayah **<region>**

Tentukan wilayah klaster Anda. Anda harus menetapkan pilihan ini dalam kombinasi dengan --cluster-id.

Jika Anda tidak menyediakan parameter --region, sistem memilih wilayah dengan mencoba untuk membaca variabel lingkungan AWS_DEFAULT_REGION atau AWS_REGION. Jika variabel-variabel tersebut tidak diatur, maka sistem memeriksa wilayah yang terkait dengan profil Anda di file AWS config Anda (biasanya ~/.aws/config) kecuali jika Anda menentukan file yang berbeda di variabel lingkungan AWS_CONFIG_FILE. Jika tidak ada variabel di atas yang diatur, sistem default ke wilayah us-east-1.

Wajib: Tidak

-- client-cert-hsm-tls berkas **<client certificate hsm tls path>**

Jalur ke sertifikat klien yang digunakan untuk otentikasi timbal balik klien TLS-HSM.

Hanya gunakan opsi ini jika Anda telah mendaftarkan setidaknya satu jangkar kepercayaan ke HSM dengan CloudHSM CLI. Anda harus menetapkan pilihan ini dalam kombinasi dengan --client-key-hsm-tls-file.

Wajib: Tidak

-- client-key-hsm-tls berkas **<client key hsm tls path>**

Jalur ke kunci klien yang digunakan untuk otentikasi timbal balik klien TLS-HSM.

Hanya gunakan opsi ini jika Anda telah mendaftarkan setidaknya satu jangkar kepercayaan ke HSM dengan CloudHSM CLI. Anda harus menetapkan pilihan ini dalam kombinasi dengan --client-cert-hsm-tls-file.

Wajib: Tidak

--log-tingkat **<error | warn | info | debug | trace>**

Menentukan tingkat pencatatan minimum yang harus ditulis sistem ke berkas log. Setiap tingkat termasuk tingkat sebelumnya, dengan kesalahan sebagai tingkat minimum dan melacak tingkat maksimum. Ini berarti bahwa jika Anda menentukan kesalahan, sistem hanya menulis kesalahan ke log. Jika Anda menentukan jejak, sistem menulis kesalahan, peringatan, informasi (info), dan pesan debug ke log. Untuk informasi lebih lanjut, lihat [Pencatatan Klien SDK 5](#).

Wajib: Tidak

`--log-rotasi <daily | weekly>`

Menentukan frekuensi sistem memutar log. Untuk informasi lebih lanjut, lihat [Pencatatan Klien SDK 5](#).

Wajib: Tidak

`--log-file <file name with path>`

Menentukan tempat sistem akan menulis berkas log. Untuk informasi lebih lanjut, lihat [Pencatatan Klien SDK 5](#).

Wajib: Tidak

`--log-jenis <term | file>`

Menentukan apakah sistem akan menulis log ke file atau terminal. Untuk informasi lebih lanjut, lihat [Pencatatan Klien SDK 5](#).

Wajib: Tidak

`-h | --help`

Menampilkan bantuan.

Wajib: Tidak

`--disable-key-availability-check`

Tandai untuk menonaktifkan kuorum ketersediaan kunci. Gunakan tanda ini untuk menunjukkan AWS CloudHSM harus menonaktifkan kuorum ketersediaan kunci dan Anda dapat menggunakan kunci yang ada hanya pada satu HSM di cluster. Untuk informasi lebih lanjut tentang penggunaan bendera ini untuk mengatur kuorum ketersediaan kunci, lihat [???](#).

Wajib: Tidak

`--enable-key-availability-check`

Tandai untuk mengaktifkan kuorum ketersediaan kunci. Gunakan tanda ini untuk menunjukkan AWS CloudHSM harus menggunakan kuorum ketersediaan kunci dan tidak mengizinkan Anda menggunakan kunci sampai kunci tersebut ada pada dua HSMs di cluster. Untuk informasi lebih lanjut tentang penggunaan bendera ini untuk mengatur kuorum ketersediaan kunci, lihat [???](#).

Diaktifkan secara default.

Wajib: Tidak

-- disable-validate-key-at -init

Meningkatkan performa dengan menentukan bahwa Anda dapat melewati panggilan inisialisasi untuk memverifikasi izin pada kunci untuk panggilan berikutnya. Berhati-hatilah saat menggunakannya.

Latar belakang: Beberapa mekanisme di pustaka PKCS #11 mendukung operasi multi-bagian di mana panggilan inisialisasi memverifikasi apakah Anda dapat menggunakan kunci untuk panggilan berikutnya. Hal ini memerlukan panggilan verifikasi ke HSM, yang menambahkan latensi untuk keseluruhan operasi. Opsi ini memungkinkan Anda untuk menonaktifkan panggilan berikutnya dan berpotensi meningkatkan performa.

Wajib: Tidak

-- enable-validate-key-at -init

Menentukan bahwa Anda harus menggunakan panggilan inisialisasi untuk memverifikasi izin pada kunci untuk panggilan berikutnya. Ini adalah pilihan default. Gunakan `enable-validate-key-at-init` untuk melanjutkan panggilan inisialisasi ini setelah Anda menggunakan `disable-validate-key-at-init` untuk menanggukannya.

Wajib: Tidak

AWS CloudHSM Contoh konfigurasi SDK 5 klien

Contoh-contoh ini menunjukkan cara menggunakan alat konfigurasi untuk AWS CloudHSM Client SDK 5.

Bootstrap Klien SDK 5

Example

Contoh ini menggunakan parameter `-a` untuk memperbarui data HSM untuk klien SDK 5. Untuk menggunakan `-a` parameter, Anda harus memiliki alamat IP untuk salah satu HSMs di cluster Anda.

PKCS #11 library

Untuk melakukan bootstrap instans EC2 Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 -a <HSM IP addresses>
```

Untuk melakukan bootstrap instans EC2 Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" -a <HSM IP addresses>
```

OpenSSL Dynamic Engine

Untuk melakukan bootstrap instans EC2 Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda.

```
$ sudo /opt/cloudhsm/bin/configure-dyn -a <HSM IP addresses>
```

OpenSSL Dynamic Engine Provider

Untuk melakukan bootstrap instans EC2 Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda.

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider -a <HSM IP addresses>
```

Key Storage Provider (KSP)

Untuk melakukan bootstrap instans EC2 Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" -a <HSM IP addresses>
```

JCE provider

Untuk melakukan bootstrap instans EC2 Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda.

```
$ sudo /opt/cloudhsm/bin/configure-jce -a <HSM IP addresses>
```

Untuk melakukan bootstrap instans EC2 Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" -a <HSM IP  
addresses>
```

CloudHSM CLI

Untuk melakukan bootstrap instans EC2 Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM (s) di cluster Anda.

```
$ sudo /opt/cloudhsm/bin/configure-cli -a <The ENI IPv4 / IPv6 addresses of the  
HSMs>
```

Untuk melakukan bootstrap instans EC2 Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM (s) di cluster Anda.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" -a <The ENI  
IPv4 / IPv6 addresses of the HSMs>
```

Note

Anda dapat menggunakan `--cluster-id` parameter sebagai pengganti `-a <HSM_IP_ADDRESSES>`. Untuk melihat persyaratan penggunaan `--cluster-id`, lihat [AWS CloudHSM Alat konfigurasi SDK 5 klien](#).

Untuk informasi tentang parameter `-a`, lihat [the section called "Parameter"](#).

Tentukan klaster, wilayah, dan titik akhir untuk Client SDK 5

Example

Contoh ini menggunakan parameter `cluster-id` untuk bootstrap klien SDK 5 dengan membuat panggilan `DescribeClusters`.

PKCS #11 library

Untuk melakukan bootstrap instans EC2 Linux untuk Klien SDK 5 dengan **cluster-id**

- Gunakan ID cluster `cluster-1234567` untuk menentukan alamat IP HSM di cluster Anda.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --cluster-id <cluster-1234567>
```

Untuk melakukan bootstrap instans EC2 Windows untuk Klien SDK 5 dengan **cluster-id**

- Gunakan ID cluster `cluster-1234567` untuk menentukan alamat IP HSM di cluster Anda.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --cluster-id <cluster-1234567>
```

OpenSSL Dynamic Engine

Untuk melakukan bootstrap instans EC2 Linux untuk Klien SDK 5 dengan **cluster-id**

- Gunakan ID cluster `cluster-1234567` untuk menentukan alamat IP HSM di cluster Anda.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --cluster-id <cluster-1234567>
```

Key Storage Provider (KSP)

Untuk melakukan bootstrap instans EC2 Windows untuk Klien SDK 5 dengan **cluster-id**

- Gunakan ID cluster `cluster-1234567` untuk menentukan alamat IP HSM di cluster Anda.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --cluster-id <cluster-1234567>
```

JCE provider

Untuk melakukan bootstrap instans EC2 Linux untuk Klien SDK 5 dengan **cluster-id**

- Gunakan ID cluster `cluster-1234567` untuk menentukan alamat IP HSM di cluster Anda.

```
$ sudo /opt/cloudhsm/bin/configure-jce --cluster-id <cluster-1234567>
```

Untuk melakukan bootstrap instans EC2 Windows untuk Klien SDK 5 dengan **cluster-id**

- Gunakan ID cluster `cluster-1234567` untuk menentukan alamat IP HSM di cluster Anda.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --cluster-id <cluster-1234567>
```

CloudHSM CLI

Untuk melakukan bootstrap instans EC2 Linux untuk Klien SDK 5 dengan **cluster-id**

- Gunakan ID cluster `cluster-1234567` untuk menentukan alamat IP HSM di cluster Anda.

```
$ sudo /opt/cloudhsm/bin/configure-cli --cluster-id <cluster-1234567>
```

Untuk melakukan bootstrap instans EC2 Windows untuk Klien SDK 5 dengan **cluster-id**

- Gunakan ID cluster `cluster-1234567` untuk menentukan alamat IP HSM di cluster Anda.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --cluster-id <cluster-1234567>
```

Anda dapat menggunakan parameter `--region` dan `--endpoint` dalam kombinasi dengan parameter `cluster-id` untuk menentukan bagaimana sistem membuat panggilan `DescribeClusters`. Misalnya, jika wilayah klaster berbeda dari yang dikonfigurasi sebagai default AWS CLI Anda, Anda harus menggunakan parameter `--region` untuk menggunakan wilayah tersebut. Selain itu, Anda memiliki kemampuan untuk menentukan titik akhir AWS CloudHSM API yang akan digunakan untuk panggilan, yang mungkin diperlukan untuk berbagai pengaturan jaringan,

seperti menggunakan titik akhir antarmuka VPC yang tidak menggunakan nama host DNS default untuk. AWS CloudHSM

PKCS #11 library

Untuk bootstrap instans EC2 Linux dengan titik akhir kustom dan wilayah

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda dengan wilayah kustom dan titik akhir.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --cluster-id <cluster-1234567> --  
region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

Untuk bootstrap instans EC2 Windows dengan titik akhir dan wilayah

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda dengan wilayah kustom dan titik akhir.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --cluster-  
id <cluster-1234567>--region <us-east-1> --endpoint <https://cloudhsmv2.us-  
east-1.amazonaws.com>
```

OpenSSL Dynamic Engine

Untuk bootstrap instans EC2 Linux dengan titik akhir kustom dan wilayah

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda dengan wilayah kustom dan titik akhir.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --cluster-id <cluster-1234567> --  
region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

Key Storage Provider (KSP)

Untuk bootstrap instans EC2 Windows dengan titik akhir dan wilayah

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda dengan wilayah kustom dan titik akhir.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --cluster-id <cluster-1234567> --region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

JCE provider

Untuk bootstrap instans EC2 Linux dengan titik akhir kustom dan wilayah

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda dengan wilayah kustom dan titik akhir.

```
$ sudo /opt/cloudhsm/bin/configure-jce --cluster-id <cluster-1234567> --region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

Untuk bootstrap instans EC2 Windows dengan titik akhir dan wilayah

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda dengan wilayah kustom dan titik akhir.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --cluster-id <cluster-1234567> --region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

CloudHSM CLI

Untuk bootstrap instans EC2 Linux dengan titik akhir kustom dan wilayah

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda dengan wilayah kustom dan titik akhir.

```
$ sudo /opt/cloudhsm/bin/configure-cli --cluster-id <cluster-1234567> --region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

Untuk bootstrap instans EC2 Windows dengan titik akhir dan wilayah

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM di cluster Anda dengan wilayah kustom dan titik akhir.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --cluster-id <cluster-1234567> --region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

Untuk informasi tentang parameter `--cluster-id`, `--region` dan `--endpoint`, lihat [the section called "Parameter"](#).

Perbarui sertifikat klien dan kunci untuk otentikasi timbal balik klien TLS-HSM

Example

Contoh ini menunjukkan cara menggunakan `--client-cert-hsm-tls-file` dan `--client-key-hsm-tls-file` parameter untuk mengkonfigurasi ulang SSL dengan menentukan kunci kustom dan sertifikat SSL untuk AWS CloudHSM

PKCS #11 library

Untuk menggunakan sertifikat dan kunci khusus untuk otentikasi timbal balik klien TLS-HSM dengan Client SDK 5 di Linux

- Salin kunci dan sertifikat Anda ke direktori yang sesuai.

```
$ sudo cp ssl-client.pem </opt/cloudhsm/etc>
$ sudo cp ssl-client.key </opt/cloudhsm/etc>
```

- Gunakan alat konfigurasi untuk menentukan `ssl-client.pem` dan `ssl-client.key`.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 \
    --client-cert-hsm-tls-file </opt/cloudhsm/etc/ssl-client.pem> \
    --client-key-hsm-tls-file </opt/cloudhsm/etc/ssl-client.key>
```

Untuk menggunakan sertifikat kustom dan kunci untuk otentikasi timbal balik klien TLS-HSM dengan Klien SDK 5 pada Windows

- Salin kunci dan sertifikat Anda ke direktori yang sesuai.

```
cp ssl-client.pem <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem>
cp ssl-client.key <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

2. Dengan PowerShell penerjemah, gunakan alat konfigurasi untuk menentukan `ssl-client.pem` dan `ssl-client.key`.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" `
    --client-cert-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.pem> `
    --client-key-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.key>
```

OpenSSL Dynamic Engine

Untuk menggunakan sertifikat dan kunci khusus untuk otentikasi timbal balik klien TLS-HSM dengan Client SDK 5 di Linux

1. Salin kunci dan sertifikat Anda ke direktori yang sesuai.

```
$ sudo cp ssl-client.pem </opt/cloudhsm/etc>
sudo cp ssl-client.key </opt/cloudhsm/etc>
```

2. Gunakan alat konfigurasi untuk menentukan `ssl-client.pem` dan `ssl-client.key`.

```
$ sudo /opt/cloudhsm/bin/configure-dyn \
    --client-cert-hsm-tls-file </opt/cloudhsm/etc/ssl-client.pem> \
    --client-key-hsm-tls-file </opt/cloudhsm/etc/ssl-client.key>
```

Key Storage Provider (KSP)

Untuk menggunakan sertifikat kustom dan kunci untuk otentikasi timbal balik klien TLS-HSM dengan Klien SDK 5 pada Windows

1. Salin kunci dan sertifikat Anda ke direktori yang sesuai.

```
cp ssl-client.pem <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem>
cp ssl-client.key <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

2. Dengan PowerShell penerjemah, gunakan alat konfigurasi untuk menentukan `ssl-client.pem` dan `ssl-client.key`.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" `
    --client-cert-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.pem> `
    --client-key-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.key>
```

JCE provider

Untuk menggunakan sertifikat dan kunci khusus untuk otentikasi timbal balik klien TLS-HSM dengan Client SDK 5 di Linux

1. Salin kunci dan sertifikat Anda ke direktori yang sesuai.

```
$ sudo cp ssl-client.pem </opt/cloudhsm/etc>
sudo cp ssl-client.key </opt/cloudhsm/etc>
```

2. Gunakan alat konfigurasi untuk menentukan `ssl-client.pem` dan `ssl-client.key`.

```
$ sudo /opt/cloudhsm/bin/configure-jce \
    --client-cert-hsm-tls-file </opt/cloudhsm/etc/ssl-client.pem> \
    --client-key-hsm-tls-file </opt/cloudhsm/etc/ssl-client.key>
```

Untuk menggunakan sertifikat kustom dan kunci untuk otentikasi timbal balik klien TLS-HSM dengan Klien SDK 5 pada Windows

1. Salin kunci dan sertifikat Anda ke direktori yang sesuai.

```
cp ssl-client.pem <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem>
cp ssl-client.key <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

2. Dengan PowerShell penerjemah, gunakan alat konfigurasi untuk menentukan `ssl-client.pem` dan `ssl-client.key`.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" `
    --client-cert-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-
client.pem> `
```

```
--client-key-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-  
client.key>
```

CloudHSM CLI

Untuk menggunakan sertifikat dan kunci khusus untuk otentikasi timbal balik klien TLS-HSM dengan Client SDK 5 di Linux

1. Salin kunci dan sertifikat Anda ke direktori yang sesuai.

```
$ sudo cp ssl-client.pem </opt/cloudhsm/etc>  
sudo cp ssl-client.key </opt/cloudhsm/etc>
```

2. Gunakan alat konfigurasi untuk menentukan `ssl-client.pem` dan `ssl-client.key`.

```
$ sudo /opt/cloudhsm/bin/configure-cli \  
--client-cert-hsm-tls-file </opt/cloudhsm/etc/ssl-client.pem> \  
--client-key-hsm-tls-file </opt/cloudhsm/etc/ssl-client.key>
```

Untuk menggunakan sertifikat kustom dan kunci untuk otentikasi timbal balik klien TLS-HSM dengan Klien SDK 5 pada Windows

1. Salin kunci dan sertifikat Anda ke direktori yang sesuai.

```
cp ssl-client.pem <C:\ProgramData\Amazon\CloudHSM\ssl-client.pem>  
cp ssl-client.key <C:\ProgramData\Amazon\CloudHSM\ssl-client.key>
```

2. Dengan PowerShell penerjemah, gunakan alat konfigurasi untuk menentukan `ssl-client.pem` dan `ssl-client.key`.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" \  
--client-cert-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-  
client.pem> \  
--client-key-hsm-tls-file <C:\ProgramData\Amazon\CloudHSM\ssl-  
client.key>
```

Untuk informasi tentang parameter `--client-cert-hsm-tls-file` dan `--client-key-hsm-tls-file`, lihat [the section called "Parameter"](#).

Nonaktifkan pengaturan daya tahan kunci klien

Example

Contoh ini menggunakan parameter `--disable-key-availability-check` untuk menonaktifkan pengaturan daya tahan kunci klien. Untuk menjalankan sebuah klaster dengan satu HSM, Anda harus menonaktifkan pengaturan daya tahan kunci klien.

PKCS #11 library

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Linux

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --disable-key-availability-check
```

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Windows

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --disable-key-availability-check
```

OpenSSL Dynamic Engine

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Linux

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --disable-key-availability-check
```

OpenSSL Dynamic Engine Provider

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Linux

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --disable-key-availability-check
```

Key Storage Provider (KSP)

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Windows

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --disable-key-availability-check
```

JCE provider

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Linux

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
$ sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check
```

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Windows

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-key-availability-check
```

CloudHSM CLI

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Linux

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
$ sudo /opt/cloudhsm/bin/configure-cli --disable-key-availability-check
```

Untuk menonaktifkan daya tahan kunci klien untuk Klien SDK 5 di Windows

- Gunakan alat konfigurasi untuk menonaktifkan pengaturan daya tahan kunci klien.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --disable-key-availability-check
```

Untuk informasi tentang parameter `--disable-key-availability-check`, lihat [the section called "Parameter"](#).

Mengelola opsi logging

Example

Client SDK 5 menggunakan log-type parameter `log-file`, `log-level`, `log-rotation`, dan untuk mengelola logging.

 Note

Untuk mengonfigurasi SDK Anda untuk lingkungan tanpa server seperti AWS Fargate atau AWS Lambda, kami sarankan Anda mengonfigurasi jenis log Anda. AWS CloudHSM term Log klien akan dikeluarkan `stderr` dan ditangkap dalam grup CloudWatch log Log yang dikonfigurasi untuk lingkungan itu.

PKCS #11 library

Lokasi pencatatan default

- Jika Anda tidak menentukan lokasi untuk file, sistem menulis log ke lokasi default berikut:

Linux

```
/opt/cloudhsm/run/cloudhsm-pkcs11.log
```

Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-pkcs11.log
```

Untuk mengonfigurasi level logging dan membiarkan opsi logging lainnya disetel ke default

- ```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-level info
```

Untuk mengonfigurasi opsi pencatatan file

- ```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-type file --log-file <file name with path> --log-rotation daily --log-level info
```

Untuk mengonfigurasi opsi pencatatan terminal

- ```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-type term --log-level info
```

## OpenSSL Dynamic Engine

Lokasi pencatatan default

- Jika Anda tidak menentukan lokasi untuk file, sistem menulis log ke lokasi default berikut:

Linux

```
stderr
```

Untuk mengonfigurasi level logging dan membiarkan opsi logging lainnya disetel ke default

- ```
$ sudo /opt/cloudhsm/bin/configure-dyn --log-level info
```

Untuk mengonfigurasi opsi pencatatan file

- ```
$ sudo /opt/cloudhsm/bin/configure-dyn --log-type file --log-file <file name with path> --log-rotation daily --log-level info
```

Untuk mengonfigurasi opsi pencatatan terminal

- ```
$ sudo /opt/cloudhsm/bin/configure-dyn --log-type term --log-level info
```

OpenSSL Dynamic Engine Provider

Lokasi pencatatan default

- Jika Anda tidak menentukan lokasi untuk file, sistem menulis log ke lokasi default berikut:

Linux

```
stderr
```

Untuk mengonfigurasi level logging dan membiarkan opsi logging lainnya disetel ke default

- ```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --log-level info
```

Untuk mengonfigurasi opsi pencatatan file

- ```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --log-type file --log-file <file name with path> --log-rotation daily --log-level info
```

Untuk mengonfigurasi opsi pencatatan terminal

- ```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --log-type term --log-level info
```

## Key Storage Provider (KSP)

Lokasi pencatatan default

- Jika Anda tidak menentukan lokasi untuk file, sistem menulis log ke lokasi default berikut:

Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-ksp.log
```

Untuk mengonfigurasi level logging dan membiarkan opsi logging lainnya disetel ke default

- ```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --log-level info
```

Untuk mengonfigurasi opsi pencatatan file

- ```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --log-type file --log-file <file name with path> --log-rotation daily --log-level info
```

Untuk mengonfigurasi opsi pencatatan terminal

- ```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --log-type term --log-level info
```

JCE provider

Lokasi pencatatan default

- Jika Anda tidak menentukan lokasi untuk file, sistem menulis log ke lokasi default berikut:

Linux

```
/opt/cloudhsm/run/cloudhsm-jce.log
```

Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-jce.log
```

Untuk mengonfigurasi level logging dan membiarkan opsi logging lainnya disetel ke default

- ```
$ sudo /opt/cloudhsm/bin/configure-jce --log-level info
```

Untuk mengonfigurasi opsi pencatatan file

- ```
$ sudo /opt/cloudhsm/bin/configure-jce --log-type file --log-file <file name with path> --log-rotation daily --log-level info
```

Untuk mengonfigurasi opsi pencatatan terminal

- ```
$ sudo /opt/cloudhsm/bin/configure-jce --log-type term --log-level info
```

## CloudHSM CLI

Lokasi pencatatan default

- Jika Anda tidak menentukan lokasi untuk file, sistem menulis log ke lokasi default berikut:

Linux

```
/opt/cloudhsm/run/cloudhsm-cli.log
```

Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-cli.log
```

Untuk mengonfigurasi level logging dan membiarkan opsi logging lainnya disetel ke default

- ```
$ sudo /opt/cloudhsm/bin/configure-cli --log-level info
```

Untuk mengonfigurasi opsi pencatatan file

- ```
$ sudo /opt/cloudhsm/bin/configure-cli --log-type file --log-file <file name with path> --log-rotation daily --log-level info
```

Untuk mengonfigurasi opsi pencatatan terminal

- ```
$ sudo /opt/cloudhsm/bin/configure-cli --log-type term --log-level info
```

Untuk informasi lebih lanjut tentang `log-file`, `log-level`, `log-rotation`, dan `log-type` parameter, lihat [the section called “Parameter”](#).

Tempatkan sertifikat penerbitan untuk Client SDK 5

Example

Contoh ini menggunakan parameter `--hsm-ca-cert` untuk memperbarui lokasi penerbitan sertifikat untuk Klien SDK 5.

PKCS #11 library

Tempat penerbitan sertifikat pada Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --hsm-ca-cert <customerCA certificate file>
```

Tempat penerbitan sertifikat pada Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --hsm-ca-cert <customerCA certificate file>
```


OpenSSL Dynamic Engine

Tempat penerbitan sertifikat pada Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --hsm-ca-cert <customerCA certificate file>
```

OpenSSL Dynamic Engine Provider

Tempat penerbitan sertifikat pada Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --hsm-ca-cert <customerCA certificate file>
```

Key Storage Provider (KSP)

Tempat penerbitan sertifikat pada Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --hsm-ca-cert <customerCA certificate file>
```

JCE provider

Tempat penerbitan sertifikat pada Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
$ sudo /opt/cloudhsm/bin/configure-jce --hsm-ca-cert <customerCA certificate file>
```

Tempat penerbitan sertifikat pada Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --hsm-ca-cert <customerCA certificate file>
```

CloudHSM CLI

Tempat penerbitan sertifikat pada Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
$ sudo /opt/cloudhsm/bin/configure-cli --hsm-ca-cert <customerCA certificate file>
```

Tempat penerbitan sertifikat pada Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan lokasi untuk menerbitkan sertifikat.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --hsm-ca-cert <customerCA certificate file>
```

Untuk informasi tentang parameter `--hsm-ca-cert`, lihat [the section called "Parameter"](#).

Penyedia OpenSSL Bootstrap

Gunakan `configure-openssl-provider` alat ini untuk mem-bootstrap instalasi Penyedia OpenSSL Anda dan menghubungkannya ke cluster Anda. AWS CloudHSM

Untuk mem-bootstrap Penyedia OpenSSL

- Jalankan `configure-openssl-provider` perintah dengan alamat IP HSM di cluster Anda:

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider -a <HSM IP address>
```

Ganti `<HSM IP address>` dengan alamat IP HSM apa pun di cluster Anda.

2. Verifikasi konfigurasi dengan memeriksa apakah Penyedia OpenSSL dapat terhubung ke cluster Anda:

```
$ openssl list -providers -provider-path /opt/cloudhsm/lib -provider cloudhsm
```

Untuk informasi selengkapnya tentang parameter konfigurasi, lihat [the section called “Parameter”](#).

Konfigurasi lanjutan untuk alat konfigurasi Client SDK 5

Alat konfigurasi AWS CloudHSM Client SDK 5 mencakup konfigurasi lanjutan yang bukan merupakan bagian dari fitur umum yang digunakan sebagian besar pelanggan. Konfigurasi lanjutan memberikan kemampuan tambahan.

Important

Setelah membuat perubahan pada konfigurasi Anda, Anda perlu me-restart aplikasi Anda agar perubahan diterapkan.

- Konfigurasi lanjutan untuk PKCS #11
 - [Beberapa konfigurasi slot dengan pustaka PKCS #11 untuk AWS CloudHSM](#)
 - [Coba lagi perintah untuk pustaka PKCS #11 untuk AWS CloudHSM](#)
- Konfigurasi lanjutan untuk OpenSSL
 - [Coba lagi perintah untuk OpenSSL untuk AWS CloudHSM](#)
- Konfigurasi lanjutan untuk KSP
 - [SDK3 mode kompatibilitas untuk Penyedia Penyimpanan Kunci \(KSP\) untuk AWS CloudHSM](#)
- Konfigurasi lanjutan untuk JCE
 - [Menghubungkan ke beberapa AWS CloudHSM cluster dengan penyedia JCE](#)
 - [Coba lagi perintah untuk JCE untuk AWS CloudHSM](#)
 - [Ekstraksi kunci menggunakan JCE untuk AWS CloudHSM](#)
- Konfigurasi lanjutan untuk Antarmuka Baris AWS CloudHSM Perintah (CLI)
 - [Menghubungkan ke beberapa cluster dengan CloudHSM CLI](#)

AWS CloudHSM Klien SDK 5 topik terkait

Lihat topik terkait berikut untuk mempelajari lebih lanjut tentang SDK AWS CloudHSM Klien 5.

- Operasi API [DescribeClusters](#)
- [describe-clusters](#) AWS CLI
- [Get-HSM2Cluster](#) PowerShell cmdlet
- [Bootstrap Klien SDK 5](#)
- [Penyedia OpenSSL Bootstrap](#)
- [AWS CloudHSM Titik akhir VPC](#)
- [Mengelola Pengaturan Daya Tahan Kunci SDK 5 Klien](#)
- [Klien SDK 5 Pencatatan](#)
- [Pengaturan mTL \(disarankan\)](#)

AWS CloudHSM Alat konfigurasi SDK 3 klien

Gunakan alat konfigurasi AWS CloudHSM Client SDK 3 untuk mem-bootstrap daemon klien dan mengkonfigurasi CloudHSM Management Utility (CMU).

Topik

- [AWS CloudHSM Sintaks konfigurasi SDK 3 klien](#)
- [AWS CloudHSM Parameter konfigurasi SDK 3 klien](#)
- [AWS CloudHSM Contoh konfigurasi SDK 3 klien](#)
- [AWS CloudHSM Topik terkait konfigurasi SDK 3 klien](#)

AWS CloudHSM Sintaks konfigurasi SDK 3 klien

Tabel berikut menggambarkan sintaks untuk file AWS CloudHSM konfigurasi untuk Client SDK 3.

```
configure -h | --help
-a <ENI IP address>
-m [-i <daemon_id>]
--ssl --pkey <private key file> --cert <certificate file>
--cmu <ENI IP address>
```

AWS CloudHSM Parameter konfigurasi SDK 3 klien

Berikut ini adalah daftar parameter untuk mengkonfigurasi SDK AWS CloudHSM Klien 3.

-h | --help

Menampilkan sintaks perintah.

Wajib: Ya

-a <ENI IP address>

Menambahkan alamat IP antarmuka jaringan elastis (ENI) HSM tertentu untuk file konfigurasi AWS CloudHSM. Masukkan alamat IP ENI dari salah satu HSMs di cluster. Tidak masalah yang mana yang Anda pilih.

Untuk mendapatkan alamat IP ENI dari cluster Anda, gunakan [DescribeClusters](#) operasi, [AWS CLI perintah deskripsi-cluster](#), atau [cmdlet](#). HSMs [Get-HSM2Cluster](#) PowerShell

Note

Sebelum menjalankan `-a configure` perintah, hentikan AWS CloudHSM klien. Kemudian, ketika `-a` perintah selesai, restart AWS CloudHSM klien. Untuk detailnya, [lihat contoh](#).

Parameter ini mengedit file konfigurasi berikut:

- `/opt/cloudhsm/etc/cloudhsm_client.cfg`: Digunakan oleh AWS CloudHSM klien dan [key_mgmt_util](#).
- `/opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg`: Digunakan oleh [cloudhsm_mgmt_util](#).

Ketika AWS CloudHSM klien mulai, ia menggunakan alamat IP ENI dalam file konfigurasi untuk query cluster dan memperbarui `cluster.info` file (`/opt/cloudhsm/daemon/1/cluster.info`) dengan alamat IP ENI yang benar untuk semua HSMs di cluster.

Wajib: Ya

-m

Memperbarui alamat IP HSM ENI dalam file konfigurasi yang menggunakan CMU.

 Note

Parameter `-m` ini untuk digunakan dengan CMU dari Klien SDK 3.2.1 dan sebelumnya. Untuk CMU dari Klien SDK 3.3.0 dan yang lebih baru, lihat parameter `--cmu`, yang menyederhanakan proses pembaruan data HSM untuk CMU.

Ketika Anda memperbarui `-a` parameter `configure` dan kemudian memulai AWS CloudHSM klien, daemon klien menanyakan cluster dan memperbarui `cluster.info` file dengan alamat IP HSM yang benar untuk semua HSMs di cluster. Menjalankan perintah `-m configure` melengkapi pembaruan dengan menyalin alamat IP HSM dari `cluster.info` ke file konfigurasi `cloudhsm_mgmt_util.cfg` yang digunakan `cloudhsm_mgmt_util`.

Pastikan untuk menjalankan `-a configure` perintah dan restart AWS CloudHSM klien sebelum menjalankan `-m` perintah. Hal ini memastikan bahwa data yang disalin ke file `cloudhsm_mgmt_util.cfg` dari file `cluster.info` lengkap dan akurat.

Wajib: Ya

`-i`

Menentukan daemon klien alternatif. Nilai default mewakili klien AWS CloudHSM .

Default: 1

Wajib: Tidak

`--ssl`

Menggantikan kunci SSL dan sertifikat untuk klaster dengan kunci privat yang ditentukan dan sertifikat. Bila Anda menggunakan parameter ini, parameter `--pkey` dan `--cert` diperlukan.

Wajib: Tidak

`--pkey`

Menentukan kunci privat baru. Masukkan jalur dan nama file dari file yang berisi kunci privat.

Diperlukan: Ya jika `--ssl` ditentukan. Jika tidak, ini tidak boleh digunakan.

--cert

Menentukan sertifikat baru. Masukkan jalur dan nama file dari file yang berisi sertifikat. Sertifikat harus rantai mengikat sertifikat `customerCA.crt`, sertifikat yang ditandatangani sendiri digunakan untuk menginisialisasi klaster. Untuk informasi lebih lanjut, lihat: [Inisialisasi Klaster](#).

Diperlukan: Ya jika --ssl ditentukan. Jika tidak, ini tidak boleh digunakan.

--cmu **<ENI IP address>**

Menggabungkan parameter -a dan -m menjadi satu parameter. Menambahkan alamat IP HSM elastic network interface (ENI) yang ditentukan ke file AWS CloudHSM konfigurasi dan kemudian memperbarui file konfigurasi CMU. Masukkan alamat IP dari setiap HSM di klaster. Untuk Klien SDK 3.2.1 dan sebelumnya, lihat [Menggunakan CMU dengan Klien SDK 3.2.1 dan Sebelumnya](#).

Wajib: Ya

AWS CloudHSM Contoh konfigurasi SDK 3 klien

Contoh-contoh ini menunjukkan cara menggunakan configure alat untuk AWS CloudHSM Client SDK 3.

Example: Perbarui data HSM untuk AWS CloudHSM klien dan key_mgmt_util

Contoh ini menggunakan -a parameter configure untuk memperbarui data HSM untuk AWS CloudHSM klien dan key_mgmt_util. Untuk menggunakan -a parameter, Anda harus memiliki alamat IP untuk salah satu HSMs di cluster Anda. Gunakan konsol atau AWS CLI untuk mendapatkan alamat IP.

Untuk mendapatkan alamat IP untuk HSM (konsol)

1. Buka AWS CloudHSM konsol di <https://console.aws.amazon.com/cloudhsm/rumah>.
2. Untuk mengubah Wilayah AWS, gunakan pemilih Wilayah di sudut kanan atas halaman.
3. Untuk membuka halaman detail klaster, dalam tabel klaster, pilih ID klaster.
4. Untuk mendapatkan alamat IP, buka HSMs tab. Untuk IPv4 cluster, pilih alamat yang tercantum di bawah IPv4 alamat ENI. Untuk cluster dual-stack gunakan ENI IPv4 atau alamat ENI. IPv6

Untuk mendapatkan alamat IP untuk HSM (AWS CLI)

- Dapatkan alamat IP dari HSM dengan menggunakan perintah [describe-clusters](#) dari AWS CLI. Dalam output dari perintah, alamat IP dari HSMs adalah nilai-nilai `EniIp` dan `EniIPv6` (jika itu adalah cluster dual-stack).

```
$ aws cloudhsmv2 describe-clusters
{
  "Clusters": [
    { ... }
    "Hsms": [
      {
...
        "EniIp": "10.0.0.9",
...
      },
      {
...
        "EniIp": "10.0.1.6",
        "EniIPv6": "2600:113f:404:be09:310e:ed34:3412:f733",
...
      }
    ]
  }
}
```

Untuk memperbarui data HSM

- Sebelum memperbarui `-a` parameter, hentikan AWS CloudHSM klien. Hal ini mencegah konflik yang mungkin terjadi saat configure mengedit file konfigurasi klien. Jika klien sudah berhenti, perintah ini tidak berpengaruh, sehingga Anda dapat menggunakannya dalam skrip.

Amazon Linux

```
$ sudo stop cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client stop
```

CentOS 7

```
$ sudo service cloudhsm-client stop
```


CentOS 8

```
$ sudo service cloudhsm-client stop
```

RHEL 7

```
$ sudo service cloudhsm-client stop
```

RHEL 8

```
$ sudo service cloudhsm-client stop
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client stop
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client stop
```

Windows

- Untuk klien Windows 1.1.2+:

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- Untuk klien Windows 1.1.1 dan yang lebih lama:

Gunakan Ctrl+C di jendela perintah tempat Anda memulai klien. AWS CloudHSM

2. Langkah ini menggunakan parameter `-a` dari `configure` untuk menambahkan alamat IP ENI `10.0.0.9` ke file konfigurasi.

Amazon Linux

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

Amazon Linux 2

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

CentOS 7

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

CentOS 8

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

RHEL 7

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

RHEL 8

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

Ubuntu 16.04 LTS

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

Ubuntu 18.04 LTS

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" -a 10.0.0.9
```

3. Selanjutnya, restart AWS CloudHSM klien. Ketika dimulai, klien menggunakan alamat IP ENI dalam file konfigurasi untuk membuat kueri klaster. Kemudian, ia menulis alamat IP ENI dari semua HSMs di cluster ke `cluster.info` file.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Untuk klien Windows 1.1.2+:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Untuk klien Windows 1.1.1 dan yang lebih lama:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe  
C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Ketika perintah selesai, data HSM yang digunakan AWS CloudHSM klien dan `key_mgmt_util` lengkap dan akurat.

Example: Perbarui Data HSM untuk CMU dari klien SDK 3.2.1 dan sebelumnya

Contoh ini menggunakan perintah `-m configure` untuk menyalin data HSM diperbarui dari file `cluster.info` ke file `cloudhsm_mgmt_util.cfg` yang digunakan `cloudhsm_mgmt_util`. Gunakan ini dengan CMU yang dikirimkan dengan klien SDK 3.2.1 dan sebelumnya.

- Sebelum menjalankan `-m`, hentikan AWS CloudHSM klien, jalankan `-a` perintah, dan kemudian restart AWS CloudHSM klien, seperti yang ditunjukkan pada [contoh sebelumnya](#). Hal ini memastikan bahwa data disalin ke file `cloudhsm_mgmt_util.cfg` dari file `cluster.info` lengkap dan akurat.

Linux

```
$ sudo /opt/cloudhsm/bin/configure -m
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" -m
```

Example: Perbarui Data HSM untuk CMU dari klien SDK 3.3.0 dan yang lebih baru

Contoh ini menggunakan parameter `--cmu` dari perintah `configure` untuk memperbarui data HSM untuk CMU. Gunakan ini dengan CMU yang dikirimkan dengan klien SDK 3.3.0 dan yang lebih baru. Untuk informasi lebih lanjut tentang penggunaan CMU, lihat [Menggunakan Utilitas Manajemen CloudHSM \(CMU\) untuk Mengelola Pengguna](#) dan [Menggunakan CMU dengan Klien SDK 3.2.1 dan Sebelumnya](#).

- Penggunaan parameter `--cmu` untuk meneruskan alamat IP dari HSM di klaster Anda.

Linux

```
$ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" --cmu <IP address>
```

AWS CloudHSM Topik terkait konfigurasi SDK 3 klien

Lihat topik terkait berikut untuk mempelajari lebih lanjut tentang SDK AWS CloudHSM Klien 3.

- [Mengatur AWS CloudHSM key_mgmt_util](#)

AWS CloudHSM Antarmuka Baris Perintah (CLI)

CloudHSM CLI membantu admin mengelola pengguna dan pengguna krypto mengelola kunci di cluster mereka. AWS CloudHSM CLI mencakup alat yang dapat digunakan untuk membuat, menghapus dan mencantumkan pengguna, mengubah kata sandi pengguna, memperbarui otentikasi multi-faktor pengguna (MFA). Ini juga mencakup perintah yang menghasilkan, menghapus, mengimpor, dan mengeksport kunci, mendapatkan dan mengatur atribut, menemukan kunci, dan melakukan operasi kriptografi.

Untuk daftar pengguna CloudHSM CLI yang ditentukan, lihat. [Manajemen pengguna HSM dengan CloudHSM CLI](#) Untuk daftar atribut kunci yang ditentukan untuk CloudHSM CLI, lihat. [Atribut kunci untuk CloudHSM CLI](#) Untuk informasi tentang cara menggunakan CloudHSM CLI untuk mengelola kunci, lihat. [Manajemen kunci dengan CloudHSM CLI](#)

Untuk memulai dengan cepat, lihat [Memulai dengan AWS CloudHSM Command Line Interface \(CLI\)](#). Untuk informasi rinci tentang perintah CloudHSM CLI dan contoh penggunaan perintah, lihat. [Referensi untuk perintah CloudHSM CLI](#)

Topik

- [AWS CloudHSM Platform yang didukung Antarmuka Baris Perintah \(CLI\)](#)
- [Memulai dengan AWS CloudHSM Command Line Interface \(CLI\)](#)
- [Mode perintah di CloudHSM CLI](#)

- [Atribut kunci untuk CloudHSM CLI](#)
- [Konfigurasi lanjutan untuk CloudHSM CLI](#)
- [Referensi untuk perintah CloudHSM CLI](#)

AWS CloudHSM Platform yang didukung Antarmuka Baris Perintah (CLI)

Topik ini menjelaskan platform Linux dan Windows yang didukung AWS CloudHSM CLI.

Dukungan Linux

Platform yang didukung	X86_64 Arsitektur	Arsitektur ARM
Amazon Linux 2	Ya	Ya
Amazon Linux 2023	Ya	Ya
Perusahaan Topi Merah Linux 8 (8.3+)	Ya	Ya
Perusahaan Topi Merah Linux 9 (9.2+)	Ya	Ya
Perusahaan Topi Merah Linux 10 (10.0+)	Ya	Ya
Ubuntu 22.04 LTS	Ya	Ya
Ubuntu 24.04 LTS	Ya	Ya

- SDK 5.16 adalah rilis terakhir yang menyediakan dukungan platform Ubuntu 20.04 LTS. Untuk informasi lebih lanjut, lihat situs [web Ubuntu](#).
- SDK 5.12 adalah rilis terakhir yang memberikan dukungan platform CentOS 7 (7.8+). Untuk informasi lebih lanjut, lihat situs web [CentOS](#).
- SDK 5.12 adalah rilis terakhir yang memberikan dukungan platform Red Hat Enterprise Linux 7 (7.8+). Untuk informasi lebih lanjut, lihat situs [web Red Hat](#).
- SDK 5.4.2 adalah rilis terakhir yang memberikan dukungan platform CentOS 8. Untuk informasi lebih lanjut, lihat situs web [CentOS](#).

Dukungan Windows

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019
- Server Microsoft Windows 2022
- Microsoft Windows Server 2025

Memulai dengan AWS CloudHSM Command Line Interface (CLI)

Dengan CloudHSM CLI Command Line Interface (CLI), Anda dapat mengelola pengguna di cluster Anda. AWS CloudHSM Gunakan topik ini untuk memulai tugas manajemen pengguna modul keamanan perangkat keras (HSM) dasar, seperti membuat pengguna, mencantumkan pengguna, dan menghubungkan CloudHSM CLI ke cluster.

Topik

- [Instal CloudHSM CLI](#)
- [Gunakan CloudHSM CLI](#)

Instal CloudHSM CLI

Gunakan perintah berikut untuk mengunduh dan menginstal CloudHSM CLI untuk AWS CloudHSM

Amazon Linux 2023

Amazon Linux 2023 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-cli-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.amzn2023.x86_64.rpm
```

Amazon Linux 2023 tentang ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-cli-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.amzn2023.aarch64.rpm
```

Amazon Linux 2

Amazon Linux 2 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.x86_64.rpm
```

Amazon Linux 2 tentang ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.aarch64.rpm
```

RHEL 10 (10.0+)

RHEL 10 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/cloudhsm-cli-latest.el10.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el10.x86_64.rpm
```

RHEL 10 tentang ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/cloudhsm-cli-latest.el10.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el10.aarch64.rpm
```

RHEL 9 (9.2+)

RHEL 9 pada arsitektur x86_64:


```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-cli-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el9.x86_64.rpm
```

RHEL 9 tentang ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-cli-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el9.aarch64.rpm
```

RHEL 8 (8.3+)

RHEL 8 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-cli-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el8.x86_64.rpm
```

RHEL 8 tentang ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-cli-latest.el8.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el8.aarch64.rpm
```

Ubuntu 24.04 LTS

Ubuntu 24.04 LTS pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-cli_latest_u24.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u24.04_amd64.deb
```

Ubuntu 24.04 LTS pada ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-  
cli_latest_u24.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u24.04_arm64.deb
```

Ubuntu 22.04 LTS

Ubuntu 22.04 LTS pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-  
cli_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u22.04_amd64.deb
```

Ubuntu 22.04 LTS pada ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-  
cli_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u22.04_arm64.deb
```

Windows Server 2022

Untuk Windows Server 2022 pada arsitektur x86_64, buka PowerShell sebagai administrator dan jalankan perintah berikut:

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/  
AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msixexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /  
quiet /norestart /log C:\client-install.txt' -Wait
```

Windows Server 2019

Untuk Windows Server 2019 pada arsitektur x86_64, buka PowerShell sebagai administrator dan jalankan perintah berikut:

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msixec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /quiet /norestart /log C:\client-install.txt' -Wait
```

Windows Server 2016

Untuk Windows Server 2016 pada arsitektur x86_64, buka PowerShell sebagai administrator dan jalankan perintah berikut:

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msixec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /quiet /norestart /log C:\client-install.txt' -Wait
```

Gunakan perintah berikut untuk mengkonfigurasi CloudHSM CLI.

Untuk melakukan bootstrap instans EC2 Linux untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM (s) di cluster Anda.

```
$ sudo /opt/cloudhsm/bin/configure-cli -a <The ENI IPv4 / IPv6 addresses of the HSMs>
```

Untuk melakukan bootstrap instans EC2 Windows untuk Klien SDK 5

- Gunakan alat konfigurasi untuk menentukan alamat IP HSM (s) di cluster Anda.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" -a <The ENI IPv4 / IPv6 addresses of the HSMs>
```

Gunakan CloudHSM CLI

Gunakan perintah berikut untuk memulai dan menggunakan CloudHSM CLI.

1. Gunakan perintah berikut untuk memulai CloudHSM CLI.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

2. Gunakan perintah login untuk masuk ke klaster. Semua pengguna dapat menggunakan perintah ini.

Perintah dalam contoh berikut log di admin, yang merupakan akun [admin](#) default. Anda menyetel kata sandi pengguna ini saat Anda [mengaktifkan klaster](#).

```
aws-cloudhsm > login --username admin --role admin
```

Sistem meminta kata sandi Anda. Anda memasukkan kata sandi, dan output menunjukkan bahwa perintah berhasil.

```
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin",
    "role": "admin"
  }
}
```

3. Jalankan user list perintah untuk mencantumkan semua pengguna di cluster.

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
```

```

        "mfa": [],
        "cluster-coverage": "full"
    },
    {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
    }
]
}
}

```

4. Gunakan `user create` untuk membuat pengguna CU bernama **example_user**.

Anda dapat membuat CUs karena pada langkah sebelumnya Anda masuk sebagai pengguna admin. Hanya pengguna admin yang dapat melakukan tugas manajemen pengguna, seperti membuat dan menghapus pengguna dan mengubah kata sandi pengguna lain.

```

aws-cloudhsm > user create --username example_user --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "example_user",
    "role": "crypto-user"
  }
}

```

5. Gunakan `user list` untuk mencantumkan semua pengguna di kluster.

```

aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",

```

```
    "mfa": [],
    "cluster-coverage": "full"
  },
  {
    "username": "example_user",
    "role": "crypto_user",
    "locked": "false",
    "mfa": [],
    "cluster-coverage": "full"
  },
  {
    "username": "app_user",
    "role": "internal(APPLIANCE_USER)",
    "locked": "false",
    "mfa": [],
    "cluster-coverage": "full"
  }
]
}
```

- Gunakan `logout` perintah untuk keluar dari AWS CloudHSM cluster.

```
aws-cloudhsm > logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

- Gunakan `quit` perintah untuk menghentikan CLI.

```
aws-cloudhsm > quit
```

Mode perintah di CloudHSM CLI

Di CloudHSM CLI, Anda dapat menjalankan perintah dengan dua cara berbeda: dalam mode perintah tunggal dan mode interaktif. Mode interaktif dirancang untuk pengguna, dan mode perintah tunggal dirancang untuk skrip.

 Note

Semua perintah bekerja dalam mode interaktif dan mode perintah tunggal.

Mode interaktif

Gunakan perintah berikut untuk memulai mode interaktif CloudHSM CLI

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive
```

Saat menggunakan CLI dalam Mode Interaktif, Anda dapat masuk ke akun pengguna menggunakan perintah. login

```
aws-cloudhsm > login --username <USERNAME> --role <ROLE>
```

Untuk membuat daftar semua perintah CloudHSM CLI, jalankan perintah berikut:

```
aws-cloudhsm > help
```

Untuk mendapatkan sintaks untuk perintah CloudHSM CLI, jalankan perintah berikut:

```
aws-cloudhsm > help <command-name>
```

Untuk mendapatkan daftar pengguna di HSMs, masukkan user list.

```
aws-cloudhsm > user list
```

Untuk mengakhiri sesi CloudHSM CLI Anda, jalankan perintah berikut:

```
aws-cloudhsm > quit
```

Mode Perintah Tunggal

Note

Saat menggunakan mode perintah tunggal, Anda harus melarikan diri dari karakter khusus apa pun dalam variabel lingkungan dan argumen baris perintah yang dapat ditafsirkan oleh shell Anda.

Jika Anda menjalankan CloudHSM CLI menggunakan Single Command Mode, Anda perlu mengatur dua variabel lingkungan untuk memberikan kredensi: CLOUDHSM_PIN dan CLOUDHSM_ROLE:

```
$ export CLOUDHSM_ROLE=admin
```

```
$ export CLOUDHSM_PIN=admin_username:admin_password
```

Setelah melakukan ini, Anda dapat menjalankan perintah menggunakan kredensial yang disimpan di lingkungan Anda.

```
$ cloudhsm-cli user change-password --username alice --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

Atribut kunci untuk CloudHSM CLI

Topik ini menjelaskan cara menggunakan CloudHSM CLI untuk menyetel atribut kunci. Atribut kunci di CloudHSM CLI dapat menentukan jenis kunci, bagaimana kunci dapat berfungsi, atau bagaimana kunci diberi label. Beberapa atribut mendefinisikan karakteristik unik (tipe kunci, misalnya). Atribut lain dapat disetel ke true atau false—mengubahnya mengaktifkan atau menonaktifkan bagian dari fungsionalitas kunci.

Untuk contoh yang menunjukkan cara menggunakan atribut kunci, lihat perintah yang tercantum di bawah perintah induk [Kategori kunci dalam CloudHSM CLI](#).

Topik berikut memberikan detail tambahan tentang atribut kunci di CloudHSM CLI.

Topik

- [Atribut yang didukung untuk CloudHSM CLI](#)
- [Periksa nilai di CloudHSM CLI](#)
- [Topik terkait untuk CloudHSM CLI](#)

Atribut yang didukung untuk CloudHSM CLI

Sebagai praktik terbaik, tetapkan hanya nilai untuk atribut yang ingin Anda buat ketat. Jika Anda tidak menentukan nilai, CloudHSM CLI menggunakan nilai default yang ditentukan dalam tabel di bawah ini.

Tabel berikut mencantumkan atribut kunci, kemungkinan nilai, default, dan catatan terkait untuk CloudHSM CLI. Sel kosong di kolom Nilai menunjukkan bahwa tidak ada nilai default spesifik yang ditetapkan ke atribut.

Atribut CloudHSM CLI	Nilai	Dapat dimodifikasi dengan atribut set kunci	Dapat diatur pada pembuatan kunci
<code>always-sensitive</code>	Nilainya adalah <code>True</code> if <code>sensitive</code> selalu disetel ke <code>True</code> dan tidak pernah berubah.	Tidak	Tidak
<code>check-value</code>	Nilai cek kunci. Untuk informasi lebih lanjut, lihat Detail tambahan .	Tidak	Tidak
<code>class</code>	Nilai yang mungkin: <code>secret-key</code> , <code>public-key</code> , dan <code>private-key</code> .	Tidak	Ya

Atribut CloudHSM CLI	Nilai	Dapat dimodifikasi dengan atribut set kunci	Dapat diatur pada pembuatan kunci
curve	Kurva eliptik digunakan untuk menghasilkan key pair EC. Nilai yang Valid: secp224r1 secp256r1 ,prime256v1 ,secp384r1 ,secp256k1 ,,secp521r1 , dan ed25519 ed25519 hanya didukung pada instance hsm2m.medium dalam mode non-FIPS.	Tidak	Dapat diatur dengan EC, tidak dapat diatur dengan RSA
decrypt	Default: False	Ya	Ya
derive	Default: False	Derive dapat diatur pada instance hsm2m.medium. Itu tidak dapat diatur untuk kunci RSA pada instance hsm1.medium.	Ya
destroyable	Default: True	Ya	Ya

Atribut CloudHSM CLI	Nilai	Dapat dimodifikasi dengan atribut set kunci	Dapat diatur pada pembuatan kunci
ec-point	Untuk kunci EC, pengkodean DER dari ANSI X9.62 ECPoint nilai "Q" dalam format heksadesimal. Untuk jenis kunci lainnya, atribut ini tidak ada.	Tidak	Tidak
encrypt	Default: False	Ya	Ya
extractable	Default: True	Tidak	Ya
id	Default: Kosong	id dapat diatur pada instance hsm2m.medium. Itu tidak dapat diatur pada instance hsm1.medium.	Ya
key-length-bytes	Diperlukan untuk menghasilkan kunci AES. Nilai yang valid: 16, 24, dan 32 byte.	Tidak	Tidak
key-type	Nilai yang mungkin: aes, rsa, dan ec	Tidak	Ya
label	Default: Kosong	Ya	Ya

Atribut CloudHSM CLI	Nilai	Dapat dimodifikasi dengan atribut set kunci	Dapat diatur pada pembuatan kunci
local	Default: True untuk kunci yang dihasilkan di HSM, False untuk kunci yang diimpor ke HSM.	Tidak	Tidak
modifiable	Default: True	Dapat diubah dari benar menjadi salah, tetapi tidak dari salah menjadi benar.	Ya
modulus	Modulus yang digunakan untuk menghasilkan pasangan kunci RSA Untuk jenis kunci lainnya, atribut ini tidak ada.	Tidak	Tidak
modulus-size-bits	Diperlukan untuk menghasilkan key pair RSA. Nilai minimum adalah 2048.	Tidak	Dapat diatur dengan RSA, tidak dapat diatur dengan EC

Atribut CloudHSM CLI	Nilai	Dapat dimodifikasi dengan atribut set kunci	Dapat diatur pada pembuatan kunci
never-extractable	Nilainya adalah True jika dapat diekstraksi sebelum pernah disetel keFalse. Nilainya adalah False jika dapat diekstraksi pernah disetel keTrue.	Tidak	Tidak
private	Default: True	Tidak	Ya
public-exponent	Diperlukan untuk menghasilkan key pair RSA. Nilai yang valid: Nilai harus berupa angka ganjil yang lebih besar dari atau sama dengan 65537.	Tidak	Dapat diatur dengan RSA, tidak dapat diatur dengan EC
sensitive	Default: - Nilainya True untuk kunci AES dan kunci pribadi EC dan RSA. - Nilainya False untuk kunci publik EC dan RSA.	Tidak	Dapat diatur dengan kunci pribadi, tidak dapat diatur dengan kunci publik.

Atribut CloudHSM CLI	Nilai	Dapat dimodifikasi dengan atribut set kunci	Dapat diatur pada pembuatan kunci
sign	Default: - Nilainya <code>True</code> untuk kunci AES. - Nilainya <code>False</code> untuk kunci RSA dan EC.	Ya	Ya
token	Default: <code>True</code>	Dapat diubah dari salah menjadi benar, tetapi tidak dari benar menjadi salah.	Ya
trusted	Default: <code>False</code>	Hanya pengguna admin yang dapat mengatur parameter ini.	Tidak
unwrap	Default: <code>False</code>	Ya	Ya, kecuali untuk kunci publik.
unwrap-template	Nilai harus menggunakan templat atribut yang diterapkan untuk setiap kunci terbuka menggunakan kunci pembungkus ini.	Ya	Tidak

Atribut CloudHSM CLI	Nilai	Dapat dimodifikasi dengan atribut set kunci	Dapat diatur pada pembuatan kunci
verify	Default: - Nilainya True untuk kunci AES. - Nilainya False untuk kunci RSA dan EC.	Ya	Ya
wrap	Default: False	Ya	Ya, kecuali untuk kunci pribadi.
wrap-template	Nilai harus menggunakan template atribut untuk mencocokkan kunci yang dibungkus menggunakan kunci pembungkus ini.	Ya	Tidak
wrap-with-trusted	Default: False	Ya	Ya

Periksa nilai di CloudHSM CLI

Nilai cek di CloudHSM CLI adalah hash 3-byte atau checksum dari kunci yang dihasilkan ketika HSM mengimpor atau menghasilkan kunci. Anda juga dapat menghitung nilai cek di luar HSM, seperti setelah Anda mengekspor kunci. Anda kemudian dapat membandingkan nilai nilai cek untuk mengkonfirmasi identitas dan integritas kunci. Untuk mendapatkan nilai cek kunci, gunakan [daftar kunci dengan bendera](#) verbose.

AWS CloudHSM menggunakan metode standar berikut untuk menghasilkan nilai cek:

- Kunci simetris: 3 byte pertama hasil enkripsi blok nol dengan kunci.
- Pasangan kunci asimetris: 3 byte pertama dari hash SHA-1 dari kunci publik.

- Kunci HMAC: KCV untuk kunci HMAC tidak didukung saat ini.

Topik terkait untuk CloudHSM CLI

Lihat topik berikut untuk informasi selengkapnya tentang CloudHSM CLI.

- [Kategori kunci dalam CloudHSM CLI](#)
- [Referensi untuk perintah CloudHSM CLI](#)

Konfigurasi lanjutan untuk CloudHSM CLI

AWS CloudHSM Command Line Interface (CLI) mencakup konfigurasi lanjutan berikut, yang bukan merupakan bagian dari konfigurasi umum yang digunakan sebagian besar pelanggan. Konfigurasi ini memberikan kemampuan tambahan.

- [Menghubungkan ke beberapa cluster](#)

Menghubungkan ke beberapa cluster dengan CloudHSM CLI

Dengan AWS CloudHSM Client SDK 5, Anda dapat mengonfigurasi CloudHSM CLI untuk mengizinkan koneksi ke beberapa klaster CloudHSM dari satu instance CLI.

Topik berikut menjelaskan cara menggunakan fungsionalitas multi-cluster CloudHSM CLI untuk terhubung dengan beberapa cluster.

Topik

- [Prasyarat multi-cluster untuk AWS CloudHSM](#)
- [Konfigurasi CloudHSM CLI untuk fungsionalitas multi-cluster](#)
- [Tambahkan klaster ke AWS CloudHSM konfigurasi Anda](#)
- [Hapus klaster dari AWS CloudHSM konfigurasi Anda](#)
- [Berinteraksi dengan beberapa cluster di AWS CloudHSM](#)

Prasyarat multi-cluster untuk AWS CloudHSM

Sebelum mengonfigurasi klaster AWS CloudHSM untuk terhubung ke beberapa cluster, Anda harus memenuhi prasyarat berikut:

- Dua atau lebih AWS CloudHSM cluster yang ingin Anda sambungkan, bersama dengan sertifikat klaster mereka.
- EC2 Instance dengan Grup Keamanan dikonfigurasi dengan benar untuk terhubung ke semua cluster di atas. Untuk informasi selengkapnya tentang cara menyiapkan cluster dan instance klien, lihat [Memulai AWS CloudHSM](#).
- Untuk mengatur fungsionalitas multi-cluster, Anda harus sudah mengunduh dan menginstal CloudHSM CLI. Jika Anda belum melakukan ini, lihat instruksi di [???](#).
- Anda tidak akan dapat mengakses cluster yang dikonfigurasi `./configure-cli[.exe] -a` karena tidak akan dikaitkan dengan `filecluster-id`. Anda dapat mengkonfigurasi ulang dengan mengikuti `config-cli add-cluster` seperti yang dijelaskan dalam panduan ini.

Konfigurasi CloudHSM CLI untuk fungsionalitas multi-cluster

Untuk mengonfigurasi CloudHSM CLI Anda untuk fungsionalitas multi-cluster, ikuti langkah-langkah berikut:

1. Identifikasi cluster yang ingin Anda sambungkan.
2. [Tambahkan cluster ini ke konfigurasi CloudHSM CLI Anda menggunakan subperintah `configure-cli` seperti yang dijelaskan di bawah ini.](#) `add-cluster`
3. Mulai ulang proses CloudHSM CLI apa pun agar konfigurasi baru diterapkan.

Tambahkan klaster ke AWS CloudHSM konfigurasi Anda

Saat menghubungkan ke beberapa cluster, gunakan `configure-cli add-cluster` perintah untuk menambahkan cluster ke konfigurasi Anda.

Sintaksis

```
configure-cli add-cluster [OPTIONS]
  --cluster-id <CLUSTER ID>
  [--region <REGION>]
  [--endpoint <ENDPOINT>]
  [--hsm-ca-cert <HSM CA CERTIFICATE FILE>]
  [--client-cert-hsm-tls-file <CLIENT CERTIFICATE FILE>]
  [--client-key-hsm-tls-file <CLIENT KEY FILE>]
  [-h, --help]
```

Contoh

Tambahkan cluster menggunakan **cluster-id** parameter

Example

Gunakan `configure-cli add-cluster` bersama dengan `cluster-id` parameter untuk menambahkan cluster (dengan ID `cluster-1234567`) ke konfigurasi Anda.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-cli add-cluster --cluster-id <cluster-1234567>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" add-cluster --cluster-id <cluster-1234567>
```

Tip

Jika menggunakan `configure-cli add-cluster` dengan `cluster-id` parameter tidak mengakibatkan klaster ditambahkan, lihat contoh berikut untuk versi yang lebih panjang dari perintah ini yang juga memerlukan `--region` dan `--endpoint` parameter untuk mengidentifikasi cluster yang ditambahkan. Jika, misalnya, wilayah cluster berbeda dari yang dikonfigurasi sebagai default AWS CLI Anda, Anda harus menggunakan `--region` parameter untuk menggunakan wilayah yang benar. Selain itu, Anda memiliki kemampuan untuk menentukan titik akhir AWS CloudHSM API yang akan digunakan untuk panggilan, yang mungkin diperlukan untuk berbagai pengaturan jaringan, seperti menggunakan titik akhir antarmuka VPC yang tidak menggunakan nama host DNS default untuk AWS CloudHSM

Tambahkan cluster menggunakan **cluster-id**, **endpoint**, dan **region** parameter

Example

Gunakan `region` parameter `configure-cli add-cluster` bersama dengan `cluster-id` dan `endpoint`, dan untuk menambahkan cluster (dengan ID `cluster-1234567`) ke konfigurasi Anda.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-cli add-cluster --cluster-id <cluster-1234567> --region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" add-cluster --cluster-id <cluster-1234567> --region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

Untuk informasi tentang parameter `--cluster-id`, `--region` dan `--endpoint`, lihat [the section called "Parameter"](#).

Parameter

`--kluster-id` **<Cluster ID>**

Membuat panggilan `DescribeClusters` untuk menemukan semua alamat IP antarmuka jaringan elastis (ENI) HSM dalam klaster yang terkait dengan ID klaster. Sistem menambahkan alamat IP ENI ke file AWS CloudHSM konfigurasi.

 Note

Jika Anda menggunakan `--cluster-id` parameter dari sebuah EC2 instance dalam VPC yang tidak memiliki akses ke internet publik, maka Anda harus membuat antarmuka VPC endpoint untuk terhubung dengan. AWS CloudHSM Untuk informasi lebih lanjut tentang VPC endpoint, lihat [???](#).

Wajib: Ya

`--titik akhir` **<Endpoint>**

Tentukan titik akhir AWS CloudHSM API yang digunakan untuk melakukan `DescribeClusters` panggilan. Anda harus menetapkan pilihan ini dalam kombinasi dengan `--cluster-id`.

Wajib: Tidak

`--hsm-ca-cert` **<HsmCA Certificate Filepath>**

Menentukan filepath ke sertifikat HSM CA.

Wajib: Tidak

--wilayah **<Region>**

Tentukan wilayah klaster Anda. Anda harus menetapkan pilihan ini dalam kombinasi dengan --cluster-id.

Jika Anda tidak menyediakan parameter --region, sistem memilih wilayah dengan mencoba untuk membaca variabel lingkungan AWS_DEFAULT_REGION atau AWS_REGION. Jika variabel-variabel tersebut tidak diatur, maka sistem memeriksa wilayah yang terkait dengan profil Anda di file AWS config Anda (biasanya ~/.aws/config) kecuali jika Anda menentukan file yang berbeda di variabel lingkungan AWS_CONFIG_FILE. Jika tidak ada variabel di atas yang diatur, sistem default ke wilayah us-east-1.

Wajib: Tidak

-- client-cert-hsm-tls berkas **<client certificate hsm tls path>**

Jalur ke sertifikat klien yang digunakan untuk otentikasi timbal balik klien TLS-HSM.

Hanya gunakan opsi ini jika Anda telah mendaftarkan setidaknya satu jangkar kepercayaan ke HSM dengan CloudHSM CLI. Anda harus menetapkan pilihan ini dalam kombinasi dengan --client-key-hsm-tls-file.

Wajib: Tidak

-- client-key-hsm-tls berkas **<client key hsm tls path>**

Jalur ke kunci klien yang digunakan untuk otentikasi timbal balik klien TLS-HSM.

Hanya gunakan opsi ini jika Anda telah mendaftarkan setidaknya satu jangkar kepercayaan ke HSM dengan CloudHSM CLI. Anda harus menetapkan pilihan ini dalam kombinasi dengan --client-cert-hsm-tls-file.

Wajib: Tidak

Hapus klaster dari AWS CloudHSM konfigurasi Anda

Saat menghubungkan ke beberapa cluster dengan CloudHSM CLI, gunakan perintah untuk menghapus cluster `configure-cli remove-cluster` dari konfigurasi Anda.

Sintaksis

```
configure-cli remove-cluster [OPTIONS]
```

```
--cluster-id <CLUSTER ID>  
[-h, --help]
```

Contoh

Hapus cluster menggunakan **cluster-id** parameter

Example

Gunakan `configure-cli remove-cluster` bersama dengan `cluster-id` parameter untuk menghapus cluster (dengan ID `cluster-1234567`) dari konfigurasi Anda.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-cli remove-cluster --cluster-id <cluster-1234567>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" remove-cluster --  
cluster-id <cluster-1234567>
```

Untuk informasi tentang parameter `--cluster-id`, lihat [the section called "Parameter"](#).

Parameter

`--cluster-id` <Cluster ID>

ID cluster untuk menghapus dari konfigurasi.

Wajib: Ya

Berinteraksi dengan beberapa cluster di AWS CloudHSM

Setelah mengonfigurasi beberapa cluster dengan CloudHSM CLI, gunakan perintah untuk berinteraksi dengannya. `cloudhsm-cli`

Contoh

Mengatur default **cluster-id** saat menggunakan mode interaktif

Example

Gunakan [???](#) bersama dengan `cluster-id` parameter untuk mengatur cluster default (dengan ID `daric1uster-1234567`) dari konfigurasi Anda.

Linux

```
$ cloudhsm-cli interactive --cluster-id <cluster-1234567>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" interactive --cluster-id <cluster-1234567>
```

Mengatur **cluster-id** saat menjalankan satu perintah

Example

Gunakan `cluster-id` parameter untuk mengatur cluster (dengan ID `daric1uster-1234567`) untuk mendapatkan [???](#) dari.

Linux

```
$ cloudhsm-cli cluster hsm-info --cluster-id <cluster-1234567>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" cluster hsm-info --cluster-id <cluster-1234567>
```

Referensi untuk perintah CloudHSM CLI

CloudHSM CLI membantu admin mengelola pengguna di klaster mereka. AWS CloudHSM CloudHSM CLI dapat dijalankan dalam dua mode: Mode Interaktif dan Mode Perintah Tunggal. Untuk memulai dengan cepat, lihat [Memulai dengan AWS CloudHSM Command Line Interface \(CLI\)](#).

Untuk menjalankan sebagian besar perintah CloudHSM CLI, Anda harus memulai CloudHSM CLI dan masuk ke HSM. Jika Anda menambah atau menghapus HSMs, perbarui file konfigurasi untuk CloudHSM CLI. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSMs di cluster.

Topik berikut menjelaskan perintah di CloudHSM CLI:

Perintah	Deskripsi	Jenis Pengguna
mengaktifkan	Mengaktifkan kluster CloudHSM dan memberikan konfirmasi bahwa kluster tersebut baru. Ini harus dilakukan sebelum operasi lain dapat dilakukan.	Admin yang tidak aktif
info hsm	Daftar HSMs di cluster Anda.	Semua 1 , termasuk pengguna yang tidak diautentikasi. Login tidak diperlukan.
ecdsa	Menghasilkan tanda tangan menggunakan kunci pribadi EC dan mekanisme penandatanganan ECDSA.	Pengguna krypto (CU)
ed25519ph	Menghasilkan tanda tangan menggunakan kunci pribadi Ed25519 dan mekanisme penandatanganan HashEdDSA.	CU
rsa-PKC	Menghasilkan tanda tangan menggunakan kunci pribadi RSA dan mekanisme penandatanganan RSA-PKCS.	CU

Perintah	Deskripsi	Jenis Pengguna
rsa-pkcs-pss	Menghasilkan tanda tangan menggunakan kunci pribadi RSA dan mekanisme RSA-PKCS-PSS penandatanganan.	CU
ecdsa	Mengonfirmasi file telah ditandatangani di HSM oleh kunci publik yang diberikan. Memverifikasi tanda tangan dihasilkan menggunakan mekanisme penandatanganan ECDSA. Membandingkan file yang ditandatangani dengan file sumber dan menentukan apakah keduanya terkait secara kriptografi berdasarkan kunci publik ecdsa dan mekanisme penandatanganan yang diberikan.	CU
ed25519ph	Memverifikasi tanda tangan HashEd DSA menggunakan kunci publik Ed25519.	CU

Perintah	Deskripsi	Jenis Pengguna
rsa-PKC	Mengonfirmasi file telah ditandatangani di HSM oleh kunci publik yang diberikan. Verifikasi tanda tangan dihasilkan menggunakan mekanisme penandatanganan RSA-PKCS. Membandingkan file yang ditandatangani dengan file sumber dan menentukan apakah keduanya terkait secara kriptografi berdasarkan kunci publik rsa dan mekanisme penandatanganan yang diberikan.	CU
rsa-pkcs-pss	Mengonfirmasi file telah ditandatangani di HSM oleh kunci publik yang diberikan. Memverifikasi tanda tangan dihasilkan menggunakan mekanisme RSA-PKCS-PSS penandatanganan. Membandingkan file yang ditandatangani dengan file sumber dan menentukan apakah keduanya terkait secara kriptografi berdasarkan kunci publik rsa dan mekanisme penandatanganan yang diberikan.	CU
hapus kunci	Menghapus kunci dari AWS CloudHSM cluster Anda.	CU

Perintah	Deskripsi	Jenis Pengguna
file menghasilkan kunci	Menghasilkan file kunci di AWS CloudHSM cluster Anda.	CU
kunci generate-asymmetric-pair rsa	Menghasilkan key pair RSA asimetris di cluster Anda AWS CloudHSM .	CU
kunci generate-asymmetric-pair ec	Menghasilkan key pair elliptic-curve (EC) asimetris di cluster Anda. AWS CloudHSM	CU
kunci menghasilkan aes simetris	Menghasilkan kunci AES simetris di cluster Anda AWS CloudHSM .	CU
kunci generik-rahasia generik-simetris	Menghasilkan kunci Rahasia Generik simetris di cluster Anda AWS CloudHSM .	CU
kunci impor pem	Mengimpor kunci format PEM ke HSM. Anda dapat menggunakannya untuk mengimpor kunci privat yang dihasilkan di luar HSM.	CU
daftar kunci	Menemukan semua kunci untuk pengguna saat ini yang ada di AWS CloudHSM cluster Anda.	CU
replikasi kunci	Replikasi kunci dari cluster sumber ke klaster tujuan kloning.	CU
kunci set-attribut	Menetapkan atribut kunci di AWS CloudHSM cluster Anda.	CUs dapat menjalankan perintah ini, admin dapat mengatur atribut tepercaya.

Perintah	Deskripsi	Jenis Pengguna
berbagi kunci	Berbagi kunci dengan yang lain CUs di AWS CloudHSM cluster Anda.	CU
kunci unshare	Membatalkan kunci dengan yang lain CUs di AWS CloudHSM cluster Anda.	CU
aes-gcm	Membuka kunci payload ke dalam cluster menggunakan kunci pembungkus AES dan mekanisme unwrapping AES-GCM.	CU
aes-no-pad	Membuka kunci payload ke dalam cluster menggunakan kunci pembungkus AES dan mekanisme unwrapping. AES-NO-PAD	CU
aes-pkcs5-pad	Membuka kunci payload menggunakan kunci pembungkus AES dan mekanisme unwrapping AES-PAD. PKCS5	CU
aes-zero-pad	Membuka kunci payload ke dalam cluster menggunakan kunci pembungkus AES dan mekanisme unwrapping. AES-ZERO-PAD	CU

Perintah	Deskripsi	Jenis Pengguna
cloudhsm-aes-gcm	Membuka kunci payload ke dalam cluster menggunakan kunci pembungkus AES dan mekanisme unwrapping. CLOUDHSM-AES-GCM	CU
rsa-aes	Membuka kunci payload menggunakan kunci pribadi RSA dan mekanisme unwrapping RSA-AES.	CU
rsa-oaep	Membuka kunci payload menggunakan kunci pribadi RSA dan mekanisme unwrapping RSA-OAEP.	CU
rsa-PKC	Membuka kunci payload menggunakan kunci pribadi RSA dan mekanisme unwrapping RSA-PKCS.	CU
aes-gcm	Membungkus kunci payload menggunakan kunci AES pada HSM dan mekanisme pembungkus AES-GCM.	CU
aes-no-pad	Membungkus kunci payload menggunakan kunci AES pada HSM dan mekanisme pembungkus. AES-NO-PAD	CU
aes-pkcs5-pad	Membungkus kunci payload menggunakan kunci AES pada HSM dan mekanisme pembungkus AES- -PAD. PKCS5	CU

Perintah	Deskripsi	Jenis Pengguna
aes-zero-pad	Membungkus kunci payload menggunakan kunci AES pada HSM dan mekanisme pembungkus. AES-ZERO-PAD	CU
cloudhsm-aes-gcm	Membungkus kunci payload menggunakan kunci AES pada HSM dan mekanisme pembungkus. CLOUDHSM-AES-GCM	CUs
rsa-aes	Membungkus kunci payload menggunakan kunci publik RSA pada HSM dan mekanisme pembungkus RSA-AES.	CU
rsa-oaep	Membungkus kunci payload menggunakan kunci publik RSA pada HSM dan mekanisme pembungkus RSA-OAEP.	CU

Perintah	Deskripsi	Jenis Pengguna
Gunakan key wrap rsa-pkcs perintah di CloudHSM CLI untuk membungkus kunci payload menggunakan kunci publik RSA pada modul keamanan perangkat keras (HSM) dan mekanisme pembungkus. RSA-PKCS extractable Atribut kunci payload harus disetel ke true. Hanya pemilik kunci, yaitu pengguna kriptografi (CU) yang membuat kunci, yang dapat membungkus kunci tersebut. Pengguna yang berbagi kunci dapat menggunakan kunci dalam operasi kriptografi. Untuk menggunakan key wrap rsa-pkcs perintah, Anda harus terlebih dahulu memiliki kunci RSA di AWS CloudHSM cluster Anda. Anda dapat menghasilkan key pair RSA menggunakan generate-asymmetric-pair Kategori di CloudHSM CLI perintah dan wrap atribut yang disetel ke true. Jenis pengguna Jenis pengguna berikut dapat menjalankan perintah ini. • Pengguna Crypto (CUs)	Membungkus kunci payload menggunakan kunci publik RSA pada HSM dan mekanisme pembungkus RSA-PKCS.	CU
Referensi Persyaratan • Untuk menjalankan perintah		

Perintah	Deskripsi	Jenis Pengguna
login	Masuk ke AWS CloudHSM cluster Anda.	Admin, pengguna kriptografi (CU), dan pengguna alat (AU)
logout	Keluar dari AWS CloudHSM cluster Anda.	Admin, CU, dan pengguna alat (AU)
kuorum tanda token hapus	Menghapus satu atau beberapa token untuk layanan resmi kuorum.	Admin
tanda token kuorum menghasilkan	Menghasilkan token untuk layanan resmi kuorum.	Admin
daftar tanda token kuorum	Daftar semua token kuorum tanda token yang ada di kluster CloudHSM Anda.	Semua ¹ , termasuk pengguna yang tidak diautentikasi. Login tidak diperlukan.
tanda token kuorum list-quorum-values	Daftar nilai kuorum yang ditetapkan dalam kluster CloudHSM Anda.	Semua ¹ , termasuk pengguna yang tidak diautentikasi. Login tidak diperlukan.
tanda token kuorum set-quorum-value	Menetapkan nilai kuorum baru untuk layanan resmi kuorum.	Admin
perubahan pengguna-mfa	Mengubah strategi otentikasi multi-faktor (MFA) pengguna.	Admin, CU
perubahan kata sandi pengguna	Mengubah kata sandi pengguna di file HSMs. Setiap pengguna dapat mengubah kata sandi mereka sendiri. Admin dapat mengubah kata sandi siapa pun.	Admin, CU
pengguna membuat	Membuat pengguna di AWS CloudHSM cluster Anda.	Admin

Perintah	Deskripsi	Jenis Pengguna
menghapus pengguna	Menghapus pengguna di AWS CloudHSM cluster Anda.	Admin
daftar pengguna	Daftar pengguna di AWS CloudHSM cluster Anda.	Semua ¹ , termasuk pengguna yang tidak diautentikasi. Login tidak diperlukan.
daftar tanda token ubah-kuorum pengguna	Mendaftarkan strategi kuorum tanda token kuorum untuk pengguna.	Admin

Anotasi

- [1] Semua pengguna menyertakan semua peran yang terdaftar dan pengguna yang tidak masuk.

Kategori cluster di CloudHSM CLI

Di CloudHSM cluster CLI, adalah kategori induk untuk sekelompok perintah yang, bila dikombinasikan dengan kategori induk, membuat perintah khusus untuk cluster. Saat ini, kategori cluster terdiri dari perintah berikut:

Topik

- [Aktifkan cluster dengan CloudHSM CLI](#)
- [Daftar HSMs dengan CloudHSM CLI](#)
- [Kategori cluster mTLS di CloudHSM CLI](#)

Aktifkan cluster dengan CloudHSM CLI

Gunakan cluster activate perintah di CloudHSM [CLI untuk](#) mengaktifkan cluster baru di. AWS CloudHSM Perintah ini harus dijalankan sebelum cluster dapat digunakan untuk melakukan operasi kriptografi.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Admin yang tidak aktif

Sintaksis

Perintah ini tidak memiliki parameter.

```
aws-cloudhsm > help cluster activate
```

Activate a cluster

This command will set the initial Admin password. This process will cause your CloudHSM cluster to move into the ACTIVE state.

USAGE:

```
cloudhsm-cli cluster activate [OPTIONS] [--password <PASSWORD>]
```

Options:

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

```
--password <PASSWORD>
```

Optional: Plaintext activation password If you do not include this argument you will be prompted for it

```
-h, --help
```

Print help (see a summary with '-h')

Contoh

Perintah ini mengaktifkan cluster Anda dengan mengatur kata sandi awal untuk pengguna admin Anda.

```
aws-cloudhsm > cluster activate
```

Enter password:

Confirm password:

```
{  
  "error_code": 0,  
  "data": "Cluster activation successful"  
}
```

Topik terkait

- [pengguna membuat](#)
- [menghapus pengguna](#)
- [perubahan kata sandi pengguna](#)

Daftar HSMs dengan CloudHSM CLI

Gunakan cluster `hsm-info` perintah di CloudHSM CLI untuk membuat daftar HSMs modul keamanan perangkat keras () di cluster Anda. AWS CloudHSM Anda tidak perlu login ke CloudHSM CLI untuk menjalankan perintah ini.

Note

Jika Anda menambah atau menghapus HSMs, perbarui file konfigurasi yang digunakan AWS CloudHSM klien dan alat baris perintah. Jika tidak, perubahan yang Anda buat mungkin tidak efektif pada semua HSMs di cluster.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Semua pengguna. Anda tidak perlu masuk untuk menjalankan perintah ini.

Sintaksis

```
aws-cloudhsm > help cluster hsm-info  
List info about each HSM in the cluster
```

```
Usage: cloudhsm-cli cluster hsm-info [OPTIONS]
```

Options:

```
--cluster-id <CLUSTER_ID> Unique Id to choose which of the clusters in the  
config file to run the operation against. If not provided, will fall back to the value  
provided when interactive mode was started, or error  
-h, --help Print help
```

Contoh

Perintah ini mencantumkan yang HSMs ada di AWS CloudHSM cluster Anda.

```
aws-cloudhsm > cluster hsm-info
{
  "error_code": 0,
  "data": {
    "hsm": [
      {
        "vendor": "Marvell Semiconductors, Inc.",
        "model": "NITROX-III CNN35XX-NFBE",
        "serial-number": "5.3G1941-ICM000590",
        "hardware-version-major": "5",
        "hardware-version-minor": "3",
        "firmware-version-major": "2",
        "firmware-version-minor": "6",
        "firmware-build-number": "16",
        "firmware-id": "CNN35XX-NFBE-FW-2.06-16"
        "fips-state": "2 [FIPS mode with single factor authentication]"
      },
      {
        "vendor": "Marvell Semiconductors, Inc.",
        "model": "NITROX-III CNN35XX-NFBE",
        "serial-number": "5.3G1941-ICM000625",
        "hardware-version-major": "5",
        "hardware-version-minor": "3",
        "firmware-version-major": "2",
        "firmware-version-minor": "6",
        "firmware-build-number": "16",
        "firmware-id": "CNN35XX-NFBE-FW-2.06-16"
        "fips-state": "2 [FIPS mode with single factor authentication]"
      },
      {
        "vendor": "Marvell Semiconductors, Inc.",
        "model": "NITROX-III CNN35XX-NFBE",
        "serial-number": "5.3G1941-ICM000663",
        "hardware-version-major": "5",
        "hardware-version-minor": "3",
        "firmware-version-major": "2",
        "firmware-version-minor": "6",
        "firmware-build-number": "16",
        "firmware-id": "CNN35XX-NFBE-FW-2.06-16"
        "fips-state": "2 [FIPS mode with single factor authentication]"
      }
    ]
  }
}
```

```
}  
]  
}  
}
```

Output memiliki atribut berikut:

- Vendor: Nama vendor dari HSM.
- Model: Nomor model HSM.
- Nomor seri: Nomor seri HSM. Ini dapat berubah karena penggantian.
- Hardware-version-major: Versi perangkat keras utama.
- Hardware-version-minor: Versi perangkat keras minor.
- Firmware-version-major: Versi firmware utama.
- Firmware-version-minor: Versi firmware minor.
- Firmware-build-number: Nomor build firmware.
- Firmware-ID: ID firmware, yang mencakup versi mayor dan minor bersama dengan build.
- FIPS-state: FIPS modus cluster dan di dalamnya. HSMs Jika dalam mode FIPS, outputnya adalah “2 [mode FIPS dengan otentikasi faktor tunggal].” Jika dalam mode non-FIPS, outputnya adalah “0 [mode non-FIPS dengan otentikasi faktor tunggal]”.

Topik terkait

- [Aktifkan cluster dengan CloudHSM CLI](#)

Kategori cluster mtls di CloudHSM CLI

Di CloudHSM cluster mtls CLI, adalah kategori induk untuk sekelompok perintah yang, bila dikombinasikan dengan kategori induk, membuat perintah khusus untuk cluster. AWS CloudHSM Saat ini, kategori ini terdiri dari perintah berikut:

Topik

- [Deregister jangkar kepercayaan dengan CloudHSM CLI](#)
- [Dapatkan tingkat penegakan mTLS dengan CloudHSM CLI](#)
- [Daftar jangkar kepercayaan dengan CloudHSM CLI](#)
- [Daftarkan jangkar kepercayaan dengan CloudHSM CLI](#)

- [Atur tingkat penegakan mTLS dengan CloudHSM CLI](#)

Deregister jangkar kepercayaan dengan CloudHSM CLI

Gunakan cluster mTLS deregister-trust-anchor perintah di CloudHSM CLI untuk membatalkan pendaftaran jangkar kepercayaan untuk TLS timbal balik antara klien dan AWS CloudHSM

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Admin

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai pengguna admin.

Sintaksis

```
aws-cloudhsm > help cluster mTLS deregister-trust-anchor
```

Deregister a trust anchor for mTLS

Usage: cluster mTLS deregister-trust-anchor [OPTIONS] --certificate-reference
[<CERTIFICATE_REFERENCE>...]

Options:

```
--certificate-reference <CERTIFICATE_REFERENCE> A hexadecimal or decimal  
certificate reference  
--cluster-id <CLUSTER_ID> Unique Id to choose which of the clusters in the  
config file to run the operation against. If not provided, will fall back to the value  
provided when interactive mode was started, or error  
--approval <APPROVAL> Filepath of signed quorum token file to approve operation  
-h, --help Print help
```

Contoh

Example

Dalam contoh berikut, perintah ini menghapus jangkar kepercayaan dari HSM.

```
aws-cloudhsm > cluster mtls deregister-trust-anchor --certificate-reference 0x01

{
  "error_code": 0,
  "data": {
    "message": "Trust anchor with reference 0x01 deregistered successfully"
  }
}
```

Anda kemudian dapat menjalankan list-trust-anchors perintah untuk mengonfirmasi bahwa jangka kepercayaan telah dideregistrasi dari: AWS CloudHSM

```
aws-cloudhsm > cluster mtls list-trust-anchors

{
  "error_code": 0,
  "data": {
    "trust_anchors": []
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Diperlukan: Jika beberapa cluster telah [dikonfigurasi](#).

<CERTIFICATE_REFERENCE>

Referensi sertifikat heksadesimal atau desimal.

Wajib: Ya

Warning

Setelah Anda membatalkan pendaftaran jangka kepercayaan di cluster, semua koneksi mTL yang ada menggunakan sertifikat klien yang ditandatangani oleh jangka kepercayaan tersebut akan dihapus.

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan klaster kuorum lebih besar dari 1.

Topik terkait

- [cluster mtls reregister-trust-anchor](#)
- [cluster mtls list-trust-anchors](#)
- [Pengaturan mTL \(disarankan\)](#)

Dapatkan tingkat penegakan mTLS dengan CloudHSM CLI

Gunakan `cluster mtls get-enforcement` perintah di CloudHSM CLI untuk mendapatkan tingkat penegakan penggunaan TLS timbal balik antara klien dan. AWS CloudHSM

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Admin
- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai pengguna admin atau pengguna krypto (CUs).

Sintaksis

```
aws-cloudhsm > help cluster mtls get-enforcement
```

```
Get the status of mtls enforcement in the cluster
```

```
Usage: cluster mtls get-enforcement [OPTIONS]
```

```
Options:
```

```
--cluster-id <CLUSTER_ID> Unique Id to choose which of the clusters in the
config file to run the operation against. If not provided, will fall back to the value
provided when interactive mode was started, or error
-h, --help Print help
```

Contoh

Example

Dalam contoh berikut, perintah ini mencantumkan tingkat penegakan mtls dari. AWS CloudHSM

```
aws-cloudhsm > cluster mtls get-enforcement

{
  "error_code": 0,
  "data": {
    "mtls-enforcement-level": "none"
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

Topik terkait

- [cluster mtls set-penegakan](#)
- [Pengaturan mTL \(disarankan\)](#)

Daftar jangkar kepercayaan dengan CloudHSM CLI

Gunakan cluster mtls list-trust-anchors perintah di CloudHSM CLI untuk membuat daftar semua jangkar kepercayaan yang dapat digunakan untuk TLS timbal balik antara klien dan. AWS CloudHSM

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Semua pengguna. Anda tidak perlu masuk untuk menjalankan perintah ini.

Sintaksis

```
aws-cloudhsm > help cluster mtls list-trust-anchors
```

List all trust anchors for mtls

Usage: cluster mtls list-trust-anchors [OPTIONS]

Options:

`--cluster-id <CLUSTER_ID>` Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

`-h, --help` Print help

Contoh

Example

Dalam contoh berikut, perintah ini mencantumkan semua jangkar kepercayaan terdaftar dari AWS CloudHSM

```
aws-cloudhsm > cluster mtls list-trust-anchors
```

```
{
  "error_code": 0,
  "data": {
    "trust_anchors": [
      {
        "certificate-reference": "0x01",
        "certificate": "<PEM Encoded Certificate 1>",
        "cluster-coverage": "full"
      },
      {
        "certificate-reference": "0x02",
        "certificate": "<PEM Encoded Certificate 2>",

```

```
    "cluster-coverage": "full"
  }
]
}
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

Topik terkait

- [kluster mtls reregister-trust-anchor](#)
- [kluster mtls deregister-trust-anchor](#)
- [Pengaturan mTL \(disarankan\)](#)

Daftarkan jangkar kepercayaan dengan CloudHSM CLI

Gunakan cluster mtls register-trust-anchor perintah di CloudHSM CLI untuk mendaftarkan jangkar kepercayaan untuk TLS timbal balik antara klien dan. AWS CloudHSM

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Admin

Persyaratan

AWS CloudHSM Menerima jangkar kepercayaan dengan jenis kunci berikut:

Tipe Kunci	Deskripsi
EC	kurva secp256r1 (P-256), secp384r1 (P-384), dan secp521r1 (P-521).

Tipe Kunci	Deskripsi
RSA	Kunci RSA 2048-bit, 3072-bit, dan 4096-bit.

Sintaksis

```
aws-cloudhsm > help cluster mtls register-trust-anchor
```

Register a trust anchor for mtls

Usage: cluster mtls register-trust-anchor [OPTIONS] --path [*<PATH>*...]

Options:

- cluster-id *<CLUSTER_ID>* Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error
- path *<PATH>* Filepath of the trust anchor to register
- approval *<APPROVAL>* Filepath of signed quorum token file to approve operation
- h, --help Print help

Contoh

Example

Dalam contoh berikut, perintah ini mendaftarkan jangkar kepercayaan ke HSM. Jumlah maksimum jangkar kepercayaan yang dapat didaftarkan adalah dua (2).

```
aws-cloudhsm > cluster mtls register-trust-anchor --path /home/rootCA
```

```
{
  "error_code": 0,
  "data": {
    "trust_anchor": {
      "certificate-reference": "0x01",
      "certificate": "<PEM Encoded Certificate>",
      "cluster-coverage": "full"
    }
  }
}
```

Anda kemudian dapat menjalankan `list-trust-anchors` perintah untuk mengonfirmasi bahwa jangkar kepercayaan telah terdaftar ke AWS CloudHSM:

```
aws-cloudhsm > cluster mtls list-trust-anchors

{
  "error_code": 0,
  "data": {
    "trust_anchors": [
      {
        "certificate-reference": "0x01",
        "certificate": "<PEM Encoded Certificate>",
        "cluster-coverage": "full"
      }
    ]
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<PATH>

Filepath dari jangkar kepercayaan untuk mendaftar.

Wajib: Ya

Note

AWS CloudHSM mendukung pendaftaran sertifikat perantara sebagai jangkar kepercayaan. Dalam kasus seperti itu, seluruh file rantai sertifikat yang dikodekan PEM perlu didaftarkan ke HSM, dengan sertifikat dalam urutan hierarkis. AWS CloudHSM mendukung rantai sertifikat 6980 byte.

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan kluster kuorum lebih besar dari 1.

Topik terkait

- [kluster mtls deregister-trust-anchor](#)
- [kluster mtls list-trust-anchors](#)
- [Pengaturan mTL \(disarankan\)](#)

Atur tingkat penegakan mTLS dengan CloudHSM CLI

Gunakan `cluster mtls set-enforcement` perintah di CloudHSM CLI untuk mengatur tingkat penegakan penggunaan TLS timbal balik antara klien dan AWS CloudHSM

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Admin dengan nama pengguna sebagai admin

Persyaratan

Untuk menjalankan perintah ini:

- Setidaknya satu jangkar kepercayaan telah berhasil didaftarkan ke AWS CloudHSM
- Konfigurasi CloudHSM CLI dengan kunci pribadi dan sertifikat klien yang tepat, dan mulai CloudHSM CLI di bawah koneksi TLS timbal balik.
- Anda harus masuk sebagai admin default dengan nama pengguna "admin". Pengguna admin lain tidak akan dapat menjalankan perintah ini.

Sintaksis

```
aws-cloudhsm > help cluster mtls set-enforcement
```

```
Set mtls enforcement policy in the cluster
```

```
Usage: cluster mtls set-enforcement [OPTIONS] --level [<LEVEL>...]
```

Options:

```
--cluster-id <CLUSTER_ID> Unique Id to choose which of the clusters in the
config file to run the operation against. If not provided, will fall back to the value
provided when interactive mode was started, or error
--level <LEVEL> Level to be set for mtls in the cluster [possible values: none,
cluster]
--approval <APPROVAL> Filepath of signed quorum token file to approve operation
-h, --help Print help
```

Contoh**Example**

Dalam contoh berikut, perintah ini mengatur tingkat penegakan mtls AWS CloudHSM untuk menjadi cluster. Perintah set-enforcement hanya dapat dilakukan dalam koneksi TLS timbal balik dan masuk sebagai pengguna admin dengan nama pengguna sebagai admin, lihat [mengatur penegakan mTLS untuk](#). AWS CloudHSM

```
aws-cloudhsm > cluster mtls set-enforcement --level cluster

{
  "error_code": 0,
  "data": {
    "message": "Mtls enforcement level set to Cluster successfully"
  }
}
```

Anda kemudian dapat menjalankan get-enforcement perintah untuk mengonfirmasi bahwa tingkat penegakan telah disetel ke cluster:

```
aws-cloudhsm > cluster mtls get-enforcement

{
  "error_code": 0,
  "data": {
    "mtls-enforcement-level": "cluster"
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<LEVEL>

Level yang akan ditetapkan untuk mtls di cluster.

Nilai yang valid

- cluster: Menerapkan penggunaan TLS timbal balik antara klien dan AWS CloudHSM di cluster.
- none: Jangan memaksakan penggunaan TLS timbal balik antara klien dan AWS CloudHSM di cluster.

Wajib: Ya

Warning

Setelah Anda menerapkan penggunaan mTLS di cluster, semua koneksi non-MTLS yang ada akan dihapus dan Anda hanya dapat terhubung ke cluster dengan sertifikat mTLS.

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan klaster kuorum lebih besar dari 1.

Topik terkait

- [cluster mtls mendapatkan penegakan](#)
- [Pengaturan mTL \(disarankan\)](#)

Kategori krypto di CloudHSM CLI

Dalam CloudHSM crypto CLI, adalah kategori induk untuk sekelompok perintah yang, ketika dikombinasikan dengan kategori induk, membuat perintah khusus untuk operasi kriptografi. Saat ini, kategori ini terdiri dari perintah berikut:

- [tanda tangan](#)
 - [ecdsa](#)
 - [ed25519ph](#)
 - [rsa-PKC](#)
 - [rsa-pkcs-pss](#)
- [verifikasi](#)
 - [ecdsa](#)
 - [ed25519ph](#)
 - [rsa-PKC](#)
 - [rsa-pkcs-pss](#)

Kategori tanda kriptografi di CloudHSM CLI

Di CloudHSM crypto sign CLI, adalah kategori induk untuk sekelompok perintah yang, bila digabungkan dengan kategori induk, menggunakan kunci pribadi yang dipilih AWS CloudHSM di cluster Anda untuk menghasilkan tanda tangan. crypto sign memiliki subperintah berikut:

- [Hasilkan tanda tangan dengan mekanisme ECDSA di CloudHSM CLI](#)
- [Hasilkan tanda tangan dengan mekanisme HashEd DSA di CloudHSM CLI](#)
- [Hasilkan tanda tangan dengan mekanisme RSA-PKCS di CloudHSM CLI](#)
- [Hasilkan tanda tangan dengan RSA-PKCS-PSS mekanisme di CloudHSM CLI](#)

Untuk menggunakannya crypto sign, Anda harus memiliki kunci pribadi di HSM Anda. Anda dapat membuat kunci pribadi dengan perintah berikut:

- [kunci generate-asymmetric-pair ec](#)
- [kunci generate-asymmetric-pair rsa](#)

Hasilkan tanda tangan dengan mekanisme ECDSA di CloudHSM CLI

Gunakan crypto sign ecdsa perintah di CloudHSM CLI untuk menghasilkan tanda tangan menggunakan kunci pribadi EC dan mekanisme penandatanganan ECDSA.

Untuk menggunakan crypto sign ecdsa perintah, Anda harus terlebih dahulu memiliki kunci pribadi EC di AWS CloudHSM cluster Anda. Anda dapat menghasilkan kunci pribadi EC menggunakan

[Hasilkan key pair EC asimetris dengan CloudHSM CLI](#) perintah dengan `sign` atribut yang disetel `ketrue`.

Tanda tangan ECDSA yang dihasilkan dihasilkan dalam format `r | s`, di mana komponen `r` dan `s` digabungkan sebagai data biner mentah dan dikembalikan dalam format yang dikodekan base64.

Note

Tanda tangan dapat diverifikasi AWS CloudHSM dengan [Kategori verifikasi krypto di CloudHSM CLI](#) subperintah.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help crypto sign ecdsa
```

Sign with the ECDSA mechanism

Usage: `crypto sign ecdsa --key-filter [<KEY_FILTER>] --hash-function <HASH_FUNCTION> [--data-path <DATA_PATH> | --data <DATA>]`

Options:

`--cluster-id <CLUSTER_ID>`

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

`--key-filter [<KEY_FILTER>]`

Key reference (e.g. `key-reference=0xabc`) or space separated list of key attributes in the form of `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` to select a matching key

`--hash-function <HASH_FUNCTION>`

[possible values: `sha1`, `sha224`, `sha256`, `sha384`, `sha512`]

```

--data-path <DATA_PATH>
    The path to the file containing the data to be signed
--data <DATA>
    Base64 Encoded data to be signed
--approval <APPROVAL>
    Filepath of signed quorum token file to approve operation
--data-type <DATA_TYPE>
    The type of data passed in, either raw or digest [possible values: raw,
digest]
-h, --help
    Print help

```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `crypto sign ecdsa` untuk menghasilkan tanda tangan menggunakan mekanisme penandatanganan ECDSA dan fungsi SHA256 hash. Perintah ini menggunakan kunci pribadi di HSM.

Example Contoh: Menghasilkan tanda tangan untuk data dasar 64 yang dikodekan

```

aws-cloudhsm > crypto sign ecdsa --key-filter attr.label=ec-private --hash-function
sha256 --data YWJjMTIz
{
  "error_code": 0,
  "data": {
    "key-reference": "0x000000000007808dd",
    "signature": "4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2w0S3Q5bzk
+Hc5irV5iTkuxQbropPttVFZ8V6FgR2fz+sPegwCw=="
  }
}

```

Example Contoh: Menghasilkan tanda tangan untuk file data

```

aws-cloudhsm > crypto sign ecdsa --key-filter attr.label=ec-private --hash-function
sha256 --data-path data.txt
{
  "error_code": 0,
  "data": {
    "key-reference": "0x000000000007808dd",
    "signature": "4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2w0S3Q5bzk
+Hc5irV5iTkuxQbropPttVFZ8V6FgR2fz+sPegwCw=="
  }
}

```

```
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<DATA>

Data yang dikodekan Base64 untuk ditandatangani.

Wajib: Ya (kecuali disediakan melalui jalur data)

<DATA_PATH>

Menentukan lokasi data yang akan ditandatangani.

Wajib: Ya (kecuali disediakan melalui jalur data)

<HASH_FUNCTION>

Menentukan fungsi hash.

Nilai valid:

- sha1
- sha224
- sha256
- sha384
- sha512

Wajib: Ya

<KEY_FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `ATTR.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang cocok.

Untuk daftar atribut kunci CloudHSM CLI yang didukung, lihat Atribut kunci untuk CloudHSM CLI.

Wajib: Ya

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan penggunaan kunci dari kunci pribadi lebih besar dari 1.

<DATA_TYPE>

Menentukan apakah nilai parameter data harus di-hash sebagai bagian dari algoritma penandatanganan. Gunakan `raw` untuk data yang tidak di-hash; gunakan `digest` untuk intisari, yang sudah di-hash.

Nilai valid:

- mentah
- mencerna

Topik terkait

- [Kategori tanda krypto di CloudHSM CLI](#)
- [Kategori verifikasi krypto di CloudHSM CLI](#)

Hasilkan tanda tangan dengan mekanisme HashEd DSA di CloudHSM CLI

Important

HashEdOperasi penandatanganan DSA hanya didukung pada instance `hsm2m.medium` dalam mode non-FIPS.

Gunakan `crypto sign ed25519ph` perintah di CloudHSM CLI untuk menghasilkan tanda tangan menggunakan kunci pribadi Ed25519 dan mekanisme penandatanganan DSA. HashEd Untuk informasi tambahan tentang HashEd DSA, lihat [NIST SP 186-5](#), Bagian 7.8.

Untuk menggunakan `crypto sign ed25519ph` perintah, Anda harus terlebih dahulu memiliki kunci pribadi Ed25519 di cluster Anda. AWS CloudHSM Anda dapat menghasilkan kunci pribadi Ed25519 menggunakan [Hasilkan key pair EC asimetris dengan CloudHSM CLI](#) perintah dengan `curve` parameter disetel ke `ed25519` dan `sign` atribut disetel ke `true`

Note

Tanda tangan dapat diverifikasi AWS CloudHSM dengan [Kategori verifikasi krypto di CloudHSM CLI](#) subperintah.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.
- HashEdOperasi penandatanganan DSA hanya didukung pada instance hsm2m.medium dalam mode non-FIPS.

Sintaksis

```
aws-cloudhsm > help crypto sign ed25519ph
```

Sign with the Ed25519ph mechanism

Usage: `crypto sign ed25519ph [OPTIONS] --key-filter [<KEY_FILTER>...] --data-type <DATA_TYPE> --hash-function <HASH_FUNCTION> <--data-path <DATA_PATH>|--data <DATA>>`

Options:

`--cluster-id <CLUSTER_ID>`

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

`--key-filter [<KEY_FILTER>...]`

Key reference (e.g. `key-reference=0xabc`) or space separated list of key attributes in the form of `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` to select a matching key

`--data-path <DATA_PATH>`

The path to the file containing the data to be signed

`--data <DATA>`

Base64 Encoded data to be signed

```

--approval <APPROVAL>
    Filepath of signed quorum token file to approve operation
--data-type <DATA_TYPE>
    The type of data passed in, either raw or digest [possible values: raw,
digest]
--hash-function <HASH_FUNCTION>
    Hash function [possible values: sha512]
-h, --help
    Print help

```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan crypto sign ed25519ph untuk menghasilkan tanda tangan menggunakan mekanisme penandatanganan ED25519ph dan fungsi hash. sha512. Perintah ini menggunakan kunci pribadi di HSM.

Example Contoh: Menghasilkan tanda tangan untuk data dasar 64 yang dikodekan

```

aws-cloudhsm > crypto sign ed25519ph \
  --key-filter attr.label=ed25519-private \
  --data-type raw \
  --hash-function sha512 \
  --data YWJj
{
  "error_code": 0,
  "data": {
    "key-reference": "0x000000000000401cdf",
    "signature": "mKcCIvC4Ehqp0w+BPWg/gJ5GK0acf/h20UmbuU5trkEx
+FBCRjwqNVogA9BirfWqoQuMYeY2Biqq0RwqJgg0Bg=="
  }
}

```

Example Contoh: Menghasilkan tanda tangan untuk file data

```

aws-cloudhsm > crypto sign ed25519ph \
  --key-filter attr.label=ed25519-private \
  --data-type raw \
  --hash-function sha512 \
  --data-path data.txt
{
  "error_code": 0,
  "data": {

```

```

    "key-reference": "0x00000000000401cdf",
    "signature": "mKcCIvC4Ehqp0w+BPWg/gJ5GK0acf/h20UmbuU5trkEx
+FBCRjwqNVogA9BirfWqoQuMYeY2Biqq0RwqJgg0Bg=="
  }
}

```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<DATA>

Data yang dikodekan Base64 untuk ditandatangani.

Wajib: Ya (kecuali disediakan melalui jalur data)

<DATA_PATH>

Menentukan lokasi data yang akan ditandatangani.

Diperlukan: Ya (kecuali disediakan melalui parameter data)

<HASH_FUNCTION>

Menentukan fungsi hash. Ed25519ph hanya mendukung. SHA512

Nilai valid:

- sha512

Wajib: Ya

<KEY_FILTER>

Referensi kunci (misalnya, key-reference=0xabc) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk ATTR.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk memilih kunci yang cocok.

Untuk daftar atribut kunci CloudHSM CLI yang didukung, lihat. [Atribut kunci untuk CloudHSM CLI](#)

Wajib: Ya

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan penggunaan kunci dari kunci pribadi lebih besar dari 1.

<DATA_TYPE>

Menentukan apakah nilai parameter data harus di-hash sebagai bagian dari algoritma penandatanganan. Gunakan `raw` untuk data yang tidak di-hash; gunakan `digest` untuk intisari, yang sudah di-hash.

Nilai valid:

- `mentah`
- `mencerna`

Wajib: Ya

Topik terkait

- [Kategori tanda krypto di CloudHSM CLI](#)
- [Kategori verifikasi krypto di CloudHSM CLI](#)

Hasilkan tanda tangan dengan mekanisme RSA-PKCS di CloudHSM CLI

Gunakan `crypto sign rsa-pkcs` perintah di CloudHSM CLI untuk menghasilkan tanda tangan menggunakan kunci pribadi RSA dan mekanisme penandatanganan RSA-PKCS.

Untuk menggunakan `crypto sign rsa-pkcs` perintah, Anda harus terlebih dahulu memiliki kunci pribadi RSA di AWS CloudHSM cluster Anda. Anda dapat menghasilkan kunci pribadi RSA menggunakan [Hasilkan key pair RSA asimetris dengan CloudHSM CLI](#) perintah dengan `sign` atribut yang disetel `true`.

Note

Tanda tangan dapat diverifikasi AWS CloudHSM dengan [Kategori verifikasi krypto di CloudHSM CLI](#) subperintah.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help crypto sign rsa-pkcs
```

Sign with the RSA-PKCS mechanism

Usage: `crypto sign rsa-pkcs --key-filter [<KEY_FILTER>...] --hash-function <HASH_FUNCTION> [--data-path <DATA_PATH>|--data <DATA>]`

Options:

`--cluster-id <CLUSTER_ID>`

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

`--key-filter [<KEY_FILTER>...]`

Key reference (e.g. `key-reference=0xabc`) or space separated list of key attributes in the form of `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` to select a matching key

`--hash-function <HASH_FUNCTION>`

[possible values: sha1, sha224, sha256, sha384, sha512]

`--data-path <DATA_PATH>`

The path to the file containing the data to be signed

`--data <DATA>`

Base64 Encoded data to be signed

`--approval <APPROVAL>`

Filepath of signed quorum token file to approve operation

`--data-type <DATA_TYPE>`

The type of data passed in, either raw or digest [possible values: raw, digest]

`-h, --help`

Print help

Contoh

Contoh-contoh ini menunjukkan cara menggunakan crypto sign rsa-pkcs untuk menghasilkan tanda tangan menggunakan mekanisme penandatanganan RSA-PKCS dan fungsi hash. SHA256 Perintah ini menggunakan kunci pribadi di HSM.

Example Contoh: Menghasilkan tanda tangan untuk data dasar 64 yang dikodekan

```
aws-cloudhsm > crypto sign rsa-pkcs --key-filter attr.label=rsa-private --hash-function sha256 --data YWJjMTIz
{
  "error_code": 0,
  "data": {
    "key-reference": "0x000000000007008db",
    "signature": "XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjg0QW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBj0BhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+G09BUT9FFU/
of9cvu40yn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEivFLG61sexJjQi8EdTxeDA
+I3IT00qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ=="
  }
}
```

Example Contoh: Menghasilkan tanda tangan untuk file data

```
aws-cloudhsm > crypto sign rsa-pkcs --key-filter attr.label=rsa-private --hash-function sha256 --data-path data.txt
{
  "error_code": 0,
  "data": {
    "key-reference": "0x000000000007008db",
    "signature": "XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjg0QW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBj0BhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+G09BUT9FFU/
of9cvu40yn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEivFLG61sexJjQi8EdTxeDA
+I3IT00qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ=="
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<DATA>

Data yang dikodekan Base64 untuk ditandatangani.

Wajib: Ya (kecuali disediakan melalui jalur data)

<DATA_PATH>

Menentukan lokasi data yang akan ditandatangani.

Wajib: Ya (kecuali disediakan melalui data)

<HASH_FUNCTION>

Menentukan fungsi hash.

Nilai valid:

- sha1
- sha224
- sha256
- sha384
- sha512

Wajib: Ya

<KEY_FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang cocok.

Untuk daftar atribut kunci CloudHSM CLI yang didukung, lihat Atribut kunci untuk CloudHSM CLI.

Wajib: Ya

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan penggunaan kunci dari kunci pribadi lebih besar dari 1.

<DATA_TYPE>

Menentukan apakah nilai parameter data harus di-hash sebagai bagian dari algoritma penandatanganan. Gunakan `raw` untuk data yang tidak di-hash; gunakan `digest` untuk intisari, yang sudah di-hash.

[Untuk RSA-PKCS, data harus diteruskan dalam format DER yang dikodekan seperti yang ditentukan dalam RFC 8017, Bagian 9.2](#)

Nilai valid:

- mentah
- mencerna

Topik terkait

- [Kategori tanda kriptografi di CloudHSM CLI](#)
- [Kategori verifikasi kriptografi di CloudHSM CLI](#)

Hasilkan tanda tangan dengan RSA-PKCS-PSS mekanisme di CloudHSM CLI

Gunakan `crypto sign rsa-pkcs-pss` perintah di CloudHSM CLI untuk menghasilkan tanda tangan menggunakan kunci pribadi RSA dan mekanisme penandatanganan. RSA-PKCS-PSS

Untuk menggunakan `crypto sign rsa-pkcs-pss` perintah, Anda harus terlebih dahulu memiliki kunci pribadi RSA di AWS CloudHSM cluster Anda. Anda dapat menghasilkan kunci pribadi RSA menggunakan [Hasilkan key pair RSA asimetris dengan CloudHSM CLI](#) perintah dengan `sign` atribut yang disetel ke `true`.

Note

Tanda tangan dapat diverifikasi AWS CloudHSM dengan [Kategori verifikasi kriptografi di CloudHSM CLI](#) subperintah.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help crypto sign rsa-pkcs-pss
```

Sign with the RSA-PKCS-PSS mechanism

```
Usage: crypto sign rsa-pkcs-pss [OPTIONS] --key-filter [<KEY_FILTER>...] --
hash-function <HASH_FUNCTION> --mgf <MGF> --salt-length <SALT_LENGTH> <--data-
path <DATA_PATH>|--data <DATA>>
```

Options:

<code>--cluster-id <CLUSTER_ID></code>	Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error
<code>--key-filter [<KEY_FILTER>...]</code>	Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key
<code>--hash-function <HASH_FUNCTION></code>	[possible values: sha1, sha224, sha256, sha384, sha512]
<code>--data-path <DATA_PATH></code>	The path to the file containing the data to be signed
<code>--data <DATA></code>	Base64 Encoded data to be signed
<code>--mgf <MGF></code>	The mask generation function [possible values: mgf1-sha1, mgf1-sha224, mgf1-sha256, mgf1-sha384, mgf1-sha512]
<code>--salt-length <SALT_LENGTH></code>	The salt length
<code>--approval <APPROVAL></code>	Filepath of signed quorum token file to approve operation
<code>--data-type <DATA_TYPE></code>	The type of data passed in, either raw or digest [possible values: raw, digest]
<code>-h, --help</code>	Print help

Contoh

Contoh-contoh ini menunjukkan cara menggunakan crypto sign rsa-pkcs-pss untuk menghasilkan tanda tangan menggunakan mekanisme RSA-PKCS-PSS penandatanganan dan fungsi SHA256 hash. Perintah ini menggunakan kunci pribadi di HSM.

Example Contoh: Menghasilkan tanda tangan untuk data dasar 64 yang dikodekan

```
aws-cloudhsm > crypto sign rsa-pkcs-pss --key-filter attr.label=rsa-private --hash-function sha256 --data YWJjMTIz --salt-length 10 --mgf mgf1-sha256
{
  "error_code": 0,
  "data": {
    "key-reference": "0x000000000007008db",
    "signature": "H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKV0Vm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRT7h/g4o6YERm1tTQL17q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7Yme0Ic1cZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNuds30GAemo0M16asSrEJSthaZWV530BsD0qzA8Rt8JdhXS+GZp3vNLdL10TBELDPweXVgAu4dBX0F0vpw/
gg6sNvuaDK4Y0Bv2fqKg=="
  }
}
```

Example Contoh: Menghasilkan tanda tangan untuk file data

```
aws-cloudhsm > crypto sign rsa-pkcs-pss --key-filter attr.label=rsa-private --hash-function sha256 --data-path data.txt --salt-length 10 --mgf mgf1-sha256
{
  "error_code": 0,
  "data": {
    "key-reference": "0x000000000007008db",
    "signature": "H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKV0Vm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRT7h/g4o6YERm1tTQL17q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7Yme0Ic1cZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNuds30GAemo0M16asSrEJSthaZWV530BsD0qzA8Rt8JdhXS+GZp3vNLdL10TBELDPweXVgAu4dBX0F0vpw/
gg6sNvuaDK4Y0Bv2fqKg=="
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<DATA>

Data yang dikodekan Base64 untuk ditandatangani.

Wajib: Ya (kecuali disediakan melalui jalur data)

<DATA_PATH>

Menentukan lokasi data yang akan ditandatangani.

Wajib: Ya (kecuali disediakan melalui data)

<HASH_FUNCTION>

Menentukan fungsi hash.

Nilai valid:

- sha1
- sha224
- sha256
- sha384
- sha512

Wajib: Ya

<KEY_FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang cocok.

Untuk daftar atribut kunci CloudHSM CLI yang didukung, lihat Atribut kunci untuk CloudHSM CLI.

Wajib: Ya

<MGF>

Menentukan fungsi pembuatan topeng.

 Note

Fungsi hash fungsi pembuatan topeng harus sesuai dengan fungsi hash mekanisme penandatanganan.

Nilai valid:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Wajib: Ya

<SALT_LENGTH>

Menentukan panjang garam.

Wajib: Ya

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan penggunaan kunci dari kunci pribadi lebih besar dari 1.

<DATA_TYPE>

Menentukan apakah nilai parameter data harus di-hash sebagai bagian dari algoritma penandatanganan. Gunakan `raw` untuk data yang tidak di-hash; gunakan `digest` untuk intisari, yang sudah di-hash.

Nilai valid:

- mentah
- mencerna

Topik terkait

- [Kategori tanda kriptografi di CloudHSM CLI](#)

- [Kategori verifikasi kriptografi di CloudHSM CLI](#)

Topik terkait

- [Kategori verifikasi kriptografi di CloudHSM CLI](#)

Kategori verifikasi kriptografi di CloudHSM CLI

Di CloudHSM `crypto verify CLI`, adalah kategori induk untuk sekelompok perintah yang, bila digabungkan dengan kategori induk, mengonfirmasi apakah file telah ditandatangani oleh kunci yang diberikan. `crypto verify` memiliki subperintah berikut:

- [kriptografi verifikasi ecdsa](#)
- [verifikasi kriptografi ed25519ph](#)
- [verifikasi kriptografi rsa-pkcs](#)
- [verifikasi kriptografi rsa-pkcs-pss](#)

`crypto verify` Perintah membandingkan file yang ditandatangani dengan file sumber dan menganalisis apakah mereka terkait secara kriptografi berdasarkan kunci publik dan mekanisme penandatanganan yang diberikan.

Note

File dapat masuk AWS CloudHSM dengan [Kategori tanda kriptografi di CloudHSM CLI](#) operasi.

Verifikasi tanda tangan yang ditandatangani dengan mekanisme ECDSA di CloudHSM CLI

Gunakan `crypto verify ecdsa` perintah di CloudHSM CLI untuk menyelesaikan operasi berikut:

- Konfirmasikan file telah ditandatangani di HSM dengan kunci publik yang diberikan.
- Verifikasi tanda tangan dihasilkan menggunakan mekanisme penandatanganan ECDSA.
- Bandingkan file yang ditandatangani dengan file sumber dan tentukan apakah keduanya terkait secara kriptografi berdasarkan kunci publik ecdsa dan mekanisme penandatanganan yang diberikan.
- Fungsi verifikasi ECDSA mengharapkan tanda tangan dalam format `r | s`, di mana komponen `r` dan `s` digabungkan sebagai data biner mentah.

Untuk menggunakan `crypto verify ecdsa` perintah, Anda harus terlebih dahulu memiliki kunci publik EC di AWS CloudHSM cluster Anda. Anda dapat mengimpor kunci publik EC menggunakan [Impor kunci format PEM dengan CloudHSM CLI](#) perintah dengan `verify` atribut yang disetel ke `true`.

Note

Anda dapat menghasilkan tanda tangan di CloudHSM CLI [Kategori tanda krypto di CloudHSM CLI](#) dengan subperintah.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help crypto verify ecdsa
```

Verify with the ECDSA mechanism

```
Usage: crypto verify ecdsa --key-filter [<KEY_FILTER>...] --hash-  
function <HASH_FUNCTION> <--data-path <DATA_PATH>|--data <DATA>> <--signature-  
path <SIGNATURE_PATH>|--signature <SIGNATURE>>
```

Options:

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

```
--key-filter [<KEY_FILTER>...]
```

Key reference (e.g. `key-reference=0xabc`) or space separated list of key attributes in the form of `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` to select a matching key

```
--hash-function <HASH_FUNCTION>
```

[possible values: `sha1`, `sha224`, `sha256`, `sha384`, `sha512`]

```
--data-path <DATA_PATH>
```

```

    The path to the file containing the data to be verified
--data <DATA>
    Base64 encoded data to be verified
--signature-path <SIGNATURE_PATH>
    The path to where the signature is located
--signature <SIGNATURE>
    Base64 encoded signature to be verified
--data-type <DATA_TYPE>
    The type of data passed in, either raw or digest [possible values: raw,
digest]
-h, --help
    Print help

```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `crypto verify ecdsa` untuk memverifikasi tanda tangan yang dihasilkan menggunakan mekanisme penandatanganan ECDSA dan fungsi SHA256 hash. Perintah ini menggunakan kunci publik di HSM.

Example Contoh: Verifikasi tanda tangan yang disandikan Base64 dengan data yang disandikan Base64

```

aws-cloudhsm > crypto verify ecdsa --hash-function sha256 --key-filter attr.label=ec-
public --data YWJjMTIz --signature 4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2w0S3Q5bzk
+Hc5irV5iTkuxQbropPttVFZ8V6FgR2fz+sPegwCw==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}

```

Example Contoh: Verifikasi file tanda tangan dengan file data

```

aws-cloudhsm > crypto verify ecdsa --hash-function sha256 --key-filter attr.label=ec-
public --data-path data.txt --signature-path signature-file
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}

```

Example Contoh: Buktikan hubungan penandatanganan palsu

Perintah ini memverifikasi apakah data yang terletak di `/home/data` ditandatangani oleh kunci publik dengan label `ecdsa-public` menggunakan mekanisme penandatanganan ECDSA untuk menghasilkan tanda tangan yang terletak di `/home/signature`. Karena argumen yang diberikan tidak membentuk hubungan penandatanganan yang benar, perintah mengembalikan pesan kesalahan.

```
aws-cloudhsm > crypto verify ecdsa --hash-function sha256 --
key-filter attr.label=ec-public --data aW52YWxpZA== --signature
+ogk7M7S3iTqFg3SndJfd91dZFr5Qo6YixJl8JwcvqqVgsVu06o+VKvTRjz0/V05kf3JJbBLr87Q
+wLWcMAJfA==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<DATA>

Data yang dikodekan Base64 untuk ditandatangani.

Wajib: Ya (kecuali disediakan melalui jalur data)

<DATA_PATH>

Menentukan lokasi data yang akan ditandatangani.

Wajib: Ya (kecuali disediakan melalui jalur data)

<HASH_FUNCTION>

Menentukan fungsi hash.

Nilai valid:

- sha1

- sha224
- sha256
- sha384
- sha512

Wajib: Ya

<KEY_FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang cocok.

Untuk daftar atribut kunci CloudHSM CLI yang didukung, lihat Atribut kunci untuk CloudHSM CLI.

Wajib: Ya

<SIGNATURE>

Tanda tangan yang dikodekan Base64.

Wajib: Ya (kecuali disediakan melalui jalur tanda tangan)

<SIGNATURE_PATH>

Menentukan lokasi tanda tangan.

Wajib: Ya (kecuali disediakan melalui jalur tanda tangan)

<DATA_TYPE>

Menentukan apakah nilai parameter data harus di-hash sebagai bagian dari algoritma penandatanganan. Gunakan `raw` untuk data yang tidak di-hash; gunakan `digest` untuk intisari, yang sudah di-hash.

Nilai valid:

- mentah
- mencerna

Topik terkait

- [Kategori tanda kriptografi di CloudHSM CLI](#)

- [Kategori verifikasi krypto di CloudHSM CLI](#)

Verifikasi tanda tangan yang ditandatangani dengan mekanisme HashEd DSA di CloudHSM CLI

 Important

HashEdOperasi verifikasi tanda tangan DSA hanya didukung pada instans hsm2m.medium dalam mode non-FIPS.

Gunakan `crypto verify ed25519ph` perintah di CloudHSM CLI untuk menyelesaikan operasi berikut:

- Verifikasi tanda tangan data atau file menggunakan kunci publik Ed25519 yang diberikan.
- Konfirmasikan tanda tangan dibuat menggunakan mekanisme penandatanganan HashEd DSA. Untuk informasi tambahan tentang HashEd DSA, lihat [NIST SP 186-5](#), Bagian 7.8.

Untuk menggunakan `crypto verify ed25519ph` perintah, Anda harus terlebih dahulu memiliki kunci publik Ed25519 di cluster Anda. AWS CloudHSM Anda dapat membuat key pair Ed25519 menggunakan [Hasilkan key pair EC asimetris dengan CloudHSM CLI](#) perintah dengan curve parameter disetel ke `ed25519` dan `verify` atribut disetel ke `true`, atau mengimpor kunci publik Ed25519 menggunakan [Impor kunci format PEM dengan CloudHSM CLI](#) perintah dengan atribut disetel ke `verify true`

 Note

Anda dapat menghasilkan tanda tangan di CloudHSM CLI [Kategori tanda krypto di CloudHSM CLI](#) dengan subperintah.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

- HashEdOperasi verifikasi tanda tangan DSA hanya didukung pada instans hsm2m.medium dalam mode non-FIPS.

Sintaksis

```
aws-cloudhsm > help crypto verify ed25519ph
```

Verify with the Ed25519ph mechanism

Usage: crypto verify ed25519ph [OPTIONS] --key-filter [<KEY_FILTER>...] --data-type <DATA_TYPE> --hash-function <HASH_FUNCTION> <--data-path <DATA_PATH>|--data <DATA>> <--signature-path <SIGNATURE_PATH>|--signature <SIGNATURE>>

Options:

--cluster-id <CLUSTER_ID>

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

--key-filter [<KEY_FILTER>...]

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key

--data-path <DATA_PATH>

The path to the file containing the data to be verified

--data <DATA>

Base64 encoded data to be verified

--signature-path <SIGNATURE_PATH>

The path to where the signature is located

--signature <SIGNATURE>

Base64 encoded signature to be verified

--data-type <DATA_TYPE>

The type of data passed in, either raw or digest [possible values: raw, digest]

--hash-function <HASH_FUNCTION>

Hash function [possible values: sha512]

-h, --help

Print help

Contoh

Contoh-contoh ini menunjukkan cara menggunakan crypto verify ed25519ph untuk memverifikasi tanda tangan yang dihasilkan menggunakan mekanisme penandatanganan ED25519ph dan fungsi hash. sha512 Perintah ini menggunakan kunci publik Ed25519 di HSM.

Example Contoh: Verifikasi tanda tangan yang disandikan Base64 dengan data yang disandikan Base64

```
aws-cloudhsm > crypto verify ed25519ph \
  --hash-function sha512 \
  --key-filter attr.label=ed25519-public \
  --data-type raw \
  --data YWJj \
  --signature mKcCIvC4Ehqp0w+BPWg/gJ5GK0acf/h20UmbuU5trkEx
+FBCRjwqNVogA9BirfWqoQuMYeY2Biqq0RwqJgg0Bg==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example Contoh: Verifikasi file tanda tangan dengan file data

```
aws-cloudhsm > crypto verify ed25519ph \
  --hash-function sha512 \
  --key-filter attr.label=ed25519-public \
  --data-type raw \
  --data-path data.txt \
  --signature-path signature-file
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<DATA>

Data yang dikodekan Base64 untuk diverifikasi.

Wajib: Ya (kecuali disediakan melalui jalur data)

<DATA_PATH>

Menentukan lokasi data yang akan diverifikasi.

Diperlukan: Ya (kecuali disediakan melalui parameter data)

<HASH_FUNCTION>

Menentukan fungsi hash. Ed25519ph hanya mendukung. SHA512

Nilai valid:

- sha512

Wajib: Ya

<KEY_FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang cocok.

Untuk daftar atribut kunci CloudHSM CLI yang didukung, lihat. [Atribut kunci untuk CloudHSM CLI](#)

Wajib: Ya

<SIGNATURE>

Tanda tangan yang dikodekan Base64.

Wajib: Ya (kecuali disediakan melalui jalur tanda tangan)

<SIGNATURE_PATH>

Menentukan lokasi tanda tangan.

Diperlukan: Ya (kecuali disediakan melalui parameter tanda tangan)

<DATA_TYPE>

Menentukan apakah nilai parameter data harus di-hash sebagai bagian dari algoritma verifikasi. Gunakan `raw` untuk data yang tidak di-hash; gunakan `digest` untuk intisari, yang sudah di-hash.

Nilai valid:

- mentah

- mencerna

Wajib: Ya

Topik terkait

- [Kategori tanda kriptografi di CloudHSM CLI](#)
- [Kategori verifikasi kriptografi di CloudHSM CLI](#)
- [Hasilkan tanda tangan dengan mekanisme HashEd DSA di CloudHSM CLI](#)

Verifikasi tanda tangan yang ditandatangani dengan mekanisme RSA-PKCS di CloudHSM CLI

Gunakan `crypto verify rsa-pkcs` perintah di CloudHSM CLI menyelesaikan operasi berikut:

- Konfirmasikan file telah ditandatangani di HSM dengan kunci publik yang diberikan.
- Verifikasi tanda tangan dibuat menggunakan mekanisme RSA-PKCS penandatanganan.
- Bandingkan file yang ditandatangani dengan file sumber dan tentukan apakah keduanya terkait secara kriptografi berdasarkan kunci publik rsa dan mekanisme penandatanganan yang diberikan.

Untuk menggunakan `crypto verify rsa-pkcs` perintah, Anda harus terlebih dahulu memiliki kunci publik RSA di AWS CloudHSM cluster Anda.

 Note

Anda dapat membuat tanda tangan menggunakan CloudHSM CLI dengan subperintah.

[Kategori tanda kriptografi di CloudHSM CLI](#)

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help crypto verify rsa-pkcs
```

Verify with the RSA-PKCS mechanism

Usage: `crypto verify rsa-pkcs --key-filter [<KEY_FILTER>...] --hash-function <HASH_FUNCTION> <--data-path <DATA_PATH>|--data <DATA>> <--signature-path <SIGNATURE_PATH>|--signature <SIGNATURE>>`

Options:

`--cluster-id <CLUSTER_ID>`

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

`--key-filter [<KEY_FILTER>...]`

Key reference (e.g. `key-reference=0xabc`) or space separated list of key attributes in the form of `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` to select a matching key

`--hash-function <HASH_FUNCTION>`

[possible values: sha1, sha224, sha256, sha384, sha512]

`--data-path <DATA_PATH>`

The path to the file containing the data to be verified

`--data <DATA>`

Base64 encoded data to be verified

`--signature-path <SIGNATURE_PATH>`

The path to where the signature is located

`--signature <SIGNATURE>`

Base64 encoded signature to be verified

`--data-type <DATA_TYPE>`

The type of data passed in, either raw or digest [possible values: raw, digest]

`-h, --help`

Print help

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `crypto verify rsa-pkcs` untuk memverifikasi tanda tangan yang dihasilkan menggunakan mekanisme penandatanganan RSA-PKCS dan fungsi hash. SHA256 Perintah ini menggunakan kunci publik di HSM.

Example Contoh: Verifikasi tanda tangan yang disandikan Base64 dengan data yang disandikan Base64

```
aws-cloudhsm > crypto verify rsa-pkcs --hash-function sha256 --key-filter
attr.label=rsa-public --data YWJjMTIz --signature XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjg0QW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBj0BhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+G09BUT9FFU/
of9cvu40yn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6ev1P7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3IT00qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example Contoh: Verifikasi file tanda tangan dengan file data

```
aws-cloudhsm > crypto verify rsa-pkcs --hash-function sha256 --key-filter
attr.label=rsa-public --data-path data.txt --signature-path signature-file
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example Contoh: Buktikan hubungan penandatanganan palsu

Perintah ini memverifikasi apakah data yang tidak valid ditandatangani oleh kunci publik dengan label `rsa-public` menggunakan mekanisme penandatanganan RSAPKCS untuk menghasilkan tanda tangan yang terletak di `/home/signature` Karena argumen yang diberikan tidak membentuk hubungan penandatanganan yang benar, perintah mengembalikan pesan kesalahan.

```
aws-cloudhsm > crypto verify rsa-pkcs --hash-function sha256 --key-filter
attr.label=rsa-public --data aW52YWxpZA== --signature XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjg0QW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBj0BhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+G09BUT9FFU/
of9cvu40yn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6ev1P7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3IT00qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ==
```

```
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<DATA>

Data yang dikodekan Base64 untuk ditandatangani.

Wajib: Ya (kecuali disediakan melalui jalur data)

<DATA_PATH>

Menentukan lokasi data yang akan ditandatangani.

Wajib: Ya (kecuali disediakan melalui jalur data)

<HASH_FUNCTION>

Menentukan fungsi hash.

Nilai valid:

- sha1
- sha224
- sha256
- sha384
- sha512

Wajib: Ya

<KEY_FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang cocok.

Untuk daftar atribut kunci CloudHSM CLI yang didukung, lihat Atribut kunci untuk CloudHSM CLI.

Wajib: Ya

<SIGNATURE>

Tanda tangan yang dikodekan Base64.

Wajib: Ya (kecuali disediakan melalui jalur tanda tangan)

<SIGNATURE_PATH>

Menentukan lokasi tanda tangan.

Wajib: Ya (kecuali disediakan melalui jalur tanda tangan)

<DATA_TYPE>

Menentukan apakah nilai parameter data harus di-hash sebagai bagian dari algoritma penandatanganan. Gunakan `raw` untuk data yang tidak di-hash; gunakan `digest` untuk intisari, yang sudah di-hash.

[Untuk RSA-PKCS, data harus diteruskan dalam format DER yang dikodekan seperti yang ditentukan dalam RFC 8017, Bagian 9.2](#)

Nilai valid:

- mentah
- mencerna

Topik terkait

- [Kategori tanda kriptografi di CloudHSM CLI](#)
- [Kategori verifikasi kriptografi di CloudHSM CLI](#)

Verifikasi tanda tangan yang ditandatangani dengan RSA-PKCS-PSS mekanisme di CloudHSM CLI

Gunakan `crypto sign rsa-pkcs-pss` perintah di CloudHSM CLI untuk menyelesaikan operasi berikut.

- Konfirmasikan file telah ditandatangani di HSM dengan kunci publik yang diberikan.
- Verifikasi tanda tangan dibuat menggunakan mekanisme RSA-PKCS-PSS penandatanganan.
- Bandingkan file yang ditandatangani dengan file sumber dan tentukan apakah keduanya terkait secara kriptografi berdasarkan kunci publik rsa dan mekanisme penandatanganan yang diberikan.

Untuk menggunakan `crypto verify rsa-pkcs-pss` perintah, Anda harus terlebih dahulu memiliki kunci publik RSA di AWS CloudHSM cluster Anda. Anda dapat mengimpor kunci publik RSA menggunakan perintah `key import pem ADD UNWRAP LINK HERE)` dengan `verify` atribut yang disetel ke `true`

Note

Anda dapat membuat tanda tangan menggunakan CloudHSM CLI dengan subperintah.

[Kategori tanda kriptografi di CloudHSM CLI](#)

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help crypto verify rsa-pkcs-pss
```

Verify with the RSA-PKCS-PSS mechanism

```
Usage: crypto verify rsa-pkcs-pss --key-filter [<KEY_FILTER>...] --hash-  
function <HASH_FUNCTION> --mgf <MGF> --salt-length >SALT_LENGTH< <--data-  
path <DATA_PATH>|--data <DATA> <--signature-path <SIGNATURE_PATH>|--  
signature <SIGNATURE>>
```

Options:

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

```
--key-filter [<KEY_FILTER>...]
```

Key reference (e.g. `key-reference=0xabc`) or space separated list of key attributes in the form of `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` to select a matching key

```
--hash-function <HASH_FUNCTION>
```

[possible values: sha1, sha224, sha256, sha384, sha512]

```

--data-path <DATA_PATH>
    The path to the file containing the data to be verified
--data <DATA>
    Base64 encoded data to be verified
--signature-path <SIGNATURE_PATH>
    The path to where the signature is located
--signature <SIGNATURE>
    Base64 encoded signature to be verified
--data-type <DATA_TYPE>
    The type of data passed in, either raw or digest [possible values: raw,
digest]
--mgf <MGF>
    The mask generation function [possible values: mgf1-sha1, mgf1-sha224, mgf1-
sha256, mgf1-sha384, mgf1-sha512]
--salt-length <SALT_LENGTH>
    The salt length
-h, --help
    Print help

```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `crypto verify rsa-pkcs-pss` untuk memverifikasi tanda tangan yang dihasilkan menggunakan mekanisme RSA-PKCS-PSS penandatanganan dan fungsi SHA256 hash. Perintah ini menggunakan kunci publik di HSM.

Example Contoh: Verifikasi tanda tangan yang disandikan Base64 dengan data yang disandikan Base64

```

aws-cloudhsm > crypto verify rsa-pkcs-pss --key-filter attr.label=rsa-public
--hash-function sha256 --data YWJjMTIz --salt-length 10 --mgf mgf1-sha256
--signature H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKV0Vm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRT7h/g4o6YERm1tTQL17q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7Yme0Ic1cZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemo0M16asSrEJSthaZWV530BsD0qzA8Rt8JdhXS+GZp3vNLdL10TBELDPweXVgAu4dBX0F0vpw/
gg6sNvuaDK4Y0Bv2fqKg==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}

```


Example Contoh: Verifikasi file tanda tangan dengan file data

```
aws-cloudhsm > crypto verify rsa-pkcs-pss --key-filter attr.label=rsa-public --hash-function sha256 --data-path data.txt --salt-length 10 --mgf mgf1-sha256 --signature signature-file
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example Contoh: Buktikan hubungan penandatanganan palsu

Perintah ini memverifikasi apakah data yang tidak valid ditandatangani oleh kunci publik dengan label `rsa-public` menggunakan mekanisme penandatanganan RSAPKCS PSS untuk menghasilkan tanda tangan yang terletak di `/home/signature` Karena argumen yang diberikan tidak membentuk hubungan penandatanganan yang benar, perintah mengembalikan pesan kesalahan.

```
aws-cloudhsm > crypto verify rsa-pkcs-pss --key-filter attr.label=rsa-public --hash-function sha256 --data aW52YWxpZA== --salt-length 10 --mgf mgf1-sha256 --signature H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKV0Vm7ZuyI0fGE4sT/BUN+977mQEV2TqtWpTsiF2IpwGM1VfSBRT7h/g4o6YERm1tTQL17q+AJ7uGGK37zCsWQrAo7Vy8NzPSHxekePo/ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7Yme0Ic1cZkykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN+m4FNUds30GAemo0M16asSrEJSthaZWV530BsD0qzA8Rt8JdhXS+GZp3vNLdL10TBELDPweXVgAu4dBX0F0vpw/gg6sNvuaDK4Y0Bv2fqKg==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<DATA>

Data yang dikodekan Base64 untuk ditandatangani.

Wajib: Ya (kecuali disediakan melalui jalur data)

<DATA_PATH>

Menentukan lokasi data yang akan ditandatangani.

Wajib: Ya (kecuali disediakan melalui jalur data)

<HASH_FUNCTION>

Menentukan fungsi hash.

Nilai valid:

- sha1
- sha224
- sha256
- sha384
- sha512

Wajib: Ya

<KEY_FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang cocok.

Untuk daftar atribut kunci CloudHSM CLI yang didukung, lihat Atribut kunci untuk CloudHSM CLI.

Wajib: Ya

<MFG>

Menentukan fungsi pembuatan topeng.

Note

Fungsi hash fungsi pembuatan topeng harus sesuai dengan fungsi hash mekanisme penandatanganan.

Nilai valid:

- mgf1-sha1

- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Wajib: Ya

<SIGNATURE>

Tanda tangan yang dikodekan Base64.

Wajib: Ya (kecuali disediakan melalui jalur tanda tangan)

<SIGNATURE_PATH>

Menentukan lokasi tanda tangan.

Wajib: Ya (kecuali disediakan melalui jalur tanda tangan)

<DATA_TYPE>

Menentukan apakah nilai parameter data harus di-hash sebagai bagian dari algoritma penandatanganan. Gunakan `raw` untuk data yang tidak di-hash; gunakan `digest` untuk intisari, yang sudah di-hash.

Nilai valid:

- mentah
- mencerna

Topik terkait

- [Kategori tanda krypto di CloudHSM CLI](#)
- [Kategori verifikasi krypto di CloudHSM CLI](#)

Kategori kunci dalam CloudHSM CLI

Dalam CloudHSM key CLI, adalah kategori induk untuk sekelompok perintah yang, ketika dikombinasikan dengan kategori induk, membuat perintah khusus untuk kunci. Saat ini, kategori ini terdiri dari perintah berikut:

- [hapus](#)

- [menghasilkan file](#)
- [kunci generate-asymmetric-pair](#)
 - [kunci generate-asymmetric-pair rsa](#)
 - [kunci generate-asymmetric-pair ec](#)
- [kunci menghasilkan-simetris](#)
 - [kunci menghasilkan aes simetris](#)
 - [kunci menghasilkan rahasia-generik simetris](#)
- [impor pem](#)
- [daftar](#)
- [mereplikasi](#)
- [set-atribut](#)
- [berbagi](#)
- [unshare](#)
- [membuka bungkusnya](#)
- [bungkus](#)

Hapus kunci dengan CloudHSM CLI

Gunakan key delete perintah di CloudHSM CLI untuk menghapus kunci dari cluster. AWS CloudHSM Anda hanya dapat menghapus satu item dalam satu waktu. Menghapus satu kunci pada pasangan kunci tidak berpengaruh pada kunci lainnya pada pasangan.

Hanya CU yang membuat kunci dan akibatnya memilikinya yang dapat menghapus kunci. Pengguna yang berbagi kunci, tetapi tidak memilikinya, dapat menggunakan kunci dalam operasi kriptografi, tetapi tidak dapat menghapusnya.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key delete
Delete a key in the HSM cluster
```

```
Usage: key delete [OPTIONS] --filter [<FILTER>...]
```

Options:

```
--cluster-id <CLUSTER_ID> Unique Id to choose which of the clusters in the
config file to run the operation against. If not provided, will fall back to the value
provided when interactive mode was started, or error
```

```
--filter [<FILTER>...] Key reference (e.g. key-reference=0xabc)
```

or space separated list of key attributes in the form of

```
attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key for deletion
```

```
-h, --help Print help
```

Contoh

```
aws-cloudhsm > key delete --filter attr.label="ec-test-public-key"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<FILTER>

Referensi kunci (misalnya, key-reference=0xabc) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk memilih kunci yang cocok untuk dihapus.

Untuk daftar atribut kunci CloudHSM CLI yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#)

Wajib: Ya

Topik terkait

- [Daftar kunci untuk pengguna dengan CloudHSM CLI](#)
- [Ekspor kunci asimetris dengan CloudHSM CLI](#)
- [Hapus berbagi kunci menggunakan CloudHSM CLI](#)
- [Atribut kunci untuk CloudHSM CLI](#)
- [Tombol filter menggunakan CloudHSM CLI](#)

Ekspor kunci asimetris dengan CloudHSM CLI

Gunakan `key generate-file` perintah di CloudHSM CLI untuk mengekspor kunci asimetris dari modul keamanan perangkat keras (HSM). Jika targetnya adalah kunci pribadi, maka referensi ke kunci pribadi akan diekspor dalam format PEM palsu. Jika targetnya adalah kunci publik, maka byte kunci publik akan diekspor dalam format PEM.

File PEM palsu, yang tidak berisi materi kunci pribadi yang sebenarnya tetapi merujuk kunci pribadi di HSM, dapat digunakan untuk membuat SSL/TLS pembongkaran dari server web Anda ke AWS CloudHSM. Untuk informasi lebih lanjut, lihat pembongkaran [SSL/TLS](#).

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key generate-file
```

```
Generate a key file from a key in the HSM cluster. This command does not export any  
private key data from the HSM
```

```
Usage: key generate-file --encoding <ENCODING> --path <PATH> --filter [<FILTER>...]
```

```
Options:
```

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

`--encoding <ENCODING>`

Encoding format for the key file

Possible values:

- reference-pem: PEM formatted key reference (supports private keys)
- pem: PEM format (supports public keys)

`--path <PATH>`

Filepath where the key file will be written

`--filter [<FILTER>...]`

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key for file generation

`-h, --help`

Print help (see a summary with '-h')

Contoh

Contoh ini menunjukkan cara menggunakan key generate-file untuk menghasilkan file kunci di AWS CloudHSM cluster Anda.

Example

```
aws-cloudhsm > key generate-file --encoding reference-pem --path /tmp/ec-private-
key.pem --filter attr.label="ec-test-private-key"
{
  "error_code": 0,
  "data": {
    "message": "Successfully generated key file"
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang cocok untuk dihapus.

Untuk daftar atribut kunci CloudHSM CLI yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#)

Wajib: Tidak

<ENCODING>

Menentukan format encoding untuk file kunci

Wajib: Ya

<PATH>

Menentukan path file di mana file kunci akan ditulis

Wajib: Ya

Menghasilkan referensi kunci KSP (Windows)

Note

Fitur ini hanya ada di SDK versi 5.16.0 dan yang lebih baru.

Prasyarat

- Anda dapat menghasilkan referensi kunci KSP hanya pada platform Windows.
- Anda harus masuk sebagai pengguna krypto (CU).

Lokasi berkas

Secara default, AWS CloudHSM menyimpan file yang dihasilkan di: `C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP\GlobalPartition`

Untuk menentukan lokasi yang berbeda, gunakan `--path` parameter.

Sintaksis

```
aws-cloudhsm > help key generate-file --encoding ksp-key-reference
```

Generate a key file from a key in the HSM cluster. This command does not export any private key data from the HSM

Usage: key generate-file --encoding **<ENCODING>** --path **<PATH>** --filter [**<FILTER>**...]

Options:

--encoding <ENCODING>

Encoding format for the key file

Possible values:

- reference-pem: PEM formatted key reference (supports private keys)
- pem: PEM format (supports public keys)
- ksp-key-reference: KSP key reference format

--cluster-id <CLUSTER_ID>

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided with multiple clusters configured, will error

--path <PATH>

Directory path where the key file will be written

--filter [<FILTER>...]

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key for file generation

--all

Generate ksp key reference for all available key pairs in HSM

-h, --help

Print help (see a summary with '-h')

Contoh - Menghasilkan referensi kunci KSP menggunakan filter atribut dari kunci pribadi

Contoh berikut menghasilkan referensi kunci KSP untuk kunci pribadi dengan label tertentu.

Example

```
aws-cloudhsm > key generate-file --encoding ksp-key-reference --path --filter  
attr.label="ec-test-private-key"
```

```
{
  "error_code": 0,
  "data": {
    "message": "Successfully generated key file"
  }
}
```

Contoh - Menghasilkan referensi kunci KSP untuk semua pasangan kunci

Contoh berikut menghasilkan referensi kunci KSP untuk semua pasangan kunci di cluster Anda.

Example

```
aws-cloudhsm > key generate-file --encoding ksp-key-reference --all
{
  "error_code": 0,
  "data": {
    "message": "Successfully generated key file"
  }
}
```

Topik terkait

- [Atribut kunci untuk CloudHSM CLI](#)
- [Tombol filter menggunakan CloudHSM CLI](#)
- [generate-asymmetric-pairKategori di CloudHSM CLI](#)
- [Kategori generate-simetris di CloudHSM CLI](#)

generate-asymmetric-pairKategori di CloudHSM CLI

Dalam CloudHSM key generate-asymmetric-pair CLI, adalah kategori induk untuk sekelompok perintah yang, ketika dikombinasikan dengan kategori induk, membuat perintah yang menghasilkan pasangan kunci asimetris. Saat ini, kategori ini terdiri dari perintah berikut:

- [kunci generate-asymmetric-pair ec](#)
- [kunci generate-asymmetric-pair rsa](#)

Hasilkan key pair EC asimetris dengan CloudHSM CLI

Gunakan key asymmetric-pair ec perintah di CloudHSM CLI untuk menghasilkan key pair elliptic-curve (EC) asimetris di cluster Anda. AWS CloudHSM

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key generate-asymmetric-pair ec
```

Generate an Elliptic-Curve Cryptography (ECC) key pair

Usage: key generate-asymmetric-pair ec [OPTIONS] --public-label **<PUBLIC_LABEL>** --private-label **<PRIVATE_LABEL>** --curve **<CURVE>**

Options:

--cluster-id **<CLUSTER_ID>**

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

--public-label **<PUBLIC_LABEL>**

Label for the public key

--private-label **<PRIVATE_LABEL>**

Label for the private key

--session

Creates a session key pair that exists only in the current session. The key cannot be recovered after the session ends

--curve **<CURVE>**

Elliptic curve used to generate the key pair [possible values: prime256v1, secp256r1, secp224r1, secp384r1, secp256k1, secp521r1, ed25519]

--public-attributes [**<PUBLIC_KEY_ATTRIBUTES>**...]

Space separated list of key attributes to set for the generated EC public key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE

--private-attributes [**<PRIVATE_KEY_ATTRIBUTES>**...]

```

    Space separated list of key attributes to set for the generated EC private
    key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
    --share-crypto-users [<SHARE_CRYPTO_USERS>...]
    Space separated list of Crypto User usernames to share the EC private key
    with
    --manage-private-key-quorum-value <MANAGE_PRIVATE_KEY_QUORUM_VALUE>
    The quorum value for key management operations for the private key
    --use-private-key-quorum-value <USE_PRIVATE_KEY_QUORUM_VALUE>
    The quorum value for key usage operations for the private key
    -h, --help
    Print help

```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `key generate-asymmetric-pair ec` perintah untuk membuat key pair EC.

Example Contoh: Membuat key pair EC

```

aws-cloudhsm > key generate-asymmetric-pair ec \
  --curve secp224r1 \
  --public-label ec-public-key-example \
  --private-label ec-private-key-example
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x00000000000012000b",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ec",

```

```

    "label": "ec-public-key-example",
    "id": "",
    "check-value": "0xd7c1a7",
    "class": "public-key",
    "encrypt": false,
    "decrypt": false,
    "token": true,
    "always-sensitive": false,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": false,
    "sign": false,
    "trusted": false,
    "unwrap": false,
    "verify": false,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 57,
    "ec-point":
"0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f38a
    "curve": "secp224r1"
  }
},
"private_key": {
  "key-reference": "0x0000000000012000c",
  "key-info": {
    "key-owners": [
      {
        "username": "cu1",
        "key-coverage": "full"
      }
    ],
    "shared-users": [],
    "key-quorum-values": {
      "manage-key-quorum-value": 0,
      "use-key-quorum-value": 0
    },
    "cluster-coverage": "full"
  },
},
},

```

```

    "attributes": {
      "key-type": "ec",
      "label": "ec-private-key-example",
      "id": "",
      "check-value": "0xd7c1a7",
      "class": "private-key",
      "encrypt": false,
      "decrypt": false,
      "token": true,
      "always-sensitive": true,
      "derive": false,
      "destroyable": true,
      "extractable": true,
      "local": true,
      "modifiable": true,
      "never-extractable": false,
      "private": true,
      "sensitive": true,
      "sign": false,
      "trusted": false,
      "unwrap": false,
      "verify": false,
      "wrap": false,
      "wrap-with-trusted": false,
      "key-length-bytes": 122,
      "ec-point":
"0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f38a
      "curve": "secp224r1"
    }
  }
}
}

```

Example Contoh: Buat key pair EC dengan atribut opsional

```

aws-cloudhsm > key generate-asymmetric-pair ec \
  --curve secp224r1 \
  --public-label ec-public-key-example \
  --private-label ec-private-key-example \
  --public-attributes encrypt=true \
  --private-attributes decrypt=true
{
  "error_code": 0,

```

```

"data": {
  "public_key": {
    "key-reference": "0x000000000002806eb",
    "key-info": {
      "key-owners": [
        {
          "username": "cu1",
          "key-coverage": "full"
        }
      ],
      "shared-users": [],
      "key-quorum-values": {
        "manage-key-quorum-value": 0,
        "use-key-quorum-value": 0
      },
      "cluster-coverage": "full"
    },
    "attributes": {
      "key-type": "ec",
      "label": "ec-public-key-example",
      "id": "",
      "check-value": "0xedef86",
      "class": "public-key",
      "encrypt": true,
      "decrypt": false,
      "token": true,
      "always-sensitive": false,
      "derive": false,
      "destroyable": true,
      "extractable": true,
      "local": true,
      "modifiable": true,
      "never-extractable": false,
      "private": true,
      "sensitive": false,
      "sign": false,
      "trusted": false,
      "unwrap": false,
      "verify": false,
      "wrap": false,
      "wrap-with-trusted": false,
      "key-length-bytes": 57,
      "ec-point":
"0x0487af31882189ec29eddf17a48e8b9cebb075b7b5afc5522fe9c83a029a450cc68592889a1ebf45f32240da514

```

```
    "curve": "secp224r1"
  }
},
"private_key": {
  "key-reference": "0x0000000000280c82",
  "key-info": {
    "key-owners": [
      {
        "username": "cu1",
        "key-coverage": "full"
      }
    ],
    "shared-users": [],
    "key-quorum-values": {
      "manage-key-quorum-value": 0,
      "use-key-quorum-value": 0
    },
    "cluster-coverage": "full"
  },
  "attributes": {
    "key-type": "ec",
    "label": "ec-private-key-example",
    "id": "",
    "check-value": "0xedef86",
    "class": "private-key",
    "encrypt": false,
    "decrypt": true,
    "token": true,
    "always-sensitive": true,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": false,
    "trusted": false,
    "unwrap": false,
    "verify": false,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 122,
```



```

    "ec-point":
      "0x0487af31882189ec29eddf17a48e8b9cebb075b7b5afc5522fe9c83a029a450cc68592889a1ebf45f32240da514"
      "curve": "secp224r1"
    }
  }
}
}

```

Example Contoh: Buat key pair EC dengan nilai kuorum

Saat membuat kunci dengan kontrol kuorum, kunci harus dikaitkan dengan jumlah minimum pengguna yang sama dengan nilai kuorum kunci terbesar. Pengguna terkait termasuk pemilik kunci dan Pengguna Crypto dengan siapa kunci tersebut dibagikan. Untuk menentukan jumlah pengguna minimum untuk berbagi kunci, dapatkan nilai kuorum terbesar antara nilai kuorum penggunaan kunci dan nilai kuorum manajemen kunci dan kurangi 1 untuk memperhitungkan pemilik kunci, yang secara default dikaitkan dengan kunci. Untuk berbagi kunci dengan lebih banyak pengguna, gunakan [Bagikan kunci menggunakan CloudHSM CLI](#) perintah.

```

aws-cloudhsm > key generate-asymmetric-pair ec \
  --curve secp224r1 \
  --public-label ec-public-key-example \
  --private-label ec-private-key-example \
  --public-attributes verify=true \
  --private-attributes sign=true
  --share-crypto-users cu2 cu3 cu4 \
  --manage-private-key-quorum-value 4 \
  --use-private-key-quorum-value 2
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x000000000002806eb",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,

```

```

        "use-key-quorum-value": 0
    },
    "cluster-coverage": "full"
},
"attributes": {
    "key-type": "ec",
    "label": "ec-public-key-example",
    "id": "",
    "check-value": "0xedef86",
    "class": "public-key",
    "encrypt": false,
    "decrypt": false,
    "token": true,
    "always-sensitive": false,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": false,
    "sign": false,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 57,
    "ec-point":
"0x0487af31882189ec29eddf17a48e8b9cebb075b7b5afc5522fe9c83a029a450cc68592889a1ebf45f32240da514
    "curve": "secp224r1"
}
},
"private_key": {
    "key-reference": "0x0000000000280c82",
    "key-info": {
        "key-owners": [
            {
                "username": "cu1",
                "key-coverage": "full"
            }
        ]
    },
    "shared-users": [

```

```
{
  "username": "cu2",
  "key-coverage": "full"
},
{
  "username": "cu3",
  "key-coverage": "full"
},
{
  "username": "cu4",
  "key-coverage": "full"
},
],
"key-quorum-values": {
  "manage-key-quorum-value": 4,
  "use-key-quorum-value": 2
},
"cluster-coverage": "full"
},
"attributes": {
  "key-type": "ec",
  "label": "ec-private-key-example",
  "id": "",
  "check-value": "0xedef86",
  "class": "private-key",
  "encrypt": false,
  "decrypt": false,
  "token": true,
  "always-sensitive": true,
  "derive": false,
  "destroyable": true,
  "extractable": true,
  "local": true,
  "modifiable": true,
  "never-extractable": false,
  "private": true,
  "sensitive": true,
  "sign": true,
  "trusted": false,
  "unwrap": false,
  "verify": false,
  "wrap": false,
  "wrap-with-trusted": false,
  "key-length-bytes": 122,
```

```

    "ec-point":
      "0x0487af31882189ec29eddf17a48e8b9cebb075b7b5afc5522fe9c83a029a450cc68592889a1ebf45f32240da514
      "curve": "secp224r1"
    }
  }
}
}

```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<CURVE>

Menentukan pengenalan untuk kurva elips.

- primer256v1
- secp256r1
- secp224r1
- secp384r1
- secp256k1
- secp521r1
- ed25519 (hanya didukung pada instance hsm2m.medium dalam mode non-FIPS)

Wajib: Ya

<PUBLIC_KEY_ATTRIBUTES>

Menentukan daftar spasi dipisahkan atribut kunci untuk mengatur untuk kunci publik EC dihasilkan dalam bentuk KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (misalnya,verify=true)

Untuk daftar atribut kunci yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#).

Wajib: Tidak

<PUBLIC_LABEL>

Menentukan label yang ditetapkan pengguna untuk public-key. Ukuran maksimum yang diizinkan label adalah 127 karakter untuk Client SDK 5.11 dan setelahnya. Klien SDK 5.10 dan sebelumnya memiliki batas 126 karakter.

Wajib: Ya

<PRIVATE_KEY_ATTRIBUTES>

Menentukan daftar spasi dipisahkan atribut kunci untuk mengatur untuk kunci pribadi EC dihasilkan dalam bentuk KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (misalnya,sign=true)

Untuk daftar atribut kunci yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#).

Wajib: Tidak

<PRIVATE_LABEL>

Menentukan label yang ditetapkan pengguna untuk kunci pribadi. Ukuran maksimum yang diizinkan label adalah 127 karakter untuk Client SDK 5.11 dan setelahnya. Klien SDK 5.10 dan sebelumnya memiliki batas 126 karakter.

Wajib: Ya

<SESSION>

Membuat kunci yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Gunakan parameter ini ketika Anda memerlukan kunci hanya sebentar, seperti kunci pembungkus yang mengenkripsi, dan kemudian dengan cepat mendekripsi, kunci lain. Jangan gunakan kunci sesi untuk mengenkripsi data yang mungkin perlu Anda dekripsi setelah sesi berakhir.

Secara default, kunci yang dihasilkan adalah kunci persisten (token). Melewati <SESSION>perubahan ini, memastikan kunci yang dihasilkan dengan argumen ini adalah kunci sesi (singkat).

Wajib: Tidak

<SHARE_CRYPTO_USERS>

Menentukan daftar nama pengguna Crypto User yang dipisahkan spasi untuk berbagi kunci pribadi EC

Wajib: Tidak

<MANAGE_PRIVATE_KEY_QUORUM_VALUE>

Nilai kuorum untuk operasi manajemen kunci privat. Nilai ini harus kurang dari atau sama dengan jumlah pengguna yang terkait dengan kunci tersebut. Ini termasuk pengguna dengan siapa kunci dibagikan dan pemilik kunci. Nilai maks 8.

Wajib: Tidak

<USE_PRIVATE_KEY_QUORUM_VALUE>

Nilai kuorum untuk operasi penggunaan kunci privat. Nilai ini harus kurang dari atau sama dengan jumlah pengguna yang terkait dengan kunci tersebut. Ini termasuk pengguna dengan siapa kunci dibagikan dan pemilik kunci. Nilai maks 8.

Wajib: Tidak

Topik terkait

- [Atribut kunci untuk CloudHSM CLI](#)
- [Tombol filter menggunakan CloudHSM CLI](#)

Hasilkan key pair RSA asimetris dengan CloudHSM CLI

Gunakan key generate-asymmetric-pair rsa perintah di CloudHSM CLI untuk menghasilkan key pair RSA asimetris di cluster Anda. AWS CloudHSM

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key generate-asymmetric-pair rsa
Generate an RSA key pair

Usage: key generate-asymmetric-pair rsa [OPTIONS] --public-label <PUBLIC_LABEL>
--private-label <PRIVATE_LABEL> --modulus-size-bits <MODULUS_SIZE_BITS> --public-
exponent <PUBLIC_EXPONENT>

Options:
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

`--public-label <PUBLIC_LABEL>`

Label for the public key

`--private-label <PRIVATE_LABEL>`

Label for the private key

`--session`

Creates a session key pair that exists only in the current session. The key cannot be recovered after the session ends

`--modulus-size-bits <MODULUS_SIZE_BITS>`

Modulus size in bits used to generate the RSA key pair

`--public-exponent <PUBLIC_EXPONENT>`

Public exponent used to generate the RSA key pair

`--public-attributes [<PUBLIC_KEY_ATTRIBUTES>...]`

Space separated list of key attributes to set for the generated RSA public key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE

`--private-attributes [<PRIVATE_KEY_ATTRIBUTES>...]`

Space separated list of key attributes to set for the generated RSA private key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE

`--share-crypto-users [<SHARE_CRYPTO_USERS>...]`

Space separated list of Crypto User usernames to share the RSA key with

`--manage-private-key-quorum-value <MANAGE_PRIVATE_KEY_QUORUM_VALUE>`

The quorum value for key management operations for the private key

`--use-private-key-quorum-value <USE_PRIVATE_KEY_QUORUM_VALUE>`

The quorum value for key usage operations for the private key

`-h, --help`

Print help

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `key generate-asymmetric-pair rsa` untuk membuat key pair RSA.

Example Contoh: Membuat key pair RSA

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
--public-exponent 65537 \
--modulus-size-bits 2048 \
--public-label rsa-public-key-example \
--private-label rsa-private-key-example
{
  "error_code": 0,
```

```
"data": {
  "public_key": {
    "key-reference": "0x00000000000160010",
    "key-info": {
      "key-owners": [
        {
          "username": "cu1",
          "key-coverage": "full"
        }
      ],
      "shared-users": [],
      "key-quorum-values": {
        "manage-key-quorum-value": 0,
        "use-key-quorum-value": 0
      },
      "cluster-coverage": "full"
    },
    "attributes": {
      "key-type": "rsa",
      "label": "rsa-public-key-example",
      "id": "",
      "check-value": "0x498e1f",
      "class": "public-key",
      "encrypt": false,
      "decrypt": false,
      "token": true,
      "always-sensitive": false,
      "derive": false,
      "destroyable": true,
      "extractable": true,
      "local": true,
      "modifiable": true,
      "never-extractable": false,
      "private": true,
      "sensitive": false,
      "sign": false,
      "trusted": false,
      "unwrap": false,
      "verify": false,
      "wrap": false,
      "wrap-with-trusted": false,
      "key-length-bytes": 512,
      "public-exponent": "0x010001",
```



```

    "modulus":
      "0xdfca0669dc8288ed3bad99509bd21c7e6192661407021b3f4cdf4a593d939dd24f4d641af8e4e73b04c847731c6
      e89a065e7d1a46ced96b46b909db2ab6be871ee700fd0a448b6e975bb64cae77c49008749212463e37a577baa57ce3e
      bcebb7d20bd6df1948ae336ae23b52d73b7f3b6acc2543edb6358e08d326d280ce489571f4d34e316a2ea1904d513ca
    "modulus-size-bits": 2048
  },
  "private_key": {
    "key-reference": "0x00000000000160011",
    "key-info": {
      "key-owners": [
        {
          "username": "cu1",
          "key-coverage": "full"
        }
      ],
      "shared-users": [],
      "key-quorum-values": {
        "manage-key-quorum-value": 0,
        "use-key-quorum-value": 0
      },
      "cluster-coverage": "full"
    },
    "attributes": {
      "key-type": "rsa",
      "label": "rsa-private-key-example",
      "id": "",
      "check-value": "0x498e1f",
      "class": "private-key",
      "encrypt": false,
      "decrypt": false,
      "token": true,
      "always-sensitive": true,
      "derive": false,
      "destroyable": true,
      "extractable": true,
      "local": true,
      "modifiable": true,
      "never-extractable": false,
      "private": true,
      "sensitive": true,
      "sign": false,
      "trusted": false,
      "unwrap": false,

```

```

        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1217,
        "public-exponent": "0x010001",
        "modulus":
"0xdfca0669dc8288ed3bad99509bd21c7e6192661407021b3f4cdf4a593d939dd24f4d641af8e4e73b04c847731c6
        "modulus-size-bits": 2048
    }
}
}
}
}

```

Example Contoh: Buat key pair RSA dengan atribut opsional

```

aws-cloudhsm > key generate-asymmetric-pair rsa \
--public-exponent 65537 \
--modulus-size-bits 2048 \
--public-label rsa-public-key-example \
--private-label rsa-private-key-example \
--public-attributes encrypt=true \
--private-attributes decrypt=true
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x00000000000280cc8",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",

```

```

    "label": "rsa-public-key-example",
    "id": "",
    "check-value": "0x01fe6e",
    "class": "public-key",
    "encrypt": true,
    "decrypt": false,
    "token": true,
    "always-sensitive": false,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": false,
    "sign": false,
    "trusted": false,
    "unwrap": false,
    "verify": false,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 512,
    "public-exponent": "0x010001",
    "modulus":
      "0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1d4
      73a80fdb457aa7b20cd61e486c326e2cfd5e124a7f6a996437437812b542e3caf85928aa866f0298580f7967ee6aa01
      f6e6296d6c116d5744c6d60d14d3bf3cb978fe6b75ac67b7089bafd50d8687213b31abc7dc1bad422780d29c851d510
      133022653225bd129f8491101725e9ea33e1ded83fb57af35f847e532eb30cd7e726f23910d2671c6364092e834697e
      ac3160f0ca9725d38318b7",
    "modulus-size-bits": 2048
  },
  "private_key": {
    "key-reference": "0x00000000000280cc7",
    "key-info": {
      "key-owners": [
        {
          "username": "cu1",
          "key-coverage": "full"
        }
      ],
      "shared-users": [],
      "key-quorum-values": {

```

```

        "manage-key-quorum-value": 0,
        "use-key-quorum-value": 0
    },
    "cluster-coverage": "full"
},
"attributes": {
    "key-type": "rsa",
    "label": "rsa-private-key-example",
    "id": "",
    "check-value": "0x01fe6e",
    "class": "private-key",
    "encrypt": false,
    "decrypt": true,
    "token": true,
    "always-sensitive": true,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": false,
    "trusted": false,
    "unwrap": false,
    "verify": false,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 1217,
    "public-exponent": "0x010001",
    "modulus":
"0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1d4
    "modulus-size-bits": 2048
}
}
}
}

```

Example Contoh: Buat key pair RSA dengan nilai kuorum

Saat membuat kunci dengan kontrol kuorum, kunci harus dikaitkan dengan jumlah minimum pengguna yang sama dengan nilai kuorum kunci terbesar. Pengguna terkait termasuk pemilik kunci

dan Pengguna Crypto dengan siapa kunci tersebut dibagikan. Untuk menentukan jumlah pengguna minimum untuk berbagi kunci, dapatkan nilai kuorum terbesar antara nilai kuorum penggunaan kunci dan nilai kuorum manajemen kunci dan kurangi 1 untuk memperhitungkan pemilik kunci, yang secara default dikaitkan dengan kunci. Untuk berbagi kunci dengan lebih banyak pengguna, gunakan [Bagikan kunci menggunakan CloudHSM CLI](#) perintah.

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
--public-exponent 65537 \
--modulus-size-bits 2048 \
--public-label rsa-public-key-example \
--private-label rsa-private-key-example \
--public-attributes verify=true \
--private-attributes sign=true \
--share-crypto-users cu2 cu3 cu4 \
--manage-private-key-quorum-value 4 \
--use-private-key-quorum-value 2
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x000000000000280cc8",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-public-key-example",
        "id": "",
        "check-value": "0x01fe6e",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
```

```

    "token": true,
    "always-sensitive": false,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": false,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 512,
    "public-exponent": "0x010001",
    "modulus":
      "0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1d4
      73a80fdb457aa7b20cd61e486c326e2cfd5e124a7f6a996437437812b542e3caf85928aa866f0298580f7967ee6aa01
      f6e6296d6c116d5744c6d60d14d3bf3cb978fe6b75ac67b7089bafd50d8687213b31abc7dc1bad422780d29c851d510
      133022653225bd129f8491101725e9ea33e1ded83fb57af35f847e532eb30cd7e726f23910d2671c6364092e834697e
      ac3160f0ca9725d38318b7",
    "modulus-size-bits": 2048
  }
},
"private_key": {
  "key-reference": "0x00000000000280cc7",
  "key-info": {
    "key-owners": [
      {
        "username": "cu1",
        "key-coverage": "full"
      }
    ],
    "shared-users": [
      {
        "username": "cu2",
        "key-coverage": "full"
      },
      {
        "username": "cu3",
        "key-coverage": "full"
      }
    ]
  }
}

```

```

    },
    {
      "username": "cu4",
      "key-coverage": "full"
    },
  ],
  "key-quorum-values": {
    "manage-key-quorum-value": 4,
    "use-key-quorum-value": 2
  },
  "cluster-coverage": "full"
},
"attributes": {
  "key-type": "rsa",
  "label": "rsa-private-key-example",
  "id": "",
  "check-value": "0x01fe6e",
  "class": "private-key",
  "encrypt": false,
  "decrypt": false,
  "token": true,
  "always-sensitive": true,
  "derive": false,
  "destroyable": true,
  "extractable": true,
  "local": true,
  "modifiable": true,
  "never-extractable": false,
  "private": true,
  "sensitive": true,
  "sign": true,
  "trusted": false,
  "unwrap": false,
  "verify": false,
  "wrap": false,
  "wrap-with-trusted": false,
  "key-length-bytes": 1217,
  "public-exponent": "0x010001",
  "modulus":
"0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1d4",
  "modulus-size-bits": 2048
}
}
}

```

```
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<MODULUS_SIZE_BITS>

Menentukan panjang modulus dalam bit. Nilai minimum adalah 2048.

Wajib: Ya

<PRIVATE_KEY_ATTRIBUTES>

Menentukan daftar spasi dipisahkan atribut kunci untuk mengatur untuk kunci pribadi RSA dihasilkan dalam bentuk KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (misalnya,) sign=true

Untuk daftar atribut kunci yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#).

Wajib: Tidak

<PRIVATE_LABEL>

Menentukan label yang ditetapkan pengguna untuk kunci pribadi. Ukuran maksimum yang diizinkan label adalah 127 karakter untuk Client SDK 5.11 dan setelahnya. Klien SDK 5.10 dan sebelumnya memiliki batas 126 karakter.

Wajib: Ya

<PUBLIC_EXPONENT>

Menentukan eksponen publik. Nilai harus angka ganjil yang lebih besar dari atau sama dengan 65537.

Wajib: Ya

<PUBLIC_KEY_ATTRIBUTES>

Menentukan daftar spasi dipisahkan atribut kunci untuk mengatur untuk kunci publik RSA dihasilkan dalam bentuk KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (misalnya,) verify=true

Untuk daftar atribut kunci yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#).

Wajib: Tidak

<PUBLIC_LABEL>

Menentukan label yang ditetapkan pengguna untuk public-key. Ukuran maksimum yang diizinkan label adalah 127 karakter untuk Client SDK 5.11 dan setelahnya. Klien SDK 5.10 dan sebelumnya memiliki batas 126 karakter.

Wajib: Ya

<SESSION>

Membuat kunci yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Gunakan parameter ini ketika Anda memerlukan kunci hanya sebentar, seperti kunci pembungkus yang mengenkripsi, dan kemudian dengan cepat mendekripsi, kunci lain. Jangan gunakan kunci sesi untuk mengenkripsi data yang mungkin perlu Anda dekripsi setelah sesi berakhir.

Secara default, kunci yang dihasilkan adalah kunci persisten (token). Melewati <SESSION>perubahan ini, memastikan kunci yang dihasilkan dengan argumen ini adalah kunci sesi (singkat).

Wajib: Tidak

<SHARE_CRYPTO_USERS>

Menentukan daftar nama pengguna Crypto User yang dipisahkan spasi untuk berbagi kunci pribadi RSA

Wajib: Tidak

<MANAGE_PRIVATE_KEY_QUORUM_VALUE>

Nilai kuorum untuk operasi manajemen kunci privat. Nilai ini harus kurang dari atau sama dengan jumlah pengguna yang terkait dengan kunci tersebut. Ini termasuk pengguna dengan siapa kunci dibagikan dan pemilik kunci. Nilai maks 8.

Wajib: Tidak

<USE_PRIVATE_KEY_QUORUM_VALUE>

Nilai kuorum untuk operasi penggunaan kunci privat. Nilai ini harus kurang dari atau sama dengan jumlah pengguna yang terkait dengan kunci tersebut. Ini termasuk pengguna dengan siapa kunci dibagikan dan pemilik kunci. Nilai maks 8.

Wajib: Tidak

Topik terkait

- [Atribut kunci untuk CloudHSM CLI](#)
- [Tombol filter menggunakan CloudHSM CLI](#)

Kategori generate-simetris di CloudHSM CLI

Dalam CloudHSM key generate-symmetric CLI, adalah kategori induk untuk sekelompok perintah yang, ketika dikombinasikan dengan kategori induk, membuat perintah yang menghasilkan kunci simetris. Saat ini, kategori ini terdiri dari perintah berikut:

- [kunci menghasilkan aes simetris](#)
- [kunci menghasilkan rahasia-generik simetris](#)

Hasilkan kunci AES simetris dengan CloudHSM CLI

Gunakan key generate-symmetric aes perintah di CloudHSM CLI untuk menghasilkan kunci AES simetris di cluster Anda. AWS CloudHSM

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key generate-symmetric aes
```

```
Generate an AES key
```

```
Usage: key generate-symmetric aes [OPTIONS] --label <LABEL> --key-length-  
bytes <KEY_LENGTH_BYTES>
```

Options:

```

--cluster-id <CLUSTER_ID>
    Unique Id to choose which of the clusters in the config file to run the
    operation against. If not provided, will fall back to the value provided when
    interactive mode was started, or error
--label <LABEL>
    Label for the key
--session
    Creates a session key that exists only in the current session. The key cannot
    be recovered after the session ends
--key-length-bytes <KEY_LENGTH_BYTES>
    Key length in bytes
--attributes [<KEY_ATTRIBUTES>...]
    Space separated list of key attributes to set for the generated AES key in
    the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
--share-crypto-users [<SHARE_CRYPTO_USERS>...]
    Space separated list of Crypto User usernames to share the AES key with
--manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE>
    The quorum value for key management operations
--use-key-quorum-value <USE_KEY_QUORUM_VALUE>
    The quorum value for key usage operations
-h, --help
    Print help

```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan key generate-symmetric aes perintah untuk membuat kunci AES.

Example Contoh: Buat kunci AES

```

aws-cloudhsm > key generate-symmetric aes \
--label example-aes \
--key-length-bytes 24
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000002e06bf",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",

```

```

        "key-coverage": "full"
    },
    ],
    "shared-users": [],
    "key-quorum-values": {
        "manage-key-quorum-value": 0,
        "use-key-quorum-value": 0
    },
    "cluster-coverage": "full"
},
"attributes": {
    "key-type": "aes",
    "label": "example-aes",
    "id": "",
    "check-value": "0x9b94bd",
    "class": "secret-key",
    "encrypt": false,
    "decrypt": false,
    "token": true,
    "always-sensitive": true,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 24
}
}
}
}

```

Example Contoh: Buat kunci AES dengan atribut opsional

```
aws-cloudhsm > key generate-symmetric aes \
```

```
--label example-aes \  
--key-length-bytes 24 \  
--attributes decrypt=true encrypt=true  
{  
  "error_code": 0,  
  "data": {  
    "key": {  
      "key-reference": "0x000000000002e06bf",  
      "key-info": {  
        "key-owners": [  
          {  
            "username": "cu1",  
            "key-coverage": "full"  
          }  
        ],  
        "shared-users": [],  
        "key-quorum-values": {  
          "manage-key-quorum-value": 0,  
          "use-key-quorum-value": 0  
        },  
        "cluster-coverage": "full"  
      },  
      "attributes": {  
        "key-type": "aes",  
        "label": "example-aes",  
        "id": "",  
        "check-value": "0x9b94bd",  
        "class": "secret-key",  
        "encrypt": true,  
        "decrypt": true,  
        "token": true,  
        "always-sensitive": true,  
        "derive": false,  
        "destroyable": true,  
        "extractable": true,  
        "local": true,  
        "modifiable": true,  
        "never-extractable": false,  
        "private": true,  
        "sensitive": true,  
        "sign": true,  
        "trusted": false,  
        "unwrap": false,  
        "verify": true,
```

```

        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 24
    }
}
}
}

```

Example Contoh: Buat kunci AES dengan nilai kuorum

Saat membuat kunci dengan kontrol kuorum, kunci harus dikaitkan dengan jumlah minimum pengguna yang sama dengan nilai kuorum kunci terbesar. Pengguna terkait termasuk pemilik kunci dan Pengguna Crypto dengan siapa kunci tersebut dibagikan. Untuk menentukan jumlah pengguna minimum untuk berbagi kunci, dapatkan nilai kuorum terbesar antara nilai kuorum penggunaan kunci dan nilai kuorum manajemen kunci dan kurangi 1 untuk memperhitungkan pemilik kunci, yang secara default dikaitkan dengan kunci. Untuk berbagi kunci dengan lebih banyak pengguna, gunakan [Bagikan kunci menggunakan CloudHSM CLI](#) perintah.

```

aws-cloudhsm > key generate-symmetric aes \
--label example-aes \
--key-length-bytes 24 \
--attributes decrypt=true encrypt=true
--share-crypto-users cu2 cu3 cu4 \
--manage-key-quorum-value 4 \
--use-key-quorum-value 2
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000002e06bf",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [
          {
            "username": "cu2",
            "key-coverage": "full"
          }
        ]
      }
    }
  }
}

```

```
{
  "username": "cu3",
  "key-coverage": "full"
},
{
  "username": "cu4",
  "key-coverage": "full"
},
],
"key-quorum-values": {
  "manage-key-quorum-value": 4,
  "use-key-quorum-value": 2
},
"cluster-coverage": "full"
},
"attributes": {
  "key-type": "aes",
  "label": "example-aes",
  "id": "",
  "check-value": "0x9b94bd",
  "class": "secret-key",
  "encrypt": true,
  "decrypt": true,
  "token": true,
  "always-sensitive": true,
  "derive": false,
  "destroyable": true,
  "extractable": true,
  "local": true,
  "modifiable": true,
  "never-extractable": false,
  "private": true,
  "sensitive": true,
  "sign": true,
  "trusted": false,
  "unwrap": false,
  "verify": true,
  "wrap": false,
  "wrap-with-trusted": false,
  "key-length-bytes": 24
}
}
}
```

```
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Diperlukan: Jika beberapa cluster telah [dikonfigurasi](#).

<KEY_ATTRIBUTES>

Menentukan daftar spasi dipisahkan atribut kunci untuk mengatur kunci AES dihasilkan dalam bentuk KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (misalnya,sign=true).

Untuk daftar atribut kunci yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#).

Wajib: Tidak

<KEY-LENGTH-BYTES>

Menentukan panjang kunci dalam byte.

Nilai valid:

- 16, 24, dan 32

Wajib: Ya

<LABEL>

Menentukan label yang ditetapkan pengguna untuk kunci AES. Ukuran maksimum yang diizinkan label adalah 127 karakter untuk Client SDK 5.11 dan setelahnya. Klien SDK 5.10 dan sebelumnya memiliki batas 126 karakter.

Wajib: Ya

<SESSION>

Membuat kunci yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Gunakan parameter ini ketika Anda memerlukan kunci hanya sebentar, seperti kunci pembungkus yang mengenkripsi, dan kemudian dengan cepat mendekripsi, kunci lain. Jangan gunakan kunci sesi untuk mengenkripsi data yang mungkin perlu Anda dekripsi setelah sesi berakhir.

Secara default, kunci yang dihasilkan adalah kunci persisten (token). Melewati <SESSION>perubahan ini, memastikan kunci yang dihasilkan dengan argumen ini adalah kunci sesi (singkat).

Wajib: Tidak

<SHARE_CRYPTO_USERS>

Menentukan daftar nama pengguna Crypto User yang dipisahkan spasi untuk berbagi kunci AES dengan

Wajib: Tidak

<MANAGE_KEY_QUORUM_VALUE>

Nilai kuorum untuk operasi manajemen kunci. Nilai ini harus kurang dari atau sama dengan jumlah pengguna yang terkait dengan kunci tersebut. Ini termasuk pengguna dengan siapa kunci dibagikan dan pemilik kunci. Nilai maks 8.

Wajib: Tidak

<USE_KEY_QUORUM_VALUE>

Nilai kuorum untuk operasi penggunaan kunci. Nilai ini harus kurang dari atau sama dengan jumlah pengguna yang terkait dengan kunci tersebut. Ini termasuk pengguna dengan siapa kunci dibagikan dan pemilik kunci. Nilai maks 8.

Wajib: Tidak

Topik terkait

- [Atribut kunci untuk CloudHSM CLI](#)
- [Tombol filter menggunakan CloudHSM CLI](#)

Hasilkan kunci Rahasia Generik simetris dengan CloudHSM CLI

Gunakan key generate-symmetric generic-secret perintah di CloudHSM CLI untuk menghasilkan kunci Rahasia Generik simetris di cluster Anda. AWS CloudHSM

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > key help generate-symmetric generic-secret
```

Generate a generic secret key

Usage: key generate-symmetric generic-secret [OPTIONS] --label *<LABEL>* --key-length-bytes *<KEY_LENGTH_BYTES>*

Options:

--cluster-id *<CLUSTER_ID>*

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

--label *<LABEL>*

Label for the key

--session

Creates a session key that exists only in the current session. The key cannot be recovered after the session ends

--key-length-bytes *<KEY_LENGTH_BYTES>*

Key length in bytes

--attributes [*<KEY_ATTRIBUTES>*...]

Space separated list of key attributes to set for the generated generic secret key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE

--share-crypto-users [*<SHARE_CRYPTO_USERS>*...]

Space separated list of Crypto User usernames to share the generic secret key with

--manage-key-quorum-value *<MANAGE_KEY_QUORUM_VALUE>*

The quorum value for key management operations

--use-key-quorum-value *<USE_KEY_QUORUM_VALUE>*

The quorum value for key usage operations

-h, --help

Print help

Contoh

Contoh-contoh ini menunjukkan cara menggunakan key generate-symmetric generic-secret perintah untuk membuat kunci rahasia generik.

Example Contoh: Buat kunci rahasia generik

```
aws-cloudhsm > key generate-symmetric generic-secret \  
--label example-generic-secret \  
--key-length-bytes 256  
{  
  "error_code": 0,  
  "data": {  
    "key": {  
      "key-reference": "0x000000000002e08fd",  
      "key-info": {  
        "key-owners": [  
          {  
            "username": "cu1",  
            "key-coverage": "full"  
          }  
        ],  
        "shared-users": [],  
        "key-quorum-values": {  
          "manage-key-quorum-value": 0,  
          "use-key-quorum-value": 0  
        },  
        "cluster-coverage": "full"  
      },  
      "attributes": {  
        "key-type": "generic-secret",  
        "label": "example-generic-secret",  
        "id": "",  
        "class": "secret-key",  
        "encrypt": false,  
        "decrypt": false,  
        "token": true,  
        "always-sensitive": true,  
        "derive": false,  
        "destroyable": true,  
        "extractable": true,  
        "local": true,  
        "modifiable": true,  
        "never-extractable": false,  
        "private": true,  
        "sensitive": true,  
        "sign": true,  
        "trusted": false,  
        "unwrap": false,
```

```

    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 256
  }
}
}
}

```

Example Contoh: Buat kunci rahasia generik dengan atribut opsional

```

aws-cloudhsm > key generate-symmetric generic-secret \
--label example-generic-secret \
--key-length-bytes 256 \
--attributes encrypt=true
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000002e08fd",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "generic-secret",
        "label": "example-generic-secret",
        "id": "",
        "class": "secret-key",
        "encrypt": true,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,

```

```

    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 256
  }
}
}
}

```

Example Contoh: Buat kunci rahasia generik dengan nilai kuorum

Saat membuat kunci dengan kontrol kuorum, kunci harus dikaitkan dengan jumlah minimum pengguna yang sama dengan nilai kuorum kunci terbesar. Pengguna terkait termasuk pemilik kunci dan Pengguna Crypto dengan siapa kunci tersebut dibagikan. Untuk menentukan jumlah pengguna minimum untuk berbagi kunci, dapatkan nilai kuorum terbesar antara nilai kuorum penggunaan kunci dan nilai kuorum manajemen kunci dan kurangi 1 untuk memperhitungkan pemilik kunci, yang secara default dikaitkan dengan kunci. Untuk berbagi kunci dengan lebih banyak pengguna, gunakan [Bagikan kunci menggunakan CloudHSM CLI](#) perintah.

```

aws-cloudhsm > key generate-symmetric generic-secret \
--label example-generic-secret \
--key-length-bytes 256 \
--attributes encrypt=true
--share-crypto-users cu2 cu3 cu4 \
--manage-key-quorum-value 4 \
--use-key-quorum-value 2
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000002e08fd",

```

```
"key-info": {
  "key-owners": [
    {
      "username": "cu1",
      "key-coverage": "full"
    }
  ],
  "shared-users": [
    {
      "username": "cu2",
      "key-coverage": "full"
    },
    {
      "username": "cu3",
      "key-coverage": "full"
    },
    {
      "username": "cu4",
      "key-coverage": "full"
    }
  ],
  "key-quorum-values": {
    "manage-key-quorum-value": 4,
    "use-key-quorum-value": 2
  },
  "cluster-coverage": "full"
},
"attributes": {
  "key-type": "generic-secret",
  "label": "example-generic-secret",
  "id": "",
  "class": "secret-key",
  "encrypt": true,
  "decrypt": false,
  "token": true,
  "always-sensitive": true,
  "derive": false,
  "destroyable": true,
  "extractable": true,
  "local": true,
  "modifiable": true,
  "never-extractable": false,
  "private": true,
  "sensitive": true,
```

```
    "sign": true,  
    "trusted": false,  
    "unwrap": false,  
    "verify": true,  
    "wrap": false,  
    "wrap-with-trusted": false,  
    "key-length-bytes": 256  
  }  
}  
}
```

Argumen

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<KEY_ATTRIBUTES>

Menentukan daftar spasi dipisahkan atribut kunci untuk mengatur kunci AES yang dihasilkan dalam bentuk KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (misalnya,sign=true).

Untuk daftar atribut kunci yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#).

Wajib: Tidak

<KEY-LENGTH-BYTES>

Menentukan panjang kunci dalam byte.

Nilai valid:

- 1 hingga 800

Wajib: Ya

<LABEL>

Menentukan label yang ditetapkan pengguna untuk kunci rahasia generik. Ukuran maksimum yang diizinkan label adalah 127 karakter untuk Client SDK 5.11 dan seterusnya. Klien SDK 5.10 dan sebelumnya memiliki batas 126 karakter.

Wajib: Ya

<SESSION>

Membuat kunci yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Gunakan parameter ini ketika Anda memerlukan kunci hanya sebentar, seperti kunci pembungkus yang mengenkripsi, dan kemudian dengan cepat mendekripsi, kunci lain. Jangan gunakan kunci sesi untuk mengenkripsi data yang mungkin perlu Anda dekripsi setelah sesi berakhir.

Secara default, kunci yang dihasilkan adalah kunci persisten (token). Melewati <SESSION>perubahan ini, memastikan kunci yang dihasilkan dengan argumen ini adalah kunci sesi (singkat).

Wajib: Tidak

<SHARE_CRYPTO_USERS>

Daftar nama pengguna Crypto User yang dipisahkan ruang untuk berbagi kunci rahasia umum dengan

Wajib: Tidak

<MANAGE_KEY_QUORUM_VALUE>

Nilai kuorum untuk operasi manajemen kunci. Nilai ini harus kurang dari atau sama dengan jumlah pengguna yang terkait dengan kunci tersebut. Ini termasuk pengguna dengan siapa kunci dibagikan dan pemilik kunci. Nilai maks 8.

Wajib: Tidak

<USE_KEY_QUORUM_VALUE>

Nilai kuorum untuk operasi penggunaan kunci. Nilai ini harus kurang dari atau sama dengan jumlah pengguna yang terkait dengan kunci tersebut. Ini termasuk pengguna dengan siapa kunci dibagikan dan pemilik kunci. Nilai maks 8.

Wajib: Tidak

Topik terkait

- [Atribut kunci untuk CloudHSM CLI](#)
- [Tombol filter menggunakan CloudHSM CLI](#)

Impor kunci format PEM dengan CloudHSM CLI

Gunakan `key import pem` perintah AWS CloudHSM untuk mengimpor kunci format PEM ke modul keamanan perangkat keras (HSM). Anda dapat menggunakannya untuk mengimpor kunci privat yang dihasilkan di luar HSM.

Note

Gunakan [Ekspor kunci asimetris dengan CloudHSM CLI](#) perintah untuk membuat file PEM standar dari kunci publik atau untuk membuat file PEM referensi dari kunci pribadi.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key import pem
```

```
Import key from a PEM file
```

```
Usage: key import pem [OPTIONS] --path <PATH> --label <LABEL> --key-type-  
class <KEY_TYPE_CLASS>
```

```
Options:
```

```
    --cluster-id <CLUSTER_ID>
```

```
        Unique Id to choose which of the clusters in the config file to run the  
operation against. If not provided, will fall back to the value provided when  
interactive mode was started, or error
```

```
    --path <PATH>
```

```
        Path where the key is located in PEM format
```

```
    --label <LABEL>
```

```
        Label for the imported key
```

```
    --key-type-class <KEY_TYPE_CLASS>
```

```
        Key type and class of the imported key [possible values: ec-public, rsa-  
public]
```

```
--attributes [<IMPORT_KEY_ATTRIBUTES>...]
```

Space separated list of key attributes in the form of
KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the imported key

```
-h, --help
```

Print help

Contoh

Contoh ini menunjukkan bagaimana menggunakan `key import pem` perintah untuk mengimpor kunci publik RSA dari file dalam format PEM.

Example Contoh: Impor kunci publik RSA

```
aws-cloudhsm > key import pem --path /home/example --label example-imported-key --key-
type-class rsa-public
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000001e08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "example-imported-key",
        "id": "0x",
        "check-value": "0x99fe93",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
```

```

    "always-sensitive": false,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": false,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": false,
    "sign": false,
    "trusted": false,
    "unwrap": false,
    "verify": false,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 512,
    "public-exponent": "0x010001",
    "modulus":
"0x8e9c172c37aa22ed1ce25f7c3a7c936dadcd532201400128b044ebb4b96#..3e4930ab910df5a2896eae8853cfe
    "modulus-size-bits": 2048
  }
},
  "message": "Successfully imported key"
}
}

```

Example Contoh: Impor kunci publik RSA dengan atribut opsional

```

aws-cloudhsm > key import pem --path /home/example --label example-imported-key-with-
attributes --key-type-class rsa-public --attributes verify=true
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000001e08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],

```

```

    "key-quorum-values": {
      "manage-key-quorum-value": 0,
      "use-key-quorum-value": 0
    },
    "cluster-coverage": "full"
  },
  "attributes": {
    "key-type": "rsa",
    "label": "example-imported-key-with-attributes",
    "id": "0x",
    "check-value": "0x99fe93",
    "class": "public-key",
    "encrypt": false,
    "decrypt": false,
    "token": true,
    "always-sensitive": false,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": false,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": false,
    "sign": false,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 512,
    "public-exponent": "0x010001",
    "modulus":
      "0x8e9c172c37aa22ed1ce25f7c3a7c936dad532201400128b044ebb4b96#..3e4930ab910df5a2896eae8853cfe",
    "modulus-size-bits": 2048
  }
},
"message": "Successfully imported key"
}
}

```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Diperlukan: Jika beberapa cluster telah [dikonfigurasi](#).

<PATH>

Menentukan path file di mana file kunci berada.

Wajib: Ya

<LABEL>

Menentukan label yang ditetapkan pengguna untuk kunci yang diimpor. Ukuran maksimum yang diijinkan label adalah 126 karakter.

Wajib: Ya

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus.

Kemungkinan nilai:

- ec-publik
- rsa-publik

Wajib: Ya

<IMPORT_KEY_ATTRIBUTES>

Menentukan daftar spasi dipisahkan atribut kunci untuk mengatur untuk kunci diimpor dalam bentuk KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (misalnya,sign=true). Untuk daftar atribut kunci yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#).

Wajib: Tidak

Topik terkait

- [Kategori tanda kriptografi di CloudHSM CLI](#)
- [Kategori verifikasi kriptografi di CloudHSM CLI](#)

Daftar kunci untuk pengguna dengan CloudHSM CLI

Gunakan key list perintah di CloudHSM CLI untuk menemukan semua kunci untuk pengguna saat ini yang ada di cluster Anda. AWS CloudHSM Outputnya mencakup kunci yang dimiliki dan dibagikan pengguna, serta semua kunci publik di klaster CloudHSM.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Admin () COs
- Pengguna Crypto (CUs)

Sintaksis

```
aws-cloudhsm > help key list
```

List the keys the current user owns, shares, and all public keys in the HSM cluster

Usage: key list [OPTIONS]

Options:

--cluster-id *<CLUSTER_ID>*

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

--filter [*<FILTER>...*]

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select matching key(s) to list

--max-items *<MAX_ITEMS>*

The total number of items to return in the command's output. If the total number of items available is more than the value specified, a next-token is provided in the command's output. To resume pagination, provide the next-token value in the starting-token argument of a subsequent command [default: 10]

--starting-token *<STARTING_TOKEN>*

A token to specify where to start paginating. This is the next-token from a previously truncated response

-v, --verbose

If included, prints all attributes and key information for each matched key. By default each matched key only displays its key-reference and label attribute. This flag when used by Admins has no effect

-h, --help

[Print help](#)

Contoh

Contoh berikut menunjukkan berbagai cara Anda menjalankan key list perintah. Contoh berikut menunjukkan output sebagai pengguna crypto.

Example Contoh: Temukan semua kunci — default

Perintah ini mencantumkan kunci pengguna yang masuk yang ada di AWS CloudHSM cluster.

Note

Secara default, hanya 10 kunci dari pengguna yang saat ini masuk ditampilkan, dan hanya key-reference dan label ditampilkan sebagai output. Gunakan opsi pagination yang sesuai untuk menampilkan lebih banyak atau lebih sedikit tombol sebagai output.

```
aws-cloudhsm > key list
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000000003d5",
        "attributes": {
          "label": "test_label_1"
        }
      },
      {
        "key-reference": "0x00000000000000626",
        "attributes": {
          "label": "test_label_2"
        }
      },
      ...8 keys later...
    ],
    "total_key_count": 56,
    "returned_key_count": 10,
    "next_token": "10"
  }
}
```

Example Contoh: Temukan semua kunci — verbose

Outputnya mencakup kunci yang dimiliki dan dibagikan pengguna, serta semua kunci publik di HSMs.

Note

Catatan: Secara default, hanya 10 kunci dari pengguna yang sedang masuk yang ditampilkan. Gunakan opsi pagination yang sesuai untuk menampilkan lebih banyak atau lebih sedikit tombol sebagai output.

```
aws-cloudhsm > key list --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000012000c",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "key-quorum-values": {
            "manage-key-quorum-value": 0,
            "use-key-quorum-value": 0
          },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "ec",
          "label": "ec-test-private-key",
          "id": "",
          "check-value": "0x2a737d",
          "class": "private-key",
          "encrypt": false,
          "decrypt": false,
          "token": true,
          "always-sensitive": true,
```



```

    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": false,
    "trusted": false,
    "unwrap": false,
    "verify": false,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 122,
    "ec-point":
"0x0442d53274a6c0ec1a23c165dcb9ccdd72c64e98ae1a9594bb5284e752c746280667e11f1e983493c1c605e0a80
    "curve": "secp224r1"
  }
},
{
  "key-reference": "0x0000000000012000d",
  "key-info": {
    "key-owners": [
      {
        "username": "cu1",
        "key-coverage": "full"
      }
    ],
    "shared-users": [],
    "key-quorum-values": {
      "manage-key-quorum-value": 0,
      "use-key-quorum-value": 0
    }
  },
  "cluster-coverage": "full"
},
{
  "attributes": {
    "key-type": "ec",
    "label": "ec-test-public-key",
    "id": "",
    "check-value": "0x2a737d",
    "class": "public-key",
    "encrypt": false,
    "decrypt": false,

```

```

        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 57,
        "ec-point":
"0x0442d53274a6c0ec1a23c165dcb9ccdd72c64e98ae1a9594bb5284e752c746280667e11f1e983493c1c605e0a80
        "curve": "secp224r1"
    }
}
],
    ...8 keys later...
    "total_key_count": 1580,
    "returned_key_count": 10
}
}

```

Example Contoh: Pengembalian paginasi

Contoh berikut menampilkan subset paginasi tombol yang hanya menampilkan dua tombol. Contoh kemudian memberikan panggilan berikutnya untuk menampilkan dua tombol berikutnya.

```

aws-cloudhsm > key list --verbose --max-items 2
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000000000030",
        "key-info": {
          "key-owners": [

```

```

        {
            "username": "cu1",
            "key-coverage": "full"
        }
    ],
    "shared-users": [],
    "key-quorum-values": {
        "manage-key-quorum-value": 0,
        "use-key-quorum-value": 0
    },
    "cluster-coverage": "full"
},
"attributes": {
    "key-type": "aes",
    "label": "98a6688d1d964ed7b45b9cec5c4b1909",
    "id": "",
    "check-value": "0xb28a46",
    "class": "secret-key",
    "encrypt": false,
    "decrypt": false,
    "token": true,
    "always-sensitive": true,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 32
}
},
{
    "key-reference": "0x000000000000000042",
    "key-info": {
        "key-owners": [
            {

```

```
        "username": "cu1",
        "key-coverage": "full"
    }
],
"shared-users": [],
"key-quorum-values": {
    "manage-key-quorum-value": 0,
    "use-key-quorum-value": 0
},
"cluster-coverage": "full"
},
"attributes": {
    "key-type": "aes",
    "label": "4ad6cdcbc02044e09fa954143efde233",
    "id": "",
    "check-value": "0xc98104",
    "class": "secret-key",
    "encrypt": true,
    "decrypt": true,
    "token": true,
    "always-sensitive": true,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": true,
    "verify": true,
    "wrap": true,
    "wrap-with-trusted": false,
    "key-length-bytes": 16
}
}
],
"total_key_count": 1580,
"returned_key_count": 2,
"next_token": "2"
}
```

```
}
```

Untuk menampilkan 2 tombol berikutnya, panggilan berikutnya dapat dilakukan:

```
aws-cloudhsm > key list --verbose --max-items 2 --starting-token 2
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000000000081",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "key-quorum-values": {
            "manage-key-quorum-value": 0,
            "use-key-quorum-value": 0
          },
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "6793b8439d044046982e5b895791e47f",
          "id": "",
          "check-value": "0x3f986f",
          "class": "secret-key",
          "encrypt": false,
          "decrypt": false,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
```

```
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 32
  }
},
{
  "key-reference": "0x00000000000000089",
  "key-info": {
    "key-owners": [
      {
        "username": "cu1",
        "key-coverage": "full"
      }
    ],
    "shared-users": [],
    "key-quorum-values": {
      "manage-key-quorum-value": 0,
      "use-key-quorum-value": 0
    },
    "cluster-coverage": "full"
  },
  "attributes": {
    "key-type": "aes",
    "label": "56b30fa05c6741faab8f606d3b7fe105",
    "id": "",
    "check-value": "0xe9201a",
    "class": "secret-key",
    "encrypt": false,
    "decrypt": false,
    "token": true,
    "always-sensitive": true,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
```

```

        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 32
    }
},
"total_key_count": 1580,
"returned_key_count": 2,
"next_token": "4"
}
}

```

Untuk contoh lebih lanjut yang menunjukkan cara kerja mekanisme filtrasi kunci di CloudHSM CLI, lihat. [Tombol filter menggunakan CloudHSM CLI](#)

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang cocok untuk daftar.

Untuk daftar atribut kunci CloudHSM CLI yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#)

Wajib: Tidak

<MAX_ITEMS>

Jumlah total item yang akan dikembalikan dalam output perintah. Jika jumlah total item yang tersedia lebih dari nilai yang ditentukan, token berikutnya disediakan dalam output perintah. Untuk melanjutkan pagination, berikan nilai next-token dalam argumen starting-token dari perintah berikutnya.

Wajib: Tidak

<STARTING_TOKEN>

Token untuk menentukan di mana harus memulai paginating. Ini adalah token berikutnya dari respons yang sebelumnya terpotong.

Wajib: Tidak

<VERBOSE>

Jika disertakan, mencetak semua atribut dan informasi kunci untuk setiap kunci yang cocok. Secara default setiap kunci yang cocok hanya menampilkan atribut kunci-referensi dan label. Bendera ini saat digunakan oleh Admin tidak berpengaruh.

Wajib: Tidak

Topik terkait

- [Hapus kunci dengan CloudHSM CLI](#)
- [Ekspor kunci asimetris dengan CloudHSM CLI](#)
- [Hapus berbagi kunci menggunakan CloudHSM CLI](#)
- [Atribut kunci untuk CloudHSM CLI](#)
- [Tombol filter menggunakan CloudHSM CLI](#)

Replikasi kunci dengan CloudHSM CLI

Gunakan key replicate perintah di CloudHSM CLI untuk mereplikasi kunci dari AWS CloudHSM cluster sumber ke cluster tujuan. AWS CloudHSM

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Admin () COs
- Pengguna Crypto (CUs)

 Note

Pengguna Crypto harus memiliki kunci untuk menggunakan perintah ini.

Persyaratan

- Cluster sumber dan tujuan harus klon. Ini berarti satu dibuat dari cadangan yang lain, atau keduanya dibuat dari cadangan umum. Untuk informasi selengkapnya, lihat [Membuat cluster dari backup](#).
- Pemilik kunci harus ada di cluster tujuan. Selain itu, jika kunci dibagikan dengan pengguna mana pun, pengguna tersebut juga harus ada di cluster tujuan.
- Untuk menjalankan perintah ini, Anda harus masuk sebagai pengguna krypto atau admin di cluster sumber dan tujuan.
- Dalam mode perintah tunggal, perintah akan menggunakan variabel lingkungan CLOUDHSM_PIN dan CLOUDHSM_ROLE untuk mengautentikasi pada cluster sumber. Untuk informasi selengkapnya, lihat [Mode Perintah Tunggal](#). Untuk memberikan kredensi untuk klaster tujuan, Anda perlu menyetel dua variabel lingkungan tambahan: DESTINATION_CLOUDHSM_PIN dan DESTINATION_CLOUDHSM_ROLE:

```
$ export DESTINATION_CLOUDHSM_ROLE=<role>
```

```
$ export DESTINATION_CLOUDHSM_PIN=<username:password>
```

- Dalam mode interaktif, pengguna harus secara eksplisit masuk ke cluster sumber dan tujuan.

Sintaksis

```
aws-cloudhsm > help key replicate
```

Replicate a key from a source to a destination cluster

```
Usage: key replicate --filter [<FILTER>...] --source-cluster-id <SOURCE_CLUSTER_ID> --  
destination-cluster-id <DESTINATION_CLUSTER_ID>
```

Options:

```
--filter [<FILTER>...]
```

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select matching key on the source cluster

```
--source-cluster-id <SOURCE_CLUSTER_ID>
```

Source cluster ID

```
--destination-cluster-id <DESTINATION_CLUSTER_ID>
```

Destination cluster ID

```
-h, --help
```

[Print help](#)

Contoh

Example Contoh: Kunci replikasi

Perintah ini mereplikasi kunci dari cluster sumber dengan ke klaster tujuan kloning. Contoh di bawah ini menunjukkan output saat masuk sebagai pengguna krypto di kedua cluster.

```
crypto-user-1@cluster-1234abcdefg > key replicate \
  --filter attr.label=example-key \
  --source-cluster-id cluster-1234abcdefg \
  --destination-cluster-id cluster-2345bcdefgh
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000003000006",
      "key-info": {
        "key-owners": [
          {
            "username": "crypto-user-1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "example-key",
        "id": "0x",
        "check-value": "0x5e118e",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
```

```
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": true,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 16
  }
},
"message": "Successfully replicated key"
}
```

Pendapat

<FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang cocok pada cluster sumber.

Untuk daftar atribut kunci CloudHSM CLI yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#)

Wajib: Ya

<SOURCE_CLUSTER_ID>

ID cluster sumber.

Wajib: Ya

<DESTINATION_CLUSTER_ID>

ID cluster tujuan.

Wajib: Ya

Topik terkait

- [Menghubungkan ke beberapa cluster dengan CloudHSM CLI](#)

Mengatur atribut kunci dengan CloudHSM CLI

Gunakan `key set-attribute` perintah di CloudHSM CLI untuk mengatur atribut kunci di cluster Anda. AWS CloudHSM Hanya CU yang membuat kunci dan akibatnya memilikinya yang dapat mengubah atribut kunci.

Untuk daftar atribut kunci yang dapat digunakan di CloudHSM CLI, lihat. [Atribut kunci untuk CloudHSM CLI](#)

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs) dapat menjalankan perintah ini.
- Admin dapat mengatur atribut tepercaya.

Persyaratan

Untuk menjalankan perintah ini, Anda harus masuk sebagai CU. Untuk mengatur atribut tepercaya, Anda harus masuk sebagai pengguna admin.

Sintaksis

```
aws-cloudhsm > help key set-attribute
```

```
Set an attribute for a key in the HSM cluster
```

```
Usage: cloudhsm-cli key set-attribute [OPTIONS] --filter [<FILTER>...] --  
name <KEY_ATTRIBUTE> --value <KEY_ATTRIBUTE_VALUE>
```

Options:

```
--cluster-id <CLUSTER_ID>           Unique Id to choose which of the clusters in  
the config file to run the operation against. If not provided, will fall back to the  
value provided when interactive mode was started, or error  
--filter [<FILTER>...]               Key reference (e.g. key-  
reference=0xabc) or space separated list of key attributes in the form of  
attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key to modify  
--name <KEY_ATTRIBUTE>              Name of attribute to be set
```

```

--value <KEY_ATTRIBUTE_VALUE>... Attribute value to be set
--approval <APPROVAL>           Filepath of signed quorum token file to approve
operation
-h, --help                       Print help

```

Contoh: Mengatur atribut kunci

Contoh berikut menunjukkan bagaimana menggunakan key set-attribute perintah untuk mengatur label.

Example

1. Gunakan kunci dengan label `my_key`, seperti yang ditunjukkan di sini:

```

aws-cloudhsm > key set-attribute --filter attr.label=my_key --name encrypt --value
false
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}

```

2. Gunakan key list perintah untuk mengonfirmasi bahwa encrypt atribut telah berubah:

```

aws-cloudhsm > key list --filter attr.label=my_key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000006400ec",
        "key-info": {
          "key-owners": [
            {
              "username": "bob",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "key-quorum-values": {
            "manage-key-quorum-value": 0,
            "use-key-quorum-value": 0
          }
        }
      }
    ]
  }
}

```

```

    },
    "cluster-coverage": "full"
  },
  "attributes": {
    "key-type": "aes",
    "label": "my_key",
    "id": "",
    "check-value": "0x6bd9f7",
    "class": "secret-key",
    "encrypt": false,
    "decrypt": true,
    "token": true,
    "always-sensitive": true,
    "derive": true,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": true,
    "unwrap": true,
    "verify": true,
    "wrap": true,
    "wrap-with-trusted": false,
    "key-length-bytes": 32
  }
},
"total_key_count": 1,
"returned_key_count": 1
}
}

```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<KEY_ATTRIBUTE>

Menentukan nama atribut kunci.

Wajib: Ya

<FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang cocok untuk dihapus.

Untuk daftar atribut kunci CloudHSM CLI yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#)

Wajib: Tidak

<KEY_ATTRIBUTE_VALUE>

Menentukan nilai atribut kunci.

Wajib: Ya

<KEY_REFERENCE>

Representasi heksadesimal atau desimal dari kunci. (seperti pegangan kunci).

Wajib: Tidak

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan manajemen kunci lebih besar dari 1.

Topik terkait

- [Tombol filter menggunakan CloudHSM CLI](#)
- [Atribut kunci untuk CloudHSM CLI](#)

Bagikan kunci menggunakan CloudHSM CLI

Gunakan `key share` perintah di CloudHSM CLI untuk berbagi kunci CUs dengan yang lain di cluster Anda. AWS CloudHSM

Hanya CU yang membuat kunci dan akibatnya memilikinya yang dapat berbagi kunci. Pengguna dengan siapa kunci dibagikan dapat menggunakan kunci dalam operasi kriptografi, tetapi mereka

tidak dapat menghapus, mengekspor, berbagi, atau membatalkan pembagian kunci. Selain itu, pengguna ini tidak dapat mengubah [atribut kunci](#).

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key share
```

Share a key in the HSM cluster with another user

Usage: key share --filter [**<FILTER>**...] --username **<USERNAME>** --role **<ROLE>**

Options:

--cluster-id **<CLUSTER_ID>**

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

--filter [<FILTER>**...]**

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key for sharing

--username **<USERNAME>**

A username with which the key will be shared

--role **<ROLE>**

Role the user has in the cluster

Possible values:

- crypto-user: A CryptoUser has the ability to manage and use keys
- admin: An Admin has the ability to manage user accounts

--approval **<APPROVAL>**

Filepath of signed quorum token file to approve operation


```
-h, --help
    Print help (see a summary with '-h')
```

Contoh: Bagikan kunci dengan CU lain

Contoh berikut menunjukkan cara menggunakan key share perintah untuk berbagi kunci dengan CUalice.

Example

1. Jalankan key share perintah untuk berbagi kunci denganalice.

```
aws-cloudhsm > key share --filter attr.label="rsa_key_to_share" attr.class=private-
key --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "message": "Key shared successfully"
  }
}
```

2. Jalankan perintah key list.

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" attr.class=private-
key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            }
          ]
        }
      }
    ]
  }
}
```

```
    },
    {
      "username": "cu1",
      "key-coverage": "full"
    },
    {
      "username": "cu4",
      "key-coverage": "full"
    },
    {
      "username": "cu5",
      "key-coverage": "full"
    },
    {
      "username": "cu6",
      "key-coverage": "full"
    },
    {
      "username": "cu7",
      "key-coverage": "full"
    },
    {
      "username": "alice",
      "key-coverage": "full"
    }
  ],
  "key-quorum-values": {
    "manage-key-quorum-value": 0,
    "use-key-quorum-value": 0
  },
  "cluster-coverage": "full"
},
"attributes": {
  "key-type": "rsa",
  "label": "rsa_key_to_share",
  "id": "",
  "check-value": "0xae8ff0",
  "class": "private-key",
  "encrypt": false,
  "decrypt": true,
  "token": true,
  "always-sensitive": true,
  "derive": false,
  "destroyable": true,
```

```

        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": true,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1219,
        "public-exponent": "0x010001",
        "modulus":
"0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
        "modulus-size-bits": 2048
    }
}
],
"total_key_count": 1,
"returned_key_count": 1
}
}

```

3. Dalam daftar di atas, verifikasi `alice` ada dalam daftar `shared-users`

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang cocok untuk dihapus.

Untuk daftar atribut kunci yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#).

Wajib: Ya

<USERNAME>

Menentukan nama yang mudah diingat untuk pengguna. Panjang maksimum adalah 31 karakter. Satu-satunya karakter khusus yang diizinkan adalah garis bawah (_). Nama pengguna tidak peka huruf besar/kecil dalam perintah ini, nama pengguna selalu ditampilkan dalam huruf kecil.

Wajib: Ya

<ROLE>

Menentukan peran yang ditetapkan untuk pengguna ini. Parameter ini diperlukan. Untuk mendapatkan peran pengguna, gunakan perintah daftar pengguna. Untuk informasi detail tentang jenis pengguna pada HSM, lihat [Jenis pengguna HSM untuk CloudHSM CLI](#).

Wajib: Ya

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan manajemen kunci kunci lebih besar dari 1.

Topik terkait

- [Tombol filter menggunakan CloudHSM CLI](#)
- [Atribut kunci untuk CloudHSM CLI](#)

Hapus berbagi kunci menggunakan CloudHSM CLI

Gunakan key unshare perintah di CloudHSM CLI untuk membatalkan berbagi kunci dengan yang lain di cluster Anda. CUs AWS CloudHSM

Hanya CU yang membuat kunci dan akibatnya memilikinya yang dapat membatalkan pembagian kunci. Pengguna dengan siapa kunci dibagikan dapat menggunakan kunci dalam operasi kriptografi, tetapi mereka tidak dapat menghapus, mengekspor, berbagi, atau membatalkan berbagi kunci. Selain itu, pengguna ini tidak dapat mengubah [atribut kunci](#).

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key unshare
```

Unshare a key in the HSM cluster with another user

Usage: key unshare --filter [**<FILTER>**...] --username **<USERNAME>** --role **<ROLE>**

Options:

--cluster-id <CLUSTER_ID>

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

--filter [<FILTER>**...]**

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key for unsharing

--username <USERNAME>

A username with which the key will be unshared

--role <ROLE>

Role the user has in the cluster

Possible values:

- crypto-user: A CryptoUser has the ability to manage and use keys
- admin: An Admin has the ability to manage user accounts

--approval <APPROVAL>

Filepath of signed quorum token file to approve operation

-h, --help

Print help (see a summary with '-h')

Contoh: Unshare kunci dengan CU lain

Contoh berikut menunjukkan cara menggunakan key unshare perintah untuk membatalkan berbagi kunci dengan CUalice.

Example

1. Jalankan key list perintah dan filter dengan kunci spesifik yang ingin Anda unshare. alice

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" attr.class=private-  
key --verbose  
{  
  "error_code": 0,  
  "data": {  
    "matched_keys": [  
      {  
        "key-reference": "0x000000000001c0686",  
        "key-info": {  
          "key-owners": [  
            {  
              "username": "cu3",  
              "key-coverage": "full"  
            }  
          ],  
          "shared-users": [  
            {  
              "username": "cu2",  
              "key-coverage": "full"  
            },  
            {  
              "username": "cu1",  
              "key-coverage": "full"  
            },  
            {  
              "username": "cu4",  
              "key-coverage": "full"  
            },  
            {  
              "username": "cu5",  
              "key-coverage": "full"  
            },  
            {  
              "username": "cu6",  
              "key-coverage": "full"  
            },  
            {  
              "username": "cu7",  
              "key-coverage": "full"  
            }  
          ]  
        }  
      }  
    ]  
  }  
}
```

```

    },
    {
      "username": "alice",
      "key-coverage": "full"
    }
  ],
  "key-quorum-values": {
    "manage-key-quorum-value": 0,
    "use-key-quorum-value": 0
  },
  "cluster-coverage": "full"
},
"attributes": {
  "key-type": "rsa",
  "label": "rsa_key_to_share",
  "id": "",
  "check-value": "0xae8ff0",
  "class": "private-key",
  "encrypt": false,
  "decrypt": true,
  "token": true,
  "always-sensitive": true,
  "derive": false,
  "destroyable": true,
  "extractable": true,
  "local": true,
  "modifiable": true,
  "never-extractable": false,
  "private": true,
  "sensitive": true,
  "sign": true,
  "trusted": false,
  "unwrap": true,
  "verify": false,
  "wrap": false,
  "wrap-with-trusted": false,
  "key-length-bytes": 1219,
  "public-exponent": "0x010001",
  "modulus":
    "0xa855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254",
  "modulus-size-bits": 2048
}
}
],

```

```

    "total_key_count": 1,
    "returned_key_count": 1
  }
}

```

2. Konfirmasikan `alice` ada di `shared-users` output, dan jalankan `key unshare` perintah berikut untuk membatalkan pembagian kunci dengan `alice`.

```

aws-cloudhsm > key unshare --filter attr.label="rsa_key_to_share"
attr.class=private-key --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "message": "Key unshared successfully"
  }
}

```

3. Jalankan `key list` perintah lagi untuk mengonfirmasi bahwa kunci telah tidak dibagikan `alice`.

```

aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" attr.class=private-
key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ]
        }
      }
    ]
  }
}

```



```
{
  "username": "cu4",
  "key-coverage": "full"
},
{
  "username": "cu5",
  "key-coverage": "full"
},
{
  "username": "cu6",
  "key-coverage": "full"
},
{
  "username": "cu7",
  "key-coverage": "full"
},
],
"key-quorum-values": {
  "manage-key-quorum-value": 0,
  "use-key-quorum-value": 0
},
"cluster-coverage": "full"
},
"attributes": {
  "key-type": "rsa",
  "label": "rsa_key_to_share",
  "id": "",
  "check-value": "0xae8ff0",
  "class": "private-key",
  "encrypt": false,
  "decrypt": true,
  "token": true,
  "always-sensitive": true,
  "derive": false,
  "destroyable": true,
  "extractable": true,
  "local": true,
  "modifiable": true,
  "never-extractable": false,
  "private": true,
  "sensitive": true,
  "sign": true,
  "trusted": false,
  "unwrap": true,
```

```

        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1219,
        "public-exponent": "0x010001",
        "modulus":
"0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
        "modulus-size-bits": 2048
    }
}
],
"total_key_count": 1,
"returned_key_count": 1
}
}

```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Diperlukan: Jika beberapa cluster telah [dikonfigurasi](#).

<FILTER>

Referensi kunci (misalnya, key-reference=0xabc) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk memilih kunci yang cocok untuk dihapus.

Untuk daftar atribut kunci yang didukung, lihat [Atribut kunci untuk CloudHSM CLI](#).

Wajib: Ya

<USERNAME>

Menentukan nama yang mudah diingat untuk pengguna. Panjang maksimum adalah 31 karakter. Satu-satunya karakter khusus yang diizinkan adalah garis bawah (_). Nama pengguna tidak peka huruf besar/kecil dalam perintah ini, nama pengguna selalu ditampilkan dalam huruf kecil.

Wajib: Ya

<ROLE>

Menentukan peran yang ditetapkan untuk pengguna ini. Parameter ini diperlukan. Untuk mendapatkan peran pengguna, gunakan perintah daftar pengguna. Untuk informasi detail tentang jenis pengguna pada HSM, lihat [Jenis pengguna HSM untuk CloudHSM CLI](#).

Wajib: Ya

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan manajemen kunci lebih besar dari 1.

Topik terkait

- [Tombol filter menggunakan CloudHSM CLI](#)
- [Atribut kunci untuk CloudHSM CLI](#)

Perintah buka kunci di CloudHSM CLI

Perintah `key unwrap` induk di CloudHSM CLI mengimpor kunci pribadi simetris atau asimetris terenkripsi (dibungkus) dari file dan ke HSM. Perintah ini dirancang untuk mengimpor kunci terenkripsi yang dibungkus oleh [Perintah bungkus kunci di CloudHSM CLI](#) perintah, tetapi juga dapat digunakan untuk membuka kunci yang dibungkus dengan alat lain. Namun, dalam situasi tersebut, sebaiknya gunakan PKCS #11 atau pustaka perangkat lunak JCE untuk membuka kunci.

- [aes-gcm](#)
- [aes-no-pad](#)
- [aes-pkcs5-pad](#)
- [aes-zero-pad](#)
- [cloudhsm-aes-gcm](#)
- [rsa-aes](#)
- [rsa-oaep](#)
- [rsa-PKC](#)

Buka kunci dengan AES-GCM menggunakan CloudHSM CLI

Gunakan `key unwrap aes-gcm` perintah di CloudHSM CLI untuk membuka kunci payload ke dalam cluster menggunakan kunci pembungkus AES dan mekanisme unwrapping. AES-GCM

Kunci yang tidak dibungkus dapat digunakan dengan cara yang sama seperti kunci yang dihasilkan oleh AWS CloudHSM. Untuk menunjukkan bahwa mereka tidak dihasilkan secara lokal, `local` atribut mereka disetel ke `false`.

Untuk menggunakan `key unwrap aes-gcm` perintah, Anda harus memiliki kunci pembungkus AES di AWS CloudHSM cluster Anda, dan `unwrap` atributnya harus disetel ke `true`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key unwrap aes-gcm
Usage: key unwrap aes-gcm [OPTIONS] --filter [<FILTER>...] --tag-length-
bits <TAG_LENGTH_BITS> --key-type-class <KEY_TYPE_CLASS> --label <LABEL> --iv <IV> <--
data-path <DATA_PATH>|--data <DATA>>

Options:
  --cluster-id <CLUSTER_ID>
      Unique Id to choose which of the clusters in the config file to run the
      operation against. If not provided, will fall back to the value provided when
      interactive mode was started, or error
  --filter [<FILTER>...]
      Key reference (e.g. key-reference=0xabc) or space separated list of key
      attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
      to unwrap with
  --data-path <DATA_PATH>
      Path to the binary file containing the wrapped key data
  --data <DATA>
      Base64 encoded wrapped key data
```

```

--attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
    Space separated list of key attributes in the form of
    KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
--share-crypto-users [<SHARE_CRYPTO_USERS>;...]
    Space separated list of Crypto User usernames to share the unwrapped key with
--manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE>;
    The quorum value for key management operations for the unwrapped key
--use-key-quorum-value <USE_KEY_QUORUM_VALUE>;
    The quorum value for key usage operations for the unwrapped key
--aad <AAD>
    Aes GCM Additional Authenticated Data (AAD) value, in hex
--tag-length-bits <TAG_LENGTH_BITS>
    Aes GCM tag length in bits
--key-type-class <KEY_TYPE_CLASS>
    Key type and class of wrapped key [possible values: aes, des3, ec-private,
    generic-secret, rsa-private]
--label <LABEL>
    Label for the unwrapped key
--session
    Creates a session key that exists only in the current session. The key cannot
    be recovered after the session ends
--iv <IV>
    Initial value used to wrap the key, in hex
--approval <APPROVAL>
    Filepath of signed quorum token file to approve operation
-h, --help
    Print help

```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan key unwrap aes-gcm perintah menggunakan kunci AES dengan nilai unwrap atribut yang disetel ke true.

Example Contoh: Buka kunci payload dari data kunci dibungkus yang dikodekan Base64

```

aws-cloudhsm > key unwrap aes-gcm --key-type-class aes --label aes-unwrapped
--filter attr.label=aes-example --tag-length-bits 64 --aad 0x10 --iv
0xf90613bb8e337ec0339aad21 --data xvslgrtg8kHrzvekn97tLSIeokpPwV8
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x0000000000001808e4",

```

```
"key-info": {
  "key-owners": [
    {
      "username": "cu1",
      "key-coverage": "full"
    }
  ],
  "shared-users": [],
  "key-quorum-values": {
    "manage-key-quorum-value": 0,
    "use-key-quorum-value": 0
  },
  "cluster-coverage": "full"
},
"attributes": {
  "key-type": "aes",
  "label": "aes-unwrapped",
  "id": "0x",
  "check-value": "0x8d9099",
  "class": "secret-key",
  "encrypt": false,
  "decrypt": false,
  "token": true,
  "always-sensitive": false,
  "derive": false,
  "destroyable": true,
  "extractable": true,
  "local": false,
  "modifiable": true,
  "never-extractable": false,
  "private": true,
  "sensitive": true,
  "sign": true,
  "trusted": false,
  "unwrap": false,
  "verify": true,
  "wrap": false,
  "wrap-with-trusted": false,
  "key-length-bytes": 16
}
}
}
```

Example Contoh: Buka kunci payload yang disediakan melalui jalur data

```
aws-cloudhsm > key unwrap aes-gcm --key-type-class aes --label aes-unwrapped  
--filter attr.label=aes-example --tag-length-bits 64 --aad 0x10 --iv  
0xf90613bb8e337ec0339aad21 --data-path payload-key.pem
```

```
{  
  "error_code": 0,  
  "data": {  
    "key": {  
      "key-reference": "0x000000000001808e4",  
      "key-info": {  
        "key-owners": [  
          {  
            "username": "cu1",  
            "key-coverage": "full"  
          }  
        ],  
        "shared-users": [],  
        "key-quorum-values": {  
          "manage-key-quorum-value": 0,  
          "use-key-quorum-value": 0  
        },  
        "cluster-coverage": "full"  
      },  
      "attributes": {  
        "key-type": "aes",  
        "label": "aes-unwrapped",  
        "id": "0x",  
        "check-value": "0x8d9099",  
        "class": "secret-key",  
        "encrypt": false,  
        "decrypt": false,  
        "token": true,  
        "always-sensitive": false,  
        "derive": false,  
        "destroyable": true,  
        "extractable": true,  
        "local": false,  
        "modifiable": true,  
        "never-extractable": false,  
        "private": true,  
        "sensitive": true,  
        "sign": true,  
        "trusted": false,
```

```
    "unwrap": false,  
    "verify": true,  
    "wrap": false,  
    "wrap-with-trusted": false,  
    "key-length-bytes": 16  
  }  
}  
}  
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang akan dibuka.

Wajib: Ya

<DATA_PATH>

Path ke file biner yang berisi data kunci dibungkus.

Diperlukan: Ya (kecuali disediakan melalui data yang disandikan Base64)

<DATA>

Base64 mengkodekan data kunci yang dibungkus.

Wajib: Ya (kecuali disediakan melalui jalur data)

<ATTRIBUTES>

Spasi dipisahkan daftar atribut kunci dalam bentuk `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk kunci dibungkus.

Wajib: Tidak

<AAD>

Nilai Aes GCM Additional Authenticated Data (AAD), dalam hex.

Wajib: Tidak

<TAG_LENGTH_BITS>

Panjang tag Aes GCM dalam bit.

Wajib: Ya

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus [nilai yang mungkin:aes,des3,ec-private,generic-secret,rsa-private].

Wajib: Ya

<LABEL>

Label untuk kunci yang tidak dibungkus.

Wajib: Ya

<SESSION>

Membuat kunci sesi yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Wajib: Tidak

<IV>

Nilai awal yang digunakan untuk membungkus kunci, dalam hex.

Wajib: Tidak

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan manajemen kunci dari kunci unwrapping lebih besar dari 1.

Topik terkait

- [Perintah bungkus kunci di CloudHSM CLI](#)
- [Perintah buka kunci di CloudHSM CLI](#)

Buka kunci dengan AES-NO-PAD menggunakan CloudHSM CLI

Gunakan `key unwrap aes-no-pad` perintah di CloudHSM CLI untuk membuka kunci payload ke AWS CloudHSM dalam cluster menggunakan kunci pembungkus AES dan mekanisme unwrapping. AES-NO-PAD

Kunci yang tidak dibungkus dapat digunakan dengan cara yang sama seperti kunci yang dihasilkan oleh AWS CloudHSM. Untuk menunjukkan bahwa mereka tidak dihasilkan secara lokal, `local` atribut mereka disetel ke `false`.

Untuk menggunakan `key unwrap aes-no-pad` perintah, Anda harus memiliki kunci pembungkus AES di AWS CloudHSM cluster Anda, dan `unwrap` atributnya harus disetel ke `true`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key unwrap aes-no-pad
```

```
Usage: key unwrap aes-no-pad [OPTIONS] --filter [<FILTER>...] --key-type-class <KEY_TYPE_CLASS> --label <LABEL> <--data-path <DATA_PATH>|--data <DATA>>
```

Options:

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

```
--filter [<FILTER>...]
```

Key reference (e.g. `key-reference=0xabc`) or space separated list of key attributes in the form of `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` to select a key to unwrap with

```
--data-path <DATA_PATH>
```

Path to the binary file containing the wrapped key data

```
--data <DATA>
```

Base64 encoded wrapped key data

```

--attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
    Space separated list of key attributes in the form of
    KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
--share-crypto-users [<SHARE_CRYPTO_USERS>;...]
    Space separated list of Crypto User usernames to share the unwrapped key with
--manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE>;
    The quorum value for key management operations for the unwrapped key
--use-key-quorum-value <USE_KEY_QUORUM_VALUE>;
    The quorum value for key usage operations for the unwrapped key
--key-type-class <KEY_TYPE_CLASS>
    Key type and class of wrapped key [possible values: aes, des3, ec-private,
    generic-secret, rsa-private]
--label <LABEL>
    Label for the unwrapped key
--session
    Creates a session key that exists only in the current session. The key cannot
    be recovered after the session ends
--approval <APPROVAL>
    Filepath of signed quorum token file to approve operation
-h, --help
    Print help

```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan key unwrap aes-no-pad perintah menggunakan kunci AES dengan nilai unwrap atribut yang disetel ke true.

Example Contoh: Buka kunci payload dari data kunci dibungkus yang dikodekan Base64

```

aws-cloudhsm > key unwrap aes-no-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data eXK3PMA0nKM9y3YX6brbhtMoC060E0H9
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000001c08ec",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ]
      }
    }
  },

```

```

    "shared-users": [],
    "key-quorum-values": {
      "manage-key-quorum-value": 0,
      "use-key-quorum-value": 0
    },
    "cluster-coverage": "full"
  },
  "attributes": {
    "key-type": "aes",
    "label": "aes-unwrapped",
    "id": "0x",
    "check-value": "0x8d9099",
    "class": "secret-key",
    "encrypt": false,
    "decrypt": false,
    "token": true,
    "always-sensitive": false,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": false,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 16
  }
}
}
}
}

```

Example Contoh: Buka kunci payload yang disediakan melalui jalur data

```

aws-cloudhsm > key unwrap aes-no-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data-path payload-key.pem
{
  "error_code": 0,

```

```
"data": {
  "key": {
    "key-reference": "0x000000000001c08ec",
    "key-info": {
      "key-owners": [
        {
          "username": "cu1",
          "key-coverage": "full"
        }
      ],
      "shared-users": [],
      "key-quorum-values": {
        "manage-key-quorum-value": 0,
        "use-key-quorum-value": 0
      },
      "cluster-coverage": "full"
    },
    "attributes": {
      "key-type": "aes",
      "label": "aes-unwrapped",
      "id": "0x",
      "check-value": "0x8d9099",
      "class": "secret-key",
      "encrypt": false,
      "decrypt": false,
      "token": true,
      "always-sensitive": false,
      "derive": false,
      "destroyable": true,
      "extractable": true,
      "local": false,
      "modifiable": true,
      "never-extractable": false,
      "private": true,
      "sensitive": true,
      "sign": true,
      "trusted": false,
      "unwrap": false,
      "verify": true,
      "wrap": false,
      "wrap-with-trusted": false,
      "key-length-bytes": 16
    }
  }
}
```

```
}  
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Diperlukan: Jika beberapa cluster telah [dikonfigurasi](#).

<FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang akan dibuka.

Wajib: Ya

<DATA_PATH>

Path ke file biner yang berisi data kunci dibungkus.

Diperlukan: Ya (kecuali disediakan melalui data yang disandikan Base64)

<DATA>

Base64 mengkodekan data kunci yang dibungkus.

Diperlukan: Ya (kecuali disediakan melalui jalur data)

<ATTRIBUTES>

Spasi dipisahkan daftar atribut kunci dalam bentuk `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk kunci dibungkus.

Wajib: Tidak

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus [nilai yang mungkin: `aes`, `des3`, `ec-private`, `generic-secret`, `rsa-private`].

Wajib: Ya

<LABEL>

Label untuk kunci yang tidak dibungkus.

Wajib: Ya

<SESSION>

Membuat kunci sesi yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Wajib: Tidak

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan manajemen kunci dari kunci unwrapping lebih besar dari 1.

Topik terkait

- [Perintah bungkus kunci di CloudHSM CLI](#)
- [Perintah buka kunci di CloudHSM CLI](#)

Buka kunci dengan AES- PKCS5 -PAD menggunakan CloudHSM CLI

Gunakan key unwrap aes-pkcs5-pad perintah di CloudHSM CLI untuk membuka kunci payload menggunakan kunci pembungkus AES dan mekanisme unwrapping. AES-PKCS5-PAD

Kunci yang tidak dibungkus dapat digunakan dengan cara yang sama seperti kunci yang dihasilkan oleh AWS CloudHSM. Untuk menunjukkan bahwa mereka tidak dihasilkan secara lokal, `local` atribut mereka disetel ke `false`.

Untuk menggunakan key unwrap aes-pkcs5-pad perintah, Anda harus memiliki kunci pembungkus AES di AWS CloudHSM cluster Anda, dan unwrap atributnya harus disetel ke `true`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key unwrap aes-pkcs5-pad
```

```
Usage: key unwrap aes-pkcs5-pad [OPTIONS] --filter [<FILTER>...] --key-type-  
class <KEY_TYPE_CLASS> --label <LABEL> [--data-path <DATA_PATH>|--data <DATA>]
```

Options:

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

```
--filter [<FILTER>...]
```

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key to unwrap with

```
--data-path <DATA_PATH>
```

Path to the binary file containing the wrapped key data

```
--data <DATA>
```

Base64 encoded wrapped key data

```
--attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
```

Space separated list of key attributes in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key

```
--share-crypto-users [<SHARE_CRYPTO_USERS>...]
```

Space separated list of Crypto User usernames to share the unwrapped key with

```
--manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE>;
```

The quorum value for key management operations for the unwrapped key

```
--use-key-quorum-value <USE_KEY_QUORUM_VALUE>;
```

The quorum value for key usage operations for the unwrapped key

```
--key-type-class <KEY_TYPE_CLASS>
```

Key type and class of wrapped key [possible values: aes, des3, ec-private, generic-secret, rsa-private]

```
--label <LABEL>
```

Label for the unwrapped key

```
--session
```

Creates a session key that exists only in the current session. The key cannot be recovered after the session ends

```
--approval <APPROVAL>
```

Filepath of signed quorum token file to approve operation

```
-h, --help
```

Print help

Contoh

Contoh-contoh ini menunjukkan cara menggunakan key unwrap aes-pkcs5-pad perintah menggunakan kunci AES dengan nilai unwrap atribut yang disetel ke true.

Example Contoh: Buka kunci payload dari data kunci dibungkus yang dikodekan Base64

```
aws-cloudhsm > key unwrap aes-pkcs5-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data MbuYNresf0KyGNnxKwen88nSfX+uUE/0qmGofSisicY=
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000001c08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
```

```

    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 16
  }
}
}
}

```

Example Contoh: Buka kunci payload yang disediakan melalui jalur data

```

aws-cloudhsm > key unwrap aes-pkcs5-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000001c08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",

```

```
"encrypt": false,
"decrypt": false,
"token": true,
"always-sensitive": false,
"derive": false,
"destroyable": true,
"extractable": true,
"local": false,
"modifiable": true,
"never-extractable": false,
"private": true,
"sensitive": true,
"sign": true,
"trusted": false,
"unwrap": false,
"verify": true,
"wrap": false,
"wrap-with-trusted": false,
"key-length-bytes": 16
}
}
}
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang akan dibuka.

Wajib: Ya

<DATA_PATH>

Path ke file biner yang berisi data kunci yang dibungkus.

Diperlukan: Ya (kecuali disediakan melalui data yang disandikan Base64)

<DATA>

Base64 mengkodekan data kunci yang dibungkus.

Wajib: Ya (kecuali disediakan melalui jalur data)

<ATTRIBUTES>

Spasi dipisahkan daftar atribut kunci dalam bentuk
KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk kunci dibungkus.

Wajib: Tidak

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus [nilai yang mungkin:aes,des3,ec-private,generic-secret,rsa-private].

Wajib: Ya

<LABEL>

Label untuk kunci yang tidak dibungkus.

Wajib: Ya

<SESSION>

Membuat kunci sesi yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Wajib: Tidak

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan manajemen kunci dari kunci unwrapping lebih besar dari 1.

Topik terkait

- [Perintah bungkus kunci di CloudHSM CLI](#)
- [Perintah buka kunci di CloudHSM CLI](#)

Buka kunci dengan AES-ZERO-PAD menggunakan CloudHSM CLI

Gunakan `key unwrap aes-zero-pad` perintah di CloudHSM CLI untuk membuka kunci payload ke AWS CloudHSM dalam cluster menggunakan kunci pembungkus AES dan mekanisme unwrapping. AES-ZERO-PAD

Kunci yang tidak dibungkus dapat digunakan dengan cara yang sama seperti kunci yang dihasilkan oleh AWS CloudHSM. Untuk menunjukkan bahwa mereka tidak dihasilkan secara lokal, `local` atribut mereka disetel ke `false`.

Untuk menggunakan `key unwrap aes-no-pad` perintah, Anda harus memiliki kunci pembungkus AES di AWS CloudHSM cluster Anda, dan `unwrap` atributnya harus disetel ke `true`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key unwrap aes-zero-pad
Usage: key unwrap aes-zero-pad [OPTIONS] --filter [<FILTER>...] --key-type-
class <KEY_TYPE_CLASS> --label <LABEL> <--data-path <DATA_PATH>|--data <DATA>>

Options:
  --cluster-id <CLUSTER_ID>
      Unique Id to choose which of the clusters in the config file to run the
      operation against. If not provided, will fall back to the value provided when
      interactive mode was started, or error
  --filter [<FILTER>...]
      Key reference (e.g. key-reference=0xabc) or space separated list of key
      attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
      to unwrap with
  --data-path <DATA_PATH>
      Path to the binary file containing the wrapped key data
  --data <DATA>
      Base64 encoded wrapped key data
```

```

--attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
    Space separated list of key attributes in the form of
    KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
--share-crypto-users [<SHARE_CRYPTO_USERS>;...]
    Space separated list of Crypto User usernames to share the unwrapped key with
--manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE>;
    The quorum value for key management operations for the unwrapped key
--use-key-quorum-value <USE_KEY_QUORUM_VALUE>;
    The quorum value for key usage operations for the unwrapped key
--key-type-class <KEY_TYPE_CLASS>
    Key type and class of wrapped key [possible values: aes, des3, ec-private,
    generic-secret, rsa-private]
--label <LABEL>
    Label for the unwrapped key
--session
    Creates a session key that exists only in the current session. The key cannot
    be recovered after the session ends
--approval <APPROVAL>
    Filepath of signed quorum token file to approve operation
-h, --help
    Print help

```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan key unwrap aes-zero-pad perintah menggunakan kunci AES dengan nilai unwrap atribut yang disetel ke true.

Example Contoh: Buka kunci payload dari data kunci dibungkus yang dikodekan Base64

```

aws-cloudhsm > key unwrap aes-zero-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data L1wV1L/YeBNVAw6Mpk3owFJZXBzDL0nt
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000001c08e7",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ]
      }
    }
  },

```

```

    "shared-users": [],
    "key-quorum-values": {
      "manage-key-quorum-value": 0,
      "use-key-quorum-value": 0
    },
    "cluster-coverage": "full"
  },
  "attributes": {
    "key-type": "aes",
    "label": "aes-unwrapped",
    "id": "0x",
    "check-value": "0x8d9099",
    "class": "secret-key",
    "encrypt": false,
    "decrypt": false,
    "token": true,
    "always-sensitive": false,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": false,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 16
  }
}
}
}
}

```

Example Contoh: Buka kunci payload yang disediakan melalui jalur data

```

aws-cloudhsm > key unwrap aes-zero-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data-path payload-key.pem
{
  "error_code": 0,

```

```
"data": {
  "key": {
    "key-reference": "0x000000000001c08e7",
    "key-info": {
      "key-owners": [
        {
          "username": "cu1",
          "key-coverage": "full"
        }
      ],
      "shared-users": [],
      "key-quorum-values": {
        "manage-key-quorum-value": 0,
        "use-key-quorum-value": 0
      },
      "cluster-coverage": "full"
    },
    "attributes": {
      "key-type": "aes",
      "label": "aes-unwrapped",
      "id": "0x",
      "check-value": "0x8d9099",
      "class": "secret-key",
      "encrypt": false,
      "decrypt": false,
      "token": true,
      "always-sensitive": false,
      "derive": false,
      "destroyable": true,
      "extractable": true,
      "local": false,
      "modifiable": true,
      "never-extractable": false,
      "private": true,
      "sensitive": true,
      "sign": true,
      "trusted": false,
      "unwrap": false,
      "verify": true,
      "wrap": false,
      "wrap-with-trusted": false,
      "key-length-bytes": 16
    }
  }
}
```



```
}  
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Diperlukan: Jika beberapa cluster telah [dikonfigurasi](#).

<FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang akan dibuka.

Wajib: Ya

<DATA_PATH>

Path ke file biner yang berisi data kunci dibungkus.

Diperlukan: Ya (kecuali disediakan melalui data yang disandikan Base64)

<DATA>

Base64 mengkodekan data kunci yang dibungkus.

Diperlukan: Ya (kecuali disediakan melalui jalur data)

<ATTRIBUTES>

Spasi dipisahkan daftar atribut kunci dalam bentuk `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk kunci dibungkus.

Wajib: Tidak

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus [nilai yang mungkin: `aes`, `des3`, `ec-private`, `generic-secret`, `rsa-private`].

Wajib: Ya

<LABEL>

Label untuk kunci yang tidak dibungkus.

Wajib: Ya

<SESSION>

Membuat kunci sesi yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Wajib: Tidak

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan manajemen kunci dari kunci unwrapping lebih besar dari 1.

Topik terkait

- [Perintah bungkus kunci di CloudHSM CLI](#)
- [Perintah buka kunci di CloudHSM CLI](#)

Buka kunci dengan CLOUDHSM-AES-GCM menggunakan CloudHSM CLI

Gunakan key unwrap cloudhsm-aes-gcm perintah di CloudHSM CLI untuk membuka kunci payload ke AWS CloudHSM dalam cluster menggunakan kunci pembungkus AES dan mekanisme unwrapping. CLOUDHSM-AES-GCM

Kunci yang tidak dibungkus dapat digunakan dengan cara yang sama seperti kunci yang dihasilkan oleh AWS CloudHSM. Untuk menunjukkan bahwa mereka tidak dihasilkan secara lokal, `local` atribut mereka disetel ke `false`.

Untuk menggunakan key unwrap cloudhsm-aes-gcm perintah, Anda harus memiliki kunci pembungkus AES di AWS CloudHSM cluster Anda dan `unwrap` atributnya harus disetel ke `true`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key unwrap cloudhsm-aes-gcm
```

```
Usage: key unwrap cloudhsm-aes-gcm [OPTIONS] --filter [<FILTER>...] --tag-length-bits <TAG_LENGTH_BITS> --key-type-class <KEY_TYPE_CLASS> --label <LABEL> <--data-path <DATA_PATH>|--data <DATA>>
```

Options:

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

```
--filter [<FILTER>...]
```

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key to unwrap with

```
--data-path <DATA_PATH>
```

Path to the binary file containing the wrapped key data

```
--data <DATA>
```

Base64 encoded wrapped key data

```
--attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
```

Space separated list of key attributes in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key

```
--share-crypto-users [<SHARE_CRYPTO_USERS;...]
```

Space separated list of Crypto User usernames to share the unwrapped key with

```
--manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE;
```

The quorum value for key management operations for the unwrapped key

```
--use-key-quorum-value <USE_KEY_QUORUM_VALUE;
```

The quorum value for key usage operations for the unwrapped key

```
--aad <AAD>
```

Aes GCM Additional Authenticated Data (AAD) value, in hex

```
--tag-length-bits <TAG_LENGTH_BITS>
```

Aes GCM tag length in bits

```
--key-type-class <KEY_TYPE_CLASS>
```

Key type and class of wrapped key [possible values: aes, des3, ec-private, generic-secret, rsa-private]

```
--label <LABEL>
```

Label for the unwrapped key

```
--session
```

Creates a session key that exists only in the current session. The key cannot be recovered after the session ends

```
--approval <APPROVAL>
```

Filepath of signed quorum token file to approve operation

```
-h, --help
```

Print help

Contoh

Contoh-contoh ini menunjukkan cara menggunakan key unwrap cloudhsm-aes-gcm perintah menggunakan kunci AES dengan nilai unwrap atribut yang disetel ke true.

Example Contoh: Buka kunci payload dari data kunci dibungkus yang dikodekan Base64

```
aws-cloudhsm > key unwrap cloudhsm-aes-gcm --key-type-class aes --label aes-
unwrapped --filter attr.label=aes-example --tag-length-bits 64 --aad 0x10 --data
6Rn8nkjEriDYlnP3P8nPkYQ8hp10EJ899zsrF+aTB0i/f1lZ
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000001408e8",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
```

```

    "extractable": true,
    "local": false,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 16
  }
}
}
}

```

Example Contoh: Buka kunci payload yang disediakan melalui jalur data

```

aws-cloudhsm > key unwrap cloudhsm-aes-gcm --key-type-class aes --label aes-unwrapped
--filter attr.label=aes-example --tag-length-bits 64 --aad 0x10 --data-path payload-
key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000001408e8",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {

```

```

    "key-type": "aes",
    "label": "aes-unwrapped",
    "id": "0x",
    "check-value": "0x8d9099",
    "class": "secret-key",
    "encrypt": false,
    "decrypt": false,
    "token": true,
    "always-sensitive": false,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": false,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 16
  }
}
}
}

```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<FILTER>

Referensi kunci (misalnya, key-reference=0xabc) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk memilih kunci yang akan dibuka.

Wajib: Ya

<DATA_PATH>

Path ke file biner yang berisi data kunci dibungkus.

Diperlukan: Ya (kecuali disediakan melalui data yang disandikan Base64)

<DATA>

Base64 mengkodekan data kunci yang dibungkus.

Wajib: Ya (kecuali disediakan melalui jalur data)

<ATTRIBUTES>

Spasi dipisahkan daftar atribut kunci dalam bentuk

KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk kunci dibungkus.

Wajib: Tidak

<AAD>

Nilai Aes GCM Additional Authenticated Data (AAD), dalam hex.

Wajib: Tidak

<TAG_LENGTH_BITS>

Panjang tag Aes GCM dalam bit.

Wajib: Ya

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus [nilai yang mungkin:aes,des3,ec-private,generic-secret,rsa-private].

Wajib: Ya

<LABEL>

Label untuk kunci yang tidak dibungkus.

Wajib: Ya

<SESSION>

Membuat kunci sesi yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Wajib: Tidak

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan manajemen kunci dari kunci unwrapping lebih besar dari 1.

Topik terkait

- [Perintah bungkus kunci di CloudHSM CLI](#)
- [Perintah buka kunci di CloudHSM CLI](#)

Buka kunci dengan RSA-AES menggunakan CloudHSM CLI

Gunakan `key unwrap rsa-aes` perintah di CloudHSM CLI untuk membuka kunci payload menggunakan kunci pribadi RSA dan mekanisme unwrapping. RSA-AES

Kunci yang tidak dibungkus dapat digunakan dengan cara yang sama seperti kunci yang dihasilkan oleh AWS CloudHSM. Untuk menunjukkan bahwa mereka tidak dihasilkan secara lokal, `local` atribut mereka disetel ke `false`.

Untuk menggunakan `key unwrap rsa-aes`, Anda harus memiliki kunci pribadi RSA dari kunci pembungkus publik RSA di AWS CloudHSM cluster Anda, dan `unwrap` atributnya harus disetel ke `true`.

 Note

Perintah ini hanya tersedia dengan CloudHSM CLI 5.11+.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key unwrap rsa-aes
```

```
Usage: key unwrap rsa-aes [OPTIONS] --filter [<FILTER>...] --hash-  
function <HASH_FUNCTION> --mgf <MGF> --key-type-class <KEY_TYPE_CLASS> --label <LABEL>  
<--data-path <DATA_PATH>|--data <DATA>>
```

Options:

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

```
--filter [<FILTER>...]
```

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key to unwrap with

```
--data-path <DATA_PATH>
```

Path to the binary file containing the wrapped key data

```
--data <DATA>
```

Base64 encoded wrapped key data

```
--attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
```

Space separated list of key attributes in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key

```
--share-crypto-users [<SHARE_CRYPTO_USERS;...]
```

Space separated list of Crypto User usernames to share the unwrapped key with

```
--manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE;
```

The quorum value for key management operations for the unwrapped key

```
--use-key-quorum-value <USE_KEY_QUORUM_VALUE;
```

The quorum value for key usage operations for the unwrapped key

```
--hash-function <HASH_FUNCTION>
```

Hash algorithm [possible values: sha1, sha224, sha256, sha384, sha512]

```
--mgf <MGF>
```

Mask Generation Function algorithm [possible values: mgf1-sha1, mgf1-sha224, mgf1-sha256, mgf1-sha384, mgf1-sha512]

```
--key-type-class <KEY_TYPE_CLASS>
```

Key type and class of wrapped key [possible values: aes, des3, ec-private, generic-secret, rsa-private]

```
--label <LABEL>
```

Label for the unwrapped key

```
--session
```

Creates a session key that exists only in the current session. The key cannot be recovered after the session ends

```
--approval <APPROVAL>
```

Filepath of signed quorum token file to approve operation

```
-h, --help
    Print help
```

Contoh

Contoh-contoh ini menunjukkan bagaimana menggunakan key unwrap rsa-aes perintah menggunakan kunci pribadi RSA dengan nilai unwrap atribut yang disetel ke true.

Example Contoh: Buka kunci payload dari data kunci dibungkus yang dikodekan Base64

```
aws-cloudhsm > key unwrap rsa-aes --key-type-class aes --label aes-unwrapped
--filter attr.label=rsa-private-key-example --hash-function sha256 --
mgf mgf1-sha256 --data HrSE1DEyLjIeyGdPa9R+ebiqB5TIJGyamPker31ZebPwRA
+NcerbAJ08DJ1lXPygZcI21vIFSZJuWMEiWpe1R9D/5WSYgxLVKex30xCFqebtEzxbKuv4D0mU4meSofqREYvtb3EoIKwjy
+RL5WGxKe4nAboAkC5G07veI5yHL1SaKlssSJtTL/CFpbSLsAFuYbv/NUCWwMY5mwyVTCslw+HlgKK
+5TH1MzBaSi8fpfyepLT8sHy2Q/VRl6ifb49p6m0KQFbRVvz/OWUd6l4d97BdgtaEz6ueg==
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x0000000000001808e2",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
```

```

    "token": true,
    "always-sensitive": false,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": false,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 16
  }
}
}
}

```

Example Contoh: Buka kunci payload yang disediakan melalui jalur data

```

aws-cloudhsm > key unwrap rsa-aes --key-type-class aes --label aes-unwrapped --filter
attr.label=rsa-private-key-example --hash-function sha256 --mgf mgf1-sha256 --data-
path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x0000000000001808e2",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        }
      }
    }
  }
}

```

```
    },
    "cluster-coverage": "full"
  },
  "attributes": {
    "key-type": "aes",
    "label": "aes-unwrapped",
    "id": "0x",
    "check-value": "0x8d9099",
    "class": "secret-key",
    "encrypt": false,
    "decrypt": false,
    "token": true,
    "always-sensitive": false,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": false,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 16
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang akan dibuka.

Wajib: Ya

<DATA_PATH>

Path ke file biner yang berisi data kunci dibungkus.

Diperlukan: Ya (kecuali disediakan melalui data yang disandikan Base64)

<DATA>

Base64 mengkodekan data kunci yang dibungkus.

Wajib: Ya (kecuali disediakan melalui jalur data)

<ATTRIBUTES>

Spasi dipisahkan daftar atribut kunci dalam bentuk `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk kunci dibungkus.

Wajib: Tidak

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus [nilai yang mungkin: `aes`, `des3`, `ec-private`, `generic-secret`, `rsa-private`].

Wajib: Ya

<HASH_FUNCTION>

Menentukan fungsi hash.

Nilai valid:

- sha1
- sha224
- sha256
- sha384
- sha512

Wajib: Ya

<MGF>

Menentukan fungsi pembuatan topeng.

 Note

Fungsi hash fungsi pembuatan topeng harus sesuai dengan fungsi hash mekanisme penandatanganan.

Nilai valid:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Wajib: Ya

<LABEL>

Label untuk kunci yang tidak dibungkus.

Wajib: Ya

<SESSION>

Membuat kunci sesi yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Wajib: Tidak

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan manajemen kunci dari kunci unwrapping lebih besar dari 1.

Topik terkait

- [Perintah bungkus kunci di CloudHSM CLI](#)

- [Perintah buka kunci di CloudHSM CLI](#)

Buka kunci dengan RSA-OAEP menggunakan CloudHSM CLI

Gunakan key unwrap rsa-oaep perintah di CloudHSM CLI untuk membuka kunci payload menggunakan kunci pribadi RSA dan mekanisme unwrapping. RSA-OAEP

Kunci yang tidak dibungkus dapat digunakan dengan cara yang sama seperti kunci yang dihasilkan oleh AWS CloudHSM. Untuk menunjukkan bahwa mereka tidak dihasilkan secara lokal, `local` atribut mereka disetel ke `false`.

Untuk menggunakan key unwrap rsa-oaep perintah, Anda harus memiliki kunci pribadi RSA dari kunci pembungkus publik RSA di AWS CloudHSM cluster Anda, dan unwrap atributnya harus disetel ke `true`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key unwrap rsa-oaep
Usage: key unwrap rsa-oaep [OPTIONS] --filter [<FILTER>...] --hash-
function <HASH_FUNCTION> --mgf <MGF> --key-type-class <KEY_TYPE_CLASS> --label <LABEL>
--data-path <DATA_PATH>|--data <DATA>

Options:
  --cluster-id <CLUSTER_ID>
      Unique Id to choose which of the clusters in the config file to run the
      operation against. If not provided, will fall back to the value provided when
      interactive mode was started, or error
  --filter [<FILTER>...]
      Key reference (e.g. key-reference=0xabc) or space separated list of key
      attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
      to unwrap with
```

```

--data-path <DATA_PATH>
    Path to the binary file containing the wrapped key data
--data <DATA>
    Base64 encoded wrapped key data
--attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
    Space separated list of key attributes in the form of
    KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
--share-crypto-users [<SHARE_CRYPTO_USERS>;...]
    Space separated list of Crypto User usernames to share the unwrapped key with
--manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE>;
    The quorum value for key management operations for the unwrapped key
--use-key-quorum-value <USE_KEY_QUORUM_VALUE>;
    The quorum value for key usage operations for the unwrapped key
--hash-function <HASH_FUNCTION>
    Hash algorithm [possible values: sha1, sha224, sha256, sha384, sha512]
--mgf <MGF>
    Mask Generation Function algorithm [possible values: mgf1-sha1, mgf1-sha224,
    mgf1-sha256, mgf1-sha384, mgf1-sha512]
--key-type-class <KEY_TYPE_CLASS>
    Key type and class of wrapped key [possible values: aes, des3, ec-private,
    generic-secret, rsa-private]
--label <LABEL>
    Label for the unwrapped key
--session
    Creates a session key that exists only in the current session. The key cannot
    be recovered after the session ends
--approval <APPROVAL>
    Filepath of signed quorum token file to approve operation
-h, --help
    Print help

```

Contoh

Contoh-contoh ini menunjukkan bagaimana menggunakan key unwrap rsa-oaep perintah menggunakan kunci pribadi RSA dengan nilai unwrap atribut yang disetel ke true.

Example Contoh: Buka kunci payload dari data kunci dibungkus yang dikodekan Base64

```

aws-cloudhsm > key unwrap rsa-oaep --key-type-class aes --label aes-unwrapped --filter
attr.label=rsa-private-example-key --hash-function sha256 --mgf mgf1-sha256 --data
OjJe4msobPLz9TuSAdULEu17T5rMDWtS1LyBSkLbaZnYzzpdrhsbGLbwZJCtB/jGkDNdB4qyTA0QwEpggGf6v
+Yx6JcesNeKKNU8XZa1/YBoHC8noTGUSD12qr+u2tDc84NPv6d+F2K00NXsSxMhmxzzNG/
gzTVIJh0uy/B1yHjGP4m0XoDZf5+7f5M1CjxBmz4Vva/wrWHGCSG0y0aWb1Ev0iHAIt3UBdyKmU+/

```


My4xjfJv7WGGu3DFUUIZ06TihRtKQhUYU1M9u6NPf9riJJfHsk6QCuSZ9yWThDT9as6i7e3htnyDhIhGWaoK8JU855cN/
YNKAUqkNpC4FPL3iw==

```
{
  "data": {
    "key": {
      "key-reference": "0x000000000001808e9",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,

```

```

    "key-length-bytes": 16
  }
}
}
}

```

Example Contoh: Buka kunci payload yang disediakan melalui jalur data

```

aws-cloudhsm > key unwrap rsa-oaep --key-type-class aes --label aes-unwrapped --filter
attr.label=rsa-private-example-key --hash-function sha256 --mgf mgf1-sha256 --data-
path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000001808e9",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,

```

```

    "local": false,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 16
  }
}
}
}

```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang akan dibuka.

Wajib: Ya

<DATA_PATH>

Path ke file biner yang berisi data kunci dibungkus.

Diperlukan: Ya (kecuali disediakan melalui data yang disandikan Base64)

<DATA>

Base64 mengkodekan data kunci yang dibungkus.

Wajib: Ya (kecuali disediakan melalui jalur data)

<ATTRIBUTES>

Spasi dipisahkan daftar atribut kunci dalam bentuk

KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk kunci dibungkus.

Wajib: Tidak

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus [nilai yang mungkin:aes,des3,ec-private,generic-secret,rsa-private].

Wajib: Ya

<HASH_FUNCTION>

Menentukan fungsi hash.

Nilai valid:

- sha1
- sha224
- sha256
- sha384
- sha512

Wajib: Ya

<MGF>

Menentukan fungsi pembuatan topeng.

Note

Fungsi hash fungsi pembuatan topeng harus sesuai dengan fungsi hash mekanisme penandatanganan.

Nilai valid:

- mgf1-sha1
- mgf1-sha224

- `mgf1-sha256`
- `mgf1-sha384`
- `mgf1-sha512`

Wajib: Ya

<LABEL>

Label untuk kunci yang tidak dibungkus.

Wajib: Ya

<SESSION>

Membuat kunci sesi yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Wajib: Tidak

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan manajemen kunci dari kunci unwrapping lebih besar dari 1.

Topik terkait

- [Perintah bungkus kunci di CloudHSM CLI](#)
- [Perintah buka kunci di CloudHSM CLI](#)

Buka kunci dengan RSA-PKCS menggunakan CloudHSM CLI

Gunakan `key unwrap rsa-pkcs` perintah di CloudHSM CLI untuk membuka kunci payload menggunakan kunci pribadi RSA dan mekanisme unwrapping. `RSA-PKCS`

Kunci yang tidak dibungkus dapat digunakan dengan cara yang sama seperti kunci yang dihasilkan oleh AWS CloudHSM. Untuk menunjukkan bahwa mereka tidak dihasilkan secara lokal, `local` atribut mereka disetel ke `false`.

Untuk menggunakan `unwrap rsa-pkcs` perintah kunci, Anda harus memiliki kunci pribadi RSA dari kunci pembungkus publik RSA di AWS CloudHSM cluster Anda, dan `unwrap` atributnya harus disetel ke `true`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key unwrap rsa-pkcs
```

```
Usage: key unwrap rsa-pkcs [OPTIONS] --filter [<FILTER>...] --key-type-  
class <KEY_TYPE_CLASS> --label <LABEL> <--data-path <DATA_PATH>|--data <DATA>>
```

Options:

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

```
--filter [<FILTER>...]
```

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key to unwrap with

```
--data-path <DATA_PATH>
```

Path to the binary file containing the wrapped key data

```
--data <DATA>
```

Base64 encoded wrapped key data

```
--attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
```

Space separated list of key attributes in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key

```
--share-crypto-users [<SHARE_CRYPTO_USERS;...]
```

Space separated list of Crypto User usernames to share the unwrapped key with

```
--manage-key-quorum-value <MANAGE_KEY_QUORUM_VALUE;
```

The quorum value for key management operations for the unwrapped key

```
--use-key-quorum-value <USE_KEY_QUORUM_VALUE;
```

The quorum value for key usage operations for the unwrapped key

```
--key-type-class <KEY_TYPE_CLASS>
```

Key type and class of wrapped key [possible values: aes, des3, ec-private, generic-secret, rsa-private]

```
--label <LABEL>
```

```

    Label for the unwrapped key
--session
    Creates a session key that exists only in the current session. The key cannot
be recovered after the session ends
--approval <APPROVAL>
    Filepath of signed quorum token file to approve operation
-h, --help
    Print help

```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan key unwrap rsa-oaep perintah menggunakan kunci AES dengan nilai unwrap atribut yang disetel ke true.

Example Contoh: Buka kunci payload dari data kunci dibungkus yang dikodekan Base64

```

aws-cloudhsm > key unwrap rsa-pkcs --key-type-class aes --label
aes-unwrapped --filter attr.label=rsa-private-key-example --data
am0Nc7+YE8FWs+5HvU7sIBcXVb24QA0165nbNAD+1bK+e18BpSfnaI3P+r8Dp+pLu1ofuUy/
vtzRjZoCiDofcz4EqCFnG14GdcJ1/3W/5WRvMatCa2d7cx02swaeZcjKsermPXYR011G1fq6NskwMeeTkV8R7Rx9artFrs1
c3XdFJ2+0Bo94c6og/
yfPcp00obJlITCoXhtMRepSd040ggYq/6nUDuHCtJ86pPGnNahyr7+sAaSI3a5ECQLUjwaIARUCyoRh7EFK3qPXcg==
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000001c08ef",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "key-quorum-values": {
          "manage-key-quorum-value": 0,
          "use-key-quorum-value": 0
        },
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",

```

```

    "label": "aes-unwrapped",
    "id": "0x",
    "check-value": "0x8d9099",
    "class": "secret-key",
    "encrypt": false,
    "decrypt": false,
    "token": true,
    "always-sensitive": false,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": false,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 16
  }
}
}
}

```

Example Contoh: Buka kunci payload yang disediakan melalui jalur data

```

aws-cloudhsm > key unwrap rsa-pkcs --key-type-class aes --label aes-unwrapped --filter
attr.label=rsa-private-key-example --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x000000000001c08ef",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ]
      }
    }
  }
}

```



```

    ],
    "shared-users": [],
    "key-quorum-values": {
        "manage-key-quorum-value": 0,
        "use-key-quorum-value": 0
    },
    "cluster-coverage": "full"
},
"attributes": {
    "key-type": "aes",
    "label": "aes-unwrapped",
    "id": "0x",
    "check-value": "0x8d9099",
    "class": "secret-key",
    "encrypt": false,
    "decrypt": false,
    "token": true,
    "always-sensitive": false,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": false,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 16
}
}
}
}

```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau daftar atribut kunci yang dipisahkan spasi dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci yang akan dibuka.

Wajib: Ya

<DATA_PATH>

Path ke file biner yang berisi data kunci yang dibungkus.

Diperlukan: Ya (kecuali disediakan melalui data yang disandikan Base64)

<DATA>

Base64 mengkodekan data kunci yang dibungkus.

Wajib: Ya (kecuali disediakan melalui jalur data)

<ATTRIBUTES>

Spasi dipisahkan daftar atribut kunci dalam bentuk `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk kunci dibungkus.

Wajib: Tidak

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus [nilai yang mungkin: `aes`, `des3`, `ec-private`, `generic-secret`, `rsa-private`].

Wajib: Ya

<LABEL>

Label untuk kunci yang tidak dibungkus.

Wajib: Ya

<SESSION>

Membuat kunci sesi yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Wajib: Tidak

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan manajemen kunci dari kunci unwrapping lebih besar dari 1.

Topik terkait

- [Perintah bungkus kunci di CloudHSM CLI](#)
- [Perintah buka kunci di CloudHSM CLI](#)

Perintah bungkus kunci di CloudHSM CLI

key wrapPerintah di CloudHSM CLI mengeksport salinan terenkripsi kunci pribadi simetris atau asimetris dari modul keamanan perangkat keras (HSM) ke file. Ketika Anda menjalankankey wrap, Anda menentukan dua hal: Kunci untuk mengeksport dan file output. Kunci untuk mengeksport adalah kunci pada HSM yang akan mengenkripsi (membungkus) kunci yang ingin Anda ekspor.

key wrapPerintah tidak menghapus kunci dari HSM atau mencegah Anda menggunakannya dalam operasi kriptografi. Anda dapat mengeksport kunci yang sama beberapa kali. Untuk mengimpor kunci terenkripsi kembali ke HSM, gunakan [Perintah buka kunci di CloudHSM CLI](#) Hanya pemilik kunci, yaitu pengguna kriptografi (CU) yang membuat kunci, yang dapat membungkus kunci tersebut. Pengguna dengan siapa kunci dibagikan hanya dapat menggunakan kunci dalam operasi kriptografi.

key wrapPerintah terdiri dari subperintah berikut:

- [aes-gcm](#)
- [aes-no-pad](#)
- [aes-pkcs5-pad](#)
- [aes-zero-pad](#)
- [cloudhsm-aes-gcm](#)
- [rsa-aes](#)
- [rsa-oaep](#)
- [rsa-PKC](#)

Bungkus kunci dengan AES-GCM menggunakan CloudHSM CLI

Gunakan `key wrap aes-gcm` perintah di CloudHSM CLI untuk membungkus kunci payload menggunakan kunci AES pada modul keamanan perangkat keras (HSM) dan mekanisme pembungkus. AES-GCM `extractable` atribut kunci payload harus disetel ke `true`.

Hanya pemilik kunci, yaitu pengguna kriptografi (CU) yang membuat kunci, yang dapat membungkus kunci tersebut. Pengguna yang berbagi kunci dapat menggunakan kunci dalam operasi kriptografi.

Untuk menggunakan `key wrap aes-gcm` perintah, Anda harus terlebih dahulu memiliki kunci AES di AWS CloudHSM cluster Anda. Anda dapat menghasilkan kunci AES untuk membungkus dengan [Hasilkan kunci AES simetris dengan CloudHSM CLI](#) perintah dan `wrap` atribut yang disetel ke `true`.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key wrap aes-gcm
```

```
Usage: key wrap aes-gcm [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-  
filter [<WRAPPING_FILTER>...] --tag-length-bits <TAG_LENGTH_BITS>
```

Options:

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

```
--payload-filter [<PAYLOAD_FILTER>...]
```

Key reference (e.g. `key-reference=0xabc`) or space separated list of key attributes in the form of `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` to select a payload key

```
--wrapping-filter [<WRAPPING_FILTER>...]
```

Key reference (e.g. `key-reference=0xabc`) or space separated list of key attributes in the form of `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` to select a wrapping key

```

--path <PATH>
    Path to the binary file where the wrapped key data will be saved
--wrapping-approval <WRAPPING_APPROVALR>
    File path of signed quorum token file to approve operation for wrapping key
--payload-approval <PAYLOAD_APPROVALR>
    File path of signed quorum token file to approve operation for payload key
--aad <AAD>
    Aes GCM Additional Authenticated Data (AAD) value, in hex
--tag-length-bits <TAG_LENGTH_BITS>
    Aes GCM tag length in bits
-h, --help
    Print help

```

Contoh

Contoh ini menunjukkan cara menggunakan key wrap aes-gcm perintah menggunakan kunci AES.

Example

```

aws-cloudhsm > key wrap aes-gcm --payload-filter attr.label=payload-key --wrapping-
filter attr.label=aes-example --tag-length-bits 64 --aad 0x10
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x000000000001c08f1",
    "wrapping_key_reference": "0x000000000001c08ea",
    "iv": "0xf90613bb8e337ec0339aad21",
    "wrapped_key_data": "xvslgrtg8kHrzvekny97tLSIeokpPwV8"
  }
}

```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<PAYLOAD_FILTER>

Referensi kunci (misalnya, key-reference=0xabc) atau spasi dipisahkan daftar atribut kunci dalam bentuk attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk memilih kunci payload.

Wajib: Ya

<PATH>

Path ke file biner tempat data kunci yang dibungkus akan disimpan.

Wajib: Tidak

<WRAPPING_FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau spasi dipisahkan daftar atribut kunci dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci pembungkus.

Wajib: Ya

<AAD>

Nilai AES GCM Additional Authenticated Data (AAD), dalam hex.

Wajib: Tidak

<TAG_LENGTH_BITS>

Panjang tag AES GCM dalam bit.

Wajib: Ya

<WRAPPING_APPROVALR>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi untuk kunci pembungkus. Hanya diperlukan jika nilai kuorum layanan manajemen kunci kunci wrapping lebih besar dari 1.

<PAYLOAD_APPROVALR>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi kunci payload. Hanya diperlukan jika nilai kuorum layanan manajemen kunci payload key lebih besar dari 1.

Topik terkait

- [Perintah bungkus kunci di CloudHSM CLI](#)

- [Perintah buka kunci di CloudHSM CLI](#)

Bungkus kunci dengan AES-NO-PAD menggunakan CloudHSM CLI

Gunakan key wrap aes-no-pad perintah di CloudHSM CLI untuk membungkus kunci payload menggunakan kunci AES pada modul keamanan perangkat keras (HSM) dan mekanisme pembungkus. AES-NO-PAD extractableAtribut kunci payload harus disetel ke true.

Hanya pemilik kunci, yaitu pengguna kriptografi (CU) yang membuat kunci, yang dapat membungkus kunci tersebut. Pengguna yang berbagi kunci dapat menggunakan kunci dalam operasi kriptografi.

Untuk menggunakan key wrap aes-no-pad perintah, Anda harus terlebih dahulu memiliki kunci AES di AWS CloudHSM cluster Anda. Anda dapat menghasilkan kunci AES untuk pembungkus menggunakan [Hasilkan kunci AES simetris dengan CloudHSM CLI](#) perintah dan wrap atribut yang disetel ke. true

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key wrap aes-no-pad
```

```
Usage: key wrap aes-no-pad [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-filter [<WRAPPING_FILTER>...]
```

Options:

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

```
--payload-filter [<PAYLOAD_FILTER>...]
```

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a payload key

```

--wrapping-filter [<WRAPPING_FILTER>...]
    Key reference (e.g. key-reference=0xabc) or space separated list of key
    attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
    wrapping key
--path <PATH>
    Path to the binary file where the wrapped key data will be saved
--wrapping-approval <WRAPPING_APPROVAL>
    File path of signed quorum token file to approve operation for wrapping key
--payload-approval <PAYLOAD_APPROVAL>
    File path of signed quorum token file to approve operation for payload key
-h, --help
    Print help

```

Contoh

Contoh ini menunjukkan cara menggunakan key wrap aes-no-pad perintah menggunakan kunci AES dengan nilai wrap atribut yang disetel ke true.

Example

```

aws-cloudhsm > key wrap aes-no-pad --payload-filter attr.label=payload-key --wrapping-
filter attr.label=aes-example
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x000000000001c08f1",
    "wrapping_key_reference": "0x000000000001c08ea",
    "wrapped_key_data": "eXK3PMA0nKM9y3YX6brbhtMoC060E0H9"
  }
}

```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<PAYLOAD_FILTER>

Referensi kunci (misalnya, key-reference=0xabc) atau spasi dipisahkan daftar atribut kunci dalam bentuk attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk memilih kunci payload.

Wajib: Ya

<PATH>

Path ke file biner tempat data kunci yang dibungkus akan disimpan.

Wajib: Tidak

<WRAPPING_FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau spasi dipisahkan daftar atribut kunci dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci pembungkus.

Wajib: Ya

<WRAPPING_APPROVALR>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi untuk kunci pembungkus. Hanya diperlukan jika nilai kuorum layanan manajemen kunci kunci wrapping lebih besar dari 1.

<PAYLOAD_APPROVALR>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi kunci payload. Hanya diperlukan jika nilai kuorum layanan manajemen kunci payload key lebih besar dari 1.

Topik terkait

- [Perintah bungkus kunci di CloudHSM CLI](#)
- [Perintah buka kunci di CloudHSM CLI](#)

Bungkus kunci dengan AES- PKCS5 -PAD menggunakan CloudHSM CLI

Gunakan `key wrap aes-pkcs5-pad` perintah di CloudHSM CLI untuk membungkus kunci payload menggunakan kunci AES pada modul keamanan perangkat keras (HSM) dan mekanisme pembungkus. `AES-PKCS5-PAD extractableAtribut` kunci payload harus disetel ke `true`.

Hanya pemilik kunci, yaitu pengguna kriptografi (CU) yang membuat kunci, yang dapat membungkus kunci tersebut. Pengguna yang berbagi kunci dapat menggunakan kunci dalam operasi kriptografi.

Untuk menggunakan key wrap aes-pkcs5-pad perintah, Anda harus terlebih dahulu memiliki kunci AES di AWS CloudHSM cluster Anda. Anda dapat menghasilkan kunci AES untuk pembungkus menggunakan [Hasilkan kunci AES simetris dengan CloudHSM CLI](#) perintah dan wrap atribut yang disetel ke. true

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key wrap aes-pkcs5-pad
Usage: key wrap aes-pkcs5-pad [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --
wrapping-filter [<WRAPPING_FILTER>...]

Options:
  --cluster-id <CLUSTER_ID>
      Unique Id to choose which of the clusters in the config file to run the
      operation against. If not provided, will fall back to the value provided when
      interactive mode was started, or error
  --payload-filter [<PAYLOAD_FILTER>...]
      Key reference (e.g. key-reference=0xabc) or space separated list of key
      attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
      payload key
  --wrapping-filter [<WRAPPING_FILTER>...]
      Key reference (e.g. key-reference=0xabc) or space separated list of key
      attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
      wrapping key
  --path <PATH>
      Path to the binary file where the wrapped key data will be saved
  --wrapping-approval <WRAPPING_APPROVALR>
      File path of signed quorum token file to approve operation for wrapping key
  --payload-approval <PAYLOAD_APPROVALR>
      File path of signed quorum token file to approve operation for payload key
  -h, --help
```

[Print help](#)

Contoh

Contoh ini menunjukkan cara menggunakan key wrap aes-pkcs5-pad perintah menggunakan kunci AES dengan nilai wrap atribut yang disetel ke true.

Example

```
aws-cloudhsm > key wrap aes-pkcs5-pad --payload-filter attr.label=payload-key --
wrapping-filter attr.label=aes-example
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x000000000001c08f1",
    "wrapping_key_reference": "0x000000000001c08ea",
    "wrapped_key_data": "MbuYNresf0KyGNnxKWen88nSfX+uUE/0qmGofSisicY="
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<PAYLOAD_FILTER>

Referensi kunci (misalnya, key-reference=0xabc) atau spasi dipisahkan daftar atribut kunci dalam bentuk attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk memilih kunci payload.

Wajib: Ya

<PATH>

Path ke file biner tempat data kunci yang dibungkus akan disimpan.

Wajib: Tidak

<WRAPPING_FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau spasi dipisahkan daftar atribut kunci dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci pembungkus.

Wajib: Ya

<WRAPPING_APPROVALR>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi untuk kunci pembungkus. Hanya diperlukan jika nilai kuorum layanan manajemen kunci kunci wrapping lebih besar dari 1.

<PAYLOAD_APPROVALR>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi kunci payload. Hanya diperlukan jika nilai kuorum layanan manajemen kunci payload key lebih besar dari 1.

Topik terkait

- [Perintah bungkus kunci di CloudHSM CLI](#)
- [Perintah buka kunci di CloudHSM CLI](#)

Bungkus kunci dengan AES-ZERO-PAD menggunakan CloudHSM CLI

Gunakan `key wrap aes-zero-pad` perintah di CloudHSM CLI untuk membungkus kunci payload menggunakan kunci AES pada modul keamanan perangkat keras HSM) dan mekanisme pembungkus. `AES-ZERO-PAD extractableAtribut` kunci payload harus disetel ke `true`.

Hanya pemilik kunci, yaitu pengguna kriptografi (CU) yang membuat kunci, yang dapat membungkus kunci tersebut. Pengguna yang berbagi kunci dapat menggunakan kunci dalam operasi kriptografi.

Untuk menggunakan `key wrap aes-zero-pad` perintah, Anda harus terlebih dahulu memiliki kunci AES di AWS CloudHSM cluster Anda. Anda dapat menghasilkan kunci AES untuk membungkus menggunakan [Hasilkan kunci AES simetris dengan CloudHSM CLI](#) perintah dengan `wrap` atribut yang disetel ke `true`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key wrap aes-zero-pad
```

```
Usage: key wrap aes-zero-pad [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --  
wrapping-filter [<WRAPPING_FILTER>...]
```

Options:

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

```
--payload-filter [<PAYLOAD_FILTER>...]
```

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a payload key

```
--wrapping-filter [<WRAPPING_FILTER>...]
```

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a wrapping key

```
--path <PATH>
```

Path to the binary file where the wrapped key data will be saved

```
--wrapping-approval <WRAPPING_APPROVALR>
```

File path of signed quorum token file to approve operation for wrapping key

```
--payload-approval <PAYLOAD_APPROVALR>
```

File path of signed quorum token file to approve operation for payload key

```
-h, --help
```

Print help

Contoh

Contoh ini menunjukkan cara menggunakan key wrap aes-zero-pad perintah menggunakan kunci AES dengan nilai wrap atribut yang disetel ke true.

Example

```
aws-cloudhsm > key wrap aes-zero-pad --payload-filter attr.label=payload-key --
wrapping-filter attr.label=aes-example
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x000000000001c08f1",
    "wrapping_key_reference": "0x000000000001c08ea",
    "wrapped_key_data": "L1wV1L/YeBNVAw6Mpk3owFJZXBzDL0Nt"
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<PAYLOAD_FILTER>

Referensi kunci (misalnya, key-reference=0xabc) atau spasi dipisahkan daftar atribut kunci dalam bentuk attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk memilih kunci payload.

Wajib: Ya

<PATH>

Path ke file biner tempat data kunci yang dibungkus akan disimpan.

Wajib: Tidak

<WRAPPING_FILTER>

Referensi kunci (misalnya, key-reference=0xabc) atau spasi dipisahkan daftar atribut kunci dalam bentuk attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk memilih kunci pembungkus.

Wajib: Ya

<WRAPPING_APPROVALR>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi untuk kunci pembungkus. Hanya diperlukan jika nilai kuorum layanan manajemen kunci kunci wrapping lebih besar dari 1.

<PAYLOAD_APPROVALR>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi kunci payload. Hanya diperlukan jika nilai kuorum layanan manajemen kunci payload key lebih besar dari 1.

Topik terkait

- [Perintah bungkus kunci di CloudHSM CLI](#)
- [Perintah buka kunci di CloudHSM CLI](#)

Bungkus kunci dengan CLOUDHSM-AES-GCM menggunakan CloudHSM CLI

Gunakan key wrap cloudhsm-aes-gcm perintah di CloudHSM CLI untuk membungkus kunci payload menggunakan kunci AES pada modul keamanan perangkat keras (HSM) dan mekanisme pembungkus. CLOUDHSM-AES-GCM extractableAtribut kunci payload harus disetel ke true.

Hanya pemilik kunci, yaitu pengguna kriptografi (CU) yang membuat kunci, yang dapat membungkus kunci tersebut. Pengguna yang berbagi kunci dapat menggunakan kunci dalam operasi kriptografi.

Untuk menggunakan key wrap cloudhsm-aes-gcm perintah, Anda harus terlebih dahulu memiliki kunci AES di AWS CloudHSM cluster Anda. Anda dapat menghasilkan kunci AES untuk membungkus dengan [Hasilkan kunci AES simetris dengan CloudHSM CLI](#) perintah dan wrap atribut yang disetel ke. true

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key wrap cloudhsm-aes-gcm
```

```
Usage: key wrap cloudhsm-aes-gcm [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --
wrapping-filter [<WRAPPING_FILTER>...] --tag-length-bits <TAG_LENGTH_BITS>
```

Options:

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

```
--payload-filter [<PAYLOAD_FILTER>...]
```

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a payload key

```
--wrapping-filter [<WRAPPING_FILTER>...]
```

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a wrapping key

```
--path <PATH>
```

Path to the binary file where the wrapped key data will be saved

```
--wrapping-approval <WRAPPING_APPROVAL>
```

File path of signed quorum token file to approve operation for wrapping key

```
--payload-approval <PAYLOAD_APPROVAL>
```

File path of signed quorum token file to approve operation for payload key

```
--aad <AAD>
```

Aes GCM Additional Authenticated Data (AAD) value, in hex

```
--tag-length-bits <TAG_LENGTH_BITS>
```

Aes GCM tag length in bits

```
-h, --help
```

Print help

Contoh

Contoh ini menunjukkan cara menggunakan key wrap cloudhsm-aes-gcm perintah menggunakan kunci AES.

Example

```
aws-cloudhsm > key wrap cloudhsm-aes-gcm --payload-filter attr.label=payload-key --
wrapping-filter attr.label=aes-example --tag-length-bits 64 --aad 0x10
{
  "error_code": 0,
```



```
"data": {  
  "payload_key_reference": "0x000000000001c08f1",  
  "wrapping_key_reference": "0x000000000001c08ea",  
  "wrapped_key_data": "6Rn8nkjEriDYlnP3P8nPkyQ8hpl0EJ899zsrF+aTB0i/fIlZ"  
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<PAYLOAD_FILTER>

Referensi kunci (misalnya, key-reference=0xabc) atau spasi dipisahkan daftar atribut kunci dalam bentuk attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk memilih kunci payload.

Wajib: Ya

<PATH>

Path ke file biner tempat data kunci yang dibungkus akan disimpan.

Wajib: Tidak

<WRAPPING_FILTER>

Referensi kunci (misalnya, key-reference=0xabc) atau spasi dipisahkan daftar atribut kunci dalam bentuk attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk memilih kunci pembungkus.

Wajib: Ya

<AAD>

Nilai AES GCM Additional Authenticated Data (AAD), dalam hex.

Wajib: Tidak

<TAG_LENGTH_BITS>

Panjang tag AES GCM dalam bit.

Wajib: Ya

<WRAPPING_APPROVALR>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi untuk kunci pembungkus. Hanya diperlukan jika nilai kuorum layanan manajemen kunci kunci wrapping lebih besar dari 1.

<PAYLOAD_APPROVALR>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi kunci payload. Hanya diperlukan jika nilai kuorum layanan manajemen kunci payload key lebih besar dari 1.

Topik terkait

- [Perintah bungkus kunci di CloudHSM CLI](#)
- [Perintah buka kunci di CloudHSM CLI](#)

Bungkus kunci dengan RSA-AES menggunakan CloudHSM CLI

Gunakan key wrap rsa-aes perintah di CloudHSM CLI untuk membungkus kunci payload menggunakan kunci publik RSA pada modul keamanan perangkat keras (HSM) dan mekanisme pembungkus RSA-AES. `extractableAtribut` kunci payload harus disetel ke `true`.

Hanya pemilik kunci, yaitu pengguna kriptografi (CU) yang membuat kunci, yang dapat membungkus kunci tersebut. Pengguna yang berbagi kunci dapat menggunakan kunci dalam operasi kriptografi.

Untuk menggunakan key wrap rsa-aes perintah, Anda harus terlebih dahulu memiliki kunci RSA di AWS CloudHSM cluster Anda. Anda dapat menghasilkan key pair RSA menggunakan [generate-asymmetric-pair](#) [Kategori di CloudHSM CLI](#) perintah dan `wrap` atribut yang disetel ke `true`.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key wrap rsa-aes
```

```
Usage: key wrap rsa-aes [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-  
filter [<WRAPPING_FILTER>...] --hash-function <HASH_FUNCTION> --mgf <MGF>
```

Options:

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

```
--payload-filter [<PAYLOAD_FILTER>...]
```

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a payload key

```
--wrapping-filter [<WRAPPING_FILTER>...]
```

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a wrapping key

```
--path <PATH>
```

Path to the binary file where the wrapped key data will be saved

```
--wrapping-approval <WRAPPING_APPROVALR>
```

File path of signed quorum token file to approve operation for wrapping key

```
--payload-approval <PAYLOAD_APPROVALR>
```

File path of signed quorum token file to approve operation for payload key

```
--hash-function <HASH_FUNCTION>
```

Hash algorithm [possible values: sha1, sha224, sha256, sha384, sha512]

```
--mgf <MGF>
```

Mask Generation Function algorithm [possible values: mgf1-sha1, mgf1-sha224, mgf1-sha256, mgf1-sha384, mgf1-sha512]

```
-h, --help
```

Print help

Contoh

Contoh ini menunjukkan cara menggunakan key wrap rsa-ae perintah menggunakan kunci publik RSA dengan nilai wrap atribut yang disetel ke true.

Example

```
aws-cloudhsm > key wrap rsa-aes --payload-filter attr.label=payload-key --wrapping-  
filter attr.label=rsa-public-key-example --hash-function sha256 --mgf mgf1-sha256  
{
```

```

"error_code": 0,
"data": {
  "payload-key-reference": "0x000000000001c08f1",
  "wrapping-key-reference": "0x000000000007008da",
  "wrapped-key-data": "HrSE1DEyLjIeyGdPa9R+ebiqB5TIJGyamPker31ZebPwRA
+NcerbAJ08DJ1lXPYgZcI21vIFSZJuWMEiWpe1R9D/5WSYgxLVKex30xCFqebtEzxbKuv4D0mU4meSofqREYvtb3EoIKwjy
+RL5WGXXKe4nAboAkC5G07veI5yHL1SaKlssSJtTL/CFpbSLsAFuYbv/NUCWwMY5mwyVTCSlw+HlgKK
+5TH1MzBaSi8fpfyepLT8sHy2Q/VRl6ifb49p6m0KQFbRVvz/0WUd6l4d97BdgtaEz6ueg=="
}
}

```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<PAYLOAD_FILTER>

Referensi kunci (misalnya, key-reference=0xabc) atau spasi dipisahkan daftar atribut kunci dalam bentuk attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk memilih kunci payload.

Wajib: Ya

<PATH>

Path ke file biner tempat data kunci yang dibungkus akan disimpan.

Wajib: Tidak

<WRAPPING_FILTER>

Referensi kunci (misalnya, key-reference=0xabc) atau spasi dipisahkan daftar atribut kunci dalam bentuk attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk memilih kunci pembungkus.

Wajib: Ya

<MGF>

Menentukan fungsi pembuatan topeng.

 Note

Fungsi hash fungsi pembuatan topeng harus sesuai dengan fungsi hash mekanisme penandatanganan.

Nilai valid

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Wajib: Ya

<WRAPPING_APPROVALR>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi untuk kunci pembungkus. Hanya diperlukan jika nilai kuorum layanan manajemen kunci kunci wrapping lebih besar dari 1.

<PAYLOAD_APPROVALR>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi kunci payload. Hanya diperlukan jika nilai kuorum layanan manajemen kunci payload key lebih besar dari 1.

Topik terkait

- [Perintah bungkus kunci di CloudHSM CLI](#)
- [Perintah buka kunci di CloudHSM CLI](#)

Bungkus kunci dengan RSA-OAEP menggunakan CloudHSM CLI

Gunakan key wrap rsa-oaep perintah di CloudHSM CLI untuk membungkus kunci payload menggunakan kunci publik RSA pada modul keamanan perangkat keras (HSM) dan mekanisme pembungkus. RSA-OAEP extractableAtribut kunci payload harus disetel ke true.

Hanya pemilik kunci, yaitu pengguna kriptografi (CU) yang membuat kunci, yang dapat membungkus kunci tersebut. Pengguna yang berbagi kunci dapat menggunakan kunci dalam operasi kriptografi.

Untuk menggunakan key wrap rsa-oeap perintah, Anda harus terlebih dahulu memiliki kunci RSA di AWS CloudHSM cluster Anda. Anda dapat menghasilkan key pair RSA menggunakan [generate-asymmetric-pair](#) Kategori di CloudHSM CLI perintah dan wrap atribut yang disetel ke true.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key wrap rsa-oeap
```

```
Usage: key wrap rsa-oeap [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-filter [<WRAPPING_FILTER>...] --hash-function <HASH_FUNCTION> --mgf <MGF>
```

Options:

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

```
--payload-filter [<PAYLOAD_FILTER>...]
```

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a payload key

```
--wrapping-filter [<WRAPPING_FILTER>...]
```

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a wrapping key

```
--path <PATH>
```

Path to the binary file where the wrapped key data will be saved

```
--wrapping-approval <WRAPPING_APPROVAL>
```

File path of signed quorum token file to approve operation for wrapping key

```
--payload-approval <PAYLOAD_APPROVAL>
```

File path of signed quorum token file to approve operation for payload key

```
--hash-function <HASH_FUNCTION>
    Hash algorithm [possible values: sha1, sha224, sha256, sha384, sha512]
--mgf <MGF>
    Mask Generation Function algorithm [possible values: mgf1-sha1, mgf1-sha224,
mgf1-sha256, mgf1-sha384, mgf1-sha512]
-h, --help
    Print help
```

Contoh

Contoh ini menunjukkan cara menggunakan key wrap rsa-oaep perintah menggunakan kunci publik RSA dengan nilai wrap atribut yang disetel ke true.

Example

```
aws-cloudhsm > key wrap rsa-oaep --payload-filter attr.label=payload-key --wrapping-
filter attr.label=rsa-public-key-example --hash-function sha256 --mgf mgf1-sha256
{
  "error_code": 0,
  "data": {
    "payload-key-reference": "0x000000000001c08f1",
    "wrapping-key-reference": "0x000000000007008da",
    "wrapped-key-data": "0jJe4msobPLz9TuSAdULEu17T5rMDWtS1LyBSkLbaZnYzzpdrhsbGLbwZJCtB/
jGkDNdB4qyTA0QwEpggGf6v+Yx6JcesNeKKNu8XZa1/YBoHC8noTGUSDI2qr+u2tDc84NPv6d
+F2K00NXsSxMhmxzzNG/gzTVIJh0uy/B1yHjGP4m0XoDZf5+7f5M1CjxBmz4Vva/
wrWHGCSG0y0aWblEv0iHAIt3UBdyKmU+/
My4xjfJv7WGGu3DFUUIZ06TihRtKQhUYU1M9u6NPf9riJJfHsk6QCuSZ9yWThDT9as6i7e3htnyDhIhGWaoK8JU855cN/
YNKAUqkNpC4FPL3iw=="
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<PAYLOAD_FILTER>

Referensi kunci (misalnya, key-reference=0xabc) atau spasi dipisahkan daftar atribut kunci dalam bentuk attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE untuk memilih kunci payload.

Wajib: Ya

<PATH>

Path ke file biner tempat data kunci yang dibungkus akan disimpan.

Wajib: Tidak

<WRAPPING_FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau spasi dipisahkan daftar atribut kunci dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci pembungkus.

Wajib: Ya

<MGF>

Menentukan fungsi pembuatan topeng.

Note

Fungsi hash fungsi pembuatan topeng harus sesuai dengan fungsi hash mekanisme penandatanganan.

Nilai valid

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Wajib: Ya

<WRAPPING_APPROVALR>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi untuk kunci pembungkus. Hanya diperlukan jika nilai kuorum layanan manajemen kunci kunci wrapping lebih besar dari 1.

<PAYLOAD_APPROVALR>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi kunci payload. Hanya diperlukan jika nilai kuorum layanan manajemen kunci payload key lebih besar dari 1.

Topik terkait

- [Perintah bungkus kunci di CloudHSM CLI](#)
- [Perintah buka kunci di CloudHSM CLI](#)

Bungkus kunci dengan RSA-PKCS menggunakan CloudHSM CLI

Gunakan key wrap rsa-pkcs perintah di CloudHSM CLI untuk membungkus kunci payload menggunakan kunci publik RSA pada modul keamanan perangkat keras (HSM) dan mekanisme pembungkus. RSA-PKCS extractableAtribut kunci payload harus disetel ke true.

Hanya pemilik kunci, yaitu pengguna kriptografi (CU) yang membuat kunci, yang dapat membungkus kunci tersebut. Pengguna yang berbagi kunci dapat menggunakan kunci dalam operasi kriptografi.

Untuk menggunakan key wrap rsa-pkcs perintah, Anda harus terlebih dahulu memiliki kunci RSA di AWS CloudHSM cluster Anda. Anda dapat menghasilkan key pair RSA menggunakan [generate-asymmetric-pair](#) [Kategori di CloudHSM CLI](#) perintah dan wrap atribut yang disetel ke true.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CUs)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key wrap rsa-pkcs
Usage: key wrap rsa-pkcs [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-
filter [<WRAPPING_FILTER>...]
```

Options:**--cluster-id** *<CLUSTER_ID>*

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

--payload-filter [*<PAYLOAD_FILTER>*...]

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a payload key

--wrapping-filter [*<WRAPPING_FILTER>*...]

Key reference (e.g. key-reference=0xabc) or space separated list of key attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a wrapping key

--path *<PATH>*

Path to the binary file where the wrapped key data will be saved

--wrapping-approval *<WRAPPING_APPROVALR>*

File path of signed quorum token file to approve operation for wrapping key

--payload-approval *<PAYLOAD_APPROVALR>*

File path of signed quorum token file to approve operation for payload key

-h, --help

Print help

Contoh

Contoh ini menunjukkan cara menggunakan key wrap rsa-pkcs perintah menggunakan kunci publik RSA.

Example

```
aws-cloudhsm > key wrap rsa-pkcs --payload-filter attr.label=payload-key --wrapping-
filter attr.label=rsa-public-key-example
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x000000000001c08f1",
    "wrapping_key_reference": "0x00000000007008da",
    "wrapped_key_data": "am0Nc7+YE8FWs+5HvU7sIBcXVb24QA0l65nbNAD+1bK+e18BpSfnaI3P+r8Dp
+pLu1ofoUy/
vtzRjZoCiDofcz4EqCFnG14GdcJ1/3W/5WRvMatCa2d7cx02swaeZcjKsermPXYR01lG1fq6NskwMeeTkV8R7Rx9artFrs1
c3XdFJ2+0Bo94c6og/
yfPcp00obJlITCoXhtMRepSd040ggYq/6nUDuHCtJ86pPGnNahyr7+sAaSI3a5ECQLUjwaIARUCyoRh7EFK3qPXcg=="
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<PAYLOAD_FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau spasi dipisahkan daftar atribut kunci dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci payload.

Wajib: Ya

<PATH>

Path ke file biner tempat data kunci yang dibungkus akan disimpan.

Wajib: Tidak

<WRAPPING_FILTER>

Referensi kunci (misalnya, `key-reference=0xabc`) atau spasi dipisahkan daftar atribut kunci dalam bentuk `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` untuk memilih kunci pembungkus.

Wajib: Ya

<WRAPPING_APPROVALR>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi untuk kunci pembungkus. Hanya diperlukan jika nilai kuorum layanan manajemen kunci kunci wrapping lebih besar dari 1.

<PAYLOAD_APPROVALR>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi kunci payload. Hanya diperlukan jika nilai kuorum layanan manajemen kunci payload key lebih besar dari 1.

Topik terkait

- [Perintah bungkus kunci di CloudHSM CLI](#)

- [Perintah buka kunci di CloudHSM CLI](#)

Masuk ke HSM menggunakan CloudHSM CLI

Anda dapat menggunakan login perintah di CloudHSM CLI untuk masuk dan keluar dari setiap keamanan perangkat keras (HSM) dalam sebuah cluster. AWS CloudHSM Perintah ini memiliki sub-perintah berikut:

- [mfa-token-sign](#)

Note

Jika Anda melebihi lima upaya masuk yang salah, akun Anda terkunci. Untuk membuka kunci akun, admin harus mengatur ulang kata sandi Anda menggunakan perintah [ubah kata sandi pengguna di cloudhsm_cli](#).

Untuk memecahkan masalah login dan logout

Jika Anda memiliki lebih dari satu HSM di klaster Anda, Anda mungkin diizinkan untuk upaya login salah tambahan sebelum akun Anda terkunci. Ini karena klien CloudHSM menyeimbangkan beban di berbagai jenis. HSMs Oleh karena itu, upaya login mungkin tidak dimulai pada HSM yang sama setiap saat. Jika Anda menguji fungsionalitas ini, kami sarankan Anda melakukannya pada sebuah klaster dengan hanya satu HSM aktif.

Jika Anda membuat klaster sebelum Februari 2018, akun Anda terkunci setelah 20 upaya login salah.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah ini.

- Admin yang tidak aktif
- Admin
- Pengguna krypto (CU)

Sintaksis

```
aws-cloudhsm > help login
```

Login to your cluster

USAGE:

```
cloudhsm-cli login [OPTIONS] --username <USERNAME> --role <ROLE> [COMMAND]
```

Commands:

```
mfa-token-sign  Login with token-sign mfa
help           Print this message or the help of the given subcommand(s)
```

OPTIONS:

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

```
--username <USERNAME>
```

Username to access the Cluster

```
--role <ROLE>
```

Role the user has in the Cluster

Possible values:

- crypto-user: A CryptoUser has the ability to manage and use keys
- admin: An Admin has the ability to manage user accounts

```
--password <PASSWORD>
```

Optional: Plaintext user's password. If you do not include this argument you will be prompted for it

```
-h, --help
```

Print help (see a summary with '-h')

Contoh

Example

Perintah ini mencatat Anda ke semua HSMs dalam kluster dengan kredensi pengguna admin bernama. admin1

```
aws-cloudhsm > login --username admin1 --role admin
Enter password:
{
```

```
"error_code": 0,  
"data": {  
  "username": "admin1",  
  "role": "admin"  
}  
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<USERNAME>

Menentukan nama yang mudah diingat untuk pengguna. Panjang maksimum adalah 31 karakter. Satu-satunya karakter khusus yang diizinkan adalah garis bawah (_). Nama pengguna tidak peka huruf besar/kecil dalam perintah ini, nama pengguna selalu ditampilkan dalam huruf kecil.

Wajib: Ya

<ROLE>

Menentukan peran yang ditetapkan untuk pengguna ini. Nilai yang valid adalah admin, crypto-user.

Untuk mendapatkan peran pengguna, gunakan user list perintah. Untuk informasi rinci tentang jenis pengguna di HSM, lihat [Memahami pengguna HSM](#).

Wajib: Ya

<PASSWORD>

Menentukan password dari pengguna yang masuk ke. HSMs

Wajib: Pelanggan harus memberikan kata sandi baik melalui argumen baris --password perintah atau menghilangkannya untuk diminta secara interaktif.

Note

Menggunakan --password argumen pada baris perintah dapat mengekspos kredensial dalam riwayat shell Anda. Pertimbangkan untuk meminta kata sandi secara interaktif atau menghapus riwayat shell Anda setelah digunakan.

Topik terkait

- [Memulai dengan CloudHSM CLI](#)
- [Aktifkan Cluster](#)

Masuk dengan MFA ke HSM menggunakan CloudHSM CLI

Gunakan login mfa-token-sign perintah di AWS CloudHSM CloudHSM CLI untuk masuk ke modul keamanan perangkat keras (HSM) menggunakan otentikasi multi-faktor (MFA). Untuk menggunakan perintah ini, Anda harus terlebih dahulu mengatur [MFA untuk CloudHSM CLI](#).

Jenis pengguna

Pengguna berikut dapat menjalankan perintah ini.

- Admin
- Pengguna krypto (CU)

Sintaksis

```
aws-cloudhsm > help login mfa-token-sign
```

```
Login with token-sign mfa
```

```
USAGE:
```

```
login --username <username> --role <role> mfa-token-sign --token <token>
```

```
OPTIONS:
```

```
--cluster-id <CLUSTER_ID> Unique Id to choose which of the clusters in the  
config file to run the operation against. If not provided, will fall back to the value  
provided when interactive mode was started, or error
```

```
--token <TOKEN> Filepath where the unsigned token file will be written
```

```
-h, --help Print help
```

Contoh

Example

```
aws-cloudhsm > login --username test_user --role admin mfa-token-sign --token /home/  
valid.token
```

```
Enter password:
```

Enter signed token file path (press enter if same as the unsigned token file):

```
{
  "error_code": 0,
  "data": {
    "username": "test_user",
    "role": "admin"
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Diperlukan: Jika beberapa cluster telah [dikonfigurasi](#).

<TOKEN>

Filepath tempat file token yang tidak ditandatangani akan ditulis.

Wajib: Ya

Topik terkait

- [Memulai dengan CloudHSM CLI](#)
- [Aktifkan Cluster](#)
- [Menggunakan CloudHSM CLI untuk mengelola MFA](#)

Keluar dari HSM menggunakan CloudHSM CLI

Gunakan logout perintah di CloudHSM CLI untuk keluar dari setiap modul keamanan perangkat keras (HSM) dalam sebuah cluster. AWS CloudHSM

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Admin
- Pengguna krypto (CU)

Sintaksis

```
aws-cloudhsm > help logout
```

Logout of your cluster

USAGE:

logout

OPTIONS:

`--cluster-id <CLUSTER_ID>` Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

`-h, --help` Print help information

`-V, --version` Print version information

Contoh

Example

Perintah ini mencatat Anda keluar dari semua HSMs dalam sebuah cluster.

```
aws-cloudhsm > logout
```

```
{  
  "error_code": 0,  
  "data": "Logout successful"  
}
```

Topik terkait

- [Memulai dengan CloudHSM CLI](#)
- [Aktifkan Cluster](#)

Kategori pengguna di CloudHSM CLI

Di CloudHSM user CLI, adalah kategori induk untuk sekelompok perintah yang, bila dikombinasikan dengan kategori induk, membuat perintah khusus untuk pengguna. Saat ini, kategori pengguna terdiri dari perintah berikut:

- [perubahan pengguna-mfa](#)
- [perubahan kata sandi pengguna](#)

- [pengguna membuat](#)
- [menghapus pengguna](#)
- [daftar pengguna](#)
- [replikasi pengguna](#)

Kategori perubahan pengguna-mfa di CloudHSM CLI

Dalam CloudHSM user change-mfa CLI, adalah kategori induk untuk sekelompok perintah yang, ketika dikombinasikan dengan kategori induk, membuat perintah khusus untuk mengubah otentikasi multi-faktor (MFA) untuk pengguna.

Saat ini, kategori ini terdiri dari sub-perintah berikut:

- [tanda token](#)

Ubah pengaturan MFA pengguna dengan CloudHSM CLI

Gunakan user change-mfa token-sign perintah di CloudHSM CLI untuk memperbarui pengaturan otentikasi multi-faktor (MFA) akun pengguna. Setiap akun pengguna dapat menjalankan perintah ini. Akun dengan peran Admin dapat menjalankan perintah ini untuk pengguna lain.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Admin
- Pengguna Crypto

Sintaksis

Saat ini, hanya ada satu strategi multi-faktor yang tersedia untuk pengguna: Token Sign.

```
aws-cloudhsm > help user change-mfa
```

```
Change a user's Mfa Strategy
```

```
Usage:
```

```
    user change-mfa <COMMAND>
```

```
Commands:
```

```
token-sign  Register or Deregister a public key using token-sign mfa strategy
help        Print this message or the help of the given subcommand(s)
```

Strategi Token Sign meminta file Token untuk menulis token yang tidak ditandatangani.

```
aws-cloudhsm > help user change-mfa token-sign
```

```
Register or Deregister a public key using token-sign mfa strategy
```

```
Usage: user change-mfa token-sign [OPTIONS] --username <USERNAME> --role <ROLE> <--
token <TOKEN>|--deregister>
```

Options:

```
--cluster-id <CLUSTER_ID>
```

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

```
--username <USERNAME>
```

Username of the user that will be modified

```
--role <ROLE>
```

Role the user has in the cluster

Possible values:

- crypto-user: A CryptoUser has the ability to manage and use keys
- admin: An Admin has the ability to manage user accounts

```
--change-password <CHANGE_PASSWORD>
```

Optional: Plaintext user's password. If you do not include this argument you will be prompted for it

```
--token <TOKEN>
```

Filepath where the unsigned token file will be written. Required for enabling MFA for a user

```
--approval <APPROVAL>
```

Filepath of signed quorum token file to approve operation

```
--deregister
```

Deregister the MFA public key, if present

```
--change-quorum
```

Change the Quorum public key along with the MFA key

```
-h, --help
    Print help (see a summary with '-h')
```

Contoh

Perintah ini akan menulis satu token yang tidak ditandatangani per HSM di cluster Anda ke file yang ditentukan oleh. token Saat Anda diminta, tandatangani token dalam file.

Example: Tulis satu token yang tidak ditandatangani per HSM di cluster Anda

```
aws-cloudhsm > user change-mfa token-sign --username cu1 --change-password password --
role crypto-user --token /path/myfile
Enter signed token file path (press enter if same as the unsigned token file):
Enter public key PEM file path:/path/mypemfile
{
  "error_code": 0,
  "data": {
    "username": "test_user",
    "role": "admin"
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<ROLE>

Menentukan peran yang diberikan ke akun pengguna. Parameter ini diperlukan. Untuk informasi rinci tentang jenis pengguna di HSM, lihat [Memahami pengguna HSM](#).

Nilai yang valid

- Admin: Admin dapat mengelola pengguna, tetapi mereka tidak dapat mengelola kunci.
- Pengguna Crypto: Pengguna Crypto dapat membuat kunci kelola dan menggunakan kunci dalam operasi kriptografi.

<USERNAME>

Menentukan nama yang mudah diingat untuk pengguna. Panjang maksimum adalah 31 karakter. Satu-satunya karakter khusus yang diizinkan adalah garis bawah (_).

Anda tidak dapat mengubah nama pengguna setelah dibuat. Dalam perintah CloudHSM CLI, peran dan kata sandi peka huruf besar/kecil, tetapi nama pengguna tidak.

Wajib: Ya

<CHANGE_PASSWORD>

Menentukan kata sandi baru plaintext dari pengguna yang MFA sedang didaftarkan/dideregistrasi.

Wajib: Ya

<TOKEN>

Jalur file tempat file token yang tidak ditandatangani akan ditulis.

Wajib: Ya

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan pengguna kuorum lebih besar dari 1.

<DEREGISTER>

Deregister kunci publik MFA, jika ada.

<CHANGE-QUORUM>

Mengubah kunci publik kuorum bersama dengan kunci MFA.

Topik terkait

- [Memahami 2FA untuk pengguna HSM](#)

Ubah kata sandi pengguna dengan CloudHSM CLI

Gunakan user change-password perintah di CloudHSM CLI untuk mengubah kata sandi pengguna yang ada di cluster Anda. AWS CloudHSM Untuk mengaktifkan MFA bagi pengguna, gunakan perintah. `user change-mfa`

Setiap pengguna dapat mengubah kata sandi mereka sendiri. Selain itu, pengguna dengan peran admin dapat mengubah kata sandi pengguna lain di cluster. Anda tidak perlu memasukkan kata sandi saat ini untuk melakukan perubahan.

 Note

Anda tidak dapat mengubah kata sandi pengguna yang saat ini masuk ke cluster.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Admin
- Pengguna kriptografi (CU)

Sintaksis

 Note

Untuk mengaktifkan otentikasi multi-faktor (MFA) bagi pengguna, gunakan perintah. `user change-mfa`

```
aws-cloudhsm > help user change-password
```

```
Change a user's password
```

```
Usage:
```

```
cloudhsm-cli user change-password [OPTIONS] --username <USERNAME> --role <ROLE>
[--password <PASSWORD>]
```

```
Options:
```

```
--cluster-id <CLUSTER_ID>
```

```
Unique Id to choose which of the clusters in the config file to run the
operation against. If not provided, will fall back to the value provided when
interactive mode was started, or error
```

```
--username <USERNAME>
```

```
Username of the user that will be modified
```

```

--role <ROLE>
    Role the user has in the cluster

    Possible values:
    - crypto-user: A CryptoUser has the ability to manage and use keys
    - admin:       An Admin has the ability to manage user accounts

--password <PASSWORD>
    Optional: Plaintext user's password. If you do not include this argument you
    will be prompted for it

--approval <APPROVAL>
    Filepath of signed quorum token file to approve operation

--deregister-mfa <DEREGISTER-MFA>
    Deregister the user's mfa public key, if present

--deregister-quorum <DEREGISTER-QUORUM>
    Deregister the user's quorum public key, if present
-h, --help
    Print help (see a summary with '-h')

```

Contoh

Contoh berikut menunjukkan cara menggunakan `user change-password` untuk mengatur ulang kata sandi untuk pengguna saat ini atau pengguna lain di cluster Anda.

Example: Ubah kata sandi Anda

Setiap pengguna di cluster dapat menggunakan `user change-password` untuk mengubah kata sandi mereka sendiri.

Output berikut menunjukkan bahwa Bob saat ini login sebagai pengguna krypto (CU).

```

aws-cloudhsm > user change-password --username bob --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "bob",
    "role": "crypto-user"
  }
}

```

```
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum kuorum layanan pengguna kuorum lebih besar dari 1.

<DEREGISTER-MFA>

Deregister kunci publik MFA, jika ada.

<DEREGISTER-QUORUM>

Batalkan pendaftaran kunci publik Kuorum, jika ada.

<PASSWORD>

Menentukan plaintext password baru dari pengguna. Karakter berikut tidak diizinkan ':'

Wajib: Ya

<ROLE>

Menentukan peran yang diberikan ke akun pengguna. Parameter ini diperlukan. Untuk informasi rinci tentang jenis pengguna di HSM, lihat [Memahami pengguna HSM](#).

Nilai yang valid

- Admin: Admin dapat mengelola pengguna, tetapi mereka tidak dapat mengelola kunci.
- Pengguna Crypto: Pengguna Crypto dapat membuat kunci kelola dan menggunakan kunci dalam operasi kriptografi.

<USERNAME>

Menentukan nama yang mudah diingat untuk pengguna. Panjang maksimum adalah 31 karakter. Satu-satunya karakter khusus yang diizinkan adalah garis bawah (_).

Anda tidak dapat mengubah nama pengguna setelah dibuat. Dalam perintah CloudHSM CLI, peran dan kata sandi peka huruf besar/kecil, tetapi nama pengguna tidak.

Wajib: Ya

Topik terkait

- [daftar pengguna](#)
- [pengguna membuat](#)
- [menghapus pengguna](#)

Kategori kuorum perubahan pengguna di CloudHSM CLI

Di CloudHSM user change-quorum CLI, adalah kategori induk untuk sekelompok perintah yang, bila digabungkan dengan kategori induk, membuat perintah khusus untuk mengubah kuorum bagi pengguna.

user change-quorum digunakan untuk mendaftarkan otentikasi kuorum pengguna menggunakan strategi kuorum tertentu. Pada SDK 5.8.0, hanya ada satu strategi kuorum yang tersedia untuk pengguna seperti yang ditunjukkan di bawah ini.

Saat ini, kategori ini terdiri dari kategori dan sub-perintah berikut:

- [tanda token](#)
 - [mendaftar](#)

Kategori tanda token ubah-kuorum pengguna di CloudHSM CLI

Di CloudHSM user change-quorum token-sign CLI, adalah kategori induk untuk perintah yang, bila digabungkan dengan kategori induk ini, membuat perintah khusus untuk operasi kuorum tanda token.

Saat ini, kategori ini terdiri dari perintah berikut:

- [mendaftar](#)

Daftarkan strategi kuorum tanda token pengguna menggunakan CloudHSM CLI

Gunakan user change-quorum token-sign register perintah di CloudHSM CLI untuk mendaftarkan strategi kuorum tanda token untuk pengguna admin.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Admin

Sintaksis

```
aws-cloudhsm > help user change-quorum token-sign register
```

Register a user for quorum authentication with a public key

Usage: user change-quorum token-sign register --public-key **<PUBLIC_KEY>** --signed-token **<SIGNED_TOKEN>**

Options:

--cluster-id **<CLUSTER_ID>** Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

--public-key **<PUBLIC_KEY>** Filepath to public key PEM file

--signed-token **<SIGNED_TOKEN>** Filepath with token signed by user private key

-h, --help Print help (see a summary with '-h')

Contoh

Example

Untuk menjalankan perintah ini, Anda harus masuk sebagai pengguna yang Anda register quorum token-sign inginkan.

```
aws-cloudhsm > login --username admin1 --role admin
```

Enter password:

```
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

user change-quorum token-sign registerPerintah akan mendaftarkan kunci publik Anda dengan HSM. Akibatnya, Anda akan memenuhi syarat sebagai pemberi persetujuan kuorum untuk operasi yang

diperlukan kuorum yang membutuhkan pengguna untuk mendapatkan tanda tangan kuorum untuk memenuhi ambang nilai kuorum yang diperlukan.

```
aws-cloudhsm > user change-quorum token-sign register \  
  --public-key /home/mypemfile \  
  --signed-token /home/mysignedtoken  
{  
  "error_code": 0,  
  "data": {  
    "username": "admin1",  
    "role": "admin"  
  }  
}
```

Anda sekarang dapat menjalankan user list perintah dan mengonfirmasi bahwa tanda token kuorum telah terdaftar untuk pengguna ini.

```
aws-cloudhsm > user list  
{  
  "error_code": 0,  
  "data": {  
    "users": [  
      {  
        "username": "admin",  
        "role": "admin",  
        "locked": "false",  
        "mfa": [],  
        "quorum": [],  
        "cluster-coverage": "full"  
      },  
      {  
        "username": "admin1",  
        "role": "admin",  
        "locked": "false",  
        "mfa": [],  
        "quorum": [  
          {  
            "strategy": "token-sign",  
            "status": "enabled"  
          }  
        ],  
        "cluster-coverage": "full"  
      }  
    ]  
  }  
}
```

```
]
}
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<PUBLIC-KEY>

Filepath ke file PEM kunci publik.

Wajib: Ya

<SIGNED-TOKEN>

Filepath dengan token yang ditandatangani oleh kunci pribadi pengguna.

Wajib: Ya

Topik terkait

- [Menggunakan CloudHSM CLI untuk mengelola otentikasi kuorum](#)
- [Menggunakan otentikasi kuorum untuk admin: penyiapan pertama kali](#)
- [Mengubah nilai minimum kuorum untuk admin](#)
- [Nama dan jenis layanan yang mendukung otentikasi kuorum](#)

Buat AWS CloudHSM pengguna dengan CloudHSM CLI

user createPerintah di CloudHSM CLI membuat pengguna di cluster Anda. AWS CloudHSM Hanya akun pengguna dengan peran admin yang dapat menjalankan perintah ini.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Admin

Persyaratan

Untuk menjalankan perintah ini, Anda harus masuk sebagai pengguna admin

Sintaksis

```
aws-cloudhsm > help user create
```

Create a new user

Usage: cloudhsm-cli user create [OPTIONS] --username *<USERNAME>* --role *<ROLE>* [--password *<PASSWORD>*]

Options:

--cluster-id *<CLUSTER_ID>*

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

--username *<USERNAME>*

Username to access the HSM cluster

--role *<ROLE>*

Role the user has in the cluster

Possible values:

- crypto-user: A CryptoUser has the ability to manage and use keys
- admin: An Admin has the ability to manage user accounts

--password *<PASSWORD>*

Optional: Plaintext user's password. If you do not include this argument you will be prompted for it

--approval *<APPROVAL>*

Filepath of signed quorum token file to approve operation

-h, --help

Print help (see a summary with '-h')

Contoh

Contoh-contoh ini menunjukkan cara menggunakan user create untuk membuat pengguna baru di Anda HSMs.

Example: Buat pengguna krypto

Contoh ini membuat akun di AWS CloudHSM klaster Anda dengan peran pengguna krypto.

```
aws-cloudhsm > user create --username alice --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Diperlukan: Jika beberapa cluster telah [dikonfigurasi](#).

<USERNAME>

Menentukan nama yang mudah diingat untuk pengguna. Panjang maksimum adalah 31 karakter. Satu-satunya karakter khusus yang diizinkan adalah garis bawah (_). Nama pengguna tidak peka huruf besar/kecil dalam perintah ini, nama pengguna selalu ditampilkan dalam huruf kecil.

Wajib: Ya

<ROLE>

Menentukan peran yang ditetapkan untuk pengguna ini. Parameter ini diperlukan. Nilai yang valid adalah admin, crypto-user.

Untuk mendapatkan peran pengguna, gunakan user list perintah. Untuk informasi rinci tentang jenis pengguna di HSM, lihat [Memahami pengguna HSM](#).

<PASSWORD>

Menentukan password dari pengguna yang masuk ke HSMs Karakter berikut tidak diizinkan ':'

Wajib: Ya

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai kuorum layanan pengguna kuorum lebih besar dari 1.

Topik terkait

- [daftar pengguna](#)
- [menghapus pengguna](#)
- [perubahan kata sandi pengguna](#)

Hapus AWS CloudHSM pengguna dengan CloudHSM CLI

user deletePerintah di CloudHSM CLI menghapus pengguna dari cluster Anda. AWS CloudHSM Hanya akun pengguna dengan peran admin yang dapat menjalankan perintah ini. Anda tidak dapat menghapus pengguna yang saat ini masuk ke HSM.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Admin

Persyaratan

- Anda tidak dapat menghapus akun pengguna yang memiliki kunci.
- Akun pengguna Anda harus memiliki peran admin untuk menjalankan perintah ini.

Sintaksis

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

```
aws-cloudhsm > help user delete
```

```
Delete a user
```

```
Usage: user delete [OPTIONS] --username <USERNAME> --role <ROLE>
```

Options:**--cluster-id** *<CLUSTER_ID>*

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

--username *<USERNAME>*

Username to access the HSM cluster

--role *<ROLE>*

Role the user has in the cluster

Possible values:

- crypto-user: A CryptoUser has the ability to manage and use keys
- admin: An Admin has the ability to manage user accounts

--approval *<APPROVAL>*

Filepath of signed quorum token file to approve operation

Contoh

```
aws-cloudhsm > user delete --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

Pendapat***<CLUSTER_ID>***

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<USERNAME>

Menentukan nama yang mudah diingat untuk pengguna. Panjang maksimum adalah 31 karakter. Satu-satunya karakter khusus yang diizinkan adalah garis bawah (_). Nama pengguna tidak peka huruf besar/kecil dalam perintah ini, nama pengguna selalu ditampilkan dalam huruf kecil.

Wajib: Ya

<ROLE>

Menentukan peran yang ditetapkan untuk pengguna ini. Parameter ini diperlukan. Nilai yang valid adalah `admin`, `crypto-user`.

Untuk mendapatkan peran pengguna, gunakan `user list` perintah. Untuk informasi rinci tentang jenis pengguna di HSM, lihat [Memahami pengguna HSM](#).

Wajib: Ya

<APPROVAL>

Menentukan jalur file ke file token kuorum yang ditandatangani untuk menyetujui operasi. Hanya diperlukan jika nilai `kuorum` layanan pengguna kuorum lebih besar dari 1.

Wajib: Ya

Topik terkait

- [daftar pengguna](#)
- [pengguna membuat](#)
- [perubahan kata sandi pengguna](#)

Daftar semua AWS CloudHSM pengguna dengan CloudHSM CLI

`user list` Perintah di CloudHSM CLI mencantumkan akun pengguna yang ada di cluster Anda. AWS CloudHSM Anda tidak perlu login ke CloudHSM CLI untuk menjalankan perintah ini.

Note

Jika Anda menambah atau menghapus HSMs, perbarui file konfigurasi yang digunakan AWS CloudHSM klien dan alat baris perintah. Jika tidak, perubahan yang Anda buat mungkin tidak efektif pada semua HSMs di cluster.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Semua pengguna. Anda tidak perlu masuk untuk menjalankan perintah ini.

Sintaksis

```
aws-cloudhsm > help user list
List the users in your cluster

USAGE:
    user list

Options:
    --cluster-id <CLUSTER_ID> Unique Id to choose which of the clusters in the
    config file to run the operation against. If not provided, will fall back to the value
    provided when interactive mode was started, or error
    -h, --help                    Print help
```

Contoh

Perintah ini mencantumkan pengguna yang ada di klaster CloudHSM Anda.

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
      {
        "username": "test_user",
        "role": "admin",
        "locked": "false",
        "mfa": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ]
      }
    ]
  }
}
```

```
    "cluster-coverage": "full"
  },
  {
    "username": "app_user",
    "role": "internal(APPLIANCE_USER)",
    "locked": "false",
    "mfa": [],
    "cluster-coverage": "full"
  }
]
```

Output mencakup atribut pengguna berikut:

- Nama pengguna: Menampilkan nama ramah yang ditentukan pengguna untuk pengguna. Nama pengguna selalu ditampilkan dalam huruf kecil.
- Peran: Menentukan operasi yang dapat dilakukan pengguna pada HSM.
- Terkunci: Menunjukkan apakah akun pengguna ini telah dikunci.
- MFA: Menunjukkan mekanisme otentikasi multi-faktor yang didukung untuk akun pengguna ini.
- Cakupan cluster: Menunjukkan ketersediaan seluruh cluster dari akun pengguna ini.

Topik terkait

- [listUsers](#) di key_mgmt_util
- [pengguna membuat](#)
- [menghapus pengguna](#)
- [perubahan kata sandi pengguna](#)

Replikasi pengguna dengan CloudHSM CLI

Gunakan user replicate perintah di CloudHSM CLI untuk mereplikasi pengguna dari AWS CloudHSM cluster sumber ke cluster tujuan. AWS CloudHSM

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Admin () COs

Persyaratan

- Cluster sumber dan tujuan harus klon. Ini berarti satu dibuat dari cadangan yang lain, atau keduanya dibuat dari cadangan umum. Untuk informasi selengkapnya, lihat [Membuat cluster dari backup](#).
- Untuk menjalankan perintah ini, Anda harus masuk sebagai admin di cluster sumber dan tujuan.
 - Dalam mode perintah tunggal, perintah akan menggunakan variabel lingkungan CLOUDHSM_PIN dan CLOUDHSM_ROLE untuk mengautentikasi pada cluster sumber. Untuk informasi selengkapnya, lihat [Mode Perintah Tunggal](#). Untuk memberikan kredensi untuk klaster tujuan, Anda perlu menyetel dua variabel lingkungan tambahan: DESTINATION_CLOUDHSM_PIN dan DESTINATION_CLOUDHSM_ROLE:

```
$ export DESTINATION_CLOUDHSM_ROLE=<role>
```

```
$ export DESTINATION_CLOUDHSM_PIN=<username:password>
```

- Dalam mode interaktif, pengguna harus secara eksplisit masuk ke cluster sumber dan tujuan.

Sintaksis

```
aws-cloudhsm > help user replicate
```

```
Replicate a user from a source to a destination cluster
```

```
Usage: user replicate --username <USERNAME> --role <ROLE> --source-cluster-id <SOURCE_CLUSTER_ID> --destination-cluster-id <DESTINATION_CLUSTER_ID>
```

```
Options:
```

```
--username <USERNAME>
```

```
Username of the user to replicate
```

```
--role <ROLE>
```

```
Role the user has in the cluster
```

```
Possible values:
```

- crypto-user: A CryptoUser has the ability to manage and use keys
- admin: An Admin has the ability to manage user accounts

```

--source-cluster-id <SOURCE_CLUSTER_ID>
    Source cluster ID

--destination-cluster-id <DESTINATION_CLUSTER_ID>
    Destination cluster ID

-h, --help
    Print help (see a summary with '-h')
```

Contoh

Example Contoh: Replikasi pengguna

Perintah ini mereplikasi pengguna dari cluster sumber dengan ke klaster tujuan kloning. Contoh di bawah ini menunjukkan output saat login sebagai admin di kedua cluster.

```

admin-user@cluster-1234abcdefg > user replicate \
    --username example-admin \
    --role admin \
    --source-cluster-id cluster-1234abcdefg \
    --destination-cluster-id cluster-2345bcdefgh
{
  "error_code": 0,
  "data": {
    "user": {
      "username": "example-admin",
      "role": "admin",
      "locked": "false",
      "mfa": [],
      "quorum": [],
      "cluster-coverage": "full"
    },
    "message": "Successfully replicated user"
  }
}
```

Pendapat

<USERNAME>

Menentukan nama pengguna pengguna untuk mereplikasi di cluster sumber.

Wajib: Ya

<ROLE>

Menentukan peran yang ditetapkan untuk pengguna ini. Parameter ini diperlukan. Nilai yang valid adalah `admin`, `crypto-user`.

Untuk mendapatkan peran pengguna, gunakan `user list` perintah. Untuk informasi rinci tentang jenis pengguna di HSM, lihat [Memahami pengguna HSM](#).

Wajib: Ya

<SOURCE_CLUSTER_ID>

ID cluster sumber.

Wajib: Ya

<DESTINATION_CLUSTER_ID>

ID cluster tujuan.

Wajib: Ya

Topik terkait

- [Menghubungkan ke beberapa cluster dengan CloudHSM CLI](#)

Kategori kuorum di CloudHSM CLI

Dalam CloudHSM quorum CLI, adalah kategori induk untuk sekelompok perintah yang, bila quorum digabungkan dengan, membuat perintah khusus untuk otentikasi kuorum, atau M dari operasi N. Saat ini, kategori ini terdiri dari `token-sign` sub-kategori yang terdiri dari perintahnya sendiri. Klik tautan di bawah ini untuk detailnya.

- [tanda token](#)

Layanan Admin: Otentikasi kuorum digunakan untuk layanan istimewa admin seperti membuat pengguna, menghapus pengguna, mengubah kata sandi pengguna, mengatur nilai kuorum, dan menonaktifkan kemampuan kuorum dan MFA.

Layanan Pengguna Crypto: Otentikasi kuorum digunakan untuk layanan istimewa pengguna kriptografi yang terkait dengan kunci tertentu seperti menandatangani dengan kunci, kunci, `sharing/unsharing` kunci, dan `wrapping/unwrapping` pengaturan atribut kunci. Nilai kuorum kunci terkait dikonfigurasi

saat kunci dihasilkan, diimpor, atau dibuka. Nilai kuorum harus sama dengan atau kurang dari jumlah pengguna yang dikaitkan dengan kunci, yang mencakup pengguna yang kunci dibagikan dan pemilik kunci.

Setiap jenis layanan selanjutnya dipecah menjadi nama layanan yang memenuhi syarat, yang berisi serangkaian operasi layanan yang didukung kuorum tertentu yang dapat dilakukan.

Nama layanan	Jenis Layanan	Operasi layanan
user	Admin	• pengguna membuat • menghapus pengguna • perubahan kata sandi pengguna • perubahan pengguna-mfa
kuorum	Admin	• tanda token kuorum set-quorum-value
gugus ¹	Admin	• cluster mtls register-trust-anchor • cluster mtls deregister-trust-anchor • cluster mtls set-penegakan
manajemen kunci	Pengguna Crypto	• bungkus kunci • buka kunci • berbagi kunci • kunci unshare • kunci set-attribut
penggunaan kunci	Pengguna Crypto	• tanda kunci

[1] Layanan cluster tersedia secara eksklusif di hsm2m.medium

Topik terkait

- [Mengatur otentikasi kuorum untuk AWS CloudHSM admin menggunakan CloudHSM CLI](#)

- [Kelola otentikasi kuorum \(M of N access control\) menggunakan CloudHSM CLI](#)

Kategori tanda token kuorum di CloudHSM CLI

Dalam CloudHSM quorum token-sign CLI, adalah kategori untuk sekelompok perintah yang, ketika quorum token-sign dikombinasikan dengan, membuat perintah khusus untuk otentikasi kuorum, atau M dari operasi N.

Saat ini, kategori ini terdiri dari perintah berikut:

- [hapus](#)
- [menghasilkan](#)
- [daftar](#)
- [list-quorum-values](#)
- [set-quorum-value](#)

Hapus token kuorum menggunakan CloudHSM CLI

Gunakan quorum token-sign delete perintah di CloudHSM CLI untuk menghapus satu atau beberapa token untuk layanan resmi kuorum.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Admin

Sintaksis

```
aws-cloudhsm > help quorum token-sign delete
```

```
Delete one or more Quorum Tokens
```

```
Usage: quorum token-sign delete --scope <SCOPE>
```

```
Options:
```

```
--cluster-id <CLUSTER_ID>
```

```
    Unique Id to choose which of the clusters in the config file to run the  
    operation against. If not provided, will fall back to the value provided when  
    interactive mode was started, or error
```



```
--scope <SCOPE>
    Scope of which token(s) will be deleted

Possible values:
- user: Deletes all token(s) of currently logged in user
- all:  Deletes all token(s) on the HSM
-h, --help
    Print help (see a summary with '-h')
```

Contoh

Contoh berikut menunjukkan bagaimana quorum token-sign delete perintah di CloudHSM CLI dapat digunakan untuk menghapus satu atau beberapa token untuk layanan resmi kuorum.

Example: Hapus satu atau beberapa token untuk layanan resmi kuorum

```
aws-cloudhsm > quorum token-sign delete --scope all
{
  "error_code": 0,
  "data": "Deletion of quorum token(s) successful"
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<SCOPE>

Ruang lingkup di mana token (s) akan dihapus di AWS CloudHSM cluster.

Nilai yang valid

- Pengguna: Digunakan untuk menghapus hanya token yang dimiliki oleh pengguna yang masuk.
- Semua: Digunakan untuk menghapus semua token di AWS CloudHSM cluster.

Topik terkait

- [daftar pengguna](#)
- [pengguna membuat](#)

- [menghapus pengguna](#)

Buat token kuorum menggunakan CloudHSM CLI

Gunakan quorum token-sign generate perintah di CloudHSM CLI untuk menghasilkan token untuk layanan resmi kuorum.

Ada batasan untuk mendapatkan satu token aktif per pengguna per layanan pada cluster HSM untuk pengguna layanan dan kuorum. Batas ini tidak berlaku untuk token yang terkait dengan layanan utama.

 Note

Hanya Admin dan Pengguna Crypto yang dapat menghasilkan token layanan tertentu. Untuk informasi selengkapnya tentang jenis dan nama layanan, lihat [Nama dan jenis layanan yang mendukung otentikasi kuorum](#)

Layanan Admin: Otentikasi kuorum digunakan untuk layanan istimewa admin seperti membuat pengguna, menghapus pengguna, mengubah kata sandi pengguna, mengatur nilai kuorum, dan menonaktifkan kemampuan kuorum dan MFA.

Layanan Pengguna Crypto: Otentikasi kuorum digunakan untuk layanan istimewa pengguna kriptografi yang terkait dengan kunci tertentu seperti menandatangani dengan kunci, kunci, sharing/unsharing kunci, dan wrapping/unwrapping pengaturan atribut kunci. Nilai kuorum kunci terkait dikonfigurasi saat kunci dihasilkan, diimpor, atau dibuka. Nilai kuorum harus sama dengan atau kurang dari jumlah pengguna yang dikaitkan dengan kunci, yang mencakup pengguna yang kunci dibagikan dan pemilik kunci.

Setiap jenis layanan selanjutnya dipecah menjadi nama layanan yang memenuhi syarat, yang berisi serangkaian operasi layanan yang didukung kuorum tertentu yang dapat dilakukan.

Nama layanan	Jenis Layanan	Operasi layanan
user	Admin	• pengguna membuat • menghapus pengguna • perubahan kata sandi pengguna

Nama layanan	Jenis Layanan	Operasi layanan
		• perubahan pengguna-mfa
kuorum	Admin	• tanda token kuorum set-quorum-value
kluster ¹	Admin	• kluster mtls register-trust-anchor • kluster mtls deregister-trust-anchor • cluster mtls set-penegakan
manajemen kunci	Pengguna Crypto	• bungkus kunci • buka kunci • pembagian kunci • kunci unshare • kunci set-attribut
penggunaan kunci	Pengguna Crypto	• tanda kunci

[1] Layanan cluster tersedia secara eksklusif di hsm2m.medium

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Admin
- Pengguna krypto (CU)

Sintaksis

```
aws-cloudhsm > help quorum token-sign generate
```

Generate a token

Usage: quorum token-sign generate --service *<SERVICE>* --token *<TOKEN>*

Options:

`--cluster-id <CLUSTER_ID>`

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

`--service <SERVICE>`

Service the token will be used for

Possible values:

- user:
User management service is used for executing quorum authenticated user management operations
- quorum:
Quorum management service is used for setting quorum values for any quorum service
- cluster:
Cluster management service is used for executing quorum for cluster wide configuration managements like mtls enforcement, mtls registration and mtls deregistration
- registration:
Registration service is used for registering a public key for quorum authentication
- key-usage:
Key usage service is used for executing quorum authenticated key usage operations
- key-management:
Key management service is used for executing quorum authenticated key management operations

`--token <TOKEN>`

Filepath where the unsigned token file will be written

`-h, --help` Print help

Contoh

Perintah ini akan menulis satu token yang tidak ditandatangani per HSM di cluster Anda ke file yang ditentukan oleh. token

Example: Tulis satu token yang tidak ditandatangani per HSM di cluster Anda

```
aws-cloudhsm > quorum token-sign generate --service user --token /home/tfile
{
  "error_code": 0,
```

```
"data": {  
  "filepath": "/home/tfile"  
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<SERVICE>

Menentukan layanan resmi kuorum untuk menghasilkan token. Parameter ini diperlukan.

Nilai yang valid

- user: Layanan manajemen pengguna yang digunakan untuk menjalankan operasi manajemen pengguna resmi kuorum.
- quorum: Layanan manajemen kuorum yang digunakan untuk menetapkan nilai kuorum resmi kuorum untuk setiap layanan resmi kuorum.
- cluster: Layanan manajemen cluster yang digunakan untuk mengeksekusi kuorum untuk manajemen konfigurasi luas cluster seperti penegakan mtls, pendaftaran mtls dan deregistrasi mtls.
- registrasi: Menghasilkan token yang tidak ditandatangani untuk digunakan dalam mendaftarkan kunci publik untuk otorisasi kuorum.
- key-usage: Menghasilkan token yang tidak ditandatangani yang digunakan untuk menjalankan operasi penggunaan kunci resmi kuorum.
- key-management: Menghasilkan token yang tidak ditandatangani yang digunakan untuk menjalankan operasi manajemen kunci resmi kuorum.

Wajib: Ya

<TOKEN>

Filepath tempat file token yang tidak ditandatangani akan ditulis.

Wajib: Ya

Topik terkait

- [Nama dan jenis layanan yang mendukung otentikasi kuorum](#)

Daftar token kuorum menggunakan CloudHSM CLI

Gunakan quorum token-sign list perintah di CloudHSM CLI untuk mencantumkan semua token kuorum tanda token yang ada di klaster Anda. AWS CloudHSM Ini termasuk token yang dihasilkan oleh pengguna lain. Token terikat pada pengguna, jadi meskipun Anda mungkin melihat token dari pengguna lain, Anda hanya dapat menggunakan token yang terkait dengan pengguna yang saat ini masuk.

Untuk informasi selengkapnya tentang jenis dan nama layanan, lihat [Nama dan jenis layanan yang mendukung otentikasi kuorum](#). Untuk informasi selengkapnya tentang konten yang ditampilkan dari token yang terdaftar, lihat [the section called “Manajemen kunci dan penggunaan dengan kuorum \(M dari N\)”](#) token yang terkait dengan key-management dan key-usage layanan, dan lihat [the section called “Manajemen pengguna dengan kuorum \(M dari N\)”](#) token yang terkait dengan user, quorum, atau cluster layanan, masing-masing.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Admin
- Pengguna krypto (CU)

Sintaksis

```
aws-cloudhsm > help quorum token-sign list
```

```
List the token-sign tokens in your cluster
```

```
Usage: quorum token-sign list
```

```
Options:
```

```
--cluster-id <CLUSTER_ID> Unique Id to choose which of the clusters in the  
config file to run the operation against. If not provided, will fall back to the value  
provided when interactive mode was started, or error
```

```
-h, --help Print help
```

Contoh

Perintah ini akan mencantumkan semua token tanda token yang ada di cluster Anda. AWS CloudHSM Ini termasuk token yang dihasilkan oleh pengguna lain. Token terikat pada pengguna, jadi meskipun Anda mungkin melihat token dari pengguna lain, Anda hanya dapat menggunakan token yang terkait dengan pengguna yang saat ini masuk.

Example

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": [
      {
        "username": "admin",
        "service": "quorum",
        "approvals-required": 2,
        "number-of-approvals": 0,
        "token-timeout-seconds": 397,
        "cluster-coverage": "full"
      },
      {
        "username": "admin",
        "service": "user",
        "approvals-required": 2,
        "number-of-approvals": 0,
        "token-timeout-seconds": 588,
        "cluster-coverage": "full"
      },
      {
        "username": "crypto_user1",
        "service": "key-management",
        "key-reference": "0x000000000002c33f7",
        "minimum-token-count": 1
      },
      {
        "username": "crypto_user1",
        "service": "key-usage",
        "key-reference": "0x000000000002c33f7",
        "minimum-token-count": 1
      }
    ]
  }
}
```

```
}  
}
```

Topik terkait

- [tanda token kuorum menghasilkan](#)

Tampilkan nilai kuorum menggunakan CloudHSM CLI

Gunakan `quorum token-sign list-quorum-values` perintah di CloudHSM CLI untuk mencantumkan nilai kuorum yang ditetapkan dalam klaster Anda. AWS CloudHSM

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Semua pengguna. Anda tidak perlu masuk untuk menjalankan perintah ini.

Sintaksis

```
aws-cloudhsm > help quorum token-sign list-quorum-values
```

```
List current quorum values
```

```
Usage: quorum token-sign list-quorum-values
```

```
Options:
```

```
--cluster-id <CLUSTER_ID> Unique Id to choose which of the clusters in the  
config file to run the operation against. If not provided, will fall back to the value  
provided when interactive mode was started, or error
```

```
-h, --help Print help
```

Contoh

Perintah ini mencantumkan nilai kuorum yang ditetapkan dalam AWS CloudHSM klaster Anda untuk setiap layanan.

Example

hsm1.sedang:

```
aws-cloudhsm > quorum token-sign list-quorum-values
```

```
{
```



```
"error_code": 0,  
"data": {  
  "user": 1,  
  "quorum": 1  
}  
}
```

hsm2m.sedang:

```
aws-cloudhsm > quorum token-sign list-quorum-values  
{  
  "error_code": 0,  
  "data": {  
    "user": 1,  
    "quorum": 1,  
    "cluster": 1  
  }  
}
```

Topik terkait

- [Nama dan jenis layanan yang mendukung otentikasi kuorum](#)
- [Pengaturan mTL \(disarankan\)](#)

Perbarui nilai kuorum menggunakan CloudHSM CLI

Gunakan quorum token-sign set-quorum-value perintah di CloudHSM CLI untuk menetapkan nilai kuorum baru untuk layanan resmi kuorum.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Admin

Sintaksis

```
aws-cloudhsm > help quorum token-sign set-quorum-value  
Set a quorum value  
  
Usage: quorum token-sign set-quorum-value [OPTIONS] --service <SERVICE> --value <VALUE>
```

Options:**--cluster-id** *<CLUSTER_ID>*

Unique Id to choose which of the clusters in the config file to run the operation against. If not provided, will fall back to the value provided when interactive mode was started, or error

--service *<SERVICE>*

Service the token will be used for

Possible values:

- user:

User management service is used for executing quorum authenticated user management operations

- quorum:

Quorum management service is used for setting quorum values for any quorum service

- cluster:

Cluster management service is used for executing quorum for cluster wide configuration managements like mtls enforcement, mtls registration and mtls deregistration

--value *<VALUE>*

Value to set for service

--approval *<APPROVAL>*

Filepath of signed quorum token file to approve operation

-h, --help

Print help (see a summary with '-h')

Contoh**Example**

Dalam contoh berikut, perintah ini menulis satu token yang tidak ditandatangani per HSM di cluster Anda ke file yang ditentukan oleh token. Saat Anda diminta, tandatangani token dalam file.

```
aws-cloudhsm > quorum token-sign set-quorum-value --service quorum --value 2
{
  "error_code": 0,
  "data": "Set Quorum Value successful"
}
```

Anda kemudian dapat menjalankan `list-quorum-values` perintah untuk mengonfirmasi bahwa nilai kuorum untuk layanan manajemen kuorum telah ditetapkan:

hsm1.sedang:

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 1,
    "quorum": 2
  }
}
```

hsm2m.sedang:

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 1,
    "quorum": 2,
    "cluster": 1
  }
}
```

Pendapat

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah [dikonfigurasi](#).

<APPROVAL>

Filepath dari file token yang ditandatangani akan disetujui pada HSM.

<SERVICE>

Menentukan layanan resmi kuorum untuk menghasilkan token. Parameter ini diperlukan. Untuk informasi selengkapnya tentang jenis dan nama layanan, lihat [Nama dan jenis layanan yang mendukung otentikasi kuorum](#).

Nilai yang valid

- **pengguna:** Layanan manajemen pengguna. Layanan yang digunakan untuk menjalankan operasi manajemen pengguna resmi kuorum.
- **kuorum:** Layanan manajemen kuorum. Layanan yang digunakan untuk menetapkan nilai kuorum resmi kuorum untuk setiap layanan resmi kuorum.
- **cluster:** Layanan manajemen cluster yang digunakan untuk mengeksekusi kuorum untuk manajemen konfigurasi luas cluster seperti penegakan mtls, pendaftaran mtls dan deregistrasi mtls.
- **registrasi:** Menghasilkan token yang tidak ditandatangani untuk digunakan dalam mendaftarkan kunci publik untuk otorisasi kuorum.

Wajib: Ya

<VALUE>

Menentukan nilai kuorum yang akan ditetapkan. Nilai kuorum maksimum adalah delapan (8).

Membutuhkan: Ya

Topik terkait

- [tanda token kuorum list-quorum-values](#)
- [Nama dan jenis layanan yang mendukung otentikasi kuorum](#)
- [Pengaturan mTL \(disarankan\)](#)

AWS CloudHSM Utilitas Manajemen (CMU)

Alat baris perintah `cloudhsm_mgmt_util` membantu petugas krypto mengelola pengguna dalam modul keamanan perangkat keras () dalam cluster. HSMs AWS CloudHSM AWS CloudHSM Management Utility (CMU) mencakup alat yang membuat, menghapus, dan mencantumkan pengguna, dan mengubah kata sandi pengguna.

CMU dan Key Management Utility (KMU) adalah bagian dari [Client SDK 3](#) suite. Klien SDK 3 dan alat baris perintah terkait (Key Management Utility dan CloudHSM Management Utility) hanya tersedia dalam tipe HSM `hsm1.medium`.

cloudhsm_mgmt_util juga menyertakan perintah yang memungkinkan pengguna krypto (CUs) untuk berbagi kunci dan mendapatkan dan mengatur atribut kunci. Perintah ini melengkapi perintah manajemen kunci dalam alat manajemen kunci primer, [key_mgmt_util](#).

Untuk memulai dengan cepat, lihat [Memulai dengan AWS CloudHSM Management Utility \(CMU\)](#). Untuk informasi rinci tentang perintah cloudhsm_mgmt_util dan contoh penggunaan perintah, lihat [Referensi untuk perintah Utilitas AWS CloudHSM Manajemen](#).

Topik

- [Platform yang didukung untuk Utilitas AWS CloudHSM Manajemen](#)
- [Memulai dengan AWS CloudHSM Management Utility \(CMU\)](#)
- [Instal dan konfigurasikan AWS CloudHSM klien untuk CMU \(Linux\)](#)
- [Instal dan konfigurasikan AWS CloudHSM klien untuk CMU \(Windows\)](#)
- [Referensi untuk perintah Utilitas AWS CloudHSM Manajemen](#)

Platform yang didukung untuk Utilitas AWS CloudHSM Manajemen

Topik ini menjelaskan platform Linux dan Windows yang didukung oleh AWS CloudHSM Management Utility (CMU).

Dukungan Linux

- Amazon Linux
- Amazon Linux 2
- CentOS 6.10+
- CentOS 7.3+
- CentOS 8
- Red Hat Enterprise Linux (RHEL) 6.10+
- Red Hat Enterprise Linux (RHEL) 7.9+
- Red Hat Enterprise Linux (RHEL) 8
- Ubuntu 16.04 LTS
- Ubuntu 18.04 LTS

Dukungan Windows

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

Memulai dengan AWS CloudHSM Management Utility (CMU)

AWS CloudHSM Management Utility (CMU) memungkinkan Anda mengelola pengguna modul keamanan perangkat keras (HSM). Gunakan topik ini untuk memulai tugas manajemen pengguna HSM dasar, seperti membuat pengguna, mendaftar pengguna, dan menghubungkan CMU ke klaster.

1. Untuk menggunakan CMU, Anda harus terlebih dahulu menggunakan alat konfigurasi untuk memperbarui konfigurasi CMU lokal dengan `--cmu` parameter dan alamat IP dari salah satu HSMs di cluster Anda. Lakukan ini setiap kali Anda menggunakan CMU untuk memastikan Anda mengelola pengguna HSM di setiap HSM di klaster.

Linux

```
$ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" --cmu <IP address>
```

2. Gunakan perintah berikut untuk memulai CLI dalam mode interaktif.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\cloudhsm_mgmt_util.exe" C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_util.cfg
```

Output harus serupa dengan yang berikut ini tergantung pada berapa banyak yang HSMs Anda miliki.

```
Connecting to the server(s), it may take time
depending on the server(s) load, please wait...

Connecting to server '10.0.2.9': hostname '10.0.2.9', port 2225...
Connected to server '10.0.2.9': hostname '10.0.2.9', port 2225.

Connecting to server '10.0.3.11': hostname '10.0.3.11', port 2225...
Connected to server '10.0.3.11': hostname '10.0.3.11', port 2225.

Connecting to server '10.0.1.12': hostname '10.0.1.12', port 2225...
Connected to server '10.0.1.12': hostname '10.0.1.12', port 2225.
```

Prompt berubah jadi `aws-ccloudhsm>` saat `cloudhsm_mgmt_util` berjalan.

- Gunakan perintah `loginHSM` untuk masuk ke klaster. Setiap jenis pengguna dapat menggunakan perintah ini untuk login ke klaster.

Perintah di contoh berikut mencatat di admin, yang merupakan [petugas krypto \(CO\)](#) default. Anda menyetel kata sandi pengguna ini saat Anda mengaktifkan klaster. Anda dapat menggunakan parameter `-hpswd` untuk menyembunyikan kata sandi Anda.

```
aws-ccloudhsm>loginHSM CO admin -hpswd
```

Sistem meminta kata sandi Anda. Anda memasukkan kata sandi, sistem menyembunyikan kata sandi, dan output menunjukkan bahwa perintah berhasil dan bahwa Anda telah terhubung ke semua HSMs di cluster.

```
Enter password:

loginHSM success on server 0(10.0.2.9)
loginHSM success on server 1(10.0.3.11)
loginHSM success on server 2(10.0.1.12)
```

- Gunakan `listUsers` untuk mencantumkan semua pengguna di klaster.

```
aws-cloudhsm>listUsers
```

CMU mendaftarkan semua pengguna di kluster.

Users on server 0(10.0.2.9):

Number of users found:2

User Id	User Type	User Name	
MofnPubKey	LoginFailureCnt	2FA	
1	CO	admin	NO
	0		NO
2	AU	app_user	NO
	0		NO

Users on server 1(10.0.3.11):

Number of users found:2

User Id	User Type	User Name	
MofnPubKey	LoginFailureCnt	2FA	
1	CO	admin	NO
	0		NO
2	AU	app_user	NO
	0		NO

Users on server 2(10.0.1.12):

Number of users found:2

User Id	User Type	User Name	
MofnPubKey	LoginFailureCnt	2FA	
1	CO	admin	NO
	0		NO
2	AU	app_user	NO
	0		NO

- Gunakan `createUser` untuk membuat pengguna CU bernama **example_user** dengan kata sandi **password1**.

Anda menggunakan pengguna CU dalam aplikasi Anda untuk melakukan operasi manajemen kriptografi dan kunci. Anda dapat membuat pengguna CU karena pada langkah 3 Anda login sebagai pengguna CO. Hanya pengguna CO yang dapat melakukan tugas manajemen pengguna dengan CMU, seperti membuat dan menghapus pengguna dan mengubah kata sandi pengguna lain.


```
aws-cloudhsm>createUser CU example_user password1
```

CMU meminta Anda tentang operasi membuat pengguna.

```
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****

Do you want to continue(y/n)?
```

6. Untuk membuat pengguna CU **example_user**, ketik **y**.
7. Gunakan `listUsers` untuk mencantumkan semua pengguna di klaster.

```
aws-cloudhsm>listUsers
```

CMU mendaftarkan semua pengguna di klaster, termasuk pengguna CU baru yang baru saja Anda buat.

```
Users on server 0(10.0.2.9):
```

```
Number of users found:3
```

User Id	User Type	User Name	
MofnPubKey	LoginFailureCnt	2FA	
1	0	admin	NO
2	0	app_user	NO
3	0	example_user	NO

```
Users on server 1(10.0.3.11):
```

```
Number of users found:3
```

User Id	User Type	User Name
MofnPubKey	LoginFailureCnt	2FA

```

      1      0      CO      NO      admin      NO
      2      0      AU      NO      app_user      NO
      3      0      CU      NO      example_user      NO
Users on server 2(10.0.1.12):
Number of users found:3

      User Id      User Type      User Name      2FA
MofnPubKey      LoginFailureCnt
      1      CO      admin      NO
      0      NO
      2      AU      app_user      NO
      0      NO
      3      CU      example_user      NO
      0      NO

```

8. Gunakan `logoutHSM` perintah untuk keluar dari file HSMs.

```
aws-cloudhsm>logoutHSM
```

```
logoutHSM success on server 0(10.0.2.9)
logoutHSM success on server 1(10.0.3.11)
logoutHSM success on server 2(10.0.1.12)

```

9. Gunakan perintah `quit` untuk menghentikan `cloudhsm_mgmt_util`.

```
aws-cloudhsm>quit
```

```
disconnecting from servers, please wait...
```

Instal dan konfigurasi AWS CloudHSM klien untuk CMU (Linux)

Untuk berinteraksi dengan modul keamanan perangkat keras (HSM) di AWS CloudHSM cluster Anda menggunakan `cloudhsm_mgmt_util` (CMU), Anda memerlukan perangkat lunak klien untuk Linux. AWS CloudHSM Anda harus menginstalnya di instance klien Amazon EC2 Linux yang Anda buat sebelumnya. Anda juga dapat menginstal klien jika Anda menggunakan Windows. Untuk informasi selengkapnya, lihat [Instal dan konfigurasi AWS CloudHSM klien untuk CMU \(Windows\)](#).

Tugas

- [Langkah 1. Instal AWS CloudHSM klien dan alat baris perintah](#)
- [Langkah 2. Edit konfigurasi klien](#)

Langkah 1. Instal AWS CloudHSM klien dan alat baris perintah

Connect ke instance klien Anda dan jalankan perintah berikut untuk men-download dan menginstal AWS CloudHSM client dan command line tools.

Amazon Linux

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-latest.el6.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm
```

RHEL 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

RHEL 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client_latest_amd64.deb
```

```
sudo apt install ./cloudhsm-client_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client_latest_u18.04_amd64.deb
```

```
sudo apt install ./cloudhsm-client_latest_u18.04_amd64.deb
```

Langkah 2. Edit konfigurasi klien

Sebelum Anda dapat menggunakan AWS CloudHSM klien untuk terhubung ke cluster Anda, Anda harus mengedit konfigurasi klien.

Untuk mengedit konfigurasi klien

1. Jika menginstal Client SDK 3 di `cloudhsm_mgmt_util`, selesaikan langkah-langkah berikut untuk memastikan semua node di cluster disinkronkan.
 - a. Jalankan `configure -a <IP of one of the HSMs>`.
 - b. Mulai ulang layanan klien.
 - c. Jalankan `configure -m`.
2. Salin sertifikat penerbitan Anda—[salah satu yang Anda gunakan untuk menandatangani sertifikat klaster](#)—ke lokasi berikut pada instans klien: `/opt/cloudhsm/etc/customerCA.crt`. Anda perlu izin pengguna root instans pada instans klien untuk menyalin sertifikat Anda ke lokasi ini.
3. Gunakan perintah [konfigurasi](#) berikut untuk memperbarui file konfigurasi untuk AWS CloudHSM klien dan alat baris perintah, menentukan alamat IP HSM di cluster Anda. Untuk mendapatkan alamat IP HSM, lihat cluster Anda di [AWS CloudHSM konsol](#), atau jalankan [describe-clusters](#) AWS CLI perintah. Dalam output perintah, alamat IP HSM adalah nilai bidang `EniIp`. Jika Anda memiliki lebih dari satu HSM, pilih alamat IP untuk salah satu HSMs; tidak masalah yang mana.

```
sudo /opt/cloudhsm/bin/configure -a <IP address>

Updating server config in /opt/cloudhsm/etc/cloudhsm_client.cfg
Updating server config in /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

4. Kunjungi [Aktifkan cluster di AWS CloudHSM](#).

Instal dan konfigurasikan AWS CloudHSM klien untuk CMU (Windows)

Untuk bekerja dengan modul keamanan perangkat keras (HSM) di AWS CloudHSM cluster Anda di Windows menggunakan `cloudhsm_mgmt_util` (CMU), Anda memerlukan perangkat lunak klien untuk Windows. AWS CloudHSM Anda harus menginstalnya pada instans Windows Server yang Anda buat sebelumnya.

 Note

- Jika Anda memperbarui klien, file konfigurasi yang ada dari instalasi sebelumnya tidak ditimpa.
- Penginstal AWS CloudHSM klien untuk Windows secara otomatis mendaftarkan API Kriptografi: Generasi Berikutnya (CNG) dan Penyedia Penyimpanan Kunci (KSP). Untuk mencopot pemasangan klien, jalankan penginstal lagi dan ikuti petunjuk mencopot pemasangan.
- Jika Anda menggunakan Linux, Anda dapat menginstal klien Linux. Untuk informasi selengkapnya, lihat [Instal dan konfigurasi AWS CloudHSM klien untuk CMU \(Linux\)](#).

Untuk menginstal (atau memperbarui) klien Windows terbaru dan alat baris perintah

1. Hubungkan ke instans Windows Server Anda.
2. Unduh [AWSCloudHSMClient-latestpenginstal.msi](#).
3. Jika menginstal Client SDK 3 di cloudhsm_mgmt_util, selesaikan langkah-langkah berikut untuk memastikan semua node di cluster disinkronkan.
 - a. Jalankan `configure.exe -a <IP of one of the HSMs>`.
 - b. Mulai ulang layanan klien.
 - c. Jalankan `configure.exe -m`.
4. Buka lokasi unduhan Anda dan jalankan penginstal (`AWSCloudHSMClient-latest.msi`) dengan hak administratif.
5. Ikuti petunjuk penginstal, lalu pilih Tutup setelah penginstal selesai.
6. Salin sertifikat penerbitan yang ditandatangani sendiri [-salah satu yang Anda gunakan untuk menandatangani sertifikat klaster](#)—ke folder `C:\ProgramData\Amazon\CloudHSM`.
7. Jalankan perintah berikut untuk memperbarui file konfigurasi Anda. Pastikan untuk berhenti dan memulai klien selama konfigurasi ulang jika Anda memperbaruinya:

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" -a <HSM IP address>
```

8. Kunjungi [Aktifkan cluster di AWS CloudHSM](#).

Referensi untuk perintah Utilitas AWS CloudHSM Manajemen

Alat baris perintah AWS CloudHSM `cloudhsm_mgmt_util` membantu petugas krypto mengelola pengguna dalam modul keamanan perangkat keras () di cluster. HSMs AWS CloudHSM Ini juga mencakup perintah yang memungkinkan pengguna krypto (CUs) untuk berbagi kunci, dan mendapatkan dan mengatur atribut kunci. Perintah ini melengkapi perintah manajemen kunci primer di alat baris perintah [key_mgmt_util](#).

Untuk memulai dengan cepat, lihat [Memulai dengan AWS CloudHSM Management Utility \(CMU\)](#).

Sebelum menjalankan perintah `cloudhsm_mgmt_util`, Anda harus memulai `cloudhsm_mgmt_util` dan masuk ke HSM. Pastikan bahwa Anda masuk dengan jenis akun pengguna yang dapat menjalankan perintah yang ingin Anda gunakan.

Untuk membuat daftar semua perintah `cloudhsm_mgmt_util`, jalankan perintah berikut:

```
aws-cloudhsm> help
```

Untuk mendapatkan sintaks untuk perintah `cloudhsm_mgmt_util`, jalankan perintah berikut:

```
aws-cloudhsm> help <command-name>
```

Note

Gunakan sintaks sesuai dokumentasi. Walaupun bantuan perangkat lunak bawaan dapat memberikan opsi tambahan, ini tidak boleh dianggap didukung dan tidak boleh digunakan dalam kode produksi.

Untuk menjalankan perintah, masukkan nama perintah, atau cukup nama untuk membedakannya dari nama perintah `cloudhsm_mgmt_util` lainnya.

Misalnya, untuk mendapatkan daftar pengguna di HSMs, masukkan `listUsers` atau `listU`.

```
aws-cloudhsm> listUsers
```

Untuk mengakhiri sesi `cloudhsm_mgmt_util` Anda, jalankan perintah berikut:

```
aws-cloudhsm> quit
```

Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

Topik berikut menjelaskan perintah di `cloudhsm_mgmt_util`.

 Note

Beberapa perintah di `key_mgmt_util` dan `cloudhsm_mgmt_util` memiliki nama yang sama. Namun, perintah biasanya memiliki sintaks yang berbeda, output yang berbeda, dan fungsi yang sedikit berbeda.

Perintah	Deskripsi	Jenis Pengguna
ChangePSWD	Mengubah kata sandi pengguna di file HSMs. Setiap pengguna dapat mengubah kata sandi mereka sendiri. COs dapat mengubah kata sandi siapa pun.	CO
CreateUser	Menciptakan pengguna dari semua jenis di HSMs.	CO
DeleteUser	Menghapus pengguna dari semua jenis dari HSMs	CO
findAllKeys	Mendapatkan kunci yang pengguna memiliki atau bagikan. Juga mendapatkan hash dari kepemilikan kunci dan berbagi data untuk semua kunci pada setiap HSM.	CO, AU
getAttribute	Mendapat nilai atribut untuk AWS CloudHSM kunci dan menuliskannya ke file atau stdout (output standar).	CU

Perintah	Deskripsi	Jenis Pengguna
<u>mendapatkan HSMInfo</u>	Mendapat informasi tentang perangkat keras tempat HSM berjalan.	Semua. Login tidak diperlukan.
<u>getKeyInfo</u>	Mendapatkan pemilik, pengguna bersama, dan status autentikasi kuorum kunci.	Semua. Login tidak diperlukan.
<u>info</u>	Mendapat informasi tentang HSM, termasuk alamat IP, nama host, port, dan pengguna saat ini.	Semua. Login tidak diperlukan.
<u>ListUsers</u>	Mendapat pengguna di masing-masing HSMs, jenis pengguna dan ID mereka, dan atribut lainnya.	Semua. Login tidak diperlukan.
<u>LoginHSM dan LogouthSM</u>	Login dan log out dari HSM.	Semua.
<u>berhenti</u>	Keluar dari cloudhsm_mgmt_util.	Semua. Login tidak diperlukan.
<u>server</u>	Memasuki dan keluar modus server pada HSM.	Semua.
<u>registerQuorumPubKunci</u>	Mengaitkan pengguna HSM dengan pasangan kunci RSA-2048 asimetris.	CO
<u>setAttribute</u>	Mengubah nilai-nilai label, mengenkripsi, mendekripsi, membungkus, dan membuka atribut kunci yang ada.	CU

Perintah	Deskripsi	Jenis Pengguna
ShareKey	Berbagi kunci yang ada dengan pengguna lain.	CU
SyncKey	Menyinkronkan kunci di seluruh kloning kloning AWS CloudHSM .	CU, CO
SyncUser	Menyinkronkan pengguna di seluruh kloning kloning AWS CloudHSM .	CO

Mengubah kata sandi pengguna menggunakan CMU

Gunakan `changePswd` perintah di AWS CloudHSM `cloudhsm_mgmt_util` (CMU) untuk mengubah kata sandi pengguna yang ada pada modul keamanan perangkat keras (HSM) di cluster. AWS CloudHSM

Setiap pengguna dapat mengubah kata sandi mereka sendiri. Selain itu, petugas Crypto (COs dan PCOs) dapat mengubah kata sandi pengguna CO atau krypto lain (CU). Anda tidak perlu memasukkan kata sandi saat ini untuk melakukan perubahan.

Note

Anda tidak dapat mengubah kata sandi pengguna yang saat ini masuk ke AWS CloudHSM klien atau `key_mgmt_util`.

Untuk memecahkan `changePswd`

Sebelum Anda menjalankan perintah CMU, Anda harus memulai CMU dan masuk ke HSM. Pastikan Anda masuk dengan tipe pengguna yang dapat menjalankan perintah yang ingin Anda gunakan.

Jika Anda menambah atau menghapus HSMs, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSMs di cluster.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Petugas krito (CO)
- Pengguna krypto (CU)

Sintaksis

Masukkan argumen dalam urutan yang ditentukan dalam diagram sintaks. Gunakan parameter `-hpswd` untuk menutupi kata sandi Anda. Untuk mengaktifkan autentikasi dua faktor (2FA) untuk pengguna CO, gunakan parameter `-2fa` dan termasuk jalur file. Untuk informasi selengkapnya, lihat [the section called “Pendapat”](#).

```
changePswd <user-type> <user-name> <password> [-hpswd] [-2fa </path/to/authdata>]
```

Contoh

Contoh berikut menunjukkan cara menggunakan `changePassword` untuk mengatur ulang kata sandi untuk pengguna saat ini atau pengguna lain di Anda HSMs.

Example: Ubah kata sandi Anda

Setiap pengguna di HSMs dapat menggunakan `changePswd` untuk mengubah kata sandi mereka sendiri. Sebelum Anda mengubah kata sandi, gunakan [info](#) untuk mendapatkan informasi tentang masing-masing HSMs di cluster, termasuk nama pengguna dan jenis pengguna dari pengguna yang masuk.

Output berikut menunjukkan bahwa Bob saat ini login sebagai pengguna krypto (CU).

```
aws-cloudhsm> info server 0
```

Id	Name	Hostname	Port	State	Partition
0	10.1.9.193	10.1.9.193	2225	Connected	hsm-jqici4covtv

```
  LoginState
  Logged in as 'bob(CU)'
```

```
aws-cloudhsm> info server 1
```

Id	Name	Hostname	Port	State	Partition
1	10.1.10.7	10.1.10.7	2225	Connected	hsm-ogi3sywxbqx

```
  LoginState
  Logged in as 'bob(CU)'
```

Untuk mengubah kata sandi, Bob menjalankan `changePswd` diikuti dengan jenis pengguna, nama pengguna, dan kata sandi baru.

```
aws-cloudhsm> changePswd CU bob newPassword
```

```
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****

Do you want to continue(y/n)?y
Changing password for bob(CU) on 2 nodes
```

Example: Ubah kata sandi pengguna lain

Anda harus menjadi CO atau PCO untuk mengubah kata sandi CO lain, atau CU pada HSMs. Sebelum Anda mengubah kata sandi untuk pengguna lain, gunakan perintah [Info](#) untuk mengonfirmasi bahwa jenis pengguna Anda adalah CO atau PCO.

Output berikut menegaskan bahwa Alice, yang merupakan CO, saat ini sedang login.

```
aws-cloudhsm>info server 0
```

Id	Name	Hostname	Port	State	Partition
0	10.1.9.193	10.1.9.193	2225	Connected	hsm-jqici4covtv

LoginState
Logged in as 'alice(CO)'

```
aws-cloudhsm>info server 1
```

Id	Name	Hostname	Port	State	Partition
0	10.1.10.7	10.1.10.7	2225	Connected	hsm-ogi3sywxbqx

LoginState
Logged in as 'alice(CO)'

Alice ingin mengatur ulang kata sandi dari pengguna lain, John. Sebelum mengganti kata sandi, dia menggunakan perintah [listUsers](#) untuk memverifikasi jenis pengguna John.

Output berikut mendaftar John sebagai pengguna CO.

```
aws-cloudhsm> listUsers
Users on server 0(10.1.9.193):
Number of users found:5
```

User Id	User Type	User Name	MofnPubKey
LoginFailureCnt	2FA		
1	PCO	admin	YES
NO			0
2	AU	jane	NO
NO			0
3	CU	bob	NO
NO			0
4	CU	alice	NO
NO			0
5	CO	john	NO
NO			0

```
Users on server 1(10.1.10.7):
Number of users found:5
```

User Id	User Type	User Name	MofnPubKey
LoginFailureCnt	2FA		
1	PCO	admin	YES
NO			0
2	AU	jane	NO
NO			0
3	CU	bob	NO
NO			0
4	CO	alice	NO
NO			0
5	CO	john	NO
NO			0

Untuk mengubah kata sandi, Alice menjalankan `changePswd` diikuti dengan jenis pengguna John, nama pengguna, dan kata sandi baru.

```
aws-cloudhsm>changePswd CO john newPassword
```

```
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****

Do you want to continue(y/n)?y
Changing password for john(CO) on 2 nodes
```

Pendapat

Masukkan argumen dalam urutan yang ditentukan dalam diagram sintaks. Gunakan parameter `-hpswd` untuk menutupi kata sandi Anda. Untuk mengaktifkan 2FA untuk pengguna CO, gunakan parameter `-2fa` dan termasuk jalur file. Untuk informasi lebih lanjut tentang bekerja dengan 2FA, lihat [Kelola pengguna 2FA](#)

```
changePswd <user-type> <user-name> <password> [-hpswd] [-2fa </path/to/authdata>]
```

<user-type>

Menentukan jenis pengguna saat ini yang kata sandinya Anda ubah. Anda tidak dapat menggunakan `changePswd` untuk mengubah jenis pengguna.

Nilai yang valid adalah CO, CU, PCO, dan PRECO.

Untuk mendapatkan jenis pengguna, gunakan [listUsers](#). Untuk informasi detail tentang jenis pengguna pada HSM, lihat [Jenis pengguna HSM untuk Utilitas AWS CloudHSM Manajemen](#).

Wajib: Ya

<user name>

Menentukan nama yang mudah diingat oleh pengguna. Parameter ini tidak peka huruf besar kecil. Anda tidak dapat menggunakan `changePswd` untuk mengubah nama pengguna.

Wajib: Ya

<password | -hpswd >

Menentukan kata sandi baru untuk pengguna. Masukkan string 7 sampai 32 karakter. Nilai ini peka huruf besar kecil. Kata sandi muncul di plaintext saat Anda mengetikkannya. Untuk

menyembunyikan kata sandi, gunakan parameter `-hpswd` di tempat kata sandi dan ikuti petunjuknya.

Wajib: Ya

`[-2fa path/to/authdata </ >]`

Menentukan pengaktifan 2FA untuk pengguna CO ini. Untuk mendapatkan data yang diperlukan untuk menyiapkan 2FA, sertakan jalur ke lokasi dalam sistem file dengan nama file setelah parameter `-2fa`. Untuk informasi lebih lanjut tentang bekerja dengan 2FA, lihat [Kelola pengguna 2FA](#).

Wajib: Tidak

Topik terkait

- [Info](#)
- [ListUsers](#)
- [CreateUser](#)
- [DeleteUser](#)

Buat AWS CloudHSM pengguna dengan CMU

Gunakan `createUser` perintah di `cloudhsm_mgmt_util` (CMU) untuk membuat pengguna pada modul keamanan perangkat keras (HSM) di cluster. AWS CloudHSM Hanya petugas crypto (COs dan PRECOs) yang dapat menjalankan perintah ini. Ketika perintah berhasil, itu menciptakan pengguna di semua HSMs dalam cluster.

Untuk memecahkan masalah `createUser`

Jika konfigurasi HSM Anda tidak akurat, pengguna mungkin tidak dibuat pada semua. HSMs Untuk menambahkan pengguna ke salah satu HSMs yang hilang, gunakan perintah [Syncuser](#) atau `createUser` hanya HSMs pada yang hilang pengguna itu. Untuk mencegah kesalahan konfigurasi, jalankan alat [konfigurasi](#) dengan alat pilihan `-m`.

Sebelum Anda menjalankan perintah CMU, Anda harus memulai CMU dan masuk ke HSM. Pastikan Anda masuk dengan tipe pengguna yang dapat menjalankan perintah yang Anda rencanakan untuk digunakan.

Jika Anda menambah atau menghapus HSMs, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSMs di cluster.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Petugas Crypto (CO, PRECO)

Sintaksis

Masukkan argumen dalam urutan yang ditentukan dalam diagram sintaks. Gunakan parameter `-hpswd` untuk menutupi kata sandi Anda. Untuk membuat pengguna CO dengan autentikasi dua faktor (2FA), gunakan parameter `-2fa` dan termasuk jalur file. Untuk informasi selengkapnya, lihat [the section called “Pendapat”](#).

```
createUser <user-type> <user-name> <password> [-hpswd> [-2fa </path/to/authdata>]
```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `createUser` untuk membuat pengguna baru di Anda HSMs.

Example: Buat petugas crypto

Contoh ini membuat crypto officer (CO) di HSMs dalam cluster. Perintah pertama menggunakan [loginHSM](#) untuk masuk ke HSM sebagai pegawai krypto.

```
aws-cloudhsm> loginHSM CO admin 735782961
```

```
loginHSM success on server 0(10.0.0.1)
```

```
loginHSM success on server 1(10.0.0.2)
```

```
loginHSM success on server 1(10.0.0.3)
```

Perintah kedua menggunakan perintah `createUser` untuk membuat `alice`, seorang perwira krypto baru di HSM.

Pesan peringatan menjelaskan bahwa perintah membuat pengguna pada semua yang ada HSMs di cluster. Tetapi, jika perintah gagal pada salah satu HSMs, pengguna tidak akan ada pada itu HSMs. Untuk melanjutkan, ketik `y`.

Output menunjukkan bahwa pengguna baru dibuat pada ketiganya HSMs di cluster.

```
aws-cloudhsm> createUser C0 alice 391019314
```

```
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****
```

```
Do you want to continue(y/n)?Invalid option, please type 'y' or 'n'
```

```
Do you want to continue(y/n)?y
```

```
Creating User alice(C0) on 3 nodes
```

Ketika perintah selesai, alice memiliki izin yang sama pada HSM sebagai pengguna admin CO, termasuk mengubah kata sandi setiap pengguna di file. HSMs

Perintah terakhir menggunakan perintah [ListUsers](#) untuk memverifikasi yang alice ada pada ketiganya HSMs di cluster. Output juga menunjukkan bahwa alice ditetapkan ID pengguna 3.. Anda menggunakan ID pengguna untuk mengidentifikasi alice perintah lain, seperti [findAllKeys](#).

```
aws-cloudhsm> listUsers
```

```
Users on server 0(10.0.0.1):
```

```
Number of users found:3
```

User Id	User Type	User Name	MofnPubKey
LoginFailureCnt	2FA		
1	PRECO	admin	YES
0	NO		
2	AU	app_user	NO
0	NO		
3	C0	alice	NO
0	NO		

```
Users on server 1(10.0.0.2):
```

```
Number of users found:3
```

User Id	User Type	User Name	MofnPubKey
LoginFailureCnt	2FA		
1	PRECO	admin	YES
0	NO		

2	AU	app_user	NO
0	NO		
3	CO	alice	NO
0	NO		

Users on server 1(10.0.0.3):

Number of users found:3

User Id	User Type	User Name	MofnPubKey
1	PRECO	admin	YES
0	NO		
2	AU	app_user	NO
0	NO		
3	CO	alice	NO
0	NO		

Example: Buat pengguna krypto

Contoh ini membuat pengguna krypto (CU), bob, pada HSM. Pengguna krypto dapat membuat dan mengelola kunci, tetapi mereka tidak dapat mengelola pengguna.

Setelah Anda mengetik y untuk menanggapi pesan peringatan, output menunjukkan bahwa bob itu dibuat pada ketiganya HSMs di cluster. CU baru dapat login ke HSM untuk membuat dan mengelola kunci.

Perintah menggunakan nilai kata sandi defaultPassword. Nanti, bob atau CO apa pun dapat menggunakan perintah [changePswd](#) untuk mengubah kata sandinya.

```
aws-cloudhsm> createUser CU bob defaultPassword
```

```
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****
```

```
Do you want to continue(y/n)?Invalid option, please type 'y' or 'n'
```

```
Do you want to continue(y/n)?y
Creating User bob(CU) on 3 nodes
```

Pendapat

Masukkan argumen dalam urutan yang ditentukan dalam diagram sintaks. Gunakan parameter `-hpswd` untuk menutupi kata sandi Anda. Untuk membuat pengguna CO dengan 2FA diaktifkan, gunakan parameter `-2fa` dan termasuk jalur file. Untuk informasi lebih lanjut tentang 2FA, lihat [Kelola pengguna 2FA](#).

```
createUser <user-type> <user-name> <password> | -hpswd> [-2fa </path/to/authdata>]
```

<user-type>

Menentukan jenis pengguna. Parameter ini diperlukan.

Untuk informasi detail tentang jenis pengguna pada HSM, lihat [Jenis pengguna HSM untuk Utilitas AWS CloudHSM Manajemen](#).

Nilai yang valid:

- CO: Petugas krypto dapat mengelola pengguna, tetapi mereka tidak dapat mengelola kunci.
- CU: Pengguna krypto dapat membuat mengelola kunci dan menggunakan kunci dalam operasi kriptografi.

PRECO dikonversi ke CO ketika Anda menetapkan kata sandi selama aktivasi [HSM](#).

Wajib: Ya

<user name>

Menentukan nama yang mudah diingat untuk pengguna. Panjang maksimum adalah 31 karakter. Satu-satunya karakter khusus yang diizinkan adalah garis bawah (_).

Anda tidak dapat mengubah nama pengguna setelah dibuat. Dalam perintah `cloudhsm_mgmt_util`, jenis pengguna dan kata sandi peka huruf besar-kecil, tetapi nama pengguna tidak.

Wajib: Ya

<password | -hpswd >

Menentukan kata sandi untuk pengguna. Masukkan string 7 sampai 32 karakter. Nilai ini peka huruf besar kecil. Kata sandi muncul di plaintext saat Anda mengetikkannya. Untuk menyembunyikan kata sandi, gunakan parameter `-hpswd` di tempat kata sandi dan ikuti petunjuknya.

Untuk mengubah kata sandi pengguna, gunakan [changePswd](#). Setiap pengguna HSM dapat mengubah kata sandi mereka sendiri, tetapi pengguna CO dapat mengubah kata sandi pengguna mana pun (dari jenis apa pun) di HSMs.

Wajib: Ya

[-2fa path/to/authdata </ >]

Menentukan pembuatan pengguna CO dengan 2FA diaktifkan. Untuk mendapatkan data yang diperlukan untuk menyiapkan autentikasi 2FA, termasuk jalur ke lokasi dalam sistem file dengan nama file setelah parameter -2fa. Untuk informasi lebih lanjut tentang menyiapkan dan bekerja dengan 2FA, lihat [Kelola pengguna 2FA](#).

Wajib: Tidak

Topik terkait

- [ListUsers](#)
- [DeleteUser](#)
- [SynCuser](#)
- [ChangePswd](#)

Hapus AWS CloudHSM pengguna menggunakan CMU

Gunakan deleteUser perintah di AWS CloudHSM cloudhsm_mgmt_util (CMU) untuk menghapus pengguna dari modul keamanan perangkat keras (HSM) di cluster. AWS CloudHSM Hanya petugas krypto (CO) yang dapat menjalankan perintah ini. Anda tidak dapat menghapus pengguna yang saat ini masuk ke HSM. Untuk informasi lebih lanjut tentang penghapusan pengguna, lihat [Cara Menghapus Pengguna HSM](#).

Tip

Anda tidak dapat menghapus pengguna krypto (CU) yang memiliki kunci.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- CO

Sintaksis

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

```
deleteUser <user-type> <user-name>
```

Contoh

Contoh ini menghapus crypto officer (CO) dari HSMs dalam cluster. Perintah pertama menggunakan [ListUsers](#) untuk mencantumkan semua pengguna di file. HSMs

Output menunjukkan bahwa pengguna 3alice,, adalah CO pada file HSMs.

```
aws-cloudhsm> listUsers
```

```
Users on server 0(10.0.0.1):
```

```
Number of users found:3
```

User Id	User Type	User Name	MofnPubKey
1	PCO	admin	YES
0	NO		
2	AU	app_user	NO
0	NO		
3	CO	alice	NO
0	NO		

```
Users on server 1(10.0.0.2):
```

```
Number of users found:3
```

User Id	User Type	User Name	MofnPubKey
1	PCO	admin	YES
0	NO		
2	AU	app_user	NO
0	NO		
3	CO	alice	NO
0	NO		

```
Users on server 1(10.0.0.3):
```

```
Number of users found:3
```

User Id	User Type	User Name	MofnPubKey
LoginFailureCnt	2FA		
1	PC0	admin	YES
0	N0		
2	AU	app_user	N0
0	N0		
3	C0	alice	N0
0	N0		

Perintah kedua menggunakan `deleteUser` perintah untuk menghapus `alice` dari HSMs.

Output menunjukkan bahwa perintah berhasil pada ketiganya HSMs di cluster.

```
aws-cloudhsm> deleteUser C0 alice
Deleting user alice(C0) on 3 nodes
deleteUser success on server 0(10.0.0.1)
deleteUser success on server 0(10.0.0.2)
deleteUser success on server 0(10.0.0.3)
```

Perintah terakhir menggunakan `listUsers` perintah untuk memverifikasi `alice` yang dihapus dari ketiganya HSMs di cluster.

```
aws-cloudhsm> listUsers
Users on server 0(10.0.0.1):
Number of users found:2
```

User Id	User Type	User Name	MofnPubKey
LoginFailureCnt	2FA		
1	PC0	admin	YES
0	N0		
2	AU	app_user	N0
0	N0		

```
Users on server 1(10.0.0.2):
Number of users found:2
```

User Id	User Type	User Name	MofnPubKey
LoginFailureCnt	2FA		
1	PC0	admin	YES
0	N0		
2	AU	app_user	N0
0	N0		

```
Users on server 1(10.0.0.3):
```

Number of users found:2

User Id	User Type	User Name	MofnPubKey
LoginFailureCnt	2FA		
1	PCO	admin	YES
0	NO		
2	AU	app_user	NO
0	NO		

Pendapat

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

```
deleteUser <user-type> <user-name>
```

<user-type>

Menentukan jenis pengguna. Parameter ini diperlukan.

Tip

Anda tidak dapat menghapus pengguna krypto (CU) yang memiliki kunci.

Nilai yang valid adalah CO, CU.

Untuk mendapatkan jenis pengguna, gunakan [listUsers](#). Untuk informasi detail tentang jenis pengguna pada HSM, lihat [Jenis pengguna HSM untuk Utilitas AWS CloudHSM Manajemen](#).

Wajib: Ya

<user name>

Menentukan nama yang mudah diingat untuk pengguna. Panjang maksimum adalah 31 karakter. Satu-satunya karakter khusus yang diizinkan adalah garis bawah (_).

Anda tidak dapat mengubah nama pengguna setelah dibuat. Dalam perintah `cloudhsm_mgmt_util`, jenis pengguna dan kata sandi peka huruf besar-kecil, tetapi nama pengguna tidak.

Wajib: Ya

Topik terkait

- [ListUsers](#)
- [CreateUser](#)
- [SynCuser](#)
- [ChangePSWD](#)

Buat daftar kunci yang dimiliki pengguna AWS CloudHSM krypto menggunakan CMU

Gunakan `findAllKeys` perintah di AWS CloudHSM `cloudhsm_mgmt_util` (CMU) untuk mendapatkan kunci yang dimiliki atau dibagikan oleh pengguna krypto tertentu (CU). AWS CloudHSM Perintah ini juga mengembalikan hash dari data pengguna pada masing-masing HSMs Anda dapat menggunakan hash untuk menentukan sekilas apakah pengguna, kepemilikan kunci, dan data berbagi kunci sama pada semua HSMs di cluster. Dalam output, kunci yang dimiliki oleh pengguna dianotasi oleh (o) dan kunci bersama dianotasi oleh (s).

`findAllKeys` mengembalikan kunci publik hanya ketika CU yang ditentukan memiliki kunci, meskipun semua CUs di HSM dapat menggunakan kunci publik apa pun. Perilaku ini berbeda dari [findKey](#) di `key_mgmt_util`, yang mengembalikan kunci publik untuk semua pengguna CU.

Hanya petugas krypto (COs dan PCOs) dan pengguna alat (AUs) yang dapat menjalankan perintah ini. Pengguna Crypto (CUs) dapat menjalankan perintah berikut:

- [listUsers](#) untuk menemukan semua pengguna
- [findKey](#) di `key_mgmt_util` untuk menemukan kunci yang dapat mereka gunakan
- [getKeyInfo](#) di `key_mgmt_util` untuk menemukan pemilik dan pengguna bersama dari kunci tertentu yang mereka miliki atau bagikan

Sebelum Anda menjalankan perintah CMU, Anda harus memulai CMU dan masuk ke HSM. Pastikan Anda masuk dengan tipe pengguna yang dapat menjalankan perintah yang Anda rencanakan untuk digunakan.

Jika Anda menambah atau menghapus HSMs, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSMs di cluster.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Petugas krypto (CO, PCO)
- Pengguna peralatan (AU)

Sintaksis

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

```
findAllKeys <user id> <key hash (0/1)> [<output file>]
```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `findAllKeys` untuk menemukan semua kunci untuk pengguna dan mendapatkan hash informasi pengguna kunci pada HSMs masing-masing.

Example: Temukan kunci untuk CU

Contoh ini digunakan `findAllKeys` untuk menemukan kunci di pengguna HSMs yang dimiliki dan dibagikan 4. Perintah menggunakan nilai `0` untuk argumen kedua untuk menekan nilai hash. Karena menghilangkan nama file opsional, perintah menulis ke stdout (output standar).

Output menunjukkan bahwa pengguna 4 dapat menggunakan 6 kunci: 8, 9, 17, 262162, 19, dan 31. Output menggunakan (s) untuk menunjukkan kunci yang secara eksplisit dibagi oleh pengguna. Kunci yang dimiliki pengguna ditunjukkan oleh (o) dan termasuk kunci simetris dan privat yang tidak bagi pengguna, dan kunci publik yang tersedia untuk semua pengguna krypto.

```
aws-cloudhsm> findAllKeys 4 0
Keys on server 0(10.0.0.1):
Number of keys found 6
number of keys matched from start index 0::6
8(s),9(s),17,262162(s),19(o),31(o)
findAllKeys success on server 0(10.0.0.1)

Keys on server 1(10.0.0.2):
Number of keys found 6
number of keys matched from start index 0::6
8(s),9(s),17,262162(s),19(o),31(o)
findAllKeys success on server 1(10.0.0.2)

Keys on server 1(10.0.0.3):
Number of keys found 6
```

```
number of keys matched from start index 0::6
8(s),9(s),17,262162(s),19(o),31(o)
findAllKeys success on server 1(10.0.0.3)
```

Example: Verifikasi bahwa data pengguna disinkronkan

Contoh ini digunakan `findAllKeys` untuk memverifikasi bahwa semua HSMs dalam cluster berisi pengguna yang sama, kepemilikan kunci, dan nilai berbagi kunci. Untuk melakukan hal ini, dapatkan hash dari data pengguna kunci pada setiap HSM dan membandingkan nilai-nilai hash.

Untuk mendapatkan hash kunci, perintah menggunakan nilai 1 dalam argumen kedua. Nama file opsional dihilangkan, sehingga perintah menulis hash kunci untuk stdout.

Contoh menentukan pengguna6, tetapi nilai hash akan sama untuk setiap pengguna yang memiliki atau berbagi salah satu kunci pada HSMs. Jika pengguna tertentu tidak memiliki atau berbagi kunci apa pun, seperti CO, perintah tidak mengembalikan nilai hash.

Output menunjukkan bahwa hash kunci identik dengan kedua HSMs di cluster. Jika salah satu HSM memiliki pengguna yang berbeda, pemilik kunci yang berbeda, atau pengguna bersama yang berbeda, nilai hash kunci tidak akan sama.

```
aws-cloudhsm> findAllKeys 6 1
Keys on server 0(10.0.0.1):
Number of keys found 3
number of keys matched from start index 0::3
8(s),9(s),11,17(s)
Key Hash:
55655676c95547fd4e82189a072ee1100eccfca6f10509077a0d6936a976bd49

findAllKeys success on server 0(10.0.0.1)
Keys on server 1(10.0.0.2):
Number of keys found 3
number of keys matched from start index 0::3
8(s),9(s),11(o),17(s)
Key Hash:
55655676c95547fd4e82189a072ee1100eccfca6f10509077a0d6936a976bd49

findAllKeys success on server 1(10.0.0.2)
```

Perintah ini menunjukkan bahwa nilai hash mewakili data pengguna untuk semua kunci pada HSM. Perintah menggunakan `findAllKeys` untuk pengguna 3. Tidak seperti pengguna 6, yang memiliki atau

berbagi hanya 3 kunci, pengguna 3 memiliki atau berbeagi 17 kunci, tetapi nilai hash kunci adalah sama.

```
aws-cloudhsm> findAllKeys 3 1
Keys on server 0(10.0.0.1):
Number of keys found 17
number of keys matched from start index 0::17
6(o),7(o),8(s),11(o),12(o),14(o),262159(o),262160(o),17(s),262162(s),19(s),20(o),21(o),262177(o)
Key Hash:
55655676c95547fd4e82189a072ee1100eccfca6f10509077a0d6936a976bd49

findAllKeys success on server 0(10.0.0.1)
Keys on server 1(10.0.0.2):
Number of keys found 17
number of keys matched from start index 0::17
6(o),7(o),8(s),11(o),12(o),14(o),262159(o),262160(o),17(s),262162(s),19(s),20(o),21(o),262177(o)
Key Hash:
55655676c95547fd4e82189a072ee1100eccfca6f10509077a0d6936a976bd49

findAllKeys success on server 1(10.0.0.2)
```

Pendapat

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

```
findAllKeys <user id> <key hash (0/1)> [<output file>]
```

<user id>

Mendapat semua kunci yang pengguna tertentu miliki atau bagikan. Masukkan ID pengguna pengguna di file HSMs. Untuk menemukan pengguna IDs dari semua pengguna, gunakan [ListUsers](#).

IDs Semua pengguna valid, tetapi `findAllKeys` mengembalikan kunci hanya untuk pengguna krypto (CUs).

Wajib: Ya

<key hash>

Mencakup (1) atau mengecualikan (0) hash dari kepemilikan pengguna dan berbagi data untuk semua kunci di setiap HSM.

Saat argumen `user_id` mewakili pengguna yang memiliki atau berbagi kunci, hash kunci diisi. Nilai hash kunci identik untuk semua pengguna yang memiliki atau berbagi kunci pada HSM, meskipun mereka memiliki dan berbagi kunci yang berbeda. Namun, ketika `user_id` mewakili pengguna yang tidak memiliki atau berbagi kunci apa pun, seperti CO, nilai hash tidak diisi.

Wajib: Ya

<output file>

Menulis output ke file yang ditentukan.

Wajib: Tidak

Default: Stdout

Topik terkait

- [ChangePSWD](#)
- [DeleteUser](#)
- [ListUsers](#)
- [SyncUser](#)
- [findKey](#) di `key_mgmt_util`
- [getKeyInfo](#) di `key_mgmt_util`

Dapatkan nilai atribut AWS CloudHSM kunci menggunakan CMU

Gunakan `getAttribute` perintah di AWS CloudHSM `cloudhsm_mgmt_util` (CMU) untuk mendapatkan satu nilai atribut untuk kunci dari semua modul keamanan perangkat keras (HSM) di AWS CloudHSM cluster dan tulis ke stdout (output standar) atau ke file. Hanya pengguna crypto (CUs) yang dapat menjalankan perintah ini.

Atribut Kunci adalah properti dari kunci. Properti tersebut termasuk karakteristik, seperti jenis kunci, kelas, label, dan ID, dan nilai-nilai yang mewakili tindakan yang dapat Anda lakukan pada kunci, seperti mengenkripsi, mendekripsi, membungkus, menandatangani, dan memverifikasi.

Anda dapat menggunakan `getAttribute` hanya pada tombol yang Anda miliki dan kunci yang dibagikan dengan Anda. Anda dapat menjalankan perintah ini atau perintah [getAttribute](#) di `key_mgmt_util`, yang menulis satu atau semua nilai atribut kunci ke file.

Untuk mendapatkan daftar atribut dan konstanta yang mewakilinya, gunakan perintah [listAttributes](#). Untuk mengubah nilai atribut kunci yang ada, gunakan [setAttribute](#) di `key_mgmt_util` dan [setAttribute](#) di `cloudhsm_mgmt_util`. Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

Sebelum Anda menjalankan perintah CMU, Anda harus memulai CMU dan masuk ke HSM. Pastikan Anda masuk dengan tipe pengguna yang dapat menjalankan perintah yang Anda rencanakan untuk digunakan.

Jika Anda menambah atau menghapus HSMs, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSMs di cluster.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Pengguna krypto (CU)

Sintaksis

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

```
getAttribute <key handle> <attribute id> [<filename>]
```

Contoh

Contoh ini mendapatkan nilai atribut yang dapat diekstrak untuk kunci di HSMs Anda dapat menggunakan perintah seperti ini untuk menentukan apakah Anda dapat mengekspor kunci dari HSMs.

Perintah pertama menggunakan [listAttributes](#) untuk menemukan konstanta yang mewakili atribut yang dapat diekstrak. Output menunjukkan bahwa konstanta untuk `OBJ_ATTR_EXTRACTABLE` adalah 354. Anda juga dapat menemukan informasi ini dengan deskripsi atribut dan nilai-nilainya dalam [AWS CloudHSM referensi atribut kunci untuk KMU](#).

```
aws-cloudhsm> listAttributes
```

```
Following are the possible attribute values for getAttribute:
```

```
OBJ_ATTR_CLASS          = 0
```

OBJ_ATTR_TOKEN	= 1
OBJ_ATTR_PRIVATE	= 2
OBJ_ATTR_LABEL	= 3
OBJ_ATTR_TRUSTED	= 134
OBJ_ATTR_KEY_TYPE	= 256
OBJ_ATTR_ID	= 258
OBJ_ATTR_SENSITIVE	= 259
OBJ_ATTR_ENCRYPT	= 260
OBJ_ATTR_DECRYPT	= 261
OBJ_ATTR_WRAP	= 262
OBJ_ATTR_UNWRAP	= 263
OBJ_ATTR_SIGN	= 264
OBJ_ATTR_VERIFY	= 266
OBJ_ATTR_DERIVE	= 268
OBJ_ATTR_LOCAL	= 355
OBJ_ATTR_MODULUS	= 288
OBJ_ATTR_MODULUS_BITS	= 289
OBJ_ATTR_PUBLIC_EXPONENT	= 290
OBJ_ATTR_VALUE_LEN	= 353
OBJ_ATTR_EXTRACTABLE	= 354
OBJ_ATTR_NEVER_EXTRACTABLE	= 356
OBJ_ATTR_ALWAYS_SENSITIVE	= 357
OBJ_ATTR_DESTROYABLE	= 370
OBJ_ATTR_KCV	= 371
OBJ_ATTR_WRAP_WITH_TRUSTED	= 528
OBJ_ATTR_WRAP_TEMPLATE	= 1073742353
OBJ_ATTR_UNWRAP_TEMPLATE	= 1073742354
OBJ_ATTR_ALL	= 512

Perintah kedua digunakan `getAttribute` untuk mendapatkan nilai atribut yang dapat diekstrak untuk kunci dengan pegangan kunci 262170 di HSMs. Untuk menentukan atribut yang dapat diekstrak, perintah menggunakan 354, konstanta yang mewakili atribut. Karena perintah tidak menentukan nama file, `getAttribute` menulis output ke stdout.

Output menunjukkan bahwa nilai atribut yang dapat diekstrak adalah 1 pada semua HSM. Nilai ini menunjukkan bahwa pemilik kunci dapat mengekspornya. Ketika nilainya 0 (0x0), itu tidak dapat diekspor dari. HSMs Anda menetapkan nilai atribut yang dapat diekstrak ketika Anda membuat kunci, tetapi Anda tidak dapat mengubahnya.

```
aws-cloudhsm> getAttribute 262170 354
```

```
Attribute Value on server 0(10.0.1.10):  
OBJ_ATTR_EXTRACTABLE  
0x00000001
```

```
Attribute Value on server 1(10.0.1.12):  
OBJ_ATTR_EXTRACTABLE  
0x00000001
```

```
Attribute Value on server 2(10.0.1.7):  
OBJ_ATTR_EXTRACTABLE  
0x00000001
```

Pendapat

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

```
getAttribute <key handle> <attribute id> [<filename>]
```

<key-handle>

Menentukan handel kunci dari kunci target. Anda hanya dapat menentukan satu kunci di setiap perintah. Untuk mendapatkan handel kunci dari suatu kunci, gunakan [findKey](#) di key_mgmt_util.

Anda harus memiliki kunci tertentu atau kunci itu harus dibagi dengan Anda. Untuk menemukan pengguna kunci, gunakan [getKeyInfo](#) di key_mgmt_util.

Wajib: Ya

<attribute id>

Mengidentifikasi atribut. Masukkan konstanta yang mewakili atribut, atau 512, yang mewakili semua atribut. Sebagai contoh, untuk mendapatkan jenis kunci, masukkan 256, yang merupakan konstanta untuk atribut OBJ_ATTR_KEY_TYPE.

Untuk mendaftar atribut dan konstantanya, gunakan [listAttributes](#). Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

Wajib: Ya

<filename>

Menulis output ke file yang ditentukan. Masukkan jalur file.

Jika file yang ditentukan ada, `getAttribute` menimpa file tanpa peringatan.

Wajib: Tidak

Default: Stdout

Topik terkait

- [getAttribute](#) di `key_mgmt_util`
- [ListatTributes](#)
- [setAttribute](#) di `cloudhsm_mgmt_util`
- [setAttribute](#) di `key_mgmt_util`
- [Referensi Atribut Utama](#)

Dapatkan informasi perangkat keras untuk setiap HSM dalam AWS CloudHSM cluster dengan CMU

Gunakan `getHSMInfo` perintah di AWS CloudHSM `cloudhsm_mgmt_util` (CMU) untuk mendapatkan informasi tentang perangkat keras tempat setiap modul keamanan perangkat keras (HSM) berjalan, termasuk model, nomor seri, status FIPS, memori, suhu, dan nomor versi perangkat keras dan firmware. Informasi tersebut juga mencakup ID server yang digunakan `cloudhsm_mgmt_util` untuk merujuk ke HSM.

Sebelum Anda menjalankan perintah CMU, Anda harus memulai CMU dan masuk ke HSM. Pastikan Anda masuk dengan tipe pengguna yang dapat menjalankan perintah yang ingin Anda gunakan.

Jika Anda menambah atau menghapus HSMs, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSMs di cluster.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Semua pengguna. Anda tidak perlu masuk untuk menjalankan perintah ini.

Sintaksis

Perintah ini tidak memiliki parameter.


```
getHSMInfo
```

Contoh

Contoh ini digunakan `getHSMInfo` untuk mendapatkan informasi tentang HSMs di cluster.

```
aws-cloudhsm> getHSMInfo
Getting HSM Info on 3 nodes
*** Server 0 HSM Info ***

Label           :cavium
Model           :NITROX-III CNN35XX-NFBE

Serial Number    :3.0A0101-ICM000001
HSM Flags       :0
FIPS state      :2 [FIPS mode with single factor authentication]

Manufacturer ID  :
Device ID       :10
Class Code      :100000
System vendor ID :177D
SubSystem ID    :10

TotalPublicMemory :560596
FreePublicMemory  :294568
TotalPrivateMemory :0
FreePrivateMemory :0

Hardware Major   :3
Hardware Minor   :0

Firmware Major   :2
Firmware Minor   :03

Temperature      :56 C

Build Number     :13

Firmware ID      :xxxxxxxxxxxxxxxx

...
```

Topik terkait

- [info](#)

Dapatkan info AWS CloudHSM pengguna tentang kunci menggunakan CMU

Gunakan `getKeyInfo` perintah di AWS CloudHSM `key_mgmt_util` (KMU) untuk mengembalikan pengguna modul keamanan perangkat keras (HSM) pengguna IDs yang dapat menggunakan kunci, termasuk pemilik dan pengguna kriptografi (CU) dengan siapa kunci tersebut dibagikan. Ketika autentikasi kuorum diaktifkan pada kunci, `getKeyInfo` juga mengembalikan jumlah pengguna yang harus menyetujui operasi kriptografi yang menggunakan kunci. Anda dapat menjalankan `getKeyInfo` hanya pada kunci yang Anda miliki dan kunci yang dibagikan dengan Anda.

Ketika Anda menjalankan `getKeyInfo` pada kunci publik, `getKeyInfo` mengembalikan hanya pemilik kunci, meskipun semua pengguna HSM dapat menggunakan kunci publik. Untuk menemukan pengguna HSM pengguna IDs di Anda HSMs, gunakan [ListUsers](#). Untuk menemukan kunci untuk pengguna tertentu, gunakan [findKey](#) -u di `key_mgmt_util`. Petugas Crypto dapat menggunakan [findAllKeys](#) di `cloudhsm_mgmt_util`.

Anda memiliki kunci yang Anda buat. Anda dapat berbagi kunci dengan pengguna lain saat Anda membuatnya. Kemudian, untuk berbagi atau batal berbagi kunci yang ada, gunakan [shareKey](#) di `cloudhsm_mgmt_util`.

Sebelum Anda menjalankan perintah CMU, Anda harus memulai CMU dan masuk ke HSM. Pastikan Anda masuk dengan tipe pengguna yang dapat menjalankan perintah yang Anda rencanakan untuk digunakan.

Jika Anda menambah atau menghapus HSMs, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSMs di cluster.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna kriptografi (CU)

Sintaksis

```
getKeyInfo -k <key-handle> [<output file>]
```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `getKeyInfo` untuk mendapatkan informasi tentang pengguna kunci.

Example: Dapatkan pengguna untuk kunci asimetris

Perintah ini mendapatkan pengguna yang dapat menggunakan kunci AES (asimetris) kunci dengan handel kunci 262162. Output menunjukkan bahwa pengguna 3 memiliki kunci dan telah membaginya dengan pengguna 4 dan 6.

Hanya pengguna 3, 4, dan 6 yang dapat menjalankan `getKeyInfo` pada kunci 262162.

```
aws-cloudhsm>getKeyInfo 262162
Key Info on server 0(10.0.0.1):

    Token/Flash Key,

    Owned by user 3

    also, shared to following 2 user(s):

        4
        6
Key Info on server 1(10.0.0.2):

    Token/Flash Key,

    Owned by user 3

    also, shared to following 2 user(s):

        4
        6
```

Example: Dapatkan pengguna untuk symmetric key pair

Perintah ini menggunakan `getKeyInfo` untuk mendapatkan pengguna yang dapat menggunakan kunci dalam [pasangan kunci ECC \(simetris\)](#). Kunci publik memiliki handel kunci 262179. Kunci privat memiliki handel kunci 262177.

Ketika Anda menjalankan `getKeyInfo` pada kunci pribadi (262177), ia mengembalikan pemilik kunci (3) dan pengguna kriptografi (CUs) 4, dengan siapa kunci tersebut dibagikan.

```
aws-cloudhsm>getKeyInfo -k 262177
Key Info on server 0(10.0.0.1):

    Token/Flash Key,

    Owned by user 3

    also, shared to following 1 user(s):

        4

Key Info on server 1(10.0.0.2):

    Token/Flash Key,

    Owned by user 3

    also, shared to following 1 user(s):

        4
```

Ketika Anda menjalankan `getKeyInfo` pada kunci publik (262179), perintah ini mengembalikan hanya pemilik kunci, pengguna 3.

```
aws-cloudhsm>getKeyInfo -k 262179
Key Info on server 0(10.0.3.10):

    Token/Flash Key,

    Owned by user 3

Key Info on server 1(10.0.3.6):

    Token/Flash Key,

    Owned by user 3
```

Untuk mengonfirmasi bahwa pengguna 4 dapat menggunakan kunci publik (dan semua kunci publik pada HSM), gunakan parameter `-u` dari [findKey](#) di `key_mgmt_util`.

Output menunjukkan bahwa pengguna 4 dapat menggunakan kunci publik (262179) dan privat (262177) dalam pasangan kunci. Pengguna 4 juga dapat menggunakan semua kunci publik lainnya dan kunci privat yang telah mereka buat atau yang telah dibagi dengan mereka.

```
Command: findKey -u 4
```

```
Total number of keys present 8
```

```
number of keys matched from start index 0::7
11, 12, 262159, 262161, 262162, 19, 20, 21, 262177, 262179
```

```
Cluster Error Status
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Example: Dapatkan nilai otentikasi kuorum (`m_value`) untuk kunci

Contoh ini menunjukkan bagaimana cara mendapatkan `m_value` untuk sebuah kunci. The `m_value` adalah jumlah pengguna dalam kuorum yang harus menyetujui operasi kriptografi yang menggunakan kunci dan operasi untuk berbagi atau batal berbagi kunci.

Ketika kuorum autentikasi diaktifkan pada kunci, kuorum pengguna harus menyetujui operasi kriptografi yang menggunakan kunci. Untuk mengaktifkan autentikasi kuorum dan menetapkan ukuran kuorum, gunakan parameter `-m_value` saat Anda membuat kunci.

Perintah ini digunakan [genSymKey](#) untuk membuat kunci AES 256-bit yang dibagikan dengan pengguna 4. Perintah ini menggunakan parameter `m_value` untuk mengaktifkan autentikasi kuorum dan menetapkan ukuran kuorum untuk dua pengguna. Jumlah pengguna harus cukup besar untuk memberikan persetujuan yang diperlukan.

Output menunjukkan bahwa perintah membuat kunci 10.

```
Command: genSymKey -t 31 -s 32 -l aes256m2 -u 4 -m_value 2
```

```
Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS
```

```
Symmetric Key Created. Key Handle: 10
```

```
Cluster Error Status
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Perintah ini menggunakan `getKeyInfo` di `cloudhsm_mgmt_util` untuk mendapatkan informasi tentang pengguna kunci 10. Output menunjukkan bahwa kunci tersebut dimiliki oleh pengguna 3 dan berbagi

dengan pengguna 4. Ini juga menunjukkan bahwa kuorum dua pengguna harus menyetujui setiap operasi kriptografi yang menggunakan kunci.

```
aws-cloudhsm>getKeyInfo 10
```

```
Key Info on server 0(10.0.0.1):
```

```
    Token/Flash Key,
```

```
    Owned by user 3
```

```
    also, shared to following 1 user(s):
```

```
        4
```

```
        2 Users need to approve to use/manage this key
```

```
Key Info on server 1(10.0.0.2):
```

```
    Token/Flash Key,
```

```
    Owned by user 3
```

```
    also, shared to following 1 user(s):
```

```
        4
```

```
        2 Users need to approve to use/manage this key
```

Pendapat

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

```
getKeyInfo -k <key-handle> <output file>
```

<key-handle>

Menentukan handel kunci dari satu kunci di HSM. Masukkan handel kunci dari kunci yang Anda miliki atau bagikan. Parameter ini diperlukan.

Wajib: Ya

<output file>

Menulis output ke file yang ditentukan, selain stdout. Jika file ada, perintah akan menimpa tanpa peringatan.

Wajib: Tidak

Default: stdout

Topik terkait

- [getKeyInfo](#) di key_mgmt_util
- [findKey](#) di key_mgmt_util
- [findAllKeys](#) di cloudhsm_mgmt_util
- [ListUsers](#)
- [ShareKey](#)

Dapatkan informasi untuk setiap HSM dalam sebuah AWS CloudHSM cluster menggunakan CMU

Gunakan info perintah di AWS CloudHSM cloudhsm_mgmt_util (CMU) untuk mendapatkan informasi tentang masing-masing modul keamanan perangkat keras (HSM) di AWS CloudHSM cluster, termasuk nama host, port, alamat IP dan nama dan jenis pengguna yang masuk ke cloudhsm_mgmt_util di HSM.

Sebelum Anda menjalankan perintah CMU, Anda harus memulai CMU dan masuk ke HSM. Pastikan Anda masuk dengan tipe pengguna yang dapat menjalankan perintah yang Anda rencanakan untuk digunakan.

Jika Anda menambah atau menghapus HSMs, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSMs di cluster.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Semua pengguna. Anda tidak perlu masuk untuk menjalankan perintah ini.

Sintaksis

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

```
info server <server ID>
```

Contoh

Contoh ini menggunakan `info` untuk mendapatkan informasi tentang HSM di kluster. Perintah menggunakan `0` untuk merujuk ke HSM pertama di kluster. Output menunjukkan alamat IP, port, serta jenis dan nama pengguna saat ini.

```
aws-cloudhsm> info server 0
```

Id	Name	Hostname	Port	State	Partition
	LoginState				
0	10.0.0.1	10.0.0.1	2225	Connected	hsm-udw0tkfg1ab
	Logged in as 'testuser(CU)'				

Pendapat

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

```
info server <server ID>
```

<server id>

Menentukan ID server dari HSM. HSMs Diberikan nomor urut yang mewakili urutan di mana mereka ditambahkan ke cluster, dimulai dengan 0. Untuk menemukan ID server HSM, gunakan `getHSMInfo`.

Wajib: Ya

Topik terkait

- [mendapatkan HSMInfo](#)
- [LoginHSM dan LogouthSM](#)

Buat daftar atribut AWS CloudHSM kunci menggunakan CMU

Gunakan `listAttributes` perintah di AWS CloudHSM `cloudhsm_mgmt_util` (CMU) untuk mencantumkan atribut kunci dan konstanta yang mewakilinya. AWS CloudHSM Anda menggunakan konstanta ini untuk mengidentifikasi atribut di [getAttribute](#) dan perintah [setAttribute](#).

Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [Masuk](#) ke HSM sebagai pengguna krypto (CU).

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Semua pengguna. Anda tidak perlu masuk untuk menjalankan perintah ini.

Sintaksis

```
listAttributes [-h]
```

Contoh

Perintah ini berisi daftar atribut kunci yang bisa Anda dapatkan dan ubah dalam `key_mgmt_util` dan konstanta yang mewakilinya. Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#). Untuk mewakili semua atribut, gunakan 512.

Command: **listAttributes**

Description

=====

The following are all of the possible attribute values for `getAttribute`.

OBJ_ATTR_CLASS	= 0
OBJ_ATTR_TOKEN	= 1
OBJ_ATTR_PRIVATE	= 2
OBJ_ATTR_LABEL	= 3
OBJ_ATTR_TRUSTED	= 134
OBJ_ATTR_KEY_TYPE	= 256
OBJ_ATTR_ID	= 258
OBJ_ATTR_SENSITIVE	= 259

OBJ_ATTR_ENCRYPT	= 260
OBJ_ATTR_DECRYPT	= 261
OBJ_ATTR_WRAP	= 262
OBJ_ATTR_UNWRAP	= 263
OBJ_ATTR_SIGN	= 264
OBJ_ATTR_VERIFY	= 266
OBJ_ATTR_DERIVE	= 268
OBJ_ATTR_LOCAL	= 355
OBJ_ATTR_MODULUS	= 288
OBJ_ATTR_MODULUS_BITS	= 289
OBJ_ATTR_PUBLIC_EXPONENT	= 290
OBJ_ATTR_VALUE_LEN	= 353
OBJ_ATTR_EXTRACTABLE	= 354
OBJ_ATTR_NEVER_EXTRACTABLE	= 356
OBJ_ATTR_ALWAYS_SENSITIVE	= 357
OBJ_ATTR_DESTROYABLE	= 370
OBJ_ATTR_KCV	= 371
OBJ_ATTR_WRAP_WITH_TRUSTED	= 528
OBJ_ATTR_WRAP_TEMPLATE	= 1073742353
OBJ_ATTR_UNWRAP_TEMPLATE	= 1073742354
OBJ_ATTR_ALL	= 512

Parameter

-h

Menampilkan bantuan untuk perintah.

Wajib: Ya

Topik terkait

- [getAttribute](#)
- [setAttribute](#)
- [Referensi Atribut Utama](#)

Daftar semua AWS CloudHSM pengguna yang menggunakan CMU

Gunakan listUsers perintah di AWS CloudHSM cloudhsm_mgmt_util untuk mendapatkan pengguna di setiap modul keamanan perangkat keras (HSM), bersama dengan jenis pengguna dan atribut

lainnya. Semua jenis pengguna dapat menjalankan perintah ini. Anda bahkan tidak perlu masuk ke `cloudhsm_mgmt_util` untuk menjalankan perintah ini.

Sebelum Anda menjalankan perintah CMU, Anda harus memulai CMU dan masuk ke HSM. Pastikan Anda masuk dengan tipe pengguna yang dapat menjalankan perintah yang ingin Anda gunakan.

Jika Anda menambah atau menghapus HSMs, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSMs di cluster.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Semua pengguna. Anda tidak perlu masuk untuk menjalankan perintah ini.

Sintaksis

Perintah ini tidak memiliki parameter.

```
listUsers
```

Contoh

Perintah ini mencantumkan pengguna pada masing-masing HSMs di cluster dan menampilkan atribut mereka. Anda dapat menggunakan atribut `User ID` untuk mengidentifikasi pengguna dalam perintah lain, seperti `deleteUser`, `changePswd`, dan `findAllKeys`.

```
aws-cloudhsm> listUsers
Users on server 0(10.0.0.1):
Number of users found:6
```

User Id	User Type	User Name	MofnPubKey	
LoginFailureCnt	2FA			
1	PC0	admin	YES	0
N0				
2	AU	app_user	N0	0
N0				
3	CU	crypto_user1	N0	0
N0				
4	CU	crypto_user2	N0	0
N0				

5	C0	officer1	YES	0
NO				
6	C0	officer2	NO	0
NO				
Users on server 1(10.0.0.2):				
Number of users found:5				
User Id	User Type	User Name	MofnPubKey	
LoginFailureCnt	2FA			
1	PC0	admin	YES	0
NO				
2	AU	app_user	NO	0
NO				
3	CU	crypto_user1	NO	0
NO				
4	CU	crypto_user2	NO	0
NO				
5	C0	officer1	YES	0
NO				

Output mencakup atribut pengguna berikut:

- ID Pengguna: Mengidentifikasi pengguna di `key_mgmt_util` dan [cloudhsm_mgmt_util](#).
- [Jenis pengguna](#): Menentukan operasi yang pengguna dapat lakukan pada HSM.
- Nama Pengguna: Menampilkan nama mudah diingat yang ditetapkan pengguna untuk pengguna.
- MofnPubKey: Menunjukkan apakah pengguna telah mendaftarkan key pair untuk menandatangani token [otentikasi kuorum](#).
- LoginFailureCnt: Menunjukkan berapa kali pengguna tidak berhasil masuk.
- 2FA: Menunjukkan bahwa pengguna telah mengaktifkan autentikasi multi-faktor.

Topik terkait

- [listUsers](#) di `key_mgmt_util`
- [CreateUser](#)
- [DeleteUser](#)
- [ChangePSWD](#)

Masuk dan keluar dari HSM menggunakan Utilitas AWS CloudHSM Manajemen

Gunakan `logoutHSM` perintah `loginHSM` and di AWS CloudHSM `cloudhsm_mgmt_util` untuk masuk dan keluar dari setiap HSM dalam sebuah cluster. Setiap pengguna dari jenis apa pun dapat menggunakan perintah ini.

Note

Jika Anda melebihi lima upaya masuk yang salah, akun Anda terkunci. Untuk membuka akun, petugas kriptografi (CO) harus mengatur ulang kata sandi Anda menggunakan perintah [changePswd](#) di `cloudhsm_mgmt_util`.

Untuk memecahkan masalah `loginHSM` dan `logoutHSM`

Sebelum menjalankan perintah `cloudhsm_mgmt_util` ini, Anda harus memulai `cloudhsm_mgmt_util`.

Jika Anda menambah atau menghapus HSMs, perbarui file konfigurasi yang digunakan AWS CloudHSM klien dan alat baris perintah. Jika tidak, perubahan yang Anda buat mungkin tidak efektif pada semua HSMs di cluster.

Jika Anda memiliki lebih dari satu HSM di klaster Anda, Anda mungkin diizinkan untuk upaya login salah tambahan sebelum akun Anda terkunci. Ini karena klien CloudHSM menyeimbangkan beban di berbagai jenis. HSMs Oleh karena itu, upaya login mungkin tidak dimulai pada HSM yang sama setiap saat. Jika Anda menguji fungsionalitas ini, kami sarankan Anda melakukannya pada sebuah klaster dengan hanya satu HSM aktif.

Jika Anda membuat klaster sebelum Februari 2018, akun Anda terkunci setelah 20 upaya login salah.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah ini.

- Petugas pra-crypto (PRECO)
- Petugas krypto (CO)
- Pengguna krypto (CU)

Sintaksis

Masukkan argumen dalam urutan yang ditentukan dalam diagram sintaks. Gunakan parameter `-hpswd` untuk menutupi kata sandi Anda. Untuk login dengan autentikasi dua faktor (2FA), gunakan parameter `-2fa` dan termasuk jalur file. Untuk informasi selengkapnya, lihat [the section called “Pendapat”](#).

```
loginHSM <user-type> <user-name> <password> [-hpswd> [-2fa </path/to/authdata>]
```

```
logoutHSM
```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `loginHSM` dan `logoutHSM` masuk dan keluar dari semua HSMs dalam sebuah cluster.

Example: Masuk ke HSMs dalam cluster

Perintah ini mencatat Anda ke semua HSMs dalam cluster dengan kredensi pengguna CO bernama `admin` dan kata sandi `co12345`. Output menunjukkan bahwa perintah berhasil dan bahwa Anda telah terhubung ke HSMs (yang, dalam hal ini, adalah `server 0` dan `server 1`).

```
aws-cloudhsm>loginHSM CO admin co12345

loginHSM success on server 0(10.0.2.9)
loginHSM success on server 1(10.0.3.11)
```

Example: Masuk dengan kata sandi tersembunyi

Perintah ini sama dengan instans di atas, kecuali kali ini Anda menentukan bahwa sistem harus menyembunyikan kata sandinya.

```
aws-cloudhsm>loginHSM CO admin -hpswd
```

Sistem meminta kata sandi Anda. Anda memasukkan kata sandi, sistem menyembunyikan kata sandi, dan output menunjukkan bahwa perintah berhasil dan bahwa Anda telah terhubung ke HSMs.

```
Enter password:

loginHSM success on server 0(10.0.2.9)
```

```
loginHSM success on server 1(10.0.3.11)
```

```
aws-cloudhsm>
```

Example: Keluar dari HSM

Perintah ini membuat Anda keluar dari HSMs yang saat ini Anda masuki (yang, dalam hal ini, adalah server 0 dan server 1). Output menunjukkan bahwa perintah berhasil dan bahwa Anda telah terputus dari HSMs

```
aws-cloudhsm>logoutHSM
```

```
logoutHSM success on server 0(10.0.2.9)
```

```
logoutHSM success on server 1(10.0.3.11)
```

Pendapat

Masukkan argumen dalam urutan yang ditentukan dalam diagram sintaks. Gunakan parameter `-hpswd` untuk menutupi kata sandi Anda. Untuk login dengan autentikasi dua faktor (2FA), gunakan parameter `-2fa` dan termasuk jalur file. Untuk informasi lebih lanjut tentang bekerja dengan 2FA, lihat [Kelola pengguna 2FA](#)

```
loginHSM <user-type> <user-name> <password> | -hpswd> [-2fa </path/to/authdata>]
```

<user type>

Menentukan jenis pengguna yang masuk ke HSMs Untuk informasi lebih lanjut, lihat [Jenis Pengguna](#) di atas.

Wajib: Ya

<user name>

Menentukan nama pengguna pengguna yang masuk ke HSMs

Wajib: Ya

<password | -hpswd >

Menentukan password dari pengguna yang masuk ke HSMs Untuk menyembunyikan kata sandi, gunakan parameter `-hpswd` di tempat kata sandi dan ikuti petunjuknya.

Wajib: Ya

[-2fa path/to/authdata </ >]

Menentukan bahwa sistem harus menggunakan faktor kedua untuk mengautentikasi pengguna CO dengan 2FA diaktifkan ini. Untuk mendapatkan data yang diperlukan untuk masuk dengan 2FA, sertakan jalur ke lokasi dalam sistem file dengan nama file setelah parameter -2fa. Untuk informasi lebih lanjut tentang bekerja dengan 2FA, lihat [Kelola pengguna 2FA](#).

Wajib: Tidak

Topik terkait

- [Memulai dengan cloudhsm_mgmt_util](#)
- [Aktifkan Cluster](#)

Kaitkan AWS CloudHSM pengguna dengan kunci menggunakan CMU

Gunakan registerQuorumPubKey perintah di AWS CloudHSM cloudhsm_mgmt_util untuk mengaitkan pengguna modul keamanan perangkat keras (HSM) dengan pasangan kunci RSA-2048 asimetris. Setelah Anda mengaitkan pengguna HSM dengan kunci, pengguna tersebut dapat menggunakan kunci privat untuk menyetujui permintaan kuorum dan klaster dapat menggunakan kunci publik terdaftar untuk memverifikasi tanda tangan dari pengguna. Untuk informasi lebih lanjut tentang autentikasi kuorum, lihat [Mengelola Otentikasi Kuorum \(M dari N Kontrol Akses\)](#).

Tip

Dalam AWS CloudHSM dokumentasi, otentikasi kuorum kadang-kadang disebut sebagai M dari N (MoFN), yang berarti minimum M approvers dari jumlah total N approvers.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Petugas krito (CO)

Sintaksis

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.


```
registerQuorumPubKey <user-type> <user-name> <registration-token> <signed-registration-token> <public-key>
```

Contoh

Contoh ini menunjukkan cara menggunakan `registerQuorumPubKey` untuk mendaftarkan petugas kriptografi (CO) sebagai pemberi persetujuan pada permintaan autentikasi kuorum. Untuk menjalankan perintah ini, Anda harus memiliki pasangan kunci RSA-2048 asimetris, token yang ditandatangani, dan token yang tidak ditandatangani. Untuk informasi lebih lanjut tentang persyaratan ini, lihat [the section called “Pendapat”](#).

Example: Daftarkan pengguna HSM untuk otentikasi kuorum

Contoh ini mendaftarkan CO bernama `quorum_officer` sebagai pemberi persetujuan untuk autentikasi kuorum.

```
aws-cloudhsm> registerQuorumPubKey CO <quorum_officer> </path/to/quorum_officer.token> </path/to/quorum_officer.token.sig> </path/to/quorum_officer.pub>
```

```
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****
```

```
Do you want to continue(y/n)?y
registerQuorumPubKey success on server 0(10.0.0.1)
```

Perintah terakhir menggunakan perintah [ListUsers](#) untuk memverifikasi bahwa `quorum_officer` terdaftar sebagai pengguna MofN.

```
aws-cloudhsm> listUsers
Users on server 0(10.0.0.1):
Number of users found:3
```

User Id	User Type	User Name	MofnPubKey
LoginFailureCnt	2FA		
1	PCO	admin	N0
0	N0		

2	AU	app_user	NO
0	NO		
3	CO	quorum_officer	YES
0	NO		

Pendapat

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

```
registerQuorumPubKey <user-type> <user-name> <registration-token> <signed-registration-token> <public-key>
```

<user-type>

Menentukan jenis pengguna. Parameter ini diperlukan.

Untuk informasi detail tentang jenis pengguna pada HSM, lihat [Jenis pengguna HSM untuk Utilitas AWS CloudHSM Manajemen](#).

Nilai yang valid:

- CO: Petugas krypto dapat mengelola pengguna, tetapi mereka tidak dapat mengelola kunci.

Wajib: Ya

<user name>

Menentukan nama yang mudah diingat untuk pengguna. Panjang maksimum adalah 31 karakter. Satu-satunya karakter khusus yang diizinkan adalah garis bawah (_).

Anda tidak dapat mengubah nama pengguna setelah dibuat. Dalam perintah `cloudhsm_mgmt_util`, jenis pengguna dan kata sandi peka huruf besar-kecil, tetapi nama pengguna tidak.

Wajib: Ya

<registration-token>

Menentukan jalur ke file yang berisi token pendaftaran tanpa tanda tangan. Dapat memiliki data acak dengan ukuran file maks 245 byte. Untuk informasi lebih lanjut tentang cara membuat token pendaftaran tanpa tanda tangan, lihat [Buat dan Tandatangani Token Pendaftaran](#).

Wajib: Ya

<signed-registration-token>

Menentukan path ke file yang berisi mekanisme SHA256 _PKCS hash ditandatangani dari registration-token. Untuk informasi lebih lanjut, lihat [Buat dan Tandatangani Token Pendaftaran](#).

Wajib: Ya

<public-key>

Menentukan jalur ke file yang berisi kunci publik dari pasangan kunci RSA-2048 asimetris. Gunakan kunci privat untuk menandatangani token pendaftaran. Untuk informasi lebih lanjut, lihat [Buat Pasangan Kunci RSA](#).

Wajib: Ya

Note

Klaster menggunakan kunci yang sama untuk autentikasi kuorum dan autentikasi dua faktor (2FA). Ini berarti Anda tidak dapat memutar kunci kuorum untuk pengguna yang mengaktifkan 2FA menggunakan registerQuorumPubKey. Untuk memutar kunci, Anda harus menggunakan changePswd. Untuk informasi lebih lanjut tentang penggunaan autentikasi kuorum dan 2FA, lihat [Autentikasi Kuorum dan 2FA](#).

Topik terkait

- [Buat Pasangan Kunci RSA](#)
- [Membuat dan Menandatangani Token Pendaftaran](#)
- [Daftarkan Kunci Publik dengan HSM](#)
- [Mengelola Otentikasi Kuorum \(M of N Access Control\)](#)
- [Otentikasi Kuorum dan 2FA](#)
- [ListUsers](#)

Berinteraksi dengan satu HSM dalam sebuah AWS CloudHSM cluster menggunakan CMU

Gunakan server perintah di AWS CloudHSM cloudhsm_mgmt_util untuk masuk ke mode server dan berinteraksi langsung dengan instance modul keamanan perangkat keras (HSM) tertentu.

Biasanya, ketika Anda mengeluarkan perintah di `cloudhsm_mgmt_util`, perintah akan mempengaruhi semua HSMs dalam cluster yang ditunjuk (mode global). Namun, mungkin ada keadaan yang Anda butuhkan untuk mengeluarkan perintah ke satu HSM. Misalnya, jika sinkronisasi otomatis gagal, Anda mungkin perlu untuk menyinkronkan kunci dan pengguna pada HSM untuk mempertahankan konsistensi di klaster.

Setelah inisiasi berhasil, prompt perintah `aws-cloudhsm` > diganti dengan prompt perintah `server` >.

Untuk keluar dari mode server, gunakan perintah `exit`. Setelah berhasil keluar, Anda akan dikembalikan ke prompt perintah `cloudhsm_mgmt_util`.

Sebelum menjalankan perintah `cloudhsm_mgmt_util`, Anda harus memulai `cloudhsm_mgmt_util`.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Semua pengguna.

Prasyarat

Untuk masuk ke mode server, Anda harus terlebih dahulu mengetahui nomor server dari target HSM. Nomor server tercantum dalam jejak output yang dihasilkan oleh `cloudhsm_mgmt_util` setelah inisiasi. Nomor server ditetapkan dalam urutan yang sama dengan yang HSMs muncul dalam file konfigurasi. Untuk contoh ini, kami berasumsi bahwa `server 0` adalah server yang sesuai dengan HSM yang diinginkan.

Sintaksis

Untuk memulai mode server:

```
server <server-number>
```

Untuk keluar dari mode server:

```
server> exit
```

Contoh

Perintah ini memasuki modus server pada HSM dengan nomor server 0.

```
aws-cloudhsm> server 0
```

```
Server is in 'E2' mode...
```

Untuk keluar dari mode server, gunakan perintah `exit`.

```
server0> exit
```

Pendapat

```
server <server-number>
```

<server-number>

Menentukan jumlah server dari target HSM.

Wajib: Ya

Tidak ada argumen untuk perintah `exit`.

Topik terkait

- [SyncKey](#)
- [CreateUser](#)
- [DeleteUser](#)

Mengatur atribut AWS CloudHSM kunci menggunakan CMU

Gunakan `setAttribute` perintah di AWS CloudHSM `cloudhsm_mgmt_util` untuk mengubah nilai label, mengenkripsi, mendekripsi, membungkus, dan membuka atribut kunci di HSMs Anda juga dapat menggunakan [setAttribute](#) di `key_mgmt_util` untuk mengubah kunci sesi ke kunci persisten. Anda hanya dapat mengubah atribut kunci yang Anda miliki.

Sebelum Anda menjalankan perintah CMU, Anda harus memulai CMU dan masuk ke HSM. Pastikan Anda masuk dengan tipe pengguna yang dapat menjalankan perintah yang ingin Anda gunakan.

Jika Anda menambah atau menghapus HSMs, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSMs di cluster.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Pengguna krypto (CU)

Sintaksis

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

```
setAttribute <key handle> <attribute id>
```

Contoh

Contoh ini menunjukkan bagaimana cara menonaktifkan fungsionalitas dekripsi dari kunci simetris. Anda dapat menggunakan perintah seperti ini untuk mengatur konfigurasi kunci pembungkus, yang harus dapat membungkus dan membuka kunci lain, tetapi tidak mengenkripsi atau mendekripsi data.

Langkah pertama adalah membuat kunci pembungkus. Perintah ini menggunakan [genSymKey](#)key_mgmt_util untuk menghasilkan kunci simetris AES 256-bit. Output menunjukkan bahwa kunci baru memiliki handel kunci 14.

```
$ genSymKey -t 31 -s 32 -l aes256
```

```
Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS
```

```
Symmetric Key Created. Key Handle: 14
```

```
Cluster Error Status
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Selanjutnya, kita ingin mengonfirmasi nilai saat ini dari atribut dekripsi. Untuk mendapatkan ID atribut dari atribut dekripsi, gunakan [listAttributes](#). Output menunjukkan bahwa konstanta yang mewakili atribut OBJ_ATTR_DECRYPT adalah 261. Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

```
aws-cloudhsm> listAttributes
```

Following are the possible attribute values for `getAttribute`:

<code>OBJ_ATTR_CLASS</code>	<code>= 0</code>
<code>OBJ_ATTR_TOKEN</code>	<code>= 1</code>
<code>OBJ_ATTR_PRIVATE</code>	<code>= 2</code>
<code>OBJ_ATTR_LABEL</code>	<code>= 3</code>
<code>OBJ_ATTR_TRUSTED</code>	<code>= 134</code>
<code>OBJ_ATTR_KEY_TYPE</code>	<code>= 256</code>
<code>OBJ_ATTR_ID</code>	<code>= 258</code>
<code>OBJ_ATTR_SENSITIVE</code>	<code>= 259</code>
<code>OBJ_ATTR_ENCRYPT</code>	<code>= 260</code>
<code>OBJ_ATTR_DECRYPT</code>	<code>= 261</code>
<code>OBJ_ATTR_WRAP</code>	<code>= 262</code>
<code>OBJ_ATTR_UNWRAP</code>	<code>= 263</code>
<code>OBJ_ATTR_SIGN</code>	<code>= 264</code>
<code>OBJ_ATTR_VERIFY</code>	<code>= 266</code>
<code>OBJ_ATTR_DERIVE</code>	<code>= 268</code>
<code>OBJ_ATTR_LOCAL</code>	<code>= 355</code>
<code>OBJ_ATTR_MODULUS</code>	<code>= 288</code>
<code>OBJ_ATTR_MODULUS_BITS</code>	<code>= 289</code>
<code>OBJ_ATTR_PUBLIC_EXPONENT</code>	<code>= 290</code>
<code>OBJ_ATTR_VALUE_LEN</code>	<code>= 353</code>
<code>OBJ_ATTR_EXTRACTABLE</code>	<code>= 354</code>
<code>OBJ_ATTR_NEVER_EXTRACTABLE</code>	<code>= 356</code>
<code>OBJ_ATTR_ALWAYS_SENSITIVE</code>	<code>= 357</code>
<code>OBJ_ATTR_DESTROYABLE</code>	<code>= 370</code>
<code>OBJ_ATTR_KCV</code>	<code>= 371</code>
<code>OBJ_ATTR_WRAP_WITH_TRUSTED</code>	<code>= 528</code>
<code>OBJ_ATTR_WRAP_TEMPLATE</code>	<code>= 1073742353</code>
<code>OBJ_ATTR_UNWRAP_TEMPLATE</code>	<code>= 1073742354</code>
<code>OBJ_ATTR_ALL</code>	<code>= 512</code>

Untuk mendapatkan nilai saat ini dari atribut dekripsi untuk kunci 14, perintah berikutnya menggunakan [getAttribute](#) di `cloudhsm_mgmt_util`.

Output menunjukkan bahwa nilai atribut dekripsi adalah `true (1)` pada keduanya HSMs di cluster.

```
aws-cloudhsm> getAttribute 14 261
```

```
Attribute Value on server 0(10.0.0.1):
```

```
OBJ_ATTR_DECRYPT
```

```
0x00000001
```

```
Attribute Value on server 1(10.0.0.2):
OBJ_ATTR_DECRYPT
0x00000001
```

Perintah ini menggunakan `setAttribute` untuk mengubah nilai atribut dekripsi (atribut 261) dari kunci 14 menjadi 0. Ini menonaktifkan fungsionalitas dekripsi pada kunci.

Output menunjukkan bahwa perintah berhasil pada keduanya HSMs di cluster.

```
aws-cloudhsm> setAttribute 14 261 0
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****

Do you want to continue(y/n)? y
setAttribute success on server 0(10.0.0.1)
setAttribute success on server 1(10.0.0.2)
```

Perintah terakhir mengulangi perintah `getAttribute`. Sekali lagi, perintah ini mendapatkan atribut dekripsi (atribut 261) dari kunci 14.

Kali ini, output menunjukkan bahwa nilai atribut dekripsi adalah false (0) pada keduanya HSMs di cluster.

```
aws-cloudhsm > getAttribute 14 261
Attribute Value on server 0(10.0.3.6):
OBJ_ATTR_DECRYPT
0x00000000

Attribute Value on server 1(10.0.1.7):
OBJ_ATTR_DECRYPT
0x00000000
```

Pendapat

```
setAttribute <key handle> <attribute idb
```


<key-handle>

Menentukan handel kunci dari kunci yang Anda miliki. Anda hanya dapat menentukan satu kunci di setiap perintah. Untuk mendapatkan handel kunci dari suatu kunci, gunakan [findKey](#) di `key_mgmt_util`. Untuk menemukan pengguna kunci, gunakan [getKeyInfo](#).

Wajib: Ya

<attribute id>

Menentukan konstanta yang mewakili atribut yang ingin Anda ubah. Anda hanya dapat menentukan satu atribut di setiap perintah. Untuk mendapatkan atribut dan nilai integernya, gunakan [listAttributes](#). Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

Nilai yang valid:

- 3 – OBJ_ATTR_LABEL.
- 134 – OBJ_ATTR_TRUSTED.
- 260 – OBJ_ATTR_ENCRYPT.
- 261 – OBJ_ATTR_DECRYPT.
- 262 – OBJ_ATTR_WRAP.
- 263 – OBJ_ATTR_UNWRAP.
- 264 – OBJ_ATTR_SIGN.
- 266 – OBJ_ATTR_VERIFY.
- 268 – OBJ_ATTR_DERIVE.
- 370 – OBJ_ATTR_DESTROYABLE.
- 528 – OBJ_ATTR_WRAP_WITH_TRUSTED.
- 1073742353 – OBJ_ATTR_WRAP_TEMPLATE.
- 1073742354 – OBJ_ATTR_UNWRAP_TEMPLATE.

Wajib: Ya

Topik terkait

- [setAttribute](#) di `key_mgmt_util`

- [getAttribute](#)
- [ListAttributes](#)
- [Referensi Atribut Utama](#)

Keluar dari CMU

Gunakan quit perintah di cloudhsm_mgmt_util untuk AWS CloudHSM keluar dari cloudhsm_mgmt_util. Setiap pengguna dari jenis apa pun dapat menggunakan perintah ini.

Sebelum menjalankan perintah cloudhsm_mgmt_util, Anda harus memulai cloudhsm_mgmt_util.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Semua pengguna. Anda tidak perlu masuk untuk menjalankan perintah ini.

Sintaksis

```
quit
```

Contoh

Perintah ini mengeluarkan dari cloudhsm_mgmt_util. Setelah berhasil menyelesaikan, Anda akan dikembalikan ke baris perintah biasa Anda. Perintah ini tidak memiliki parameter output.

```
aws-cloudhsm> quit

disconnecting from servers, please wait...
```

Topik terkait

- [Memulai dengan cloudhsm_mgmt_util](#)

Bagikan AWS CloudHSM kunci menggunakan CMU

Gunakan shareKey perintah di AWS CloudHSM cloudhsm_mgmt_util untuk berbagi dan membatalkan berbagi kunci yang Anda miliki dengan pengguna krypto lainnya. Hanya pemilik

kunci yang dapat berbagi dan membatalkan berbagi kunci. Anda juga dapat berbagi kunci saat membuatnya.

Pengguna yang berbagi kunci dapat menggunakan kunci dalam operasi kriptografi, tetapi mereka tidak dapat menghapus, mengeksport, berbagi, atau membatalkan berbagi kunci, atau mengubah atributnya. Ketika kuorum autentikasi diaktifkan pada kunci, kuorum harus menyetujui setiap operasi yang membagikan atau batal membagikan kunci.

Sebelum Anda menjalankan perintah CMU, Anda harus memulai CMU dan masuk ke HSM. Pastikan Anda masuk dengan tipe pengguna yang dapat menjalankan perintah yang Anda rencanakan untuk digunakan.

Jika Anda menambah atau menghapus HSMs, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSMs di cluster.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna kriptografi (CU)

Sintaksis

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

Jenis Pengguna: Pengguna Crypto (CU)

```
shareKey <key handle> <user id> <(share/unshare key?) 1/0>
```

Contoh

Contoh berikut menunjukkan cara menggunakan shareKey untuk berbagi dan membatalkan berbagi kunci yang Anda miliki dengan pengguna kriptografi lainnya.

Example: Bagikan kunci

Contoh ini digunakan shareKey untuk membagikan [kunci pribadi ECC](#) yang dimiliki pengguna saat ini dengan pengguna kriptografi lain di. HSMs Kunci publik tersedia untuk semua pengguna HSM, sehingga Anda tidak dapat membagi atau batal membagikannya.

Perintah pertama digunakan [getKeyInfo](#) untuk mendapatkan informasi pengguna untuk kunci 262177, kunci pribadi ECC pada HSMs.

Output menunjukkan bahwa kunci 262177 dimiliki oleh pengguna 3, tetapi tidak dibagi.

```
aws-cloudhsm>getKeyInfo 262177

Key Info on server 0(10.0.3.10):

    Token/Flash Key,

    Owned by user 3

Key Info on server 1(10.0.3.6):

    Token/Flash Key,

    Owned by user 3
```

Perintah ini digunakan `shareKey` untuk berbagi kunci 262177 dengan pengguna 4, pengguna kriptografi lain di file HSMs. Argumen terakhir menggunakan nilai 1 untuk menunjukkan operasi berbagi.

Output menunjukkan bahwa operasi berhasil pada keduanya HSMs di cluster.

```
aws-cloudhsm>shareKey 262177 4 1
*****CAUTION*****
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****

Do you want to continue(y/n)?y
shareKey success on server 0(10.0.3.10)
shareKey success on server 1(10.0.3.6)
```

Untuk memverifikasi bahwa operasi berhasil, instans mengulangi perintah `getKeyInfo` yang pertama.

Output menunjukkan bahwa kunci 262177 sekarang dibagi dengan pengguna 4.

```
aws-cloudhsm>getKeyInfo 262177
```

Key Info on server 0(10.0.3.10):

Token/Flash Key,

Owned by user 3

also, shared to following 1 user(s):

4

Key Info on server 1(10.0.3.6):

Token/Flash Key,

Owned by user 3

also, shared to following 1 user(s):

4

Example: Batalkan pembagian kunci

Contoh ini batal membagikan kunci simetris, yaitu menghapus pengguna krypto dari daftar pengguna bersama untuk kunci tersebut.

Perintah ini menggunakan `shareKey` untuk menghapus pengguna 4 dari daftar pengguna bersama untuk kunci 6. Argumen terakhir menggunakan nilai 0 untuk menunjukkan operasi batal berbagi.

Output menunjukkan bahwa perintah berhasil pada keduanya HSMs. Akibatnya, pengguna 4 tidak dapat lagi menggunakan kunci 6 dalam operasi kriptografi.

```
aws-cloudhsm>shareKey 6 4 0
```

```
*****CAUTION*****
```

```
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
```

```
*****
```

```
Do you want to continue(y/n)?y
```

```
shareKey success on server 0(10.0.3.10)
```

```
shareKey success on server 1(10.0.3.6)
```

Pendapat

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

```
shareKey <key handle> <user id> <(share/unshare key?) 1/0>
```

<key-handle>

Menentukan handel kunci dari kunci yang Anda miliki. Anda hanya dapat menentukan satu kunci di setiap perintah. Untuk mendapatkan handel kunci dari suatu kunci, gunakan [findKey](#) di `key_mgmt_util`. Untuk memverifikasi bahwa Anda memiliki kunci, gunakan [getKeyInfo](#).

Wajib: Ya

<user id>

Menentukan ID pengguna dari pengguna krypto (CU) dengan siapa Anda berbagi atau tidak berbagi kunci. Untuk menemukan ID pengguna dari seorang pengguna, gunakan [listUsers](#).

Wajib: Ya

<share 1 or unshare 0>

Untuk berbagi kunci dengan pengguna tertentu, ketik 1. Untuk membatalkan berbagi kunci, yaitu, untuk menghapus pengguna tertentu dari daftar pengguna bersama untuk kunci, ketik 0.

Wajib: Ya

Topik terkait

- [getKeyInfo](#)

Sinkronisasi kunci di seluruh AWS CloudHSM cluster menggunakan CMU

Gunakan `syncKey` perintah di AWS CloudHSM `cloudhsm_mgmt_util` untuk menyinkronkan kunci secara manual di seluruh instance HSM dalam cluster atau di seluruh kloning kloning. Secara umum, Anda tidak perlu menggunakan perintah ini, karena instans HSM dalam klaster menyinkronkan kunci secara otomatis. Namun, sinkronisasi kunci di klaster kloning harus dilakukan secara manual. Cluster kloning biasanya dibuat di AWS Wilayah yang berbeda untuk menyederhanakan proses penskalaan global dan pemulihan bencana.

Anda tidak dapat menggunakan `syncKey` untuk menyinkronkan kunci di kluster arbitrer: salah satu kluster harus telah dibuat dari cadangan yang lain. Selain itu, kedua kluster harus konsisten dengan kredensial CO dan CU agar operasi menjadi sukses. Untuk informasi lebih lanjut, lihat [Pengguna HSM](#).

Untuk menggunakannya `syncKey`, Anda harus terlebih dahulu [membuat file AWS CloudHSM konfigurasi](#) yang menentukan satu HSM dari cluster sumber dan satu dari cluster tujuan. Ini akan memungkinkan `cloudhsm_mgmt_util` untuk terhubung ke kedua instans HSM. Gunakan file konfigurasi ini untuk memulai `cloudhsm_mgmt_util`. Kemudian masuk dengan kredensial CO atau CU yang memiliki kunci yang ingin Anda sinkronkan.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Petugas krito (CO)
- Pengguna krito (CU)

Note

COs dapat digunakan `syncKey` pada tombol apa saja, sementara hanya CUs dapat menggunakan perintah ini pada tombol yang mereka miliki. Untuk informasi selengkapnya, lihat [the section called “Jenis pengguna”](#).

Prasyarat

Sebelum memulai, Anda harus mengetahui `key handle` kunci pada HSM sumber yang akan disinkronkan dengan HSM tujuan. Untuk menemukan `key handle`, gunakan perintah [listUsers](#) untuk mendaftar semua pengenalan untuk pengguna yang bernama. Kemudian, gunakan [findAllKeys](#) perintah untuk menemukan semua kunci milik pengguna tertentu.

Anda juga perlu mengetahui yang `server IDs` ditetapkan ke sumber dan tujuan HSMs, yang ditampilkan dalam keluaran jejak yang dikembalikan oleh `cloudhsm_mgmt_util` saat inisiasi. Ini ditetapkan dalam urutan yang sama dengan yang HSMs muncul di file konfigurasi.

Ikuti instruksi di [Menggunakan CMU di Kluster Kloning](#) dan menginisialisasi `cloudhsm_mgmt_util` dengan file config baru. Kemudian, masukkan mode server pada HSM sumber dengan mengeluarkan perintah [server](#).

Sintaksis

Note

Untuk menjalankan `syncKey`, pertama masukkan mode server pada HSM yang berisi kunci yang akan disinkronkan.

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

Jenis Pengguna: Pengguna Crypto (CU)

```
syncKey <key handle> <destination hsm>
```

Contoh

Jalankan perintah server untuk masuk ke HSM sumber dan masuk ke mode server. Untuk contoh ini, kami berasumsi bahwa `server 0` adalah HSM sumber.

```
aws-cloudhsm> server 0
```

Sekarang jalankan perintah `syncKey`. Dalam contoh ini, kami menganggap kunci 261251 akan disinkronkan ke `server 1`.

```
aws-cloudhsm> syncKey 261251 1  
syncKey success
```

Pendapat

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

```
syncKey <key handle> <destination hsm>
```

<key handle>

Menentukan handel kunci dari kunci yang akan disinkronkan. Anda hanya dapat menentukan satu kunci di setiap perintah. Untuk mendapatkan pegangan kunci kunci, gunakan [findAllKeys](#) saat masuk ke server HSM.

Wajib: Ya

<destination hsm>

Menentukan jumlah server yang akan Anda sinkronkan dengan kunci.

Wajib: Ya

Topik terkait

- [ListUsers](#)
- [findAllKeys](#)
- [jelaskan-cluster](#) di AWS CLI
- [server](#)

Sinkronisasi pengguna di seluruh AWS CloudHSM cluster menggunakan CMU

Gunakan `syncUser` perintah di AWS CloudHSM `cloudhsm_mgmt_util` untuk menyinkronkan pengguna kriptografi () atau petugas kriptografi (CUs) secara manual di seluruh instance HSM dalam kluster atau di seluruh kloning COs kloning. AWS CloudHSM tidak secara otomatis menyinkronkan pengguna. Umumnya, Anda mengelola pengguna dalam mode global sehingga semua HSMs dalam cluster diperbarui bersama-sama. Anda mungkin harus menggunakan `syncUser` jika HSM sengaja membatalkan sinkronisasi (misalnya, karena perubahan kata sandi) atau jika Anda ingin memutar kredensial pengguna di kluster kloning. Cluster kloning biasanya dibuat di AWS Wilayah yang berbeda untuk menyederhanakan proses penskalaan global dan pemulihan bencana.

Sebelum Anda menjalankan perintah CMU, Anda harus memulai CMU dan masuk ke HSM. Pastikan Anda masuk dengan tipe pengguna yang dapat menjalankan perintah yang Anda rencanakan untuk digunakan.

Jika Anda menambah atau menghapus HSMs, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSMs di cluster.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Petugas kriptografi (CO)

Prasyarat

Sebelum memulai, Anda harus mengetahui `user ID` pengguna pada HSM sumber yang akan disinkronkan dengan HSM tujuan. Untuk menemukan `user ID`, gunakan perintah [ListUsers](#) untuk mencantumkan semua pengguna di HSMs dalam klaster.

Anda juga perlu mengetahui yang `server ID` ditetapkan ke sumber dan tujuan HSMs, yang ditampilkan dalam keluaran jejak yang dikembalikan oleh `cloudhsm_mgmt_util` saat inisiasi. Ini ditetapkan dalam urutan yang sama dengan yang HSMs muncul di file konfigurasi.

Jika Anda menyinkronkan HSMs lintas kloning, ikuti petunjuk di [Menggunakan CMU Across Cloned Clusters dan inisialisasi cloudhsm_mgmt_util](#) dengan file konfigurasi baru.

Ketika Anda siap untuk menjalankan `syncUser`, masuk ke mode server pada HSM sumber dengan mengeluarkan perintah [server](#).

Sintaksis

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

```
syncUser <user ID> <server ID>
```

Contoh

Jalankan perintah `server` untuk masuk ke HSM sumber dan masuk ke mode server. Untuk contoh ini, kami berasumsi bahwa `server 0` adalah HSM sumber.

```
aws-cloudhsm> server 0
```

Sekarang jalankan perintah `syncUser`. Untuk contoh ini, kami berasumsi bahwa pengguna 6 adalah pengguna yang akan disinkronkan, dan `server 1` adalah HSM tujuan.

```
server 0> syncUser 6 1
ExtractMaskedObject: 0x0 !
InsertMaskedObject: 0x0 !
syncUser success
```

Pendapat

Karena perintah ini tidak memiliki parameter bernama, Anda harus memasukkan argumen dalam urutan yang ditentukan dalam diagram sintaks.

```
syncUser <user ID> <server ID>
```

<user ID>

Menentukan ID pengguna untuk disinkronkan. Anda hanya dapat menentukan satu pengguna di setiap perintah. Untuk mendapatkan ID pengguna, gunakan [listUsers](#).

Wajib: Ya

<server ID>

Menentukan nomor server HSM yang akan Anda sinkronkan dengan pengguna.

Wajib: Ya

Topik terkait

- [ListUsers](#)
- [jelaskan-cluster](#) di AWS CLI
- [server](#)

AWS CloudHSM Utilitas Manajemen Kunci (KMU)

Utilitas manajemen kunci (KMU) adalah alat baris perintah untuk AWS CloudHSM membantu pengguna kriptografi (CU) mengelola kunci pada modul keamanan perangkat keras (HSM). KMU mencakup beberapa perintah yang menghasilkan, menghapus, mengimpor, dan mengekspor kunci, mendapatkan dan mengatur atribut, menemukan kunci, dan melakukan operasi kriptografi.

KMU dan CMU adalah bagian dari [Client SDK 3](#) suite.

Untuk memulai dengan cepat, lihat [Memulai dengan AWS CloudHSM key_mgmt_util](#). Untuk informasi detail mengenai perintah, lihat [Referensi untuk perintah AWS CloudHSM Key Management Utility](#). Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

Untuk menggunakan `key_mgmt_util`, jika Anda menggunakan Linux, hubungkan ke instans klien Anda dan kemudian lihat [Instal dan konfigurasi AWS CloudHSM klien untuk KMU \(Linux\)](#). Jika Anda menggunakan Windows, lihat [Instal dan konfigurasi AWS CloudHSM klien untuk KMU \(Windows\)](#).

Topik

- [Memulai dengan AWS CloudHSM `key\_mgmt\_util`](#)
- [Instal dan konfigurasi AWS CloudHSM klien untuk KMU \(Linux\)](#)
- [Instal dan konfigurasi AWS CloudHSM klien untuk KMU \(Windows\)](#)
- [Referensi untuk perintah AWS CloudHSM Key Management Utility](#)

Memulai dengan AWS CloudHSM `key_mgmt_util`

AWS CloudHSM mencakup dua alat baris perintah dengan perangkat [lunak AWS CloudHSM klien](#). Alat [cloudhsm_mgmt_util](#) mencakup perintah untuk mengelola pengguna HSM. Alat [key_mgmt_util](#) mencakup perintah untuk mengelola kunci. Untuk memulai dengan alat baris perintah `key_mgmt_util`, lihat topik berikut.

Topik

- [Mengatur AWS CloudHSM `key\_mgmt\_util`](#)
- [Masuk ke HSMs dalam AWS CloudHSM cluster menggunakan KMU](#)
- [Keluar dari HSMs dalam AWS CloudHSM cluster menggunakan KMU](#)
- [Hentikan AWS CloudHSM `key\_mgmt\_util`](#)

Jika Anda mendapatkan pesan kesalahan atau hasil tak terduga untuk perintah, lihat topik [Pemecahan masalah AWS CloudHSM](#) untuk bantuan. Untuk detail tentang perintah `key_mgmt_util`, lihat [Referensi untuk perintah AWS CloudHSM Key Management Utility](#).

Mengatur AWS CloudHSM `key_mgmt_util`

Selesaikan pengaturan berikut sebelum Anda menggunakan AWS CloudHSM `key_mgmt_util` (KMU).

Topik

- [Langkah 1. Mulai AWS CloudHSM klien](#)
- [Langkah 2. Mulai `key\_mgmt\_util`](#)

Langkah 1. Mulai AWS CloudHSM klien

Sebelum Anda menggunakan `key_mgmt_util`, Anda harus memulai klien. AWS CloudHSM Klien adalah daemon yang membangun komunikasi end-to-end terenkripsi dengan di cluster Anda. HSMs Alat `key_mgmt_util` menggunakan koneksi klien untuk berkomunikasi dengan di cluster Anda. HSMs Tanpa itu, `key_mgmt_util` tidak bekerja.

Untuk memulai AWS CloudHSM klien

Gunakan perintah berikut untuk memulai AWS CloudHSM klien.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Untuk klien Windows 1.1.2+:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Untuk klien Windows 1.1.1 dan yang lebih lama:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Langkah 2. Mulai key_mgmt_util

Setelah Anda memulai AWS CloudHSM klien, gunakan perintah berikut untuk memulai key_mgmt_util.

Amazon Linux

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

Amazon Linux 2

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

CentOS 7

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

CentOS 8

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

RHEL 7

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

RHEL 8

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

Ubuntu 16.04 LTS

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

Ubuntu 18.04 LTS

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\key_mgmt_util.exe"
```

Prompt berubah jadi Command: ketika key_mgmt_util berjalan.

Jika perintah gagal, seperti mengembalikan pesan Daemon socket connection error, coba [memperbarui file konfigurasi](#).

Masuk ke HSMs dalam AWS CloudHSM cluster menggunakan KMU

Gunakan loginHSM perintah di key_mgmt_util (KMU) untuk masuk ke modul keamanan perangkat keras (HSM) dalam sebuah cluster. AWS CloudHSM Perintah berikut masuk sebagai [pengguna krypto \(CU\)](#) bernama example_user. Output menunjukkan login yang berhasil untuk ketiganya HSMs di cluster.

```
Command: loginHSM -u CU -s example_user -p <PASSWORD>  
Cfm3LoginHSM returned: 0x00 : HSM Return: SUCCESS
```

Cluster Error Status

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS  
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS  
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Berikut ini adalah sintaks untuk perintah loginHSM.

```
Command: loginHSM -u <USER TYPE> -s <USERNAME> -p <PASSWORD>
```

Keluar dari HSMs dalam AWS CloudHSM cluster menggunakan KMU

Gunakan `logoutHSM` perintah di `key_mgmt_util` (KMU) untuk keluar dari modul keamanan perangkat keras (HSM) dalam sebuah cluster. AWS CloudHSM

```
Command: logoutHSM
Cfm3LogoutHSM returned: 0x00 : HSM Return: SUCCESS
```

```
Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Hentikan AWS CloudHSM `key_mgmt_util`

Gunakan `exit` perintah untuk menghentikan AWS CloudHSM `key_mgmt_util`.

```
Command: exit
```

Instal dan konfigurasikan AWS CloudHSM klien untuk KMU (Linux)

Untuk berinteraksi dengan modul keamanan perangkat keras (HSM) di AWS CloudHSM cluster Anda menggunakan `key_mgmt_util` (KMU), Anda memerlukan perangkat lunak klien untuk Linux. AWS CloudHSM Anda harus menginstalnya pada instans klien Linux EC2 yang Anda buat sebelumnya. Anda juga dapat menginstal klien jika Anda menggunakan Windows. Untuk informasi selengkapnya, lihat [Instal dan konfigurasikan AWS CloudHSM klien untuk KMU \(Windows\)](#).

Tugas

- [Langkah 1. Instal AWS CloudHSM klien dan alat baris perintah](#)
- [Langkah 2. Edit konfigurasi klien](#)

Langkah 1. Instal AWS CloudHSM klien dan alat baris perintah

Connect ke instance klien Anda dan jalankan perintah berikut untuk men-download dan menginstal AWS CloudHSM client dan command line tools.

Amazon Linux

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-latest.el6.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm
```

RHEL 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

RHEL 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client_latest_amd64.deb
```

```
sudo apt install ./cloudhsm-client_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client_latest_u18.04_amd64.deb
```

```
sudo apt install ./cloudhsm-client_latest_u18.04_amd64.deb
```

Langkah 2. Edit konfigurasi klien

Sebelum Anda dapat menggunakan AWS CloudHSM klien untuk terhubung ke cluster Anda, Anda harus mengedit konfigurasi klien.

Untuk mengedit konfigurasi klien

1. Salin sertifikat penerbitan Anda—[salah satu yang Anda gunakan untuk menandatangani sertifikat klaster](#)—ke lokasi berikut pada instans klien: `/opt/cloudhsm/etc/customerCA.crt`. Anda perlu izin pengguna root instans pada instans klien untuk menyalin sertifikat Anda ke lokasi ini.
2. Gunakan perintah [konfigurasi](#) berikut untuk memperbarui file konfigurasi untuk AWS CloudHSM klien dan alat baris perintah, menentukan alamat IP HSM di cluster Anda. Untuk mendapatkan alamat IP HSM, lihat cluster Anda di [AWS CloudHSM konsol](#), atau jalankan [describe-clusters](#) AWS CLI perintah. Dalam output perintah, alamat IP HSM adalah nilai bidang `EniIp`. Jika Anda memiliki lebih dari satu HSM, pilih alamat IP untuk salah satu HSMs; tidak masalah yang mana.

```
sudo /opt/cloudhsm/bin/configure -a <IP address>

Updating server config in /opt/cloudhsm/etc/cloudhsm_client.cfg
Updating server config in /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

3. Kunjungi [Aktifkan cluster di AWS CloudHSM](#).

Instal dan konfigurasikan AWS CloudHSM klien untuk KMU (Windows)

Untuk bekerja dengan modul keamanan perangkat keras (HSM) di AWS CloudHSM cluster Anda di Windows menggunakan `key_mgmt_util` (KMU), Anda memerlukan perangkat lunak klien untuk Windows. AWS CloudHSM Anda harus menginstalnya pada instans Windows Server yang Anda buat sebelumnya.

Untuk menginstal (atau memperbarui) klien Windows terbaru dan alat baris perintah

1. Hubungkan ke instans Windows Server Anda.
2. Unduh (`AWSCloudHSMClient-latest.msi`) terbaru dari halaman [unduh](#).
3. Buka lokasi unduhan Anda dan jalankan penginstal (`AWSCloudHSMClient-latest.msi`) dengan hak administratif.
4. Ikuti petunjuk penginstal, lalu pilih Tutup setelah penginstal selesai.
5. Salin sertifikat penerbitan yang ditandatangani sendiri -[salah satu yang Anda gunakan untuk menandatangani sertifikat klaster](#)—ke folder `C:\ProgramData\Amazon\CloudHSM`.
6. Jalankan perintah berikut untuk memperbarui file konfigurasi Anda. Pastikan untuk berhenti dan memulai klien selama konfigurasi ulang jika Anda memperbaruinya:

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" -a <HSM IP address>
```

7. Kunjungi [Aktifkan cluster di AWS CloudHSM](#).

Catatan:

- Jika Anda memperbarui klien, file konfigurasi yang ada dari instalasi sebelumnya tidak ditimpa.
- Penginstal AWS CloudHSM klien untuk Windows secara otomatis mendaftarkan API Kriptografi: Generasi Berikutnya (CNG) dan Penyedia Penyimpanan Kunci (KSP). Untuk mencopot pemasangan klien, jalankan penginstal lagi dan ikuti petunjuk mencopot pemasangan.
- Jika Anda menggunakan Linux, Anda dapat menginstal klien Linux. Lihat informasi yang lebih lengkap di [Instal dan konfigurasi AWS CloudHSM klien untuk KMU \(Linux\)](#).

Referensi untuk perintah AWS CloudHSM Key Management Utility

Alat baris perintah `key_mgmt_util` membantu Anda mengelola kunci dalam modul keamanan perangkat keras (HSM) di AWS CloudHSM cluster Anda, termasuk membuat, menghapus, dan menemukan kunci dan atributnya. Ini mencakup beberapa perintah, yang masing-masing dijelaskan secara detail dalam topik ini.

Untuk memulai dengan cepat, lihat [Memulai dengan AWS CloudHSM key_mgmt_util](#). Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#). Untuk informasi tentang alat baris perintah `cloudhsm_mgmt_util`, yang mencakup perintah untuk mengelola HSM dan pengguna di kluster Anda, lihat [AWS CloudHSM Utilitas Manajemen \(CMU\)](#).

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna krypto (CU).

Untuk mencantumkan semua perintah `key_mgmt_util`, ketik:

```
Command: help
```

Untuk mendapatkan bantuan untuk perintah `key_mgmt_util` tertentu, ketik:

```
Command: <command-name> -h
```

Untuk mengakhiri sesi `key_mgmt_util` Anda, ketik:

Command: **exit**

Topik berikut menjelaskan perintah dalam `key_mgmt_util`.

 Note

Beberapa perintah di `key_mgmt_util` dan `cloudhsm_mgmt_util` memiliki nama yang sama. Namun, perintah biasanya memiliki sintaks yang berbeda, output yang berbeda, dan fungsi yang sedikit berbeda.

Perintah	Deskripsi
<code>aesWrapUnwrap</code>	Mengenkripsi dan mendekripsi isi kunci dalam sebuah file.
<code>DeleteKey</code>	Menghapus kunci dari file. HSMs
<code>Error2String</code>	Mendapatkan kesalahan yang sesuai dengan kode kesalahan heksadesimal <code>key_mgmt_util</code> .
<code>keluar</code>	Keluar dari <code>key_mgmt_util</code> .
<code>exportPrivateKey</code>	Mengekspor salinan kunci pribadi dari HSM ke file pada disk.
<code>exportPubKey</code>	Ekspor salinan kunci publik dari HSM ke file.
<code>exSymKey</code>	Mengekspor salinan teks biasa dari kunci simetris dari file ke HSMs file.
<code>extractMaskedObject</code>	Ekstrak kunci dari HSM sebagai file objek tertutup.
<code>FindKey</code>	Cari kunci berdasarkan nilai atribut kunci.
<code>findSingleKey</code>	Memverifikasi bahwa kunci ada pada semua HSMs di cluster.

Perintah	Deskripsi
DSAKeypasangan gen	Menghasilkan key pair Digital Signing Algorithm (DSA) di file Anda HSMs.
ECCKeypasangan gen	Menghasilkan key pair Elliptic Curve Cryptography (ECC) di file Anda. HSMs
RSAKeypasangan gen	Menghasilkan key pair asimetris RSA di file Anda. HSMs
genSymKey	Menghasilkan kunci simetris di HSMs
getAttribute	Mendapat nilai atribut untuk kunci AWS CloudHSM dan menuliskannya ke file.
getCaviumPrivKunci	Membuat kunci privat versi format PEM palsu dan mengeksponnya ke file.
GetCert	Mengambil sertifikat partisi HSM dan menyimpannya ke file.
getKeyInfo	Mendapatkan pengguna HSM IDs dari pengguna yang dapat menggunakan kunci. Jika kuncinya dikendalikan kuorum, kunci mendapat jumlah pengguna dalam kuorum.
membantu	Menampilkan informasi bantuan tentang perintah yang tersedia di key_mgmt_util.
importPrivateKey	Mengimpor kunci privat ke HSM.
importPubKey	Mengimpor kunci privat ke HSM.
imSymKey	Ekspor salinan plaintext dari kunci simetris dari HSM ke file.

Perintah	Deskripsi
<u>insertMaskedObject</u>	Menyisipkan objek tertutup dari file pada disk ke dalam HSM yang terkandung dalam klaster terkait dengan klaster asal objek. Klaster terkait adalah setiap klaster <u>yang dihasilkan dari cadangan dari klaster asal</u> .
<u>???</u>	Menentukan apakah file yang diberikan berisi kunci pribadi nyata atau contoh kunci PEM.
<u>ListatTributes</u>	Daftar atribut AWS CloudHSM kunci dan konstanta yang mewakili mereka.
<u>ListUsers</u>	Mendapatkan pengguna di HSMs, tipe pengguna dan ID mereka, dan atribut lainnya.
<u>LoginHSM dan LogouthSM</u>	Masuk dan keluar dari HSMs cluster.
<u>setAttribute</u>	Mengonversi kunci sesi untuk kunci persisten.
<u>tanda</u>	Hasilkan tanda tangan untuk file menggunakan kunci privat yang dipilih.
<u>unWrapKey</u>	Mengimpor kunci yang dibungkus (dienkripsi) dari file ke file. HSMs
<u>verifikasi</u>	Memverifikasi apakah kunci yang diberikan digunakan untuk menandatangani file yang diberikan.
<u>WrapKey</u>	Mengekspor salinan kunci terenkripsi dari HSM ke file.

Mengenkripsi dan mendekripsi file menggunakan AWS CloudHSM KMU

Gunakan `aesWrapUnwrap` perintah di AWS CloudHSM `key_mgmt_util` untuk mengenkripsi atau mendekripsi isi file pada disk. Perintah ini dirancang untuk membungkus dan membuka kunci enkripsi, tetapi Anda dapat menggunakannya pada file yang berisi kurang dari 4 KB (4096 byte) data.

aesWrapUnwrap menggunakan AES Key Wrap dengan PKCS #5 Padding. Kunci AES digunakan pada HSM sebagai pembungkus atau pembuka bungkus kunci. Kemudian, hasilnya ditulis ke file lain pada disk.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
aesWrapUnwrap -h

aesWrapUnwrap -m <wrap-unwrap mode>
                -f <file-to-wrap-unwrap>
                -w <wrapping-key-handle>
                [-i <wrapping-IV>]
                [-out <output-file>]
```

Contoh

Contoh ini menunjukkan cara menggunakan `aesWrapUnwrap` untuk mengenkripsi dan mendekripsi kunci enkripsi dalam sebuah file.

Example: Bungkus kunci enkripsi

Perintah ini menggunakan `aesWrapUnwrap` untuk membungkus kunci simetris Triple DES yang [diekspor dari HSM dalam plaintext](#) ke dalam file `3DES.key`. Anda dapat menggunakan perintah yang sama untuk membungkus kunci yang disimpan dalam file.

Perintah menggunakan parameter `-m` dengan nilai 1 untuk mengindikasikan mode bungkus. Ini menggunakan parameter `-w` untuk menentukan kunci AES di HSM (handel kunci 6) sebagai kunci pembungkus. Ini menulis hasil kunci dibungkus untuk file `3DES.key.wrapped`.

Output menunjukkan bahwa perintah berhasil dan bahwa operasi menggunakan IV default, yang lebih disukai.

```
Command: aesWrapUnwrap -f 3DES.key -w 6 -m 1 -out 3DES.key.wrapped
```

```
Warning: IV (-i) is missing.
```

```
0xA6A6A6A6A6A6A6A6 is considered as default IV
```

```
result data:
```

```
49 49 E2 D0 11 C1 97 22
```



```
17 43 BD E3 4E F4 12 75
8D C1 34 CF 26 10 3A 8D
6D 0A 7B D5 D3 E8 4D C2
79 09 08 61 94 68 51 B7
```

```
result written to file 3DES.key.wrapped
```

```
Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

Example: Buka kunci enkripsi

Contoh ini menunjukkan cara menggunakan `aesWrapUnwrap` untuk membuka (mendekripsi) kunci yang dibungkus (terenkripsi) dalam sebuah file. Anda mungkin ingin melakukan operasi seperti ini sebelum mengimpor kunci ke HSM. Misalnya, jika Anda mencoba menggunakan [imSymKey](#) perintah untuk mengimpor kunci terenkripsi, itu mengembalikan kesalahan karena kunci terenkripsi tidak memiliki format yang diperlukan untuk kunci teks biasa dari jenis itu.

Perintah membuka kunci dalam `3DES.key.wrapped` dan menulis plaintext ke file `3DES.key.unwrapped`. Perintah menggunakan parameter `-m` dengan nilai `0` untuk mengindikasikan mode buka. Ini menggunakan parameter `-w` untuk menentukan kunci AES di HSM (handel kunci 6) sebagai kunci pembungkus. Ini menulis hasil kunci dibungkus untuk file `3DES.key.unwrapped`.

```
Command: aesWrapUnwrap -m 0 -f 3DES.key.wrapped -w 6 -out 3DES.key.unwrapped
```

```
Warning: IV (-i) is missing.
```

```
0xA6A6A6A6A6A6A6A6 is considered as default IV
```

```
result data:
```

```
14 90 D7 AD D6 E4 F5 FA
A1 95 6F 24 89 79 F3 EE
37 21 E6 54 1F 3B 8D 62
```

```
result written to file 3DES.key.unwrapped
```

```
Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

Parameter

`-h`

Menampilkan bantuan untuk perintah.

Wajib: Ya

-m

Menentukan mode. Untuk membungkus (mengenkripsi) isi file, ketik 1; untuk membuka (mendekripsi) isi file, ketik 0.

Wajib: Ya

-f

Menentukan file untuk dibungkus. Masukkan file yang berisi data kurang dari 4 KB (4096 byte). Operasi ini dirancang untuk membungkus dan membuka kunci enkripsi.

Wajib: Ya

-w

Menentukan kunci pembungkus. Masukkan handle kunci dari kunci AES pada HSM. Parameter ini diperlukan. Untuk menemukan handle kunci, gunakan perintah [findKey](#).

Untuk membuat kunci pembungkus, gunakan [genSymKey](#) untuk menghasilkan kunci AES (tipe 31).

Wajib: Ya

-i

Menentukan nilai awal alternatif (IV) untuk algoritme. Gunakan nilai default kecuali Anda memiliki syarat khusus yang memerlukan alternatif.

Default: 0xA6A6A6A6A6A6A6A6. Nilai default ditentukan dalam spesifikasi algoritme [Bungkus Kunci AES](#).

Wajib: Tidak

-out

Menentukan nama alternatif untuk file output yang berisi kunci terbungkus atau terbuka. Default-nya `wrapped_key` (untuk operasi bungkus) dan `unwrapped_key` (untuk operasi buka) di direktori lokal.

Jika file ada, `aesWrapUnwrap` menimpa tanpa peringatan. Jika perintah gagal, `aesWrapUnwrap` membuat sebuah file output tanpa isi.

Default: Untuk bungkus: `wrapped_key`. Untuk membuka: `unwrapped_key`.

Wajib: Tidak

Topik terkait

- [exSymKey](#)
- [imSymKey](#)
- [unWrapKey](#)
- [WrapKey](#)

Hapus AWS CloudHSM kunci menggunakan KMU

Gunakan `deleteKey` perintah di AWS CloudHSM `key_mgmt_util` untuk menghapus kunci dari modul keamanan perangkat keras (HSM) dalam sebuah cluster. AWS CloudHSM Anda hanya dapat menghapus satu item dalam satu waktu. Menghapus satu kunci pada pasangan kunci tidak berpengaruh pada kunci lainnya pada pasangan.

Hanya pemilik kunci yang dapat menghapus kunci. Pengguna yang berbagi kunci dapat menggunakannya dalam operasi kriptografi, tetapi tidak menghapusnya.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
deleteKey -h  
  
deleteKey -k
```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `deleteKey` untuk menghapus kunci dari Anda HSMs.

Example: Hapus kunci

Perintah ini menghapus kunci dengan handel kunci 6. Ketika perintah berhasil, `deleteKey` mengembalikan pesan sukses dari setiap HSM di klaster.

```
Command: deleteKey -k 6
```

```
Cfm3DeleteKey returned: 0x00 : HSM Return: SUCCESS
```

Cluster Error Status

Node id 1 and err state 0x00000000 : HSM Return: SUCCESS

Node id 2 and err state 0x00000000 : HSM Return: SUCCESS

Example: Hapus kunci (kegagalan)

Ketika perintah gagal karena tidak ada kunci yang memiliki handel kunci yang ditentukan, `deleteKey` mengembalikan pesan kesalahan penanganan objek tidak valid.

Command: **`deleteKey -k 252126`**

Cfm3FindKey returned: 0xa8 : HSM Error: Invalid object handle is passed to this operation

Cluster Error Status

Node id 1 and err state 0x000000a8 : HSM Error: Invalid object handle is passed to this operation

Node id 2 and err state 0x000000a8 : HSM Error: Invalid object handle is passed to this operation

Ketika perintah gagal karena pengguna saat ini bukan pemilik kunci, perintah mengembalikan kesalahan akses ditolak.

Command: **`deleteKey -k 262152`**

Cfm3DeleteKey returned: 0xc6 : HSM Error: Key Access is denied.

Parameter

-h

Menampilkan bantuan baris perintah untuk perintah.

Wajib: Ya

-k

Menentukan handel kunci dari kunci untuk dihapus. Untuk menemukan kunci handel dari kunci di HSM, gunakan [findKey](#).

Wajib: Ya

Topik terkait

- [FindKey](#)

Jelaskan AWS CloudHSM kesalahan menggunakan KMU

Gunakan perintah `Error2String` helper di AWS CloudHSM `key_mgmt_util` untuk mengembalikan kesalahan yang sesuai dengan kode kesalahan heksadesimal `key_mgmt_util`. Anda dapat menggunakan perintah ini saat memecahkan masalah perintah dan skrip Anda.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna kriptografi (CU).

Sintaksis

```
Error2String -h
```

```
Error2String -r <response-code>
```

Contoh

Contoh ini menunjukkan cara menggunakan `Error2String` untuk mendapatkan string kesalahan untuk kode kesalahan `key_mgmt_util`.

Example: Dapatkan deskripsi kesalahan

Perintah ini mendapat deskripsi kesalahan untuk kode kesalahan `0xdb`. Deskripsi menjelaskan bahwa upaya untuk masuk ke `key_mgmt_util` gagal karena pengguna memiliki jenis pengguna yang salah. Hanya pengguna kriptografi (CU) yang bisa masuk ke `key_mgmt_util`.

```
Command: Error2String -r 0xdb
```

```
Error Code db maps to HSM Error: Invalid User Type.
```

Example: Temukan kode kesalahan

Contoh ini menunjukkan tempat untuk menemukan kode kesalahan dalam kesalahan `key_mgmt_util`. Kode kesalahan, `0xc6`, muncul setelah string: `Cfm3<command-name> returned: .`

Dalam contoh ini, [getKeyInfo](#) menunjukkan bahwa pengguna saat ini (pengguna 4) dapat menggunakan kunci dalam operasi kriptografi. Meski begitu, saat pengguna mencoba

menggunakannya [deleteKey](#) untuk menghapus kunci, perintah mengembalikan kode kesalahan 0xc6.

```
Command: deleteKey -k 262162
```

```
Cfm3DeleteKey returned: <0xc6> : HSM Error: Key Access is denied
```

```
Cluster Error Status
```

```
Command: getKeyInfo -k 262162
```

```
Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS
```

```
Owned by user 3
```

```
also, shared to following 1 user(s):
```

```
4
```

Jika kesalahan 0xc6 dilaporkan kepada Anda, Anda dapat menggunakan perintah `Error2String` seperti ini untuk mencari kesalahan. Dalam hal ini, perintah `deleteKey` gagal dengan kesalahan akses ditolak karena kunci dibagi dengan pengguna saat ini tetapi dimiliki oleh pengguna yang berbeda. Hanya pemilik kunci yang memiliki izin untuk menghapus kunci.

```
Command: Error2String -r 0xa8
```

```
Error Code c6 maps to HSM Error: Key Access is denied
```

Parameter

-h

Menampilkan bantuan untuk perintah.

Wajib: Ya

-r

Menentukan kode kesalahan heksadesimal. Indikator heksadesimal 0x diperlukan.

Wajib: Ya

Keluar dari AWS CloudHSM KMU

Gunakan `exit` perintah di `key_mgmt_util` untuk AWS CloudHSM keluar dari `key_mgmt_util`. Setelah berhasil keluar, Anda akan dikembalikan ke baris perintah standar Anda.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#).

Sintaks

```
exit
```

Parameter

Tidak ada parameter untuk perintah ini.

Topik terkait

- [Mulai key_mgmt_util](#)

Ekspor AWS CloudHSM kunci pribadi menggunakan KMU

Gunakan `exportPrivateKey` perintah di AWS CloudHSM `key_mgmt_util` untuk mengeksport kunci pribadi asimetris dari modul keamanan perangkat keras (HSM) ke file. HSM tidak mengizinkan ekspor langsung kunci dalam cleartext. Perintah membungkus kunci pribadi menggunakan kunci pembungkus AES yang Anda tentukan, mendekripsi byte yang dibungkus, dan menyalin kunci pribadi cleartext ke file.

`exportPrivateKey` perintah tidak menghapus kunci dari HSM, mengubah [atribut kuncinya](#), atau mencegah Anda menggunakan kunci dalam operasi kriptografi lebih lanjut. Anda dapat mengeksport kunci yang sama beberapa kali.

Anda hanya dapat mengeksport kunci privat yang memiliki atribut `OBJ_ATTR_EXTRACTABLE` bernilai 1. Anda harus menentukan kunci pembungkus AES yang memiliki `OBJ_ATTR_WRAP` dan nilai `OBJ_ATTR_DECRYPT` atribut. 1 Untuk menemukan atribut kunci, gunakan perintah [getAttribute](#).

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
exportPrivateKey -h
```

```
exportPrivateKey -k <private-key-handle>
                 -w <wrapping-key-handle>
                 -out <key-file>
                 [-m <wrapping-mechanism>]
                 [-wk <wrapping-key-file>]
```

Contoh

Contoh ini menunjukkan cara menggunakan `exportPrivateKey` untuk mengekspor kunci privat dari HSM.

Example: Ekspor kunci pribadi

Perintah ini mengekspor kunci privat dengan handel 15 menggunakan kunci pembungkus dengan handel 16 ke file PEM bernama `exportKey.pem`. Ketika perintah berhasil, `exportPrivateKey` mengembalikan pesan sukses.

```
Command: exportPrivateKey -k 15 -w 16 -out exportKey.pem
```

```
Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS
```

```
Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

```
PEM formatted private key is written to exportKey.pem
```

Parameter

Perintah ini membawa parameter berikut.

-h

Menampilkan bantuan baris perintah untuk perintah.

Wajib: Ya

-k

Menentukan handel kunci dari kunci privat yang akan diekspor.

Wajib: Ya

-w

Menentukan handel kunci dari kunci pembungkus. Parameter ini diperlukan. Untuk menemukan handel kunci, gunakan perintah [findKey](#).

Untuk menentukan apakah kunci dapat digunakan sebagai kunci pembungkus, gunakan [getAttribute](#) untuk mendapatkan nilai dari atribut OBJ_ATTR_WRAP (262). Untuk membuat kunci pembungkus, gunakan [genSymKey](#) untuk membuat kunci AES (tipe 31).

Jika Anda menggunakan parameter -wk untuk menentukan kunci pembuka pembungkus eksternal, kunci pembungkus -w digunakan untuk membungkus, tapi tidak membuka, kunci selama ekspor.

Wajib: Ya

-out

Menentukan nama file tempat kunci privat yang diekspor akan ditulis.

Wajib: Ya

-m

Menentukan mekanisme pembungkus yang digunakan untuk membungkus kunci privat yang diekspor. Satu-satunya nilai yang valid adalah 4, yang mewakili NIST_AES_WRAP mechanism.

Default: 4 (NIST_AES_WRAP)

Wajib: Tidak

-wk

Menentukan kunci yang akan digunakan untuk membuka kunci yang diekspor. Masukkan jalur dan nama file yang berisi kunci AES plaintext.

Bila Anda menyertakan parameter ini, exportPrivateKey menggunakan kunci dalam file -w untuk membungkus kunci yang diekspor dan menggunakan kunci yang ditentukan oleh parameter -wk untuk membukanya.

Default: Gunakan kunci pembungkus yang ditentukan dalam parameter -w untuk membungkus dan membuka.

Wajib: Tidak

Topik terkait

- [importPrivateKey](#)
- [WrapKey](#)
- [unWrapKey](#)
- [genSymKey](#)

Ekspor AWS CloudHSM kunci publik menggunakan KMU

Gunakan `exportPubKey` perintah di AWS CloudHSM `key_mgmt_util` untuk mengekspor kunci publik dalam HSM ke file. Anda dapat menggunakannya untuk mengekspor kunci privat yang Anda hasilkan pada HSM. Anda juga dapat menggunakan perintah ini untuk mengekspor kunci publik yang diimpor ke HSM, seperti yang diimpor dengan perintah [importPubKey](#)

Operasi `exportPubKey` menyalin bahan kunci untuk file yang Anda tentukan. Tapi itu tidak menghapus kunci dari HSM, mengubah [atribut kunci](#), atau mencegah Anda menggunakan kunci dalam operasi kriptografi lebih lanjut. Anda dapat mengekspor kunci yang sama beberapa kali.

Anda hanya dapat mengekspor kunci publik yang memiliki `OBJ_ATTR_EXTRACTABLE` bernilai 1. Untuk menemukan atribut kunci, gunakan perintah [getAttribute](#).

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
exportPubKey -h

exportPubKey -k <public-key-handle>
               -out <key-file>
```

Contoh

Contoh ini menunjukkan cara menggunakan `exportPubKey` untuk mengekspor kunci publik dari HSM.

Example: Ekspor kunci publik

Perintah ini ekspor kunci publik dengan handle 10 ke file bernama `public.pem`. Ketika perintah berhasil, `exportPubKey` mengembalikan pesan sukses.

```
Command: exportPubKey -k 10 -out public.pem
```

```
PEM formatted public key is written to public.pem
```

```
Cfm3ExportPubKey returned: 0x00 : HSM Return: SUCCESS
```

Parameter

Perintah ini membawa parameter berikut.

-h

Menampilkan bantuan baris perintah untuk perintah.

Wajib: Ya

-k

Menentukan handel kunci dari kunci publik yang akan diekspor.

Wajib: Ya

-out

Menentukan nama file tempat kunci publik yang diekspor akan ditulis.

Wajib: Ya

Topik terkait

- [importPubKey](#)
- [Menghasilkan Kunci](#)

Ekspor salinan teks biasa dari AWS CloudHSM kunci menggunakan KMU

Gunakan `exSymKey` perintah di alat AWS CloudHSM `key_mgmt_util` untuk mengekspor salinan teks biasa dari kunci simetris dari modul keamanan perangkat keras (HSM) dan menyimpannya dalam file pada disk. Untuk mengekspor salinan kunci yang dienkripsi (dibungkus), gunakan [wrapKey](#). Untuk mengimpor kunci plaintext, seperti yang `exSymKey` mengekspor, gunakan. [imSymKey](#)

Selama proses ekspor, `exSymKey` menggunakan kunci AES yang Anda tentukan (kunci pembungkus) untuk membungkus (mengenkripsi) dan kemudian membuka(dekripsi) kunci yang akan diekspor. Namun, hasil dari operasi ekspor adalah plaintext (terbuka) pada disk.

Hanya pemilik kunci, yaitu pengguna CU yang membuat kunci, dapat mengekspornya. Pengguna yang berbagi kunci dapat menggunakannya dalam operasi kriptografi, tetapi mereka tidak dapat mengekspornya.

Operasi `exSymKey` menyalin materi kunci ke file yang Anda tentukan, tetapi tidak menghapus kunci dari HSM, mengubah [atribut kunci](#), atau mencegah Anda menggunakan kunci dalam operasi kriptografi. Anda dapat mengekspor kunci yang sama beberapa kali.

`exSymKey` mengekspor hanya kunci simetris. Untuk mengekspor kunci publik, gunakan [exportPubKey](#). Untuk mengekspor kunci pribadi, gunakan [exportPrivateKey](#).

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [Masuk](#) ke HSM sebagai pengguna kriptografi (CU).

Sintaksis

```
exSymKey -h

exSymKey -k <key-to-export>
          -w <wrapping-key>
          -out <key-file>
          [-m 4]
          [-wk <unwrapping-key-file> ]
```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `exSymKey` untuk mengekspor kunci simetris yang Anda miliki dari kunci simetris Anda HSMs.

Example: Ekspor kunci simetris 3DES

Perintah ini mengekspor kunci simetris Triple DES (3DES) (handel kunci 7). Perintah ini menggunakan kunci AES yang ada (handel kunci 6) di HSM sebagai kunci pembungkus. Kemudian, perintah ini menulis plaintext dari kunci 3DES ke file `3DES.key`.

Output menunjukkan bahwa kunci 7 (kunci 3DES) berhasil dibungkus dan dibuka, dan kemudian ditulis ke file `3DES.key`.

 Warning

Meskipun output mengatakan bahwa “Kunci Simetris Terbungkus” ditulis ke file output, file output berisi kunci plaintext (terbuka).

```
Command: exSymKey -k 7 -w 6 -out 3DES.key
```

```
Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS
```

```
Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

```
Wrapped Symmetric Key written to file "3DES.key"
```

Example: Mengekspor dengan kunci pembungkus khusus sesi

Contoh ini menunjukkan bagaimana menggunakan kunci yang ada hanya dalam sesi sebagai kunci pembungkus. Karena kunci yang akan diekspor dibungkus, segera dibuka, dan dikirim sebagai plaintext, tidak perlu menyimpan kunci pembungkus.

Serangkaian perintah ini mengekspor kunci AES dengan handel kunci 8 dari HSM. Perintah ini menggunakan kunci sesi AES yang dibuat khusus untuk tujuan tersebut.

Perintah pertama digunakan [genSymKey](#) untuk membuat kunci AES 256-bit. Perintah ini menggunakan parameter `-sess` untuk membuat kunci yang ada hanya dalam sesi saat ini.

Output menunjukkan bahwa HSM membuat kunci 262168.

```
Command: genSymKey -t 31 -s 32 -l AES-wrapping-key -sess
```

```
Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS
```

```
Symmetric Key Created. Key Handle: 262168
```

```
Cluster Error Status
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

Selanjutnya, contoh memverifikasi bahwa kunci 8, kunci yang akan diekspor, adalah kunci simetris yang dapat diekstrak. Contoh juga memverifikasi bahwa kunci pembungkus, kunci 262168, adalah

kunci AES yang ada hanya dalam sesi. Anda dapat menggunakan perintah [findKey](#), tetapi contoh ini mengeksport atribut dari kedua kunci ke file dan kemudian menggunakan grep untuk menemukan nilai atribut yang relevan dalam file.

Perintah ini menggunakan `getAttribute` dengan `-a` bernilai 512 (all) untuk mendapatkan semua atribut untuk kunci 8 dan 262168. Untuk informasi tentang elemen atribut kunci, lihat [the section called "Referensi atribut kunci"](#).

```
getAttribute -o 8 -a 512 -out attributes/attr_8
getAttribute -o 262168 -a 512 -out attributes/attr_262168
```

Perintah ini menggunakan `grep` untuk memverifikasi atribut kunci yang akan diekspor (kunci 8) dan kunci pembungkus hanya sesi (kunci 262168).

```
// Verify that the key to be exported is a symmetric key.
$ grep -A 1 "OBJ_ATTR_CLASS" attributes/attr_8
OBJ_ATTR_CLASS
0x04

// Verify that the key to be exported is extractable.
$ grep -A 1 "OBJ_ATTR_KEY_TYPE" attributes/attr_8
OBJ_ATTR_EXTRACTABLE
0x00000001

// Verify that the wrapping key is an AES key
$ grep -A 1 "OBJ_ATTR_KEY_TYPE" attributes/attr_262168
OBJ_ATTR_KEY_TYPE
0x1f

// Verify that the wrapping key is a session key
$ grep -A 1 "OBJ_ATTR_TOKEN" attributes/attr_262168
OBJ_ATTR_TOKEN
0x00

// Verify that the wrapping key can be used for wrapping
$ grep -A 1 "OBJ_ATTR_WRAP" attributes/attr_262168
OBJ_ATTR_WRAP
0x00000001
```

Akhirnya, kami menggunakan perintah `exSymKey` untuk mengeksport kunci 8 menggunakan kunci sesi (kunci 262168) sebagai kunci pembungkus.

Saat sesi berakhir, kunci 262168 sudah tidak ada lagi.

```
Command:  exSymKey -k 8 -w 262168 -out aes256_H8.key
```

```
Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS
```

```
Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

```
Wrapped Symmetric Key written to file "aes256_H8.key"
```

Example: Gunakan kunci membuka bungkus eksternal

Contoh ini menunjukkan bagaimana menggunakan kunci pembuka pembungkus eksternal untuk mengekspor kunci dari HSM.

Ketika Anda mengekspor kunci dari HSM, Anda menentukan kunci AES pada HSM menjadi kunci pembungkus. Secara default, bahwa kunci pembungkus digunakan untuk membungkus dan membuka kunci yang akan diekspor. Namun, Anda dapat menggunakan parameter `-wk` untuk memberitahu `exSymKey` untuk menggunakan kunci eksternal dalam file pada disk untuk membuka bungkus. Ketika Anda melakukannya, kunci yang ditentukan oleh parameter `-w` membungkus kunci target, dan kunci dalam file yang ditentukan oleh parameter `-wk` membuka kunci.

Karena kunci pembungkus harus kunci AES, yang simetris, kunci pembungkus di HSM dan kunci pembuka pada disk harus memiliki bahan kunci yang sama. Untuk melakukan ini, Anda harus mengimpor kunci pembungkus ke HSM atau mengekspor kunci pembungkus dari HSM sebelum operasi ekspor.

Contoh ini membuat kunci di luar HSM dan mengimpornya ke dalam HSM. Contoh ini menggunakan salinan internal kunci untuk membungkus kunci simetris yang sedang diekspor, dan salinan kunci dalam file untuk membukanya.

Perintah pertama menggunakan `genSymKey` Untuk membuat kunci AES 256-bit. Perintah ini menyimpan kunci ke file `aes256-forImport.key`. Perintah `OpenSSL` tidak mengembalikan output apa pun, tetapi Anda dapat menggunakan beberapa perintah untuk mengonfirmasi keberhasilannya. Contoh ini menggunakan alat `wc` (jumlah kata), yang menegaskan bahwa file yang berisi 32 byte data.

```
$ openssl rand -out keys/aes256-forImport.key 32
```

```
$ wc keys/aes256-forImport.key
0  2 32 keys/aes256-forImport.key
```

Perintah ini menggunakan [imSymKey](#) perintah untuk mengimpor kunci AES dari `aes256-forImport.key` file ke HSM. Ketika perintah selesai, kunci ada di HSM dengan handel kunci 262167 dan dalam file `aes256-forImport.key`.

```
Command: imSymKey -f keys/aes256-forImport.key -t 31 -l aes256-imported -w 6
```

```
Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

```
Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS
```

```
Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS
```

```
Symmetric Key Unwrapped. Key Handle: 262167
```

```
Cluster Error Status
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Perintah ini menggunakan kunci dalam operasi ekspor. Perintah menggunakan `exSymKey` untuk mengekspor kunci 21, kunci AES 192-bit. Untuk membungkus kunci, perintah menggunakan kunci 262167, yang merupakan salinan yang diimpor ke dalam HSM. Untuk membuka kunci, perintah menggunakan bahan kunci yang sama dalam file `aes256-forImport.key`. Saat perintah selesai, kunci 21 diekspor ke file `aes192_h21.key`.

```
Command: exSymKey -k 21 -w 262167 -out aes192_H21.key -wk aes256-forImport.key
```

```
Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS
```

```
Wrapped Symmetric Key written to file "aes192_H21.key"
```

Parameter

`-h`

Menampilkan bantuan untuk perintah.

Wajib: Ya

-k

Menentukan handel kunci dari kunci untuk ekspor. Parameter ini diperlukan. Masukkan handel kunci dari kunci simetris yang Anda miliki. Parameter ini diperlukan. Untuk menemukan handel kunci, gunakan perintah [findKey](#).

Untuk memverifikasi bahwa kunci dapat diekspor, gunakan perintah [getAttribute](#) untuk mendapatkan nilai atribut OBJ_ATTR_EXTRACTABLE, yang diwakili oleh konstanta 354. Selain itu, Anda hanya dapat mengekspor kunci yang Anda miliki. Untuk menemukan pemilik kunci, gunakan [getKeyInfo](#) perintah.

Wajib: Ya

-w

Menentukan handel kunci dari kunci pembungkus. Parameter ini diperlukan. Untuk menemukan handel kunci, gunakan perintah [findKey](#).

Kunci pembungkus adalah kunci dalam HSM yang digunakan untuk mengenkripsi (membungkus) dan kemudian mendekripsi (membuka) kunci untuk diekspor. Hanya kunci AES yang dapat digunakan sebagai kunci pembungkus.

Anda dapat menggunakan kunci AES (dari berbagai ukuran) sebagai kunci pembungkus. Karena kunci pembungkus membungkus, dan kemudian segera membuka, kunci target, Anda dapat menggunakan sebagai kunci AES hanya sesi sebagai kunci pembungkus. Untuk menentukan apakah kunci dapat digunakan sebagai kunci pembungkus, gunakan [getAttribute](#) untuk mendapatkan nilai dari atribut OBJ_ATTR_WRAP, yang diwakili oleh konstanta 262. Untuk membuat kunci pembungkus, gunakan [genSymKey](#) untuk membuat kunci AES (tipe 31).

Jika Anda menggunakan parameter -wk untuk menentukan kunci membuka pembungkus eksternal, kunci pembungkus -w digunakan untuk membungkus, tetapi tidak untuk membuka, kunci selama ekspor.

 Note

Kunci 4 merupakan kunci internal yang tidak didukung. Kami merekomendasikan bahwa Anda menggunakan kunci AES yang Anda buat dan kelola sebagai kunci pembungkus.

Wajib: Ya

-out

Menentukan jalur dan nama file output. Ketika perintah berhasil, file ini berisi kunci yang diekspor dalam plaintext. Jika file sudah ada, perintah akan menimpa tanpa peringatan.

Wajib: Ya

-m

Menentukan mekanisme pembungkus. Satu-satunya nilai yang valid adalah 4, yang mewakili mekanisme NIST_AES_WRAP.

Wajib: Tidak

Default: 4 ()

-wk

Gunakan kunci AES dalam file tertentu untuk membuka kunci yang sedang diekspor. Masukkan jalur dan nama file yang berisi kunci AES plaintext.

Bila Anda menyertakan parameter ini, `exSymKey` menggunakan kunci dalam HSM yang ditentukan oleh parameter `-w` untuk membungkus kunci yang sedang diekspor dan menggunakan kunci dalam file `-wk` untuk membukanya. Nilai parameter `-w` dan `-wk` harus menyelesaikan ke kunci plaintext yang sama.

Wajib: Tidak

Default: Gunakan kunci pembungkus pada HSM untuk membuka.

Topik terkait

- [genSymKey](#)
- [imSymKey](#)
- [WrapKey](#)

Ekstrak AWS CloudHSM kunci menggunakan KMU

Gunakan `extractMaskedObject` perintah di AWS CloudHSM `key_mgmt_util` untuk mengekstrak kunci dari modul keamanan perangkat keras (HSM) dan menyimpannya ke file sebagai objek bertopeng. Objek tertutup adalah objek dikloning yang hanya dapat digunakan setelah memasukkannya

kembali ke dalam klaster asli dengan menggunakan perintah [insertMaskedObject](#) Anda hanya dapat memasukkan objek tertutup ke dalam klaster yang sama dengan klaster asalnya dihasilkan, atau kloning dari klaster tersebut. Klaster ini termasuk versi kloning dari klaster yang dihasilkan dengan [menyalin cadangan di seluruh wilayah](#) dan [menggunakan cadangan tersebut untuk membuat klaster baru](#).

Objek tertutup adalah cara yang efisien untuk membongkar dan menyinkronkan kunci, termasuk kunci yang tidak dapat diekstrak (yaitu, kunci yang memiliki nilai [OBJ_ATTR_EXTRACTABLE](#) dari 0). [Dengan cara ini, kunci dapat disinkronkan dengan aman di seluruh cluster terkait di berbagai wilayah tanpa perlu memperbarui file konfigurasi. AWS CloudHSM](#)

Important

Setelah penyisipan, benda tertutup didekripsi dan diberi handel kunci yang berbeda dari handel kunci dari kunci asli. Sebuah objek tertutup mencakup semua metadata yang terkait dengan kunci asli, termasuk atribut, kepemilikan dan berbagi informasi, dan pengaturan kuorum. Jika Anda perlu untuk menyinkronkan kunci di klaster dalam aplikasi, gunakan [syncKey](#) di `cloudhsm_mgmt_util` sebagai gantinya.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM. Perintah `extractMaskedObject` dapat digunakan baik oleh CU yang memiliki kunci atau CO.

Sintaksis

```
extractMaskedObject -h

extractMaskedObject -o <object-handle>
                    -out <object-file>
```

Contoh

Contoh ini menunjukkan cara menggunakan `extractMaskedObject` untuk mengekstraksi kunci dari HSM sebagai objek tertutup.

Example: Ekstrak objek bertopeng

Perintah ini mengekstraksi objek tertutup dari HSM dari kunci dengan handel 524295 dan menyimpannya sebagai file bernama `maskedObj`. Ketika perintah berhasil, `extractMaskedObject` mengembalikan pesan sukses.

```
Command: extractMaskedObject -o 524295 -out maskedObj
```

```
Object was masked and written to file "maskedObj"
```

```
Cfm3ExtractMaskedObject returned: 0x00 : HSM Return: SUCCESS
```

Parameter

Perintah ini membawa parameter berikut.

-h

Menampilkan bantuan baris perintah untuk perintah.

Wajib: Ya

-o

Menentukan handel kunci untuk mengekstraksi sebagai objek tertutup.

Wajib: Ya

-out

Menentukan nama file yang objek tertutup akan disimpan.

Wajib: Ya

Topik terkait

- [insertMaskedObject](#)
- [SyncKey](#)
- [Menyalin Cadangan di Seluruh Wilayah](#)
- [Membuat AWS CloudHSM Cluster dari Backup Sebelumnya](#)

Cari AWS CloudHSM kunci berdasarkan atribut menggunakan KMU

Gunakan findKey perintah di AWS CloudHSM key_mgmt_util untuk mencari kunci berdasarkan nilai atribut kunci. Ketika kunci cocok dengan semua kriteria yang Anda tetapkan, findKey mengembalikan handel kunci. Tanpa parameter, findKey mengembalikan handel kunci dari semua kunci yang dapat Anda gunakan dalam HSM. Untuk menemukan nilai atribut dari kunci tertentu, gunakan [getAttribute](#).

Seperti semua perintah `key_mgmt_util`, `findKey` adalah pengguna tertentu. Perintah ini mengembalikan hanya kunci yang dapat digunakan pengguna saat ini dalam operasi kriptografi. Ini termasuk kunci yang dimiliki pengguna saat ini dan kunci yang telah dibagi dengan pengguna saat ini.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai `key\_mgmt\_util`](#) dan [masuk ke HSM sebagai pengguna kriptografi \(CU\)](#).

Sintaksis

```
findKey -h

findKey [-c <key class>]
        [-t <key type>]
        [-l <key label>]
        [-id <key ID>]
        [-sess (0 | 1)]
        [-u <user-ids>]
        [-m <modulus>]
        [-kcv <key_check_value>]
```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `findKey` untuk menemukan dan mengidentifikasi kunci di Anda HSMs.

Example: Temukan semua kunci

Perintah ini menemukan semua kunci untuk pengguna saat ini di HSM. Outputnya mencakup kunci yang dimiliki dan dibagikan pengguna, dan semua kunci publik di HSMs.

Untuk mendapatkan atribut kunci dengan handle kunci tertentu, gunakan [getAttribute](#). Untuk menentukan apakah pengguna saat ini memiliki atau membagikan kunci tertentu, gunakan [getKeyInfo](#) atau [findAllKeys](#) di `cloudhsm_mgmt_util`.

Command: **findKey**

Total number of keys present 13

number of keys matched from start index 0::12

6, 7, 524296, 9, 262154, 262155, 262156, 262157, 262158, 262159, 262160, 262161, 262162

Cluster Error Status

Node id 1 and err state 0x00000000 : HSM Return: SUCCESS

Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS

Example: Temukan kunci berdasarkan jenis, pengguna, dan sesi

Perintah ini menemukan kunci AES persisten yang dapat digunakan pengguna saat ini dan pengguna 3. (Pengguna 3 mungkin dapat menggunakan kunci lain yang tidak dapat dilihat pengguna saat ini.)

```
Command: findKey -t 31 -sess 0 -u 3
```

Example: Temukan kunci berdasarkan kelas dan label

Perintah ini menemukan semua kunci publik untuk pengguna saat ini dengan label 2018-sept .

```
Command: findKey -c 2 -l 2018-sept
```

Example: Temukan kunci RSA dengan modulus

Perintah ini menemukan kunci RSA (tipe 0) untuk pengguna saat ini yang dibuat dengan menggunakan modulus di file m4.txt.

```
Command: findKey -t 0 -m m4.txt
```

Parameter

-h

Menampilkan bantuan untuk perintah.

Wajib: Ya

-t

Menemukan kunci dari jenis tertentu. Masukkan konstanta yang mewakili kelas kunci. Sebagai contoh, untuk menemukan kunci 3DES, ketik **-t 21**.

Nilai yang valid:

- 0: [RSA](#)
- 1: [DSA](#)
- 3: [EC](#)
- 16: [GENERIC_SECRET](#)
- 18: [RC4](#)
- 21: [Triple DES \(3DES\)](#)
- 31: [AES](#)

Wajib: Tidak

-c

Menemukan kunci di kelas tertentu. Masukkan konstanta yang mewakili kelas kunci. Misalnya, untuk menemukan kunci publik, ketik -c 2.

Nilai yang valid untuk setiap jenis kunci:

- 2: Publik. Kelas ini berisi kunci publik dari pasangan kunci publik-privat.
- 3: Privat. Kelas ini berisi kunci privat dari pasangan kunci publik-privat.
- 4: Rahasia. Kelas ini berisi semua kunci simetris.

Wajib: Tidak

-l

Menemukan kunci dengan label tertentu. Ketik label yang tepat. Anda tidak dapat menggunakan karakter wildcard atau ekspresi reguler di nilai -l.

Wajib: Tidak

-id

Menemukan kunci dengan ID tertentu. Ketik string ID yang tepat. Anda tidak dapat menggunakan karakter wildcard atau ekspresi reguler di nilai -id.

Wajib: Tidak

-sess

Menemukan kunci berdasarkan status sesi. Untuk menemukan kunci yang valid hanya dalam sesi saat ini, ketik 1. Untuk menemukan bukti kunci persisten, ketik 0.

Wajib: Tidak

-u

Menemukan kunci yang dibagi pengguna tertentu dan pengguna saat ini. Ketik daftar pengguna HSM yang dipisahkan koma IDs, seperti `atau. -u 3 -u 4,7` Untuk menemukan pengguna di HSM, gunakan [ListUsers](#). IDs

Bila Anda menentukan satu ID pengguna, `findKey` mengembalikan kunci untuk pengguna tersebut. Saat Anda menentukan beberapa pengguna IDs, `findKey` mengembalikan kunci yang dapat digunakan oleh semua pengguna tertentu.

Karena `findKey` hanya mengembalikan kunci yang dapat digunakan pengguna saat ini, hasil `-u` selalu identik dengan atau subset dari kunci pengguna saat ini. Untuk mendapatkan semua kunci yang dimiliki oleh atau dibagikan dengan pengguna mana pun, petugas kriptografi (COs) dapat menggunakan [findAllKeys](#) di `cloudhsm_mgmt_util`.

Wajib: Tidak

-m

Menemukan kunci yang dibuat dengan menggunakan modulus RSA dalam file tertentu. Ketik jalur ke file yang menyimpan modulus.

`-m` menentukan file biner yang berisi modulus RSA untuk dicocokkan dengan (opsional).

Wajib: Tidak

-kcv

Menemukan kunci dengan nilai pemeriksaan kunci yang ditentukan.

Nilai pemeriksaan kunci (KCV) adalah hash 3-byte atau checksum dari kunci yang dihasilkan ketika HSM mengimpor atau menghasilkan kunci. Anda juga dapat menghitung KCV di luar HSM, seperti setelah Anda mengeksport kunci. Anda kemudian dapat membandingkan nilai KCV untuk mengonfirmasi identitas dan integritas kunci. Untuk mendapatkan KCV kunci, gunakan [getAttribute](#).

AWS CloudHSM menggunakan metode standar berikut untuk menghasilkan nilai cek kunci:

- Kunci simetris: 3 byte pertama hasil enkripsi blok nol dengan kunci.
- Pasangan kunci asimetris: 3 byte pertama dari hash SHA-1 dari kunci publik.
- Kunci HMAC: KCV untuk kunci HMAC tidak didukung saat ini.

Wajib: Tidak

Output

Output findKey menampilkan jumlah total kunci yang cocok dan handel kuncinya.

```
Command: findKey
Total number of keys present 10

number of keys matched from start index 0::9
6, 7, 8, 9, 10, 11, 262156, 262157, 262158, 262159

Cluster Error Status
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS

Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Topik terkait

- [findSingleKey](#)
- [getKeyInfo](#)
- [getAttribute](#)
- [findAllKeys](#) di cloudhsm_mgmt_util
- [Referensi Atribut Utama](#)

Verifikasi AWS CloudHSM kunci menggunakan KMU

Gunakan findSingleKey perintah di alat AWS CloudHSM key_mgmt_util untuk memverifikasi bahwa kunci ada pada semua modul keamanan perangkat keras (HSM) di cluster. AWS CloudHSM

Sebelum Anda menjalankan perintah key_mgmt_util, Anda harus [memulai key_mgmt_util](#) dan [Masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
findSingleKey -h

findSingleKey -k <key-handle>
```

Contoh

Example

Perintah ini memverifikasi bahwa kunci 252136 ada pada ketiganya HSMs di cluster.

```
Command: findSingleKey -k 252136
Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS

Cluster Error Status
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Parameter

-h

Menampilkan bantuan untuk perintah.

Wajib: Ya

-k

Menentukan handel kunci dari satu kunci di HSM. Parameter ini diperlukan.

Untuk menemukan handel kunci, gunakan perintah [findKey](#).

Wajib: Ya

Topik terkait

- [FindKey](#)
- [getKeyInfo](#)
- [getAttribute](#)

Menghasilkan key pair AWS CloudHSM DSA menggunakan KMU

Gunakan genDSAKeyPair perintah di alat AWS CloudHSM key_mgmt_util untuk menghasilkan key pair [Digital Signing Algorithm \(DSA\)](#) dalam modul keamanan perangkat keras (HSM) Anda.

Anda harus menentukan panjang modulus; perintah menghasilkan nilai modulus. Anda juga dapat menetapkan ID, berbagi kunci dengan pengguna HSM lainnya, membuat kunci yang tidak dapat diekstrak, dan membuat kunci yang kedaluwarsa ketika sesi berakhir. Ketika perintah berhasil, perintah mengembalikan handel kunci yang ditetapkan HSM ke kunci publik dan privat. Anda dapat menggunakan handel kunci untuk mengidentifikasi kunci untuk perintah lain.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna krypto (CU).

Tip

Untuk menemukan atribut kunci yang telah Anda buat, seperti jenis, panjang, label, dan ID, gunakan [getAttribute](#). Untuk menemukan kunci untuk pengguna tertentu, gunakan [getKeyInfo](#). Untuk menemukan kunci berdasarkan nilai atribut mereka, gunakan [findKey](#).

Sintaksis

```
genDSAKeyPair -h

genDSAKeyPair -m <modulus length>
               -l <label>
               [-id <key ID>]
               [-min_srv <minimum number of servers>]
               [-m_value <0..8>]
               [-nex]
               [-sess]
               [-timeout <number of seconds> ]
               [-u <user-ids>]
               [-attest]
```

Contoh

Contoh ini menunjukkan cara menggunakan `genDSAKeyPair` untuk membuat pasangan kunci DSA.

Example: Buat key pair DSA

Perintah ini membuat pasangan kunci DSA dengan label DSA. Output menunjukkan bahwa handel kunci dari kunci publik adalah 19 dan handel kunci privat adalah 21.

```
Command: genDSAKeyPair -m 2048 -l DSA
```

```
Cfm3GenerateKeyPair: returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair:    public key handle: 19    private key handle: 21

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example: Buat key pair DSA khusus sesi

Perintah ini membuat pasangan kunci DSA yang berlaku hanya dalam sesi saat ini. Perintah menetapkan ID unik DSA_temp_pair sebagai tambahan untuk label (non-unik) yang diperlukan. Anda mungkin ingin membuat pasangan kunci seperti ini untuk menandatangani dan memverifikasi token hanya sesi. Output menunjukkan bahwa handel kunci dari kunci publik adalah 12 dan handel kunci privat adalah 14.

```
Command: genDSAKeyPair -m 2048 -l DSA-temp -id DSA_temp_pair -sess

Cfm3GenerateKeyPair: returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair:    public key handle: 12    private key handle: 14

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Untuk mengonfirmasi bahwa pasangan kunci hanya ada dalam sesi, gunakan parameter `-sess` [findKey](#) dengan nilai 1 (benar).

```
Command: findKey -sess 1

Total number of keys present 2

number of keys matched from start index 0::1
12, 14

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Example: Buat key pair DSA bersama yang tidak dapat diekstraksi

Perintah ini membuat pasangan kunci DSA. Kunci privat dibagi dengan tiga pengguna lain, dan tidak dapat diekspor dari HSM. Kunci publik dapat digunakan oleh setiap pengguna dan selalu dapat diekstraksi.

```
Command: genDSAKeyPair -m 2048 -l DSA -id DSA_shared_pair -nex -u 3,5,6

Cfm3GenerateKeyPair: returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair:    public key handle: 11    private key handle: 19

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example: Buat key pair yang dikendalikan kuorum

Perintah ini membuat pasangan kunci DSA dengan label DSA-mV2. Perintah menggunakan parameter `-u` untuk berbagi kunci privat dengan pengguna 4 dan 6. Perintah menggunakan parameter `-m_value` untuk memerlukan kuorum setidaknya dua persetujuan untuk setiap operasi kriptografi yang menggunakan kunci privat. Perintah ini juga menggunakan parameter `-attest` untuk memverifikasi integritas firmware tempat pasangan kunci dihasilkan.

Output menunjukkan bahwa perintah menghasilkan kunci publik dengan handel kunci 12 dan kunci privat dengan handel kunci 17, dan bahwa pemeriksaan pengesahan pada firmware klaster diteruskan.

```
Command: genDSAKeyPair -m 2048 -l DSA-mV2 -m_value 2 -u 4,6 -attest

Cfm3GenerateKeyPair: returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair:    public key handle: 12    private key handle: 17

Attestation Check : [PASS]

Cluster Error Status
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Perintah ini menggunakan [getKeyInfo](#) pada kunci pribadi (pegangan kunci17). Output menegaskan bahwa kunci dimiliki oleh pengguna saat ini (pengguna 3) dan bahwa itu dibagi dengan pengguna 4 dan 6 (dan tidak ada orang lain). Output juga menunjukkan bahwa autentikasi kuorum diaktifkan dan ukuran kuorum adalah dua.

```
Command:  getKeyInfo -k 17

Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

Owned by user 3

also, shared to following 2 user(s):

        4
        6

2 Users need to approve to use/manage this key
```

Parameter

-h

Menampilkan bantuan untuk perintah.

Wajib: Ya

-m

Menentukan panjang modulus dalam bit. Satu-satunya nilai yang valid adalah 2048.

Wajib: Ya

-l

Menentukan label yang ditetapkan pengguna untuk pasangan kunci. Ketik string. Label yang sama berlaku untuk kedua kunci dalam pasangan. Ukuran maksimum yang diijinkan label adalah 127 karakter.

Anda dapat menggunakan frasa apa pun yang membantu Anda mengidentifikasi kunci. Karena label tidak harus unik, Anda dapat menggunakannya pada grup dan mengelompokkan kunci.

Wajib: Ya

-id

Menentukan pengenal yang ditetapkan pengguna untuk pasangan kunci. Ketik string yang unik dalam klaster. Default-nya adalah string kosong. ID yang Anda tentukan berlaku untuk kedua kunci dalam pasangan.

Default: Tidak ada nilai ID.

Wajib: Tidak

-min_srv

Menentukan jumlah minimum HSMs di mana kunci disinkronkan sebelum nilai `-timeout` parameter berakhir. Jika kunci tidak disinkronkan ke jumlah tertentu server dalam waktu yang ditentukan, kunci tidak dibuat.

AWS CloudHSM secara otomatis menyinkronkan setiap kunci ke setiap HSM di cluster. Untuk mempercepat proses Anda, tetapkan nilai kurang dari jumlah HSMs di cluster dan tetapkan nilai batas waktu yang rendah. `min_srv` Namun, perhatikan bahwa beberapa permintaan mungkin tidak menghasilkan kunci.

Default: 1

Wajib: Tidak

-m_value

Menentukan jumlah pengguna yang harus menyetujui operasi kriptografi yang menggunakan kunci privat pada pasangan. Ketik nilai dari 0 sampai 8.

Parameter ini menetapkan persyaratan autentikasi kuorum untuk kunci privat. Nilai default, 0, menonaktifkan fitur autentikasi kuorum untuk kunci. Ketika autentikasi kuorum diaktifkan, pengguna dalam jumlah tertentu harus menandatangani token untuk menyetujui operasi kriptografi yang menggunakan kunci privat, dan operasi yang berbagi atau batal berbagi kunci privat.

Untuk menemukan kunci, gunakan [getKeyInfo](#). `m_value`

Parameter ini hanya valid jika parameter `-u` dalam perintah membagikan pasangan kunci dengan cukup pengguna untuk memenuhi persyaratan `m_value`.

Default: 0

Wajib: Tidak

-nex

Membuat kunci privat yang tidak dapat diekstrak. Kunci privat yang dihasilkan tidak dapat [diekspor dari HSM](#). Kunci publik selalu dapat diekstrak.

Default: Kedua kunci publik dan privat dalam pasangan kunci dapat diekstrak.

Wajib: Tidak

-sess

Membuat kunci yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Gunakan parameter ini ketika Anda memerlukan kunci hanya sebentar, seperti kunci pembungkus yang mengenkripsi, dan kemudian dengan cepat mendekripsi, kunci lain. Jangan gunakan kunci sesi untuk mengenkripsi data yang mungkin perlu Anda dekripsi setelah sesi berakhir.

Untuk mengubah kunci sesi menjadi kunci (token) persisten, gunakan [setAttribute](#).

Default: Kunci persisten.

Wajib: Tidak

-timeout

Menentukan berapa lama (dalam detik) perintah menunggu kunci untuk disinkronkan dengan jumlah yang HSMs ditentukan oleh parameter. `min_srv`

Parameter ini hanya valid jika parameter `min_srv` juga digunakan dalam perintah.

Default: Tidak ada waktu habis. Perintah menunggu tanpa batas waktu dan kembali hanya ketika kunci disinkronkan ke jumlah minimum server.

Wajib: Tidak

-u

Berbagi kunci privat pada pasangan dengan pengguna tertentu. Parameter ini memberikan izin kepada pengguna krypto HSM lainnya (CUs) untuk menggunakan kunci pribadi dalam operasi kriptografi. Kunci publik dapat digunakan oleh pengguna tanpa berbagi.

Ketik daftar pengguna HSM yang dipisahkan koma IDs, seperti -. u 5,6 Jangan sertakan ID pengguna HSM dari pengguna saat ini. [Untuk menemukan pengguna IDs HSM CUs di HSM, gunakan ListUsers](#). Untuk berbagi atau batal berbagi kunci yang ada, gunakan [shareKey](#) di `cloudhsm_mgmt_util`.

Default: Hanya pengguna saat ini dapat menggunakan kunci privat.

Wajib: Tidak

-attest

Menjalankan pemeriksaan integritas yang memverifikasi bahwa firmware tempat kluster berjalan belum dirusak.

Default: Tidak ada pemeriksaan pengesahan.

Wajib: Tidak

Topik terkait

- [RSAKeypasangan gen](#)
- [genSymKey](#)
- [ECCKeypasangan gen](#)

Menghasilkan key pair AWS CloudHSM ECC menggunakan KMU

Gunakan `genECCKeypair` perintah di alat AWS CloudHSM `key_mgmt_util` untuk menghasilkan key pair [Elliptic Curve Cryptography \(ECC\) di modul keamanan perangkat keras \(HSM\)](#) Anda. Ketika menjalankan perintah `genECCKeypair`, Anda harus menentukan pengenalan kurva elips dan label untuk pasangan kunci. Anda juga dapat berbagi kunci privat dengan pengguna CU lainnya, membuat kunci yang tidak dapat diekstrak, kunci dikendalikan kuorum, dan kunci yang kedaluwarsa ketika sesi berakhir. Ketika perintah berhasil, perintah mengembalikan handle kunci yang HSM tetapkan ke kunci ECC publik dan privat. Anda dapat menggunakan handle kunci untuk mengidentifikasi kunci untuk perintah lain.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna krypto (CU).

Tip

Untuk menemukan atribut kunci yang telah Anda buat, seperti jenis, panjang, label, dan ID, gunakan [getAttribute](#). Untuk menemukan kunci untuk pengguna tertentu, gunakan [getKeyInfo](#). Untuk menemukan kunci berdasarkan nilai atribut mereka, gunakan [findKey](#).

Sintaksis

```
genECCKeypair -h

genECCKeypair -i <EC curve id>
                -l <label>
                [-id <key ID>]
                [-min_srv <minimum number of servers>]
                [-m_value <0..8>]
                [-nex]
                [-sess]
                [-timeout <number of seconds> ]
                [-u <user-ids>]
                [-attest]
```

Contoh

Contoh berikut menunjukkan cara menggunakan `genECCKeypair` untuk membuat pasangan kunci ECC di Anda HSMs.

Example: Membuat dan memeriksa sebuah key pair ECC

Perintah ini menggunakan kurva eliptik `NID_Secp384R1` dan label untuk membuat key pair ECC. `ecc14` Output menunjukkan bahwa handel kunci dari kunci privat adalah 262177 dan handel kunci dari kunci publik adalah 262179. Label berlaku untuk kunci publik dan privat.

Command: **genECCKeypair -i 14 -l ecc14**

```
Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS
```

```
Cfm3GenerateKeyPair:    public key handle: 262179    private key handle: 262177
```

```
Cluster Error Status
```

```
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Setelah menghasilkan kunci, Anda dapat memeriksa atributnya. Gunakan [getAttribute](#) untuk menulis semua atribut (diwakili oleh konstanta 512) dari kunci privat ECC baru ke file `attr_262177`.

Command: **getAttribute -o 262177 -a 512 -out attr_262177**
got all attributes of size 529 attr cnt 19

```
Attributes dumped into attr_262177
```

```
Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS
```

Kemudian gunakan perintah `cat` untuk melihat isi file atribut `attr_262177`. Output menunjukkan kunci adalah kunci privat kurva elips yang dapat digunakan untuk penandatanganan, tetapi tidak untuk mengenkripsi, mendekripsi, membungkus, membuka bungkus, atau memverifikasi. Kunci bersifat persisten dan dapat diekspor.

```
$ cat attr_262177

OBJ_ATTR_CLASS
0x03
OBJ_ATTR_KEY_TYPE
0x03
OBJ_ATTR_TOKEN
0x01
OBJ_ATTR_PRIVATE
0x01
OBJ_ATTR_ENCRYPT
0x00
OBJ_ATTR_DECRYPT
0x00
OBJ_ATTR_WRAP
0x00
OBJ_ATTR_UNWRAP
0x00
OBJ_ATTR_SIGN
0x01
OBJ_ATTR_VERIFY
0x00
OBJ_ATTR_LOCAL
0x01
OBJ_ATTR_SENSITIVE
0x01
OBJ_ATTR_EXTRACTABLE
0x01
OBJ_ATTR_LABEL
ecc2
OBJ_ATTR_ID

OBJ_ATTR_VALUE_LEN
0x0000008a
```

```

OBJ_ATTR_KCV
0xbbb32a
OBJ_ATTR_MODULUS
044a0f9d01d10f7437d9fa20995f0cc742552e5ba16d3d7e9a65a33e20ad3e569e68eb62477a9960a87911e6121d112
OBJ_ATTR_MODULUS_BITS
0x0000019f

```

Example Menggunakan kurva EEC yang tidak valid

Perintah ini mencoba untuk membuat pasangan kunci ECC dengan menggunakan kurva NID_X9_62_prime192v1. Karena kurva elips ini tidak berlaku untuk mode FIPS HSMs, perintah gagal. Pesan melaporkan bahwa server di cluster tidak tersedia, tetapi ini biasanya tidak menunjukkan masalah dengan HSMs di cluster.

Command: **genECCKeyPair -i 1 -l ecc1**

```

Cfm3GenerateKeyPair returned: 0xb3 : HSM Error: This operation violates the
current configured/FIPS policies

```

```

Cluster Error Status

```

```

Node id 0 and err state 0x30000085 : HSM CLUSTER ERROR: Server in cluster is
unavailable

```

Parameter

-h

Menampilkan bantuan untuk perintah.

Wajib: Ya

-i

Menentukan pengenalan untuk kurva elips. Masukkan pengenalan.

Nilai yang valid:

- 2: NID_X9_62_prime256v1
- 14: NID_secp384r1
- 16: NID_secp256k1

Wajib: Ya

-l

Menentukan label yang ditetapkan pengguna untuk pasangan kunci. Ketik string. Label yang sama berlaku untuk kedua kunci dalam pasangan. Ukuran maksimum yang diijinkan label adalah 127 karakter.

Anda dapat menggunakan frasa apa pun yang membantu Anda mengidentifikasi kunci. Karena label tidak harus unik, Anda dapat menggunakannya pada grup dan mengelompokkan kunci.

Wajib: Ya

-id

Menentukan pengenal yang ditetapkan pengguna untuk pasangan kunci. Ketik string yang unik dalam klaster. Default-nya adalah string kosong. ID yang Anda tentukan berlaku untuk kedua kunci dalam pasangan.

Default: Tidak ada nilai ID.

Wajib: Tidak

-min_srv

Menentukan jumlah minimum HSMs di mana kunci disinkronkan sebelum nilai `-timeout` parameter berakhir. Jika kunci tidak disinkronkan ke jumlah tertentu server dalam waktu yang ditentukan, kunci tidak dibuat.

AWS CloudHSM secara otomatis menyinkronkan setiap kunci ke setiap HSM di cluster. Untuk mempercepat proses Anda, tetapkan nilai kurang dari jumlah HSMs di cluster dan tetapkan nilai batas waktu yang rendah. `min_srv` Namun, perhatikan bahwa beberapa permintaan mungkin tidak menghasilkan kunci.

Default: 1

Wajib: Tidak

-m_value

Menentukan jumlah pengguna yang harus menyetujui operasi kriptografi yang menggunakan kunci privat pada pasangan. Ketik nilai dari 0 sampai 8.

Parameter ini menetapkan persyaratan autentikasi kuorum untuk kunci privat. Nilai default, 0, menonaktifkan fitur autentikasi kuorum untuk kunci. Ketika autentikasi kuorum diaktifkan,

pengguna dalam jumlah tertentu harus menandatangani token untuk menyetujui operasi kriptografi yang menggunakan kunci privat, dan operasi yang berbagi atau batal berbagi kunci privat.

Untuk menemukan kunci, gunakan [getKeyInfo](#). `m_value`

Parameter ini hanya valid jika parameter `-u` dalam perintah membagikan pasangan kunci dengan cukup pengguna untuk memenuhi persyaratan `m_value`.

Default: 0

Wajib: Tidak

`-nex`

Membuat kunci privat yang tidak dapat diekstrak. Kunci privat yang dihasilkan tidak dapat [diekspor dari HSM](#). Kunci publik selalu dapat diekstrak.

Default: Kedua kunci publik dan privat dalam pasangan kunci dapat diekstrak.

Wajib: Tidak

`-sess`

Membuat kunci yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Gunakan parameter ini ketika Anda memerlukan kunci hanya sebentar, seperti kunci pembungkus yang mengenkripsi, dan kemudian dengan cepat mendekripsi, kunci lain. Jangan gunakan kunci sesi untuk mengenkripsi data yang mungkin perlu Anda dekripsi setelah sesi berakhir.

Untuk mengubah kunci sesi menjadi kunci (token) persisten, gunakan [setAttribute](#).

Default: Kunci persisten.

Wajib: Tidak

`-timeout`

Menentukan berapa lama (dalam detik) perintah menunggu kunci untuk disinkronkan dengan jumlah yang HSMs ditentukan oleh parameter. `min_srv`

Parameter ini hanya valid jika parameter `min_srv` juga digunakan dalam perintah.

Default: Tidak ada waktu habis. Perintah menunggu tanpa batas waktu dan kembali hanya ketika kunci disinkronkan ke jumlah minimum server.

Wajib: Tidak

-u

Berbagi kunci privat pada pasangan dengan pengguna tertentu. Parameter ini memberikan izin kepada pengguna krypto HSM lainnya (CUs) untuk menggunakan kunci pribadi dalam operasi kriptografi. Kunci publik dapat digunakan oleh pengguna tanpa berbagi.

Ketik daftar pengguna HSM yang dipisahkan koma IDs, seperti -. u 5,6 Jangan sertakan ID pengguna HSM dari pengguna saat ini. [Untuk menemukan pengguna IDs HSM CUs di HSM, gunakan ListUsers.](#) Untuk berbagi atau batal berbagi kunci yang ada, gunakan [shareKey](#) di `cloudhsm_mgmt_util`.

Default: Hanya pengguna saat ini dapat menggunakan kunci privat.

Wajib: Tidak

-attest

Menjalankan pemeriksaan integritas yang memverifikasi bahwa firmware tempat kluster berjalan belum dirusak.

Default: Tidak ada pemeriksaan pengesahan.

Wajib: Tidak

Topik terkait

- [genSymKey](#)
- [RSAKeypasangan gen](#)
- [DSAKeypasangan gen](#)

Menghasilkan key pair AWS CloudHSM RSA menggunakan KMU

[Gunakan genRSAKeyPair perintah di alat AWS CloudHSM key_mgmt_util untuk menghasilkan key pair asimetris RSA.](#) Anda menentukan jenis kunci, panjang modulus, dan eksponen publik.

Perintah menghasilkan modulus dari panjang yang ditentukan dan membuat pasangan kunci. Anda dapat menetapkan ID, berbagi kunci dengan pengguna HSM lainnya, membuat kunci yang tidak dapat diekstrak dan kunci yang kedaluwarsa ketika sesi berakhir. Ketika perintah berhasil, perintah mengembalikan handle kunci yang HSM tetapkan ke kunci. Anda dapat menggunakan handle kunci untuk mengidentifikasi kunci untuk perintah lain.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai `key\_mgmt\_util`](#) dan [masuk](#) ke HSM sebagai pengguna krypto (CU).

Tip

Untuk menemukan atribut kunci yang telah Anda buat, seperti jenis, panjang, label, dan ID, gunakan [getAttribute](#). Untuk menemukan kunci untuk pengguna tertentu, gunakan [getKeyInfo](#). Untuk menemukan kunci berdasarkan nilai atribut mereka, gunakan [findKey](#).

Sintaksis

```
genRSAKeyPair -h

genRSAKeyPair -m <modulus length>
               -e <public exponent>
               -l <label>
               [-id <key ID>]
               [-min_srv <minimum number of servers>]
               [-m_value <0..8>]
               [-nex]
               [-sess]
               [-timeout <number of seconds> ]
               [-u <user-ids>]
               [-attest]
```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `genRSAKeyPair` untuk membuat pasangan kunci asimetris di Anda HSMs.

Example: Buat dan periksa key pair RSA

Perintah ini menciptakan key pair RSA dengan modulus 2048-bit dan eksponen 65537. Output menunjukkan bahwa handle kunci publik adalah 2100177 dan handle kunci privat adalah 2100426.

Command: **genRSAKeyPair -m 2048 -e 65537 -l rsa_test**

Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair: public key handle: 2100177 private key handle:
2100426

Cluster Status:

Node id 0 status: 0x00000000 : HSM Return: SUCCESS

Node id 1 status: 0x00000000 : HSM Return: SUCCESS

Perintah berikutnya menggunakan [getAttribute](#) untuk mendapatkan atribut kunci publik yang baru saja kita buat. Perintah ini menulis output ke file `attr_2100177`. Hal ini diikuti oleh perintah `cat` yang mendapat isi dari file atribut. Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

Nilai heksadesimal yang dihasilkan mengonfirmasi bahwa itu adalah kunci publik (OBJ_ATTR_CLASS 0x02) dengan jenis RSA (OBJ_ATTR_KEY_TYPE 0x00). Anda dapat menggunakan kunci publik ini untuk mengenkripsi (OBJ_ATTR_ENCRYPT 0x01), tetapi tidak untuk mendekripsi (OBJ_ATTR_DECRYPT 0x00). Hasilnya juga mencakup panjang kunci (512, 0x200), modulus, panjang modulus (2048, 0x800), dan eksponen publik (65537, 0x10001).

```
Command: getAttribute -o 2100177 -a 512 -out attr_2100177
```

```
Attribute size: 801, count: 26
```

```
Written to: attr_2100177 file
```

```
Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS
```

```
$ cat attr_2100177
```

```
OBJ_ATTR_CLASS
```

```
0x02
```

```
OBJ_ATTR_KEY_TYPE
```

```
0x00
```

```
OBJ_ATTR_TOKEN
```

```
0x01
```

```
OBJ_ATTR_PRIVATE
```

```
0x01
```

```
OBJ_ATTR_ENCRYPT
```

```
0x01
```

```
OBJ_ATTR_DECRYPT
```

```
0x00
```

```
OBJ_ATTR_WRAP
```

```
0x01
```

```
OBJ_ATTR_UNWRAP
```

```
0x00
```

```
OBJ_ATTR_SIGN
```

```
0x00
```

```
OBJ_ATTR_VERIFY
0x01
OBJ_ATTR_LOCAL
0x01
OBJ_ATTR_SENSITIVE
0x00
OBJ_ATTR_EXTRACTABLE
0x01
OBJ_ATTR_LABEL
rsa_test
OBJ_ATTR_ID

OBJ_ATTR_VALUE_LEN
0x00000200
OBJ_ATTR_KCV
0xc51c18
OBJ_ATTR_MODULUS
0xbb9301cc362c1d9724eb93da8adab0364296bde7124a241087d9436b9be57e4f7780040df03c2c
1c0fe6e3b61aa83c205280119452868f66541bbbfacbbe787b8284fc81deaeef2b8ec0ba25a077d
6983c77a1de7b17cbe8e15b203868704c6452c2810344a7f2736012424cf0703cf15a37183a1d2d0
97240829f8f90b063dd3a41171402b162578d581980976653935431da0c1260bfe756d85dca63857
d9f27a541676cb9c7def0ef6a2a89c9b9304bcac16fdf8183c0a555421f9ad5dfefb534cf26b65873
970cdf1a07484f1c128b53e10209cc6f7ac308669112968c81a5de408e7f644fe58b1a9ae128fec
b3e4203294a96fae06f8f0db7982cb5d7f
OBJ_ATTR_MODULUS_BITS
0x00000800
OBJ_ATTR_PUBLIC_EXPONENT
0x010001
OBJ_ATTR_TRUSTED
0x00
OBJ_ATTR_WRAP_WITH_TRUSTED
0x00
OBJ_ATTR_DESTROYABLE
0x01
OBJ_ATTR_DERIVE
0x00
OBJ_ATTR_ALWAYS_SENSITIVE
0x00
OBJ_ATTR_NEVER_EXTRACTABLE
0x00
```

Example: Buat key pair RSA bersama

Perintah ini menghasilkan pasangan kunci RSA dan berbagi kunci privat dengan pengguna 4, CU lain pada HSM. Perintah menggunakan parameter `m_value` untuk mengharuskan setidaknya dua persetujuan sebelum kunci privat dalam pasangan dapat digunakan dalam operasi kriptografi. Ketika Anda menggunakan parameter `m_value`, Anda juga harus menggunakan `-u` dalam perintah dan `m_value` tidak dapat melebihi jumlah total pengguna (jumlah nilai dalam `-u` + pemilik).

```
Command: genRSAKeyPair -m 2048 -e 65537 -l rsa_mofn -id rsa_mv2 -u 4 -m_value 2
```

```
Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS
```

```
Cfm3GenerateKeyPair:      public key handle: 27      private key handle: 28
```

```
Cluster Error Status
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

Parameter

`-h`

Menampilkan bantuan untuk perintah.

Wajib: Ya

`-m`

Menentukan panjang modulus dalam bit. Nilai minimum adalah 2048.

Wajib: Ya

`-e`

Menentukan eksponen publik. Nilai harus angka ganjil yang lebih besar dari atau sama dengan 65537.

Wajib: Ya

`-l`

Menentukan label yang ditetapkan pengguna untuk pasangan kunci. Ketik string. Label yang sama berlaku untuk kedua kunci dalam pasangan. Ukuran maksimum yang diijinkan `label` adalah 127 karakter.

Anda dapat menggunakan frasa apa pun yang membantu Anda mengidentifikasi kunci. Karena label tidak harus unik, Anda dapat menggunakannya pada grup dan mengelompokkan kunci.

Wajib: Ya

-id

Menentukan pengenalan yang ditetapkan pengguna untuk pasangan kunci. Ketik string yang unik dalam kluster. Default-nya adalah string kosong. ID yang Anda tentukan berlaku untuk kedua kunci dalam pasangan.

Default: Tidak ada nilai ID.

Wajib: Tidak

-min_srv

Menentukan jumlah minimum HSMs di mana kunci disinkronkan sebelum nilai `-timeout` parameter berakhir. Jika kunci tidak disinkronkan ke jumlah tertentu server dalam waktu yang ditentukan, kunci tidak dibuat.

AWS CloudHSM secara otomatis menyinkronkan setiap kunci ke setiap HSM di cluster. Untuk mempercepat proses Anda, tetapkan nilai kurang dari jumlah HSMs di cluster dan tetapkan nilai batas waktu yang rendah. `min_srv` Namun, perhatikan bahwa beberapa permintaan mungkin tidak menghasilkan kunci.

Default: 1

Wajib: Tidak

-m_value

Menentukan jumlah pengguna yang harus menyetujui operasi kriptografi yang menggunakan kunci privat pada pasangan. Ketik nilai dari 0 sampai 8.

Parameter ini menetapkan persyaratan autentikasi kuorum untuk kunci privat. Nilai default, 0, menonaktifkan fitur autentikasi kuorum untuk kunci. Ketika autentikasi kuorum diaktifkan, pengguna dalam jumlah tertentu harus menandatangani token untuk menyetujui operasi kriptografi yang menggunakan kunci privat, dan operasi yang berbagi atau batal berbagi kunci privat.

Untuk menemukan kunci, gunakan [getKeyInfo](#). `m_value`

Parameter ini hanya valid jika parameter `-u` dalam perintah membagikan pasangan kunci dengan cukup pengguna untuk memenuhi persyaratan `m_value`.

Default: 0

Wajib: Tidak

`-nex`

Membuat kunci privat yang tidak dapat diekstrak. Kunci privat yang dihasilkan tidak dapat [diekspor dari HSM](#). Kunci publik selalu dapat diekstrak.

Default: Kedua kunci publik dan privat dalam pasangan kunci dapat diekstrak.

Wajib: Tidak

`-sess`

Membuat kunci yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Gunakan parameter ini ketika Anda memerlukan kunci hanya sebentar, seperti kunci pembungkus yang mengenkripsi, dan kemudian dengan cepat mendekripsi, kunci lain. Jangan gunakan kunci sesi untuk mengenkripsi data yang mungkin perlu Anda dekripsi setelah sesi berakhir.

Untuk mengubah kunci sesi menjadi kunci (token) persisten, gunakan [setAttribute](#).

Default: Kunci persisten.

Wajib: Tidak

`-timeout`

Menentukan berapa lama (dalam detik) perintah menunggu kunci untuk disinkronkan dengan jumlah yang HSMs ditentukan oleh parameter `min_srv`

Parameter ini hanya valid jika parameter `min_srv` juga digunakan dalam perintah.

Default: Tidak ada waktu habis. Perintah menunggu tanpa batas waktu dan kembali hanya ketika kunci disinkronkan ke jumlah minimum server.

Wajib: Tidak

`-u`

Berbagi kunci privat pada pasangan dengan pengguna tertentu. Parameter ini memberikan izin kepada pengguna krypto HSM lainnya (CUs) untuk menggunakan kunci pribadi dalam operasi kriptografi. Kunci publik dapat digunakan oleh pengguna tanpa berbagi.

Ketik daftar pengguna HSM yang dipisahkan koma IDs, seperti -. u 5,6 Jangan sertakan ID pengguna HSM dari pengguna saat ini. [Untuk menemukan pengguna IDs HSM CUs di HSM, gunakan ListUsers](#). Untuk berbagi atau batal berbagi kunci yang ada, gunakan [shareKey](#) di `cloudhsm_mgmt_util`.

Default: Hanya pengguna saat ini dapat menggunakan kunci privat.

Wajib: Tidak

-attest

Menjalankan pemeriksaan integritas yang memverifikasi bahwa firmware tempat kluster berjalan belum dirusak.

Default: Tidak ada pemeriksaan pengesahan.

Wajib: Tidak

Topik terkait

- [genSymKey](#)
- [DSAKeypasangan gen](#)
- [ECCKeypasangan gen](#)

Hasilkan kunci AWS CloudHSM simetris menggunakan KMU

Gunakan `genSymKey` perintah di alat AWS CloudHSM `key_mgmt_util` untuk menghasilkan kunci simetris dalam modul keamanan perangkat keras (HSM) Anda. Anda dapat menentukan jenis dan ukuran kunci, menetapkan ID dan label, dan berbagi kunci dengan pengguna HSM lainnya. Anda juga dapat membuat kunci yang tidak dapat diekstrak dan kunci yang kedaluwarsa ketika sesi berakhir. Ketika perintah berhasil, perintah mengembalikan handel kunci yang HSM tetapkan ke kunci. Anda dapat menggunakan handel kunci untuk mengidentifikasi kunci untuk perintah lain.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
genSymKey -h
```

```
genSymKey -t <key-type>
          -s <key-size>
          -l <label>
          [-id <key-ID>]
          [-min_srv <minimum-number-of-servers>]
          [-m_value <0..8>]
          [-nex]
          [-sess]
          [-timeout <number-of-seconds> ]
          [-u <user-ids>]
          [-attest]
```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan genSymKey untuk membuat kunci simetris di Anda HSMs.

Tip

Untuk menggunakan kunci yang Anda buat dengan contoh-contoh ini untuk operasi HMAC, Anda harus mengatur OBJ_ATTR_SIGN dan OBJ_ATTR_VERIFY menjadi TRUE setelah Anda menghasilkan kunci. Untuk menetapkan nilai-nilai ini, gunakan setAttribute di Utilitas Manajemen CloudHSM (CMU). Untuk informasi lebih lanjut, lihat [setAttribute](#).

Example: Hasilkan kunci AES

Perintah ini membuat kunci AES 256-bit dengan label aes256. Output menunjukkan bahwa handle kunci dari kunci baru adalah 6.

Command: **genSymKey -t 31 -s 32 -l aes256**

Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

Symmetric Key Created. Key Handle: 6

Cluster Error Status

Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

Example: Buat kunci sesi

Perintah ini membuat kunci AES 192-bit yang tidak dapat diekstrak yang valid hanya dalam sesi saat ini. Anda mungkin perlu membuat kunci seperti ini untuk membungkus (dan kemudian segera membuka) kunci yang sedang diekspor.

```
Command: genSymKey -t 31 -s 24 -l tmpAES -id wrap01 -nex -sess
```

Example: Kembali dengan cepat

Perintah ini membuat kunci 512-byte generik dengan label `IT_test_key`. Perintah tidak menunggu kunci disinkronkan ke semua HSMs di cluster. Sebaliknya, perintah kembali segera setelah kunci dibuat pada salah satu HSM (`-min_srv 1`) atau dalam 1 detik (`-timeout 1`), mana yang lebih pendek. Jika kunci tidak disinkronkan dengan jumlah minimum yang ditentukan HSMs sebelum batas waktu berakhir, itu tidak dihasilkan. Anda mungkin ingin menggunakan perintah seperti ini dalam skrip yang membuat banyak kunci, seperti `for` pada contoh berikut.

```
Command: genSymKey -t 16 -s 512 -l IT_test_key -min_srv 1 -timeout 1

$ for i in {1..30};
  do /opt/cloudhsm/bin/key_mgmt_util singlecmd loginHSM -u CU -s example_user -p
example_pwd genSymKey -l aes -t 31 -s 32 -min_srv 1 -timeout 1;
done;
```

Example: Buat kunci generik resmi kuorum

Perintah ini membuat kunci rahasia generik 2048-bit dengan label `generic-mV2`. Perintah menggunakan parameter `-u` untuk berbagi kunci dengan CU lain, pengguna 6. Perintah menggunakan parameter `-m_value` untuk mengharuskan kuorum setidaknya dua persetujuan untuk setiap operasi kriptografi yang menggunakan kunci. Perintah ini juga menggunakan parameter `-attest` untuk memverifikasi integritas firmware tempat kunci dihasilkan.

Output menunjukkan bahwa perintah menghasilkan kunci dengan handel kunci 9 dan bahwa pemeriksaan pengesahan pada firmware klaster telah dilewatkan.

```
Command: genSymKey -t 16 -s 2048 -l generic-mV2 -m_value 2 -u 6 -
attest
```



```
Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

Symmetric Key Created.  Key Handle: 9

Attestation Check : [PASS]

Cluster Error Status
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example: Buat dan periksa kunci

Perintah ini membuat kunci Triple DES dengan label 3DES_shared dan ID dari IT-02. Kuncinya bisa digunakan oleh pengguna saat ini, dan pengguna 4 dan 5. Perintah gagal jika ID tidak unik dalam klaster atau jika pengguna saat ini adalah pengguna 4 atau 5.

Output menunjukkan bahwa kunci baru memiliki handel kunci 7.

```
Command: genSymKey -t 21 -s 24 -l 3DES_shared -id IT-02 -u 4,5

Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

Symmetric Key Created.  Key Handle: 7

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Untuk memverifikasi bahwa kunci 3DES baru dimiliki oleh pengguna saat ini dan dibagikan dengan pengguna 4 dan 5, gunakan [getKeyInfo](#). Perintah menggunakan handel yang ditetapkan ke kunci baru (Key Handle: 7).

Output menunjukkan bahwa kunci tersebut dimiliki oleh pengguna 3 dan berbagi dengan pengguna 4 dan 5.

```
Command: getKeyInfo -k 7

Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

Owned by user 3

also, shared to following 2 user(s):
```

4, 5

Untuk mengonfirmasi properti lain dari kunci, gunakan [getAttribute](#). Perintah pertama menggunakan `getAttribute` untuk mendapatkan semua atribut (`-a 512`) dari handle kunci 7 (`-o 7`). Perintah menuliskannya ke file `attr_7`. Perintah kedua menggunakan `cat` untuk mendapatkan konten file `attr_7`.

Perintah ini mengonfirmasi bahwa kunci 7 adalah 192-bit (`OBJ_ATTR_VALUE_LEN 0x00000018` atau 24-byte) 3DES (`OBJ_ATTR_KEY_TYPE 0x15`) kunci simetris (`OBJ_ATTR_CLASS 0x04`) dengan label `3DES_shared` (`OBJ_ATTR_LABEL 3DES_shared`) dan ID dari `IT_02` (`OBJ_ATTR_ID IT-02`). Kunci persisten (`OBJ_ATTR_TOKEN 0x01`) dan dapat diekstrak (`OBJ_ATTR_EXTRACTABLE 0x01`) dan dapat digunakan untuk enkripsi, dekripsi, dan pembungkus.

 Tip

Untuk menemukan atribut kunci yang telah Anda buat, seperti jenis, panjang, label, dan ID, gunakan [getAttribute](#). Untuk menemukan kunci untuk pengguna tertentu, gunakan [getKeyInfo](#). Untuk menemukan kunci berdasarkan nilai atribut mereka, gunakan [findKey](#).

Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

```
Command: getAttribute -o 7 -a 512 -out attr_7
```

```
got all attributes of size 444 attr cnt 17
Attributes dumped into attr_7 file
```

```
Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS
```

```
$ cat attr_7
```

```
OBJ_ATTR_CLASS
0x04
OBJ_ATTR_KEY_TYPE
0x15
OBJ_ATTR_TOKEN
0x01
OBJ_ATTR_PRIVATE
0x01
```

```
OBJ_ATTR_ENCRYPT
0x01
OBJ_ATTR_DECRYPT
0x01
OBJ_ATTR_WRAP
0x00
OBJ_ATTR_UNWRAP
0x00
OBJ_ATTR_SIGN
0x00
OBJ_ATTR_VERIFY
0x00
OBJ_ATTR_LOCAL
0x01
OBJ_ATTR_SENSITIVE
0x01
OBJ_ATTR_EXTRACTABLE
0x01
OBJ_ATTR_LABEL
3DES_shared
OBJ_ATTR_ID
IT-02
OBJ_ATTR_VALUE_LEN
0x00000018
OBJ_ATTR_KCV
0x59a46e
```

 Tip

Untuk menggunakan kunci yang Anda buat dengan contoh-contoh ini untuk operasi HMAC, Anda harus mengatur OBJ_ATTR_SIGN dan OBJ_ATTR_VERIFY menjadi TRUE setelah Anda menghasilkan kunci. Untuk menetapkan nilai-nilai ini, gunakan `setAttribute` di CMU. Untuk informasi lebih lanjut, lihat [setAttribute](#).

Parameter

-h

Menampilkan bantuan untuk perintah.

Wajib: Ya

-t

Menentukan jenis kunci simetris. Masukkan konstanta yang mewakili jenis kunci. Misalnya, untuk membuat kunci AES, ketik -t 31.

Nilai yang valid:

- 16: [GENERIC_SECRET](#). Kunci rahasia generik adalah array byte yang tidak sesuai dengan standar tertentu, seperti persyaratan untuk kunci AES.
- 18: [RC4](#). RC4 kunci tidak valid pada mode FIPS HSMs
- 21: [Tiga DES \(3DES\)](#). Sesuai dengan panduan NIST, ini tidak diizinkan untuk cluster dalam mode FIPS setelah 2023. Untuk cluster dalam mode non-FIPS, masih diperbolehkan setelah 2023. Lihat [Kepatuhan FIPS 140: Penutupan Mekanisme 2024](#) untuk detail.
- 31: [AES](#)

Wajib: Ya

-s

Menentukan ukuran kunci dalam byte. Misalnya, untuk membuat kunci 192-bit, ketik 24.

Nilai yang valid untuk setiap jenis kunci:

- AES: 16 (128 bit), 24 (192 bit), 32 (256 bit)
- 3DES: 24 (192 bit)
- Rahasia Generik: <3584 (28672 bit)

Wajib: Ya

-l

Menentukan label yang ditetapkan pengguna untuk kunci. Ketik string.

Anda dapat menggunakan frasa apa pun yang membantu Anda mengidentifikasi kunci. Karena label tidak harus unik, Anda dapat menggunakannya pada grup dan mengelompokkan kunci.

Wajib: Ya

-attest

Menjalankan pemeriksaan integritas yang memverifikasi bahwa firmware tempat kluster berjalan belum dirusak.

Default: Tidak ada pemeriksaan pengesahan.

Wajib: Tidak

-id

Menentukan pengenal yang ditetapkan pengguna untuk kunci. Ketik string yang unik dalam kluster. Default-nya adalah string kosong.

Default: Tidak ada nilai ID.

Wajib: Tidak

-min_srv

Menentukan jumlah minimum HSMs di mana kunci disinkronkan sebelum nilai `-timeout` parameter berakhir. Jika kunci tidak disinkronkan ke jumlah tertentu server dalam waktu yang ditentukan, kunci tidak dibuat.

AWS CloudHSM secara otomatis menyinkronkan setiap kunci ke setiap HSM di cluster. Untuk mempercepat proses Anda, tetapkan nilai kurang dari jumlah HSMs di cluster dan tetapkan nilai batas waktu yang rendah. `min_srv` Namun, perhatikan bahwa beberapa permintaan mungkin tidak menghasilkan kunci.

Default: 1

Wajib: Tidak

-m_value

Menentukan jumlah pengguna yang harus menyetujui operasi kriptografi yang menggunakan kunci. Ketik nilai dari 0 sampai 8.

Parameter ini menetapkan persyaratan autentikasi kuorum untuk kunci. Nilai default, 0, menonaktifkan fitur autentikasi kuorum untuk kunci. Ketika autentikasi kuorum diaktifkan, jumlah tertentu pengguna harus menandatangani token untuk menyetujui operasi kriptografi yang menggunakan kunci, dan operasi yang berbagi atau batal berbagi kunci.

Untuk menemukan kunci, gunakan [getKeyInfo](#). `m_value`

Parameter ini hanya valid jika parameter `-u` dalam perintah membagikan kunci dengan cukup pengguna untuk memenuhi persyaratan `m_value`.

Default: 0

Wajib: Tidak

-nex

Membuat kunci tidak dapat diekstrak. Kunci yang dihasilkan tidak dapat [diekspor dari HSM](#).

Default: Kunci dapat diekstrak.

Wajib: Tidak

-sess

Membuat kunci yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Gunakan parameter ini ketika Anda memerlukan kunci hanya sebentar, seperti kunci pembungkus yang mengenkripsi, dan kemudian dengan cepat mendekripsi, kunci lain. Jangan gunakan kunci sesi untuk mengenkripsi data yang mungkin perlu Anda dekripsi setelah sesi berakhir.

Untuk mengubah kunci sesi menjadi kunci (token) persisten, gunakan [setAttribute](#).

Default: Kunci persisten.

Wajib: Tidak

-timeout

Menentukan berapa lama (dalam detik) perintah menunggu kunci untuk disinkronkan dengan jumlah yang HSMs ditentukan oleh parameter. `min_srv`

Parameter ini hanya valid jika parameter `min_srv` juga digunakan dalam perintah.

Default: Tidak ada waktu habis. Perintah menunggu tanpa batas waktu dan kembali hanya ketika kunci disinkronkan ke jumlah minimum server.

Wajib: Tidak

-u

Berbagi kunci dengan pengguna tertentu. Parameter ini memberikan izin kepada pengguna kriptografi HSM lainnya (CUs) untuk menggunakan kunci ini dalam operasi kriptografi.

Ketik daftar pengguna HSM yang dipisahkan koma IDs, seperti `-u 5,6` Jangan sertakan ID pengguna HSM dari pengguna saat ini. [Untuk menemukan pengguna IDs HSM CUs di HSM, gunakan ListUsers](#). Untuk berbagi atau batal berbagi kunci yang ada, gunakan [shareKey](#) di `cloudhsm_mgmt_util`.

Default: Hanya pengguna saat ini dapat menggunakan kunci privat.

Wajib: Tidak

Topik terkait

- [exSymKey](#)
- [RSAKeypasangan gen](#)
- [DSAKeypasangan gen](#)
- [ECCKeypasangan gen](#)
- [setAttribute](#)

Dapatkan atribut AWS CloudHSM kunci menggunakan KMU

Gunakan `getAttribute` perintah di AWS CloudHSM `key_mgmt_util` untuk menulis satu atau semua nilai atribut untuk kunci ke file. AWS CloudHSM Jika atribut yang Anda tentukan tidak ada untuk jenis kunci, seperti modulus kunci AES, `getAttribute` mengembalikan kesalahan.

Atribut Kunci adalah properti dari kunci. Atribut termasuk karakteristik, seperti jenis kunci, kelas, label, dan ID, dan nilai-nilai yang mewakili tindakan yang dapat Anda lakukan dengan kunci, seperti mengenkripsi, mendekripsi, membungkus, menandatangani, dan memverifikasi.

Anda dapat menggunakan `getAttribute` hanya pada kunci yang Anda miliki dan kunci yang dibagikan dengan Anda. Anda dapat menjalankan perintah ini atau perintah [getAttribute](#) di `cloudhsm_mgmt_util`, yang mendapatkan satu nilai atribut kunci dari semua HSMs dalam cluster, dan menulisnya ke stdout atau ke file.

Untuk mendapatkan daftar atribut dan konstanta yang mewakilinya, gunakan perintah [listAttributes](#). Untuk mengubah nilai atribut kunci yang ada, gunakan [setAttribute](#) di `key_mgmt_util` dan [setAttribute](#) di `cloudhsm_mgmt_util`. Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [Masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
getAttribute -h

getAttribute -o <key handle>
               -a <attribute constant>
```

```
-out <file>
```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `getAttribute` untuk mendapatkan atribut kunci di Anda HSMs.

Example: Dapatkan tipe kuncinya

Contoh ini mendapat jenis kunci, seperti AES, 3DES, atau kunci generik, atau pasangan kunci RSA atau kurva elips.

Perintah pertama menjalankan [listAttributes](#), yang mendapat atribut kunci dan konstanta yang mewakilinya. Output menunjukkan bahwa konstanta untuk jenis kunci adalah 256. Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

Command: **listAttributes**

Description

=====

The following are all of the possible attribute values for `getAttribute`s.

OBJ_ATTR_CLASS	= 0
OBJ_ATTR_TOKEN	= 1
OBJ_ATTR_PRIVATE	= 2
OBJ_ATTR_LABEL	= 3
OBJ_ATTR_KEY_TYPE	= 256
OBJ_ATTR_ID	= 258
OBJ_ATTR_SENSITIVE	= 259
OBJ_ATTR_ENCRYPT	= 260
OBJ_ATTR_DECRYPT	= 261
OBJ_ATTR_WRAP	= 262
OBJ_ATTR_UNWRAP	= 263
OBJ_ATTR_SIGN	= 264
OBJ_ATTR_VERIFY	= 266
OBJ_ATTR_LOCAL	= 355
OBJ_ATTR_MODULUS	= 288
OBJ_ATTR_MODULUS_BITS	= 289
OBJ_ATTR_PUBLIC_EXPONENT	= 290
OBJ_ATTR_VALUE_LEN	= 353
OBJ_ATTR_EXTRACTABLE	= 354
OBJ_ATTR_KCV	= 371

Perintah kedua menjalankan `getAttribute`. Perintah ini meminta jenis kunci (atribut 256) untuk handel kunci 524296 dan menuliskannya ke file `attribute.txt`.

```
Command: getAttribute -o 524296 -a 256 -out attribute.txt  
Attributes dumped into attribute.txt file
```

Perintah terakhir mendapat isi dari file kunci. Output mengungkapkan bahwa jenis kunci adalah `0x15` atau 21, yang merupakan kunci Tiga DES (3DES). Untuk definisi kelas dan nilai jenis, lihat [Referensi Atribut Kunci](#).

```
$ cat attribute.txt  
OBJ_ATTR_KEY_TYPE  
0x00000015
```

Example: Dapatkan semua atribut kunci

Perintah ini mendapat semua atribut kunci dengan handel kunci 6 dan menuliskannya ke file `attr_6`. Perintah menggunakan nilai atribut 512, yang mewakili semua atribut.

```
Command: getAttribute -o 6 -a 512 -out attr_6  
  
got all attributes of size 444 attr cnt 17  
Attributes dumped into attribute.txt file  
  
Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS>
```

Perintah ini menunjukkan isi dari file atribut sampel dengan semua nilai atribut. Di antara nilai-nilai, perintah melaporkan bahwa kunci tersebut adalah kunci AES 256-bit dengan ID `test_01` dan label `aes256`. Kunci dapat diekstrak dan persisten, yaitu, bukan kunci hanya sesi. Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

```
$ cat attribute.txt  
  
OBJ_ATTR_CLASS  
0x04  
OBJ_ATTR_KEY_TYPE  
0x15  
OBJ_ATTR_TOKEN  
0x01  
OBJ_ATTR_PRIVATE  
0x01
```

```
OBJ_ATTR_ENCRYPT
0x01
OBJ_ATTR_DECRYPT
0x01
OBJ_ATTR_WRAP
0x01
OBJ_ATTR_UNWRAP
0x01
OBJ_ATTR_SIGN
0x00
OBJ_ATTR_VERIFY
0x00
OBJ_ATTR_LOCAL
0x01
OBJ_ATTR_SENSITIVE
0x01
OBJ_ATTR_EXTRACTABLE
0x01
OBJ_ATTR_LABEL
aes256
OBJ_ATTR_ID
test_01
OBJ_ATTR_VALUE_LEN
0x00000020
OBJ_ATTR_KCV
0x1a4b31
```

Parameter

-h

Menampilkan bantuan untuk perintah.

Wajib: Ya

-o

Menentukan handel kunci dari kunci target. Anda hanya dapat menentukan satu kunci pada setiap perintah. Untuk mendapatkan handel kunci dari suatu kunci, gunakan [findKey](#).

Selain itu, Anda harus memiliki kunci tertentu atau kunci harus dibagikan dengan Anda. Untuk menemukan pengguna kunci, gunakan [getKeyInfo](#).

Wajib: Ya

-a

Mengidentifikasi atribut. Masukkan konstanta yang mewakili atribut, atau 512, yang mewakili semua atribut. Misalnya, untuk mendapatkan jenis kunci, ketik 256, yang merupakan konstanta untuk atribut OBJ_ATTR_KEY_TYPE.

Untuk mendaftar atribut dan konstantanya, gunakan [listAttributes](#). Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

Wajib: Ya

-out

Menulis output ke file yang ditentukan. Ketik jalur file. Anda tidak dapat menuliskan output ke stdout.

Jika file yang ditentukan ada, getAttribute menimpa file tanpa peringatan.

Wajib: Ya

Topik terkait

- [getAttribute](#) di cloudhsm_mgmt_util
- [ListAttributes](#)
- [setAttribute](#)
- [FindKey](#)
- [Referensi Atribut Utama](#)

Ekspor AWS CloudHSM kunci ke format PEM palsu menggunakan KMU

Gunakan perintah getCaviumPrivKey di AWS CloudHSM key_mgmt_util untuk mengekspor kunci pribadi dari modul keamanan perangkat keras (HSM) dalam format PEM palsu. File PEM palsu, yang tidak berisi materi kunci pribadi yang sebenarnya tetapi merujuk kunci pribadi di HSM, kemudian dapat digunakan untuk membuat SSL/TLS pembongkaran dari server web Anda ke AWS CloudHSM. Untuk informasi selengkapnya, lihat [SSL/TLS Offload di Linux menggunakan Tomcat atau SSL/TLS Offload di Linux menggunakan NGINX atau Apache](#).

Sebelum Anda menjalankan perintah key_mgmt_util, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
getCaviumPrivKey -h

getCaviumPrivKey -k <private-key-handle>
                  -out <fake-PEM-file>
```

Contoh

Contoh ini menunjukkan cara menggunakan `getCaviumPrivKey` untuk mengekspor kunci privat dalam format PEM palsu.

Example: Ekspor file PEM palsu

Perintah ini membuat dan mengekspor versi PEM palsu dari kunci privat dengan handel 15 dan menyimpannya ke file bernama `cavKey.pem`. Ketika perintah berhasil, `exportPrivateKey` mengembalikan pesan sukses.

```
Command: getCaviumPrivKey -k 15 -out cavKey.pem

Private Key Handle is written to cavKey.pem in fake PEM format

    getCaviumPrivKey returned: 0x00 : HSM Return: SUCCESS
```

Parameter

Perintah ini membawa parameter berikut.

-h

Menampilkan bantuan baris perintah untuk perintah.

Wajib: Ya

-k

Menentukan handel kunci dari kunci privat yang akan diekspor dalam format PEM palsu.

Wajib: Ya

-out

Menentukan nama file tempat PEM palsu akan ditulis.

Wajib: Ya

Topik terkait

- [importPrivateKey](#)
- [SSL/TLS Offload di Linux menggunakan Tomcat](#)
- [SSL/TLS Offload di Linux menggunakan NGINX atau Apache](#)

Dapatkan sertifikat partisi HSM menggunakan AWS CloudHSM KMU

Gunakan `getCert` perintah di AWS CloudHSM `key_mgmt_util` untuk mengambil sertifikat partisi modul keamanan perangkat keras (HSM) dan menyimpannya ke file. Ketika Anda menjalankan perintah, Anda menetapkan jenis sertifikat untuk diambil. Untuk melakukannya, Anda menggunakan salah satu integer yang sesuai seperti yang dijelaskan dalam bagian [Argumen](#) di bawah ini. Untuk mempelajari tentang peran masing-masing sertifikat ini, lihat [Verifikasi Identitas HSM](#).

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
getCert -h

getCert -f <file-name>
        -t <certificate-type>
```

Contoh

Contoh ini menunjukkan cara menggunakan `getCert` untuk mengambil sebuah klaster pelanggan root sertifikat dan menyimpannya sebagai file.

Example: Ambil sertifikat root pelanggan

Perintah ini ekspor sertifikat root pelanggan (diwakili oleh `integer4`) dan menyimpannya ke file bernama `userRoot.crt`. Ketika perintah berhasil, `getCert` mengembalikan pesan sukses.

```
Command: getCert -f userRoot.crt -s 4
```

```
Cfm3GetCert() returned 0 :HSM Return: SUCCESS
```

Parameter

Perintah ini membawa parameter berikut.

-h

Menampilkan bantuan baris perintah untuk perintah.

Wajib: Ya

-f

Menentukan nama file tempat sertifikat disimpan.

Wajib: Ya

-s

Bilangan bulat yang menentukan jenis sertifikat untuk diambil. Bilangan bulat dan jenis sertifikat yang sesuai mereka adalah sebagai berikut:

- 1— Sertifikat root produsen
- 2 — Sertifikat Perangkat Keras Produsen
- 4— Sertifikat root pelanggan
- 8— Klaster sertifikat (ditandatangani oleh sertifikat root pelanggan)
- 16— Klaster sertifikat (dirantai ke sertifikat root produsen)

Wajib: Ya

Topik terkait

- [Verifikasi Identitas HSM](#)

Dapatkan pengguna AWS CloudHSM kunci menggunakan KMU

Gunakan getKeyInfo perintah di AWS CloudHSM key_mgmt_util untuk mengembalikan pengguna modul keamanan perangkat keras (HSM) pengguna IDs yang dapat menggunakan kunci, termasuk pemilik dan pengguna krypto (CU) dengan siapa kunci tersebut dibagikan. Ketika autentikasi kuorum diaktifkan pada kunci, getKeyInfo juga mengembalikan jumlah pengguna yang harus menyetujui

operasi kriptografi yang menggunakan kunci. Anda dapat menjalankan `getKeyInfo` hanya pada kunci yang Anda miliki dan kunci yang dibagikan dengan Anda.

Ketika Anda menjalankan `getKeyInfo` pada kunci publik, `getKeyInfo` mengembalikan hanya pemilik kunci, meskipun semua pengguna HSM dapat menggunakan kunci publik. Untuk menemukan pengguna HSM pengguna IDs di Anda HSMs, gunakan [ListUsers](#). Untuk menemukan kunci untuk pengguna tertentu, gunakan [findKey](#) -u.

Anda memiliki kunci yang Anda buat. Anda dapat berbagi kunci dengan pengguna lain saat Anda membuatnya. Kemudian, untuk berbagi atau batal berbagi kunci yang ada, gunakan [shareKey](#) di `cloudhsm_mgmt_util`.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
getKeyInfo -h  
  
getKeyInfo -k <key-handle>
```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `getKeyInfo` untuk mendapatkan informasi tentang pengguna kunci.

Example: Dapatkan pengguna untuk kunci simetris

Perintah ini mendapat pengguna yang dapat menggunakan kunci AES (simetris) dengan handel kunci 9. Output menunjukkan bahwa pengguna 3 memiliki kunci dan telah membaginya dengan pengguna 4.

```
Command: getKeyInfo -k 9  
  
Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS  
  
Owned by user 3  
  
also, shared to following 1 user(s):  
  
4
```

Example: Dapatkan pengguna untuk key pair asimetris

Perintah ini menggunakan `getKeyInfo` untuk mendapatkan pengguna yang dapat menggunakan kunci dalam pasangan kunci RSA (asimetris). Kunci publik memiliki handel kunci 21. Kunci privat memiliki handel kunci 20.

Ketika Anda menjalankan `getKeyInfo` pada kunci pribadi (20), ia mengembalikan pemilik kunci (3) dan pengguna krypto (CUs) 4 dan 5, dengan siapa kunci tersebut dibagikan.

```
Command: getKeyInfo -k 20
```

```
Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS
```

```
Owned by user 3
```

```
also, shared to following 2 user(s):
```

```
4
```

```
5
```

Ketika Anda menjalankan `getKeyInfo` pada kunci publik (21), ini mengembalikan hanya pemilik kunci (3).

```
Command: getKeyInfo -k 21
```

```
Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS
```

```
Owned by user 3
```

Untuk mengonfirmasi bahwa pengguna 4 dapat menggunakan kunci publik (dan semua kunci publik pada HSM), gunakan parameter `-u` dari [findKey](#).

Output menunjukkan bahwa pengguna 4 dapat menggunakan kunci publik (21) dan privat (20) dalam pasangan kunci. Pengguna 4 juga dapat menggunakan semua kunci publik lainnya dan kunci privat yang telah mereka buat atau yang telah dibagi dengan mereka.

```
Command: findKey -u 4
```

```
Total number of keys present 8
```

```
number of keys matched from start index 0::7
```

```
11, 12, 262159, 262161, 262162, 19, 20, 21
```



```
Cluster Error Status
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Example: Dapatkan nilai otentikasi kuorum (`m_value`) untuk kunci

Contoh ini menunjukkan cara untuk mendapatkan `m_value` untuk kunci, yaitu, jumlah pengguna dalam kuorum yang harus menyetujui operasi kriptografi yang menggunakan kunci.

Ketika kuorum autentikasi diaktifkan pada kunci, kuorum pengguna harus menyetujui operasi kriptografi yang menggunakan kunci. Untuk mengaktifkan autentikasi kuorum dan menetapkan ukuran kuorum, gunakan parameter `-m_value` saat Anda membuat kunci.

Perintah ini menggunakan [gen RSAKey Pair](#) untuk membuat key pair RSA yang dibagikan dengan pengguna 4. Perintah menggunakan parameter `m_value` untuk mengaktifkan autentikasi kuorum pada kunci privat dalam pasangan dan menetapkan ukuran kuorum untuk dua pengguna. Jumlah pengguna harus cukup besar untuk memberikan persetujuan yang diperlukan.

Output menunjukkan bahwa perintah membuat kunci publik 27 dan kunci privat 28.

```
Command: genRSAKeyPair -m 2048 -e 195193 -l rsa_mofn -id rsa_mv2 -u 4 -m_value 2
```

```
Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS
```

```
Cfm3GenerateKeyPair:    public key handle: 27    private key handle: 28
```

```
Cluster Error Status
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

Perintah ini menggunakan `getKeyInfo` untuk mendapatkan informasi tentang pengguna kunci privat. Output menunjukkan bahwa kunci tersebut dimiliki oleh pengguna 3 dan berbagi dengan pengguna 4. Ini juga menunjukkan bahwa kuorum dua pengguna harus menyetujui setiap operasi kriptografi yang menggunakan kunci.

```
Command: getKeyInfo -k 28
```

```
Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS
```

Owned by user 3

also, shared to following 1 user(s):

4

2 Users need to approve to use/manage this key

Parameter

-h

Menampilkan bantuan baris perintah untuk perintah.

Wajib: Ya

-k

Menentukan handel kunci dari satu kunci di HSM. Masukkan handel kunci dari kunci yang Anda miliki atau bagikan. Parameter ini diperlukan.

Untuk menemukan handel kunci, gunakan perintah [findKey](#).

Wajib: Ya

Topik terkait

- [getKeyInfo](#) di cloudhsm_mgmt_util
- [ListUsers](#)
- [FindKey](#)
- [findAllKeys](#) di cloudhsm_mgmt_util

Menampilkan informasi bantuan untuk AWS CloudHSM KMU

Gunakan help perintah di AWS CloudHSM key_mgmt_util untuk menampilkan informasi tentang semua perintah key_mgmt_util yang tersedia.

Sebelum Anda menjalankan help, Anda harus [memulai key_mgmt_util](#).

Sintaksis

```
help
```

Contoh

Contoh berikut menunjukkan output perintah `help`.

Example

Command: **help**

Help Commands Available:

Syntax: <command> -h

Command	Description
=====	=====
exit	Exits this application
help	Displays this information
Configuration and Admin Commands	
getHSMInfo	Gets the HSM Information
getPartitionInfo	Gets the Partition Information
listUsers	Lists all users of a partition
loginStatus	Gets the Login Information
loginHSM	Login to the HSM
logoutHSM	Logout from the HSM
M of N commands	
getToken	Initiate an MxN service and get Token
delToken	delete Token(s)
approveToken	Approves an MxN service
listTokens	List all Tokens in the current partition
Key Generation Commands	
Asymmetric Keys:	
genRSAKeyPair	Generates an RSA Key Pair
genDSAKeyPair	Generates a DSA Key Pair
genECCKeyPair	Generates an ECC Key Pair
Symmetric Keys:	
genPBEKey	Generates a PBE DES3 key
genSymKey	Generates a Symmetric keys

Key Import/Export Commands

<code>createPublicKey</code>	Creates an RSA public key
<code>importPubKey</code>	Imports RSA/DSA/EC Public key
<code>exportPubKey</code>	Exports RSA/DSA/EC Public key
<code>importPrivateKey</code>	Imports RSA/DSA/EC private key
<code>exportPrivateKey</code>	Exports RSA/DSA/EC private key
<code>imSymKey</code>	Imports a Symmetric key
<code>exSymKey</code>	Exports a Symmetric key
<code>wrapKey</code>	Wraps a key from from HSM using the specified handle
<code>unwrapKey</code>	UnWraps a key into HSM using the specified handle

Key Management Commands

<code>deleteKey</code>	Delete Key
<code>setAttribute</code>	Sets an attribute of an object
<code>getKeyInfo</code>	Get Key Info about shared users/sessions
<code>findKey</code>	Find Key
<code>findSingleKey</code>	Find single Key
<code>getAttribute</code>	Reads an attribute from an object

Certificate Setup Commands

<code>getCert</code>	Gets Partition Certificates stored on HSM

Key Transfer Commands

<code>insertMaskedObject</code>	Inserts a masked object
<code>extractMaskedObject</code>	Extracts a masked object

Management Crypto Commands

<code>sign</code>	Generates a signature
<code>verify</code>	Verifies a signature
<code>aesWrapUnwrap</code>	Does NIST AES Wrap/Unwrap

Helper Commands

<code>Error2String</code>	Converts Error codes to Strings
<code>save key handle in fake PEM format</code>	
<code>getCaviumPrivKey</code>	Saves an RSA private key handle in fake PEM format
<code>IsValidKeyHandlefile</code>	Checks if private key file has an HSM key handle or a real key
<code>listAttributes</code>	List all attributes for getAttributes
<code>listECCCurveIds</code>	List HSM supported ECC CurveIds

Parameter

Tidak ada parameter untuk perintah ini.

Topik terkait

- [LoginHSM dan LogouthSM](#)

Impor kunci pribadi menggunakan AWS CloudHSM KMU

Gunakan `importPrivateKey` perintah di AWS CloudHSM `key_mgmt_util` untuk mengimpor kunci pribadi asimetris dari file ke modul keamanan perangkat keras (HSM). HSM tidak mengizinkan impor langsung kunci dalam cleartext. Perintah mengenkripsi kunci pribadi menggunakan kunci pembungkus AES yang Anda tentukan dan membuka kunci di dalam HSM. Jika Anda mencoba mengaitkan AWS CloudHSM kunci dengan sertifikat, lihat [topik ini](#).

Note

Anda tidak dapat mengimpor kunci PEM yang dilindungi kata sandi menggunakan kunci simetris atau pribadi.

Anda harus menentukan kunci pembungkus AES yang memiliki `OBJ_ATTR_UNWRAP` dan nilai `OBJ_ATTR_ENCRYPT` atribut. 1 Untuk menemukan atribut kunci, gunakan perintah [getAttribute](#).

Note

Perintah ini tidak menawarkan pilihan untuk menandai kunci yang diimpor sebagai tidak dapat ekspor.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
importPrivateKey -h

importPrivateKey -l <label>
                  -f <key-file>
                  -w <wrapping-key-handle>
                  [-sess]
                  [-id <key-id>]
```

```
[-m_value <0...8>]  
[min_srv <minimum-number-of-servers>]  
[-timeout <number-of-seconds>]  
[-u <user-ids>]  
[-wk <wrapping-key-file>]  
[-attest]
```

Contoh

Contoh ini menunjukkan cara menggunakan `importPrivateKey` untuk mengimpor kunci privat ke HSM.

Example: Impor kunci pribadi

Perintah ini mengimpor kunci privat dari sebuah file bernama `rsa2048.key` dengan label `rsa2048-imported` dan kunci pembungkus dengan handel 524299. Ketika perintah berhasil, `importPrivateKey` mengembalikan handel kunci untuk kunci yang diimpor dan pesan sukses.

```
Command: importPrivateKey -f rsa2048.key -l rsa2048-imported -w 524299
```

```
BER encoded key length is 1216
```

```
Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

```
Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS
```

```
Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS
```

```
Private Key Unwrapped. Key Handle: 524301
```

```
Cluster Error Status
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Parameter

Perintah ini membawa parameter berikut.

-h

Menampilkan bantuan baris perintah untuk perintah.

Wajib: Ya

-l

Menentukan label kunci privat yang ditetapkan pengguna.

Wajib: Ya

-f

Menentukan nama file kunci untuk diimpor.

Wajib: Ya

-w

Menentukan handel kunci dari kunci pembungkus. Parameter ini diperlukan. Untuk menemukan handel kunci, gunakan perintah [findKey](#).

Untuk menentukan apakah kunci dapat digunakan sebagai kunci pembungkus, gunakan [getAttribute](#) untuk mendapatkan nilai dari atribut OBJ_ATTR_WRAP (262). Untuk membuat kunci pembungkus, gunakan [genSymKey](#) untuk membuat kunci AES (tipe 31).

Jika Anda menggunakan parameter `-wk` untuk menentukan kunci pembuka bungkus eksternal, kunci pembungkus `-w` digunakan untuk membungkus, tapi tidak membuka, kunci selama impor.

Wajib: Ya

-sess

Menentukan kunci diimpor sebagai kunci sesi.

Default: Kunci yang diimpor disimpan sebagai kunci persisten (token) di klaster.

Wajib: Tidak

-id

Menentukan ID kunci yang akan diimpor.

Default: Tidak ada nilai ID.

Wajib: Tidak

-m_value

Menentukan jumlah pengguna yang harus menyetujui operasi kriptografi yang menggunakan kunci impor. Masukkan nilai dari **0** sampai **8**.

Parameter ini hanya valid jika parameter `-u` dalam perintah membagikan kunci dengan cukup pengguna untuk memenuhi persyaratan `m_value`.

Default: 0

Wajib: Tidak

-min_srv

Menentukan jumlah minimum HSMs di mana kunci impor disinkronkan sebelum nilai `-timeout` parameter berakhir. Jika kunci tidak disinkronkan ke jumlah tertentu server dalam waktu yang ditentukan, kunci tidak dibuat.

AWS CloudHSM secara otomatis menyinkronkan setiap kunci ke setiap HSM di cluster. Untuk mempercepat proses Anda, tetapkan nilai kurang dari jumlah HSMs di cluster dan tetapkan nilai batas waktu yang rendah. `min_srv` Namun, perhatikan bahwa beberapa permintaan mungkin tidak menghasilkan kunci.

Default: 1

Wajib: Tidak

-timeout

Menentukan jumlah detik untuk menunggu kunci untuk disinkronkan HSMs ketika `min-serv` parameter disertakan. Jika tidak ada nomor yang ditentukan, polling berlanjut selamanya.

Default: Tanpa batas

Wajib: Tidak

-u

Menentukan daftar pengguna dengan siapa berbagi kunci privat yang diimpor. Parameter ini memberikan izin kepada pengguna krypto HSM lainnya (CUs) untuk menggunakan kunci yang diimpor dalam operasi kriptografi.

Masukkan daftar pengguna HSM yang dipisahkan koma IDs, seperti. `-u 5,6` Jangan sertakan ID pengguna HSM dari pengguna saat ini. [Untuk menemukan pengguna IDs HSM CUs di HSM, gunakan ListUsers.](#)

Default: Hanya pengguna saat ini dapat menggunakan kunci privat.

Wajib: Tidak

-wk

Menentukan kunci yang akan digunakan untuk membungkus kunci yang sedang diimpor. Masukkan jalur dan nama file yang berisi kunci AES plaintext.

Bila Anda menyertakan parameter ini, `importPrivateKey` menggunakan kunci dalam file `-wk` untuk membungkus kunci yang sedang diimpor. Parameter ini juga menggunakan kunci yang ditentukan oleh parameter `-w` untuk membukanya.

Default: Gunakan kunci pembungkus yang ditentukan dalam parameter `-w` untuk membungkus dan membuka.

Wajib: Tidak

-attest

Melakukan pemeriksaan pengesahan pada respons firmware untuk memastikan bahwa firmware tempat klaster berjalan belum disisipi.

Wajib: Tidak

Topik terkait

- [WrapKey](#)
- [unWrapKey](#)
- [genSymKey](#)
- [exportPrivateKey](#)

Impor kunci publik menggunakan AWS CloudHSM KMU

Gunakan `importPubKey` perintah di AWS CloudHSM `key_mgmt_util` untuk mengimpor kunci publik format PEM ke dalam modul keamanan perangkat keras (HSM). Anda dapat menggunakannya untuk mengimpor kunci privat yang dihasilkan di luar HSM. Anda juga dapat menggunakan perintah untuk mengimpor kunci yang diekspor dari HSM, seperti yang diekspor oleh perintah. [exportPubKey](#)

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [Masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
importPubKey -h
```

```
importPubKey -l <label>
             -f <key-file>
             [-sess]
             [-id <key-id>]
             [min_srv <minimum-number-of-servers>]
             [-timeout <number-of-seconds>]
```

Contoh

Contoh ini menunjukkan cara menggunakan importPubKey untuk mengimpor kunci publik ke HSM.

Example: Impor kunci publik

Perintah ini mengimpor kunci publik dari sebuah file bernama public.pem dengan label importedPublicKey. Ketika perintah berhasil, importPubKey mengembalikan handel kunci untuk kunci yang diimpor dan pesan sukses.

Command: **importPubKey -l importedPublicKey -f public.pem**

Cfm3CreatePublicKey returned: 0x00 : HSM Return: SUCCESS

Public Key Handle: 262230

Cluster Error Status

Node id 2 and err state 0x00000000 : HSM Return: SUCCESS

Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

Node id 1 and err state 0x00000000 : HSM Return: SUCCESS

Parameter

Perintah ini membawa parameter berikut.

-h

Menampilkan bantuan baris perintah untuk perintah.

Wajib: Ya

-l

Menentukan label kunci privat yang ditetapkan pengguna.

Wajib: Ya

-f

Menentukan nama file kunci untuk diimpor.

Wajib: Ya

-sess

Menunjuk kunci yang diimpor sebagai kunci sesi.

Default: Kunci yang diimpor disimpan sebagai kunci persisten (token) di klaster.

Wajib: Tidak

-id

Menentukan ID kunci yang akan diimpor.

Default: Tidak ada nilai ID.

Wajib: Tidak

-min_srv

Menentukan jumlah minimum HSMs yang kunci impor disinkronkan sebelum nilai `-timeout` parameter berakhir. Jika kunci tidak disinkronkan ke jumlah tertentu server dalam waktu yang ditentukan, kunci tidak dibuat.

AWS CloudHSM secara otomatis menyinkronkan setiap kunci ke setiap HSM di cluster. Untuk mempercepat proses Anda, tetapkan nilai kurang dari jumlah HSMs di cluster dan tetapkan nilai batas waktu yang rendah. `min_srv` Namun, perhatikan bahwa beberapa permintaan mungkin tidak menghasilkan kunci.

Default: 1

Wajib: Tidak

-timeout

Menentukan jumlah detik untuk menunggu kunci untuk disinkronkan HSMs ketika `min-serv` parameter disertakan. Jika tidak ada nomor yang ditentukan, polling berlanjut selamanya.

Default: Tanpa batas

Wajib: Tidak

Topik terkait

- [exportPubKey](#)
- [Menghasilkan Kunci](#)

Impor kunci simetris plaintext menggunakan KMU AWS CloudHSM

Gunakan `imSymKey` perintah di alat AWS CloudHSM `key_mgmt_util` untuk mengimpor salinan teks biasa dari kunci simetris dari file ke modul keamanan perangkat keras (HSM). Anda dapat menggunakannya untuk mengimpor kunci yang Anda hasilkan dengan metode apa pun di luar HSM dan kunci yang diekspor dari HSM, seperti kunci yang ditulis oleh perintah [exSymKey](#), ke file.

Selama proses impor, `imSymKey` menggunakan kunci AES yang Anda pilih (kunci pembungkus) untuk membungkus (mengkripsi) dan kemudian membuka(dekripsi) kunci yang akan diimpor. Namun, `imSymKey` bekerja hanya pada file yang berisi kunci teks terang. Untuk mengekspor dan mengimpor kunci terenkripsi, gunakan [WrapKey](#) dan perintah. [unWrapKey](#)

Selain itu, perintah `imSymKey` mengimpor hanya kunci simetris. Untuk mengimpor kunci publik, gunakan [importPubKey](#). Untuk mengimpor kunci pribadi, gunakan [importPrivateKey](#) atau [WrapKey](#).

Note

Anda tidak dapat mengimpor kunci PEM yang dilindungi kata sandi menggunakan kunci simetris atau pribadi.

Kunci impor bekerja sangat mirip dengan kunci yang dihasilkan di HSM. Namun, nilai [atribut OBJ_ATTR_LOCAL](#) adalah nol, yang menunjukkan bahwa itu tidak dihasilkan secara lokal. Anda dapat menggunakan perintah berikut untuk berbagi kunci simetris saat Anda mengimpornya. Anda dapat menggunakan perintah `shareKey` di [cloudhsm_mgmt_util](#) untuk berbagi kunci setelah diimpor.

```
imSymKey -l aesShared -t 31 -f kms.key -w 3296 -u 5
```

Setelah Anda mengimpor kunci, pastikan untuk menandai atau menghapus file kunci. Perintah ini tidak mencegah Anda dari mengimpor materi kunci yang sama beberapa kali. Hasilnya, beberapa

kunci dengan handel kunci yang berbeda dan materi kunci yang sama, menyulitkan untuk melacak penggunaan materi kunci dan mencegahnya melebihi batas kriptografinya.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
imSymKey -h

imSymKey -f <key-file>
          -w <wrapping-key-handle>
          -t <key-type>
          -l <label>
          [-id <key-ID>]
          [-sess]
          [-wk <wrapping-key-file> ]
          [-attest]
          [-min_srv <minimum-number-of-servers>]
          [-timeout <number-of-seconds> ]
          [-u <user-ids>]
```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `imSymKey` untuk mengimpor kunci simetris ke dalam kunci Anda HSMs.

Example: Impor kunci simetris AES

Contoh ini digunakan `imSymKey` untuk mengimpor kunci simetris AES ke dalam. HSMs

Perintah pertama menggunakan OpenSSL untuk menghasilkan kunci simetris AES 256-bit acak. Menyimpan kunci dalam file `aes256.key`.

```
$ openssl rand -out aes256.key 32
```

Perintah kedua digunakan `imSymKey` untuk mengimpor kunci AES dari `aes256.key` file ke file. HSMs Perintah menggunakan kunci 20, kunci AES di HSM, sebagai kunci pembungkus dan menentukan label `imported`. Berbeda dengan ID, label tidak perlu unik di klaster. Nilai dari parameter `-t` (tipe) adalah 31, yang mewakili AES.

Output menunjukkan bahwa kunci dalam file dibungkus dan dibuka, kemudian diimpor ke HSM, di mana handle kunci 262180 ditugaskan.

```
Command: imSymKey -f aes256.key -w 20 -t 31 -l imported

Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

Symmetric Key Unwrapped. Key Handle: 262180

Cluster Error Status
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Perintah berikutnya menggunakan [getAttribute](#) untuk mendapatkan atribut OBJ_ATTR_LOCAL ([Atribut 355](#)) dari kunci yang baru diimpor dan menuliskannya ke file `attr_262180`.

```
Command: getAttribute -o 262180 -a 355 -out attributes/attr_262180
Attributes dumped into attributes/attr_262180_imported file

Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS
```

Ketika Anda memeriksa atribut file, Anda dapat melihat bahwa nilai atribut OBJ_ATTR_LOCAL adalah nol, yang menunjukkan bahwa materi kunci tidak dihasilkan dalam HSM.

```
$ cat attributes/attr_262180_local
OBJ_ATTR_LOCAL
0x00000000
```

Example: Pindahkan kunci simetris antar cluster

Contoh ini menunjukkan cara menggunakan [exSymKey](#) dan `imSymKey` memindahkan kunci AES plaintext antar cluster. Anda dapat menggunakan proses seperti ini untuk membuat pembungkus AES yang ada di HSMs kedua cluster. Setelah kunci pembungkus bersama berada di tempat, Anda dapat menggunakan [WrapKey](#) dan [unWrapKey](#) untuk memindahkan kunci terenkripsi antara cluster.

Pengguna CU yang melakukan operasi ini harus memiliki izin untuk masuk ke HSMs kedua cluster.

Perintah pertama digunakan [exSymKey](#) untuk mengekspor kunci 14, kunci AES 32-bit, dari cluster 1 ke dalam file. `aes.key` Ini menggunakan kunci 6, kunci AES HSMs pada cluster 1, sebagai kunci pembungkus.

```
Command: exSymKey -k 14 -w 6 -out aes.key
```

```
Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS
```

```
Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

```
Wrapped Symmetric Key written to file "aes.key"
```

Pengguna kemudian masuk ke `key_mgmt_util` di cluster 2 dan menjalankan `imSymKey` perintah untuk mengimpor kunci dalam file ke dalam cluster 2. `aes.key` HSMs Perintah ini menggunakan kunci 252152, kunci AES HSMs di cluster 2, sebagai kunci pembungkus.

Karena tombol pembungkus itu [exSymKey](#) dan `imSymKey` gunakan bungkus dan segera buka kunci target, tombol pembungkus pada cluster yang berbeda tidak perlu sama.

Output menunjukkan bahwa kunci berhasil diimpor ke dalam klaster 2 dan handel kunci 21 ditugaskan.

```
Command: imSymKey -f aes.key -w 262152 -t 31 -l xcluster
```

```
Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

```
Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS
```

```
Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS
```

```
Symmetric Key Unwrapped. Key Handle: 21
```

```
Cluster Error Status
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Untuk membuktikan bahwa kunci 14 klaster 1 dan kunci 21 di klaster 2 memiliki materi kunci yang sama, dapatkan nilai pemeriksaan kunci (KCV) dari setiap kunci. Jika nilai KCV adalah sama, bahan utama adalah sama.

Perintah berikut menggunakan [getAttribute](#) di klaster 1 untuk menuliskan nilai atribut KCV (atribut 371) dari kunci 14 ke file `attr_14_kcv`. Kemudian, ini menggunakan perintah `cat` untuk mendapatkan isi dari file `attr_14_kcv`.

```
Command: getAttribute -o 14 -a 371 -out attr_14_kcv  
Attributes dumped into attr_14_kcv file  
  
$ cat attr_14_kcv  
OBJ_ATTR_KCV  
0xc33cbd
```

Perintah serupa ini menggunakan [getAttribute](#) di klaster 2 untuk menulis nilai atribut KCV (atribut 371) kunci 21 ke file `attr_21_kcv`. Kemudian, ini menggunakan perintah `cat` untuk mendapatkan isi dari file `attr_21_kcv`.

```
Command: getAttribute -o 21 -a 371 -out attr_21_kcv  
Attributes dumped into attr_21_kcv file  
  
$ cat attr_21_kcv  
OBJ_ATTR_KCV  
0xc33cbd
```

Output menunjukkan bahwa nilai KCV dari dua kunci adalah sama, yang membuktikan bahwa materi kunci adalah sama.

Karena materi kunci yang sama ada di kedua cluster, Anda sekarang dapat berbagi kunci terenkripsi antara cluster tanpa pernah mengekspos kunci plaintext. HSMs Misalnya, Anda dapat menggunakan perintah `wrapKey` dengan kunci pembungkus 14 untuk mengeksport kunci terenkripsi dari klaster 1, dan kemudian gunakan `unwrapKey` dengan pembungkus kunci 21 untuk mengimpor kunci dienkripsi ke dalam klaster 2.

Example: Impor kunci sesi

Perintah ini menggunakan parameter `-sess` dari `imSymKey` untuk mengimpor kunci 192-bit Triple DES yang hanya berlaku dalam sesi saat ini.

Perintah menggunakan parameter `-f` untuk menentukan file yang berisi kunci untuk diimpor, parameter `-t` untuk menentukan jenis kunci, dan parameter `-w` untuk menentukan kunci pembungkus. Perintah menggunakan parameter `-l` untuk menentukan label yang mengelompokkan

kunci dan parameter `-id` untuk membuat pengenalan yang mudah diingat, tapi unik, untuk kunci. Perintah ini juga menggunakan parameter `-attest` untuk memverifikasi firmware yang mengimpor kunci.

Output menunjukkan bahwa kunci berhasil dibungkus dan dibuka, diimpor ke HSM, dan handle kunci 37 ditugaskan. Selain itu, pemeriksaan pengesahan dilewatkan, yang menunjukkan bahwa firmware belum dirusak.

```
Command: imSymKey -f 3des192.key -w 6 -t 21 -l temp -id test01 -sess -attest
```

```
Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

```
Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS
```

```
Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS
```

```
Symmetric Key Unwrapped. Key Handle: 37
```

```
Attestation Check : [PASS]
```

```
Cluster Error Status
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Selanjutnya, Anda dapat menggunakan perintah [getAttribute](#) atau [findKey](#) untuk memverifikasi atribut kunci yang baru diimpor. Perintah berikut menggunakan `findKey` untuk memverifikasi bahwa kunci 37 memiliki jenis, label, dan ID yang ditentukan oleh perintah, dan bahwa itu adalah kunci sesi. Ditampilkan pada baris 5 output, `findKey` melaporkan bahwa satu-satunya kunci yang cocok dengan semua atribut adalah kunci 37.

```
Command: findKey -t 21 -l temp -id test01 -sess 1
```

```
Total number of keys present 1
```

```
number of keys matched from start index 0::0  
37
```

```
Cluster Error Status
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Parameter

-attest

Menjalankan pemeriksaan integritas yang memverifikasi bahwa firmware tempat kluster berjalan belum dirusak.

Default: Tidak ada pemeriksaan pengesahan.

Wajib: Tidak

-f

Menentukan file yang berisi kunci untuk diimpor.

File harus berisi salinan teks biasa dari kunci AES atau Triple DES dengan panjang yang ditentukan. RC4 dan kunci DES tidak valid pada mode FIPS HSMs.

- AES: 16, 24 atau 32 byte
- Triple DES (3DES): 24 byte

Wajib: Ya

-h

Menampilkan bantuan untuk perintah.

Wajib: Ya

-id

Menentukan pengenal yang ditetapkan pengguna untuk kunci. Ketik string yang unik dalam kluster. Default-nya adalah string kosong.

Default: Tidak ada nilai ID.

Wajib: Tidak

-l

Menentukan label yang ditetapkan pengguna untuk kunci. Ketik string.

Anda dapat menggunakan frasa apa pun yang membantu Anda mengidentifikasi kunci. Karena label tidak harus unik, Anda dapat menggunakannya pada grup dan mengelompokkan kunci.

Wajib: Ya

`-min_srv`

Menentukan jumlah minimum HSMs di mana kunci disinkronkan sebelum nilai `-timeout` parameter berakhir. Jika kunci tidak disinkronkan ke jumlah tertentu server dalam waktu yang ditentukan, kunci tidak dibuat.

AWS CloudHSM secara otomatis menyinkronkan setiap kunci ke setiap HSM di cluster. Untuk mempercepat proses Anda, tetapkan nilai kurang dari jumlah HSMs di cluster dan tetapkan nilai batas waktu yang rendah. `min_srv` Namun, perhatikan bahwa beberapa permintaan mungkin tidak menghasilkan kunci.

Default: 1

Wajib: Tidak

`-sess`

Membuat kunci yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Gunakan parameter ini ketika Anda memerlukan kunci hanya sebentar, seperti kunci pembungkus yang mengenkripsi, dan kemudian dengan cepat mendekripsi, kunci lain. Jangan gunakan kunci sesi untuk mengenkripsi data yang mungkin perlu Anda dekripsi setelah sesi berakhir.

Untuk mengubah kunci sesi menjadi kunci (token) persisten, gunakan [setAttribute](#).

Default: Kunci persisten.

Wajib: Tidak

`-timeout`

Menentukan berapa lama (dalam detik) perintah menunggu kunci untuk disinkronkan dengan jumlah yang HSMs ditentukan oleh parameter. `min_srv`

Parameter ini hanya valid jika parameter `min_srv` juga digunakan dalam perintah.

Default: Tidak ada waktu habis. Perintah menunggu tanpa batas waktu dan kembali hanya ketika kunci disinkronkan ke jumlah minimum server.

Wajib: Tidak

-t

Menentukan jenis kunci simetris. Masukkan konstanta yang mewakili jenis kunci. Misalnya, untuk membuat kunci AES, masukkan -t 31.

Nilai yang valid:

- 21: [Tiga DES \(3DES\)](#).
- 31: [AES](#)

Wajib: Ya

-u

Berbagi kunci yang Anda impor dengan pengguna tertentu. Parameter ini memberikan izin kepada pengguna krypto HSM lainnya (CUs) untuk menggunakan kunci ini dalam operasi kriptografi.

Ketik satu ID atau daftar pengguna HSM yang dipisahkan koma IDs, seperti -u. 5,6 Jangan sertakan ID pengguna HSM dari pengguna saat ini. Untuk menemukan ID, Anda dapat menggunakan perintah [listUsers](#) di alat baris perintah cloudhsm_mgmt_util atau perintah [listUsers](#) pada alat baris perintah key_mgmt_util.

Wajib: Tidak

-w

Menentukan handel kunci dari kunci pembungkus. Parameter ini diperlukan. Untuk menemukan handel kunci, gunakan perintah [findKey](#).

Kunci pembungkus adalah kunci dalam HSM yang digunakan untuk mengenkripsi (“membungkus”) dan kemudian mendekripsi (“membuka bungkus”) kunci selama proses impor. Hanya kunci AES yang dapat digunakan sebagai kunci pembungkus.

Anda dapat menggunakan kunci AES (dari berbagai ukuran) sebagai kunci pembungkus. Karena kunci pembungkus membungkus, dan kemudian segera membuka, kunci target, Anda dapat menggunakan sebagai kunci AES hanya sesi sebagai kunci pembungkus. Untuk menentukan apakah kunci dapat digunakan sebagai kunci pembungkus, gunakan [getAttribute](#) untuk mendapatkan nilai dari atribut OBJ_ATTR_WRAP (262). Untuk membuat kunci pembungkus, gunakan [genSymKey](#) untuk membuat kunci AES (tipe 31).

Jika Anda menggunakan parameter `-wk` untuk menentukan kunci pembungkus eksternal, kunci pembungkus `-w` digunakan untuk membuka, tetapi tidak untuk membungkus, kunci yang sedang diimpor.

 Note

Kunci 4 adalah kunci internal yang tidak didukung. Kami merekomendasikan bahwa Anda menggunakan kunci AES yang Anda buat dan kelola sebagai kunci pembungkus.

Wajib: Ya

`-wk`

Gunakan kunci AES dalam file tertentu untuk membungkus kunci yang sedang diimpor. Masukkan jalur dan nama file yang berisi kunci AES plaintext.

Bila Anda menyertakan parameter ini, `imSymKey` menggunakan kunci dalam file `-wk` untuk membungkus kunci yang diimpor dan menggunakan kunci dalam HSM yang ditentukan oleh parameter `-w` untuk membukanya. Nilai parameter `-w` dan `-wk` harus menyelesaikan ke kunci plaintext yang sama.

Default: Gunakan kunci pembungkus pada HSM untuk membuka.

Wajib: Tidak

Topik terkait

- [genSymKey](#)
- [exSymKey](#)
- [WrapKey](#)
- [unWrapKey](#)
- [exportPrivateKey](#)
- [exportPubKey](#)

Masukkan objek bertopeng menggunakan AWS CloudHSM KMU

Gunakan `insertMaskedObject` perintah di AWS CloudHSM `key_mgmt_util` untuk menyisipkan objek bertopeng dari file ke modul keamanan perangkat keras yang ditunjuk (HSM). Objek tertutup adalah

objek dikloning yang diekstrak dari HSM dengan menggunakan perintah [extractMaskedObject](#) Objek itu hanya dapat digunakan setelah memasukkannya kembali ke klaster asli. Anda hanya dapat memasukkan objek tertutup ke dalam klaster yang sama dengan klaster asalnya dihasilkan, atau kloning dari klaster tersebut. Ini termasuk versi kloning dari klaster asli yang dihasilkan dengan [menyalin cadangan di seluruh wilayah](#) dan [menggunakan cadangan tersebut untuk membuat sebuah klaster baru](#).

Objek tertutup adalah cara yang efisien untuk membongkar dan menyinkronkan kunci, termasuk kunci yang tidak dapat diekstrak (yaitu, kunci yang memiliki nilai [OBJ_ATTR_EXTRACTABLE](#) dari 0). [Dengan cara ini, kunci dapat disinkronkan dengan aman di seluruh cluster terkait di berbagai wilayah tanpa perlu memperbarui file konfigurasi.](#) [AWS CloudHSM](#)

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [Masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
insertMaskedObject -h

insertMaskedObject -f <filename>
                    [-min_srv <minimum-number-of-servers>]
                    [-timeout <number-of-seconds>]
```

Contoh

Contoh ini menunjukkan cara menggunakan `insertMaskedObject` untuk menyisipkan file objek tertutup ke dalam HSM.

Example: Masukkan objek bertopeng

Perintah ini menyisipkan objek tertutup ke dalam HSM dari sebuah file bernama `maskedObj`. Ketika perintah berhasil, `insertMaskedObject` mengembalikan handel kunci untuk kunci didekripsi dari objek tertutup, dan pesan sukses.

```
Command: insertMaskedObject -f maskedObj

Cfm3InsertMaskedObject returned: 0x00 : HSM Return: SUCCESS
    New Key Handle: 262433

    Cluster Error Status
```

```
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

Parameter

Perintah ini membawa parameter berikut.

-h

Menampilkan bantuan baris perintah untuk perintah.

Wajib: Ya

-f

Menentukan nama file dari objek tertutup untuk disisipkan.

Wajib: Ya

-min_srv

Menentukan jumlah minimum server tempat objek tertutup yang dimasukkan disinkronkan sebelum nilai parameter `-timeout` kedaluwarsa. Jika objek tidak disinkronkan dengan jumlah tertentu server dalam waktu yang ditentukan, objek tidak dimasukkan.

Default: 1

Wajib: Tidak

-timeout

Menentukan jumlah detik untuk menunggu kunci untuk disinkronkan di server ketika parameter `min-serv` disertakan. Jika tidak ada nomor yang ditentukan, polling berlanjut selamanya.

Default: Tanpa batas

Wajib: Tidak

Topik terkait

- [extractMaskedObject](#)
- [SyncKey](#)

- [Menyalin Cadangan di Seluruh Wilayah](#)
- [Membuat AWS CloudHSM Cluster dari Backup Sebelumnya](#)

Validasi file kunci menggunakan AWS CloudHSM KMU

Gunakan `IsValidKeyHandlefile` perintah di AWS CloudHSM `key_mgmt_util` untuk mengetahui apakah file kunci berisi kunci pribadi nyata atau kunci RSA PEM palsu. Sebuah file PEM palsu tidak mengandung materi kunci privat yang sebenarnya, tetapi sebaliknya, mereferensikan kunci privat di HSM. File semacam itu dapat digunakan untuk membuat SSL/TLS pembongkaran dari server web Anda ke AWS CloudHSM. Untuk informasi selengkapnya, lihat [SSL/TLS Offload di Linux menggunakan Tomcat atau SSL/TLS Offload di Linux menggunakan NGINX atau Apache](#).

Note

`IsValidKeyHandlefile` hanya berfungsi untuk kunci RSA.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [Masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
IsValidKeyHandlefile -h  
  
IsValidKeyHandlefile -f <rsa-private-key-file>
```

Contoh

Contoh ini menunjukkan cara menggunakan `IsValidKeyHandlefile` untuk menentukan apakah file kunci yang diberikan berisi materi kunci nyata atau materi kunci PEM palsu.

Example: Validasi kunci pribadi nyata

Perintah ini menegaskan bahwa file bernama `privateKey.pem` berisi materi kunci yang nyata.

```
Command: IsValidKeyHandlefile -f privateKey.pem
```



```
Input key file has real private key
```

Example: Membatalkan kunci PEM palsu

Perintah ini menegaskan bahwa file bernama `caviumKey.pem` berisi materi kunci PEM palsu yang terbuat dari handel kunci 15.

```
Command: IsValidKeyHandlefile -f caviumKey.pem
```

```
Input file has invalid key handle: 15
```

Parameter

Perintah ini membawa parameter berikut.

-h

Menampilkan bantuan baris perintah untuk perintah.

Wajib: Ya

-f

Menentukan file kunci pribadi RSA yang akan diperiksa untuk materi kunci yang valid.

Wajib: Ya

Topik terkait

- [getCaviumPrivKunci](#)
- [SSL/TLS Offload di Linux menggunakan Tomcat](#)
- [SSL/TLS Offload di Linux menggunakan NGINX atau Apache](#)

Buat daftar atribut AWS CloudHSM kunci menggunakan KMU

Gunakan `listAttributes` perintah di AWS CloudHSM `key_mgmt_util` untuk daftar atribut AWS CloudHSM kunci dan konstanta yang mewakili mereka. Anda menggunakan konstanta ini untuk mengidentifikasi atribut di [getAttribute](#) dan perintah [setAttribute](#). Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [Masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

Perintah ini tidak memiliki parameter.

```
listAttributes
```

Contoh

Perintah ini berisi daftar atribut kunci yang bisa Anda dapatkan dan ubah dalam `key_mgmt_util` dan konstanta yang mewakilinya. Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

Untuk mewakili semua atribut dalam [getAttribute](#) di `key_mgmt_util`, gunakan 512.

Command: **listAttributes**

Following are the possible attribute values for `getAttribute`:

OBJ_ATTR_CLASS	= 0
OBJ_ATTR_TOKEN	= 1
OBJ_ATTR_PRIVATE	= 2
OBJ_ATTR_LABEL	= 3
OBJ_ATTR_KEY_TYPE	= 256
OBJ_ATTR_ENCRYPT	= 260
OBJ_ATTR_DECRYPT	= 261
OBJ_ATTR_WRAP	= 262
OBJ_ATTR_UNWRAP	= 263
OBJ_ATTR_SIGN	= 264
OBJ_ATTR_VERIFY	= 266
OBJ_ATTR_LOCAL	= 355
OBJ_ATTR_MODULUS	= 288
OBJ_ATTR_MODULUS_BITS	= 289
OBJ_ATTR_PUBLIC_EXPONENT	= 290
OBJ_ATTR_VALUE_LEN	= 353
OBJ_ATTR_EXTRACTABLE	= 354
OBJ_ATTR_KCV	= 371

Topik terkait

- [listAttributes](#) di `cloudhsm_mgmt_util`

- [getAttribute](#)
- [setAttribute](#)
- [Referensi Atribut Utama](#)

Daftar semua AWS CloudHSM pengguna yang menggunakan KMU

Gunakan `listUsers` perintah di AWS CloudHSM `key_mgmt_util` untuk mendapatkan pengguna dalam modul keamanan perangkat keras (HSM), bersama dengan tipe pengguna dan atribut lainnya.

Di `key_mgmt_util`, `ListUsers` mengembalikan output yang mewakili HSMs semua dalam cluster, bahkan jika mereka tidak konsisten. Untuk mendapatkan informasi tentang pengguna di setiap HSM, gunakan [listUsers](#) di `cloudhsm_mgmt_util`.

Perintah pengguna di `key_mgmt_util`, `listUsers` dan [getKeyInfo](#), adalah perintah hanya-baca yang pengguna kriptografi () memiliki izin untuk dijalankan. CUs Perintah manajemen pengguna yang tersisa adalah bagian dari `cloudhsm_mgmt_util`. Perintah itu dijalankan oleh petugas kriptografi (CO) yang memiliki izin manajemen pengguna.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna kriptografi (CU).

Sintaksis

```
listUsers
```

```
listUsers -h
```

Contoh

Perintah ini mencantumkan pengguna HSMs di cluster dan atributnya. Anda dapat menggunakan `User ID` atribut untuk mengidentifikasi pengguna dalam perintah lain, seperti [FindKey](#), `getAttribute`, dan [getKeyInfo](#).

```
Command: listUsers
```

```
Number Of Users found 4
```

Index	User ID	User Type	User Name	MofnPubKey
LoginFailureCnt	2FA			

1	1	PCO	admin	NO
0	NO			
2	2	AU	app_user	NO
0	NO			
3	3	CU	alice	YES
0	NO			
4	4	CU	bob	NO
0	NO			
5	5	CU	trent	YES
0	NO			

Cfm3ListUsers returned: 0x00 : HSM Return: SUCCESS

Output mencakup atribut pengguna berikut:

- ID Pengguna: Mengidentifikasi pengguna di `key_mgmt_util` dan [cloudhsm_mgmt_util](#).
- [Jenis pengguna](#): Menentukan operasi yang pengguna dapat lakukan pada HSM.
- Nama Pengguna: Menampilkan nama mudah diingat yang ditetapkan pengguna untuk pengguna.
- MofnPubKey: Menunjukkan apakah pengguna telah mendaftarkan key pair untuk menandatangani token [otentikasi kuorum](#).
- LoginFailureCnt: Menunjukkan berapa kali pengguna tidak berhasil masuk.
- 2FA: Menunjukkan bahwa pengguna telah mengaktifkan autentikasi multi-faktor.

Parameter

-h

Menampilkan bantuan untuk perintah.

Wajib: Ya

Topik terkait

- [listUsers](#) di `cloudhsm_mgmt_util`
- [FindKey](#)
- [getAttribute](#)
- [getKeyInfo](#)

Masuk dan keluar dari HSM menggunakan AWS CloudHSM KMU

Gunakan `logoutHSM` perintah `loginHSM` dan di AWS CloudHSM `key_mgmt_util` untuk masuk dan keluar dari modul keamanan perangkat keras (HSM) dalam sebuah cluster. Setelah masuk ke HSMs, Anda dapat menggunakan `key_mgmt_util` untuk melakukan berbagai operasi manajemen kunci, termasuk pembuatan kunci publik dan pribadi, sinkronisasi, dan pembungkus.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#). [Untuk mengelola kunci dengan key_mgmt_util, Anda harus masuk ke HSMs sebagai pengguna krypto \(CU\)](#).

Note

Jika Anda melebihi lima upaya masuk yang salah, akun Anda terkunci. Jika Anda membuat klaster sebelum Februari 2018, akun Anda terkunci setelah 20 upaya login salah. Untuk membuka akun, petugas kriptografi (CO) harus mengatur ulang kata sandi Anda menggunakan perintah [changePswd](#) di `cloudhsm_mgmt_util`.

Jika Anda memiliki lebih dari satu HSM di klaster Anda, Anda mungkin diizinkan untuk upaya login salah tambahan sebelum akun Anda terkunci. Ini karena klien CloudHSM menyeimbangkan beban di berbagai jenis HSMs. Oleh karena itu, upaya login mungkin tidak dimulai pada HSM yang sama setiap saat. Jika Anda menguji fungsionalitas ini, kami sarankan Anda melakukannya pada sebuah klaster dengan hanya satu HSM aktif.

Sintaksis

```
loginHSM -h

loginHSM -u <user type>
          { -p | -hpswd } <password>
          -s <username>
```

Contoh

Contoh ini menunjukkan cara masuk dan keluar dari HSMs cluster dengan `logoutHSM` perintah `loginHSM` dan.

Example: Masuk ke HSMs

Perintah ini mencatat Anda ke HSMs sebagai pengguna kriptografi (CU) dengan nama pengguna `example_user` dan kata sandi `aws`. Output menunjukkan bahwa Anda telah masuk ke semua HSMs di cluster.

```
Command: loginHSM -u CU -s example_user -p aws
```

```
Cfm3LoginHSM returned: 0x00 : HSM Return: SUCCESS
```

Cluster Status

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Example: Masuk dengan kata sandi tersembunyi

Perintah ini sama dengan instans di atas, kecuali kali ini Anda menentukan bahwa sistem harus menyembunyikan kata sandinya.

```
Command: loginHSM -u CU -s example_user -hpswd
```

Sistem meminta kata sandi Anda. Anda memasukkan kata sandi, sistem menyembunyikan kata sandi, dan output menunjukkan bahwa perintah berhasil dan bahwa Anda telah terhubung ke HSMs.

```
Enter password:
```

```
Cfm3LoginHSM returned: 0x00 : HSM Return: SUCCESS
```

Cluster Status

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Command:
```

Example: Keluar dari HSMs

Perintah ini membuat Anda keluar dari file HSMs. Output menunjukkan bahwa Anda telah keluar dari semua HSMs di cluster.

```
Command: logoutHSM
```

```
Cfm3LogoutHSM returned: 0x00 : HSM Return: SUCCESS
```

Cluster Status

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Parameter

-h

Menampilkan bantuan untuk perintah ini.

-u

Menentukan jenis pengguna login. Untuk menggunakan `key_mgmt_util`, Anda harus login sebagai CU.

Wajib: Ya

-s

Menentukan nama pengguna login.

Wajib: Ya

{-p | -hpswd}

Tentukan kata sandi login dengan `-p`. Kata sandi muncul di plaintext saat Anda mengetikkannya. Untuk menyembunyikan kata sandi Anda, gunakan opsi parameter `-hpswd`, bukan `-p`, dan ikuti prompt.

Wajib: Ya

Topik terkait

- [keluar](#)

Mengatur atribut AWS CloudHSM kunci menggunakan KMU

Gunakan `setAttribute` perintah di AWS CloudHSM `key_mgmt_util` untuk mengonversi kunci yang hanya valid di sesi saat ini menjadi kunci persisten yang ada sampai Anda menghapusnya. Hal ini

dilakukan dengan mengubah nilai atribut token kunci (OBJ_ATTR_TOKEN) dari palsu (0) menjadi benar (1). Anda hanya dapat mengubah atribut kunci yang Anda miliki.

Anda juga dapat menggunakan `setAttribute` di `cloudhsm_mgmt_util` untuk mengubah label, membungkus, membuka, mengenkripsi, dan mendekripsi atribut.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
setAttribute -h  
  
setAttribute -o <object handle>  
              -a 1
```

Contoh

Contoh ini menunjukkan cara mengonversi kunci sesi menjadi kunci persisten.

Perintah pertama menggunakan `-sess` parameter [genSymKey](#) untuk membuat kunci AES 192-bit yang hanya valid di sesi saat ini. Output menunjukkan bahwa handle kunci dari kunci sesi baru adalah 262154.

```
Command: genSymKey -t 31 -s 24 -l tmpAES -sess  
  
Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS  
  
Symmetric Key Created.  Key Handle: 262154  
  
Cluster Error Status  
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

Perintah ini menggunakan [findKey](#) untuk menemukan kunci sesi dalam sesi saat ini. Output memverifikasi bahwa kunci 262154 adalah kunci sesi.

```
Command: findKey -sess 1  
  
Total number of keys present 1
```



```
number of keys matched from start index 0::0
262154
```

```
Cluster Error Status
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Perintah ini menggunakan `setAttribute` untuk mengonversi kunci 262154 dari kunci sesi ke kunci persisten. Untuk melakukannya, perintah mengubah nilai atribut token (`OBJ_ATTR_TOKEN`) kunci dari 0 (palsu) ke 1 (benar). Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

Perintah menggunakan parameter `-o` untuk menentukan handel kunci (262154) dan parameter `-a` untuk menentukan konstanta yang mewakili atribut token (1). Ketika Anda menjalankan perintah, perintah tersebut akan meminta nilai atribut token. Satu-satunya nilai yang valid adalah 1 (true); nilai untuk kunci persisten.

```
Command: setAttribute -o 262154 -a 1
```

```
This attribute is defined as a boolean value.
```

```
Enter the boolean attribute value (0 or 1):1
```

```
Cfm3SetAttribute returned: 0x00 : HSM Return: SUCCESS
```

```
Cluster Error Status
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Untuk mengonfirmasi bahwa kunci 262154 sekarang persisten, perintah ini menggunakan `findKey` untuk mencari kunci sesi (`-sess 1`) dan kunci persisten (`-sess 0`). Kali ini, perintah tidak menemukan kunci sesi, tetapi mengembalikan 262154 dalam daftar kunci persisten.

```
Command: findKey -sess 1
```

```
Total number of keys present 0
```

```
Cluster Error Status
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

```
Command: findKey -sess 0
```

```
Total number of keys present 5
```

```
number of keys matched from start index 0::4
```

```
6, 7, 524296, 9, 262154
```

```
Cluster Error Status
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Parameter

-h

Menampilkan bantuan untuk perintah.

Wajib: Ya

-o

Menentukan handel kunci dari kunci target. Anda hanya dapat menentukan satu kunci pada setiap perintah. Untuk mendapatkan handel kunci dari suatu kunci, gunakan [findKey](#).

Wajib: Ya

-a

Menentukan konstanta yang mewakili atribut yang ingin Anda ubah. Satu-satunya nilai yang valid adalah 1, yang mewakili atribut token, OBJ_ATTR_TOKEN.

Untuk mendapatkan atribut dan nilai integernya, gunakan [listAttributes](#).

Wajib: Ya

Topik terkait

- [setAttribute](#) di cloudhsm_mgmt_util

- [getAttribute](#)
- [ListAttributes](#)
- [Referensi Atribut Utama](#)

Hasilkan tanda tangan menggunakan AWS CloudHSM KMU

Gunakan sign perintah di AWS CloudHSM key_mgmt_util untuk menggunakan kunci pribadi yang dipilih untuk menghasilkan tanda tangan untuk file.

Untuk menggunakan sign, Anda harus terlebih dahulu memiliki kunci privat di HSM Anda. Anda dapat menghasilkan kunci privat dengan perintah [genSymKey](#), [genRSAKeyPair](#), atau [genECCKeyPair](#). Anda juga dapat mengimpor satu dengan perintah [importPrivateKey](#). Untuk informasi lebih lanjut, lihat [Hasilkan Kunci](#).

Perintah sign menggunakan mekanisme penandatanganan yang ditunjuk pengguna, diwakili oleh integer, untuk menandatangani file pesan. Untuk daftar mekanisme penandatanganan yang mungkin terjadi, lihat [Parameter](#).

Sebelum Anda menjalankan perintah key_mgmt_util, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna kriptografi (CU).

Sintaksis

```
sign -h

sign -f <file name>
    -k <private key handle>
    -m <signature mechanism>
    -out <signed file name>
```

Contoh

Contoh ini menunjukkan cara menggunakan sign untuk menandatangani file.

Example: Tanda tangani file

Perintah ini menandatangani sebuah file bernama messageFile dengan kunci privat dengan handle 266309. Perintah ini menggunakan mekanisme penandatanganan SHA256_RSA_PKCS (1) dan menyimpan file bertanda tangan yang dihasilkan sebagai signedFile.

```
Command: sign -f messageFile -k 266309 -m 1 -out signedFile
```

```
Cfm3Sign returned: 0x00 : HSM Return: SUCCESS
```

```
signature is written to file signedFile
```

Cluster Error Status

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Parameter

Perintah ini membawa parameter berikut.

-f

Nama file yang akan ditandatangani.

Wajib: Ya

-k

Handel kunci privat yang akan digunakan untuk penandatanganan.

Wajib: Ya

-m

Integer yang mewakili mekanisme penandatanganan yang akan digunakan untuk penandatanganan. Mekanisme yang mungkin sesuai dengan integer berikut:

Mekanisme Penandatanganan	Integer yang Sesuai
SHA1_RSA_PKCS	0
SHA256_RSA_PKCS	1
SHA384_RSA_PKCS	2
SHA512_RSA_PKCS	3
SHA224_RSA_PKCS	4

Mekanisme Penandatanganan	Integer yang Sesuai
SHA1_RSA_PKCS_PSS	5
SHA256_RSA_PKCS_PSS	6
SHA384_RSA_PKCS_PSS	7
SHA512_RSA_PKCS_PSS	8
SHA224_RSA_PKCS_PSS	9
ECDSA_SHA1	15
ECDSA_SHA224	16
ECDSA_SHA256	17
ECDSA_SHA384	18
ECDSA_SHA512	19

Wajib: Ya

-out

Nama file tempat file yang ditandatangani akan disimpan.

Wajib: Ya

Topik terkait

- [verifikasi](#)
- [importPrivateKey](#)
- [RSAKeypasangan gen](#)
- [ECCKeypasangan gen](#)
- [genSymKey](#)
- [Menghasilkan Kunci](#)

Buka AWS CloudHSM kunci menggunakan KMU

Gunakan `unWrapKey` perintah di alat AWS CloudHSM `key_mgmt_util` untuk mengimpor kunci simetris atau pribadi yang dibungkus (terenkripsi) dari file ke HSM. Hal ini dirancang untuk mengimpor kunci terenkripsi yang dibungkus oleh [wrapKey](#) di `key_mgmt_util`, tetapi juga dapat digunakan untuk membuka kunci yang dibungkus dengan alat-alat lain. Namun, dalam situasi tersebut, sebaiknya gunakan [PKCS #11](#) atau pustaka perangkat lunak [JCE](#) untuk membuka kunci.

Kunci yang diimpor berfungsi seperti kunci yang dihasilkan oleh AWS CloudHSM. Walau bagaimanapun, nilai [atribut OBJ_ATTR](#) adalah nol, yang menunjukkan bahwa kunci itu tidak dihasilkan secara lokal.

Setelah Anda mengimpor kunci, pastikan bahwa Anda menandai atau menghapus file kunci. Perintah ini tidak mencegah Anda dari mengimpor materi kunci yang sama beberapa kali. Hasilnya—beberapa kunci dengan handle kunci yang berbeda dan materi kunci yang sama—menyulitkan untuk melacak penggunaan materi utama dan mencegahnya melebihi batas kriptografinya.

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
unWrapKey -h

unWrapKey -f <key-file-name>
           -w <wrapping-key-handle>
           [-sess]
           [-min_srv <minimum-number-of-HSMs>]
           [-timeout <number-of-seconds>]
           [-aad <additional authenticated data filename>]
           [-tag_size <tag size>]
           [-iv_file <IV file>]
           [-attest]
           [-m <wrapping-mechanism>]
           [-t <hash-type>]
           [-nex]
           [-u <user id list>]
           [-m_value <number of users needed for approval>]
           [-noheader]
           [-l <key-label>]
           [-id <key-id>]
           [-kt <key-type>]
```

```
[-kc <key-class>]  
[-i <unwrapping-IV>]
```

Contoh

Contoh-contoh ini menunjukkan cara menggunakan `unWrapKey` untuk mengimpor kunci yang dibungkus dari file ke file HSMs. Pada contoh pertama, kita membuka kunci yang dibungkus dengan perintah `key_mgmt_util` [wrapKey](#), dan dengan demikian memiliki header. Pada contoh kedua, kita membuka kunci yang dibungkus di luar `key_mgmt_util`, dan dengan demikian tidak memiliki header.

Example: Buka kunci (dengan header)

Perintah ini mengimpor salinan kunci simetris 3DES yang dibungkus ke dalam HSM. Kuncinya dibuka dengan kunci AES dengan label 6, yang secara kriptografi identik dengan yang digunakan untuk membungkus kunci 3DES. Output menunjukkan bahwa kunci dalam file itu dibuka dan diimpor, dan bahwa handle kunci yang diimpor adalah 29.

```
Command: unWrapKey -f 3DES.key -w 6 -m 4  
  
Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS  
  
Key Unwrapped. Key Handle: 29  
  
Cluster Error Status  
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS  
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example: Buka kunci (tanpa header)

Perintah ini mengimpor salinan kunci simetris 3DES yang dibungkus ke dalam HSM. Kuncinya dibuka dengan kunci AES dengan label 6, yang secara kriptografi identik dengan yang digunakan untuk membungkus kunci 3DES. Karena kunci 3DES ini tidak dibungkus dengan `key_mgmt_util`, parameter `noheader` ditentukan, bersama dengan parameter yang menyertainya yang diperlukan: label kunci (`unwrapped3DES`), kelas kunci (4), dan jenis kunci (21). Output menunjukkan bahwa kunci dalam file itu dibuka dan diimpor, dan bahwa handle kunci yang diimpor adalah 8.

```
Command: unWrapKey -f 3DES.key -w 6 -noheader -l unwrapped3DES -kc 4 -kt 21 -m 4  
  
Cfm3CreateUnwrapTemplate2 returned: 0x00 : HSM Return: SUCCESS  
Cfm2UnWrapWithTemplate3 returned: 0x00 : HSM Return: SUCCESS
```

```
Key Unwrapped. Key Handle: 8
```

```
Cluster Error Status
```

```
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Parameter

-h

Menampilkan bantuan untuk perintah.

Wajib: Ya

-f

Jalur dan nama file yang berisi kunci yang dibungkus.

Wajib: Ya

-w

Menentukan kunci pembungkus. Masukkan handel kunci dari kunci AES atau kunci RSA pada HSM. Parameter ini diperlukan. Untuk menemukan handel kunci, gunakan perintah [findKey](#).

Untuk membuat kunci pembungkus, gunakan [genSymKey](#) untuk menghasilkan kunci AES (tipe 31) atau [gen RSAKey Pair untuk menghasilkan key pair](#) RSA (tipe 0). Jika Anda menggunakan pasangan kunci RSA, pastikan untuk membungkus kunci dengan salah satu kunci, dan buka dengan yang lain. Untuk memverifikasi bahwa kunci dapat digunakan sebagai kunci pembungkus, gunakan [getAttribute](#) untuk mendapatkan nilai atribut OBJ_ATTR_WRAP, yang diwakili oleh konstanta 262.

Wajib: Ya

-sess

Membuat kunci yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Gunakan parameter ini ketika Anda memerlukan kunci hanya sebentar, seperti kunci pembungkus yang mengenkripsi, dan kemudian dengan cepat mendekripsi, kunci lain. Jangan gunakan kunci sesi untuk mengenkripsi data yang mungkin perlu Anda dekripsi setelah sesi berakhir.

Untuk mengubah kunci sesi menjadi kunci (token) persisten, gunakan [setAttribute](#).

Default: Kunci persisten.

Wajib: Tidak

`-min_srv`

Menentukan jumlah minimum HSMs di mana kunci disinkronkan sebelum nilai `-timeout` parameter berakhir. Jika kunci tidak disinkronkan ke jumlah tertentu server dalam waktu yang ditentukan, kunci tidak dibuat.

AWS CloudHSM secara otomatis menyinkronkan setiap kunci ke setiap HSM di cluster. Untuk mempercepat proses Anda, tetapkan nilai kurang dari jumlah HSMs di cluster dan tetapkan nilai batas waktu yang rendah. `min_srv` Namun, perhatikan bahwa beberapa permintaan mungkin tidak menghasilkan kunci.

Default: 1

Wajib: Tidak

`-timeout`

Menentukan berapa lama (dalam detik) perintah menunggu kunci untuk disinkronkan dengan jumlah yang HSMs ditentukan oleh parameter. `min_srv`

Parameter ini hanya valid jika parameter `min_srv` juga digunakan dalam perintah.

Default: Tidak ada waktu habis. Perintah menunggu tanpa batas waktu dan kembali hanya ketika kunci disinkronkan ke jumlah minimum server.

Wajib: Tidak

`-attest`

Menjalankan pemeriksaan integritas yang memverifikasi bahwa firmware tempat klaster berjalan belum dirusak.

Default: Tidak ada pemeriksaan pengesahan.

Wajib: Tidak

`-nex`

Membuat kunci tidak dapat diekstrak. Kunci yang dihasilkan tidak dapat [diekspor dari HSM](#).

Default: Kunci dapat diekstrak.

Wajib: Tidak

-m

Nilai yang mewakili mekanisme pembungkus. CloudHSM mendukung mekanisme berikut:

Mekanisme	Nilai
AES_KEY_WRAP_PAD_PKCS5	4
NIST_AES_WRAP_NO_PAD	5
NIST_AES_WRAP_PAD	6
RSA_AES	7
RSA_OAEP (untuk ukuran data maksimum, lihat catatan di bagian ini nanti)	8
AES_GCM	10
CLOUDHSM_AES_GCM	11
RSA_PKCS(untuk ukuran data maksimum, lihat catatan nanti di bagian ini). Lihat catatan 1 di bawah untuk perubahan yang akan datang.	12

Wajib: Ya

Note

Saat menggunakan mekanisme RSA_OAEP pembungkus, ukuran kunci maksimum yang dapat Anda bungkus ditentukan oleh modulus kunci RSA dan panjang hash yang ditentukan sebagai berikut: $\text{Ukuran kunci maksimum} = \text{modulusLengthIn Bytes} - (2 * \text{Bytes}) - 2 * \text{hashLengthIn}$

Saat menggunakan mekanisme pembungkus RSA_PKCS, ukuran kunci maksimum yang dapat Anda bungkus ditentukan oleh modulus kunci RSA sebagai berikut: $\text{Ukuran kunci maksimum} = (\text{Bytes} - 11) * \text{modulusLengthIn}$

-t

Algoritma hash	Nilai
SHA1	2
SHA256	3
SHA384	4
SHA512	5
SHA224 (berlaku untuk mekanisme RSA_AES dan RSA_OAEP)	6

Wajib: Tidak

-noheader

Jika Anda membuka kunci yang dibungkus di luar `key_mgmt_util`, Anda harus menentukan parameter ini dan semua parameter terkait lainnya.

Wajib: Tidak

Note

Jika Anda menentukan parameter ini, Anda juga harus menentukan parameter berikut -noheader:

• -l

Menentukan label yang akan ditambahkan ke kunci terbuka.

Wajib: Ya

• -kc

Menentukan label yang akan ditambahkan ke kunci terbuka. Berikut ini adalah nilai yang dapat diterima:

3 = kunci privat dari pasangan kunci publik-privat

4 = kunci rahasia (simetris)

Wajib: Ya

- -kt

Menentukan jenis kunci yang akan dibuka. Berikut ini adalah nilai yang dapat diterima:

0 = RSA

1 = DSA

3 = ECC

16 = GENERIC_SECRET

21 = DES3

31 = AES

Wajib: Ya

Anda juga dapat secara opsional menentukan parameter berikut -noheader:

- -id

ID yang akan ditambahkan ke kunci terbuka.

Wajib: Tidak

- -i

Vektor inisialisasi (IV) pembuka bungkus yang akan digunakan.

Wajib: Tidak

[1] Sesuai dengan panduan NIST, ini tidak diizinkan untuk cluster dalam mode FIPS setelah 2023. Untuk cluster dalam mode non-FIPS, masih diperbolehkan setelah 2023. Lihat [Kepatuhan FIPS 140: Penutupan Mekanisme 2024](#) untuk detail.

Topik terkait

- [WrapKey](#)

- [exSymKey](#)
- [imSymKey](#)

Verifikasi tanda tangan file menggunakan AWS CloudHSM KMU

Gunakan `verify` perintah di AWS CloudHSM `key_mgmt_util` untuk mengonfirmasi apakah file telah ditandatangani oleh kunci yang diberikan atau tidak. Untuk melakukannya, `verify` membandingkan file yang ditandatangani dengan file sumber dan menganalisis apakah file tersebut secara kriptografi terkait berdasarkan kunci publik tertentu dan mekanisme penandatanganan. File dapat masuk AWS CloudHSM dengan [sign](#) operasi.

Mekanisme penandatanganan diwakili oleh integer yang tercantum di bagian [parameter](#).

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [masuk](#) ke HSM sebagai pengguna kriptografi (CU).

Sintaksis

```
verify -h

verify -f <message-file>
       -s <signature-file>
       -k <public-key-handle>
       -m <signature-mechanism>
```

Contoh

Contoh ini menunjukkan cara menggunakan `verify` untuk memeriksa apakah kunci publik tertentu digunakan untuk menandatangani file tertentu.

Example: Verifikasi tanda tangan file

Perintah ini mencoba untuk memverifikasi apakah sebuah file bernama `hardwareCert.crt` ditandatangani oleh kunci publik 262276 menggunakan mekanisme penandatanganan SHA256_RSA_PKCS untuk menghasilkan file bertanda tangan `hardwareCertSigned`. Karena parameter yang diberikan mewakili hubungan penandatanganan yang benar, perintah mengembalikan pesan sukses.

```
Command: verify -f hardwareCert.crt -s hardwareCertSigned -k 262276 -m 1
```

```
Signature verification successful
```

```
Cfm3Verify returned: 0x00 : HSM Return: SUCCESS
```

Example: Buktikan hubungan penandatanganan palsu

Perintah ini memverifikasi apakah sebuah file bernama `hardwareCert.crt` ditandatangani oleh kunci publik 262276 menggunakan mekanisme penandatanganan SHA256_RSA_PKCS untuk menghasilkan file bertandatangan `userCertSigned`. Karena parameter yang diberikan mewakili hubungan penandatanganan yang benar, perintah mengembalikan pesan sukses.

```
Command: verify -f hardwarecert.crt -s usercertsigned -k 262276 -m 1
```

```
Cfm3Verify returned: 0x1b
```

```
CSP Error: ERR_BAD_PKCS_DATA
```

Parameter

Perintah ini membawa parameter berikut.

-f

Nama file pesan asal.

Wajib: Ya

-s

Nama file yang akan ditandatangani.

Wajib: Ya

-k

Handel kunci publik yang diduga digunakan untuk menandatangani file.

Wajib: Ya

-m

Integer yang mewakili mekanisme penandatanganan yang diusulkan yang digunakan untuk menandatangani file. Mekanisme yang mungkin sesuai dengan integer berikut:

Mekanisme Penandatanganan	Integer yang Sesuai
SHA1_RSA_PKCS	0
SHA256_RSA_PKCS	1
SHA384_RSA_PKCS	2
SHA512_RSA_PKCS	3
SHA224_RSA_PKCS	4
SHA1_RSA_PKCS_PSS	5
SHA256_RSA_PKCS_PSS	6
SHA384_RSA_PKCS_PSS	7
SHA512_RSA_PKCS_PSS	8
SHA224_RSA_PKCS_PSS	9
ECDSA_SHA1	15
ECDSA_SHA224	16
ECDSA_SHA256	17
ECDSA_SHA384	18
ECDSA_SHA512	19

Wajib: Ya

Topik terkait

- [tanda](#)
- [GetCert](#)
- [Menghasilkan Kunci](#)

Ekspor AWS CloudHSM kunci menggunakan KMU

Gunakan `wrapKey` perintah di AWS CloudHSM `key_mgmt_util` untuk mengekspor salinan kunci simetris atau pribadi terenkripsi dari modul keamanan perangkat keras (HSM) ke file. Ketika Anda menjalankan `wrapKey`, Anda menentukan kunci untuk ekspor, kunci pada HSM untuk mengenkripsi (membungkus) kunci yang ingin Anda ekspor, dan file output.

Perintah `wrapKey` menuliskan kunci dienkripsi ke file yang Anda tentukan, tetapi tidak menghapus kunci dari HSM atau mencegah Anda dari menggunakannya dalam operasi kriptografi. Anda dapat mengekspor kunci yang sama beberapa kali.

Hanya pemilik kunci, yaitu pengguna krypto (CU) yang membuat kunci, dapat mengekspornya. Pengguna yang berbagi kunci dapat menggunakannya dalam operasi kriptografi, tetapi mereka tidak dapat mengekspornya.

Untuk mengimpor kunci terenkripsi kembali ke HSM, gunakan [unWrapKey](#). Untuk mengekspor kunci plaintext dari HSM, gunakan [exSymKey](#) atau [exportPrivateKey](#) sesuai kebutuhan.

[aesWrapUnwrap](#) Perintah tidak dapat mendekripsi (membuka) kunci yang mengenkripsi. `wrapKey`

Sebelum Anda menjalankan perintah `key_mgmt_util`, Anda harus [memulai key_mgmt_util](#) dan [Masuk](#) ke HSM sebagai pengguna krypto (CU).

Sintaksis

```
wrapKey -h

wrapKey -k <exported-key-handle>
        -w <wrapping-key-handle>
        -out <output-file>
        [-m <wrapping-mechanism>]
        [-aad <additional authenticated data filename>]
        [-t <hash-type>]
        [-noheader]
        [-i <wrapping IV>]
        [-iv_file <IV file>]
        [-tag_size <num_tag_bytes>]
```


Contoh

Example

Perintah ini mengekspor kunci simetris 192-bit Triple DES (3DES) (kunci handel 7). Perintah menggunakan kunci 256-bit AES di HSM (handel kunci 14) untuk membungkus kunci 7. Kemudian, perintah menulis kunci 3DES terenkripsi ke file `3DES-encrypted.key`.

Output menunjukkan bahwa kunci 7 (kunci 3DES) berhasil dibungkus dan ditulis ke file yang ditentukan. Kunci terenkripsi mempunyai panjang 307 byte.

```
Command: wrapKey -k 7 -w 14 -out 3DES-encrypted.key -m 4
```

```
Key Wrapped.
```

```
Wrapped Key written to file "3DES-encrypted.key" length 307
```

```
Cfm2WrapKey returned: 0x00 : HSM Return: SUCCESS
```

Parameter

-h

Menampilkan bantuan untuk perintah.

Wajib: Ya

-k

Handel kunci dari kunci yang ingin Anda ekspor. Masukkan handel kunci dari kunci simetris atau privat yang Anda miliki. Untuk menemukan handel kunci, gunakan perintah [findKey](#).

Untuk memverifikasi bahwa kunci dapat diekspor, gunakan perintah [getAttribute](#) untuk mendapatkan nilai atribut `OBJ_ATTR_EXTRACTABLE`, yang diwakili oleh konstanta 354. Untuk membantu menafsirkan atribut kunci, lihat [AWS CloudHSM referensi atribut kunci untuk KMU](#).

Anda hanya dapat mengekspor kunci yang Anda miliki. Untuk menemukan pemilik kunci, gunakan [getKeyInfo](#) perintah.

Wajib: Ya

-w

Menentukan kunci pembungkus. Masukkan handel kunci dari kunci AES atau kunci RSA pada HSM. Parameter ini diperlukan. Untuk menemukan handel kunci, gunakan perintah [findKey](#).

Untuk membuat kunci pembungkus, gunakan [genSymKey](#) untuk menghasilkan kunci AES (tipe 31) atau [gen RSAKey Pair untuk menghasilkan key pair](#) RSA (tipe 0). Jika Anda menggunakan pasangan kunci RSA, pastikan untuk membungkus kunci dengan salah satu kunci, dan buka dengan yang lain. Untuk memverifikasi bahwa kunci dapat digunakan sebagai kunci pembungkus, gunakan [getAttribute](#) untuk mendapatkan nilai atribut OBJ_ATTR_WRAP, yang diwakili oleh konstanta 262.

Wajib: Ya

-out

Jalur dan nama file output. Ketika perintah berhasil, file ini berisi kunci yang diekspor dalam plaintext. Jika file sudah ada, perintah akan menimpa tanpa peringatan.

Wajib: Ya

-m

Nilai yang mewakili mekanisme pembungkus. CloudHSM mendukung mekanisme berikut:

Mekanisme	Nilai
AES_KEY_WRAP_PAD_PKCS5	4
NIST_AES_WRAP_NO_PAD	5
NIST_AES_WRAP_PAD	6
RSA_AES	7
RSA_OAEP (untuk ukuran data maksimum, lihat catatan di bagian ini nanti)	8
AES_GCM	10
CLOUDHSM_AES_GCM	11

Mekanisme	Nilai
RSA_PKCS(untuk ukuran data maksimum, lihat catatan nanti di bagian ini). Lihat catatan 1 di bawah untuk perubahan yang akan datang.	12

Wajib: Ya

 Note

Saat menggunakan mekanisme RSA_OAEP pembungkus, ukuran kunci maksimum yang dapat Anda bungkus ditentukan oleh modulus kunci RSA dan panjang hash yang ditentukan sebagai berikut: Ukuran kunci maksimum = $(\text{modulusLengthInBytes} - 2 * \text{hashLengthInBytes} - 2)$.

Saat menggunakan mekanisme pembungkus RSA_PKCS, ukuran kunci maksimum yang dapat Anda bungkus ditentukan oleh modulus kunci RSA sebagai berikut: Ukuran kunci maksimum = $(\text{Bytes} - 11) * \text{modulusLengthIn}$

-t

Nilai yang mewakili algoritme hash. CloudHSM mendukung algoritme berikut:

Algoritma hash	Nilai
SHA1	2
SHA256	3
SHA384	4
SHA512	5
SHA224 (berlaku untuk mekanisme RSA_AES dan RSA_OAEP)	6

Wajib: Tidak

AAD

Nama file yang berisi AAD.

 Note

Hanya berlaku untuk mekanisme AES_GCM dan CLOUDHSM_AES_GCM.

Wajib: Tidak

-noheader

Menghilangkan header yang menentukan [atribut kunci](#) khusus CloudHSM. Gunakan parameter ini hanya jika Anda ingin membuka kunci dengan alat di luar key_mgmt_util.

Wajib: Tidak

-i

Vektor inisialisasi (IV) (nilai hex).

 Note

Hanya berlaku ketika diteruskan dengan -noheader parameter untuk CLOUDHSM_AES_KEY_WRAP, dan NIST_AES_WRAP mekanisme.

Wajib: Tidak

-iv_file

File tempat Anda ingin menuliskan nilai IV yang diperoleh sebagai tanggapan.

 Note

Berlaku hanya ketika dilewatkan dengan parameter -noheader untuk mekanisme AES_GCM.

Wajib: Tidak

-tag_size

Ukuran tanda yang akan disimpan bersama dengan blob terbungkus.

Note

Berlaku hanya ketika dilewatkan dengan parameter `-noheader` untuk mekanisme AES_GCM dan CLOUDHSM_AES_GCM. Ukuran tanda minimum adalah delapan.

Wajib: Tidak

[1] Sesuai dengan panduan NIST, ini tidak diizinkan untuk cluster dalam mode FIPS setelah 2023. Untuk cluster dalam mode non-FIPS, masih diperbolehkan setelah 2023. Lihat [Kepatuhan FIPS 140: Penutupan Mekanisme 2024](#) untuk detail.

Topik terkait

- [exSymKey](#)
- [imSymKey](#)
- [unWrapKey](#)

AWS CloudHSM referensi atribut kunci untuk KMU

Perintah AWS CloudHSM `key_mgmt_util` menggunakan konstanta untuk mewakili atribut kunci dalam modul keamanan perangkat keras (HSM). Topik ini dapat membantu Anda untuk mengidentifikasi atribut, menemukan konstanta yang mewakilinya dalam perintah, dan memahami nilai-nilainya.

Anda mengatur atribut kunci saat Anda membuatnya. Untuk mengubah atribut token, yang menunjukkan apakah kunci persisten atau hanya ada di sesi tersebut, gunakan perintah [setAttribute](#) dalam `key_mgmt_util`. Untuk mengubah label, membungkus, membuka, mengenkripsi, atau mendekripsi atribut, gunakan `setAttribute` di `cloudhsm_mgmt_util`.

Untuk mendapatkan daftar atribut dan konstanta mereka, gunakan [listAttributes](#). Untuk mendapatkan nilai atribut untuk kunci, gunakan [getAttribute](#).

Tabel berikut mencantumkan atribut kunci, konstantanya, dan nilai-nilai yang valid.

Atribut	Konstan	Nilai
OBJ_ATTR_ALL	512	Mewakili semua atribut.
OBJ_ATTR_ALWAYS_SENSITIVE	357	0: Salah. 1: Benar.
OBJ_ATTR_CLASS	0	2: Kunci publik dalam pasangan kunci publik—privat. 3: Kunci privat dalam pasangan kunci publik—privat. 4: Kunci rahasia (simetris).
OBJ_ATTR_DECRYPT	261	0: Salah. 1: Benar. Kuncinya dapat digunakan untuk mendekripsi data.
OBJ_ATTR_DERIVE	268	0: Salah. 1: Benar. Fungsi asal kunci.
OBJ_ATTR_DESTROYABLE	370	0: Salah. 1: Benar.
OBJ_ATTR_ENCRYPT	260	0: Salah. 1: Benar. Kuncinya dapat digunakan untuk mengenkripsi data.
OBJ_ATTR_EXTRACTABLE	354	0: Salah. 1: Benar. Kuncinya dapat diekspor dari HSMs

Atribut	Konstan	Nilai
OBJ_ATTR_ID	258	String yang ditentukan pengguna. Harus unik di klaster. Default-nya adalah string kosong.
OBJ_ATTR_KCV	371	Nilai pemeriksaan kunci. Untuk informasi lebih lanjut, lihat Detail tambahan .
OBJ_ATTR_KEY_TYPE	256	0: RSA 1: DSA. 3: EC. 16: GENERIC_SECRET. 18: RC4. 21: Tiga DES (3DES). 31: AE.
OBJ_ATTR_LABEL	3	String yang ditentukan pengguna. Tidak harus unik di klaster.
OBJ_ATTR_LOCAL	355	0. Salah. Kuncinya diimpor ke HSMs. 1: Benar.

Atribut	Konstan	Nilai
OBJ_ATTR_MODULUS	288	Modulus yang digunakan untuk menghasilkan pasangan kunci RSA Untuk kunci EC, nilai ini mewakili pengkodean DER dari nilai ANSI X9.62 “Q” dalam ECPoint format heksadesimal. Untuk jenis kunci lainnya, atribut ini tidak ada.
OBJ_ATTR_MODULUS_BITS	289	Panjang modulus yang digunakan untuk menghasilkan pasangan kunci RSA. Untuk kunci EC ini mewakili ID kurva elips yang digunakan untuk menghasilkan kunci. Untuk jenis kunci lainnya, atribut ini tidak ada.
OBJ_ATTR_NEVER_EXPORTABLE	356	0: Salah. 1: Benar. Kunci tidak dapat diekspor dari. HSMs
OBJ_ATTR_PUBLIC_EXPONENT	290	Eksponen publik yang digunakan untuk menghasilkan pasangan kunci RSA. Untuk jenis kunci lainnya, atribut ini tidak ada.

Atribut	Konstan	Nilai
OBJ_ATTR_PRIVATE	2	0: Salah. 1: Benar. Atribut ini menunjukkan apakah pengguna yang tidak terautentikasi dapat mendaftar atribut kunci. Karena penyedia PKCS#11 CloudHSM saat ini tidak mendukung sesi publik, semua kunci (termasuk kunci publik dalam pasangan kunci privat-publik) mengatur atribut ini ke 1.
OBJ_ATTR_SENSITIVE	259	0: Salah. Kunci publik dalam pasangan kunci publik-privat. 1: Benar.
OBJ_ATTR_SIGN	264	0: Salah. 1: Benar. Kuncinya bisa digunakan untuk penandatanganan (kunci privat).
OBJ_ATTR_TOKEN	1	0: Salah. Kunci sesi 1: Benar. Kunci yang persisten.
OBJ_ATTR_TRUSTED	134	0: Salah. 1: Benar.

Atribut	Konstan	Nilai
OBJ_ATTR_UNWRAP	263	0: Salah. 1: Benar. Kuncinya dapat digunakan untuk mendekripsi kunci.
OBJ_ATTR_UNWRAP_TEMPLATE	1073742354	Nilai harus menggunakan templat atribut yang diterapkan untuk setiap kunci terbuka menggunakan kunci pembungkus ini.
OBJ_ATTR_VALUE_LEN	353	Panjang kunci dalam byte.
OBJ_ATTR_VERIFY	266	0: Salah. 1: Benar. Kuncinya dapat digunakan untuk verifikasi (kunci publik).
OBJ_ATTR_WRAP	262	0: Salah. 1: Benar. Kuncinya bisa digunakan untuk mengenkripsi kunci.
OBJ_ATTR_WRAP_TEMPLATE	1073742353	Nilai harus menggunakan template atribut untuk mencocokkan kunci yang dibungkus menggunakan kunci pembungkus ini.
OBJ_ATTR_WRAP_WITH_TRUSTED	528	0: Salah. 1: Benar.

Detail Tambahan

Nilai pemeriksaan kunci (KCV)

Nilai pemeriksaan kunci (KCV) adalah hash 3-byte atau checksum dari kunci yang dihasilkan ketika HSM mengimpor atau menghasilkan kunci. Anda juga dapat menghitung KCV di luar HSM, seperti setelah Anda mengekspor kunci. Anda kemudian dapat membandingkan nilai KCV untuk mengonfirmasi identitas dan integritas kunci. Untuk mendapatkan KCV kunci, gunakan [getAttribute](#).

AWS CloudHSM menggunakan metode standar berikut untuk menghasilkan nilai cek kunci:

- Kunci simetris: 3 byte pertama hasil enkripsi blok nol dengan kunci.
- Pasangan kunci asimetris: 3 byte pertama dari hash SHA-1 dari kunci publik.
- Kunci HMAC: KCV untuk kunci HMAC tidak didukung saat ini.

Operasi pembongkaran dengan Klien AWS CloudHSM SDKs

Gunakan SDK Klien untuk melepaskan operasi kriptografi dari platform atau aplikasi berbasis bahasa ke modul keamanan perangkat keras (). HSMs

AWS CloudHSM menawarkan dua versi utama, dan Client SDK 5 adalah yang terbaru. Ini menawarkan berbagai keunggulan dibandingkan Client SDK 3 (seri sebelumnya). Untuk informasi selengkapnya, lihat [Manfaat SDK Klien 5](#). Untuk informasi tentang dukungan platform, lihat [AWS CloudHSM Platform yang didukung SDK Klien 5](#).

Topik berikut menjelaskan cara bekerja dengan AWS CloudHSM Klien SDKs.

AWS CloudHSM mendukung komponen-komponen berikut:

[the section called “Pustaka PKCS #11”](#)

PKCS #11 adalah standar untuk melakukan operasi kriptografi pada modul keamanan perangkat keras (). HSMs AWS CloudHSM menawarkan implementasi pustaka PKCS #11 yang sesuai dengan PKCS #11 versi 2.40.

[the section called “OpenSSL Dynamic Engine”](#)

AWS CloudHSM OpenSSL Dynamic Engine memungkinkan Anda untuk membongkar operasi kriptografi ke cluster CloudHSM Anda melalui OpenSSL API.

[the section called “Penyedia OpenSSL”](#)

Penyedia Mesin Dinamis OpenSSL memungkinkan Anda untuk membongkar operasi kriptografi ke kluster CloudHSM Anda melalui OpenSSL Provider API. AWS CloudHSM

[the section called “Penyedia JCE”](#)

Penyedia AWS CloudHSM JCE sesuai dengan Java Cryptographic Architecture (JCA). Penyedia memungkinkan Anda untuk melakukan operasi kriptografi pada HSM.

[the section called “Penyedia penyimpanan kunci \(KSP\)”](#)

AWS CloudHSM Klien untuk Windows termasuk penyedia CNG dan KSP.

Topik

- [Periksa versi SDK AWS CloudHSM Klien Anda](#)
- [Bandingkan AWS CloudHSM dukungan komponen SDK Klien](#)
- [Migrasi dari SDK AWS CloudHSM Klien 3 ke SDK Klien 5](#)
- [Menggunakan Client SDK 5 untuk bekerja dengan AWS CloudHSM](#)
- [Menggunakan versi SDK sebelumnya untuk bekerja dengan AWS CloudHSM](#)

Periksa versi SDK AWS CloudHSM Klien Anda

Gunakan perintah berikut untuk memverifikasi versi SDK Klien yang Anda gunakan. AWS CloudHSM

Amazon Linux

Gunakan perintah berikut ini.

```
rpm -qa | grep ^cloudhsm
```

Amazon Linux 2

Gunakan perintah berikut ini.

```
rpm -qa | grep ^cloudhsm
```

CentOS 6

Gunakan perintah berikut ini.

```
rpm -qa | grep ^cloudhsm
```

CentOS 7

Gunakan perintah berikut ini.

```
rpm -qa | grep ^cloudhsm
```

CentOS 8

Gunakan perintah berikut ini.

```
rpm -qa | grep ^cloudhsm
```

RHEL 6

Gunakan perintah berikut ini.

```
rpm -qa | grep ^cloudhsm
```

RHEL 7

Gunakan perintah berikut ini.

```
rpm -qa | grep ^cloudhsm
```

RHEL 8

Gunakan perintah berikut ini.

```
rpm -qa | grep ^cloudhsm
```

Ubuntu 16.04 LTS

Gunakan perintah berikut ini.

```
apt list --installed | grep ^cloudhsm
```

Ubuntu 18.04 LTS

Gunakan perintah berikut ini.

```
apt list --installed | grep ^cloudhsm
```

Ubuntu 20.04 LTS

Gunakan perintah berikut ini.

```
apt list --installed | grep ^cloudhsm
```

Windows Server

Gunakan perintah berikut ini.

```
wmic product get name,version
```

Bandingkan AWS CloudHSM dukungan komponen SDK Klien

Selain alat baris perintah, Client SDK 3 berisi komponen yang memungkinkan operasi kriptografi off-loading ke HSM dari berbagai platform atau aplikasi berbasis bahasa. Klien SDK 5 memiliki paritas dengan Klien SDK 3, kecuali belum mendukung penyedia CNG dan KSP. Tabel berikut membandingkan ketersediaan komponen di SDK Klien 3 dan SDK Klien 5.

Komponen	SDK Klien 5	SDK Klien 3
Pustaka PKCS #11	Ya	Ya
Penyedia JCE	Ya	Ya
OpenSSL Dynamic Engine	Ya	Ya
Penyedia Penyimpanan Kunci (KSP)	Ya	Ya
Utilitas Manajemen CloudHSM (CMU) ¹	Ya	Ya
Utilitas Manajemen Kunci (KMU) ¹	Ya	Ya
Alat konfigurasi	Ya	Ya

[1] Komponen CMU dan KMU disertakan dalam CloudHSM CLI dengan Client SDK 5.

Bagian berikut menjelaskan komponen.

Pustaka PKCS #11

PKCS #11 adalah standar untuk melakukan operasi kriptografi pada modul keamanan perangkat keras (. HSMs AWS CloudHSM menawarkan implementasi pustaka PKCS #11 yang sesuai dengan PKCS #11 versi 2.40.

- Untuk Client SDK 3, pustaka PKCS #11 adalah komponen Linux saja yang cocok dengan dukungan basis Linux. Untuk informasi selengkapnya, lihat [the section called “Dukungan Linux untuk Client SDK 3”](#).

- Untuk Client SDK 5, pustaka PKCS #11 adalah komponen lintas platform yang cocok dengan dukungan dasar Linux dan Windows Client SDK 5. Untuk informasi selengkapnya, lihat [the section called “Dukungan Linux untuk Client SDK 5”](#) dan [the section called “Dukungan Windows untuk Klien SDK 5”](#).

Utilitas Manajemen CloudHSM (CMU)

Alat baris perintah CloudHSM Management Utility (CMU) membantu petugas kripto mengelola pengguna di HSMs. Ini termasuk alat yang membuat, menghapus, dan mendaftarkan pengguna serta mengubah kata sandi pengguna. Untuk informasi selengkapnya, lihat [AWS CloudHSM Utilitas Manajemen \(CMU\)](#).

Utilitas Manajemen Kunci (KMU)

Key Management Utility (KMU) adalah alat baris perintah yang membantu pengguna kripto (CU) mengelola kunci pada modul keamanan perangkat keras (HSM). Untuk informasi selengkapnya, lihat [AWS CloudHSM Utilitas Manajemen Kunci \(KMU\)](#).

Penyedia JCE

Penyedia AWS CloudHSM JCE sesuai dengan Java Cryptographic Architecture (JCA). Penyedia memungkinkan Anda untuk melakukan operasi kriptografi pada HSM.

Penyedia JCE adalah satu-satunya komponen Linux yang cocok dengan dukungan dasar Linux. Untuk informasi selengkapnya, lihat [the section called “Dukungan Linux untuk Client SDK 3”](#).

- Untuk Klien SDK 3 membutuhkan OpenJDK 1.8

OpenSSL Dynamic Engine

AWS CloudHSM OpenSSL Dynamic Engine memungkinkan Anda untuk membongkar operasi kriptografi ke cluster CloudHSM Anda melalui OpenSSL API.

- Untuk Client SDK 3, OpenSSL Dynamic Engine adalah komponen Linux saja yang tidak cocok dengan dukungan basis Linux. Lihat pengecualian di bawah ini.
 - Membutuhkan OpenSSL 1.0.2[f+]

Platform yang tidak didukung:

- CentOS 8
- Red Hat Enterprise Linux (RHEL) 8
- Ubuntu 18.04 LTS

Platform ini dikirim dengan versi OpenSSL yang tidak kompatibel dengan OpenSSL Dynamic Engine untuk Klien SDK 3. AWS CloudHSM mendukung platform ini dengan OpenSSL Dynamic Engine untuk Klien SDK 5.

- Untuk Client SDK 5, OpenSSL Dynamic Engine adalah komponen Linux saja yang membutuhkan OpenSSL 1.0.2, 1.1.1, atau 3.x.

Penyedia penyimpanan kunci (KSP)

Key Storage Provider (KSP) adalah API kriptografi khusus untuk sistem operasi Microsoft Windows.

Untuk Client SDK 3, penyedia CNG dan KSP adalah komponen khusus Windows yang cocok dengan dukungan dasar Windows. Untuk informasi selengkapnya, lihat [Dukungan Windows untuk AWS CloudHSM Client SDK 3](#).

Untuk Client SDK 5, Key Storage Provider (KSP) adalah komponen khusus Windows yang cocok dengan dukungan dasar Windows. Untuk informasi selengkapnya, lihat [Dukungan Windows untuk AWS CloudHSM Klien SDK 5](#).

Migrasi dari SDK AWS CloudHSM Klien 3 ke SDK Klien 5

Pada tahun AWS CloudHSM, aplikasi pelanggan melakukan operasi kriptografi menggunakan AWS CloudHSM Client Software Development Kit (SDK). Client SDK 5 adalah SDK utama yang terus memiliki fitur baru dan dukungan platform yang ditambahkan ke dalamnya.

Client SDK 3 mencakup dua alat baris perintah terpisah: CMU untuk mengelola pengguna dan KMU untuk mengelola kunci dan melakukan operasi dengan kunci. Klien SDK 5 mengkonsolidasikan fungsi CMU dan KMU (alat yang ditawarkan dengan Client SDK 3) ke dalam satu alat, yaitu. [AWS CloudHSM Antarmuka Baris Perintah \(CLI\)](#) Operasi manajemen pengguna dapat ditemukan di bawah subperintah [Kategori pengguna di CloudHSM CLI](#) dan [Kategori kuorum di CloudHSM CLI](#). Operasi manajemen kunci dapat ditemukan di bawah [subperintah kunci](#), dan operasi kriptografi dapat ditemukan di bawah subperintah [crypto](#). Lihat [Referensi untuk perintah CloudHSM CLI](#) daftar lengkap perintah.

Untuk manfaat migrasi, lihat [Manfaat SDK AWS CloudHSM Klien 5](#).

Lihat topik berikut untuk petunjuk terperinci tentang migrasi dari SDK Klien 3 ke SDK Klien 5. Versi terbaru AWS CloudHSM Client SDK adalah 5.16.

- [Migrasi pustaka AWS CloudHSM PKCS #11 Anda dari Client SDK 3 ke Client SDK 5](#)
- [Migrasikan OpenSSL Dynamic Engine Anda AWS CloudHSM dari Client SDK 3 ke Client SDK 5](#)
- [Migrasikan Penyedia Penyimpanan Kunci \(KSP\) Anda dari SDK AWS CloudHSM Klien 3 ke SDK Klien 5](#)
- [Migrasikan penyedia JCE Anda dari AWS CloudHSM Client SDK 3 ke Client SDK 5](#)

Untuk fungsionalitas atau kasus penggunaan yang tidak didukung oleh CloudHSM CLI, hubungi [AWS Dukungan](#)

Migrasi pustaka AWS CloudHSM PKCS #11 Anda dari Client SDK 3 ke Client SDK 5

Gunakan topik ini untuk memigrasikan [pustaka AWS CloudHSM PKCS #11](#) Anda dari SDK Klien 3 ke SDK Klien 5. Untuk manfaat migrasi, lihat [Manfaat SDK AWS CloudHSM Klien 5](#).

Pada tahun AWS CloudHSM, aplikasi pelanggan melakukan operasi kriptografi menggunakan AWS CloudHSM Client Software Development Kit (SDK). Client SDK 5 adalah SDK utama yang terus memiliki fitur baru dan dukungan platform yang ditambahkan ke dalamnya.

Untuk meninjau petunjuk migrasi untuk semua penyedia, lihat [Migrasi dari SDK AWS CloudHSM Klien 3 ke SDK Klien 5](#).

Bersiaplah dengan mengatasi perubahan yang melanggar

Tinjau perubahan yang melanggar ini dan perbarui aplikasi Anda di lingkungan pengembangan yang sesuai.

Mekanisme pembungkus telah berubah

Mekanisme SDK 3 klien	Mekanisme SDK 5 Klien Setara
CKM_AES_KEY_WRAP	CKM_CLOUDHSM_AES_KEY_WRAP_P KCS5_PAD

Mekanisme SDK 3 klien	Mekanisme SDK 5 Klien Setara
CKM_AES_KEY_WRAP_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD
CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD
CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD
CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD

ECDH

Di Client SDK 3, Anda dapat menggunakan ECDH dan menentukan KDF. Fungsionalitas ini saat ini tidak tersedia di Client SDK 5. Jika aplikasi Anda membutuhkan fungsi ini, silakan hubungi [dukungan](#).

Pegangan kunci sekarang khusus untuk sesi

Untuk berhasil menggunakan handel kunci di SDK Klien 5, Anda harus mendapatkan handel kunci setiap kali Anda menjalankan aplikasi. Jika Anda memiliki aplikasi yang ada yang akan menggunakan pegangan kunci yang sama di sesi yang berbeda, Anda harus memodifikasi kode Anda untuk mendapatkan pegangan kunci setiap kali Anda menjalankan aplikasi. Untuk informasi tentang mengambil pegangan kunci, lihat contoh [AWS CloudHSM PKCS #11 ini](#). Perubahan ini sesuai dengan [spesifikasi PKCS #11 2.40](#).

Migrasi ke SDK Klien 5

Ikuti petunjuk di bagian ini untuk bermigrasi dari Client SDK 3 ke Client SDK 5.

Note

Amazon Linux, Ubuntu 16.04, Ubuntu 18.04, CentOS 6, CentOS 8, dan RHEL 6 saat ini tidak didukung dengan Client SDK 5. Jika saat ini Anda menggunakan salah satu platform ini dengan Client SDK 3, Anda harus memilih platform yang berbeda saat bermigrasi ke Client SDK 5.

1. Copot pemasangan pustaka PKCS #11 untuk Client SDK 3.

Amazon Linux 2

```
$ sudo yum remove cloudhsm-client-pkcs11
```

CentOS 7

```
$ sudo yum remove cloudhsm-client-pkcs11
```

RHEL 7

```
$ sudo yum remove cloudhsm-client-pkcs11
```

RHEL 8

```
$ sudo yum remove cloudhsm-client-pkcs11
```

Ubuntu 16.04 LTS

```
$ sudo apt remove cloudhsm-client-pkcs11
```

Ubuntu 18.04 LTS

```
$ sudo apt remove cloudhsm-client-pkcs11
```

2. Hentikan Daemon Klien untuk SDK Klien 3.

Amazon Linux 2

```
$ sudo service cloudhsm-client stop
```

CentOS 7

```
$ sudo service cloudhsm-client stop
```

RHEL 7

```
$ sudo service cloudhsm-client stop
```

RHEL 8

```
$ sudo service cloudhsm-client stop
```

Ubuntu 16.04 LTS

```
$ sudo systemctl stop cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo systemctl stop cloudhsm-client
```

3. Copot pemasangan Daemon Klien untuk SDK Klien 3.

Amazon Linux 2

```
$ sudo yum remove cloudhsm-client
```

CentOS 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 8

```
$ sudo yum remove cloudhsm-client
```

Ubuntu 16.04 LTS

```
$ sudo apt remove cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo apt remove cloudhsm-client
```

Note

Konfigurasi khusus perlu diaktifkan lagi.

4. Instal pustaka PKCS #11 Client SDK dengan mengikuti langkah-langkahnya. [Instal pustaka PKCS #11 untuk AWS CloudHSM Client SDK 5](#)
5. Client SDK 5 memperkenalkan format file konfigurasi baru dan alat bootstrap baris perintah. Untuk mem-bootstrap pustaka Client SDK 5 PKCS #11 Anda, ikuti petunjuk yang tercantum dalam panduan pengguna di bawah. [Bootstrap Klien SDK](#)
6. Di lingkungan pengembangan Anda, uji aplikasi Anda. Lakukan pembaruan pada kode yang ada untuk menyelesaikan perubahan yang melanggar sebelum migrasi terakhir Anda.

Topik terkait

- [Praktik terbaik untuk AWS CloudHSM](#)

Migrasikan OpenSSL Dynamic Engine Anda AWS CloudHSM dari Client SDK 3 ke Client SDK 5

Gunakan topik ini untuk memigrasikan [OpenSSL Dynamic](#) Engine Anda AWS CloudHSM dari Client SDK 3 ke Client SDK 5. Untuk manfaat migrasi, lihat [Manfaat SDK AWS CloudHSM Klien 5](#).

Pada tahun AWS CloudHSM, aplikasi pelanggan melakukan operasi kriptografi menggunakan AWS CloudHSM Client Software Development Kit (SDK). Client SDK 5 adalah SDK utama yang terus memiliki fitur baru dan dukungan platform yang ditambahkan ke dalamnya.

Note

Pembuatan angka acak saat ini tidak didukung di Client SDK 5 dengan OpenSSL Dynamic Engine.

Untuk meninjau petunjuk migrasi untuk semua penyedia, lihat [Migrasi dari SDK AWS CloudHSM Klien 3 ke SDK Klien 5](#).

Migrasi ke SDK Klien 5

Ikuti petunjuk di bagian ini untuk bermigrasi dari Client SDK 3 ke Client SDK 5.

Note

Amazon Linux, Ubuntu 16.04, Ubuntu 18.04, CentOS 6, CentOS 8, dan RHEL 6 saat ini tidak didukung dengan Client SDK 5. Jika saat ini Anda menggunakan salah satu platform ini dengan Client SDK 3, Anda harus memilih platform yang berbeda saat bermigrasi ke Client SDK 5.

1. Copot pemasangan OpenSSL Dynamic Engine untuk Client SDK 3.

Amazon Linux 2

```
$ sudo yum remove cloudhsm-client-dyn
```

CentOS 7

```
$ sudo yum remove cloudhsm-client-dyn
```

RHEL 7

```
$ sudo yum remove cloudhsm-client-dyn
```

RHEL 8

```
$ sudo yum remove cloudhsm-client-dyn
```

Ubuntu 16.04 LTS

```
$ sudo apt remove cloudhsm-client-dyn
```

Ubuntu 18.04 LTS

```
$ sudo apt remove cloudhsm-client-dyn
```

2. Hentikan Daemon Klien untuk SDK Klien 3.

Amazon Linux 2

```
$ sudo service cloudhsm-client stop
```

CentOS 7

```
$ sudo service cloudhsm-client stop
```

RHEL 7

```
$ sudo service cloudhsm-client stop
```

RHEL 8

```
$ sudo service cloudhsm-client stop
```

Ubuntu 16.04 LTS

```
$ sudo systemctl stop cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo systemctl stop cloudhsm-client
```

3. Copot pemasangan Daemon Klien untuk SDK Klien 3.

Amazon Linux 2

```
$ sudo yum remove cloudhsm-client
```


CentOS 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 8

```
$ sudo yum remove cloudhsm-client
```

Ubuntu 16.04 LTS

```
$ sudo apt remove cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo apt remove cloudhsm-client
```

Note

Konfigurasi khusus perlu diaktifkan lagi.

4. Instal Client SDK OpenSSL Dynamic Engine dengan mengikuti langkah-langkahnya. [Instal OpenSSL Dynamic Engine AWS CloudHSM untuk Client SDK 5](#)
5. Client SDK 5 memperkenalkan format file konfigurasi baru dan alat bootstrap baris perintah. Untuk bootstrap Client SDK 5 OpenSSL Dynamic Engine Anda, ikuti petunjuk yang tercantum dalam panduan pengguna di bawah. [Bootstrap Klien SDK](#)
6. Di lingkungan pengembangan Anda, uji aplikasi Anda. Lakukan pembaruan pada kode yang ada untuk menyelesaikan perubahan yang melanggar sebelum migrasi terakhir Anda.

Topik terkait

- [Praktik terbaik untuk AWS CloudHSM](#)

Migrasikan Penyedia Penyimpanan Kunci (KSP) Anda dari SDK AWS CloudHSM Klien 3 ke SDK Klien 5

Topik ini menjelaskan cara memigrasikan [Penyedia Penyimpanan Kunci \(KSP\)](#) Anda dari SDK AWS CloudHSM Klien 3 ke SDK Klien 5. Versi terbaru AWS CloudHSM Client SDK adalah 5.16. Untuk informasi tentang manfaat migrasi, lihat [Manfaat SDK AWS CloudHSM Klien 5](#).

Di AWS CloudHSM, Anda menggunakan AWS CloudHSM Client Software Development Kit (SDK) untuk melakukan operasi kriptografi. Client SDK 5 adalah SDK utama yang menerima fitur baru dan pembaruan dukungan platform.

Untuk petunjuk migrasi untuk semua penyedia, lihat [Migrasi dari SDK AWS CloudHSM Klien 3 ke SDK Klien 5](#).

Migrasi ke SDK Klien 5

1. Hentikan Daemon Klien untuk SDK Klien 3.

```
PS C:\> Stop-Service "AWS CloudHSM Client"
```

2. Instal Client SDK 5 Key Storage Provider (KSP) pada instance Windows Server Anda. Untuk petunjuk, lihat [Instal penyedia penyimpanan kunci \(KSP\) untuk AWS CloudHSM Client SDK 5](#).
3. Konfigurasi Client SDK 5 Key Storage Provider (KSP) Anda menggunakan format file konfigurasi baru dan alat bootstrap baris perintah. Untuk petunjuk, lihat [Bootstrap Klien SDK](#).
4. Penyedia Penyimpanan Kunci (KSP) untuk AWS CloudHSM Klien SDK 5 menyertakan mode SDK3 kompatibilitas untuk mendukung file referensi utama yang dihasilkan di SDK3. Untuk informasi selengkapnya, lihat [SDK3 mode kompatibilitas untuk Penyedia Penyimpanan Kunci \(KSP\) untuk AWS CloudHSM](#).

Note

Anda harus mengaktifkan mode SDK3 kompatibilitas saat menggunakan Client SDK 3 yang dihasilkan file referensi kunci dengan Client SDK 5.

Migrasi ke instance Windows Server baru

1. Selesaikan semua langkah di [Migrate to Client SDK 5](#) pada instance Windows Server baru Anda.

2. Periksa file referensi kunci yang ada

Pada instance Windows Server asli Anda, periksa file referensi kunci di `C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP\GlobalPartition`.

- Jika file referensi kunci ada, salin semua konten di bawah `C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP` termasuk `GlobalPartition` ke jalur direktori yang sama pada instance Windows Server baru Anda. Buat direktori jika tidak ada.
- Jika file referensi kunci tidak ada, gunakan `cloudhsm-cli key generate-file --encoding ksp-key-reference` pada instance Windows Server baru Anda untuk membuatnya. Untuk petunjuk, lihat [Menghasilkan referensi kunci KSP \(Windows\)](#).

3. Verifikasi sertifikat root

Periksa sertifikat root Anda di otoritas sertifikasi root tepercaya:

```
PS C:\Users\Administrator\Desktop> certutil -store Root

Root "Trusted Root Certification Authorities"
===== Certificate 0 =====
Serial Number: certificate-serial-number
Issuer: CN=MYRootCA
NotBefore: 2/5/2020 1:38 PM
NotAfter: 2/5/2021 1:48 PM
Issuer: CN=MYRootCA
Signature matches Public Key
Root Certificate: Subject matches Issuer
Cert Hash(sha1): cert-hash
No key provider information
Cannot find the certificate and private key for decryption.
CertUtil: -store command completed successfully.
```

Note

Perhatikan nomor seri sertifikat untuk digunakan pada langkah berikutnya.

4. Ekspor sertifikat root

Ekspor sertifikat root ke file:

```
certutil -store Root certificate-serial-number root-certificate-name.cer
```

5. Verifikasi sertifikat HSM-backend

Periksa sertifikat HSM-backend Anda di toko sertifikat Pribadi:

```
PS C:\Users\Administrator\Desktop> certutil -store My  
  
my "Personal"  
===== Certificate 0 =====  
Serial Number: certificate-serial-number  
Issuer: CN=MYRootCA  
NotBefore: 2/5/2020 1:38 PM  
NotAfter: 2/5/2021 1:48 PM  
Subject: CN=www.mydomain.com, OU=Certificate Management, O=Information Technology,  
L=Houston, S=Texas, C=US  
Non-root Certificate  
Cert Hash(sha1): cert-hash  
Key Container = key-container-name  
Provider = Cavium Key Storage Provider  
Private key is NOT exportable  
Encryption test passed  
CertUtil: -store command completed successfully.
```

Note

Perhatikan nomor seri sertifikat untuk digunakan pada langkah berikutnya.

6. Ekspor sertifikat HSM-backend

Ekspor sertifikat HSM-backend ke file:

```
certutil -store My certificate-serial-number signed-certificate-name.cer
```

7. Impor sertifikat root

Pada instance Windows baru Anda:

1. Salin file CA root ke instance Windows baru Anda
2. Impor sertifikat:

```
certutil -addstore Root root-certificate-name.cer
```

8. Verifikasi instalasi sertifikat root

Konfirmasikan sertifikat root diinstal dengan benar:

```
PS C:\Users\Administrator\Desktop> certutil -store Root

Root "Trusted Root Certification Authorities"
===== Certificate 0 =====
Serial Number: certificate-serial-number
Issuer: CN=MYRootCA
NotBefore: 2/5/2020 1:38 PM
NotAfter: 2/5/2021 1:48 PM
Issuer: CN=MYRootCA
Signature matches Public Key
Root Certificate: Subject matches Issuer
Cert Hash(sha1): cert-hash
No key provider information
Cannot find the certificate and private key for decryption.
CertUtil: -store command completed successfully.
```

9. Impor sertifikat HSM-backend

Pada instance Windows baru Anda:

1. Salin sertifikat HSM-backend ke instance Windows baru Anda
2. Impor sertifikat:

```
certutil -addstore My signed-certificate-name.cer
```

10. Verifikasi instalasi sertifikat HSM-backend

Konfirmasikan sertifikat HSM-backend diinstal dengan benar:

```
PS C:\Users\Administrator\Desktop> certutil -store My

my "Personal"
===== Certificate 0 =====
Serial Number: certificate-serial-number
Issuer: CN=MYRootCA
NotBefore: 2/5/2020 1:38 PM
NotAfter: 2/5/2021 1:48 PM
Subject: CN=www.mydomain.com, OU=Certificate Management, O=Information Technology,
L=Houston, S=Texas, C=US
```

```
Non-root Certificate
Cert Hash(sha1): cert-hash
No key provider information
Cannot find the certificate and private key for decryption.
CertUtil: -store command completed successfully.
```

 Note

Perhatikan nomor seri sertifikat untuk digunakan pada langkah selanjutnya.

11. Buat file referensi kunci (opsional)

Selesaikan langkah ini hanya jika Anda perlu membuat file referensi kunci baru. Jika tidak, lanjutkan ke langkah berikutnya.

 Note

Fitur ini hanya ada di SDK versi 5.16.0 dan yang lebih baru.

1. Instal [OpenSSL](#) dan ekstrak modulus:

```
openssl x509 -in signed-certificate-name.cer -modulus -noout
```

 Note

Perintah OpenSSL menampilkan modulus dalam format: `Modulus=modulus-value`
Perhatikan *modulus-value* untuk digunakan dalam perintah berikutnya.

2. Buat file referensi kunci dengan CloudHSM CLI, lihat: [Menghasilkan referensi kunci KSP \(Windows\)](#)

```
& "C:\Program Files\Amazon\CloudHSM\bin\cloudhsm-cli.exe" key generate-  
file --encoding ksp-key-reference --filter attr.class=private-key  
attr.modulus=0xmodulus-value
```

 Note

Argumen perintah *modulus-value* in CloudHSM CLI harus 0x diawali dengan untuk menunjukkan format heksadesimal.

File referensi utama dibuat di C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP\GlobalPartition.

12. Buat konfigurasi perbaikan

Buat file bernama `repair.txt` dengan konten berikut:

```
[Properties]
11 = "" ; Add friendly name property
2 = "{text}" ; Add Key Provider Information property
_continue_="Container=key-container-name&"
_continue_="Provider=Cavium Key Storage Provider&"
_continue_="Flags=0&"
_continue_="KeySpec=2"
```

 Note

Ganti *key-container-name* dengan nama file referensi kunci dari C :
\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP
\GlobalPartition

13. Toko sertifikat perbaikan

Jalankan perintah perbaikan:

```
certutil -repairstore My certificate-serial-number repair.txt
```

 Note

Nomor seri sertifikat diperoleh dari langkah-langkah sebelumnya saat memverifikasi instalasi sertifikat HSM-backend.

14. Verifikasi asosiasi sertifikat

Konfirmasikan sertifikat terkait dengan benar:

```
PS C:\Users\Administrator\Desktop> certutil -store My  
  
my "Personal"  
===== Certificate 0 =====  
Serial Number: certificate-serial-number  
Issuer: CN=MYRootCA  
NotBefore: 2/5/2020 1:38 PM  
NotAfter: 2/5/2021 1:48 PM  
Subject: CN=www.mydomain.com, OU=Certificate Management, O=Information Technology,  
L=Houston, S=Texas, C=US  
Non-root Certificate  
Cert Hash(sha1): cert-hash  
Key Container = key-container-name  
Provider = Cavium Key Storage Provider  
Private key is NOT exportable  
ERROR: Could not verify certificate public key against private key  
CertUtil: -store command completed successfully.
```

Verifikasi output menunjukkan:

- Nama kontainer kunci yang benar
- Penyedia Penyimpanan Kunci Cavium
- ERROR: Could not verify certificate public key against private keyIni adalah masalah yang diketahui, lihat [Masalah: Verifikasi penyimpanan sertifikat gagal](#)

15. Uji aplikasi Anda

Sebelum menyelesaikan migrasi:

1. Uji aplikasi Anda di lingkungan pengembangan Anda
2. Perbarui kode Anda untuk menyelesaikan setiap perubahan yang melanggar
3. Untuk panduan khusus aplikasi, lihat [Mengintegrasikan aplikasi pihak ketiga dengan AWS CloudHSM](#)

Verifikasi migrasi

Setelah menyelesaikan langkah-langkah migrasi, verifikasi bahwa:

- Sertifikat Anda dipasang dengan benar di toko sertifikat yang benar
- File referensi utama hadir di lokasi yang benar
- Aplikasi Anda dapat melakukan operasi kriptografi menggunakan sertifikat yang dimigrasi

Pemecahan masalah

Jika Anda mengalami masalah selama migrasi, verifikasi:

- Semua sertifikat diekspor dengan benar dari sistem sumber
- Nomor seri sertifikat cocok antar sistem
- Nama kontainer kunci dalam file repair.txt cocok dengan file referensi kunci Anda
- SDK3 mode kompatibilitas diaktifkan jika menggunakan file referensi kunci SDK3 yang dihasilkan

Topik terkait

- [Praktik terbaik untuk AWS CloudHSM](#)

Migrasikan penyedia JCE Anda dari AWS CloudHSM Client SDK 3 ke Client SDK 5

Gunakan topik ini untuk memigrasikan [penyedia JCE](#) Anda dari AWS CloudHSM Client SDK 3 ke Client SDK 5. Untuk manfaat migrasi, lihat [Manfaat SDK AWS CloudHSM Klien 5](#).

Pada tahun AWS CloudHSM, aplikasi pelanggan melakukan operasi kriptografi menggunakan AWS CloudHSM Client Software Development Kit (SDK). Client SDK 5 adalah SDK utama yang terus memiliki fitur baru dan dukungan platform yang ditambahkan ke dalamnya.

Penyedia Client SDK 3 JCE menggunakan kelas khusus dan APIs itu bukan bagian dari spesifikasi JCE standar. Client SDK 5 untuk penyedia JCE adalah keluhan dengan spesifikasi JCE dan tidak kompatibel dengan Client SDK 3 di area tertentu. Aplikasi pelanggan mungkin memerlukan perubahan sebagai bagian dari migrasi ke SDK Klien 5. Bagian ini menguraikan perubahan yang diperlukan untuk migrasi yang berhasil.

Untuk meninjau petunjuk migrasi untuk semua penyedia, lihat [Migrasi dari SDK AWS CloudHSM Klien 3 ke SDK Klien 5](#).

Topik

- [Bersiaplah dengan mengatasi perubahan yang melanggar](#)
- [Migrasi ke SDK Klien 5](#)
- [Topik terkait](#)

Bersiaplah dengan mengatasi perubahan yang melanggar

Tinjau perubahan yang melanggar ini dan perbarui aplikasi Anda di lingkungan pengembangan yang sesuai.

Kelas dan nama Provider telah berubah

Apa yang telah berubah	Apa itu di Client SDK 3	Apa itu di Client SDK 5	Contoh
Kelas dan nama penyedia	Kelas penyedia JCE di Client SDK 3 dipanggil <code>CaviumProvider</code> dan memiliki nama Provider. <code>Cavium</code>	Di Client SDK 5, kelas Provider dipanggil <code>CloudHsmProvider</code> dan memiliki nama <code>CloudHSM Provider</code> .	Contoh cara menginisialisasi <code>CloudHsmProvider</code> objek tersedia di repositori AWS CloudHSM GitHub sampel .

Login eksplisit telah berubah, implisit belum

Apa yang telah berubah	Apa itu di Client SDK 3	Apa itu di Client SDK 5	Contoh
Login eksplisit	Klien SDK 3 menggunakan <code>LoginManager</code> kelas untuk login eksplisit. ¹	Di Client SDK 5, <code>CloudHSM</code> penyedia mengimplementasikan <code>AuthProvider</code> untuk login eksplisit. <code>AuthProvider</code> adalah kelas	Untuk contoh tentang cara menggunakan login eksplisit dengan Client SDK 5, lihat <code>LoginRunner</code> contoh di repositori sampel AWS GitHub CloudHSM .

Apa yang telah berubah	Apa itu di Client SDK 3	Apa itu di Client SDK 5	Contoh
		Java standar dan mengikuti cara idiomatik Java untuk masuk ke Provider. Dengan manajemen status login yang ditingkatkan di Client SDK 5, aplikasi tidak perlu lagi memantau dan melakukan login selama ² rekoneksi.	
Login implisit	Tidak ada perubahan yang diperlukan untuk login implisit. File properti yang sama dan semua variabel lingkungan akan terus berfungsi untuk login implisit saat bermigrasi dari Client SDK 3 ke Client SDK 5.		Untuk contoh tentang cara menggunakan login implisit dengan Client SDK 5, lihat LoginRunner sampel di repositori AWS CloudHSM GitHub sampel .

- [1] Cuplikan kode SDK 3 klien:

```

LoginManager lm = LoginManager.getInstance();

lm.login(partition, user, pass);

```

- [2] Cuplikan kode SDK 5 klien:

```

// Construct or get the existing provider object
AuthProvider provider = new CloudHsmProvider();

// Call login method on the CloudHsmProvider object
// Here loginHandler is a CallbackHandler
provider.login(null, loginHandler);

```

Untuk contoh tentang cara menggunakan login eksplisit dengan Client SDK 5, lihat [LoginRunner sampel di repositori](#) AWS CloudHSM GitHub sampel.

Generasi kunci telah berubah

Apa yang telah berubah	Apa itu di Client SDK 3	Apa itu di Client SDK 5	Contoh
Pembuatan kunci	Di Client SDK 3, <code>Cavium[Key-type]AlgorithmParameterSpec</code> digunakan untuk menentukan parameter pembuatan kunci. Untuk cuplikan kode, lihat catatan kaki. 1	Di Client SDK 5, <code>KeyAttributesMap</code> digunakan untuk menentukan atribut pembuatan kunci. Untuk cuplikan kode, lihat catatan kaki. 2	Untuk contoh tentang cara menggunakan <code>KeyAttributesMap</code> untuk menghasilkan kunci simetris, lihat SymmetricKeys sampel di repositori sampel AWS CloudHSM GitHub .
Generasi pasangan kunci	Di Client SDK 3, <code>Cavium[Key-type]AlgorithmparameterSpec</code> digunakan untuk menentukan parameter pembuatan key pair. Untuk cuplikan kode, lihat catatan kaki. 3	Di Client SDK 5, <code>KeyPairAttributesMap</code> digunakan untuk menentukan parameter ini. Untuk cuplikan kode, lihat catatan kaki. 4	Untuk contoh tentang cara menggunakan <code>KeyAttributesMap</code> untuk menghasilkan kunci asimetris, lihat AsymmetricKeys sampel di repositori i AWS CloudHSM GitHub sampel.

- [1] Cuplikan kode pembuatan kunci SDK 3 klien:

```
KeyGenerator keyGen = KeyGenerator.getInstance("AES", "Cavium");
```

```
CaviumAESKeyGenParameterSpec aesSpec = new CaviumAESKeyGenParameterSpec(
    keySizeInBits,
    keyLabel,
    isExtractable,
    isPersistent);
keyGen.init(aesSpec);
SecretKey aesKey = keyGen.generateKey();
```

- [2] Cuplikan kode pembuatan kunci SDK 5 klien:

```
KeyGenerator keyGen = KeyGenerator.getInstance("AES",
    CloudHsmProvider.PROVIDER_NAME);

final KeyAttributesMap aesSpec = new KeyAttributesMap();
aesSpec.put(KeyAttribute.LABEL, keyLabel);
aesSpec.put(KeyAttribute.SIZE, keySizeInBits);
aesSpec.put(KeyAttribute.EXTRACTABLE, isExtractable);
aesSpec.put(KeyAttribute.TOKEN, isPersistent);

keyGen.init(aesSpec);
SecretKey aesKey = keyGen.generateKey();
```

- [3] Cuplikan kode pembuatan key pair SDK 3 klien::

```
KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("rsa", "Cavium");
CaviumRSAKeyGenParameterSpec spec = new CaviumRSAKeyGenParameterSpec(
    keySizeInBits,
    new BigInteger("65537"),
    label + ":public",
    label + ":private",
    isExtractable,
    isPersistent);

keyPairGen.initialize(spec);

keyPairGen.generateKeyPair();
```

- [4] Cuplikan kode pembuatan key pair SDK 5 klien:

```
KeyPairGenerator keyPairGen =
    KeyPairGenerator.getInstance("RSA", providerName);

// Set attributes for RSA public key
```

```

final KeyAttributesMap publicKeyAttrsMap = new KeyAttributesMap();
publicKeyAttrsMap.putAll(additionalPublicKeyAttributes);
publicKeyAttrsMap.put(KeyAttribute.LABEL, label + ":Public");
publicKeyAttrsMap.put(KeyAttribute.MODULUS_BITS, keySizeInBits);
publicKeyAttrsMap.put(KeyAttribute.PUBLIC_EXPONENT,
new BigInteger("65537").toByteArray());

// Set attributes for RSA private key
final KeyAttributesMap privateKeyAttrsMap = new KeyAttributesMap();
privateKeyAttrsMap.putAll(additionalPrivateKeyAttributes);
privateKeyAttrsMap.put(KeyAttribute.LABEL, label + ":Private");

// Create KeyPairAttributesMap and use that to initialize the
// keyPair generator
KeyPairAttributesMap keyPairSpec =
new KeyPairAttributesMapBuilder()
.withPublic(publicKeyAttrsMap)
.withPrivate(privateKeyAttrsMap)
.build();

keyPairGen.initialize(keyPairSpec);
keyPairGen.generateKeyPair();

```

Menemukan, menghapus, dan mereferensikan kunci telah berubah

Menemukan kunci yang sudah dihasilkan dengan AWS CloudHSM memerlukan penggunaan. KeyStore Klien SDK 3 memiliki dua KeyStore jenis: Cavium dan CloudHSM. Klien SDK 5 hanya memiliki satu KeyStore jenis: CloudHSM.

Pindah dari Cavium KeyStore ke CloudHSM KeyStore membutuhkan perubahan KeyStore tipe. Selain itu, Client SDK 3 menggunakan pegangan kunci untuk referensi kunci, sedangkan Client SDK 5 menggunakan label kunci. Perubahan perilaku yang dihasilkan tercantum di bawah ini.

Apa yang telah berubah	Apa itu di Client SDK 3	Apa itu di Client SDK 5	Contoh
Referensi utama	Dengan Client SDK 3, aplikasi menggunakan an label kunci atau pegangan kunci	Di Client SDK 5, aplikasi dapat menggunakan tombol AWS CloudHSM	

Apa yang telah berubah	Apa itu di Client SDK 3	Apa itu di Client SDK 5	Contoh
	untuk referensi kunci di HSM. Mereka menggunakan label dengan KeyStore untuk menemukan kunci, atau mereka menggunakan pegangan dan membuat CaviumKey objek.	KeyStore Kelas Java untuk Client SDK 5 untuk menemukan kunci berdasarkan label. Untuk menemukan kunci dengan pegangan, gunakan AWS CloudHSM KeyStoreWithAttributes dengan AWS CloudHSM KeyReferenceSpec .	

Apa yang telah berubah	Apa itu di Client SDK 3	Apa itu di Client SDK 5	Contoh
Menemukan beberapa entri	Saat mencari kunci menggunakan <code>angetEntry</code> , <code>getKey</code> , atau <code>getCertificate</code> dalam skenario di mana beberapa item dengan kriteria yang sama ada di Cavium KeyStore, hanya entri pertama yang ditemukan yang akan dikembalikan.	Dengan AWS CloudHSM KeyStore dan <code>KeyStoreWithAttributes</code> , skenario yang sama ini akan menghasilkan pengecualian yang dilemparkan. Untuk memperbaiki masalah ini, disarankan untuk mengatur label unik untuk kunci menggunakan Mengatur atribut kunci dengan CloudHSM CLI perintah di CloudHSM CLI. Atau gunakan <code>KeyStoreWithAttributes#getKeys</code> untuk mengembalikan semua kunci yang sesuai dengan kriteria.	

Apa yang telah berubah	Apa itu di Client SDK 3	Apa itu di Client SDK 5	Contoh
Temukan semua kunci	Dimungkinkan di Client SDK 3 untuk menemukan semua kunci di HSM menggunakan <code>Util.findAllKeys()</code>	Client SDK 5 membuat pencarian kunci lebih sederhana dan lebih efisien dengan menggunakan <code>KeyStoreWithAttributes</code> kelas. Jika memungkinkan, cache kunci Anda untuk meminimalkan latensi. Untuk informasi selengkapnya, lihat Mengelola kunci secara efektif dalam aplikasi Anda . Bila Anda perlu mengambil semua kunci dari HSM, gunakan <code>KeyStoreWithAttributes#getKeys</code> dengan kosong. <code>KeyAttributesMap</code>	Contoh yang menggunakan <code>KeyStoreWithAttributes</code> kelas untuk menemukan kunci tersedia di repositori AWS CloudHSM GitHub sampel dan cuplikan kode ditampilkan di 1

Apa yang telah berubah	Apa itu di Client SDK 3	Apa itu di Client SDK 5	Contoh
Penghapusan kunci	Klien SDK 3 digunakan <code>Util.deleteKey()</code> untuk menghapus kunci.	KeyObjek di Client SDK 5 mengimplementasikan <code>Destroyable</code> antarmuka yang memungkinkan kunci dihapus menggunakan <code>destroy()</code> metode antarmuka ini.	Kode contoh yang menunjukkan fungsionalitas tombol hapus dapat ditemukan di repositori sampel GitHub CloudHSM . Cuplikan sampel untuk setiap SDK ditampilkan di 2

- [1] cuplikan ditunjukkan di bawah ini:

```
KeyAttributesMap findSpec = new KeyAttributesMap();
findSpec.put(KeyAttribute.LABEL, label);
findSpec.put(KeyAttribute.KEY_TYPE, keyType);
KeyStoreWithAttributes keyStore = KeyStoreWithAttributes.getInstance("CloudHSM");

keyStore.load(null, null);
keyStore.getKey(findSpec);
```

- [2] Menghapus kunci di Client SDK 3:

```
Util.deleteKey(key);
```

Menghapus kunci di Client SDK 5:

```
((Destroyable) key).destroy();
```

Operasi membuka cipher telah berubah, operasi cipher lainnya belum

Note

Tidak ada perubahan yang diperlukan untuk operasi Cipher. `encrypt/decrypt/wrap`

Operasi buka bungkus memerlukan `CaviumUnwrapParameterSpec` kelas Client SDK 3 untuk diganti dengan salah satu kelas berikut khusus untuk operasi kriptografi yang terdaftar.

- `GCMUnwrapKeySpec` untuk AES/GCM/NoPadding membuka bungkusnya
- `IvUnwrapKeySpec` untuk AESWrap unwrap dan AES/CBC/NoPadding unwrap
- `OAEPUnwrapKeySpec` untuk RSA OAEP unwrap

Contoh cuplikan untuk: `OAEPUnwrapKeySpec`

```
OAEPParameterSpec oaepParameterSpec =
new OAEPParameterSpec(
    "SHA-256",
    "MGF1",
    MGF1ParameterSpec.SHA256,
    PSpecified.DEFAULT);

KeyAttributesMap keyAttributesMap =
    new KeyAttributesMap(KeyAttributePermissiveProfile.KEY_CREATION);
keyAttributesMap.put(KeyAttribute.TOKEN, true);
keyAttributesMap.put(KeyAttribute.EXTRACTABLE, false);

OAEPUnwrapKeySpec spec = new OAEPUnwrapKeySpec(oaepParameterSpec,
    keyAttributesMap);

Cipher hsmCipher =
    Cipher.getInstance(
        "RSA/ECB/OAEPPadding",
        CloudHsmProvider.PROVIDER_NAME);
hsmCipher.init(Cipher.UNWRAP_MODE, key, spec);
```

Operasi tanda tangan tidak berubah

Tidak ada perubahan yang diperlukan untuk operasi Tanda Tangan.

Migrasi ke SDK Klien 5

Ikuti petunjuk di bagian ini untuk bermigrasi dari Client SDK 3 ke Client SDK 5.

Note

Amazon Linux, Ubuntu 16.04, Ubuntu 18.04 CentOS 6, CentOS 8, dan RHEL 6 saat ini tidak didukung dengan Client SDK 5. Jika saat ini Anda menggunakan salah satu platform ini dengan Client SDK 3, Anda harus memilih platform yang berbeda saat bermigrasi ke Client SDK 5.

1. Copot pemasangan penyedia JCE untuk Client SDK 3.

Amazon Linux 2

```
$ sudo yum remove cloudhsm-client-jce
```

CentOS 7

```
$ sudo yum remove cloudhsm-client-jce
```

RHEL 7

```
$ sudo yum remove cloudhsm-client-jce
```

RHEL 8

```
$ sudo yum remove cloudhsm-client-jce
```

Ubuntu 16.04 LTS

```
$ sudo apt remove cloudhsm-client-jce
```

Ubuntu 18.04 LTS

```
$ sudo apt remove cloudhsm-client-jce
```

2. Hentikan Daemon Klien untuk SDK Klien 3.

Amazon Linux 2

```
$ sudo service cloudhsm-client stop
```

CentOS 7

```
$ sudo service cloudhsm-client stop
```

RHEL 7

```
$ sudo service cloudhsm-client stop
```

RHEL 8

```
$ sudo service cloudhsm-client stop
```

Ubuntu 16.04 LTS

```
$ sudo systemctl stop cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo systemctl stop cloudhsm-client
```

3. Copot pemasangan Daemon Klien untuk SDK Klien 3.

Amazon Linux 2

```
$ sudo yum remove cloudhsm-client
```

CentOS 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 8

```
$ sudo yum remove cloudhsm-client
```

Ubuntu 16.04 LTS

```
$ sudo apt remove cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo apt remove cloudhsm-client
```

Note

Konfigurasi khusus perlu diaktifkan lagi.

4. Instal penyedia Client SDK JCE dengan mengikuti langkah-langkahnya. [Instal penyedia JCE untuk AWS CloudHSM Client SDK 5](#)
5. Client SDK 5 memperkenalkan format file konfigurasi baru dan alat bootstrap baris perintah. Untuk mem-bootstrap penyedia Client SDK 5 JCE Anda, ikuti petunjuk yang tercantum dalam panduan pengguna di bawah. [Bootstrap Klien SDK](#)
6. Di lingkungan pengembangan Anda, uji aplikasi Anda. Lakukan pembaruan pada kode yang ada untuk menyelesaikan perubahan yang melanggar sebelum migrasi terakhir Anda.

Topik terkait

- [Praktik terbaik untuk AWS CloudHSM](#)

Menggunakan Client SDK 5 untuk bekerja dengan AWS CloudHSM

AWS CloudHSM mencakup dua versi SDK Klien utama:

- SDK Klien 5: Ini adalah SDK Klien terbaru dan default kami. Untuk informasi tentang manfaat dan keuntungan yang diberikannya, lihat [Manfaat SDK AWS CloudHSM Klien 5](#).

- SDK Klien 3: Ini adalah SDK Klien kami yang lebih lama. Ini mencakup satu set lengkap komponen untuk platform dan aplikasi berbasis bahasa kompatibilitas dan alat manajemen.

Untuk petunjuk tentang migrasi dari SDK Klien 3 ke SDK Klien 5, lihat. [Migrasi dari SDK AWS CloudHSM Klien 3 ke SDK Klien 5](#)

Topik ini menjelaskan SDK Klien 5. Untuk memeriksa versi SDK Klien yang Anda gunakan, lihat

Topik

- [Manfaat SDK AWS CloudHSM Klien 5](#)
- [AWS CloudHSM Platform yang didukung SDK Klien 5](#)
- [Pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5](#)
- [Mesin AWS CloudHSM Dinamis OpenSSL untuk SDK Klien 5](#)
- [Penyedia AWS CloudHSM OpenSSL untuk SDK Klien 5](#)
- [Penyedia penyimpanan kunci \(KSP\) untuk AWS CloudHSM Klien SDK 5](#)
- [Penyedia JCE untuk AWS CloudHSM Klien SDK 5](#)

Manfaat SDK AWS CloudHSM Klien 5

Dibandingkan dengan AWS CloudHSM Client SDK 3, Client SDK 5 lebih mudah dikelola, menawarkan konfigurasi yang unggul, dan peningkatan keandalan. Client SDK 5 juga memberikan beberapa keuntungan utama tambahan untuk Client SDK 3.

Dirancang untuk arsitektur tanpa server

Klien SDK 5 tidak memerlukan daemon klien, jadi Anda tidak perlu lagi mengelola layanan latar belakang. Ini membantu pengguna dalam beberapa cara penting:

- Menyederhanakan proses startup aplikasi. Yang perlu Anda lakukan untuk memulai dengan CloudHSM adalah mengkonfigurasi SDK sebelum menjalankan aplikasi Anda.
- Anda tidak memerlukan proses yang terus berjalan, yang membuat integrasi dengan komponen tanpa server seperti Lambda dan Elastic Container Service (ECS) lebih mudah.

Integrasi pihak ketiga yang lebih baik dan portabilitas yang lebih mudah

Klien SDK 5 mengikuti spesifikasi JCE dengan cermat dan memberikan portabilitas yang lebih mudah antara penyedia JCE yang berbeda dan integrasi pihak ketiga yang lebih baik

Peningkatan pengalaman pengguna dan konfigurasi

Client SDK 5 meningkatkan keterbacaan pesan log dan memberikan pengecualian yang lebih jelas dan mekanisme penanganan kesalahan, yang semuanya membuat triaging swalayan jauh lebih mudah bagi pengguna. SDK 5 juga menawarkan berbagai konfigurasi, yang tercantum di halaman [Configure Tool](#).

Dukungan platform yang lebih luas

Klien SDK 5 menawarkan lebih banyak dukungan untuk platform operasi modern. [Ini termasuk dukungan untuk teknologi ARM dan dukungan yang lebih besar untuk JCE, PKCS #11, dan OpenSSL](#). Untuk informasi selengkapnya, lihat [Platform yang didukung](#).

IPv6 dukungan koneksi

Client SDK 5.14+ mendukung koneksi ke dual-stack menggunakan HSMs IPv6

Fitur dan mekanisme tambahan

SDK Klien 5 mencakup fitur dan mekanisme tambahan yang tidak tersedia di SDK Klien 3, dan SDK Klien 5 akan terus menambahkan lebih banyak mekanisme di masa mendatang.

AWS CloudHSM Platform yang didukung SDK Klien 5

Dukungan dasar berbeda untuk setiap versi SDK AWS CloudHSM Klien. Dukungan platform untuk komponen dalam SDK biasanya cocok dengan dukungan dasar, tetapi tidak selalu. Untuk menentukan dukungan platform untuk komponen tertentu, pertama-tama pastikan platform yang Anda inginkan muncul di bagian dasar SDK, lalu periksa pengecualian atau informasi terkait lainnya di bagian komponen.

AWS CloudHSM hanya mendukung sistem operasi 64-bit.

Dukungan platform berubah dari waktu ke waktu. Versi CloudHSM Client SDK sebelumnya mungkin tidak mendukung semua sistem operasi yang tercantum di sini. Gunakan catatan rilis untuk menentukan dukungan sistem operasi untuk versi CloudHSM Client SDK sebelumnya. Untuk informasi selengkapnya, lihat [Unduh untuk AWS CloudHSM Client SDK](#).

Untuk platform yang didukung untuk SDK Klien sebelumnya, lihat [AWS CloudHSM Platform yang didukung SDK Klien 3](#)

Klien SDK 5 tidak memerlukan daemon klien.

Topik

- [Dukungan Linux untuk AWS CloudHSM Client SDK 5](#)
- [Dukungan Windows untuk AWS CloudHSM Klien SDK 5](#)
- [Dukungan tanpa server untuk AWS CloudHSM Klien SDK 5](#)
- [Kompatibilitas HSM untuk AWS CloudHSM Klien SDK 5](#)

Dukungan Linux untuk AWS CloudHSM Client SDK 5

AWS CloudHSM Client SDK 5 mendukung sistem operasi dan platform Linux berikut.

Platform yang didukung	X86_64 Arsitektur	Arsitektur ARM
Amazon Linux 2	Ya	Ya
Amazon Linux 2023	Ya	Ya
Perusahaan Topi Merah Linux 8 (8.3+)	Ya	Ya
Perusahaan Topi Merah Linux 9 (9.2+)	Ya	Ya
Perusahaan Topi Merah Linux 10 (10.0+)	Ya	Ya
Ubuntu 22.04 LTS	Ya	Ya
Ubuntu 24.04 LTS	Ya	Ya

- SDK 5.16 adalah rilis terakhir yang memberikan dukungan platform Ubuntu 20.04 LTS. Untuk informasi lebih lanjut, lihat situs [web Ubuntu](#).
- SDK 5.12 adalah rilis terakhir yang memberikan dukungan platform CentOS 7 (7.8+). Untuk informasi lebih lanjut, lihat situs web [CentOS](#).
- SDK 5.12 adalah rilis terakhir yang memberikan dukungan platform Red Hat Enterprise Linux 7 (7.8+). Untuk informasi lebih lanjut, lihat situs [web Red Hat](#).

- SDK 5.4.2 adalah rilis terakhir yang memberikan dukungan platform CentOS 8. Untuk informasi lebih lanjut, lihat situs web [CentOS](#).

Dukungan Windows untuk AWS CloudHSM Klien SDK 5

AWS CloudHSM Client SDK 5 mendukung versi Windows Server berikut.

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019
- Server Microsoft Windows 2022
- Microsoft Windows Server 2025

Dukungan tanpa server untuk AWS CloudHSM Klien SDK 5

AWS CloudHSM Klien SDK 5 mendukung layanan tanpa AWS server berikut.

- AWS Lambda
- Docker/ECS

Kompatibilitas HSM untuk AWS CloudHSM Klien SDK 5

Tabel berikut menjelaskan kompatibilitas AWS CloudHSM Client SDK 5 untuk HSMs.

hsm1.sedang	hsm2m.sedang
Kompatibel dengan Client SDK versi 5.0.0 dan yang lebih baru.	Kompatibel dengan Client SDK versi 5.9.0 dan yang lebih baru.

Pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5

PKCS #11 adalah standar untuk melakukan operasi kriptografi pada modul keamanan perangkat keras (). HSMs AWS CloudHSM menawarkan implementasi pustaka PKCS #11 yang sesuai dengan PKCS #11 versi 2.40.

Untuk informasi tentang bootstrap, lihat. [Menghubungkan ke cluster](#) Untuk pemecahan masalah, lihat. [Masalah yang diketahui untuk pustaka PKCS #11 untuk AWS CloudHSM](#)

Untuk informasi tentang penggunaan SDK Klien 3, lihat [Menggunakan versi SDK sebelumnya untuk bekerja dengan AWS CloudHSM](#).

Topik

- [Instal pustaka PKCS #11 untuk AWS CloudHSM Client SDK 5](#)
- [Mengautentikasi ke pustaka PKCS #11 untuk AWS CloudHSM SDK Klien 5](#)
- [Tipe kunci yang didukung untuk pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5](#)
- [Mekanisme yang didukung untuk pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5](#)
- [Operasi API yang didukung untuk pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5](#)
- [Atribut kunci di pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5](#)
- [Contoh kode untuk pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5](#)
- [Konfigurasi lanjutan untuk pustaka PKCS #11 untuk AWS CloudHSM](#)
- [Penyimpanan sertifikat dengan pustaka PKCS #11](#)

Instal pustaka PKCS #11 untuk AWS CloudHSM Client SDK 5

Topik ini memberikan petunjuk untuk menginstal pustaka PKCS #11 versi terbaru untuk seri versi AWS CloudHSM Client SDK 5. [Untuk informasi selengkapnya tentang pustaka SDK Klien atau PKCS #11, lihat Menggunakan pustaka SDK Klien dan PKCS #11.](#)

Dengan klien SDK 5, Anda tidak diharuskan untuk menginstal atau menjalankan daemon klien.

Untuk menjalankan satu klaster HSM dengan SDK Klien 5, Anda harus terlebih dahulu mengelola pengaturan daya tahan kunci klien dengan menetapkan `disable_key_availability_check` ke `True`. Untuk informasi selengkapnya, lihat [Sinkronisasi Kunci](#) dan [Alat Konfigurasi SDK Klien 5](#).

Untuk informasi selengkapnya tentang pustaka PKCS #11 di Client SDK 5, lihat pustaka [PKCS #11](#).

Note

Untuk menjalankan satu klaster HSM dengan SDK Klien 5, Anda harus terlebih dahulu mengelola pengaturan daya tahan kunci klien dengan menetapkan `disable_key_availability_check` ke `True`. Untuk informasi selengkapnya, lihat [Sinkronisasi Kunci](#) dan [Alat Konfigurasi SDK Klien 5](#).

Untuk menginstal dan mengkonfigurasi pustaka PKCS #11

1. Gunakan perintah berikut untuk mengunduh dan menginstal pustaka PKCS #11.

Amazon Linux 2023

Instal pustaka PKCS #11 untuk Amazon Linux 2023 pada arsitektur X86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-pkcs11-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.amzn2023.x86_64.rpm
```

Instal pustaka PKCS #11 untuk Amazon Linux 2023 pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-pkcs11-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.amzn2023.aarch64.rpm
```

Amazon Linux 2

Instal pustaka PKCS #11 untuk Amazon Linux 2 pada arsitektur X86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el7.x86_64.rpm
```

Instal pustaka PKCS #11 untuk Amazon Linux 2 pada ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-pkcs11-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el7.aarch64.rpm
```

RHEL 10 (10.0+)

Instal pustaka PKCS #11 untuk RHEL 10 pada arsitektur X86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/cloudhsm-pkcs11-latest.el10.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el10.x86_64.rpm
```

Instal pustaka PKCS #11 untuk RHEL 10 pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/cloudhsm-pkcs11-latest.el10.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el10.aarch64.rpm
```

RHEL 9 (9.2+)

Instal pustaka PKCS #11 untuk RHEL 9 pada arsitektur X86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-pkcs11-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el9.x86_64.rpm
```

Instal pustaka PKCS #11 untuk RHEL 9 pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-pkcs11-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el9.aarch64.rpm
```

RHEL 8 (8.3+)

Instal pustaka PKCS #11 untuk RHEL 8 pada arsitektur X86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-pkcs11-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el8.x86_64.rpm
```

Instal pustaka PKCS #11 untuk RHEL 8 pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-pkcs11-latest.el8.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el8.aarch64.rpm
```

Ubuntu 24.04 LTS

Instal pustaka PKCS #11 untuk Ubuntu 24.04 LTS pada arsitektur X86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-pkcs11_latest_u24.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u24.04_amd64.deb
```

Instal pustaka PKCS #11 untuk Ubuntu 24.04 LTS pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-pkcs11_latest_u24.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u24.04_arm64.deb
```

Ubuntu 22.04 LTS

Instal pustaka PKCS #11 untuk Ubuntu 22.04 LTS pada arsitektur X86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-pkcs11_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u22.04_amd64.deb
```

Instal pustaka PKCS #11 untuk Ubuntu 22.04 LTS pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-pkcs11_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u22.04_arm64.deb
```

Windows Server

Instal pustaka PKCS #11 untuk Windows Server pada arsitektur X86_64:

1. Unduh [pustaka PKCS #11 untuk SDK Klien 5](#).
2. Jalankan penginstal pustaka PKCS #11 (AWSCloudHSMPC11-latest.msi) dengan hak administratif Windows.
2. Gunakan alat konfigurasi untuk menentukan lokasi sertifikat penerbitan. Untuk petunjuk, lihat [Tentukan lokasi sertifikat penerbitan](#).
3. Untuk terhubung ke cluster Anda, lihat [Bootstrap Klien SDK](#).
4. Anda dapat menemukan berkas pustaka PKCS #11 di lokasi berikut:
 - Linux biner, skrip konfigurasi, dan berkas log:

```
/opt/cloudhsm
```

Biner Windows:

```
C:\Program Files\Amazon\CloudHSM
```

Skrip konfigurasi Windows dan berkas log:

```
C:\ProgramData\Amazon\CloudHSM
```

Mengautentikasi ke pustaka PKCS #11 untuk AWS CloudHSM SDK Klien 5

Saat Anda menggunakan pustaka PKCS #11, aplikasi Anda berjalan sebagai [pengguna krypto \(CU\)](#) tertentu di dalam Anda HSMs . AWS CloudHSM Aplikasi Anda dapat melihat dan mengelola hanya kunci yang CU miliki dan bagikan. Anda dapat menggunakan CU yang ada di HSMs atau membuat CU baru untuk aplikasi Anda. Untuk informasi tentang pengelolaan CUs, lihat [Mengelola pengguna HSM dengan CloudHSM CLI dan Mengelola pengguna HSM dengan CloudHSM Management Utility \(CMU\)](#)

Untuk menentukan CU ke pustaka PKCS #11, gunakan parameter pin dari PKCS #11 [Fungsi C_Login](#). Untuk AWS CloudHSM, parameter pin memiliki format berikut:

```
<CU_user_name>:<password>
```

Sebagai contoh, perintah berikut menetapkan pin pustaka PKCS #11 ke CU dengan nama pengguna CryptoUser dan kata sandi CUPassword123!.

```
CryptoUser:CUPassword123!
```

Tipe kunci yang didukung untuk pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5

Pustaka PKCS #11 untuk AWS CloudHSM Client SDK 5 mendukung tipe kunci berikut.

Tipe Kunci	Deskripsi
AES	Hasilkan kunci AES 128, 192, dan 256-bit.
Tiga DES (3DES, DESede)	Hasilkan kunci Triple DES 192-bit. Lihat catatan 1 di bawah untuk perubahan yang akan datang.
EC	Hasilkan kunci dengan kurva secp224r1 (P-224), secp256r1 (P-256), secp256k1 (Blockchain), secp384r1 (P-384), dan secp521r1 (P-521).
GENERIC_SECRET	Hasilkan 1 hingga 800 byte rahasia generik.

Tipe Kunci	Deskripsi
RSA	Hasilkan 2048-bit sampai 4096-bit kunci RSA, dengan penambahan 256 bit.

[1] Sesuai dengan panduan NIST, ini tidak diizinkan untuk cluster dalam mode FIPS setelah 2023. Untuk cluster dalam mode non-FIPS, masih diperbolehkan setelah 2023. Lihat [Kepatuhan FIPS 140: Penutupan Mekanisme 2024](#) untuk detail.

Mekanisme yang didukung untuk pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5

Pustaka PKCS #11 sesuai dengan spesifikasi PKCS #11 versi 2.40. Untuk memanggil fitur kriptografi menggunakan PKCS #11, panggil fungsi dengan mekanisme tertentu. Bagian berikut merangkum kombinasi fungsi dan mekanisme yang didukung oleh AWS CloudHSM Client SDK 5.

Pustaka PKCS #11 mendukung algoritme berikut ini:

- Enkripsi dan dekripsi — AES-CBC, AES-CTR, AES-ECB, AES-GCM, -CBC, -ECB, RSA-OAEP, dan RSA-PKCS DES3 DES3
- Tanda tangani dan verifikasi- RSA, HMAC, dan ECDSA; dengan dan tanpa hashing
- Hash/digest — SHA1,,, SHA224, dan SHA256 SHA384 SHA512
- Bungkus kunci— Bungkus Kunci AES,¹AES-GCM, RSA-AES, dan RSA-OAEP
- Derivasi kunci — SP800 -108 Counter KDF dan ECDH dengan KDF (Algoritma KDF yang didukung adalah X9.63 dengan,,,,) SHA1 SHA224 SHA256 SHA384 SHA512

Topik

- [Menghasilkan fungsi key dan key pair](#)
- [Tanda tangani dan verifikasi fungsi](#)
- [Tanda tangani pemulihan dan verifikasi fungsi pemulihan](#)
- [Fungsi mencerna](#)
- [Enkripsi dan dekripsi fungsi](#)
- [Turunkan fungsi kunci](#)
- [Fungsi Bungkus dan Buka Bungkus](#)

- [Ukuran data maksimum untuk setiap mekanisme](#)
- [Anotasi mekanisme](#)

Menghasilkan fungsi key dan key pair

Pustaka AWS CloudHSM perangkat lunak untuk pustaka PKCS #11 memungkinkan Anda menggunakan mekanisme berikut untuk fungsi Generate Key dan Key Pair.

- CKM_RSA_PKCS_KEY_PAIR_GEN
- CKM_RSA_X9_31_KEY_PAIR_GENMekanisme ini secara fungsional identik dengan CKM_RSA_PKCS_KEY_PAIR_GEN mekanisme, tetapi menawarkan jaminan yang lebih kuat untuk p dan q generasi.
- CKM_EC_KEY_PAIR_GEN
- CKM_GENERIC_SECRET_KEY_GEN
- CKM_AES_KEY_GEN
- CKM_DES3_KEY_GEN— perubahan yang akan datang tercantum dalam catatan kaki⁵.

Tanda tangani dan verifikasi fungsi

Pustaka AWS CloudHSM perangkat lunak untuk pustaka PKCS #11 memungkinkan Anda menggunakan mekanisme berikut untuk fungsi Masuk dan Verifikasi. Dengan Client SDK 5, data di-hash secara lokal dalam perangkat lunak. Ini berarti tidak ada batasan ukuran data yang dapat di-hash oleh SDK.

Dengan Client SDK 5 RSA dan ECDSA hashing dilakukan secara lokal sehingga tidak ada batasan data. Dengan HMAC, ada batas data. Lihat catatan kaki ² untuk info lebih lanjut.

RSA

- CKM_RSA_X_509
- CKM_RSA_PKCS— operasi satu bagian saja.
- CKM_RSA_PKCS_PSS— operasi satu bagian saja.
- CKM_SHA1_RSA_PKCS
- CKM_SHA224_RSA_PKCS
- CKM_SHA256_RSA_PKCS
- CKM_SHA384_RSA_PKCS

- CKM_SHA512_RSA_PKCS
- CKM_SHA512_RSA_PKCS
- CKM_SHA1_RSA_PKCS_PSS
- CKM_SHA224_RSA_PKCS_PSS
- CKM_SHA256_RSA_PKCS_PSS
- CKM_SHA384_RSA_PKCS_PSS
- CKM_SHA512_RSA_PKCS_PSS

ECDSA

- CKM_ECDSA— operasi satu bagian saja.
- CKM_ECDSA_SHA1
- CKM_ECDSA_SHA224
- CKM_ECDSA_SHA256
- CKM_ECDSA_SHA384
- CKM_ECDSA_SHA512

HMAC

- CKM_SHA_1_HMAC²
- CKM_SHA224_HMAC²
- CKM_SHA256_HMAC²
- CKM_SHA384_HMAC²
- CKM_SHA512_HMAC²

CMAC

- CKM_AES_CMAC

Tanda tangani pemulihan dan verifikasi fungsi pemulihan

Client SDK 5 tidak mendukung fungsi Sign Recover dan Verify Recover.

Fungsi mencerna

Pustaka AWS CloudHSM perangkat lunak untuk pustaka PKCS #11 memungkinkan Anda menggunakan mekanisme berikut untuk fungsi Digest. Dengan Client SDK 5, data di-hash secara lokal dalam perangkat lunak. Ini berarti tidak ada batasan ukuran data yang dapat di-hash oleh SDK.

- CKM_SHA_1
- CKM_SHA224
- CKM_SHA256
- CKM_SHA384
- CKM_SHA512

Enkripsi dan dekripsi fungsi

Pustaka AWS CloudHSM perangkat lunak untuk pustaka PKCS #11 memungkinkan Anda menggunakan mekanisme berikut untuk fungsi Enkripsi dan Dekripsi.

- CKM_RSA_X_509
- CKM_RSA_PKCS— operasi satu bagian saja. Perubahan yang akan datang tercantum dalam catatan kaki⁵.
- CKM_RSA_PKCS_OAEP— operasi satu bagian saja.
- CKM_AES_ECB
- CKM_AES_CTR
- CKM_AES_CBC
- CKM_AES_CBC_PAD
- CKM_DES3_CBC— perubahan yang akan datang tercantum dalam catatan kaki⁵.
- CKM_DES3_ECB— perubahan yang akan datang tercantum dalam catatan kaki⁵.
- CKM_DES3_CBC_PAD— perubahan yang akan datang tercantum dalam catatan kaki⁵.
- CKM_AES_GCM ^{1, 2}
- CKM_CLOUDHSM_AES_GCM³

Turunkan fungsi kunci

Pustaka AWS CloudHSM perangkat lunak untuk pustaka PKCS #11 mendukung mekanisme derivasi kunci berikut:

- CKM_SP800_108_COUNTER_KDF
- CKM_ECDH1_DERIVE- Mendukung derivasi kunci ECDH dengan jenis KDF yang ditentukan vendor berikut: [6](#)
 - CKD_CLOUDHSM_X963_SHA1_KDF- X9.63 KDF dengan SHA1 [7](#)
 - CKD_CLOUDHSM_X963_SHA224_KDF- X9.63 KDF dengan SHA224 [7](#)
 - CKD_CLOUDHSM_X963_SHA256_KDF- X9.63 KDF dengan SHA256 [7](#)
 - CKD_CLOUDHSM_X963_SHA384_KDF- X9.63 KDF dengan SHA384 [7](#)
 - CKD_CLOUDHSM_X963_SHA512_KDF- X9.63 KDF dengan SHA512 [7](#)

Fungsi Bungkus dan Buka Bungkus

Pustaka AWS CloudHSM perangkat lunak untuk pustaka PKCS #11 memungkinkan Anda menggunakan mekanisme berikut untuk fungsi Wrap dan Unwrap.

Untuk informasi tambahan mengenai pembungkus kunci AES, lihat Pembungkus Kunci [AES](#).

- CKM_RSA_PKCS— operasi satu bagian saja. Perubahan yang akan datang tercantum dalam catatan kaki [5](#).
- CKM_RSA_PKCS_OAEP [4](#)
- CKM_AES_GCM [1](#), [3](#)
- CKM_CLOUDHSM_AES_GCM [3](#)
- CKM_RSA_AES_KEY_WRAP
- CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD [3](#)
- CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD [3](#)
- CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD [3](#)

Ukuran data maksimum untuk setiap mekanisme

Tabel berikut mencantumkan ukuran data maksimum yang ditetapkan untuk setiap mekanisme:

Ukuran set data maksimum

Mekanisme	Ukuran data maksimum dalam byte
CKM_SHA_1_HMAC	16288
CKM_SHA224_HMAC	16256
CKM_SHA256_HMAC	16288
CKM_SHA384_HMAC	16224
CKM_SHA512_HMAC	16224
CKM_AES_CBC	16272
CKM_AES_GCM	16224
CKM_CLOUDHSM_AES_GCM	16224
CKM_DES3_CBC	16280

Anotasi mekanisme

- [1] Saat melakukan enkripsi AES-GCM, HSM tidak menerima data vektor inisialisasi (IV) dari aplikasi. Anda harus menggunakan IV yang dihasilkannya. IV 12-byte yang disediakan oleh HSM ditulis ke dalam referensi memori yang ditunjukkan oleh elemen pIV dari parameter CK_GCM_PARAMS struktur yang Anda suplai. Untuk mencegah kebingungan pengguna, PKCS #11 SDK di versi 1.1.1 dan sebelumnya memastikan bahwa pIV menunjuk ke penyangga yang dinolkan ketika enkripsi AES-GCM diinisialisasi.
- [2] Saat mengoperasikan data dengan menggunakan salah satu mekanisme berikut, jika buffer data melebihi ukuran data maksimum, operasi menghasilkan kesalahan. Untuk mekanisme ini, semua pemrosesan data harus terjadi di dalam HSM. Untuk informasi tentang set ukuran data maksimum untuk setiap mekanisme, lihat [Ukuran data maksimum untuk setiap mekanisme](#).

- [3] Mekanisme yang ditentukan vendor. Untuk menggunakan mekanisme yang ditetapkan vendor CloudHSM, aplikasi PKCS #11 harus menyertakan `/opt/cloudhsm/include/pkcs11/pkcs11t.h` selama kompilasi.

CKM_CLOUDHSM_AES_GCM: Mekanisme kepemilikan ini adalah alternatif pemrograman yang lebih aman untuk CKM_AES_GCM standar. Ini menambahkan IV yang dihasilkan oleh HSM untuk ciphertext, bukan menuliskannya kembali ke dalam struktur CK_GCM_PARAMS yang disediakan selama inisialisasi cipher. Anda dapat menggunakan mekanisme ini dengan fungsi `C_Encrypt`, `C_WrapKey`, `C_Decrypt`, dan `C_UnwrapKey`. Bila menggunakan mekanisme ini, variabel `pIV` di struk CK_GCM_PARAMS harus diatur ke `NULL`. Bila menggunakan mekanisme ini dengan `C_Decrypt` dan `C_UnwrapKey`, IV diharapkan akan ditambahkan ke ciphertext yang sedang dibuka.

CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD: Bungkus Kunci AES dengan Padding PKCS #5.

CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD: Bungkus Kunci AES dengan Padding Nol.

- [4] Berikut ini CK_MECHANISM_TYPE dan CK_RSA_PKCS_MGF_TYPE didukung CK_RSA_PKCS_OAEP_PARAMS untuk CKM_RSA_PKCS_OAEP:
 - CKM_SHA_1 menggunakan CKG_MGF1_SHA1
 - CKM_SHA224 menggunakan CKG_MGF1_SHA224
 - CKM_SHA256 menggunakan CKG_MGF1_SHA256
 - CKM_SHA384 menggunakan CKM_MGF1_SHA384
 - CKM_SHA512 menggunakan CKM_MGF1_SHA512
- [5] Sesuai dengan panduan NIST, ini tidak diizinkan untuk cluster dalam mode FIPS setelah 2023. Untuk cluster dalam mode non-FIPS, masih diperbolehkan setelah 2023. Lihat [Kepatuhan FIPS 140: Penutupan Mekanisme 2024](#) untuk detail.
- [6] Jenis yang ditentukan vendor. Untuk menggunakan tipe yang ditentukan vendor CloudHSM, aplikasi PKCS #11 harus menyertakan selama kompilasi. `cloudhsm_pkcs11_vendor_defs.h` Ini ditemukan di `/opt/cloudhsm/include/pkcs11/cloudhsm_pkcs11_vendor_defs.h` platform berbasis Linux dan `C:\Program Files\Amazon\CloudHSM\include\pkcs11\cloudhsm_pkcs11_vendor_defs.h` untuk platform berbasis Windows
- [7] Fungsi derivasi kunci (KDFs) ditentukan dalam [Publikasi Khusus NIST 800-56A](#) Revisi 3.

Operasi API yang didukung untuk pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5

Pustaka PKCS #11 mendukung operasi API PKCS #11 berikut untuk AWS CloudHSM Client SDK 5.

- C_CloseAllSessions
- C_CloseSession
- C_CreateObject
- C_Decrypt
- C_DecryptFinal
- C_DecryptInit
- C_DecryptUpdate
- C_DeriveKey
- C_DestroyObject
- C_Digest
- C_DigestFinal
- C_DigestInit
- C_DigestUpdate
- C_Encrypt
- C_EncryptFinal
- C_EncryptInit
- C_EncryptUpdate
- C_Finalize
- C_FindObjects
- C_FindObjectsFinal
- C_FindObjectsInit
- C_GenerateKey
- C_GenerateKeyPair
- C_GenerateRandom
- C_GetAttributeValue
- C_GetFunctionList

- C_GetInfo
- C_GetMechanismInfo
- C_GetMechanismList
- C_GetSessionInfo
- C_GetSlotInfo
- C_GetSlotList
- C_GetTokenInfo
- C_Initialize
- C_Login
- C_Logout
- C_OpenSession
- C_Sign
- C_SignFinal
- C_SignInit
- C_SignUpdate
- C_UnWrapKey
- C_Verify
- C_VerifyFinal
- C_VerifyInit
- C_VerifyUpdate
- C_WrapKey

Atribut kunci di pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5

Objek AWS CloudHSM kunci dapat berupa kunci publik, pribadi, atau rahasia. Tindakan diizinkan pada objek kunci ditentukan melalui atribut. Atribut didefinisikan ketika objek kunci dibuat. Saat Anda menggunakan pustaka PKCS #11 untuk AWS CloudHSM, kami menetapkan nilai default seperti yang ditentukan oleh standar PKCS #11.

AWS CloudHSM tidak mendukung semua atribut yang tercantum dalam spesifikasi PKCS #11. Kami patuh dengan spesifikasi untuk semua atribut yang kami dukung. Atribut ini tercantum dalam tabel masing-masing.

Fungsi kriptografi seperti `C_CreateObject`, `C_GenerateKey`, `C_GenerateKeyPair`, `C_UnwrapKey`, dan `C_DeriveKey` yang membuat, memodifikasi, atau menyalin objek mengambil templat atribut sebagai salah satu parameternya. Untuk informasi selengkapnya tentang meneruskan templat atribut selama pembuatan objek, lihat [Menghasilkan kunci melalui pustaka PKCS #11 untuk contoh](#).

Topik berikut memberikan informasi lebih lanjut tentang atribut AWS CloudHSM kunci.

Topik

- [Tabel atribut pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5](#)
- [Memodifikasi atribut pustaka PKCS #11 untuk AWS CloudHSM SDK Klien 5](#)
- [Menafsirkan kode kesalahan pustaka PKCS #11 untuk AWS CloudHSM SDK Klien 5](#)

Tabel atribut pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5

Tabel pustaka PKCS #11 untuk AWS CloudHSM berisi daftar atribut yang berbeda menurut jenis kunci. Ini menunjukkan apakah atribut yang diberikan didukung untuk jenis kunci tertentu saat menggunakan fungsi kriptografi tertentu dengan AWS CloudHSM.

Legenda:

- ✓ menunjukkan bahwa CloudHSM mendukung atribut untuk jenis kunci tertentu.
- ✗ menunjukkan bahwa CloudHSM tidak mendukung atribut untuk jenis kunci tertentu.
- R menunjukkan bahwa nilai atribut diatur ke hanya-baca untuk jenis kunci tertentu.
- S menunjukkan bahwa atribut tidak dapat dibaca oleh `GetAttributeValue` karena sensitif.
- Sel kosong di kolom Nilai Default menunjukkan bahwa tidak ada nilai default tertentu yang ditetapkan untuk atribut.

GenerateKeyPair

Atribut	Tipe Kunci				Nilai Default
		EC pribadi	EC publik	RSA pribadi	RSA publik

Atribut	Tipe Kunci				Nilai Default
CKA_CLASS	✓	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	✓	
CKA_LABEL	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	Benar
CKA_TOKEN	✓	✓	✓	✓	Salah
CKA_PRIVATE	✓ ¹	✓ ¹	✓ ¹	✓ ¹	Benar
CKA_ENCRYPT	✗	✓	✗	✓	Salah
CKA_DECRYPT	✓	✗	✓	✗	Salah
CKA_DERIVE	✓	✓	✓	✓	Salah
CKA_MODIFIABLE	✓	✓	✓	✓	Benar
CKA_DESTROYABLE	✓	✓	✓	✓	Benar
CKA_SIGN	✓	✗	✓	✗	Salah

Atribut	Tipe Kunci				Nilai Default
CKA_SIGN_RECOVER	×	×	×	×	
CKA_VERIFY	×	✓	×	✓	Salah
CKA_VERIFY_RECOVER	×	×	×	×	
CKA_WRAP	×	✓	×	✓	Salah
CKA_WRAP_TEMPLATE	×	✓	×	✓	
CKA_TRUSTED	×	✓	×	✓	Salah
CKA_WRAP_WITH_TRUSTED	✓	×	✓	×	Salah
CKA_UNWRAP	✓	×	✓	×	Salah
CKA_UNWRAP_TEMPLATE	✓	×	✓	×	
CKA_SENSITIVE	✓ ¹	×	✓ ¹	×	Benar
CKA_ALWAYS_SENSITIVE	R	×	R	×	

Atribut	Tipe Kunci				Nilai Default
CKA_EXTRA_CTABLE	✓	✗	✓	✗	Benar
CKA_NEVER_EXTRACTABLE	R	✗	R	✗	
CKA_MODULUS	✗	✗	✗	✗	
CKA_MODULUS_BITS	✗	✗	✗	✓ ²	
CKA_PRIME_1	✗	✗	✗	✗	
CKA_PRIME_2	✗	✗	✗	✗	
CKA_COEFFICIENT	✗	✗	✗	✗	
CKA_EXPONENT_1	✗	✗	✗	✗	
CKA_EXPONENT_2	✗	✗	✗	✗	
CKA_PRIVATE_EXPONENT	✗	✗	✗	✗	
CKA_PUBLIC_EXPONENT	✗	✗	✗	✓ ²	

Atribut	Tipe Kunci				Nilai Default
CKA_EC_PA RAMS	×	✓ ²	×	×	
CKA_EC_PO INT	×	×	×	×	
CKA_VALUE	×	×	×	×	
CKA_VALUE _LEN	×	×	×	×	
CKA_CHECK _VALUE	R	R	R	R	

GenerateKey

Atribut	Tipe Kunci			Nilai Default
	AES	DES3	Rahasia Generik	
CKA_CLASS	✓	✓	✓	
CKA_KEY_T YPE	✓	✓	✓	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	Benar
CKA_TOKEN	✓	✓	✓	Salah

Atribut	Tipe Kunci			Nilai Default
CKA_PRIVATE	✓ ¹	✓ ¹	✓ ¹	Benar
CKA_ENCRYPT	✓	✓	✗	Salah
CKA_DECRYPT	✓	✓	✗	Salah
CKA_DERIVE	✓	✓	✓	Salah
CKA_MODIFIABLE	✓	✓	✓	Benar
CKA_DESTROYABLE	✓	✓	✓	Benar
CKA_SIGN	✓	✓	✓	Benar
CKA_SIGN_RECOVER	✗	✗	✗	
CKA_VERIFY	✓	✓	✓	Benar
CKA_VERIFY_RECOVER	✗	✗	✗	
CKA_WRAP	✓	✓	✗	Salah
CKA_WRAP_TEMPLATE	✓	✓	✗	
CKA_TRUSTED	✓	✓	✗	Salah

Atribut	Tipe Kunci			Nilai Default
CKA_WRAP_WITH_TRUSTED	✓	✓	✓	Salah
CKA_UNWRAP	✓	✓	✗	Salah
CKA_UNWRAP_TEMPLATE	✓	✓	✗	
CKA_SENSITIVE	✓	✓	✓	Benar
CKA_ALWAYS_SENSITIVE	✗	✗	✗	
CKA_EXTRACTABLE	✓	✓	✓	Benar
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_MODULUS	✗	✗	✗	
CKA_MODULUS_BITS	✗	✗	✗	
CKA_PRIME_1	✗	✗	✗	
CKA_PRIME_2	✗	✗	✗	

Atribut	Tipe Kunci			Nilai Default
CKA_COEFFICIENT	×	×	×	
CKA_EXPONENT_1	×	×	×	
CKA_EXPONENT_2	×	×	×	
CKA_PRIVATE_EXPONENT	×	×	×	
CKA_PUBLIC_EXPONENT	×	×	×	
CKA_EC_PARAMS	×	×	×	
CKA_EC_POINT	×	×	×	
CKA_VALUE	×	×	×	
CKA_VALUE_LEN	✓ ²	×	✓ ²	
CKA_CHECK_VALUE	R	R	R	

CreateObject

Atribut	Tipe Kunci							Nilai Default
	EC pribadi	EC publik	RSA pribadi	RSA publik	AES	DES3	Rahasia Generik	
CKA_CLASS	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	
CKA_KEY_TYPE	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	R	R	Salah
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓	Salah
CKA_PRIVATE	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	Benar
CKA_ENCRYPT	✗	✗	✗	✓	✓	✓	✗	Salah
CKA_DECRYPT	✗	✗	✓	✗	✓	✓	✗	Salah
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓	Salah
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	✓	Benar

Atribut	Tipe Kunci							Nilai Default
CKA_DESTR OYABLE	✓	✓	✓	✓	✓	✓	✓	Benar
CKA_SIGN	✓	✗	✓	✗	✓	✓	✓	Salah
CKA_SIGN_ RECOVER	✗	✗	✗	✗	✗	✗	✗	Salah
CKA_VERIF Y	✗	✓	✗	✓	✓	✓	✓	Salah
CKA_VERIF Y_RECOVER	✗	✗	✗	✗	✗	✗	✗	
CKA_WRAP	✗	✗	✗	✓	✓	✓	✗	Salah
CKA_WRAP_ TEMPLATE	✗	✓	✗	✓	✓	✓	✗	
CKA_TRUST ED	✗	✓	✗	✓	✓	✓	✗	Salah
CKA_WRAP_ WITH_TRUS TED	✓	✗	✓	✗	✓	✓	✓	Salah
CKA_UNWRA P	✗	✗	✓	✗	✓	✓	✗	Salah
CKA_UNWRA P_TEMPLAT E	✓	✗	✓	✗	✓	✓	✗	
CKA_SENSI TIVE	✓	✗	✓	✗	✓	✓	✓	Benar

Atribut	Tipe Kunci							Nilai Default
CKA_ALWAYS_SENSITIVE	R	×	R	×	R	R	R	
CKA_EXTRACTABLE	✓	×	✓	×	✓	✓	✓	Benar
CKA_NEVER_EXTRACTABLE	R	×	R	×	R	R	R	
CKA_MODULUS	×	×	✓ ²	✓ ²	×	×	×	
CKA_MODULUS_BITS	×	×	×	×	×	×	×	
CKA_PRIME_1	×	×	✓	×	×	×	×	
CKA_PRIME_2	×	×	✓	×	×	×	×	
CKA_COEFFICIENT	×	×	✓	×	×	×	×	
CKA_EXPONENT_1	×	×	✓	×	×	×	×	
CKA_EXPONENT_2	×	×	✓	×	×	×	×	
CKA_PRIVATE_EXPONENT	×	×	✓ ²	×	×	×	×	

Atribut	Tipe Kunci							Nilai Default
CKA_PUBLIC_EXPONENT	×	×	✓ ²	✓ ²	×	×	×	
CKA_EC_PARAMS	✓ ²	✓ ²	×	×	×	×	×	
CKA_EC_POINT	×	✓ ²	×	×	×	×	×	
CKA_VALUE	✓ ²	×	×	×	✓ ²	✓ ²	✓ ²	
CKA_VALUE_LEN	×	×	×	×	×	×	×	
CKA_CHECK_VALUE	R	R	R	R	R	R	R	

UnwrapKey

Atribut	Tipe Kunci						Nilai Default
	EC pribadi	RSA pribadi	AES	DES3	Rahasia Generik		
CKA_CLASS	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²		
CKA_KEY_TYPE	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²		
CKA_LABEL	✓	✓	✓	✓	✓		

Atribut	Tipe Kunci						Nilai Default
CKA_ID		✓	✓	✓	✓	✓	
CKA_LOCAL		R	R	R	R	R	Salah
CKA_TOKEN		✓	✓	✓	✓	✓	Salah
CKA_PRIVATE		✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	Benar
CKA_ENCRYPT		✗	✗	✓	✓	✗	Salah
CKA_DECRYPT		✗	✓	✓	✓	✗	Salah
CKA_DERIVE		✓	✓	✓	✓	✓	Salah
CKA_MODIFIABLE		✓	✓	✓	✓	✓	Benar
CKA_DESTROYABLE		✓	✓	✓	✓	✓	Benar
CKA_SIGN		✓	✓	✓	✓	✓	Salah
CKA_SIGN_RECOVER		✗	✗	✗	✗	✗	Salah
CKA_VERIFY		✗	✗	✓	✓	✓	Salah
CKA_VERIFY_RECOVER		✗	✗	✗	✗	✗	

Atribut	Tipe Kunci						Nilai Default
CKA_WRAP		✗	✗	✓	✓	✗	Salah
CKA_UNWRAP		✗	✓	✓	✓	✗	Salah
CKA_SENSITIVE		✓	✓	✓	✓	✓	Benar
CKA_EXTRACTABLE		✓	✓	✓	✓	✓	Benar
CKA_NEVER_EXTRACTABLE		R	R	R	R	R	
CKA_ALWAYS_SENSITIVE		R	R	R	R	R	
CKA_MODULUS		✗	✗	✗	✗	✗	
CKA_MODULUS_BITS		✗	✗	✗	✗	✗	
CKA_PRIME_1		✗	✗	✗	✗	✗	
CKA_PRIME_2		✗	✗	✗	✗	✗	
CKA_COEFFICIENT		✗	✗	✗	✗	✗	
CKA_EXPONENT_1		✗	✗	✗	✗	✗	

Atribut	Tipe Kunci						Nilai Default
CKA_EXPONENT_2		×	×	×	×	×	
CKA_PRIVATE_EXPONENT		×	×	×	×	×	
CKA_PUBLIC_EXPONENT		×	×	×	×	×	
CKA_EC_PARAMS		×	×	×	×	×	
CKA_EC_POINT		×	×	×	×	×	
CKA_VALUE		×	×	×	×	×	
CKA_VALUE_LEN		×	×	×	×	×	
CKA_CHECK_VALUE		R	R	R	R	R	

DeriveKey

Atribut	Tipe Kunci				Nilai Default
		AES	DES3	Rahasia Generik	
CKA_CLASS		✓ ₂	✓ ₂	✓ ₂	

Atribut	Tipe Kunci			Nilai Default
CKA_KEY_T YPE	✓ ²	✓ ²	✓ ²	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	Benar
CKA_TOKEN	✓	✓	✓	Salah
CKA_PRIVATE TE	✓ ¹	✓ ¹	✓ ¹	Benar
CKA_ENCRY PT	✓	✓	✗	Salah
CKA_DECRY PT	✓	✓	✗	Salah
CKA_DERIV E	✓	✓	✓	Salah
CKA_MODIF IABLE	✓	✓	✓	Benar
CKA_DESTR OYABLE	✓	✓	✓	Benar
CKA_SIGN	✓	✓	✓	Salah
CKA_SIGN_ RECOVER	✗	✗	✗	
CKA_VERIF Y	✓	✓	✓	Salah

Atribut	Tipe Kunci			Nilai Default
CKA_VERIFY_RECOVER	✖	✖	✖	
CKA_WRAP	✓	✓	✖	Salah
CKA_UNWRAP	✓	✓	✖	False
CKA_SENSITIVE	R	R	R	True
CKA_EXTRACTABLE	✓	✓	✓	Benar
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	

Atribut	Tipe Kunci			Nilai Default
CKA_EXPONENT_1	×	×	×	
CKA_EXPONENT_2	×	×	×	
CKA_PRIVATE_EXPONENT	×	×	×	
CKA_PUBLIC_EXPONENT	×	×	×	
CKA_EC_PARAMS	×	×	×	
CKA_EC_POINT	×	×	×	
CKA_VALUE	×	×	×	
CKA_VALUE_LEN	✓ ²	×	✓ ²	
CKA_CHECK_VALUE	R	R	R	

GetAttributeValue

Atribut	Tipe Kunci						
	EC pribadi	EC publik	RSA pribadi	RSA publik	AES	DES3	Rahasia Generik

Atribut	Tipe Kunci						
CKA_CLASS	✓	✓	✓	✓	✓	✓	✓
CKA_KEY_TYPE	✓	✓	✓	✓	✓	✓	✓
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓
CKA_ID	✓	✓	✓	✓	✓	✓	✓
CKA_LOCAL	✓	✓	✓	✓	✓	✓	✓
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓
CKA_PRIVATE	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹
CKA_ENCRYPT	✗	✗	✗	✓	✓	✓	✗
CKA_DECRYPT	✗	✗	✓	✗	✓	✓	✗
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	✓
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓
CKA_SIGN	✓	✗	✓	✗	✓	✓	✓

Atribut	Tipe Kunci						
CKA_SIGN_RECOVER	×	×	✓	×	×	×	×
CKA_VERIFY	×	✓	×	✓	✓	✓	✓
CKA_VERIFY_RECOVER	×	×	×	✓	×	×	×
CKA_WRAP	×	×	×	✓	✓	✓	×
CKA_WRAP_TEMPLATE	×	✓	×	✓	✓	✓	×
CKA_TRUSTED	×	✓	×	✓	✓	✓	✓
CKA_WRAP_WITH_TRUSTED	✓	×	✓	×	✓	✓	✓
CKA_UNWRAP	×	×	✓	×	✓	✓	×
CKA_UNWRAP_TEMPLATE	✓	×	✓	×	✓	✓	×
CKA_SENSITIVE	✓	×	✓	×	✓	✓	✓
CKA_EXTRACTABLE	✓	×	✓	×	✓	✓	✓

Atribut	Tipe Kunci						
CKA_NEVER_EXTRACTABLE	✓	✗	✓	✗	✓	✓	✓
CKA_ALWAYS_SENSITIVE	R	R	R	R	R	R	R
CKA_MODULUS	✗	✗	✓	✓	✗	✗	✗
CKA_MODULUS_BITS	✗	✗	✗	✓	✗	✗	✗
CKA_PRIME_1	✗	✗	S	✗	✗	✗	✗
CKA_PRIME_2	✗	✗	S	✗	✗	✗	✗
CKA_COEFFICIENT	✗	✗	S	✗	✗	✗	✗
CKA_EXPONENT_1	✗	✗	S	✗	✗	✗	✗
CKA_EXPONENT_2	✗	✗	S	✗	✗	✗	✗
CKA_PRIVATE_EXPONENT	✗	✗	S	✗	✗	✗	✗
CKA_PUBLIC_EXPONENT	✗	✗	✓	✓	✗	✗	✗

Atribut	Tipe Kunci						
CKA_EC_PA RAMS	✓	✓	✗	✗	✗	✗	✗
CKA_EC_PO INT	✗	✓	✗	✗	✗	✗	✗
CKA_VALUE	S	✗	✗	✗	✓	✓	✓
CKA_VALUE _LEN	✗	✗	✗	✗	✓	✗	✓
CKA_CHECK _VALUE	✓	✓	✓	✓	✓	✓	✗

Anotasi atribut

- [1] Atribut ini sebagian didukung oleh firmware dan harus secara eksplisit diatur hanya ke nilai default.
- [2] Atribut wajib.

Memodifikasi atribut pustaka PKCS #11 untuk AWS CloudHSM SDK Klien 5

Beberapa atribut pustaka PKCS #11 untuk AWS CloudHSM objek dapat dimodifikasi setelah objek dibuat, sedangkan beberapa tidak bisa. Untuk memodifikasi atribut, gunakan perintah [key set-attribute](#) dari CloudHSM CLI. Anda juga dapat memperoleh daftar atribut dengan menggunakan perintah [daftar kunci](#) dari CloudHSM CLI.

Daftar berikut menampilkan atribut yang diizinkan untuk modifikasi setelah pembuatan objek:

- CKA_LABEL
- CKA_TOKEN

 Note

Modifikasi hanya diperbolehkan untuk mengubah kunci sesi menjadi kunci token. Gunakan perintah [key set-attribute](#) dari CloudHSM CLI untuk mengubah nilai atribut.

- CKA_ENCRYPT
- CKA_DECRYPT
- CKA_SIGN
- CKA_VERIFY
- CKA_WRAP
- CKA_UNWRAP
- CKA_LABEL
- CKA_SENSITIVE
- CKA_DERIVE

 Note

Atribut ini mendukung derivasi kunci. Kunci harus `False` untuk semua kunci publik dan tidak dapat diatur ke `True`. Untuk kunci rahasia dan kunci privat EC, kunci dapat diatur ke `True` atau `False`.

- CKA_TRUSTED

 Note

Atribut ini dapat diatur ke `True` atau `False` oleh Petugas Kripto (CO) saja.

- CKA_WRAP_WITH_TRUSTED

 Note

Terapkan atribut ini ke kunci data yang dapat diekspor untuk menentukan bahwa Anda hanya dapat membungkus kunci ini dengan kunci yang ditandai sebagai `CKA_TRUSTED`. Setelah Anda mengatur `CKA_WRAP_WITH_TRUSTED` menjadi `true` (benar), atribut menjadi hanya-baca dan Anda tidak dapat mengubah atau menghapus atribut.

Menafsirkan kode kesalahan pustaka PKCS #11 untuk AWS CloudHSM SDK Klien 5

Menentukan dalam template atribut pustaka PKCS #11 yang tidak didukung oleh kunci tertentu menghasilkan kesalahan. Tabel berikut berisi kode galat yang dihasilkan ketika Anda melanggar spesifikasi:

Kode Kesalahan	Deskripsi
CKR_TEMPLATE_INCONSISTENT	Anda menerima galat ini ketika Anda menetapkan atribut dalam templat atribut, tempat atribut patuh dengan spesifikasi PKCS #11, tetapi tidak didukung oleh CloudHSM.
CKR_ATTRIBUTE_TYPE_INVALID	Anda menerima galat ini ketika Anda mengambil nilai untuk atribut, yang patuh dengan spesifikasi PKCS #11, tetapi tidak didukung oleh CloudHSM.
CKR_ATTRIBUTE_INCOMPLETE	Anda menerima galat ini ketika Anda tidak menentukan atribut wajib dalam templat atribut.
CKR_ATTRIBUTE_READ_ONLY	Anda menerima galat ini ketika Anda menetapkan atribut hanya-baca dalam templat atribut.

Contoh kode untuk pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5

Contoh kode GitHub menunjukkan cara menyelesaikan tugas dasar menggunakan pustaka PKCS #11 untuk AWS CloudHSM Client SDK 5.

Prasyarat

Sebelum menjalankan sampel, lakukan langkah-langkah berikut untuk mengatur lingkungan Anda:

- Instal dan konfigurasi [pustaka PKCS #11](#) untuk Client SDK 5.
- Siapkan [pengguna kriptografi \(CU\)](#). Aplikasi Anda menggunakan akun HSM ini untuk menjalankan sampel kode pada HSM.

Sampel Kode

Sampel Kode untuk Perpustakaan Perangkat AWS CloudHSM Lunak untuk PKCS #11 tersedia di [GitHub](#). Repositori ini mencakup contoh tentang bagaimana melakukan operasi umum menggunakan PKCS #11 termasuk enkripsi, dekripsi, penandatanganan, dan verifikasi.

- [Hasilkan kunci \(AES, RSA, EC\)](#)
- [Daftar atribut kunci](#)
- [Enkripsi dan dekripsi data dengan AES GCM](#)
- [Enkripsi dan dekripsi data dengan AES_CTR](#)
- [Enkripsi dan dekripsi data dengan 3DES](#)
- [Tanda tangani dan verifikasi data dengan RSA](#)
- [Turunkan kunci menggunakan HMAC KDF](#)
- [Bungkus dan buka kunci dengan AES menggunakan padding PKCS #5](#)
- [Bungkus dan buka kunci dengan AES tanpa padding](#)
- [Bungkus dan buka kunci dengan AES menggunakan bantalan nol](#)
- [Bungkus dan buka kunci dengan AES-GCM](#)
- [Bungkus dan buka kunci dengan RSA](#)

Konfigurasi lanjutan untuk pustaka PKCS #11 untuk AWS CloudHSM

Penyedia AWS CloudHSM PKCS #11 menyertakan konfigurasi lanjutan berikut, yang bukan merupakan bagian dari konfigurasi umum yang digunakan sebagian besar pelanggan. Konfigurasi ini memberikan kemampuan tambahan.

- [Menghubungkan ke beberapa slot dengan PKCS #11](#)
- [Coba lagi konfigurasi untuk PKCS #11](#)

Beberapa konfigurasi slot dengan pustaka PKCS #11 untuk AWS CloudHSM

Satu slot di pustaka Client SDK 5 PKCS #11 mewakili satu koneksi ke cluster di AWS CloudHSM. Dengan Client SDK 5, Anda dapat mengonfigurasi PKCS11 pustaka Anda untuk memungkinkan beberapa slot menghubungkan pengguna ke beberapa klaster CloudHSM dari satu aplikasi PKCS #11.

Gunakan petunjuk dalam topik ini untuk membuat aplikasi Anda menggunakan fungsionalitas multi-slot untuk terhubung dengan beberapa cluster.

Topik

- [Prasyarat multi-slot untuk pustaka PKCS #11 untuk AWS CloudHSM](#)
- [Konfigurasi pustaka PKCS #11 untuk fungsionalitas multi-slot AWS CloudHSM](#)
- [Tambahkan cluster dengan fungsionalitas multi-slot untuk AWS CloudHSM](#)
- [Hapus cluster dengan fungsionalitas multi-slot untuk AWS CloudHSM](#)

Prasyarat multi-slot untuk pustaka PKCS #11 untuk AWS CloudHSM

Sebelum mengonfigurasi beberapa slot untuk pustaka PKCS #11 AWS CloudHSM, lengkapi prasyarat berikut.

- Dua atau lebih AWS CloudHSM cluster yang ingin Anda sambungkan, bersama dengan sertifikat klaster mereka.
- Instans EC2 dengan Grup Keamanan dikonfigurasi dengan benar untuk terhubung ke semua cluster di atas. Untuk informasi selengkapnya tentang cara menyiapkan cluster dan instance klien, lihat [Memulai AWS CloudHSM](#).
- Untuk mengatur fungsionalitas multi-slot, Anda harus sudah mengunduh dan menginstal pustaka PKCS #11. Jika Anda belum melakukan ini, lihat instruksi di [???](#).

Konfigurasi pustaka PKCS #11 untuk fungsionalitas multi-slot AWS CloudHSM

Untuk mengonfigurasi pustaka PKCS #11 Anda untuk fungsionalitas multi-slot AWS CloudHSM, ikuti langkah-langkah berikut:

1. Identifikasi cluster yang ingin Anda sambungkan menggunakan fungsionalitas multi-slot.
2. Tambahkan klaster ini ke konfigurasi PKCS #11 Anda dengan mengikuti petunjuk di [???](#)
3. Lain kali aplikasi PKCS #11 Anda berjalan, itu akan memiliki fungsi multi-slot.

Tambahkan cluster dengan fungsionalitas multi-slot untuk AWS CloudHSM

Saat [menghubungkan ke beberapa slot dengan PKCS #11](#) for AWS CloudHSM, gunakan `configure-pkcs11 add-cluster` perintah untuk menambahkan cluster ke konfigurasi Anda.

Sintaksis

```
configure-pkcs11 add-cluster [OPTIONS]
  --cluster-id <CLUSTER ID>
  [--region <REGION>]
  [--endpoint <ENDPOINT>]
  [--hsm-ca-cert <HSM CA CERTIFICATE FILE>]
  [--client-cert-hsm-tls-file <CLIENT CERTIFICATE FILE>]
  [--client-key-hsm-tls-file <CLIENT KEY FILE>]
  [-h, --help]
```

Contoh

Tambahkan cluster menggunakan **cluster-id** parameter

Example

Gunakan `configure-pkcs11 add-cluster` bersama dengan `cluster-id` parameter untuk menambahkan cluster (dengan ID `cluster-1234567`) ke konfigurasi Anda.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 add-cluster --cluster-id <cluster-1234567>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" add-cluster --cluster-id <cluster-1234567>
```

Tip

Jika menggunakan `configure-pkcs11 add-cluster` dengan `cluster-id` parameter tidak mengakibatkan klaster ditambahkan, lihat contoh berikut untuk versi yang lebih panjang dari perintah ini yang juga memerlukan `--region` dan `--endpoint` parameter untuk mengidentifikasi cluster yang ditambahkan. Jika, misalnya, wilayah cluster berbeda dari yang dikonfigurasi sebagai default AWS CLI Anda, Anda harus menggunakan `--region` parameter untuk menggunakan wilayah yang benar. Selain itu, Anda memiliki kemampuan untuk menentukan titik akhir AWS CloudHSM API yang akan digunakan untuk panggilan, yang mungkin diperlukan untuk berbagai pengaturan jaringan, seperti menggunakan

titik akhir antarmuka VPC yang tidak menggunakan nama host DNS default untuk. AWS CloudHSM

Tambahkan cluster menggunakan **cluster-id**, **endpoint**, dan **region** parameter

Example

Gunakan **region** parameter `configure-pkcs11 add-cluster` bersama dengan **cluster-id**, **endpoint**, dan untuk menambahkan cluster (dengan ID `cluster-1234567`) ke konfigurasi Anda.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 add-cluster --cluster-id <cluster-1234567>
--region <us-east-1> --endpoint <https://cloudhsmv2.us-east-1.amazonaws.com>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" add-cluster --
cluster-id <cluster-1234567> --region <us-east-1> --endpoint <https://cloudhsmv2.us-
east-1.amazonaws.com>
```

Untuk informasi tentang parameter `--cluster-id`, `--region` dan `--endpoint`, lihat [the section called "Parameter"](#).

Parameter

`--cluster-id` **<Cluster ID>**

Membuat panggilan `DescribeClusters` untuk menemukan semua alamat IP antarmuka jaringan elastis (ENI) HSM dalam kluster yang terkait dengan ID kluster. Sistem menambahkan alamat IP ENI ke file AWS CloudHSM konfigurasi.

 Note

Jika Anda menggunakan `--cluster-id` parameter dari instans EC2 dalam VPC yang tidak memiliki akses ke internet publik, maka Anda harus membuat antarmuka VPC endpoint untuk terhubung. AWS CloudHSM Untuk informasi lebih lanjut tentang VPC endpoint, lihat [???](#).

Wajib: Ya

`--titik akhir` **<Endpoint>**

Tentukan titik akhir AWS CloudHSM API yang digunakan untuk melakukan `DescribeClusters` panggilan. Anda harus menetapkan pilihan ini dalam kombinasi dengan `--cluster-id`.

Wajib: Tidak

`--hsm-ca-cert` **<HsmCA Certificate Filepath>**

Menentukan filepath ke sertifikat HSM CA.

Wajib: Tidak

`--wilayah` **<Region>**

Tentukan wilayah klaster Anda. Anda harus menetapkan pilihan ini dalam kombinasi dengan `--cluster-id`.

Jika Anda tidak menyediakan parameter `--region`, sistem memilih wilayah dengan mencoba untuk membaca variabel lingkungan `AWS_DEFAULT_REGION` atau `AWS_REGION`. Jika variabel-variabel tersebut tidak diatur, maka sistem memeriksa wilayah yang terkait dengan profil Anda di file AWS config Anda (biasanya `~/.aws/config`) kecuali jika Anda menentukan file yang berbeda di variabel lingkungan `AWS_CONFIG_FILE`. Jika tidak ada variabel di atas yang diatur, sistem default ke wilayah `us-east-1`.

Wajib: Tidak

`-- client-cert-hsm-tls berkas` **<client certificate hsm tls path>**

Jalur ke sertifikat klien yang digunakan untuk otentikasi timbal balik klien TLS-HSM.

Hanya gunakan opsi ini jika Anda telah mendaftarkan setidaknya satu jangkar kepercayaan ke HSM dengan CloudHSM CLI. Anda harus menetapkan pilihan ini dalam kombinasi dengan `-- client-key-hsm-tls-file`.

Wajib: Tidak

-- client-key-hsm-tls berkas *<client key hsm tls path>*

Jalur ke kunci klien yang digunakan untuk otentikasi timbal balik klien TLS-HSM.

Hanya gunakan opsi ini jika Anda telah mendaftarkan setidaknya satu jangkar kepercayaan ke HSM dengan CloudHSM CLI. Anda harus menetapkan pilihan ini dalam kombinasi dengan -- client-cert-hsm-tls-file.

Wajib: Tidak

Hapus cluster dengan fungsionalitas multi-slot untuk AWS CloudHSM

Saat [menghubungkan ke beberapa slot dengan PKCS #11](#), gunakan configure-pkcs11 remove-cluster perintah untuk menghapus cluster dari slot PKCS #11 yang tersedia.

Sintaksis

```
configure-pkcs11 remove-cluster [OPTIONS]
    --cluster-id <CLUSTER ID>
    [-h, --help]
```

Contoh

Hapus cluster menggunakan **cluster-id** parameter

Example

Gunakan configure-pkcs11 remove-cluster bersama dengan cluster-id parameter untuk menghapus cluster (dengan ID dari cluster-1234567) dari konfigurasi Anda.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 remove-cluster --cluster-id <cluster-1234567>
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" remove-cluster  
--cluster-id <cluster-1234567>
```

Untuk informasi tentang parameter `--cluster-id`, lihat [the section called “Parameter”](#).

Parameter

`--kluster-id` **<Cluster ID>**

ID cluster untuk menghapus dari konfigurasi

Wajib: Ya

Coba lagi perintah untuk pustaka PKCS #11 untuk AWS CloudHSM

AWS CloudHSM Client SDK 5.8.0 dan yang lebih baru memiliki strategi coba ulang otomatis bawaan yang akan mencoba kembali operasi HSM-throttled dari sisi klien. Ketika HSM membatasi operasi karena terlalu sibuk melakukan operasi sebelumnya dan tidak dapat menerima lebih banyak permintaan, klien SDKs akan mencoba kembali operasi yang dibatasi hingga 3 kali sementara mundur secara eksponensial. Strategi coba ulang otomatis ini dapat diatur ke salah satu dari dua mode: off dan standar.

- off: SDK Klien tidak akan melakukan strategi coba ulang apa pun untuk operasi yang dibatasi oleh HSM.
- standard: Ini adalah mode default untuk Client SDK 5.8.0 dan yang lebih baru. Dalam mode ini, klien SDKs akan secara otomatis mencoba kembali operasi yang dibatasi dengan mundur secara eksponensial.

Untuk informasi selengkapnya, lihat [Pelambatan HSM](#).

Setel perintah coba lagi ke mode mati

Linux

Untuk mengatur perintah coba lagi off untuk Client SDK 5 di Linux

- Anda dapat menggunakan perintah berikut untuk mengatur konfigurasi coba lagi ke off mode:


```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --default-retry-mode off
```

Windows

Untuk mengatur perintah coba lagi off untuk Client SDK 5 pada Windows

- Anda dapat menggunakan perintah berikut untuk mengatur konfigurasi coba lagi ke off mode:

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --default-retry-mode off
```

Penyimpanan sertifikat dengan pustaka PKCS #11

Pustaka AWS CloudHSM PKCS #11 mendukung penyimpanan sertifikat kunci publik sebagai “objek publik” (sebagaimana didefinisikan dalam PKCS #11 2.40) pada kluster hsm2m.medium. Fitur ini memungkinkan sesi PKCS #11 publik dan pribadi untuk membuat, mengambil, memodifikasi, dan menghapus sertifikat kunci publik.

Untuk menggunakan penyimpanan sertifikat dengan pustaka PKCS #11, Anda harus mengaktifkannya dalam konfigurasi klien Anda. Setelah diaktifkan, Anda dapat mengelola objek sertifikat dari aplikasi PKCS #11 Anda. Operasi yang berlaku untuk sertifikat dan objek kunci, seperti [C_FindObjects](#), akan mengembalikan hasil dari penyimpanan kunci dan sertifikat.

Topik

- [Mengaktifkan penyimpanan sertifikat](#)
- [Operasi API penyimpanan sertifikat](#)
- [Atribut penyimpanan sertifikat](#)
- [Log audit penyimpanan sertifikat](#)

Mengaktifkan penyimpanan sertifikat

Anda dapat mengaktifkan penyimpanan sertifikat pada kluster hsm2m.medium menggunakan alat konfigurasi pustaka PKCS #11. Fitur ini tersedia dalam SDK versi 5.13 dan yang lebih baru. Untuk daftar operasi yang mendukung jenis objek sertifikat, lihat [Operasi API penyimpanan sertifikat](#).

Untuk mengaktifkan penyimpanan sertifikat, ikuti langkah-langkah berikut untuk sistem operasi Anda:

Linux

- Aktifkan penyimpanan sertifikat

Jalankan perintah berikut:

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --enable-certificate-storage
```

Windows

- Aktifkan penyimpanan sertifikat

Buka command prompt dan jalankan perintah berikut:

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --enable-certificate-storage
```

Operasi API penyimpanan sertifikat

Operasi PKCS #11 berikut mendukung tipe objek sertifikat (CKO_CERTIFICATE):

Operasi sertifikat umum

C_CreateObject

Membuat objek sertifikat baru.

C_DestroyObject

Menghapus objek sertifikat yang ada.

C_GetAttributeValue

Mendapat nilai satu atau lebih atribut dari objek sertifikat.

C_SetAttributeValue

Memperbarui nilai satu atau lebih atribut dari objek sertifikat.

Operasi pencarian objek sertifikat

C_FindObjectsInit

Memulai pencarian objek sertifikat.

C_FindObjects

Melanjutkan pencarian objek sertifikat.

C_FindObjectsFinal

Mengakhiri pencarian objek sertifikat.

Atribut penyimpanan sertifikat

Tabel berikut mencantumkan atribut objek sertifikat yang didukung dan nilainya:

Atribut	Nilai default	Deskripsi
CKA_CLASS	Diperlukan	Harus berupa CKO_CERTIFICATE .
CKA_TOKEN	True	Harus berupa True.
CKA_MODIFIABLE	True	Harus berupa True.
CKA_PRIVATE	False	Harus berupa False.
CKA_LABEL	Kosong	Batasi 127 karakter.
CKA_COPYABLE	False	Harus berupa False.
CKA_DESTROYABLE	True	Harus berupa True.
CKA_CERTIFICATE_TYPE	Diperlukan	Harus berupa CKC_X_509 .
CKA_TRUSTED	False	Harus berupa False.
CKA_CERTIFICATE_CATEGORY	CK_CERTIFICATE_CATEGORY_UNSPECIFIED	Harus berupa CK_CERTIFICATE_CATEGORY_UNSPECIFIED .

Atribut	Nilai default	Deskripsi
	EGORY_UNSPECIFIED	
CKA_CHECK_VALUE	Berasal dari CKA_VALUE	Secara otomatis diatur berdasarkan CKA_VALUE .
CKA_START_DATE	Kosong	Tanggal sertifikat 'tidak sebelum'.
CKA_END_DATE	Kosong	Tanggal sertifikat 'tidak setelah'.
CKA_PUBLIC_KEY_INFO	Kosong	Ukuran maksimum adalah 16 kilobyte.
CKA_SUBJECT	Diperlukan	Subjek sertifikat.
CKA_ID	Kosong	Ukuran maksimum adalah 128 byte. Keunikan tidak ditegakkan.
CKA_ISSUER	Kosong	Penerbit sertifikat.
CKA_SERIAL_NUMBER	Kosong	Nomor seri sertifikat.
CKA_VALUE	Diperlukan	Ukuran maksimum adalah 32 kilobyte.

Log audit penyimpanan sertifikat

AWS CloudHSM menulis log audit untuk operasi penyimpanan sertifikat yang mengubah data ke aliran log Acara CloudWatch Amazon terpisah dalam grup log kluster CloudWatch Anda. Aliran log ini dinamai untuk cluster, bukan untuk HSM tertentu di dalam cluster.

Untuk informasi tentang mengakses log audit CloudWatch, lihat [Bekerja dengan CloudWatch Log Amazon dan Log AWS CloudHSM Audit](#).

Bidang entri log

object_handle

Pengidentifikasi unik dari objek sertifikat.

op_code

Operasi dilakukan atau dicoba. Kemungkinan nilai:

- CreateObject
- DestroyObject
- SetAttributeValues

response

OK jika operasi berhasil, atau salah satu dari jenis kesalahan berikut:

- DuplicateAttribute
- InvalidAttributeValue
- ObjectNotFound
- MaxObjectsReached
- InternalFailure

attributes

Atribut dimodifikasi, jika ada.

timestamp

Waktu ketika operasi terjadi, dalam milidetik sejak zaman Unix.

Contoh-contoh log Audit

CreateObject contoh

```
{
  "object_handle": 463180677312929947,
  "op_code": "CreateObject",
  "response": "OK",
  "attributes": null,
  "timestamp": 1725482483671
}
```

DestroyObject contoh

```
{
```

```
{
  "object_handle": 463180677312929947,
  "op_code": "DestroyObject",
  "response": "OK",
  "attributes": null,
  "timestamp": 1725482484559
}
```

SetAttributeValues contoh

```
{
  "object_handle": 463180678453346687,
  "op_code": "SetAttributeValues",
  "response": "OK",
  "attributes": [
    "Label"
  ],
  "timestamp": 1725482488004
}
```

Contoh yang tidak berhasil CreateObject

```
{
  "object_handle": null,
  "op_code": "CreateObject",
  "response": "MaxObjectsReached",
  "attributes": null,
  "timestamp": 1726084937125
}
```

Mesin AWS CloudHSM Dinamis OpenSSL untuk SDK Klien 5

AWS CloudHSM OpenSSL Dynamic Engine memungkinkan Anda untuk membongkar operasi kriptografi ke cluster CloudHSM Anda melalui OpenSSL API.

AWS CloudHSM menyediakan OpenSSL Dynamic Engine, yang dapat Anda baca di atau. [AWS CloudHSM SSL/TLS offload di Linux menggunakan Tomcat dengan JSSE](#) [AWS CloudHSM SSL/TLS offload di Linux menggunakan NGINX atau Apache dengan OpenSSL](#) Untuk contoh penggunaan AWS CloudHSM dengan OpenSSL, lihat blog keamanan [AWS ini](#). Untuk informasi tentang dukungan platform SDKs, lihat [the section called "Platform yang didukung"](#). Untuk pemecahan masalah, lihat. [Masalah yang diketahui untuk OpenSSL Dynamic Engine untuk AWS CloudHSM](#)

Gunakan bagian berikut untuk menginstal dan mengkonfigurasi mesin AWS CloudHSM dinamis untuk OpenSSL, menggunakan Client SDK 5.

Untuk informasi tentang penggunaan SDK Klien 3, lihat [Menggunakan versi SDK sebelumnya untuk bekerja dengan AWS CloudHSM](#).

Topik

- [Instal OpenSSL Dynamic Engine AWS CloudHSM untuk Client SDK 5](#)
- [Tipe kunci yang didukung untuk OpenSSL Dynamic Engine AWS CloudHSM untuk Client SDK 5](#)
- [Mekanisme yang didukung untuk OpenSSL Dynamic Engine AWS CloudHSM untuk Client SDK 5](#)
- [Konfigurasi lanjutan untuk OpenSSL untuk AWS CloudHSM](#)

Instal OpenSSL Dynamic Engine AWS CloudHSM untuk Client SDK 5

Gunakan bagian berikut untuk menginstal OpenSSL Dynamic Engine AWS CloudHSM for Client SDK 5.

Note

Untuk menjalankan satu klaster HSM dengan SDK Klien 5, Anda harus terlebih dahulu mengelola pengaturan daya tahan kunci klien dengan menetapkan `disable_key_availability_check` ke `True`. Untuk informasi selengkapnya, lihat [Sinkronisasi Kunci](#) dan [Alat Konfigurasi SDK Klien 5](#).

Untuk menginstal dan mengatur konfigurasi OpenSSL Dynamic Engine

1. Gunakan salah satu perintah berikut untuk mengunduh dan menjalankan mesin OpenSSL.

Amazon Linux 2023

Instal OpenSSL Dynamic Engine untuk Amazon Linux 2023 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-dyn-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.amzn2023.x86_64.rpm
```

Instal OpenSSL Dynamic Engine untuk Amazon Linux 2023 pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-dyn-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.amzn2023.aarch64.rpm
```

Amazon Linux 2

Instal OpenSSL Dynamic Engine untuk Amazon Linux 2 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-dyn-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el7.x86_64.rpm
```

Instal OpenSSL Dynamic Engine untuk Amazon Linux 2 pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-dyn-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el7.aarch64.rpm
```

RHEL 9 (9.2+)

Instal OpenSSL Dynamic Engine untuk RHEL 9 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-dyn-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el9.x86_64.rpm
```

Instal OpenSSL Dynamic Engine untuk RHEL 9 pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-dyn-latest.el9.aarch64.rpm
```



```
$ sudo yum install ./cloudhsm-dyn-latest.el9.aarch64.rpm
```

RHEL 8 (8.3+)

Instal OpenSSL Dynamic Engine untuk RHEL 8 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-dyn-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el8.x86_64.rpm
```

Instal OpenSSL Dynamic Engine untuk RHEL 8 pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-dyn-latest.el8.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el8.aarch64.rpm
```

Ubuntu 24.04 LTS

Instal OpenSSL Dynamic Engine untuk Ubuntu 24.04 LTS pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-dyn_latest_u24.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u24.04_amd64.deb
```

Instal OpenSSL Dynamic Engine untuk Ubuntu 24.04 LTS pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-dyn_latest_u24.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u24.04_arm64.deb
```

Ubuntu 22.04 LTS

Instal OpenSSL Dynamic Engine untuk Ubuntu 22.04 LTS pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-dyn_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u22.04_amd64.deb
```

Instal OpenSSL Dynamic Engine untuk Ubuntu 22.04 LTS pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-dyn_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u22.04_arm64.deb
```

Anda telah menginstal perpustakaan bersama untuk mesin dinamis di `/opt/cloudhsm/lib/libcloudhsm_openssl_engine.so`.

2. SDK Klien Bootstrap 5. Untuk informasi selengkapnya tentang bootstrapping, lihat [Bootstrap Klien SDK](#).
3. Tetapkan variabel lingkungan dengan kredensial pengguna krypto (CU). Untuk informasi tentang membuat CUs, lihat [the section called “buat”](#).

```
$ export CLOUDHSM_PIN=<HSM user name>:<password>
```

Note

SDK Klien 5 memperkenalkan CLOUDHSM_PIN variabel lingkungan untuk menyimpan kredensial CU. Di Client SDK 3 Anda menyimpan kredensi CU dalam variabel lingkungan. `n3fips_password` Klien SDK 5 mendukung kedua variabel lingkungan, namun sebaiknya gunakan `CLOUDHSM_PIN`.

Saat mengatur variabel `CLOUDHSM_PIN` lingkungan, Anda harus melarikan diri dari karakter khusus apa pun yang dapat ditafsirkan oleh shell Anda.

4. Hubungkan instalasi OpenSSL Dynamic Engine Anda ke cluster. Untuk informasi selengkapnya, lihat [Hubungkan ke Klaster](#).
5. Bootstrap SDK Klien 5. Untuk informasi selengkapnya, lihat [the section called “Bootstrap Klien SDK”](#).

Verifikasi OpenSSL Dynamic Engine untuk SDK Klien 5

Gunakan perintah berikut untuk memverifikasi instalasi OpenSSL Dynamic Engine.

```
$ openssl engine -t cloudhsm
```

Output berikut memverifikasi konfigurasi Anda:

```
(cloudhsm) CloudHSM OpenSSL Engine  
[ available ]
```

Tipe kunci yang didukung untuk OpenSSL Dynamic Engine AWS CloudHSM untuk Client SDK 5

AWS CloudHSM OpenSSL Dynamic Engine mendukung jenis kunci berikut dengan Client SDK 5.

Tipe Kunci	Deskripsi
EC	ECDSA sign/verify untuk tipe kunci P-256, P-384, dan secp256k1. Untuk menghasilkan kunci EC yang dapat dioperasikan dengan mesin OpenSSL, lihat. Ekspor kunci asimetris dengan CloudHSM CLI
RSA	Pembuatan kunci RSA untuk kunci 2048, 3072, dan 4096-bit. Tanda/verifikasi RSA. Verifikasi diturunkan ke perangkat lunak OpenSSL.

Mekanisme yang didukung untuk OpenSSL Dynamic Engine AWS CloudHSM untuk Client SDK 5

AWS CloudHSM OpenSSL Dynamic Engine mendukung mekanisme berikut untuk fungsi Masuk dan Verifikasi dengan Client SDK 5.

Tanda tangani dan verifikasi fungsi

Dengan Client SDK 5, data di-hash secara lokal dalam perangkat lunak. Ini berarti tidak ada batasan ukuran data yang dapat di-hash.

Jenis Tanda Tangan RSA

- SHA1withRSA
- SHA224withRSA
- SHA256withRSA
- SHA384withRSA
- SHA512withRSA

Jenis Tanda Tangan ECDSA

- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

Konfigurasi lanjutan untuk OpenSSL untuk AWS CloudHSM

Penyedia AWS CloudHSM OpenSSL menyertakan konfigurasi lanjutan berikut, yang bukan merupakan bagian dari konfigurasi umum yang digunakan sebagian besar pelanggan. Konfigurasi ini memberikan kemampuan tambahan.

- [Coba lagi perintah untuk OpenSSL](#)

Coba lagi perintah untuk OpenSSL untuk AWS CloudHSM

AWS CloudHSM Client SDK 5.8.0 dan yang lebih baru memiliki strategi coba ulang otomatis bawaan yang akan mencoba kembali operasi HSM-throttled dari sisi klien. Ketika HSM membatasi operasi karena terlalu sibuk melakukan operasi sebelumnya dan tidak dapat menerima lebih banyak permintaan, klien SDKs akan mencoba kembali operasi yang dibatasi hingga 3 kali sementara mundur secara eksponensial. Strategi coba ulang otomatis ini dapat diatur ke salah satu dari dua mode: off dan standar.

- off: SDK Klien tidak akan melakukan strategi coba ulang apa pun untuk operasi yang dibatasi oleh HSM.
- standard: Ini adalah mode default untuk Client SDK 5.8.0 dan yang lebih baru. Dalam mode ini, klien SDKs akan secara otomatis mencoba kembali operasi yang dibatasi dengan mundur secara eksponensial.

Untuk informasi selengkapnya, lihat [Pelambatan HSM](#).

Setel perintah coba lagi ke mode mati

Anda dapat menggunakan perintah berikut untuk mengatur perintah coba lagi ke off mode:

```
$ sudo /opt/cloudhsm/bin/configure-dyn --default-retry-mode off
```

Penyedia AWS CloudHSM OpenSSL untuk SDK Klien 5

Penyedia AWS CloudHSM OpenSSL memungkinkan Anda untuk membongkar operasi kriptografi TLS ke kluster CloudHSM Anda melalui OpenSSL Provider API. Antarmuka Provider adalah pendekatan yang direkomendasikan untuk penerapan baru menggunakan OpenSSL 3.2 dan yang lebih baru.

Gunakan bagian berikut untuk menginstal dan mengkonfigurasi Penyedia AWS CloudHSM OpenSSL, menggunakan Client SDK 5.

Warning

Integrasi dengan OpenSSL CLI saat ini tidak didukung oleh Penyedia OpenSSL. AWS CloudHSM Lihat [the section called “Offload di Linux dengan Penyedia OpenSSL”](#) untuk integrasi yang didukung.

Platform yang didukung

Penyedia OpenSSL memerlukan OpenSSL 3.2 atau yang lebih baru, tersedia di +, Ubuntu 24.04+, dan Amazon Linux EL9 2023+.

Verifikasi kompatibilitas: `openssl version`

Topik

- [Instal Penyedia AWS CloudHSM OpenSSL untuk Klien SDK 5](#)
- [Tipe kunci yang didukung untuk Penyedia AWS CloudHSM OpenSSL untuk SDK Klien 5](#)
- [Mekanisme yang Didukung Penyedia OpenSSL](#)
- [Konfigurasi Lanjutan Penyedia OpenSSL](#)

Instal Penyedia AWS CloudHSM OpenSSL untuk Klien SDK 5

Gunakan bagian berikut untuk menginstal Penyedia OpenSSL untuk AWS CloudHSM Client SDK 5.

Note

Untuk menjalankan satu klaster HSM dengan SDK Klien 5, Anda harus terlebih dahulu mengelola pengaturan daya tahan kunci klien dengan menetapkan `disable_key_availability_check` ke `True`. Untuk informasi selengkapnya, lihat [Sinkronisasi Kunci](#) dan [Alat Konfigurasi SDK Klien 5](#).

Persyaratan

Penyedia OpenSSL memerlukan tipe klaster `hsm2m.medium` dan minimum CloudHSM Client SDK versi 5.17.0 atau yang lebih baru.

Instal Penyedia OpenSSL

Untuk menginstal Penyedia OpenSSL

1. Gunakan perintah berikut untuk mengunduh dan menginstal Penyedia OpenSSL.

Amazon Linux 2023

Instal Penyedia OpenSSL untuk Amazon Linux 2023 pada arsitektur `x86_64`:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-openssl-provider-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-openssl-provider-latest.amzn2023.x86_64.rpm
```

Instal Penyedia OpenSSL untuk Amazon Linux 2023 pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-openssl-provider-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-openssl-provider-latest.amzn2023.aarch64.rpm
```

RHEL 9 (9.2+)

Instal Penyedia OpenSSL untuk RHEL 9 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-openssl-provider-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-openssl-provider-latest.el9.x86_64.rpm
```

Instal Penyedia OpenSSL untuk RHEL 9 pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-openssl-provider-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-openssl-provider-latest.el9.aarch64.rpm
```

RHEL 10 (10.0+)

Instal Penyedia OpenSSL untuk RHEL 10 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/cloudhsm-openssl-provider-latest.el10.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-openssl-provider-latest.el10.x86_64.rpm
```

Instal Penyedia OpenSSL untuk RHEL 10 pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/cloudhsm-openssl-provider-latest.el10.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-openssl-provider-latest.el10.aarch64.rpm
```

Ubuntu 24.04

Instal Penyedia OpenSSL untuk Ubuntu 24.04 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-openssl-provider_latest_u24.04_amd64.deb
```

```
$ sudo dpkg -i ./cloudhsm-openssl-provider_latest_u24.04_amd64.deb
```

Instal Penyedia OpenSSL untuk Ubuntu 24.04 pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-openssl-provider_latest_u24.04_arm64.deb
```

```
$ sudo dpkg -i ./cloudhsm-openssl-provider_latest_u24.04_arm64.deb
```

Anda telah menginstal pustaka bersama untuk Penyedia OpenSSL di. /opt/cloudhsm/lib/licloudhsm_openssl_provider.so

2. SDK Klien Bootstrap 5. Untuk informasi selengkapnya tentang bootstrapping, lihat [Bootstrap Klien SDK](#).
3. Tetapkan variabel CLOUDHSM_PIN lingkungan dengan kredensi pengguna kriptografi (CU) Anda:

```
$ export CLOUDHSM_PIN=<username>:<password>
```

4. Connect instalasi OpenSSL Provider Anda ke cluster. Untuk informasi selengkapnya, lihat [Hubungkan ke Kluster](#).

Verifikasi instalasi.

Verifikasi bahwa Penyedia OpenSSL diinstal dengan benar:

```
$ CLOUDHSM_PIN=<username>:<password> openssl list -providers -provider cloudhsm
```

Anda akan melihat output yang mirip dengan:

```
Providers:
  cloudhsm
    name: AWS CloudHSM OpenSSL Provider
    version: 5.17.0
    status: active
  default
    name: OpenSSL Default Provider
    version: 3.2.2
    status: active
```

Tipe kunci yang didukung untuk Penyedia AWS CloudHSM OpenSSL untuk SDK Klien 5

Penyedia AWS CloudHSM OpenSSL mendukung tipe kunci berikut dengan Client SDK 5.

Tipe Kunci	Deskripsi
RSA	RSA sign/verify dan operasi enkripsi asimetris . Verifikasi diturunkan ke perangkat lunak OpenSSL. Untuk menghasilkan kunci RSA yang dapat dioperasikan dengan Penyedia OpenSSL, lihat. Ekspor kunci asimetris dengan CloudHSM CLI
EC	ECDSA sign/verify untuk kurva P-256, P-384, dan P-521. Verifikasi diturunkan ke perangkat lunak OpenSSL. Untuk menghasilkan kunci EC yang dapat dioperasikan dengan Penyedia OpenSSL, lihat. Ekspor kunci asimetris dengan CloudHSM CLI

Mekanisme yang Didukung Penyedia OpenSSL

AWS CloudHSM OpenSSL Provider SDK mendukung serangkaian mekanisme kriptografi yang komprehensif untuk berbagai operasi termasuk tanda tangan digital, enkripsi asimetris, enkripsi simetris, pertukaran kunci, dan banyak lagi.

Jenis tanda tangan RSA

Penyedia OpenSSL mendukung tanda tangan digital RSA dengan beberapa algoritma hash dan skema padding:

SHA1withRSA

Tanda tangan RSA dengan algoritma hash SHA-1

- Bantalan PKCS #1 v1.5
- Padding PSS (Skema Tanda Tangan Probabilistik)

SHA224withRSA

Tanda tangan RSA dengan algoritma hash SHA-224

- Bantalan PKCS #1 v1.5
- Padding PSS

SHA256withRSA

Tanda tangan RSA dengan algoritma hash SHA-256

- Bantalan PKCS #1 v1.5
- Padding PSS

SHA384withRSA

Tanda tangan RSA dengan algoritma hash SHA-384

- Bantalan PKCS #1 v1.5
- Padding PSS

SHA512withRSA

Tanda tangan RSA dengan algoritma hash SHA-512

- Bantalan PKCS #1 v1.5

- Padding PSS

Jenis tanda tangan ECDSA

Penyedia OpenSSL mendukung tanda tangan digital ECDSA dengan beberapa algoritma hash:

SHA1withECDSA

Tanda tangan ECDSA dengan algoritma hash SHA-1

SHA224withECDSA

Tanda tangan ECDSA dengan algoritma hash SHA-224

SHA256withECDSA

Tanda tangan ECDSA dengan algoritma hash SHA-256

SHA384withECDSA

Tanda tangan ECDSA dengan algoritma hash SHA-384

SHA512withECDSA

Tanda tangan ECDSA dengan algoritma hash SHA-512

Konfigurasi Lanjutan Penyedia OpenSSL

AWS CloudHSM OpenSSL Provider SDK menyediakan opsi konfigurasi lanjutan untuk menyesuaikan perilakunya untuk berbagai kasus penggunaan dan lingkungan. Gunakan alat konfigurasi untuk mengatur opsi ini.

- [Coba lagi konfigurasi untuk Penyedia OpenSSL](#)

Coba lagi perintah untuk Penyedia OpenSSL untuk AWS CloudHSM

AWS CloudHSM Client SDK 5.8.0 dan yang lebih baru memiliki strategi coba ulang otomatis bawaan yang akan mencoba kembali operasi HSM-throttled dari sisi klien. Ketika HSM membatasi operasi karena terlalu sibuk melakukan operasi sebelumnya dan tidak dapat menerima lebih banyak permintaan, klien SDKs akan mencoba kembali operasi yang dibatasi hingga 3 kali sementara mundur secara eksponensial. Strategi coba ulang otomatis ini dapat diatur ke salah satu dari dua mode: off dan standar.

- **off:** SDK Klien tidak akan melakukan strategi coba ulang apa pun untuk operasi yang dibatasi oleh HSM.
- **standard:** Ini adalah mode default untuk Client SDK 5.8.0 dan yang lebih baru. Dalam mode ini, klien SDKs akan secara otomatis mencoba kembali operasi yang dibatasi dengan mundur secara eksponensial.

Untuk informasi selengkapnya, lihat [Pelambatan HSM](#).

Setel perintah coba lagi ke mode mati

Linux

Untuk mengatur perintah coba lagi off untuk Client SDK 5 di Linux

- Anda dapat menggunakan perintah berikut untuk mengatur konfigurasi coba lagi ke off mode:

```
$ sudo /opt/cloudhsm/bin/configure-openssl-provider --default-retry-mode off
```

Penyedia penyimpanan kunci (KSP) untuk AWS CloudHSM Klien SDK 5

Key Storage Provider (KSP) adalah API kriptografi khusus untuk sistem operasi Microsoft Windows. Key Storage Provider (KSP) memungkinkan pengembang untuk menggunakan teknik kriptografi untuk mengamankan aplikasi berbasis Windows.

Untuk informasi tentang bootstrap, lihat [Menghubungkan ke cluster](#)

Untuk informasi tentang penggunaan SDK Klien 3, lihat [Menggunakan versi SDK sebelumnya untuk bekerja dengan AWS CloudHSM](#).

Topik

- [Instal penyedia penyimpanan kunci \(KSP\) untuk AWS CloudHSM Client SDK 5](#)
- [Otentikasi ke penyedia penyimpanan Kunci \(KSP\) untuk AWS CloudHSM Klien SDK 5](#)
- [Tipe kunci yang didukung untuk Penyedia Penyimpanan Kunci \(KSP\) untuk SDK AWS CloudHSM Klien 5](#)
- [Operasi API yang didukung Penyedia penyimpanan kunci \(KSP\) untuk AWS CloudHSM Klien SDK 5](#)
- [Konfigurasi lanjutan untuk KSP untuk AWS CloudHSM](#)

Instal penyedia penyimpanan kunci (KSP) untuk AWS CloudHSM Client SDK 5

Gunakan bagian berikut untuk menginstal penyedia penyimpanan kunci (KSP) untuk AWS CloudHSM Client SDK 5.

Note

Untuk menjalankan satu klaster HSM dengan SDK Klien 5, Anda harus terlebih dahulu mengelola pengaturan daya tahan kunci klien dengan menetapkan `disable_key_availability_check` ke `True`. Untuk informasi selengkapnya, lihat [Sinkronisasi Kunci](#) dan [Alat Konfigurasi SDK Klien 5](#).

Untuk menginstal dan mengkonfigurasi Penyedia Penyimpanan Kunci (KSP)

1. Instal Penyedia Penyimpanan Kunci (KSP) untuk Windows Server pada arsitektur x86_64, buka PowerShell sebagai administrator dan jalankan perintah berikut:

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/AWSCloudHSMKSP-latest.msi -Outfile C:\AWSCloudHSMKSP-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMKSP-latest.msi /quiet /norestart /log C:\client-install.txt' -Wait
```

2. Gunakan alat konfigurasi untuk menentukan lokasi sertifikat penerbitan. Untuk petunjuk, lihat [Tentukan lokasi sertifikat penerbitan](#).
3. Untuk terhubung ke cluster Anda, lihat [Bootstrap Klien SDK](#).
4. Anda dapat menemukan file Penyedia Penyimpanan Kunci (KSP) di lokasi berikut:

- Biner Windows:

```
C:\Program Files\Amazon\CloudHSM
```

Skip konfigurasi Windows dan berkas log:

```
C:\ProgramData\Amazon\CloudHSM
```

Otentikasi ke penyedia penyimpanan Kunci (KSP) untuk AWS CloudHSM Klien SDK 5

Sebelum Anda menggunakan Key Storage Provider (KSP) untuk AWS CloudHSM Client SDK 5, Anda harus mengatur kredensi login untuk HSM pada sistem Anda. Anda memiliki dua pilihan:

- Windows Credentials Manager (direkomendasikan untuk keamanan yang lebih baik)
- Variabel lingkungan sistem (pengaturan yang lebih sederhana)

Pengelola Kredensial Windows

Anda dapat mengatur kredensial menggunakan `set_cloudhsm_credentials` utilitas atau antarmuka Windows Credentials Manager.

- Menggunakan utilitas **`set_cloudhsm_credentials`**:

Pemasang Windows termasuk `set_cloudhsm_credentials` utilitas. Anda dapat menggunakan utilitas ini untuk dengan mudah melewati kredensial login HSM ke Windows Credential Manager. Jika Anda ingin mengkompilasi utilitas ini dari sumber, Anda dapat menggunakan kode Python yang disertakan dalam penginstal.

1. Navigasi ke `C:\Program Files\Amazon\CloudHSM\tools\`.
2. Jalankan perintah berikut:

```
set_cloudhsm_credentials.exe --username <CU USER> --password <CU PASSWORD>
```

- Menggunakan antarmuka Credential Manager:

1. Buka Manajer Kredensi:
 - Masukkan `credential manager` di kotak pencarian taskbar
 - Pilih Credential Manager
2. Pilih Kredensial Windows untuk mengelola kredensial Windows.
3. Pilih Tambahkan kredensi generik
4. Masukkan detail berikut:
 - Alamat Internet atau Jaringan: `CLOUDHSM_PIN`.
 - Nama Pengguna: `<CU USER>`.
 - Kata Sandi: `<CU PASSWORD>`.

5. Pilih OK

Variabel lingkungan sistem

Anda dapat mengatur variabel lingkungan sistem untuk mengidentifikasi HSM dan [pengguna kriptografi](#) (CU) Anda.

Warning

Menyetel kredensial melalui variabel lingkungan sistem menyimpan kata sandi Anda dalam teks biasa di sistem Anda. Untuk keamanan yang lebih baik, gunakan Windows Credential Manager sebagai gantinya.

Anda dapat mengatur variabel lingkungan menggunakan:

- Itu [setx](#).
- Panel Kontrol Properti Sistem Windows (tab Lanjutan).
- mengatur variabel lingkungan sistem permanen Metode [terprogram](#).

Untuk mengatur variabel lingkungan sistem:

CLOUDHSM_PIN=<CU USERNAME>:<CU PASSWORD>

Mengidentifikasi [pengguna kriptografi](#) (CU) di HSM dan menyediakan semua informasi login yang diperlukan. Aplikasi Anda mengautentikasi dan berjalan sebagai CU ini. Aplikasi memiliki izin CU ini dan dapat melihat dan mengelola hanya kunci yang CU miliki dan bagikan. Untuk membuat CU baru, gunakan perintah [user create](#) di CloudHSM CLI. Untuk menemukan yang ada CUs, gunakan perintah [daftar pengguna](#) di CloudHSM CLI.

Contoh:

```
setx /m CLOUDHSM_PIN test_user:password123
```

Note

Saat menyetel variabel lingkungan CLOUDHSM_PIN, Anda harus menghindari karakter khusus apa pun yang mungkin ditafsirkan oleh shell Anda.

Tipe kunci yang didukung untuk Penyedia Penyimpanan Kunci (KSP) untuk SDK AWS CloudHSM Klien 5

Penyedia Penyimpanan AWS CloudHSM Kunci (KSP) mendukung jenis kunci berikut dengan Client SDK 5.

Tipe Kunci	Deskripsi
EC	Hasilkan kunci dengan kurva secp256r1 (P-256), secp384r1 (P-384), dan secp521r1 (P-521).
RSA	Hasilkan kunci RSA 2048, 3072, dan 4096-bit.

Operasi API yang didukung Penyedia penyimpanan kunci (KSP) untuk AWS CloudHSM Klien SDK 5

Parameter dalam KSP ditentukan oleh Microsoft KSP. Lihat [dokumentasi Microsoft](#) untuk informasi selengkapnya.

Penyedia Penyimpanan Kunci (KSP) mendukung operasi API KSP berikut untuk AWS CloudHSM Client SDK 5.

- [NCryptOpenStorageProvider](#)
- [NCryptOpenKey](#)
- [NCryptCreatePersistedKey](#)
- [NCryptGetProperty](#)
- [NCryptSetProperty](#)
- [NCryptFinalizeKey](#)
- [NCryptDeleteKey](#)
- [NCryptFreeObject](#)
- [NCryptFreeBuffer](#)
- [NCryptIsAlgSupported](#)
- [NCryptEnumAlgorithms](#)

- [NCryptEnumKeys](#)
- [NCryptExportKey](#)
- [NCryptSignHash](#)
- [NCryptVerifySignature](#)

NCryptOpenStorageProvider berfungsi dengan Penyedia Penyimpanan Kunci (KSP)

NCryptOpenStorageProviderFungsi memuat dan menginisialisasi Penyedia Penyimpanan Kunci (KSP).

Parameter

phProvider[keluar]

Pointer ke NCrypt_PROV_HANDLE variabel yang menyimpan pegangan penyedia.

pszProviderName[dalam]

Pointer ke string Unicode yang dihentikan nol yang mengidentifikasi penyedia penyimpanan kunci. AWS CloudHSM Key Storage Provider (KSP) mendukung nilai-nilai berikut:

Nilai	Arti
L "Penyedia Penyimpanan Kunci Cloudhsm"	Mengidentifikasi nama penyedia SDK 5 Klien. Sebaiknya gunakan nama ini secara default.
L "Penyedia Penyimpanan Kunci Cavium"	Mengidentifikasi nama penyedia SDK 3 Klien. Didukung untuk kompatibilitas mundur.

 Note

Nilai adalah string karakter lebar literal, seperti yang ditunjukkan oleh L sebelum literal.

dwFlags[dalam]

Bendera yang mengubah perilaku fungsi. Tidak ada bendera yang ditentukan untuk fungsi ini.

Nilai yang Ditampilkan

Fungsi mengembalikan kode status untuk menunjukkan keberhasilan atau kegagalan.

Kode pengembalian umum meliputi:

Kode pengembalian	Deskripsi
ERROR_SUCCESS	Operasi selesai dengan sukses.
NTE_INVALID_PARAMETER	Satu atau lebih parameter tidak valid.
NTE_FAIL	Operasi tidak bisa selesai.

NCryptOpenKey dengan penyedia penyimpanan kunci (KSP)

NCryptOpenKeyFungsi ini membuka kunci yang ada di Penyedia Penyimpanan Kunci (KSP).

Parameter

hProvider[dalam]

Pegangan KSP yang berisi kunci. Gunakan [NCryptOpenStorageProvider](#) untuk mendapatkan pegangan.

phKey[Keluar]

Sebuah pointer ke NCryptKeyHandle variabel yang menyimpan pegangan kunci.

pszKeyName[dalam]

Sebuah pointer ke string Unicode null-terminated yang berisi nama kunci.

dwLegacyKeySpec[di, tidak terpakai]

AWS CloudHSM Penyedia Penyimpanan Kunci (KSP) tidak menggunakan parameter ini.

dwFlags[dalam]

Bendera yang memodifikasi perilaku fungsi. Tidak ada bendera yang ditentukan untuk fungsi ini.

Nilai yang Ditampilkan

Fungsi mengembalikan kode status untuk menunjukkan keberhasilan atau kegagalan.

Kode pengembalian umum meliputi:

Kode kembalikan	Deskripsi
ERROR_SUCCESS	Operasi selesai dengan sukses.
NTE_INVALID_PARAMETER	Satu atau lebih parameter tidak valid.
NTE_FAIL	Operasi tidak bisa selesai.
NTE_INVALID_HANDLE	Pegangan masuk hProvider tidak valid.
NTE_BAD_KEYSET	Nama kunci yang diberikan tidak mengembalikan hasil yang unik.

NCryptCreatePersistedKey dengan penyedia penyimpanan kunci (KSP)

NCryptCreatePersistedKeyFungsi ini membuat kunci baru dan menyimpannya di Penyedia Penyimpanan Kunci (KSP). Anda dapat menggunakan [NCryptSetProperty](#) fungsi untuk mengatur propertinya setelah pembuatan. Anda harus menelepon [NCryptFinalizeKey](#) sebelum Anda dapat menggunakan kunci.

Parameter

hProvider[dalam]

Pegangan penyedia penyimpanan kunci tempat Anda akan membuat kunci. Gunakan [NCryptOpenStorageProvider](#) untuk mendapatkan pegangan ini.

phKey[Keluar]

Alamat NCrypt_KEY_HANDLE variabel yang menyimpan pegangan kunci.

pszAlgId[dalam]

Pointer ke string Unicode yang dihentikan nol yang menentukan pengidentifikasi algoritma kriptografi untuk membuat kunci.

AWS CloudHSM Penyedia Penyimpanan Kunci (KSP) mendukung algoritma berikut:

Konstant/nilai	Deskripsi
BCRYPT_RSA_ALGORITMA "RSA"	Algoritma kunci publik RSA.
BCRYPT_ECDSA_P256_ALGORITMA "ECDSA_P256"	Algoritma tanda tangan digital kurva elips prima 256-bit (FIPS 186-2).
BCRYPT_ECDSA_P384_ALGORITMA "ECDSA_P384"	Algoritma tanda tangan digital kurva elips utama 384-bit (FIPS 186-2).
BCRYPT_ECDSA_P521_ALGORITMA "ECDSA_P521"	Algoritma tanda tangan digital kurva elips utama 521-bit (FIPS 186-2).

pszKeyName[dalam, opsional]

Pointer ke string Unicode yang dihentikan nol yang berisi nama kunci. Jika parameter ini adalah NULL, fungsi ini akan membuat kunci fana yang tidak bertahan.

dwLegacyKeySpec[di, tidak terpakai]

AWS CloudHSM Penyedia Penyimpanan Kunci (KSP) tidak menggunakan parameter ini.

dwFlags[dalam]

Bendera untuk memodifikasi perilaku fungsi. Gunakan nol atau lebih dari nilai-nilai berikut:

Nilai	Arti
NCRYPT_MACHINE_KEY_FLAG	Bendera ini tidak berpengaruh.
NCRYPT_SILENT_FLAG	Bendera ini tidak berpengaruh.
NCRYPT_OVERWRITE_KEY_FLAG	Menentukan flag ini menimpa kunci yang ada dengan nama yang sama di HSM.

Nilai	Arti
	Tanpa bendera ini, fungsi kembali NT E_EXISTS.

Nilai yang Ditampilkan

Fungsi mengembalikan kode status untuk menunjukkan keberhasilan atau kegagalan.

Kode pengembalian umum meliputi:

Kode pengembalian	Deskripsi
ERROR_SUCCESS	Fungsi selesai dengan sukses.
NTE_INVALID_PARAMETER	Satu atau lebih parameter tidak valid.
NTE_FAIL	Operasi tidak bisa selesai.
NTE_BAD_FLAGS	dwFlagsParameter berisi nilai yang tidak valid.
NTE_NOT_SUPPORTED	pszAlgIdParameter berisi nilai yang tidak didukung.
NTE_EXISTS	Kunci dengan nama yang ditentukan sudah ada dan operasi tidak digunakan NCRYPT_OV ERWRITE_KEY_FLAG .

NCryptGetProperty dengan penyedia penyimpanan kunci (KSP)

NCryptGetPropertyFungsi mengambil nilai properti untuk objek penyimpanan kunci.

Parameter

hObject[dalam]

Pegangan objek yang propertinya ingin Anda ambil. Anda dapat menggunakan:

- Pegangan penyedia (NCRYPT_PROV_HANDLE)
- Sebuah pegangan kunci (NCRYPT_KEY_HANDLE)

pszProperty [dalam]

Sebuah pointer ke string Unicode null-terminated yang berisi nama properti untuk diambil.

Saat menggunakan `NCRYPT_PROV_HANDLE`, Penyedia Penyimpanan AWS CloudHSM Kunci (KSP) mendukung pengidentifikasi KSP berikut:

Pengenal/Nilai	Deskripsi
<code>NCRYPT_IMPL_TYPE_PROPERTY</code> L "Tipe Impl"	DWORD yang berisi flag yang menentukan detail implementasi penyedia
<code>NCRYPT_MAX_NAME_LENGTH_PROPERTY</code> L "Panjang Nama Maks"	DWORD yang berisi panjang maksimum (dalam karakter) untuk nama kunci persisten.
<code>NCRYPT_NAME_PROPERTY</code> L "Nama"	Sebuah pointer ke string Unicode null-terminated yang berisi nama KSP.
<code>NCRYPT_VERSION_PROPERTY</code> L "Versi"	DWORD yang berisi versi penyedia (kata tinggi: versi utama, kata rendah: versi minor).
<code>NCRYPT_USE_CONTEXT_PROPERTY</code> L "Gunakan Konteks"	Pointer ke string Unicode yang dihentikan nol yang menjelaskan konteks operasi.
<code>NCRYPT_SECURITY_DESCR_SUPPORT_PROPERTY</code> L "Dukungan Dukungan Keamanan"	Menunjukkan apakah penyedia mendukung deskriptor keamanan untuk kunci.

Saat menggunakan `NCRYPT_KEY_HANDLE`, Penyedia Penyimpanan AWS CloudHSM Kunci (KSP) mendukung pengidentifikasi KSP berikut:

Pengenalan/Nilai	Deskripsi
NCRYPT_ALGORITHM_PROPERTY L “Nama Algoritma”	Sebuah string Unicode yang dihentikan nol yang berisi nama algoritma kunci.
NCRYPT_BLOCK_LENGTH_PROPERTY L “Panjang Blok”	Sebuah DWORD yang berisi panjang blok enkripsi dalam byte.
NCRYPT_EXPORT_POLICY_PROPERTY L “Kebijakan Ekspor”	DWORD yang berisi flag yang menentukan kebijakan ekspor kunci yang bertahan.
NCRYPT_KEY_USAGE_PROPERTY L “Penggunaan Kunci”	DWORD yang berisi flag yang menentukan detail penggunaan utama.
NCRYPT_KEY_TYPE_PROPERTY L “Jenis Kunci”	Sebuah DWORD yang berisi flag yang menentukan jenis kunci.
NCRYPT_LENGTH_PROPERTY L “Panjang”	Sebuah DWORD yang berisi panjang kunci dalam bit.
NCRYPT_LENGTHS_PROPERTY L “Panjang”	Penunjuk ke struktur NCRYPT_SUPPORTED_LENGTHS yang berisi ukuran kunci yang didukung.
NCRYPT_NAME_PROPERTY L “Nama”	Sebuah pointer ke string Unicode null-terminated yang berisi nama kunci.
NCRYPT_SECURITY_DESCR_PROPERTY L “Keamanan Descr”	Sebuah pointer ke struktur SECURITY_DESCRIPTOR yang berisi informasi kontrol akses kunci.
NCRYPT_ALGORITHM_GROUP_PROPERTY	Sebuah string Unicode yang dihentikan nol yang berisi nama grup algoritma objek.

Pengenal/Nilai	Deskripsi
L "Grup Algoritma"	
NCRYPT_UNIQUE_NAME_PROPERTY	Sebuah pointer ke string Unicode null-terminated yang berisi nama unik kunci.
L "Nama Unik"	

 Note

Nilai adalah string karakter lebar literal, seperti yang ditunjukkan oleh L sebelum literal.

`pbOutput[Keluar]`

Alamat buffer untuk menyimpan nilai properti. Tentukan ukuran buffer menggunakan `cbOutput`.

Untuk menentukan ukuran buffer yang diperlukan, atur parameter ini ke `NULL`. Fungsi menyimpan ukuran yang diperlukan (dalam byte) di lokasi yang ditunjuk oleh `pcbResult`.

`cbOutput[dalam]`

Ukuran `pbOutput` buffer dalam byte.

`pcbResult[Keluar]`

Pointer ke variabel `DWORD` yang menyimpan jumlah byte yang disalin ke buffer. `pbOutput`

Jika `NULL`, ini menyimpan ukuran yang diperlukan (dalam byte). `pbOutput`

`dwFlags[dalam]`

Bendera untuk memodifikasi perilaku fungsi. Anda dapat menggunakan nol atau:

Nilai	Arti
NCRYPT_SILENT_FLAG	Bendera ini tidak berpengaruh.

Ketika `PSZPropertyNCRYPT_SECURITY_DESCR_PROPERTY`, gunakan satu atau kombinasi dari:

Nilai	Arti
OWNER_SECURITY_INFORMATION	Bendera ini tidak berpengaruh.
GROUP_SECURITY_INFORMATION	Bendera ini tidak berpengaruh.
DACL_SECURITY_INFORMATION	Bendera ini tidak berpengaruh.
LABEL_SECURITY_INFORMATION	Bendera ini tidak berpengaruh.
SACL_SECURITY_INFORMATION	Bendera ini tidak berpengaruh.

Nilai yang Ditampilkan

Fungsi mengembalikan kode status untuk menunjukkan keberhasilan atau kegagalan.

Kode pengembalian umum meliputi:

Kode pengembalian	Deskripsi
ERROR_SUCCESS	Operasi selesai dengan sukses.
NTE_INVALID_PARAMETER	Satu atau lebih parameter tidak valid.
NTE_FAIL	Operasi tidak bisa selesai.
NTE_BAD_FLAGS	dwFlagsParameter berisi nilai yang tidak valid.
NTE_NOT_SUPPORTED	pszAlgIdParameter berisi nilai yang tidak didukung.
NTE_INVALID_HANDLE	Pegangan masuk hObject tidak valid.
NTE_BUFFER_TOO_SMALL	cbOutputParameternya terlalu kecil untuk mengembalikan nilai.

NCryptSetProperty dengan penyedia penyimpanan kunci (KSP)

NCryptSetPropertyFungsi menetapkan nilai properti untuk objek penyimpanan kunci.

Parameter

hObject[dalam]

Pegangan objek yang propertinya ingin Anda atur. Anda dapat menggunakan:

- Pegangan penyedia (NCRYPT_PROV_HANDLE)
- Sebuah pegangan kunci (NCRYPT_KEY_HANDLE)

pszProperty [dalam]

Sebuah pointer ke string Unicode null-terminated yang berisi nama properti untuk diambil.

Saat menggunakan NCRYPT_PROV_HANDLE, Penyedia Penyimpanan AWS CloudHSM Kunci (KSP) mendukung pengidentifikasi KSP berikut:

Pengenal/Nilai	Deskripsi
NCRYPT_USE_CONTEXT_PROPERTY L “Gunakan Konteks”	Pointer ke string Unicode yang dihentikan nol yang menjelaskan konteks operasi.

Saat menggunakan NCRYPT_KEY_HANDLE, Penyedia Penyimpanan AWS CloudHSM Kunci (KSP) mendukung pengidentifikasi KSP berikut:

Pengenal/Nilai	Deskripsi
NCRYPT_KEY_USAGE_PROPERTY L “Penggunaan Kunci”	Sebuah DWORD yang berisi satu set flag yang menentukan rincian penggunaan kunci. Properti ini hanya berlaku untuk kunci. Ini dapat berisi nol atau kombinasi dari satu atau lebih dari nilai-nilai berikut. NCRYPT_ALLOW_DECRYPT_FLAG (0x00000001) NCRYPT_ALLOW_SIGNING_FLAG (0x00000002)

Pengenalan/Nilai	Deskripsi
NCRYPT_LENGTH_PROPERTY L “Panjang”	Sebuah DWORD yang berisi panjang kunci dalam bit.
NCRYPT_EXPORT_POLICY_PROPERTY L “Kebijakan Ekspor”	DWORD yang berisi flag yang menentukan kebijakan ekspor kunci yang bertahan. Ini dapat berisi nol atau kombinasi dari satu atau lebih dari nilai-nilai berikut. NCRYPT_ALLOW_EXPORT_FLAG (0x00000001)

 Note

Nilai adalah string karakter lebar literal, seperti yang ditunjukkan oleh L sebelum literal.

pbInput[dalam]

Alamat buffer yang berisi nilai properti baru. cbInput berisi ukuran buffer.

cbInput[dalam]

Ukuran pbInput buffer dalam byte.

dwFlags[dalam]

Bendera yang memodifikasi perilaku fungsi. Tidak ada bendera yang ditentukan untuk fungsi ini.

Nilai yang Ditampilkan

Fungsi mengembalikan kode status untuk menunjukkan keberhasilan atau kegagalan.

Kode pengembalian umum meliputi:

Kode pengembalian	Deskripsi
ERROR_SUCCESS	Operasi selesai dengan sukses.

Kode pengembalian	Deskripsi
NTE_INVALID_PARAMETER	Satu atau lebih parameter tidak valid.
NTE_FAIL	Operasi tidak bisa selesai.
NTE_BAD_FLAGS	dwFlagsParameter berisi nilai yang tidak valid.
NTE_NOT_SUPPORTED	pszProperty Parameter berisi nilai yang tidak didukung.
NTE_INVALID_HANDLE	Pegangan masuk hObject tidak valid.
NTE_BAD_DATA	Data yang ditunjukkan oleh pbInput dan cbInput tidak valid.

NCryptFinalizeKey dengan penyedia penyimpanan kunci (KSP)

NCryptFinalizeKeyFungsi ini melengkapi kunci KSP. Anda harus memanggil fungsi ini sebelum Anda dapat menggunakan kunci.

Parameter

hKey[dalam]

Pegangan kunci untuk menyelesaikan. Dapatkan pegangan ini dengan memanggil [NCryptCreatePersistedKey](#) fungsi.

dwFlags[dalam]

Bendera untuk memodifikasi perilaku fungsi. Anda dapat menggunakan nol atau nilai-nilai ini:

Nilai	Arti
NCRYPT_SILENT_FLAG	Bendera ini tidak berpengaruh.
NCRYPT_NO_KEY_VALIDATION	Bendera ini tidak berpengaruh.

Nilai yang Ditampilkan

Fungsi mengembalikan kode status untuk menunjukkan keberhasilan atau kegagalan.

Kode pengembalian umum meliputi:

Kode pengembalian	Deskripsi
ERROR_SUCCESS	Operasi selesai dengan sukses.
NTE_FAIL	Operasi tidak bisa selesai.
NTE_INVALID_HANDLE	Pegangan masuk hKey tidak valid.
NTE_NOT_SUPPORTED	dwFlagsParameter berisi nilai yang tidak didukung.
NTE_BAD_FLAGS	dwFlagsParameter berisi nilai yang tidak valid.

NCryptDeleteKey dengan penyedia penyimpanan kunci (KSP)

NCryptDeleteKeyFungsi menghapus kunci KSP dari Penyedia Penyimpanan Kunci (KSP).

Parameter

hKey[dalam]

Pegangan kunci untuk dihapus.

dwFlags[dalam]

Bendera untuk memodifikasi perilaku fungsi. Anda dapat menggunakan nol atau lebih dari nilai-nilai berikut:

Nilai	Arti
NCRYPT_SILENT_FLAG	Bendera ini tidak berpengaruh.

Nilai yang Ditampilkan

Fungsi mengembalikan kode status untuk menunjukkan keberhasilan atau kegagalan.

Kode pengembalian umum meliputi:

Kode pengembalian	Deskripsi
ERROR_SUCCESS	Fungsinya berhasil.
NTE_INVALID_PARAMETER	Satu atau lebih parameter tidak valid.
NTE_BAD_FLAGS	dwFlagsParameter berisi nilai yang tidak valid.
NTE_FAIL	Operasi tidak bisa selesai.
NTE_INVALID_HANDLE	Pegangan masuk hKey tidak valid.
NTE_INTERNAL_ERROR	Kesalahan internal terjadi saat menghapus kunci.

NCryptFreeObject dengan penyedia penyimpanan kunci (KSP)

NCryptFreeObjectFungsi melepaskan provider atau key handle dari Key Storage Provider (KSP).

Parameter

hObject[dalam]

Pegangan objek untuk dilepaskan. Anda dapat menggunakan:

- Pegangan penyedia (NCRYPT_PROV_HANDLE)
- Sebuah pegangan kunci (NCRYPT_KEY_HANDLE)

Nilai yang Ditampilkan

Fungsi mengembalikan kode status untuk menunjukkan keberhasilan atau kegagalan.

Kode pengembalian umum meliputi:

Kode pengembalian	Deskripsi
ERROR_SUCCESS	Operasi selesai dengan sukses.

Kode pengembalian	Deskripsi
NTE_INVALID_HANDLE	Pegangan masuk hObject tidak valid.

NCryptFreeBuffer dengan penyedia penyimpanan kunci (KSP)

NCryptFreeBufferFungsi ini melepaskan blok memori yang dialokasikan oleh Penyedia Penyimpanan Kunci (KSP).

Parameter

pvInput[dalam]

Alamat memori yang akan dilepaskan.

Nilai yang Ditampilkan

Fungsi mengembalikan kode status untuk menunjukkan keberhasilan atau kegagalan.

Kode pengembalian umum meliputi:

Kode pengembalian	Deskripsi
ERROR_SUCCESS	Operasi selesai dengan sukses.
NTE_FAIL	Operasi tidak bisa selesai.

NCryptIsAlgSupported dengan penyedia penyimpanan kunci (KSP)

NCryptIsAlgSupported fungsi menentukan apakah Key Storage Provider (KSP) mendukung algoritma kriptografi tertentu.

Parameter

hProvider[dalam]

Pegangan penyedia penyimpanan kunci. Gunakan [NCryptOpenStorageProvider](#) untuk mendapatkan pegangan.

pszAlgId[dalam]

Pointer ke string Unicode yang dihentikan nol yang berisi pengidentifikasi algoritma kriptografi untuk membuat kunci. AWS CloudHSM Key Storage Provider (KSP) mendukung algoritma berikut:

Konstant/nilai	Deskripsi
BCRYPT_RSA_ALGORITMA "RSA"	Algoritma kunci publik RSA.
BCRYPT_ECDSA_P256_ALGORITMA "ECDSA_P256"	Algoritma tanda tangan digital kurva elips prima 256-bit (FIPS 186-2).
BCRYPT_ECDSA_P384_ALGORITMA "ECDSA_P384"	Algoritma tanda tangan digital kurva elips utama 384-bit (FIPS 186-2).
BCRYPT_ECDSA_P521_ALGORITMA "ECDSA_P521"	Algoritma tanda tangan digital kurva elips utama 521-bit (FIPS 186-2).

dwFlags[dalam]

Bendera yang memodifikasi perilaku fungsi. Ini bisa nol atau nilai berikut:

Nilai	Arti
NCRYPT_SILENT_FLAG	Bendera ini tidak berpengaruh.

Nilai yang Ditampilkan

Fungsi mengembalikan kode status untuk menunjukkan keberhasilan atau kegagalan.

Kode pengembalian umum meliputi:

Kode kembalikan	Deskripsi
ERROR_SUCCESS	Operasi selesai dengan sukses.
NTE_INVALID_PARAMETER	Satu atau lebih parameter tidak valid.
NTE_BAD_FLAGS	dwFlagsParameter berisi nilai yang tidak valid.
NTE_NOT_SUPPORTED	pszAlgIdParameter berisi nilai yang tidak didukung.
NTE_INVALID_HANDLE	Pegangan masuk hProvider tidak valid.

NCryptEnumAlgorithms dengan penyedia penyimpanan kunci (KSP)

NCryptEnumAlgorithmsFungsi ini mengambil nama-nama algoritma yang didukung oleh Penyedia Penyimpanan Kunci (KSP).

Parameter

hProvider[dalam]

Pegangan penyedia penyimpanan kunci untuk menghitung algoritma. Gunakan [NCryptOpenStorageProvider](#) fungsi untuk mendapatkan pegangan ini.

dwAlgOperations[dalam]

Satu set nilai yang menentukan kelas algoritma mana yang akan dihitung. Anda dapat menggunakan nol untuk menghitung semua algoritma, atau menggabungkan satu atau lebih dari nilai-nilai ini:

Nilai	Arti
NCRYPT_ASYMMETRIC_ENCRYPTIO N_OPERATION 0x00000004	Buat daftar algoritma enkripsi asimetris.
NCRYPT_SIGNATURE_OPERATION	Buat daftar algoritma tanda tangan digital.

Nilai	Arti
0x00000010	

pdwAlgCount[Keluar]

Alamat DWORD yang menyimpan jumlah elemen dalam ppAlgList array.

ppAlgList[Keluar]

Alamat pointer NCryptAlgorithmName struktur yang menyimpan array nama algoritma terdaftar. pdwAlgCountParameter menunjukkan jumlah elemen dalam array ini.

dwFlags[dalam]

Bendera untuk memodifikasi perilaku fungsi. Gunakan nol atau nilai berikut:

Nilai	Arti
NCRYPT_SILENT_FLAG	Bendera ini tidak berpengaruh.

Nilai yang Ditampilkan

Fungsi mengembalikan kode status untuk menunjukkan keberhasilan atau kegagalan.

Kode pengembalian umum meliputi:

Kode pengembalian	Deskripsi
ERROR_SUCCESS	Operasi selesai dengan sukses.
NTE_INVALID_PARAMETER	Satu atau lebih parameter tidak valid.
NTE_FAIL	Operasi tidak bisa selesai.
NTE_BAD_FLAGS	dwFlagsParameter berisi nilai yang tidak valid.
NTE_NOT_SUPPORTED	dwAlgOperations Parameter berisi nilai yang tidak didukung.

NCryptEnumKeys dengan penyedia penyimpanan kunci (KSP)

NCryptEnumKeys fungsi daftar kunci yang disimpan di Penyedia Penyimpanan Kunci (KSP).

Parameter

hProvider[dalam]

Pegangan penyedia penyimpanan utama. Gunakan [NCryptOpenStorageProvider](#) untuk mendapatkan pegangan ini.

pszScope[di, tidak terpakai]

Setel parameter ini ke NULL.

ppKeyName[keluar]

Alamat pointer ke NCryptKeyName struktur yang menyimpan nama kunci. Untuk membebaskan memori ini setelah digunakan, hubungi [NCryptFreeBuffer](#).

ppEnumState[masuk, keluar]

Alamat pointer VOID yang melacak kemajuan enumerasi. Penyedia penyimpanan kunci menggunakan informasi ini secara internal untuk mengelola urutan enumerasi. Untuk memulai enumerasi baru dari awal, atur pointer ini ke NULL.

Untuk membebaskan memori ini setelah menyelesaikan enumerasi, berikan pointer ini ke file. [NCryptFreeBuffer](#)

dwFlags[dalam]

Bendera untuk memodifikasi perilaku fungsi. Fungsi ini tidak memiliki bendera.

Nilai yang Ditampilkan

Fungsi mengembalikan kode status untuk menunjukkan keberhasilan atau kegagalan.

Kode pengembalian umum meliputi:

Kode pengembalian	Deskripsi
ERROR_SUCCESS	Operasi selesai dengan sukses.
NTE_INVALID_PARAMETER	Satu atau lebih parameter tidak valid.

Kode pengembalian	Deskripsi
NTE_FAIL	Operasi tidak bisa selesai.
NTE_INVALID_HANDLE	Pegangan masuk hProvider tidak valid.
NTE_NO_MORE_ITEMS	Enumerasi telah mencantumkan semua kunci yang tersedia.

NCryptExportKey dengan penyedia penyimpanan kunci (KSP)

NCryptExportKeyFungsi mengekspor kunci KSP ke memori. BLOB Fungsi ini hanya mendukung ekspor kunci publik.

Parameter

hKey[dalam]

Pegangan kunci untuk mengekspor.

hExportKey[di, tidak terpakai]

AWS CloudHSM Penyedia Penyimpanan Kunci (KSP) tidak menggunakan parameter ini.

pszBlobType[dalam]

String Unicode yang dihentikan nol yang menentukan jenis yang akan diekspor. BLOB AWS CloudHSM Penyedia Penyimpanan Kunci (KSP) mendukung nilai-nilai berikut:

Nilai	Arti
BCRYPT_RSAPUBLIC_BLOB	Mengekspor kunci publik RSA. pbOutputBuffer berisi BCRYPT_RS AKEY_BLOB struktur yang diikuti oleh data kunci.
BCRYPT_ECCPUBLIC_BLOB	Mengekspor kunci ECC publik. pbOutputBuffer berisi BCRYPT_EC CKEY_BLOB struktur yang diikuti oleh data kunci.

pParameterList[di, tidak terpakai]

AWS CloudHSM Penyedia Penyimpanan Kunci (KSP) tidak menggunakan parameter ini.

pbOutput[keluar, opsional]

Alamat buffer untuk menyimpan BLOB kunci. Tentukan ukuran buffer menggunakan cbOutput. Jika disetel ke NULL, fungsi menyimpan ukuran yang diperlukan (dalam byte) di DWORD yang ditunjuk oleh. pcbResult

cbOutput[dalam]

Ukuran pbOutput buffer dalam byte.

pcbResult[keluar]

Alamat variabel DWORD yang menyimpan jumlah byte yang disalin ke buffer. pbOutput. Jika pbOutput NULL, fungsi menyimpan ukuran buffer yang diperlukan dalam byte.

dwFlags[dalam]

Bendera yang memodifikasi cara kerja fungsi. Anda dapat menggunakan nol atau berikut ini:

Nilai	Arti
NCRYPT_SILENT_FLAG	Bendera ini tidak berpengaruh.

Nilai yang Ditampilkan

Fungsi mengembalikan kode status untuk menunjukkan keberhasilan atau kegagalan.

Kode pengembalian umum meliputi:

Kode pengembalian	Deskripsi
ERROR_SUCCESS	Operasi selesai dengan sukses.
NTE_INVALID_PARAMETER	Satu atau lebih parameter tidak valid.
NTE_FAIL	Operasi tidak bisa selesai.
NTE_INVALID_HANDLE	Pegangan masuk hProvider tidak valid.

Kode pengembalian	Deskripsi
NTE_BAD_FLAGS	dwFlagsParameter berisi nilai yang tidak valid.
NTE_BAD_KEY_STATE	Status kunci tidak valid.
NTE_NOT_SUPPORTED	dwFlagsParameter pszBlobType atau berisi nilai yang tidak didukung.
STATUS_INTERNAL_ERROR	Kesalahan internal terjadi selama operasi.

NCryptSignHash dengan penyedia penyimpanan kunci (KSP)

NCryptSignHashFungsi ini menciptakan tanda tangan dari nilai hash.

Parameter

hKey[dalam]

Pegangan kunci yang digunakan untuk menandatangani hash.

pPaddingInfo[dalam, opsional]

Sebuah pointer ke struktur yang berisi informasi padding. Jenis struktur tergantung pada dwFlags nilainya. Gunakan parameter ini hanya dengan tombol asimetris; atur ke NULL untuk jenis kunci lainnya.

pbHashValue[dalam]

Sebuah pointer ke buffer yang berisi nilai hash untuk ditandatangani. Tentukan ukuran buffer menggunakan cbHashValue.

cbHashValue[dalam]

Ukuran, dalam byte, dari pbHashValue buffer untuk ditandatangani.

pbSignature[keluar]

Alamat buffer untuk menyimpan tanda tangan. Tentukan ukuran buffer menggunakan cbSignature.

Untuk menentukan ukuran buffer yang diperlukan, atur parameter ini ke NULL. Fungsi menyimpan ukuran yang diperlukan (dalam byte) di lokasi yang ditunjuk oleh pcbResult.

cbSignature[dalam]

Ukuran pbSignature buffer dalam byte. Fungsi mengabaikan parameter ini jika pbSignature NULL.

pcbResult[keluar]

Pointer ke variabel DWORD yang menyimpan jumlah byte yang disalin ke buffer. pbSignature

Jika pbSignature NULL, ini menyimpan ukuran buffer yang diperlukan, dalam byte.

dwFlags[dalam]

Bendera untuk memodifikasi perilaku fungsi. Bendera yang diizinkan bergantung pada jenis kunci Anda. Gunakan salah satu dari nilai-nilai ini:

Nilai	Arti
BCRYPT_PAD_PKCS1	Menggunakan skema PKCS1 padding. Atur pPaddingInfo untuk menunjuk ke BCRYPT_PKCS1_PADDING_INFO struktur.
BCRYPT_PAD_PSS	Menggunakan skema padding Probabilistic Signature Scheme (PSS). Atur pPaddingInfo parameter untuk menunjuk ke BCRYPT_PSS_PADDING_INFO struktur.
NCRYPT_SILENT_FLAG	Bendera ini tidak berpengaruh.

Nilai yang Ditampilkan

Fungsi mengembalikan kode status untuk menunjukkan keberhasilan atau kegagalan.

Kode pengembalian umum meliputi:

Kode pengembalian	Deskripsi
ERROR_SUCCESS	Operasi selesai dengan sukses.
NTE_INVALID_PARAMETER	Satu atau lebih parameter tidak valid.

Kode pengembalian	Deskripsi
NTE_FAIL	Operasi tidak bisa selesai.
NTE_INVALID_HANDLE	Pegangan masuk hKey tidak valid.
NTE_BAD_FLAGS	dwFlagsParameter berisi nilai yang tidak valid.
NTE_BUFFER_TOO_SMALL	pcbOutput Parameternya terlalu kecil untuk mengembalikan nilai.
NTE_BAD_KEY_STATE	Status kunci tidak valid.
NTE_INTERNAL_ERROR	Kesalahan internal terjadi saat menandatangani hash.

NCryptVerifySignature dengan penyedia penyimpanan kunci (KSP)

NCryptVerifySignatureFungsi ini mengonfirmasi apakah tanda tangan cocok dengan hash tertentu.

Parameter

hKey[dalam]

Pegangan kunci yang digunakan untuk mendekripsi tanda tangan. Anda harus menggunakan bagian kunci publik dari key pair yang digunakan untuk menandatangani data dengan [NCryptSignHash](#).

pPaddingInfo[dalam, opsional]

Sebuah pointer ke struktur yang berisi informasi padding. Jenis struktur tergantung pada dwFlags nilainya. Gunakan parameter ini hanya dengan tombol asimetris; atur ke NULL untuk jenis kunci lainnya.

pbHashValue[dalam]

Sebuah pointer ke buffer yang berisi nilai hash untuk ditandatangani. Tentukan ukuran buffer menggunakan cbHashValue.

cbHashValue[dalam]

Ukuran pbHashValue buffer dalam byte.

pbSignature[Keluar]

Alamat buffer yang berisi hash data yang ditandatangani. Gunakan [NCryptSignHash](#) untuk membuat tanda tangan ini. Tentukan ukuran buffer menggunakan `cbSignature`.

cbSignature[dalam]

Ukuran `pbSignature` buffer dalam byte. Gunakan [NCryptSignHash](#) untuk membuat tanda tangan.

dwFlags[dalam]

Bendera untuk memodifikasi perilaku fungsi. Bendera yang diizinkan bergantung pada jenis kunci Anda. Gunakan salah satu dari nilai-nilai ini:

Nilai	Arti
PKCS1NCRYPT_PAD__BENDERA	Menunjukkan tanda tangan yang digunakan PKCS1 padding. Atur <code>pPaddingInfo</code> untuk menunjuk ke <code>BCRYPT_PKCS1_PADDING_INFO</code> struktur.
NCRYPT_PAD_PSS_FLAG	Menunjukkan tanda tangan yang digunakan Probabilistic Signature Scheme (PSS) padding. Atur <code>pPaddingInfo</code> untuk menunjuk ke <code>BCRYPT_PSS_PADDING_INFO</code> struktur.
NCRYPT_SILENT_FLAG	Bendera ini tidak berpengaruh.

Nilai yang Ditampilkan

Fungsi mengembalikan kode status untuk menunjukkan keberhasilan atau kegagalan.

Kode pengembalian umum meliputi:

Kode kembalikan	Deskripsi
ERROR_SUCCESS	Operasi selesai dengan sukses.

Kode kembalikan	Deskripsi
NTE_INVALID_PARAMETER	Satu atau lebih parameter tidak valid.
NTE_FAIL	Operasi tidak bisa selesai.
NTE_INVALID_HANDLE	Pegangan masuk hKey tidak valid.
NTE_BAD_FLAGS	dwFlagsParameter berisi nilai yang tidak valid.
NTE_BAD_SIGNATURE	Tanda tangan tidak diverifikasi.
NTE_BAD_KEY_STATE	Status kunci tidak valid.
NTE_INTERNAL_ERROR	Kesalahan internal terjadi saat memverifikasi tanda tangan.

Konfigurasi lanjutan untuk KSP untuk AWS CloudHSM

Penyedia Penyimpanan AWS CloudHSM Kunci (KSP) mencakup konfigurasi lanjutan berikut, yang bukan merupakan bagian dari konfigurasi umum yang digunakan sebagian besar pelanggan. Konfigurasi ini memberikan kemampuan tambahan.

- [SDK3 mode kompatibilitas untuk KSP](#)

SDK3 mode kompatibilitas untuk Penyedia Penyimpanan Kunci (KSP) untuk AWS CloudHSM

Penyedia Penyimpanan Kunci (KSP) menerapkan pendekatan yang berbeda untuk interaksi kunci HSM:

- Klien SDK 5: Menyediakan komunikasi langsung dengan kunci yang disimpan di HSM, menghilangkan kebutuhan akan file referensi lokal
- Client SDK 3: Mempertahankan file lokal di server Windows yang bertindak sebagai referensi ke kunci yang disimpan di HSM, menggunakan file-file ini untuk memfasilitasi operasi utama

Untuk pelanggan yang bermigrasi dari Client SDK 3 ke Client SDK 5, mengaktifkan opsi mode SDK3 kompatibilitas mendukung operasi menggunakan file referensi kunci yang ada sambil mempertahankan arsitektur penyimpanan kunci HSM yang mendasarinya.

Aktifkan mode SDK3 kompatibilitas

Windows

Untuk mengaktifkan mode SDK3 kompatibilitas untuk Penyedia Penyimpanan Kunci (KSP) untuk Klien SDK 5 di Windows

- Anda dapat menggunakan perintah berikut untuk mengaktifkan mode SDK3 kompatibilitas:

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --enable-  
sdk3-compatibility-mode
```

Nonaktifkan mode SDK3 kompatibilitas

Windows

Untuk menonaktifkan mode SDK3 kompatibilitas untuk Penyedia Penyimpanan Kunci (KSP) untuk Klien SDK 5 di Windows

- Anda dapat menggunakan perintah berikut untuk menonaktifkan mode SDK3 kompatibilitas:

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-ksp.exe" --disable-  
sdk3-compatibility-mode
```

Penyedia JCE untuk AWS CloudHSM Klien SDK 5

Penyedia AWS CloudHSM JCE adalah implementasi penyedia yang dibangun dari kerangka penyedia Java Cryptographic Extension (JCE). JCE memungkinkan Anda untuk melakukan operasi kriptografi menggunakan Java Development Kit (JDK). Dalam panduan ini, penyedia AWS CloudHSM JCE kadang-kadang disebut sebagai penyedia JCE. Gunakan penyedia JCE dan JDK untuk membongkar operasi kriptografi ke HSM. Untuk pemecahan masalah, lihat. [Masalah yang diketahui untuk JCE SDK untuk AWS CloudHSM](#)

Untuk informasi tentang penggunaan SDK Klien 3, lihat [Menggunakan versi SDK sebelumnya untuk bekerja dengan AWS CloudHSM](#).

Topik

- [Instal penyedia JCE untuk AWS CloudHSM Client SDK 5](#)

- [Tipe kunci yang didukung untuk penyedia JCE untuk AWS CloudHSM Client SDK 5](#)
- [Dasar-dasar manajemen utama di penyedia JCE untuk AWS CloudHSM Client SDK 5](#)
- [Mekanisme yang didukung untuk penyedia JCE untuk AWS CloudHSM Client SDK 5](#)
- [Atribut kunci Java yang didukung untuk AWS CloudHSM Klien SDK 5](#)
- [Contoh kode untuk pustaka AWS CloudHSM perangkat lunak untuk Java untuk Klien SDK 5](#)
- [AWS CloudHSM JCE penyedia Javadocs](#)
- [AWS CloudHSM KeyStore Kelas Java untuk Client SDK 5](#)
- [Konfigurasi lanjutan untuk AWS CloudHSM JCE untuk Client SDK 5](#)

Instal penyedia JCE untuk AWS CloudHSM Client SDK 5

Penyedia JCE untuk AWS CloudHSM Client SDK 5 kompatibel dengan OpenJDK 8, OpenJDK 11, OpenJDK 17, OpenJDK 21, dan OpenJDK 25. Anda dapat mengunduh keduanya dari situs web [OpenJDK](#).

Gunakan bagian berikut untuk menginstal dan memberikan kredensial kepada penyedia.

Note

Untuk menjalankan satu klaster HSM dengan SDK Klien 5, Anda harus terlebih dahulu mengelola pengaturan daya tahan kunci klien dengan menetapkan `disable_key_availability_check` ke `True`. Untuk informasi selengkapnya, lihat [Sinkronisasi Kunci](#) dan [Alat Konfigurasi SDK Klien 5](#).

Topik

- [Langkah 1: Instal penyedia JCE](#)
- [Langkah 2: Berikan kredensial ke penyedia JCE](#)

Langkah 1: Instal penyedia JCE

1. Gunakan perintah berikut untuk mengunduh dan menginstal penyedia JCE.

Amazon Linux 2023

Instal penyedia JCE untuk Amazon Linux 2023 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-jce-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.amzn2023.x86_64.rpm
```

Instal penyedia JCE untuk Amazon Linux 2023 pada ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-jce-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.amzn2023.aarch64.rpm
```

Amazon Linux 2

Instal penyedia JCE untuk Amazon Linux 2 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el7.x86_64.rpm
```

Instal penyedia JCE untuk Amazon Linux 2 pada ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-jce-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el7.aarch64.rpm
```

RHEL 10 (10.0+)

Instal penyedia JCE untuk RHEL 10 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/cloudhsm-jce-latest.el10.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el10.x86_64.rpm
```

Instal penyedia JCE untuk RHEL 10 pada ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL10/cloudhsm-jce-latest.el10.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el10.aarch64.rpm
```

RHEL 9 (9.2+)

Instal penyedia JCE untuk RHEL 9 (9.2+) pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-jce-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el9.x86_64.rpm
```

Instal penyedia JCE untuk RHEL 9 (9.2+) pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-jce-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el9.aarch64.rpm
```

RHEL 8 (8.3+)

Instal penyedia JCE untuk RHEL 8 pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-jce-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el8.x86_64.rpm
```

Instal penyedia JCE untuk RHEL 8 pada ARM64 arsitektur:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-jce-latest.el8.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el8.aarch64.rpm
```

Ubuntu 24.04 LTS

Instal penyedia JCE untuk Ubuntu 24.04 LTS pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-jce_latest_u24.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u24.04_amd64.deb
```

Instal penyedia JCE untuk Ubuntu 24.04 LTS pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Noble/cloudhsm-jce_latest_u24.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u24.04_arm64.deb
```

Ubuntu 22.04 LTS

Instal penyedia JCE untuk Ubuntu 22.04 LTS pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-jce_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u22.04_amd64.deb
```

Instal penyedia JCE untuk Ubuntu 22.04 LTS pada arsitektur: ARM64

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-jce_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u22.04_arm64.deb
```

Windows Server

Instal penyedia JCE untuk Windows Server pada arsitektur x86_64, buka PowerShell sebagai administrator dan jalankan perintah berikut:

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/AWSCloudHSMJCE-latest.msi -Outfile C:\AWSCloudHSMJCE-latest.msi
```

```
PS C:\> Start-Process msixexec.exe -ArgumentList '/i C:\AWSCloudHSMJCE-latest.msi /quiet /norestart /log C:\client-install.txt' -Wait
```

2. SDK Klien Bootstrap 5. Untuk informasi selengkapnya tentang bootstrapping, lihat [Bootstrap Klien SDK](#).
3. Temukan file penyedia JCE berikut:

Linux

- /opt/cloudhsm/java/cloudhsm-*<version>*.jar
- /opt/cloudhsm/bin/configure-jce
- /opt/cloudhsm/bin/jce-info

Windows

- C:\Program Files\Amazon\CloudHSM\java\cloudhsm-*<version>*.jar>
- C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe
- C:\Program Files\Amazon\CloudHSM\bin\jce_info.exe

Langkah 2: Berikan kredensial ke penyedia JCE

Sebelum aplikasi Java Anda dapat menggunakan HSM, HSM harus terlebih dahulu mengotentikasi aplikasi. HSMs mengotentikasi menggunakan login eksplisit atau metode login implisit.

Login eksplisit - Metode ini memungkinkan Anda memberikan AWS CloudHSM kredensial langsung dalam aplikasi. Ini menggunakan metode dari [AuthProvider](#), di mana Anda melewati nama

pengguna dan kata sandi CU dalam pola pin. Untuk informasi selengkapnya, lihat [Login ke contoh kode HSM](#).

Login implisit — Metode ini memungkinkan Anda mengatur AWS CloudHSM kredensial baik dalam file properti baru, properti sistem, atau sebagai variabel lingkungan.

- Properti sistem — Set kredensial melalui properti sistem saat menjalankan aplikasi Anda. Contoh berikut menunjukkan dua cara berbeda yang dapat dilakukan:

Linux

```
$ java -DHSM_USER=<HSM user name> -DHSM_PASSWORD=<password>
```

```
System.setProperty("HSM_USER", "<HSM user name>");  
System.setProperty("HSM_PASSWORD", "<password>");
```

Windows

```
PS C:\> java -DHSM_USER=<HSM user name> -DHSM_PASSWORD=<password>
```

```
System.setProperty("HSM_USER", "<HSM user name>");  
System.setProperty("HSM_PASSWORD", "<password>");
```

- Variabel lingkungan — Set kredensial sebagai variabel lingkungan.

Note

Saat mengatur variabel lingkungan, Anda harus melarikan diri dari karakter khusus apa pun yang dapat ditafsirkan oleh shell Anda.

Linux

```
$ export HSM_USER=<HSM user name>  
$ export HSM_PASSWORD=<password>
```

Windows

```
PS C:\> $Env:HSM_USER="<HSM user name>"
```

```
PS C:\> $Env:HSM_PASSWORD="<password>"
```

Kredensial mungkin tidak tersedia jika aplikasi tidak menyediakannya atau jika Anda mencoba operasi sebelum HSM mengautentikasi sesi. Dalam kasus tersebut, pustaka perangkat lunak CloudHSM untuk Java mencari kredensialnya dengan urutan sebagai berikut:

1. Properti sistem
2. Variabel-variabel lingkungan

Tipe kunci yang didukung untuk penyedia JCE untuk AWS CloudHSM Client SDK 5

Perpustakaan AWS CloudHSM perangkat lunak untuk Java memungkinkan Anda untuk menghasilkan jenis kunci berikut.

Tipe Kunci	Deskripsi
AES	Hasilkan kunci AES 128, 192, dan 256-bit.
Tiga DES (3DES, DESede)	Hasilkan Kunci Triple DES 192-bit. *
EC	Hasilkan pasangan kunci EC - kurva NIST secp224r1 (P-224), secp256r1 (P-256), secp256k1 (Blockchain), secp384r1 (P-384), dan secp521r1 (P-521).
GENERIC_SECRET	Hasilkan 1 hingga 800 byte rahasia generik.
HMAC	Dukungan hash untuk SHA1,, SHA224, SHA256 SHA384, SHA512.
RSA	Hasilkan 2048-bit sampai 4096-bit kunci RSA, dengan penambahan 256 bit.

* Sesuai dengan panduan NIST, ini dilarang untuk cluster dalam mode FIPS setelah 2023. Untuk cluster dalam mode non-FIPS, masih diperbolehkan setelah 2023. Lihat [Kepatuhan FIPS 140: Penutupan Mekanisme 2024](#) untuk rincian selengkapnya.

Dasar-dasar manajemen utama di penyedia JCE untuk AWS CloudHSM Client SDK 5

Dasar-dasar manajemen kunci dalam penyedia JCE melibatkan mengimpor kunci, mengeksport kunci, memuat kunci dengan pegangan, atau menghapus kunci. Untuk informasi lebih lanjut tentang mengelola kunci, lihat sampel kode [Kelola kunci](#).

Anda juga dapat menemukan lebih banyak contoh kode penyedia JCE di [Sampel Kode](#).

Mekanisme yang didukung untuk penyedia JCE untuk AWS CloudHSM Client SDK 5

Topik ini memberikan informasi tentang mekanisme yang didukung untuk penyedia JCE dengan AWS CloudHSM Client SDK 5. Untuk informasi tentang antarmuka Java Cryptography Architecture (JCA) dan kelas mesin yang didukung oleh AWS CloudHSM, lihat topik berikut.

Topik

- [Menghasilkan fungsi key dan key pair](#)
- [Fungsi cipher](#)
- [Tanda tangani dan verifikasi fungsi](#)
- [Fungsi mencerna](#)
- [Fungsi kode otentikasi pesan berbasis hash \(HMAC\)](#)
- [Fungsi kode otentikasi pesan berbasis sandi \(CMAC\)](#)
- [Fungsi Perjanjian Utama](#)
- [Konversi kunci ke spesifikasi utama menggunakan pabrik-pabrik utama](#)
- [Anotasi mekanisme](#)

Menghasilkan fungsi key dan key pair

Pustaka AWS CloudHSM perangkat lunak untuk Java memungkinkan Anda untuk menggunakan operasi berikut untuk menghasilkan fungsi key dan key pair.

- RSA
- EC
- AES
- DESede (Triple DES) ^{lihat catatan [1](#)}
- GenericSecret

Fungsi cipher

Pustaka AWS CloudHSM perangkat lunak untuk Java mendukung kombinasi algoritma, mode, dan padding berikut.

Algoritme	Mode	Bantalan	Catatan
AES	CBC	AES/CBC/N oPadding AES/CBC/P KCS5Padding	Menerapkan Cipher.EN CRYPT_MODE dan Cipher.DE CRYPT_MODE . Menerapkan Cipher.UN WRAP_MODE for AES/CBC NoPadding
AES	ECB	AES/ECB/P KCS5Padding AES/ECB/N oPadding	Menerapkan Cipher.EN CRYPT_MODE dan Cipher.DE CRYPT_MODE .
AES	CTR	AES/CTR/N oPadding	Menerapkan Cipher.EN CRYPT_MODE dan Cipher.DE CRYPT_MODE .
AES	GCM	AES/GCM/N oPadding	Menerapkan Cipher.WR AP_MODE , Cipher.UN WRAP_MODE , Cipher.EN CRYPT_MODE ,

Algoritme	Mode	Bantalan	Catatan
			dan <code>Cipher.DECRYPT_MODE</code> . Ketika melakukan enkripsi AES-GCM, HSM mengabaikan vektor inisialisasi (IV) dalam permintaan dan menggunakan IV yang dihasilkan. Saat operasi selesai, Anda harus memanggil <code>Cipher.getIV()</code> untuk mendapatkan IV.
AESWrap	ECB	AESWrap/ECB/NoPadding AESWrap/ECB/PKCS5Padding AESWrap/ECB/ZeroPadding	Menerapkan <code>Cipher.WRAP_MODE</code> dan <code>Cipher.UNWRAP_MODE</code> .
DESede (Tiga DES)	CBC	DESede/CBC/PKCS5Padding DESede/CBC/NoPadding	Menerapkan <code>Cipher.ENCRYPT_MODE</code> dan <code>Cipher.DECRYPT_MODE</code> . Lihat catatan 1 di bawah untuk perubahan yang akan datang.

Algoritme	Mode	Bantalan	Catatan
DESede (Tiga DES)	ECB	DESede/ECB/ NoPadding DESede/ECB/ PKCS5Padding	Menerapkan Cipher.EN CRYPT_MODE dan Cipher.DE CRYPT_MODE . Lihat catatan 1 di bawah untuk perubahan yang akan datang.

Algoritme	Mode	Bantalan	Catatan
RSA	ECB	RSA/ECB/P KCS1Padding lihat catatan 1	Menerapkan Cipher.WR AP_MODE , Cipher.UN WRAP_MODE , Cipher.EN CRYPT_MODE , dan Cipher.DE CRYPT_MODE .
		RSA/ECB/0 AEPPadding	
		RSA/ECB/0 AEPWithSH A-1ANDMGF 1Padding	
		RSA/ECB/0 AEPWithSH A-224ANDM GF1Padding	
		RSA/ECB/0 AEPWithSH A-256ANDM GF1Padding	
		RSA/ECB/0 AEPWithSH A-384ANDM GF1Padding	
		RSA/ECB/0 AEPWithSH A-512ANDM GF1Padding	

Algoritme	Mode	Bantalan	Catatan
RSA	ECB	RSA/ECB/NoPadding	Menerapkan Cipher.ENCRYPT_MODE dan Cipher.DECRYPT_MODE .
RSAAESWrap	ECB	RSAAESWrap/ECB/OAEP Padding RSAAESWrap/ECB/OAEPWithSHA-1 ANDMGF1Padding RSAAESWrap/ECB/OAEPWithSHA-224 ANDMGF1Padding RSAAESWrap/ECB/OAEPWithSHA-256 ANDMGF1Padding RSAAESWrap/ECB/OAEPWithSHA-384 ANDMGF1Padding RSAAESWrap/ECB/OAEPWithSHA-512 ANDMGF1Padding	Menerapkan Cipher.WRAP_MODE dan Cipher.UNWRAP_MODE .

Tanda tangan dan verifikasi fungsi

Pustaka AWS CloudHSM perangkat lunak untuk Java mendukung jenis tanda tangan dan verifikasi berikut. Dengan Client SDK 5 dan algoritma tanda tangan dengan hashing, data di-hash secara lokal dalam perangkat lunak sebelum dikirim ke HSM untuk tanda tangan/verifikasi. Ini berarti tidak ada batasan ukuran data yang dapat di-hash oleh SDK.

Jenis Tanda Tangan RSA

- `NONEwithRSA`
- `RSASSA-PSS`
- `SHA1withRSA`
- `SHA1withRSA/PSS`
- `SHA1withRSAandMGF1`
- `SHA224withRSA`
- `SHA224withRSAandMGF1`
- `SHA224withRSA/PSS`
- `SHA256withRSA`
- `SHA256withRSAandMGF1`
- `SHA256withRSA/PSS`
- `SHA384withRSA`
- `SHA384withRSAandMGF1`
- `SHA384withRSA/PSS`
- `SHA512withRSA`
- `SHA512withRSAandMGF1`
- `SHA512withRSA/PSS`

Jenis Tanda Tangan ECDSA

- `NONEwithECDSA`
- `SHA1withECDSA`
- `SHA224withECDSA`

- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

Fungsi mencerna

Pustaka AWS CloudHSM perangkat lunak untuk Java mendukung intisari pesan berikut. Dengan Client SDK 5, data di-hash secara lokal dalam perangkat lunak. Ini berarti tidak ada batasan ukuran data yang dapat di-hash oleh SDK.

- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512

Fungsi kode otentikasi pesan berbasis hash (HMAC)

Pustaka AWS CloudHSM perangkat lunak untuk Java mendukung algoritma HMAC berikut.

- HmacSHA1(Ukuran data maksimum dalam byte: 16288)
- HmacSHA224(Ukuran data maksimum dalam byte: 16256)
- HmacSHA256(Ukuran data maksimum dalam byte: 16288)
- HmacSHA384(Ukuran data maksimum dalam byte: 16224)
- HmacSHA512(Ukuran data maksimum dalam byte: 16224)

Fungsi kode otentikasi pesan berbasis sandi (CMAC)

CMACs (Kode otentikasi pesan berbasis sandi) buat kode otentikasi pesan (MACs) menggunakan sandi blok dan kunci rahasia. Mereka berbeda dari HMACs dalam bahwa mereka menggunakan metode kunci simetris blok untuk MACs bukan metode hashing.

Pustaka AWS CloudHSM perangkat lunak untuk Java mendukung algoritma CMAC berikut.

- AESCMAC

Fungsi Perjanjian Utama

Pustaka AWS CloudHSM perangkat lunak untuk Java mendukung ECDH dengan Key Derivation Functions (KDF). Jenis KDF berikut didukung:

- `ECDHwithX963SHA1KDF` Mendukung algoritma X9.63 KDF SHA1 [2](#)
- `ECDHwithX963SHA224KDF` Mendukung algoritma X9.63 KDF SHA224 [2](#)
- `ECDHwithX963SHA256KDF` Mendukung algoritma X9.63 KDF SHA256 [2](#)
- `ECDHwithX963SHA384KDF` Mendukung algoritma X9.63 KDF SHA384 [2](#)
- `ECDHwithX963SHA512KDF` Mendukung algoritma X9.63 KDF SHA512 [2](#)

Konversi kunci ke spesifikasi utama menggunakan pabrik-pabrik utama

Anda dapat menggunakan pabrik-pabrik utama untuk mengonversi kunci ke spesifikasi utama. AWS CloudHSM memiliki dua jenis pabrik utama untuk JCE:

SecretKeyFactory: Digunakan untuk mengimpor atau menurunkan kunci simetris. Dengan menggunakan `SecretKeyFactory`, Anda dapat meneruskan Kunci yang didukung atau didukung `KeySpec` untuk mengimpor atau memperoleh kunci simetris. AWS CloudHSM Berikut ini adalah spesifikasi yang didukung untuk `KeyFactory`:

- `generateSecret` Metode untuk `SecretKeyFactory` [KeySpec](#) kelas berikut didukung:
 - `KeyAttributesMap` dapat digunakan untuk mengimpor byte kunci dengan atribut tambahan sebagai CloudHSM Key. Contohnya dapat ditemukan di sini di [sini](#).
 - [SecretKeySpec](#) dapat digunakan untuk mengimpor spesifikasi kunci simetris sebagai Kunci CloudHSM.
 - `AesCmacKdfParameterSpec` dapat digunakan untuk menurunkan kunci simetris menggunakan CloudHSM AES Key lain.

Note

`SecretKeyFactory.translateKey` Metode ini mengambil kunci apa pun yang mengimplementasikan antarmuka [Kunci](#).

KeyFactory: Digunakan untuk mengimpor kunci asimetris. Dengan menggunakan KeyFactory, Anda dapat meneruskan Kunci yang didukung atau didukung KeySpec untuk mengimpor kunci AWS CloudHSM asimetris. Untuk informasi lebih lanjut, lihat sumber daya berikut:

- generatePublicMetode KeyFactory For, [KeySpec](#) kelas berikut didukung:
- KeyAttributesMap CloudHSM untuk RSA dan EC, termasuk: KeyTypes
 - KeyAttributesMap CloudHSM untuk RSA dan EC publik. KeyTypes Contoh dapat ditemukan [di sini](#)
 - [X509 EncodedKeySpec](#) untuk Kunci Publik RSA dan EC
 - [RSAPublicKeySpec](#) untuk Kunci Publik RSA
 - [ECPublicKeySpec](#) untuk Kunci Publik EC
- generatePrivateMetode KeyFactory For, [KeySpec](#) kelas berikut didukung:
- KeyAttributesMap CloudHSM untuk RSA dan EC, termasuk: KeyTypes
 - KeyAttributesMap CloudHSM untuk RSA dan EC publik. KeyTypes Contoh dapat ditemukan [di sini](#)
 - [PKCS8EncodedKeySpec](#) untuk EC dan RSA Private Key
 - [RSAPrivateCrtKeySpec](#) untuk Kunci Pribadi RSA
 - [ECPrivateKeySpec](#) untuk Kunci Pribadi EC

translateKeyMetode KeyFactory For, dibutuhkan setiap Key yang mengimplementasikan [Key Interface](#).

Anotasi mekanisme

[1] Sesuai dengan panduan NIST, ini tidak diizinkan untuk cluster dalam mode FIPS setelah 2023. Untuk cluster dalam mode non-FIPS, masih diperbolehkan setelah 2023. Lihat [Kepatuhan FIPS 140: Penutupan Mekanisme 2024](#) untuk detail.

[2] Fungsi derivasi kunci (KDFs) ditentukan dalam [Publikasi Khusus NIST 800-56A](#) Revisi 3.

Atribut kunci Java yang didukung untuk AWS CloudHSM Klien SDK 5

Topik ini memberikan informasi tentang atribut kunci Java yang didukung untuk AWS CloudHSM Client SDK 5. Topik ini menjelaskan cara menggunakan ekstensi berpemilik untuk penyedia JCE untuk menetapkan atribut kunci. Gunakan ekstensi ini untuk mengatur atribut kunci yang didukung dan nilai-nilainya selama operasi ini:

- Pembuatan kunci
- Impor kunci

Untuk contoh cara menggunakan atribut kunci, lihat [the section called “Sampel Kode”](#).

Topik

- [Memahami atribut](#)
- [Atribut yang didukung](#)
- [Menyetel atribut untuk kunci](#)

Memahami atribut

Gunakan atribut kunci untuk menentukan tindakan apa yang diizinkan pada objek utama, termasuk kunci publik, pribadi, atau rahasia. Atribut dan nilai kunci didefinisikan selama operasi pembuatan objek utama.

Ekstensi Kriptografi Java (JCE) tidak menentukan bagaimana Anda harus menetapkan nilai pada atribut kunci, sehingga sebagian besar tindakan diizinkan secara default. Sebaliknya, PKCS # 11 standar menentukan satu set lengkap atribut dengan default lebih ketat. Dimulai dengan penyedia JCE 3.1, AWS CloudHSM menyediakan ekstensi eksklusif yang memungkinkan Anda menetapkan nilai yang lebih ketat untuk atribut yang umum digunakan.

Atribut yang didukung

Anda dapat mengatur nilai untuk atribut yang tercantum dalam tabel berikut. Sebagai praktik terbaik, tetapkan hanya nilai untuk atribut yang ingin Anda buat ketat. Jika Anda tidak menentukan nilai, AWS CloudHSM menggunakan nilai default yang ditentukan dalam tabel di bawah ini. Sel kosong di kolom Nilai Default menunjukkan bahwa tidak ada nilai default tertentu yang ditetapkan untuk atribut.

Atribut	Nilai Default			Catatan
	Kunci simetris	Kunci Publik dalam Pasangan Kunci	Kunci Privat dalam Pasangan Kunci	
DECRYPT	TRUE		TRUE	Benar menunjukkan

Atribut	Nilai Default			Catatan
	Kunci simetris	Kunci Publik dalam Pasangan Kunci	Kunci Privat dalam Pasangan Kunci	
				Anda dapat menggunakan kunci untuk mendekripsi penyangga apa pun. Anda biasanya menyetel ini ke FALSE untuk kunci yang WRAPNYA disetel ke true.
DERIVE				Memungkinkan kunci untuk digunakan untuk mendapatkan kunci lainnya.
ENCRYPT	TRUE	TRUE		Benar menunjukkan Anda dapat menggunakan kunci untuk mengenkripsi penyangga apa pun.

Atribut	Nilai Default			Catatan
	Kunci simetris	Kunci Publik dalam Pasangan Kunci	Kunci Privat dalam Pasangan Kunci	
EXTRACTABLE	TRUE		TRUE	Benar menunjukkan Anda dapat mengekspor kunci ini keluar dari HSM.
ID				Nilai yang ditentukan pengguna yang digunakan untuk mengidentifikasi kunci.
KEY_TYPE				Digunakan untuk mengidentifikasi jenis kunci (AES,, rahasia generik DESede, EC, atau RSA).

Atribut	Nilai Default			Catatan
	Kunci simetris	Kunci Publik dalam Pasangan Kunci	Kunci Privat dalam Pasangan Kunci	
LABEL				String yang ditentukan pengguna memungkinkan Anda mengidentifikasi kunci dengan mudah pada HSM Anda. Untuk mengikuti praktik terbaik, gunakan label unik untuk setiap tombol sehingga lebih mudah ditemukan nanti.
LOCAL				Menunjukkan kunci yang dihasilkan oleh HSM.
OBJECT_CLASS				Digunakan untuk mengidentifikasi Object Class dari kunci (SecretKey, PublicKey atau PrivateKey).

Atribut	Nilai Default			Catatan
	Kunci simetris	Kunci Publik dalam Pasangan Kunci	Kunci Privat dalam Pasangan Kunci	
PRIVATE	TRUE	TRUE	TRUE	Benar menunjukkan bahwa pengguna tidak dapat mengakses kunci sampai pengguna diautentikasi. Untuk kejelasan, pengguna tidak dapat mengakses kunci apa pun AWS CloudHSM sampai mereka diautentikasi, bahkan jika atribut ini disetel ke FALSE.

Atribut	Nilai Default			Catatan
	Kunci simetris	Kunci Publik dalam Pasangan Kunci	Kunci Privat dalam Pasangan Kunci	
SIGN	TRUE		TRUE	Benar menunjukkan Anda dapat menggunakan kunci untuk menandatangani digest pesan. Hal ini umumnya diatur ke FALSE untuk kunci publik dan kunci privat yang telah Anda arsipkan.
SIZE				Atribut yang mendefinisikan ukuran kunci. Untuk detail selengkapnya tentang ukuran kunci yang didukung, lihat Mekanisme yang didukung untuk SDK Klien 5 .

Atribut	Nilai Default			Catatan
	Kunci simetris	Kunci Publik dalam Pasangan Kunci	Kunci Privat dalam Pasangan Kunci	
TOKEN	FALSE	FALSE	FALSE	Kunci permanen yang direplika si HSMs di semua klaster dan disertakan dalam cadangan. TOKEN = FALSE menyiratkan kunci sementara yang secara otomatis dihapus ketika koneksi ke HSM rusak atau keluar.
UNWRAP	TRUE		TRUE	Benar menunjukkan Anda dapat menggunakan kunci untuk membungkus kunci lain.

Atribut	Nilai Default			Catatan
	Kunci simetris	Kunci Publik dalam Pasangan Kunci	Kunci Privat dalam Pasangan Kunci	
VERIFY	TRUE	TRUE		Benar menunjukkan Anda dapat menggunakan kunci untuk menandatangani digest pesan. Hal ini umumnya diatur ke FALSE untuk kunci privat.
WRAP	TRUE	TRUE		Benar menunjukkan Anda dapat menggunakan kunci untuk membungkus kunci lain. Anda biasanya akan mengatur ini ke FALSE untuk kunci privat.

Atribut	Nilai Default			Catatan
	Kunci simetris	Kunci Publik dalam Pasangan Kunci	Kunci Privat dalam Pasangan Kunci	
WRAP_WITH_TRUSTED	FALSE		FALSE	True menunjukkan an kunci hanya dapat dibungkus dan dibuka dengan kunci yang memiliki TRUSTED atribut disetel ke true. Setelah kunci WRAP_WITH_TRUSTED disetel ke true, atribut itu hanya-baca dan tidak dapat disetel ke false. Untuk membaca tentang pembungkus kepercayaan, lihat Menggunakan kunci tepercaya untuk mengontrol pembukaan kunci .

Note

Anda mendapatkan dukungan yang lebih luas untuk atribut di pustaka PKCS #11. Untuk informasi lebih lanjut, lihat [Atribut PKCS #11 yang Didukung](#).

Menyetel atribut untuk kunci

`KeyAttributesMap` adalah objek seperti Peta Java, yang dapat Anda gunakan untuk mengatur nilai atribut untuk objek kunci. Metode untuk fungsi `KeyAttributesMap` yang mirip dengan metode yang digunakan untuk manipulasi peta Java.

Untuk mengatur nilai kustom pada atribut, Anda memiliki dua opsi:

- Gunakan metode yang tercantum dalam tabel berikut
- Gunakan pola pembangun yang ditunjukkan kemudian dalam dokumen ini

Atribut peta objek mendukung metode berikut untuk mengatur atribut:

Operasi	Nilai Pengembalian	Metode KeyAttributesMap
Dapatkan nilai atribut kunci untuk kunci yang ada	Obyek (berisi nilai) atau nol	<code>get(keyAttribute)</code>
Isi nilai satu atribut kunci	Nilai sebelumnya terkait dengan atribut kunci, atau nol jika tidak ada pemetaan untuk atribut kunci	<code>put(keyAttribute, value)</code>
Isi nilai untuk beberapa atribut kunci	N/A	<code>putAll () keyAttributesMap</code>
Hapus pasangan nilai kunci dari peta atribut	Nilai sebelumnya terkait dengan atribut kunci, atau nol jika tidak ada pemetaan untuk atribut kunci	<code>remove(keyAttribute)</code>

 Note

Setiap atribut yang tidak secara eksplisit Anda tentukan diatur ke default yang tercantum dalam tabel sebelumnya di [the section called “Atribut yang didukung”](#).

Menetapkan atribut untuk key pair

Gunakan kelas Java `KeyPairAttributesMap` untuk menangani atribut kunci untuk pasangan kunci. `KeyPairAttributesMap` merangkum dua objek `KeyAttributesMap`; satu untuk kunci publik dan satu untuk kunci privat.

Untuk mengatur atribut individu untuk kunci publik dan kunci privat secara terpisah, Anda dapat menggunakan metode `put()` pada objek peta `KeyAttributes` yang terkait untuk kunci itu. Gunakan metode `getPublic()` untuk mengambil peta atribut untuk kunci publik, dan gunakan `getPrivate()` untuk mengambil peta atribut untuk kunci privat. Isi nilai dari beberapa atribut kunci bersama-sama untuk kedua pasangan kunci publik dan privat menggunakan `putAll()` dengan peta atribut pasangan kunci sebagai argumen.

Contoh kode untuk pustaka AWS CloudHSM perangkat lunak untuk Java untuk Klien SDK 5

Topik ini menyediakan sumber daya dan informasi tentang contoh kode Java untuk AWS CloudHSM Client SDK 5.

Prasyarat

Sebelum menjalankan sampel, Anda harus mengatur lingkungan Anda:

- Instal dan konfigurasi penyedia [Java Cryptographic Extension \(JCE\)](#).
- Siapkan [nama pengguna HSM dan kata sandi](#) yang valid. Izin pengguna kriptografi (CU) cukup untuk tugas-tugas ini. Aplikasi Anda menggunakan kredensial ini untuk login ke HSM di setiap contoh.
- Tentukan cara memberikan kredensial ke [penyedia JCE](#).

Sampel Kode

Sampel kode berikut menunjukkan cara menggunakan [penyedia JCE AWS CloudHSM](#) untuk melakukan tugas dasar. Lebih banyak contoh kode tersedia di [GitHub](#).

- [Masuk ke HSM](#)
- [Kelola kunci](#)
- [Hasilkan Kunci Simetris](#)
- [Hasilkan Kunci Asimetris](#)
- [Enkripsi dan dekripsi dengan AES-GCM](#)
- [Enkripsi dan dekripsi dengan AES-CTR](#)
- [Enkripsi dan dekripsi dengan DESede](#) -ECB lihat catatan [1](#)
- [Masuk dan Verifikasi dengan Kunci RSA](#)
- [Masuk dan Verifikasi dengan Kunci EC](#)
- [Gunakan atribut kunci yang didukung](#)
- [Gunakan toko kunci CloudHSM](#)

[1] Sesuai dengan panduan NIST, ini tidak diizinkan untuk cluster dalam mode FIPS setelah 2023. Untuk cluster dalam mode non-FIPS, masih diperbolehkan setelah 2023. Lihat [Kepatuhan FIPS 140: Penutupan Mekanisme 2024](#) untuk rincian selengkapnya.

AWS CloudHSM JCE penyedia Javadocs

Gunakan penyedia JCE Javadocs untuk mendapatkan informasi penggunaan tentang jenis dan metode Java yang ditentukan dalam AWS CloudHSM JCE SDK. Untuk mengunduh Javadocs terbaru AWS CloudHSM, lihat [AWS CloudHSM rilis SDK Klien terbaru](#) bagian di halaman Unduhan.

Anda dapat mengimpor Javadocs ke lingkungan pengembangan terintegrasi (IDE) atau melihatnya di browser web.

AWS CloudHSM KeyStore Kelas Java untuk Client SDK 5

AWS CloudHSM KeyStoreKelas ini menyediakan toko PKCS12 kunci tujuan khusus. Penyimpanan kunci ini dapat menyimpan sertifikat bersama dengan data kunci Anda dan menghubungkannya dengan data kunci yang disimpan di AWS CloudHSM. AWS CloudHSM KeyStoreKelas mengimplementasikan KeyStore Service Provider Interface (SPI) dari Java Cryptography Extension (JCE). Untuk informasi selengkapnya tentang penggunaanKeyStore, lihat [Kelas KeyStore](#).

 Note

Karena sertifikat adalah informasi publik, dan untuk memaksimalkan kapasitas penyimpanan untuk kunci kriptografi, AWS CloudHSM tidak mendukung penyimpanan sertifikat. HSMs

Pilih penyimpanan kunci yang sesuai untuk AWS CloudHSM Client SDK 5

Penyedia AWS CloudHSM Java Cryptographic Extension (JCE) menawarkan AWS CloudHSM tujuan khusus. KeyStore AWS CloudHSM KeyStoreKelas ini mendukung pembongkaran operasi kunci ke HSM, penyimpanan lokal sertifikat dan operasi berbasis sertifikat.

Muat CloudHSM tujuan khusus sebagai berikut: KeyStore

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

Inisialisasi SDK AWS CloudHSM KeyStore Klien 5

Masuk ke cara AWS CloudHSM KeyStore yang sama seperti Anda masuk ke penyedia JCE. Anda dapat menggunakan variabel lingkungan atau file properti sistem, dan Anda harus masuk sebelum mulai menggunakan CloudHSM KeyStore. Untuk contoh masuk ke HSM menggunakan penyedia JCE, lihat [Masuk ke HSM](#).

Jika diinginkan, Anda dapat menentukan kata sandi untuk mengenkripsi PKCS12 file lokal yang menyimpan data penyimpanan kunci. Saat Anda membuat AWS CloudHSM Keystore, Anda mengatur kata sandi dan memberikannya saat menggunakan metode pemuatan, atur, dan dapatkan.

Buat instance objek CloudHSM baru sebagai berikut: KeyStore

```
ks.load(null, null);
```

Tulis data penyimpanan kunci ke file menggunakan metode store. Sejak saat itu, Anda dapat memuat penyimpanan kunci yang ada menggunakan metode load dengan file sumber dan kata sandi sebagai berikut:

```
ks.load(inputStream, password);
```

Gunakan AWS CloudHSM KeyStore atau AWS CloudHSM Klien SDK 5

AWS CloudHSM KeyStore sesuai dengan KeyStore spesifikasi JCE [Class](#) dan menyediakan fungsi-fungsi berikut.

- `load`

Memuat penyimpanan kunci dari pengaliran input yang diberikan. Jika kata sandi ditetapkan saat menyimpan penyimpanan kunci, kata sandi yang sama ini harus disediakan agar pemuatan berhasil. Atur kedua parameter ke null untuk menginisialisasi sebuah penyimpanan kunci kosong baru.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
ks.load(inputStream, password);
```

- `aliases`

Mengembalikan penghitungan nama alias dari semua entri dalam contoh instans penyimpanan kunci yang diberikan. Hasil termasuk objek yang disimpan secara lokal dalam PKCS12 file dan objek yang tinggal di HSM.

Contoh kode:

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
for(Enumeration<String> entry = ks.aliases(); entry.hasMoreElements();) {
    String label = entry.nextElement();
    System.out.println(label);
}
```

- `containsalias`

Mengembalikan nilai true jika penyimpanan kunci memiliki akses ke setidaknya satu objek dengan alias yang ditentukan. Toko kunci memeriksa objek yang disimpan secara lokal dalam PKCS12 file dan objek yang tinggal di HSM.

- `deleteEntry`

Menghapus entri sertifikat dari PKCS12 file lokal. Menghapus data kunci yang disimpan dalam HSM tidak didukung menggunakan file. AWS CloudHSM KeyStore Anda dapat menghapus kunci menggunakan `destroy` metode antarmuka [Destroyable](#).

```
((Destroyable) key).destroy();
```

- **getCertificate**

Mengembalikan sertifikat yang terkait dengan alias jika tersedia. Jika alias tidak ada atau mereferensikan objek yang bukan sertifikat, fungsi mengembalikan NULL.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");  
Certificate cert = ks.getCertificate(alias);
```

- **getCertificateAlias**

Mengembalikan nama (alias) dari entri penyimpanan kunci pertama yang datanya cocok dengan sertifikat yang diberikan.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");  
String alias = ks.getCertificateAlias(cert);
```

- **getCertificateChain**

Mengembalikan rantai sertifikat yang terkait dengan alias yang diberikan. Jika alias tidak ada atau mereferensikan objek yang bukan sertifikat, fungsi mengembalikan NULL.

- **getCreationDate**

Mengembalikan tanggal pembuatan entri yang diidentifikasi oleh alias yang diberikan. Jika tanggal pembuatan tidak tersedia, fungsi mengembalikan tanggal saat sertifikat menjadi valid.

- **getKey**

GetKey diteruskan ke HSM dan mengembalikan objek kunci yang sesuai dengan label yang diberikan. Seperti yang getKey secara langsung menanyakan HSM, itu dapat digunakan untuk kunci apa pun pada HSM terlepas dari apakah itu dihasilkan oleh. KeyStore

```
Key key = ks.getKey(keyLabel, null);
```

- **isCertificateEntry**

Memeriksa apakah entri dengan alias yang diberikan merupakan entri sertifikat.

- **isKeyEntry**

Memeriksa apakah entri dengan alias yang diberikan merupakan entri kunci. Tindakan mencari PKCS12 file dan HSM untuk alias.

- `setCertificateEntry`

Menetapkan sertifikat yang diberikan untuk alias yang diberikan. Jika alias yang diberikan sudah digunakan untuk mengidentifikasi kunci atau sertifikat, `KeyStoreException` dikeluarkan. Anda dapat menggunakan kode JCE untuk mendapatkan objek kunci dan kemudian menggunakan `KeyStore SetKeyEntry` metode untuk mengaitkan sertifikat ke kunci.

- `setKeyEntry` dengan `byte[]` kunci

API ini saat ini tidak didukung dengan Client SDK 5.

- `setKeyEntry` dengan objek `Key`

Menetapkan kunci yang diberikan untuk alias yang diberikan dan menyimpannya di dalam HSM. Jika kunci belum ada di dalam HSM, itu akan diimpor ke HSM sebagai kunci sesi yang dapat diekstraksi.

Jika objek `Key` adalah tipe `PrivateKey`, objek harus disertai dengan rantai sertifikat yang sesuai.

Jika alias sudah ada, panggilan `SetKeyEntry` mengeluarkan `KeyStoreException` dan mencegah kunci ditimpa. Jika kunci harus ditimpa, gunakan KMU atau JCE untuk tujuan itu.

- `engineSize`

Mengembalikan jumlah entri dalam penyimpanan kunci.

- `store`

Menyimpan penyimpanan kunci ke aliran output yang diberikan sebagai PKCS12 file dan mengamankannya dengan kata sandi yang diberikan. Selain itu, semua kunci dimuat tetap ada (yang ditetapkan menggunakan panggilan `setKey`).

Konfigurasi lanjutan untuk AWS CloudHSM JCE untuk Client SDK 5

Penyedia AWS CloudHSM JCE menyertakan konfigurasi lanjutan berikut, yang bukan merupakan bagian dari konfigurasi umum yang digunakan sebagian besar pelanggan.

- [Menghubungkan ke beberapa cluster](#)

- [Ekstraksi kunci menggunakan JCE](#)
- [Coba lagi konfigurasi untuk JCE](#)

Menghubungkan ke beberapa AWS CloudHSM cluster dengan penyedia JCE

Konfigurasi ini memungkinkan instance klien tunggal untuk berkomunikasi ke beberapa AWS CloudHSM cluster. Dibandingkan dengan memiliki satu instance yang hanya berkomunikasi dengan satu cluster, ini bisa menjadi fitur penghematan biaya untuk beberapa kasus penggunaan. `CloudHsmProviderKelas` ini adalah AWS CloudHSM implementasi dari [kelas Provider Java Security](#). Setiap instance dari kelas ini mewakili koneksi ke seluruh AWS CloudHSM cluster Anda. Anda membuat instance kelas ini dan menambahkannya ke daftar penyedia Java Security sehingga Anda dapat berinteraksi dengannya menggunakan kelas JCE standar.

Contoh berikut membuat instance class ini dan menambahkannya ke daftar penyedia Java Security:

```
if (Security.getProvider(CloudHsmProvider.PROVIDER_NAME) == null) {  
    Security.addProvider(new CloudHsmProvider());  
}
```

`CloudHsmProvider` dapat dikonfigurasi dengan dua cara:

1. Konfigurasi dengan file (konfigurasi default)
2. Konfigurasi menggunakan kode

Topik berikut menjelaskan konfigurasi ini, dan cara menghubungkan ke beberapa cluster.

Topik

- [Konfigurasi AWS CloudHSM `CloudHsmProvider` kelas dengan file \(Konfigurasi default\)](#)
- [Konfigurasi AWS CloudHSM `CloudHsmProvider` kelas menggunakan kode](#)
- [Connect ke beberapa AWS CloudHSM cluster](#)

Konfigurasi AWS CloudHSM **`CloudHsmProvider`** kelas dengan file (Konfigurasi default)

Cara default untuk mengkonfigurasi AWS CloudHSM `CloudHsmProvider` kelas adalah dengan file.

Ketika Anda membuat instance `CloudHsmProvider` menggunakan konstruktor default, secara default akan mencari file konfigurasi di `/opt/cloudhsm/etc/cloudhsm-jce.cfg` jalur di Linux. File konfigurasi ini dapat dikonfigurasi menggunakan `fileconfigure-jce`.

Objek yang dibuat menggunakan konstruktor default akan menggunakan nama penyedia CloudHSM default. CloudHSM Nama penyedia berguna untuk berinteraksi dengan JCE untuk memberi tahu penyedia mana yang akan digunakan untuk berbagai operasi. Contoh untuk menggunakan nama penyedia CloudHSM untuk operasi Cipher adalah sebagai berikut:

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "CloudHSM");
```

Konfigurasikan AWS CloudHSM **`CloudHsmProvider`** kelas menggunakan kode

Pada Client SDK versi 5.8.0, Anda juga dapat mengkonfigurasi AWS CloudHSM `CloudHsmProvider` kelas menggunakan kode Java. Cara untuk melakukan ini adalah dengan menggunakan objek `CloudHsmProviderConfig` kelas. Anda dapat membangun objek ini menggunakan `CloudHsmProviderConfigBuilder`.

`CloudHsmProvider` memiliki konstruktor lain yang mengambil `CloudHsmProviderConfig` objek, seperti contoh berikut menunjukkan.

Example

```
CloudHsmProviderConfig config = CloudHsmProviderConfig.builder()
    .withCluster(
        CloudHsmCluster.builder()
            .withHsmCAFilePath(hsmCAFilePath)

        .withClusterUniqueIdentifier("CloudHsmCluster1")
        .withServer(CloudHsmServer.builder().withHostIP(hostName).build())
        .build()
    ).build();
CloudHsmProvider provider = new CloudHsmProvider(config);
```

Dalam contoh ini, nama penyedia JCE adalah `CloudHsmCluster1`. Ini adalah nama yang kemudian dapat digunakan aplikasi untuk berinteraksi dengan JCE:

Example

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "CloudHsmCluster1");
```

Atau, aplikasi juga dapat menggunakan objek penyedia yang dibuat di atas untuk memberi tahu JCE untuk menggunakan penyedia itu untuk operasi:

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider);
```

Jika pengidentifikasi unik tidak ditentukan dengan `withClusterUniqueIdentifier` metode, nama penyedia yang dibuat secara acak dibuat untuk Anda. Untuk mendapatkan pengenalan yang dihasilkan secara acak ini, aplikasi dapat memanggil `provider.getName()` untuk mendapatkan pengenalan.

Connect ke beberapa AWS CloudHSM cluster

Masing-masing `CloudHsmProvider` mewakili koneksi ke AWS CloudHSM Cluster Anda. Jika Anda ingin berbicara dengan cluster lain dari aplikasi yang sama, Anda dapat membuat objek lain `CloudHsmProvider` dengan konfigurasi untuk cluster Anda yang lain dan Anda dapat berinteraksi dengan cluster lain ini baik menggunakan objek penyedia atau menggunakan nama penyedia, seperti yang ditunjukkan pada contoh berikut.

Example

```
CloudHsmProviderConfig config = CloudHsmProviderConfig.builder()
    .withCluster(
        CloudHsmCluster.builder()
            .withHsmCAFilePath(hsmCAFilePath)

        .withClusterUniqueIdentifier("CloudHsmCluster1")
        .withServer(CloudHsmServer.builder().withHostIP(hostName).build())
        .build()
    ).build();
CloudHsmProvider provider1 = new CloudHsmProvider(config);

if (Security.getProvider(provider1.getName()) == null) {
    Security.addProvider(provider1);
}

CloudHsmProviderConfig config2 = CloudHsmProviderConfig.builder()
    .withCluster(
        CloudHsmCluster.builder()
            .withHsmCAFilePath(hsmCAFilePath2)

        .withClusterUniqueIdentifier("CloudHsmCluster2")
```

```
        .withServer(CloudHsmServer.builder().withHostIP(hostName2).build())
            .build())
        .build();
CloudHsmProvider provider2 = new CloudHsmProvider(config2);

if (Security.getProvider(provider2.getName()) == null) {
    Security.addProvider(provider2);
}
```

Setelah Anda mengonfigurasi kedua penyedia (kedua cluster) di atas, Anda dapat berinteraksi dengan mereka baik menggunakan objek penyedia atau menggunakan nama penyedia.

Memperluas contoh ini yang menunjukkan cara berbicara `cluster1`, Anda dapat menggunakan sampel berikut untuk AES/GCM/NoPadding operasi:

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider1);
```

Dan dalam aplikasi yang sama untuk melakukan “AES” Pembuatan kunci pada cluster kedua menggunakan nama penyedia, Anda juga dapat menggunakan contoh berikut:

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider2.getName());
```

Ekstraksi kunci menggunakan JCE untuk AWS CloudHSM

Java Cryptography Extension (JCE) menggunakan arsitektur yang memungkinkan implementasi kriptografi yang berbeda untuk dicolokkan. AWS CloudHSM mengirimkan salah satu penyedia JCE yang menurunkan operasi kriptografi ke HSM. Agar sebagian besar penyedia JCE lainnya dapat bekerja dengan kunci yang disimpan di AWS CloudHSM, mereka harus mengekstrak byte kunci HSMs dari teks bening Anda ke dalam memori mesin Anda untuk digunakan. HSMs biasanya hanya mengizinkan kunci untuk diekstraksi sebagai objek yang dibungkus, bukan teks yang jelas. Namun, untuk mendukung kasus penggunaan integrasi antar-penyedia, AWS CloudHSM memungkinkan opsi konfigurasi opt-in untuk mengaktifkan ekstraksi byte kunci secara jelas.

Important

JCE membongkar operasi ke AWS CloudHSM setiap kali penyedia AWS CloudHSM ditentukan atau AWS CloudHSM objek kunci digunakan. Anda tidak perlu mengekstrak kunci dengan jelas jika Anda mengharapkan operasi Anda terjadi di dalam HSM. Ekstraksi kunci dalam teks yang jelas hanya diperlukan ketika aplikasi Anda tidak dapat menggunakan

mekanisme aman seperti membungkus dan membuka kunci karena pembatasan dari perpustakaan pihak ketiga atau penyedia JCE.

Penyedia AWS CloudHSM JCE memungkinkan ekstraksi kunci publik untuk bekerja dengan penyedia JCE eksternal secara default. Metode berikut selalu diperbolehkan:

Kelas	Metode	Format (GetEncoded)
EcPublicKey	getEncoded ()	X.509
	GetW ()	N/A
RSAPublicKunci	getEncoded ()	X.509
	getPublicExponent()	N/A
CloudHsmRsaPrivateCrtKey	getPublicExponent()	N/A

Penyedia AWS CloudHSM JCE tidak mengizinkan ekstraksi byte kunci secara jelas untuk kunci pribadi atau rahasia secara default. Jika kasus penggunaan Anda memerlukannya, Anda dapat mengaktifkan ekstraksi byte kunci secara jelas untuk kunci pribadi atau rahasia dalam kondisi berikut:

1. `EXTRACTABLE` atribut untuk kunci pribadi dan rahasia diatur ke `true`.
 - Secara default, `EXTRACTABLE` atribut untuk kunci pribadi dan rahasia diatur ke `true`. `EXTRACTABLE` kunci adalah kunci yang diizinkan untuk diekspor keluar dari HSM. Untuk informasi selengkapnya, lihat Atribut Java yang didukung untuk [SDK Klien 5](#).
2. `WRAP_WITH_TRUSTED` atribut untuk kunci pribadi dan rahasia diatur ke `false`.
 - `getEncoded`, `getPrivateExponent`, dan `getS` tidak dapat digunakan dengan kunci pribadi yang tidak dapat diekspor dengan jelas. `WRAP_WITH_TRUSTED` tidak mengizinkan kunci pribadi Anda untuk diekspor keluar dari HSM dengan jelas. Untuk informasi selengkapnya, lihat [Menggunakan kunci terpercaya untuk mengontrol pembukaan kunci](#).

Izinkan penyedia JCE untuk mengekstrak rahasia kunci pribadi dari AWS CloudHSM

Gunakan langkah-langkah berikut untuk memungkinkan penyedia AWS CloudHSM JCE mengekstrak rahasia kunci pribadi Anda.

⚠ Important

Perubahan konfigurasi ini memungkinkan ekstraksi semua byte EXTRACTABLE kunci secara jelas dari cluster HSM Anda. Untuk keamanan yang lebih baik, Anda harus mempertimbangkan untuk menggunakan [metode pembungkus kunci](#) untuk mengekstrak kunci dari HSM dengan aman. Ini mencegah ekstraksi byte kunci Anda yang tidak disengaja dari HSM.

1. Gunakan perintah berikut untuk mengaktifkan kunci pribadi atau rahasia Anda untuk diekstraksi di JCE:

Linux

```
$ /opt/cloudhsm/bin/configure-jce --enable-clear-key-extraction-in-software
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --enable-clear-key-extraction-in-software
```

2. Setelah Anda mengaktifkan ekstraksi kunci yang jelas, metode berikut diaktifkan untuk mengekstrak kunci pribadi ke dalam memori.

Kelas	Metode	Format (GetEncoded)
Key	getEncoded ()	RAW
ECPrivateKey	getEncoded ()	PKC #8
	Gets ()	N/A
RSAPrivateCrtKey	getEncoded ()	X.509
	getPrivateExponent()	N/A
	getPrimep ()	N/A

Kelas	Metode	Format (GetEncoded)
	getPrimeQ ()	N/A
	getPrimeExponentP ()	N/A
	getPrimeExponentQ ()	N/A
	getCrtCoefficient()	N/A

Jika Anda ingin mengembalikan perilaku default dan tidak mengizinkan JCE untuk mengekspor kunci dengan jelas, jalankan perintah berikut:

Linux

```
$ /opt/cloudhsm/bin/configure-jce --disable-clear-key-extraction-in-software
```

Windows

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-clear-key-extraction-in-software
```

Coba lagi perintah untuk JCE untuk AWS CloudHSM

AWS CloudHSM Client SDK 5.8.0 dan yang lebih baru memiliki strategi coba ulang otomatis bawaan yang akan mencoba kembali operasi HSM-throttled dari sisi klien. Ketika HSM membatasi operasi karena terlalu sibuk melakukan operasi sebelumnya dan tidak dapat menerima lebih banyak permintaan, klien SDKs akan mencoba kembali operasi yang dibatasi hingga 3 kali sementara mundur secara eksponensial. Strategi coba ulang otomatis ini dapat diatur ke salah satu dari dua mode: off dan standar.

- off: SDK Klien tidak akan melakukan strategi coba ulang apa pun untuk operasi yang dibatasi oleh HSM.
- standard: Ini adalah mode default untuk Client SDK 5.8.0 dan yang lebih baru. Dalam mode ini, klien SDKs akan secara otomatis mencoba kembali operasi yang dibatasi dengan mundur secara eksponensial.

Untuk informasi selengkapnya, lihat [Pelambatan HSM](#).

Setel perintah coba lagi ke mode mati

Linux

Untuk mengatur perintah coba lagi off untuk Client SDK 5 di Linux

- Anda dapat menggunakan perintah berikut untuk mengatur konfigurasi coba lagi ke off mode:

```
$ sudo /opt/cloudhsm/bin/configure-jce --default-retry-mode off
```

Windows

Untuk mengatur perintah coba lagi off untuk Client SDK 5 pada Windows

- Anda dapat menggunakan perintah berikut untuk mengatur konfigurasi coba lagi ke off mode:

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --default-retry-mode off
```

Menggunakan versi SDK sebelumnya untuk bekerja dengan AWS CloudHSM

 SDK versi 5.8.0 dan sebelumnya telah mencapai akhir dukungan mereka. Setelah 31 Maret 2025 dokumentasi untuk SDK versi 3.4.4 dan sebelumnya tidak akan lagi tersedia.

AWS CloudHSM mencakup dua versi SDK Klien utama:

- SDK Klien 5: Ini adalah SDK Klien terbaru dan default kami. Untuk informasi tentang manfaat dan keuntungan yang diberikannya, lihat [Manfaat SDK AWS CloudHSM Klien 5](#).
- SDK Klien 3: Ini adalah SDK Klien kami yang lebih lama. Ini mencakup satu set lengkap komponen untuk platform dan aplikasi berbasis bahasa kompatibilitas dan alat manajemen.

Untuk petunjuk tentang migrasi dari SDK Klien 3 ke SDK Klien 5, lihat [Migrasi dari SDK AWS CloudHSM Klien 3 ke SDK Klien 5](#)

Topik ini menjelaskan SDK Klien 3. Untuk melihat versi Client SDK yang Anda gunakan, lihat [Periksa versi SDK AWS CloudHSM Klien Anda](#).

Untuk mengunduh, lihat [Unduh](#).

Topik

- [Tingkatkan SDK AWS CloudHSM Klien 3 di Linux](#)
- [AWS CloudHSM Platform yang didukung SDK Klien 3](#)
- [Pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 3](#)
- [OpenSSL Dynamic Engine AWS CloudHSM untuk Klien SDK 3](#)
- [Penyedia JCE untuk AWS CloudHSM Klien SDK 3](#)
- [API Kriptografi: Generasi Berikutnya \(CNG\) dan penyedia penyimpanan kunci \(KSP\) untuk AWS CloudHSM](#)

Tingkatkan SDK AWS CloudHSM Klien 3 di Linux

 SDK versi 5.8.0 dan sebelumnya telah mencapai akhir dukungan mereka. Setelah 31 Maret 2025 dokumentasi untuk SDK versi 3.4.4 dan sebelumnya tidak akan lagi tersedia.

Dengan AWS CloudHSM Client SDK 3.1 dan yang lebih tinggi, versi daemon klien dan komponen apa pun yang Anda instal harus cocok untuk ditingkatkan. Untuk semua sistem berbasis Linux, Anda harus menggunakan satu perintah untuk mengelompokkan meningkatkan daemon klien dengan versi yang sama dari pustaka PKCS #11, penyedia Java Cryptographic Extension (JCE), atau OpenSSL Dynamic Engine. Persyaratan ini tidak berlaku untuk sistem berbasis Windows karena biner untuk penyedia CNG dan KSP sudah termasuk dalam paket daemon klien.

Untuk memeriksa versi daemon klien

- Pada sistem Linux berbasis Red Hat (termasuk Amazon Linux dan CentOS), gunakan perintah berikut:

```
rpm -qa | grep ^cloudhsm
```

- Pada sistem Linux berbasis Debian, gunakan perintah berikut:

```
apt list --installed | grep ^cloudhsm
```

- Pada sistem Windows, gunakan perintah berikut:

```
wmic product get name,version
```

Topik

- [Prasyarat](#)
- [Langkah 1: Hentikan daemon klien](#)
- [Langkah 2: Tingkatkan SDK klien](#)
- [Langkah 3: Mulai daemon klien](#)

Prasyarat

Unduh versi terbaru daemon AWS CloudHSM klien dan pilih komponen Anda.

Note

Anda tidak perlu menginstal semua komponen. Untuk setiap komponen yang telah Anda instal, Anda harus meningkatkan komponen tersebut untuk mencocokkan versi daemon klien.

Daemon klien Linux terbaru

Amazon Linux

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

RHEL 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

RHEL 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client_latest_u18.04_amd64.deb
```

Pustaka PKCS #11 terbaru

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-pkcs11-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

CentOS 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

RHEL 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-pkcs11_latest_amd64.deb
```


Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client-pkcs11_latest_u18.04_amd64.deb
```

OpenSSL Dynamic Engine Terbaru

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-dyn_latest_amd64.deb
```

Penyedia JCE terbaru

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-jce-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

CentOS 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-jce-latest.el8.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

RHEL 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-jce-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-jce_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client-jce_latest_u18.04_amd64.deb
```

Langkah 1: Hentikan daemon klien

Gunakan perintah berikut untuk menghentikan daemon klien.

Amazon Linux

```
$ sudo stop cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client stop
```

CentOS 7

```
$ sudo service cloudhsm-client stop
```

CentOS 8

```
$ sudo service cloudhsm-client stop
```

RHEL 7

```
$ sudo service cloudhsm-client stop
```

RHEL 8

```
$ sudo service cloudhsm-client stop
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client stop
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client stop
```

Langkah 2: Tingkatkan SDK klien

Perintah berikut menunjukkan sintaks yang diperlukan untuk meningkatkan daemon klien dan komponen. Sebelum menjalankan perintah, hapus komponen yang tidak ingin ditingkatkan.

Amazon Linux

```
$ sudo yum install ./cloudhsm-client-latest.el6.x86_64.rpm \  
    <./cloudhsm-client-pkcs11-latest.el6.x86_64.rpm> \  
    <./cloudhsm-client-dyn-latest.el6.x86_64.rpm> \  
    <./cloudhsm-client-jce-latest.el6.x86_64.rpm>
```

Amazon Linux 2

```
$ sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm \  
    <./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm> \  
    <./cloudhsm-client-dyn-latest.el7.x86_64.rpm> \  
    <./cloudhsm-client-jce-latest.el7.x86_64.rpm>
```

CentOS 7

```
$ sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm \  
    <./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm> \  
    <./cloudhsm-client-dyn-latest.el7.x86_64.rpm> \  
    <./cloudhsm-client-jce-latest.el7.x86_64.rpm>
```

CentOS 8

```
$ sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm \  
    <./cloudhsm-client-pkcs11-latest.el8.x86_64.rpm> \  
    <./cloudhsm-client-jce-latest.el8.x86_64.rpm>
```

RHEL 7

```
$ sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm \  
    <./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm> \  
    <./cloudhsm-client-dyn-latest.el7.x86_64.rpm> \  
    <./cloudhsm-client-jce-latest.el7.x86_64.rpm>
```

RHEL 8

```
$ sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm \  
    <./cloudhsm-client-pkcs11-latest.el8.x86_64.rpm> \  
    <./cloudhsm-client-jce-latest.el8.x86_64.rpm>
```

Ubuntu 16.04 LTS

```
$ sudo apt install ./cloudhsm-client_latest_amd64.deb \  
                  <cloudhsm-client-pkcs11_latest_amd64.deb> \  
                  <cloudhsm-client-dyn_latest_amd64.deb> \  
                  <cloudhsm-client-jce_latest_amd64.deb>
```

Ubuntu 18.04 LTS

```
$ sudo apt install ./cloudhsm-client_latest_u18.04_amd64.deb \  
                  <cloudhsm-client-pkcs11_latest_amd64.deb> \  
                  <cloudhsm-client-jce_latest_amd64.deb>
```

Langkah 3: Mulai daemon klien

Gunakan perintah berikut untuk memulai daemon klien.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo systemctl start cloudhsm-client
```

CentOS 7

```
$ sudo systemctl start cloudhsm-client
```

CentOS 8

```
$ sudo systemctl start cloudhsm-client
```

RHEL 7

```
$ sudo systemctl start cloudhsm-client
```

RHEL 8

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 16.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 20.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

AWS CloudHSM Platform yang didukung SDK Klien 3

 SDK versi 5.8.0 dan sebelumnya telah mencapai akhir dukungan mereka. Setelah 31 Maret 2025 dokumentasi untuk SDK versi 3.4.4 dan sebelumnya tidak akan lagi tersedia.

AWS CloudHSM Client SDK 3 memerlukan daemon klien dan menawarkan alat baris perintah termasuk, CloudHSM Management Utility (CMU), utilitas manajemen kunci (KMU), dan alat konfigurasi.

Dukungan dasar berbeda untuk setiap versi SDK AWS CloudHSM Klien. Biasanya dukungan platform untuk komponen dalam SDK cocok dengan dukungan dasar, tetapi tidak selalu. Untuk menentukan dukungan platform untuk komponen tertentu, pertama-tama pastikan platform yang Anda inginkan muncul di bagian dasar SDK, lalu periksa pengecualian atau informasi terkait lainnya di bagian komponen.

Dukungan platform berubah dari waktu ke waktu. Versi CloudHSM Client SDK sebelumnya mungkin tidak mendukung semua sistem operasi yang tercantum di sini. Gunakan catatan rilis untuk menentukan dukungan sistem operasi untuk versi CloudHSM Client SDK sebelumnya. Untuk informasi selengkapnya, lihat [Unduh untuk AWS CloudHSM Client SDK](#).

AWS CloudHSM hanya mendukung sistem operasi 64-bit.

Topik

- [Dukungan Linux untuk AWS CloudHSM Client SDK 3](#)
- [Dukungan Windows untuk AWS CloudHSM Client SDK 3](#)
- [Kompatibilitas HSM untuk AWS CloudHSM Klien SDK 3](#)

Dukungan Linux untuk AWS CloudHSM Client SDK 3

AWS CloudHSM Client SDK 3 mendukung sistem operasi dan platform Linux berikut.

- Amazon Linux
- Amazon Linux 2
 - CentOS 6.10+ 2
- CentOS 7.3+
 - CentOS 8 1,4
- Red Hat Enterprise Linux (RHEL) 6.10+ 2
- Red Hat Enterprise Linux (RHEL) 7.3+
 - Perusahaan Topi Merah Linux (RHEL) 8 1
- Ubuntu 16.04 LTS 3
- Ubuntu 18.04 LTS 1

[1] Tidak ada dukungan untuk OpenSSL Dynamic Engine. Untuk informasi lebih lanjut, lihat [OpenSSL Dynamic Engine](#).

[2] Tidak ada dukungan untuk Client SDK 3.3.0 dan yang lebih baru.

[3] SDK 3.4 adalah rilis terakhir yang didukung di Ubuntu 16.04.

[4] SDK 3.4 adalah rilis terakhir yang didukung di CentOS 8.3+.

Dukungan Windows untuk AWS CloudHSM Client SDK 3

AWS CloudHSM Client SDK 3 mendukung versi Windows Server berikut.

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016

- Microsoft Windows Server 2019

Kompatibilitas HSM untuk AWS CloudHSM Klien SDK 3

Tabel berikut menjelaskan kompatibilitas AWS CloudHSM Client SDK 3 untuk HSMs.

hsm1.sedang	hsm2m.sedang
Kompatibel dengan versi Klien SDK 3.1.0 dan yang lebih baru.	Tidak didukung.

Pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 3

PKCS #11 adalah standar untuk melakukan operasi kriptografi pada modul keamanan perangkat keras (HSM) di. AWS CloudHSM

Untuk informasi tentang bootstrap, lihat. [Menghubungkan ke cluster](#)

Topik

- [Instal pustaka PKCS #11 untuk AWS CloudHSM Client SDK 3](#)
- [Mengautentikasi ke pustaka PKCS #11 untuk AWS CloudHSM SDK Klien 3](#)
- [Tipe kunci yang didukung untuk pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 3](#)
- [Mekanisme yang didukung untuk AWS CloudHSM Klien SDK 3](#)
- [Operasi API yang didukung untuk AWS CloudHSM Client SDK 3](#)
- [Atribut kunci di pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 3](#)
- [Contoh kode untuk pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 3](#)

Instal pustaka PKCS #11 untuk AWS CloudHSM Client SDK 3

Topik ini memberikan petunjuk untuk menginstal pustaka PKCS #11 untuk seri versi AWS CloudHSM Client SDK 3. [Untuk informasi selengkapnya tentang pustaka SDK Klien atau PKCS #11, lihat Menggunakan pustaka SDK Klien dan PKCS #11.](#)

Prasyarat untuk Klien SDK 3

Pustaka PKCS #11 membutuhkan klien. AWS CloudHSM

Jika Anda belum menginstal dan mengkonfigurasi AWS CloudHSM klien, lakukan itu sekarang dengan mengikuti langkah-langkah di [Instal klien \(Linux\)](#). Setelah Anda menginstal dan mengatur konfigurasi klien, gunakan perintah berikut untuk memulainya.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo systemctl start cloudhsm-client
```

CentOS 7

```
$ sudo systemctl start cloudhsm-client
```

CentOS 8

```
$ sudo systemctl start cloudhsm-client
```

RHEL 7

```
$ sudo systemctl start cloudhsm-client
```

RHEL 8

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 16.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 20.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

Instal pustaka PKCS #11 untuk Client SDK 3

Perintah berikut mengunduh dan memasang pustaka PKCS #11.

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-pkcs11-latest.el6.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

CentOS 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

RHEL 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-pkcs11_latest_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-pkcs11_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client-pkcs11_latest_u18.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-pkcs11_latest_u18.04_amd64.deb
```

- Jika instans EC2 tempat Anda menginstal pustaka PKCS #11 tidak memiliki komponen lain dari SDK Klien 3 yang diinstal, Anda harus bootstrap SDK Klien 3. Anda hanya perlu melakukan ini sekali pada setiap contoh dengan komponen dari SDK Klien 3.
- Anda dapat menemukan berkas pustaka PKCS #11 di lokasi berikut:

Linux biner, skrip konfigurasi, sertifikat, dan berkas log:

```
/opt/cloudhsm/lib
```

Mengautentikasi ke pustaka PKCS #11 untuk AWS CloudHSM SDK Klien 3

Saat Anda menggunakan pustaka PKCS #11, aplikasi Anda berjalan sebagai [pengguna kriptografi \(CU\)](#) tertentu di dalam Anda HSMs . AWS CloudHSM Aplikasi Anda dapat melihat dan mengelola hanya kunci yang CU miliki dan bagikan. Anda dapat menggunakan CU yang ada di HSMs atau membuat CU baru. Untuk informasi tentang pengelolaan CUs, lihat Mengelola pengguna [HSM](#)

dengan CloudHSM CLI dan Mengelola pengguna HSM dengan CloudHSM Management Utility (CMU).

Untuk menentukan CU ke pustaka PKCS #11, gunakan parameter pin dari PKCS #11 [Fungsi C_Login](#). Untuk AWS CloudHSM, parameter pin memiliki format berikut:

```
<CU_user_name>:<password>
```

Sebagai contoh, perintah berikut menetapkan pin pustaka PKCS #11 ke CU dengan nama pengguna CryptoUser dan kata sandi CUPassword123!.

```
CryptoUser:CUPassword123!
```

Tipe kunci yang didukung untuk pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 3

Pustaka PKCS #11 mendukung tipe kunci berikut dengan AWS CloudHSM Client SDK 3.

Tipe Kunci	Deskripsi
RSA	Hasilkan 2048-bit sampai 4096-bit kunci RSA, dengan penambahan 256 bit.
EC	Hasilkan kunci dengan kurva secp224r1 (P-224), secp256r1 (P-256), secp256k1 (Blockchain), secp384r1 (P-384), dan secp521r1 (P-521).
AES	Hasilkan kunci AES 128, 192, dan 256-bit.
DES3 (Tiga DES)	Hasilkan kunci 192-bit DES3 . Lihat catatan 1 di bawah untuk perubahan yang akan datang.
GENERIC_SECRET	Hasilkan 1 sampai 64 byte rahasia generik.

- [1] Sesuai dengan panduan NIST, ini tidak diizinkan untuk cluster dalam mode FIPS setelah 2023. Untuk cluster dalam mode non-FIPS, masih diperbolehkan setelah 2023. Lihat [Kepatuhan FIPS 140: Penutupan Mekanisme 2024](#) untuk detail.

Mekanisme yang didukung untuk AWS CloudHSM Klien SDK 3

Pustaka PKCS #11 mendukung algoritme berikut untuk AWS CloudHSM Client SDK 3:

- Enkripsi dan dekripsi — AES-CBC, AES-CTR, AES-ECB, AES-GCM, -CBC, -ECB, RSA-OAEP, dan RSA-PKCS DES3 DES3
- Tanda tangani dan verifikasi- RSA, HMAC, dan ECDSA; dengan dan tanpa hashing
- Hash/digest — SHA1,,, SHA224, dan SHA256 SHA384 SHA512
- Bungkus kunci— Bungkus Kunci AES,⁴AES-GCM, RSA-AES, dan RSA-OAEP
- Derivasi kunci — ECDH, -108 CTR KDF ⁵ SP800

Tabel mekanisme-fungsi perpustakaan PKCS #11

Pustaka PKCS #11 sesuai dengan spesifikasi PKCS #11 versi 2.40. Untuk memanggil fitur kriptografi menggunakan PKCS #11, panggil fungsi dengan mekanisme tertentu. Tabel berikut merangkum kombinasi fungsi dan mekanisme yang didukung oleh AWS CloudHSM.

Menafsirkan tabel mekanisme-fungsi PKCS #11 yang didukung

Tanda ✓ menunjukkan bahwa AWS CloudHSM mendukung mekanisme untuk fungsi tersebut. Kami tidak mendukung semua fungsi yang mungkin tercantum dalam spesifikasi PKCS #11. Tanda ✕ menunjukkan bahwa AWS CloudHSM belum mendukung mekanisme untuk fungsi yang diberikan, meskipun standar PKCS #11 mengizinkannya. Sel kosong menunjukkan bahwa standar PKCS #11 tidak mendukung mekanisme fungsi tertentu.

Mekanisme dan fungsi pustaka PKCS #11 yang didukung

Mekanisme	Fungsi						
	Hasilkan Kunci atau Pasangan kunci	Tandatangan & Verifikasi	SR & VR	Intisari	Enkripsi & Dekripsi	Turunkan Kunci	Bungkus & UnWrap
CKM_RSA_PKCS_KEY_PAIR_GEN	✓						

Mekanisme	Fungsi					
CKM_RSA_X_9_31_KEY_PAIR_GEN	✓ 2					
CKM_RSA_X_509		✓			✓	
CKM_RSA_PKCS_1		✓ 1	✗		✓ 1	✓ 1
lihat catatan 8						
CKM_RSA_PKCS_OAEP					✓ 1	✓ 6
CKM_SHA1_RSA_PKCS		✓ 3.2				
CKM_SHA224_RSA_PKCS		✓ 3.2				
CKM_SHA256_RSA_PKCS		✓ 3.2				
CKM_SHA384_RSA_PKCS		✓ 2,3.2				
CKM_SHA512_RSA_PKCS		✓ 3.2				
CKM_RSA_PKCS_PSS		✓ 1				

Mekanisme	Fungsi						
CKM_SHA1_RSA_PKCS_PSS		✓ 3.2					
CKM_SHA224_RSA_PKCS_PSS		✓ 3.2					
CKM_SHA256_RSA_PKCS_PSS		✓ 3.2					
CKM_SHA384_RSA_PKCS_PSS		✓ 2,3.2					
CKM_SHA512_RSA_PKCS_PSS		✓ 3.2					
CKM_EC_KEY_PAIR_GENERATION	✓						
CKM_ECDSA		✓ 1					
CKM_ECDSA_SHA1		✓ 3.2					
CKM_ECDSA_SHA224		✓ 3.2					
CKM_ECDSA_SHA256		✓ 3.2					

Mekanisme	Fungsi						
CKM_ECDSA_SHA384		✓ 3.2					
CKM_ECDSA_SHA512		✓ 3.2					
CKM_ECDH1_DERIVE						✓ 5	
CKM_SP800_108_COUNTER_KDF						✓	
CKM_GENERIC_SECRET_KEY_GEN	✓						
CKM_AES_KEY_GEN	✓						
CKM_AES_ECB					✓		✗
CKM_AES_CTR					✓		✗
CKM_AES_CBC					✓ 3.3		✗
CKM_AES_CBC_PAD					✓		✗
CKM_DES3_KEY_GEN ^{lihat catatan 8}	✓						

Mekanisme	Fungsi						
CKM_DES3_ CBC lihat catatan 8					✓ 3.3		✗
CKM_DES3_ CBC_PAD lihat catatan 8					✓		✗
CKM_DES3_ ECB lihat catatan 8					✓		✗
CKM_AES_G CM					✓ 3.3 , 4		✓ 7.1
CKM_CLOUD HSM_AES_G CM					✓ 7.1		✓ 7.1
CKM_SHA_1				✓ 3.1			
CKM_SHA_1 _HMAC		✓ 3.3					
CKM_SHA22 4				✓ 3.1			
CKM_SHA22 4_HMAC		✓ 3.3					
CKM_SHA25 6				✓ 3.1			
CKM_SHA25 6_HMAC		✓ 3.3					

Mekanisme	Fungsi					
CKM_SHA384			✓ 3.1			
CKM_SHA384_HMAC		✓ 3.3				
CKM_SHA512			✓ 3.1			
CKM_SHA512_HMAC		✓ 3.3				
CKM_RSA_AES_KEY_WRAP						✓
CKM_AES_KEY_WRAP						✓
CKM_AES_KEY_WRAP_PAD						✓
CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD						✓ 7.1
CKM_CLOUDHSM_AES_KEY_WRAP_PAD_KCS5_PAD						✓ 7.1
CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD						✓ 7.1

Anotasi mekanisme

- [1] Operasi satu bagian saja.
- [2] Mekanisme secara fungsional identik dengan mekanisme CKM_RSA_PKCS_KEY_PAIR_GEN, tetapi menawarkan jaminan lebih kuat untuk pembuatan p dan q .
- [3.1] AWS CloudHSM mendekati hashing secara berbeda berdasarkan SDK Klien. Untuk Client SDK 3, tempat kita melakukan hashing tergantung pada ukuran data dan apakah Anda menggunakan bagian tunggal atau operasi multi_bagian.

Operasi satu bagian di SDK Klien 3

Tabel 3.1 mencantumkan ukuran kumpulan data maksimum untuk setiap mekanisme untuk SDK Klien 3. Seluruh hash dikomputasi di dalam HSM. Tidak ada dukungan untuk ukuran data yang lebih besar dari 16 KB.

Tabel 3.1, Ukuran set data maksimum untuk operasi satu bagian

Mekanisme	Ukuran Data Maksimum
CKM_SHA_1	16296
CKM_SHA224	16264
CKM_SHA256	16296
CKM_SHA384	16232
CKM_SHA512	16232

Operasi multipart Klien SDK 3

Dukungan untuk ukuran data lebih besar dari 16 KB, tetapi ukuran data menentukan tempat hashing berlangsung. Penyangga data kurang dari 16 KB di-hash dalam HSM. Penyangga antara 16 KB dan ukuran data maksimum untuk sistem Anda di-hash secara lokal dalam perangkat lunak. Ingat: Fungsi hash tidak memerlukan rahasia kriptografi, sehingga Anda dapat dengan aman melakukan komputasi pada mereka di luar HSM.

- [3.2] AWS CloudHSM mendekati hashing secara berbeda berdasarkan SDK Klien. Untuk Client SDK 3, tempat kita melakukan hashing tergantung pada ukuran data dan apakah Anda menggunakan bagian tunggal atau operasi multi_bagian.

Operasi satu bagian Klien SDK 3

Tabel 3.2 mencantumkan ukuran kumpulan data maksimum untuk setiap mekanisme untuk SDK Klien 3. Tidak ada dukungan untuk ukuran data yang lebih besar dari 16 KB.

Tabel 3.2, Ukuran set data maksimum untuk operasi satu bagian

Mekanisme	Ukuran Data Maksimum
CKM_SHA1_RSA_PKCS	16296
CKM_SHA224_RSA_PKCS	16264
CKM_SHA256_RSA_PKCS	16296
CKM_SHA384_RSA_PKCS	16232
CKM_SHA512_RSA_PKCS	16232
CKM_SHA1_RSA_PKCS_PSS	16296
CKM_SHA224_RSA_PKCS_PSS	16264
CKM_SHA256_RSA_PKCS_PSS	16296
CKM_SHA384_RSA_PKCS_PSS	16232
CKM_SHA512_RSA_PKCS_PSS	16232
CKM_ECDSA_SHA1	16296
CKM_ECDSA_SHA224	16264
CKM_ECDSA_SHA256	16296
CKM_ECDSA_SHA384	16232
CKM_ECDSA_SHA512	16232

Operasi multipart Klien SDK 3

Dukungan untuk ukuran data lebih besar dari 16 KB, tetapi ukuran data menentukan tempat hashing berlangsung. Penyangga data kurang dari 16 KB di-hash dalam HSM. Penyangga antara 16 KB dan ukuran data maksimum untuk sistem Anda di-hash secara lokal dalam perangkat lunak. Ingat: Fungsi hash tidak memerlukan rahasia kriptografi, sehingga Anda dapat dengan aman melakukan komputasi pada mereka di luar HSM.

- [3.3] Ketika beroperasi pada data dengan menggunakan salah satu mekanisme berikut, jika penyangga data melebihi ukuran data maksimum, hasil operasi dalam kesalahan. Untuk mekanisme ini, semua pemrosesan data harus terjadi di dalam HSM. Tabel berikut mencantumkan set ukuran data maksimum untuk setiap mekanisme:

Tabel 3.3, Ukuran set data maksimum

Mekanisme	Ukuran Data Maksimum
CKM_SHA_1_HMAC	16288
CKM_SHA224_HMAC	16256
CKM_SHA256_HMAC	16288
CKM_SHA384_HMAC	16224
CKM_SHA512_HMAC	16224
CKM_AES_CBC	16272
CKM_AES_GCM	16224
CKM_CLOUDHSM_AES_GCM	16224
CKM_DES3_CBC	16280

- [4] Saat melakukan enkripsi AES-GCM, HSM tidak menerima data vektor inisialisasi (IV) dari aplikasi. Anda harus menggunakan IV yang dihasilkannya. IV 12-byte yang disediakan oleh HSM ditulis ke dalam referensi memori yang ditunjukkan oleh elemen pIV dari parameter CK_GCM_PARAMS struktur yang Anda suplai. Untuk mencegah kebingungan pengguna, PKCS #11 SDK di versi 1.1.1 dan sebelumnya memastikan bahwa pIV menunjuk ke penyangga yang dinolkan ketika enkripsi AES-GCM diinisialisasi.

- [5] Klien SDK 3 saja. Mekanisme diimplementasikan untuk mendukung kasus pembongkaran SSL/TLS dan dijalankan hanya sebagian di dalam HSM. Sebelum menggunakan mekanisme ini, lihat “Masalah: Derivasi kunci ECDH dijalankan hanya sebagian di dalam HSM” di [Masalah yang diketahui untuk pustaka PKCS #11 untuk AWS CloudHSM](#) CKM_ECDH1_DERIVE tidak mendukung kurva secp521r1 (P-521).
- [6] CK_MECHANISM_TYPE dan CK_RSA_PKCS_MGF_TYPE berikut didukung sebagai CK_RSA_PKCS_OAEP_PARAMS untuk CKM_RSA_PKCS_OAEP:
 - CKM_SHA_1 menggunakan CKG_MGF1_SHA1
 - CKM_SHA224 menggunakan CKG_MGF1_SHA224
 - CKM_SHA256 menggunakan CKG_MGF1_SHA256
 - CKM_SHA384 menggunakan CKM_MGF1_SHA384
 - CKM_SHA512 menggunakan CKM_MGF1_SHA512
- [7.1] Mekanisme yang ditentukan vendor. Untuk menggunakan mekanisme yang ditetapkan vendor CloudHSM, aplikasi PKCS #11 harus menyertakan /opt/cloudhsm/include/pkcs11/cloudhsm_pkcs11_vendor_defs.h selama kompilasi.

CKM_CLOUDHSM_AES_GCM: Mekanisme kepemilikan ini adalah alternatif pemrograman yang lebih aman untuk CKM_AES_GCM standar. Ini menambahkan IV yang dihasilkan oleh HSM untuk ciphertext, bukan menuliskannya kembali ke dalam struktur CK_GCM_PARAMS yang disediakan selama inisialisasi cipher. Anda dapat menggunakan mekanisme ini dengan fungsi C_Encrypt, C_WrapKey, C_Decrypt, dan C_UnwrapKey. Bila menggunakan mekanisme ini, variabel pIV di struk CK_GCM_PARAMS harus diatur ke NULL. Bila menggunakan mekanisme ini dengan C_Decrypt dan C_UnwrapKey, IV diharapkan akan ditambahkan ke ciphertext yang sedang dibuka.

CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD: Bungkus Kunci AES dengan Padding PKCS #5

CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD: Bungkus Kunci AES dengan Padding Nol

Untuk informasi tambahan mengenai pembungkus kunci AES, lihat Pembungkus Kunci [AES](#).

- [8] Sesuai dengan panduan NIST, ini tidak diizinkan untuk cluster dalam mode FIPS setelah 2023. Untuk cluster dalam mode non-FIPS, masih diperbolehkan setelah 2023. Lihat [Kepatuhan FIPS 140: Penutupan Mekanisme 2024](#) untuk detail.

Operasi API yang didukung untuk AWS CloudHSM Client SDK 3

Pustaka PKCS #11 mendukung operasi API PKCS #11 berikut untuk AWS CloudHSM Client SDK 3.

- C_CloseAllSessions
- C_CloseSession
- C_CreateObject
- C_Decrypt
- C_DecryptFinal
- C_DecryptInit
- C_DecryptUpdate
- C_DeriveKey
- C_DestroyObject
- C_Digest
- C_DigestFinal
- C_DigestInit
- C_DigestUpdate
- C_Encrypt
- C_EncryptFinal
- C_EncryptInit
- C_EncryptUpdate
- C_Finalize
- C_FindObjects
- C_FindObjectsFinal
- C_FindObjectsInit
- C_GenerateKey
- C_GenerateKeyPair
- C_GenerateRandom
- C_GetAttributeValue
- C_GetFunctionList
- C_GetInfo

- C_GetMechanismInfo
- C_GetMechanismList
- C_GetSessionInfo
- C_GetSlotInfo
- C_GetSlotList
- C_GetTokenInfo
- C_Initialize
- C_Login
- C_Logout
- C_OpenSession
- C_Sign
- C_SignFinal
- C_SignInit
- C_SignRecover(Hanya dukungan SDK 3 Klien)
- C_SignRecoverInit(Hanya dukungan SDK 3 Klien)
- C_SignUpdate
- C_UnWrapKey
- C_Verify
- C_VerifyFinal
- C_VerifyInit
- C_VerifyRecover(Hanya dukungan SDK 3 Klien)
- C_VerifyRecoverInit(Hanya dukungan SDK 3 Klien)
- C_VerifyUpdate
- C_WrapKey

Atribut kunci di pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 3

Objek kunci bisa berupa kunci publik, pribadi, atau rahasia. Tindakan diizinkan pada objek kunci ditentukan melalui atribut. Atribut didefinisikan ketika objek kunci dibuat. Saat Anda menggunakan pustaka PKCS #11 untuk AWS CloudHSM, kami menetapkan nilai default seperti yang ditentukan oleh standar PKCS #11.

AWS CloudHSM tidak mendukung semua atribut yang tercantum dalam spesifikasi PKCS #11. Kami patuh dengan spesifikasi untuk semua atribut yang kami dukung. Atribut ini tercantum dalam tabel masing-masing.

Fungsi kriptografi seperti `C_CreateObject`, `C_GenerateKey`, `C_GenerateKeyPair`, `C_UnwrapKey`, dan `C_DeriveKey` yang membuat, memodifikasi, atau menyalin objek mengambil templat atribut sebagai salah satu parameternya. Untuk informasi lebih lanjut tentang melewati templat atribut selama pembuatan objek, lihat sampel [Hasilkan kunci melalui pustaka PKCS #11](#).

Topik berikut memberikan informasi selengkapnya tentang atribut AWS CloudHSM kunci untuk SDK Klien 3.

Topik

- [Tabel atribut pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 3](#)
- [Memodifikasi atribut pustaka PKCS #11 untuk AWS CloudHSM SDK Klien 3](#)
- [Menafsirkan kode kesalahan pustaka PKCS #11 untuk AWS CloudHSM SDK Klien 3](#)

Tabel atribut pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 3

Tabel pustaka PKCS #11 untuk AWS CloudHSM Client SDK 3 berisi daftar atribut yang berbeda menurut jenis kunci. Ini menunjukkan apakah atribut yang diberikan didukung untuk jenis kunci tertentu saat menggunakan fungsi kriptografi tertentu dengan AWS CloudHSM.

Legenda:

- ✓ menunjukkan bahwa CloudHSM mendukung atribut untuk jenis kunci tertentu.
- ✗ menunjukkan bahwa CloudHSM tidak mendukung atribut untuk jenis kunci tertentu.
- R menunjukkan bahwa nilai atribut diatur ke hanya-baca untuk jenis kunci tertentu.
- S menunjukkan bahwa atribut tidak dapat dibaca oleh `GetAttributeValue` karena sensitif.
- Sel kosong di kolom Nilai Default menunjukkan bahwa tidak ada nilai default tertentu yang ditetapkan untuk atribut.

GenerateKeyPair

Atribut	Tipe Kunci				Nilai Default
	EC pribadi	EC publik	RSA pribadi	RSA publik	
CKA_CLASS	✓	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	✓	
CKA_LABEL	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	Benar
CKA_TOKEN	✓	✓	✓	✓	Salah
CKA_PRIVATE	✓ ¹	✓ ¹	✓ ¹	✓ ¹	Benar
CKA_ENCRYPT	✗	✓	✗	✓	Salah
CKA_DECRYPT	✓	✗	✓	✗	Salah
CKA_DERIVE	✓	✓	✓	✓	Salah
CKA_MODIFIABLE	✓ ¹	✓ ¹	✓ ¹	✓ ¹	Benar

Atribut	Tipe Kunci				Nilai Default
CKA_DESTR OYABLE	✓	✓	✓	✓	Benar
CKA_SIGN	✓	✗	✓	✗	Salah
CKA_SIGN_ RECOVER	✗	✗	✓ ³	✗	
CKA_VERIF Y	✗	✓	✗	✓	Salah
CKA_VERIF Y_RECOVER	✗	✗	✗	✓ ⁴	
CKA_WRAP	✗	✓	✗	✓	Salah
CKA_WRAP_ TEMPLATE	✗	✓	✗	✓	
CKA_TRUST ED	✗	✓	✗	✓	Salah
CKA_WRAP_ WITH_TRUS TED	✓	✗	✓	✗	Salah
CKA_UNWRA P	✓	✗	✓	✗	Salah
CKA_UNWRA P_TEMPLAT E	✓	✗	✓	✗	
CKA_SENSI TIVE	✓	✗	✓	✗	Benar

Atribut	Tipe Kunci				Nilai Default
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	
CKA_EXTRACTABLE	✓	✖	✓	✖	Benar
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	
CKA_MODULUS	✖	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✓ ²	
CKA_PRIME_1	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	✖	

Atribut	Tipe Kunci				Nilai Default
CKA_PUBLIC_EXPONENT	×	×	×	✓ ²	
CKA_EC_PARAMS	×	✓ ²	×	×	
CKA_EC_POINT	×	×	×	×	
CKA_VALUE	×	×	×	×	
CKA_VALUE_LEN	×	×	×	×	
CKA_CHECK_VALUE	R	R	R	R	

GenerateKey

Atribut	Tipe Kunci			Nilai Default
	AES	DES3	Rahasia Generik	
CKA_CLASS	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	

Atribut	Tipe Kunci				Nilai Default
CKA_LOCAL		R	R	R	Benar
CKA_TOKEN		✓	✓	✓	Salah
CKA_PRIVATE		✓ ¹	✓ ¹	✓ ¹	Benar
CKA_ENCRYPT		✓	✓	✗	Salah
CKA_DECRYPT		✓	✓	✗	Salah
CKA_DERIVE		✓	✓	✓	Salah
CKA_MODIFIABLE		✓ ¹	✓ ¹	✓ ¹	Benar
CKA_DESTROYABLE		✓	✓	✓	Benar
CKA_SIGN		✓	✓	✓	Benar
CKA_SIGN_RECOVER		✗	✗	✗	
CKA_VERIFY		✓	✓	✓	Benar
CKA_VERIFY_RECOVER		✗	✗	✗	
CKA_WRAP		✓	✓	✗	Salah
CKA_WRAP_TEMPLATE		✓	✓	✗	

Atribut	Tipe Kunci				Nilai Default
CKA_TRUSTED		✓	✓	✗	Salah
CKA_WRAP_WITH_TRUSTED		✓	✓	✓	Salah
CKA_UNWRAP		✓	✓	✗	Salah
CKA_UNWRAP_TEMPLATE		✓	✓	✗	
CKA_SENSITIVE		✓	✓	✓	Benar
CKA_ALWAYS_SENSITIVE		✗	✗	✗	
CKA_EXTRACTABLE		✓	✓	✓	Benar
CKA_NEVER_EXTRACTABLE		R	R	R	
CKA_MODULUS		✗	✗	✗	
CKA_MODULUS_BITS		✗	✗	✗	
CKA_PRIME_1		✗	✗	✗	

Atribut	Tipe Kunci			Nilai Default
CKA_PRIME_2	×	×	×	
CKA_COEFFICIENT	×	×	×	
CKA_EXPONENT_1	×	×	×	
CKA_EXPONENT_2	×	×	×	
CKA_PRIVATE_EXPONENT	×	×	×	
CKA_PUBLIC_EXPONENT	×	×	×	
CKA_EC_PARAMS	×	×	×	
CKA_EC_POINT	×	×	×	
CKA_VALUE	×	×	×	
CKA_VALUE_LEN	✓ ²	×	✓ ²	
CKA_CHECK_VALUE	R	R	R	

CreateObject

Atribut	Tipe Kunci							Nilai Default
	EC pribadi	EC publik	RSA pribadi	RSA publik	AES	DES3	Rahasia Generik	
CKA_CLASS	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	
CKA_KEY_TYPE	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	R	R	Salah
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓	Salah
CKA_PRIVATE	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	Benar
CKA_ENCRYPT	✗	✗	✗	✓	✓	✓	✗	Salah
CKA_DECRYPT	✗	✗	✓	✗	✓	✓	✗	Salah
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓	Salah
CKA_MODIFIABLE	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	Benar

Atribut	Tipe Kunci							Nilai Default
CKA_DESTR OYABLE	✓	✓	✓	✓	✓	✓	✓	Benar
CKA_SIGN	✓	✗	✓	✗	✓	✓	✓	Salah
CKA_SIGN_ RECOVER	✗	✗	✓ ³	✗	✗	✗	✗	Salah
CKA_VERIF Y	✗	✓	✗	✓	✓	✓	✓	Salah
CKA_VERIF Y_RECOVER	✗	✗	✗	✓ ⁴	✗	✗	✗	
CKA_WRAP	✗	✗	✗	✓	✓	✓	✗	Salah
CKA_WRAP_ TEMPLATE	✗	✓	✗	✓	✓	✓	✗	
CKA_TRUST ED	✗	✓	✗	✓	✓	✓	✗	Salah
CKA_WRAP_ WITH_TRUS TED	✓	✗	✓	✗	✓	✓	✓	Salah
CKA_UNWRA P	✗	✗	✓	✗	✓	✓	✗	Salah
CKA_UNWRA P_TEMPLAT E	✓	✗	✓	✗	✓	✓	✗	
CKA_SENSI TIVE	✓	✗	✓	✗	✓	✓	✓	Benar

Atribut	Tipe Kunci							Nilai Default
CKA_ALWAYS_SENSITIVE	R	×	R	×	R	R	R	
CKA_EXTRACTABLE	✓	×	✓	×	✓	✓	✓	Benar
CKA_NEVER_EXTRACTABLE	R	×	R	×	R	R	R	
CKA_MODULUS	×	×	✓ ²	✓ ²	×	×	×	
CKA_MODULUS_BITS	×	×	×	×	×	×	×	
CKA_PRIME_1	×	×	✓	×	×	×	×	
CKA_PRIME_2	×	×	✓	×	×	×	×	
CKA_COEFFICIENT	×	×	✓	×	×	×	×	
CKA_EXPONENT_1	×	×	✓	×	×	×	×	
CKA_EXPONENT_2	×	×	✓	×	×	×	×	
CKA_PRIVATE_EXPONENT	×	×	✓ ²	×	×	×	×	

Atribut	Tipe Kunci							Nilai Default
CKA_PUBLIC_EXPONENT	×	×	✓ ²	✓ ²	×	×	×	
CKA_EC_PARAMS	✓ ²	✓ ²	×	×	×	×	×	
CKA_EC_POINT	×	✓ ²	×	×	×	×	×	
CKA_VALUE	✓ ²	×	×	×	✓ ²	✓ ²	✓ ²	
CKA_VALUE_LEN	×	×	×	×	×	×	×	
CKA_CHECK_VALUE	R	R	R	R	R	R	R	

UnwrapKey

Atribut	Tipe Kunci					Nilai Default
	EC pribadi	RSA pribadi	AES	DES3	Rahasia Generik	
CKA_CLASS	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	
CKA_KEY_TYPE	✓ ²	✓ ²	✓ ²	✓ ²	✓ ²	
CKA_LABEL	✓	✓	✓	✓	✓	

Atribut	Tipe Kunci						Nilai Default
CKA_ID		✓	✓	✓	✓	✓	
CKA_LOCAL		R	R	R	R	R	Salah
CKA_TOKEN		✓	✓	✓	✓	✓	Salah
CKA_PRIVATE		✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	Benar
CKA_ENCRYPT		✗	✗	✓	✓	✗	Salah
CKA_DECRYPT		✗	✓	✓	✓	✗	Salah
CKA_DERIVE		✓	✓	✓	✓	✓	Salah
CKA_MODIFIABLE		✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	Benar
CKA_DESTROYABLE		✓	✓	✓	✓	✓	Benar
CKA_SIGN		✓	✓	✓	✓	✓	Salah
CKA_SIGN_RECOVER		✗	✓ ³	✗	✗	✗	Salah
CKA_VERIFY		✗	✗	✓	✓	✓	Salah
CKA_VERIFY_RECOVER		✗	✗	✗	✗	✗	

Atribut	Tipe Kunci						Nilai Default
CKA_WRAP		✗	✗	✓	✓	✗	Salah
CKA_UNWRAP		✗	✓	✓	✓	✗	Salah
CKA_SENSITIVE		✓	✓	✓	✓	✓	Benar
CKA_EXTRACTABLE		✓	✓	✓	✓	✓	Benar
CKA_NEVER_EXTRACTABLE		R	R	R	R	R	
CKA_ALWAYS_SENSITIVE		R	R	R	R	R	
CKA_MODULUS		✗	✗	✗	✗	✗	
CKA_MODULUS_BITS		✗	✗	✗	✗	✗	
CKA_PRIME_1		✗	✗	✗	✗	✗	
CKA_PRIME_2		✗	✗	✗	✗	✗	
CKA_COEFFICIENT		✗	✗	✗	✗	✗	
CKA_EXPONENT_1		✗	✗	✗	✗	✗	

Atribut	Tipe Kunci						Nilai Default
CKA_EXPONENT_2		×	×	×	×	×	
CKA_PRIVATE_EXPONENT		×	×	×	×	×	
CKA_PUBLIC_EXPONENT		×	×	×	×	×	
CKA_EC_PARAMS		×	×	×	×	×	
CKA_EC_POINT		×	×	×	×	×	
CKA_VALUE		×	×	×	×	×	
CKA_VALUE_LEN		×	×	×	×	×	
CKA_CHECK_VALUE		R	R	R	R	R	

DeriveKey

Atribut	Tipe Kunci				Nilai Default
		AES	DES3	Rahasia Generik	
CKA_CLASS		✓ ₂	✓ ₂	✓ ₂	

Atribut	Tipe Kunci			Nilai Default
CKA_KEY_T YPE	✓ ²	✓ ²	✓ ²	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	Benar
CKA_TOKEN	✓	✓	✓	Salah
CKA_PRIVATE TE	✓ ¹	✓ ¹	✓ ¹	Benar
CKA_ENCRY PT	✓	✓	✗	Salah
CKA_DECRY PT	✓	✓	✗	Salah
CKA_DERIV E	✓	✓	✓	Salah
CKA_MODIF IABLE	✓ ¹	✓ ¹	✓ ¹	Benar
CKA_DESTR OYABLE	✓ ¹	✓ ¹	✓ ¹	Benar
CKA_SIGN	✓	✓	✓	Salah
CKA_SIGN_ RECOVER	✗	✗	✗	
CKA_VERIF Y	✓	✓	✓	Salah

Atribut	Tipe Kunci			Nilai Default
CKA_VERIFY_RECOVER	✗	✗	✗	
CKA_WRAP	✓	✓	✗	Salah
CKA_UNWRAP	✓	✓	✗	Salah
CKA_SENSITIVE	✓	✓	✓	Benar
CKA_EXTRACTABLE	✓	✓	✓	Benar
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	
CKA_MODULUS	✗	✗	✗	
CKA_MODULUS_BITS	✗	✗	✗	
CKA_PRIME_1	✗	✗	✗	
CKA_PRIME_2	✗	✗	✗	
CKA_COEFFICIENT	✗	✗	✗	

Atribut	Tipe Kunci			Nilai Default
CKA_EXPONENT_1	×	×	×	
CKA_EXPONENT_2	×	×	×	
CKA_PRIVATE_EXPONENT	×	×	×	
CKA_PUBLIC_EXPONENT	×	×	×	
CKA_EC_PARAMS	×	×	×	
CKA_EC_POINT	×	×	×	
CKA_VALUE	×	×	×	
CKA_VALUE_LEN	✓ ²	×	✓ ²	
CKA_CHECK_VALUE	R	R	R	

GetAttributeValue

Atribut	Tipe Kunci						
	EC pribadi	EC publik	RSA pribadi	RSA publik	AES	DES3	Rahasia Generik

Atribut	Tipe Kunci						
CKA_CLASS	✓	✓	✓	✓	✓	✓	✓
CKA_KEY_TYPE	✓	✓	✓	✓	✓	✓	✓
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓
CKA_ID	✓	✓	✓	✓	✓	✓	✓
CKA_LOCAL	✓	✓	✓	✓	✓	✓	✓
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓
CKA_PRIVATE	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹	✓ ¹
CKA_ENCRYPT	✗	✗	✗	✓	✓	✓	✗
CKA_DECRYPT	✗	✗	✓	✗	✓	✓	✗
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	✓
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓
CKA_SIGN	✓	✗	✓	✗	✓	✓	✓

Atribut	Tipe Kunci						
CKA_SIGN_RECOVER	×	×	✓	×	×	×	×
CKA_VERIFY	×	✓	×	✓	✓	✓	✓
CKA_VERIFY_RECOVER	×	×	×	✓	×	×	×
CKA_WRAP	×	×	×	✓	✓	✓	×
CKA_WRAP_TEMPLATE	×	✓	×	✓	✓	✓	×
CKA_TRUSTED	×	✓	×	✓	✓	✓	✓
CKA_WRAP_WITH_TRUSTED	✓	×	✓	×	✓	✓	✓
CKA_UNWRAP	×	×	✓	×	✓	✓	×
CKA_UNWRAP_TEMPLATE	✓	×	✓	×	✓	✓	×
CKA_SENSITIVE	✓	×	✓	×	✓	✓	✓
CKA_EXTRACTABLE	✓	×	✓	×	✓	✓	✓

Atribut	Tipe Kunci						
CKA_NEVER_EXTRACTABLE	✓	✗	✓	✗	✓	✓	✓
CKA_ALWAYS_SENSITIVE	R	R;	R	R	R	R	R
CKA_MODULUS	✗	✗	✓	✓	✗	✗	✗
CKA_MODULUS_BITS	✗	✗	✗	✓	✗	✗	✗
CKA_PRIME_1	✗	✗	S	✗	✗	✗	✗
CKA_PRIME_2	✗	✗	S	✗	✗	✗	✗
CKA_COEFFICIENT	✗	✗	S	✗	✗	✗	✗
CKA_EXPONENT_1	✗	✗	S	✗	✗	✗	✗
CKA_EXPONENT_2	✗	✗	S	✗	✗	✗	✗
CKA_PRIVATE_EXPONENT	✗	✗	S	✗	✗	✗	✗
CKA_PUBLIC_EXPONENT	✗	✗	✓	✓	✗	✗	✗

Atribut	Tipe Kunci						
CKA_EC_PA RAMS	✓	✓	✗	✗	✗	✗	✗
CKA_EC_PO INT	✗	✓	✗	✗	✗	✗	✗
CKA_VALUE	S	✗	✗	✗	✓ ²	✓ ²	✓ ²
CKA_VALUE _LEN	✗	✗	✗	✗	✓	✗	✓
CKA_CHECK _VALUE	✓	✓	✓	✓	✓	✓	✗

Anotasi atribut

- [1] Atribut ini sebagian didukung oleh firmware dan harus secara eksplisit diatur hanya ke nilai default.
- [2] Atribut wajib.
- [3] Klien SDK 3 saja. CKA_SIGN_RECOVERAtribut berasal dari CKA_SIGN atribut. Jika sedang diatur, itu hanya dapat diatur ke nilai yang sama yang ditetapkan untuk CKA_SIGN. Jika tidak diatur, itu menurunkan nilai default dari CKA_SIGN. Karena CloudHSM hanya mendukung mekanisme tanda tangan dipulihkan berbasis RSA, atribut ini saat ini hanya berlaku untuk kunci publik RSA.
- [4] Klien SDK 3 saja. CKA_VERIFY_RECOVERAtribut berasal dari CKA_VERIFY atribut. Jika sedang diatur, itu hanya dapat diatur ke nilai yang sama yang ditetapkan untuk CKA_VERIFY. Jika tidak diatur, itu menurunkan nilai default dari CKA_VERIFY. Karena CloudHSM hanya mendukung mekanisme tanda tangan dipulihkan berbasis RSA, atribut ini saat ini hanya berlaku untuk kunci publik RSA.

Memodifikasi atribut pustaka PKCS #11 untuk AWS CloudHSM SDK Klien 3

Beberapa atribut dari suatu objek dapat dimodifikasi setelah objek dibuat, sedangkan beberapa objek tidak bisa. Untuk mengubah atribut, gunakan perintah [setAttribute](#) dari `cloudhsm_mgmt_util`.

Anda juga dapat memperoleh daftar atribut dan konstanta yang mewakilinya dengan menggunakan perintah [listAttribute](#) dari `cloudhsm_mgmt_util`.

Daftar berikut menampilkan atribut yang diizinkan untuk modifikasi setelah pembuatan objek:

- CKA_LABEL
- CKA_TOKEN

 Note

Modifikasi hanya diperbolehkan untuk mengubah kunci sesi menjadi kunci token. Gunakan perintah [setAttribute](#) dari `key_mgmt_util` untuk mengubah nilai atribut.

- CKA_ENCRYPT
- CKA_DECRYPT
- CKA_SIGN
- CKA_VERIFY
- CKA_WRAP
- CKA_UNWRAP
- CKA_LABEL
- CKA_SENSITIVE
- CKA_DERIVE

 Note

Atribut ini mendukung derivasi kunci. Kunci harus `False` untuk semua kunci publik dan tidak dapat diatur ke `True`. Untuk kunci rahasia dan kunci privat EC, kunci dapat diatur ke `True` atau `False`.

- CKA_TRUSTED

 Note

Atribut ini dapat diatur ke `True` atau `False` oleh Petugas Kripto (CO) saja.

- CKA_WRAP_WITH_TRUSTED

Note

Terapkan atribut ini ke kunci data yang dapat diekspor untuk menentukan bahwa Anda hanya dapat membungkus kunci ini dengan kunci yang ditandai sebagai CKA_TRUSTED. Setelah Anda mengatur CKA_WRAP_WITH_TRUSTED menjadi true (benar), atribut menjadi hanya-baca dan Anda tidak dapat mengubah atau menghapus atribut.

Menafsirkan kode kesalahan pustaka PKCS #11 untuk AWS CloudHSM SDK Klien 3

Menentukan dalam template atribut pustaka PKCS #11 yang tidak didukung oleh kunci tertentu menghasilkan kesalahan. Tabel berikut berisi kode galat yang dihasilkan ketika Anda melanggar spesifikasi:

Kode Kesalahan	Deskripsi
CKR_TEMPLATE_INCONSISTENT	Anda menerima galat ini ketika Anda menetapkan atribut dalam templat atribut, tempat atribut patuh dengan spesifikasi PKCS #11, tetapi tidak didukung oleh CloudHSM.
CKR_ATTRIBUTE_TYPE_INVALID	Anda menerima galat ini ketika Anda mengambil nilai untuk atribut, yang patuh dengan spesifikasi PKCS #11, tetapi tidak didukung oleh CloudHSM.
CKR_ATTRIBUTE_INCOMPLETE	Anda menerima galat ini ketika Anda tidak menentukan atribut wajib dalam templat atribut.
CKR_ATTRIBUTE_READ_ONLY	Anda menerima galat ini ketika Anda menetapkan atribut hanya-baca dalam templat atribut.

Contoh kode untuk pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 3

Contoh kode GitHub menunjukkan kepada Anda cara menyelesaikan tugas dasar menggunakan pustaka PKCS #11 untuk AWS CloudHSM

Contoh prasyarat kode

Sebelum menjalankan sampel, lakukan langkah-langkah berikut untuk mengatur lingkungan Anda:

- Instal dan konfigurasi [pustaka PKCS #11](#) untuk Client SDK 3.
- Siapkan [pengguna kriptografi \(CU\)](#). Aplikasi Anda menggunakan akun HSM ini untuk menjalankan sampel kode pada HSM.

Sampel Kode

Sampel Kode untuk Perpustakaan Perangkat AWS CloudHSM Lunak untuk PKCS #11 tersedia di [GitHub](#). Repositori ini mencakup contoh tentang bagaimana melakukan operasi umum menggunakan PKCS #11 termasuk enkripsi, dekripsi, penandatanganan, dan verifikasi.

- [Hasilkan kunci \(AES, RSA, EC\)](#)
- [Daftar atribut kunci](#)
- [Enkripsi dan dekripsi data dengan AES GCM](#)
- [Enkripsi dan dekripsi data dengan AES_CTR](#)
- [Enkripsi dan dekripsi data dengan 3DES](#)
- [Tanda tangani dan verifikasi data dengan RSA](#)
- [Turunkan kunci menggunakan HMAC KDF](#)
- [Bungkus dan buka kunci dengan AES menggunakan padding PKCS #5](#)
- [Bungkus dan buka kunci dengan AES tanpa padding](#)
- [Bungkus dan buka kunci dengan AES menggunakan bantalan nol](#)
- [Bungkus dan buka kunci dengan AES-GCM](#)
- [Bungkus dan buka kunci dengan RSA](#)

OpenSSL Dynamic Engine AWS CloudHSM untuk Klien SDK 3

AWS CloudHSM OpenSSL Dynamic Engine memungkinkan Anda untuk menurunkan operasi kriptografi ke cluster CloudHSM Anda melalui OpenSSL API.

AWS CloudHSM Client SDK 3 memang memerlukan daemon klien untuk terhubung ke cluster. Ini mendukung:

- Pembuatan kunci RSA untuk kunci 2048, 3072, dan 4096-bit.

- Tanda tangan/verifikasi RSA.
- RSA mengenkripsi/mendekripsi.
- Pembuatan angka acak yang aman secara kriptografis dan divalidasi FIPS.

Gunakan bagian berikut untuk menginstal dan mengkonfigurasi mesin AWS CloudHSM dinamis untuk OpenSSL.

Topik

- [Prasyarat untuk OpenSSL Dynamic Engine dengan Client SDK 3 AWS CloudHSM](#)
- [Instal OpenSSL Dynamic Engine AWS CloudHSM untuk Client SDK 3](#)
- [Gunakan OpenSSL Dynamic Engine AWS CloudHSM untuk Client SDK 3](#)

Prasyarat untuk OpenSSL Dynamic Engine dengan Client SDK 3 AWS CloudHSM

Untuk informasi tentang dukungan platform, lihat [AWS CloudHSM Platform yang didukung SDK Klien 3](#).

Sebelum Anda dapat menggunakan mesin AWS CloudHSM dinamis untuk OpenSSL dengan Client SDK 3, Anda memerlukan klien. AWS CloudHSM

Klien adalah daemon yang membangun komunikasi end-to-end terenkripsi dengan HSMs di cluster Anda, dan mesin OpenSSL berkomunikasi secara lokal dengan klien. Untuk menginstal dan mengkonfigurasi AWS CloudHSM klien, lihat [Instal klien \(Linux\)](#). Kemudian gunakan perintah berikut untuk memulainya.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo systemctl start cloudhsm-client
```

CentOS 6

```
$ sudo systemctl start cloudhsm-client
```

CentOS 7

```
$ sudo systemctl start cloudhsm-client
```

RHEL 6

```
$ sudo systemctl start cloudhsm-client
```

RHEL 7

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 16.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

Instal OpenSSL Dynamic Engine AWS CloudHSM untuk Client SDK 3

Langkah-langkah berikut menjelaskan cara menginstal dan mengkonfigurasi mesin AWS CloudHSM dinamis untuk OpenSSL dengan Client SDK 3. Untuk informasi tentang peningkatan, lihat [Tingkatkan SDK Klien 3](#).

Untuk menginstal dan mengatur konfigurasi mesin OpenSSL

1. Gunakan salah satu perintah berikut untuk mengunduh dan menjalankan mesin OpenSSL.

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

CentOS 6

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

RHEL 6

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-dyn_latest_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-dyn_latest_amd64.deb
```

Mesin OpenSSL dipasang di `/opt/cloudhsm/lib/libcloudhsm_openssl.so`

- Gunakan perintah berikut untuk mengatur variabel lingkungan bernama `n3fips_password` yang berisi kredensial dari pengguna krypto (CU).

```
$ export n3fips_password=<HSM user name>:<password>
```

Gunakan OpenSSL Dynamic Engine AWS CloudHSM untuk Client SDK 3

Untuk menggunakan mesin AWS CloudHSM dinamis untuk OpenSSL dari aplikasi terintegrasi OpenSSL, pastikan bahwa aplikasi Anda menggunakan mesin dinamis OpenSSL bernama `cloudhsm`. Pustaka bersama untuk mesin dinamis terletak di `/opt/cloudhsm/lib/libcloudhsm_openssl.so`.

Untuk menggunakan mesin AWS CloudHSM dinamis untuk OpenSSL dari baris perintah OpenSSL, gunakan `-engine` opsi untuk menentukan mesin dinamis OpenSSL bernama `cloudhsm`. Contoh:

```
$ openssl s_server -cert <server.crt> -key <server.key> -engine cloudhsm
```

Penyedia JCE untuk AWS CloudHSM Klien SDK 3

Penyedia AWS CloudHSM JCE adalah implementasi penyedia yang dibangun dari kerangka penyedia Java Cryptographic Extension (JCE). JCE memungkinkan Anda untuk melakukan operasi kriptografi menggunakan Java Development Kit (JDK). Dalam panduan ini, penyedia AWS CloudHSM JCE kadang-kadang disebut sebagai penyedia JCE. Gunakan penyedia JCE dan JDK untuk membongkar operasi kriptografi ke HSM.

Topik

- [Instal penyedia JCE untuk AWS CloudHSM Client SDK 3](#)
- [Dasar-dasar manajemen utama di penyedia JCE untuk AWS CloudHSM Client SDK 3](#)
- [Mekanisme yang didukung untuk SDK Klien 3 untuk SDK AWS CloudHSM Klien 3](#)
- [Atribut kunci Java yang didukung untuk AWS CloudHSM Klien SDK 3](#)
- [Contoh kode untuk pustaka AWS CloudHSM perangkat lunak untuk Java untuk Client SDK 3](#)

- [AWS CloudHSM KeyStore Kelas Java untuk Client SDK 3](#)

Instal penyedia JCE untuk AWS CloudHSM Client SDK 3

Sebelum Anda dapat menggunakan penyedia JCE, Anda memerlukan AWS CloudHSM klien.

Klien adalah daemon yang membangun komunikasi end-to-end terenkripsi dengan di cluster Anda. HSMs Penyedia JCE berkomunikasi secara lokal dengan klien. Jika Anda belum menginstal dan mengkonfigurasi paket AWS CloudHSM klien, lakukan itu sekarang dengan mengikuti langkah-langkah di [Instal klien \(Linux\)](#). Setelah Anda menginstal dan mengatur konfigurasi klien, gunakan perintah berikut untuk memulainya.

Perhatikan bahwa penyedia JCE hanya didukung pada Linux dan sistem operasi yang kompatibel.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo systemctl start cloudhsm-client
```

CentOS 7

```
$ sudo systemctl start cloudhsm-client
```

CentOS 8

```
$ sudo systemctl start cloudhsm-client
```

RHEL 7

```
$ sudo systemctl start cloudhsm-client
```

RHEL 8

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 16.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 18.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

Ubuntu 20.04 LTS

```
$ sudo systemctl start cloudhsm-client
```

Gunakan bagian berikut untuk menginstal, memvalidasi, dan memberikan kredensi kepada penyedia.

Topik

- [Langkah 1: Instal penyedia JCE](#)
- [Langkah 2: Validasi instalasi](#)
- [Langkah 3: Berikan kredensial ke penyedia JCE](#)

Langkah 1: Instal penyedia JCE

Gunakan perintah berikut untuk mengunduh dan menginstal penyedia JCE. Penyedia ini didukung hanya di Linux dan sistem operasi yang kompatibel.

Note

Untuk meningkatkan, lihat [Tingkatkan SDK Klien 3](#).

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-jce-latest.el6.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el7.x86_64.rpm
```

CentOS 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-jce-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el8.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el7.x86_64.rpm
```

RHEL 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-jce-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el8.x86_64.rpm
```


Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-jce_latest_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-jce_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client-jce_latest_u18.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-jce_latest_u18.04_amd64.deb
```

Setelah Anda menjalankan perintah sebelumnya, Anda dapat menemukan file penyedia JCE berikut:

- /opt/cloudhsm/java/cloudhsm-*<version>*.jar
- /opt/cloudhsm/java/cloudhsm-test-*<version>*.jar
- /opt/cloudhsm/java/hamcrest-all-1.3.jar
- /opt/cloudhsm/java/junit.jar
- /opt/cloudhsm/java/log4j-api-2.17.1.jar
- /opt/cloudhsm/java/log4j-core-2.17.1.jar
- /opt/cloudhsm/lib/libcaviumjca.so

Langkah 2: Validasi instalasi

Melakukan operasi dasar pada HSM untuk memvalidasi instalasi.

Untuk memvalidasi instalasi penyedia JCE

1. (Opsional) Jika Anda belum memiliki Java diinstal di lingkungan Anda, gunakan perintah berikut untuk menginstalnya.

Linux (and compatible libraries)

```
$ sudo yum install java-1.8.0-openjdk
```

Ubuntu

```
$ sudo apt-get install openjdk-8-jre
```

- Gunakan perintah berikut untuk mengatur variabel lingkungan yang diperlukan. Ganti *<HSM user name>* dan *<password>* dengan kredensial pengguna krypto (CU).

```
$ export LD_LIBRARY_PATH=/opt/cloudhsm/lib
```

```
$ export HSM_PARTITION=PARTITION_1
```

```
$ export HSM_USER=<HSM user name>
```

```
$ export HSM_PASSWORD=<password>
```

- Gunakan perintah berikut untuk menjalankan tes fungsionalitas dasar. Jika berhasil, output perintah harus serupa dengan berikut.

```
$ java8 -classpath "/opt/cloudhsm/java/*" org.junit.runner.JUnitCore  
TestBasicFunctionality
```

```
JUnit version 4.11  
.2018-08-20 17:53:48,514 DEBUG [main] TestBasicFunctionality  
  (TestBasicFunctionality.java:33) - Adding provider.  
2018-08-20 17:53:48,612 DEBUG [main] TestBasicFunctionality  
  (TestBasicFunctionality.java:42) - Logging in.  
2018-08-20 17:53:48,612 INFO [main] cfm2.LoginManager (LoginManager.java:104) -  
  Looking for credentials in HsmCredentials.properties  
2018-08-20 17:53:48,612 INFO [main] cfm2.LoginManager (LoginManager.java:122) -  
  Looking for credentials in System.properties  
2018-08-20 17:53:48,613 INFO [main] cfm2.LoginManager (LoginManager.java:130) -  
  Looking for credentials in System.env  
SDK Version: 2.03  
2018-08-20 17:53:48,655 DEBUG [main] TestBasicFunctionality  
  (TestBasicFunctionality.java:54) - Generating AES Key with key size 256.  
2018-08-20 17:53:48,698 DEBUG [main] TestBasicFunctionality  
  (TestBasicFunctionality.java:63) - Encrypting with AES Key.  
2018-08-20 17:53:48,705 DEBUG [main] TestBasicFunctionality  
  (TestBasicFunctionality.java:84) - Deleting AES Key.
```

```
2018-08-20 17:53:48,707 DEBUG [main] TestBasicFunctionality
(TestBasicFunctionality.java:92) - Logging out.

Time: 0.205

OK (1 test)
```

Langkah 3: Berikan kredensial ke penyedia JCE

HSMs perlu mengotentikasi aplikasi Java Anda sebelum aplikasi dapat menggunakannya. Setiap aplikasi dapat menggunakan satu sesi. HSMs mengautentikasi sesi dengan menggunakan login eksplisit atau metode login implisit.

Login eksplisit — Metode ini memungkinkan Anda memberikan kredensial CloudHSM langsung dalam aplikasi. Menggunakan metode `LoginManager.login()`, yaitu Anda melewati nama pengguna CU, kata sandi, dan ID partisi HSM. Untuk informasi lebih lanjut menggunakan metode login eksplisit, lihat sampel kode [Login ke HSM](#).

Login implisit — Metode ini memungkinkan Anda mengatur kredensial CloudHSM baik dalam file properti baru, properti sistem, atau sebagai variabel lingkungan.

- **Properti baru file** — Membuat file baru dengan nama `HsmCredentials.properties` dan menambahkannya ke CLASSPATH aplikasi. File tersebut harus berisi hal berikut:

```
HSM_PARTITION = PARTITION_1
HSM_USER = <HSM user name>
HSM_PASSWORD = <password>
```

- **Properti sistem** — Set kredensial melalui properti sistem saat menjalankan aplikasi Anda. Contoh berikut menunjukkan dua cara berbeda yang dapat dilakukan:

```
$ java -DHSM_PARTITION=PARTITION_1 -DHSM_USER=<HSM user name> -
DHSM_PASSWORD=<password>
```

```
System.setProperty("HSM_PARTITION", "PARTITION_1");
System.setProperty("HSM_USER", "<HSM user name>");
System.setProperty("HSM_PASSWORD", "<password>");
```

- **Variabel lingkungan** — Set kredensial sebagai variabel lingkungan.

```
$ export HSM_PARTITION=PARTITION_1
$ export HSM_USER=<HSM user name>
$ export HSM_PASSWORD=<password>
```

Kredensial mungkin tidak tersedia jika aplikasi tidak menyediakannya atau jika Anda mencoba operasi sebelum HSM mengautentikasi sesi. Dalam kasus tersebut, pustaka perangkat lunak CloudHSM untuk Java mencari kredensialnya dengan urutan sebagai berikut:

1. `HsmCredentials.properties`
2. Properti sistem
3. Variabel lingkungan

Penanganan kesalahan

Penanganan kesalahan lebih mudah dengan login eksplisit dari metode login implisit. Saat Anda menggunakan kelas `LoginManager`, Anda memiliki kontrol lebih atas bagaimana aplikasi Anda menangani kegagalan. Metode login implisit membuat penanganan kesalahan sulit dipahami ketika kredensialnya tidak valid atau mengalami masalah dalam mengautentikasi HSMs sesi.

Dasar-dasar manajemen utama di penyedia JCE untuk AWS CloudHSM Client SDK 3

Dasar-dasar manajemen kunci dalam penyedia JCE melibatkan mengimpor kunci, mengeksport kunci, memuat kunci dengan pegangan, atau menghapus kunci. Untuk informasi lebih lanjut tentang mengelola kunci, lihat sampel kode [Kelola kunci](#).

Anda juga dapat menemukan lebih banyak contoh kode penyedia JCE di [Sampel Kode](#).

Mekanisme yang didukung untuk SDK Klien 3 untuk SDK AWS CloudHSM Klien 3

Topik ini memberikan informasi tentang mekanisme yang didukung untuk penyedia JCE dengan AWS CloudHSM Client SDK 3. Untuk informasi tentang antarmuka Java Cryptography Architecture (JCA) dan kelas mesin yang didukung oleh AWS CloudHSM, lihat topik berikut.

Topik

- [Kunci yang didukung](#)
- [Cipher yang didukung](#)
- [Intisari yang didukung](#)

- [Algoritma kode otentikasi pesan berbasis hash \(HMAC\) yang didukung](#)
- [sign/verify Mekanisme yang didukung](#)
- [Anotasi mekanisme](#)

Kunci yang didukung

Perpustakaan AWS CloudHSM perangkat lunak untuk Java memungkinkan Anda untuk menghasilkan jenis kunci berikut.

- AES — 128, 192, dan 256-bit kunci AES.
- DESede - Kunci 3DES 92 bit. Lihat catatan [1](#) di bawah untuk perubahan yang akan datang.
- Pasangan kunci ECC untuk kurva NIST secp256r1 (P-256), secp384r1 (P-384), dan secp256k1 (Rantai blok).
- RSA — 2048-bit sampai 4096-bit kunci RSA, dengan penambahan 256 bit.

Selain parameter standar, kami mendukung parameter berikut untuk setiap kunci yang dihasilkan.

- Label: Label kunci yang dapat Anda gunakan untuk mencari kunci.
- isExtractable: Menunjukkan apakah kunci dapat diekspor dari HSM.
- isPersistent: Menunjukkan apakah kunci tetap pada HSM ketika sesi saat berakhir.

Note

Java library versi 3.1 menyediakan kemampuan untuk menentukan parameter secara lebih detail. Untuk informasi lebih lanjut, lihat [Atribut Java yang Didukung](#).

Cipher yang didukung

Pustaka AWS CloudHSM perangkat lunak untuk Java mendukung kombinasi algoritma, mode, dan padding berikut.

Algoritme	Mode	Bantalan	Catatan
AES	CBC	AES/CBC/N oPadding	Menerapkan Cipher.EN

Algoritme	Mode	Bantalan	Catatan
		AES/CBC/P KCS5Padding	CRYPT_MODE dan Cipher.DE CRYPT_MODE .
AES	ECB	AES/ECB/N oPadding AES/ECB/P KCS5Padding	Menerapkan Cipher.EN CRYPT_MODE dan Cipher.DE CRYPT_MODE . Gunakan AES Transformasi.
AES	CTR	AES/CTR/N oPadding	Menerapkan Cipher.EN CRYPT_MODE dan Cipher.DE CRYPT_MODE .

Algoritme	Mode	Bantalan	Catatan
AES	GCM	AES/GCM/NoPadding	Menerapkan <code>Cipher.ENCRYPT_MODE</code> dan <code>Cipher.DECRYPT_MODE</code>, <code>Cipher.WRAP_MODE</code>, dan <code>Cipher.UNWRAP_MODE</code>. Ketika melakukan enkripsi AES-GCM, HSM mengabaikan vektor inisialisasi (IV) dalam permintaan dan menggunakan IV yang dihasilkan. Saat operasi selesai, Anda harus memanggil <code>Cipher.getIV()</code> untuk mendapatkan IV.
AESWrap	ECB	AESWrap/ECB/ZeroPadding AESWrap/ECB/NoPadding AESWrap/ECB/PKCS5Padding	Menerapkan <code>Cipher.WRAP_MODE</code> dan <code>Cipher.UNWRAP_MODE</code>. Gunakan AES Transformasi.

Algoritme	Mode	Bantalan	Catatan
DESede (Tiga DES)	CBC	DESede/CBC/ NoPadding DESede/CBC/ PKCS5Padding	Menerapkan <code>Cipher.ENCRYPT_MODE</code> dan <code>Cipher.DECRYPT_MODE</code>. Rutinitas pembuatan kunci menerima ukuran 168 atau 192 bit. Namun, secara internal, semua DESede kunci adalah 192 bit. Lihat catatan 1 di bawah untuk perubahan yang akan datang.

Algoritme	Mode	Bantalan	Catatan
DESede (Tiga DES)	ECB	DESede/ECB/ NoPadding DESede/ECB/ PKCS5Padding	Menerapkan <code>Cipher.ENCRYPT_MODE</code> dan <code>Cipher.DECRYPT_MODE</code>. Rutinitas pembuatan kunci menerima ukuran 168 atau 192 bit. Namun, secara internal, semua DESede kunci adalah 192 bit. Lihat catatan 1 di bawah untuk perubahan yang akan datang.
RSA	ECB	RSA/ECB/ NoPadding RSA/ECB/ PKCS1Padding	Menerapkan <code>Cipher.ENCRYPT_MODE</code> dan <code>Cipher.DECRYPT_MODE</code>. Lihat catatan 1 di bawah untuk perubahan yang akan datang.

Algoritme	Mode	Bantalan	Catatan
RSA	ECB	RSA/ECB/0 AEPPadding	Menerapkan Cipher.EN CRYPT_MOD E , Cipher.DE CRYPT_MOD E , Cipher.WR AP_MODE , dan Cipher.UN WRAP_MODE . OAEPPadding adalah OAEP dengan jenis bantalan SHA-1.
		RSA/ECB/0 AEPWithSH A-1ANDMGF 1Padding	
		RSA/ECB/0 AEPWithSH A-224ANDM GF1Padding	
		RSA/ECB/0 AEPWithSH A-256ANDM GF1Padding	
		RSA/ECB/0 AEPWithSH A-384ANDM GF1Padding	
		RSA/ECB/0 AEPWithSH A-512ANDM GF1Padding	
RSAAESWrap	ECB	OAEPPADDING	Menerapkan Cipher.WR AP_Mode dan Cipher.UN WRAP_MODE .

Intisari yang didukung

Pustaka AWS CloudHSM perangkat lunak untuk Java mendukung intisari pesan berikut.

- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512

Note

Data dengan panjang di bawah 16 KB di-hash pada HSM, sementara data yang lebih besar di-hash secara lokal dalam perangkat lunak.

Algoritma kode otentikasi pesan berbasis hash (HMAC) yang didukung

Perpustakaan AWS CloudHSM perangkat lunak untuk Java mendukung algoritma HMAC berikut.

- HmacSHA1
- HmacSHA224
- HmacSHA256
- HmacSHA384
- HmacSHA512

sign/verify Mekanisme yang didukung

Pustaka AWS CloudHSM perangkat lunak untuk Java mendukung jenis tanda tangan dan verifikasi berikut.

Jenis Tanda Tangan RSA

- NONEwithRSA
- SHA1withRSA
- SHA224withRSA

- SHA256withRSA
- SHA384withRSA
- SHA512withRSA
- SHA1withRSA/PSS
- SHA224withRSA/PSS
- SHA256withRSA/PSS
- SHA384withRSA/PSS
- SHA512withRSA/PSS

Jenis Tanda Tangan ECDSA

- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

Anotasi mekanisme

[1] Sesuai dengan panduan NIST, ini tidak diizinkan untuk cluster dalam mode FIPS setelah 2023. Untuk cluster dalam mode non-FIPS, masih diperbolehkan setelah 2023. Lihat [Kepatuhan FIPS 140: Penutupan Mekanisme 2024](#) untuk rincian selengkapnya.

Atribut kunci Java yang didukung untuk AWS CloudHSM Klien SDK 3

Topik ini menjelaskan cara menggunakan ekstensi proprietary untuk pustaka Java versi 3.1 untuk menetapkan atribut kunci untuk AWS CloudHSM Client SDK 3. Gunakan ekstensi ini untuk mengatur atribut kunci yang didukung dan nilai-nilainya selama operasi ini:

- Pembuatan kunci
- Impor kunci
- Buka kunci

Note

Ekstensi untuk menetapkan atribut kunci kustom adalah fitur opsional. Jika Anda sudah memiliki kode yang berfungsi di pustaka Java versi 3.0, Anda tidak perlu memodifikasi kode itu. Kunci yang Anda buat akan terus berisi atribut yang sama seperti sebelumnya.

Topik

- [Memahami atribut](#)
- [Atribut yang didukung](#)
- [Menyetel atribut untuk kunci](#)
- [Menyatukan semuanya](#)

Memahami atribut

Anda menggunakan atribut kunci untuk menentukan tindakan apa yang diizinkan pada objek kunci, termasuk kunci publik, pribadi atau rahasia. Anda menentukan atribut kunci dan nilai-nilai selama operasi pembuatan objek kunci.

Namun, Java Cryptography Extension (JCE) tidak menentukan bagaimana Anda harus menetapkan nilai pada atribut kunci, sehingga sebagian besar tindakan diizinkan secara default. Sebaliknya, PKCS # 11 standar menentukan satu set lengkap atribut dengan default lebih ketat. Dimulai dengan pustaka Java versi 3.1, CloudHSM menyediakan ekstensi proprietary yang memungkinkan Anda untuk mengatur nilai yang lebih ketat untuk atribut yang umum digunakan.

Atribut yang didukung

Anda dapat mengatur nilai untuk atribut yang tercantum dalam tabel di bawah ini. Sebagai praktik terbaik, tetapkan hanya nilai untuk atribut yang ingin Anda buat ketat. Jika Anda tidak menentukan nilai, CloudHSM menggunakan nilai default yang ditentukan dalam tabel di bawah ini. Sel kosong di kolom Nilai Default menunjukkan bahwa tidak ada nilai default tertentu yang ditetapkan untuk atribut.

Atribut	Nilai Default		Catatan
	Kunci Simetris	Kunci Publik dalam Pasangan Kunci	Kunci Pribadi dalam Pasangan Kunci

Atribut	Nilai Default			Catatan
CKA_TOKEN	FALSE	FALSE	FALSE	Kunci permanen yang direplika si HSMs di semua kluster dan disertakan dalam cadangan. CKA_TOKEN = FALSE menyiratkan kunci sesi, yang hanya dimuat ke satu HSM dan secara otomatis dihapus ketika koneksi ke HSM rusak.
CKA_LABEL				String yang ditentukan pengguna. Hal ini memungkinkan Anda untuk dengan mudah mengidentifikasi kunci pada HSM Anda.
CKA_EXTRACTABLE	TRUE		TRUE	Benar menunjukkan Anda dapat mengekspor kunci ini keluar dari HSM.

Atribut	Nilai Default			Catatan
CKA_ENCRYPT	TRUE	TRUE		Benar menunjukkan Anda dapat menggunakan kunci untuk mengenkripsi penyangga apa pun.
CKA_DECRYPT	TRUE		TRUE	Benar menunjukkan Anda dapat menggunakan kunci untuk mendekripsi penyangga apa pun. Anda umumnya mengatur ini ke FALSE untuk kunci dengan CKA_WRAP diatur ke true.
CKA_WRAP	TRUE	TRUE		Benar menunjukkan Anda dapat menggunakan kunci untuk membungkus kunci lain. Anda biasanya akan mengatur ini ke FALSE untuk kunci privat.

Atribut	Nilai Default			Catatan
CKA_UNWRAP	TRUE		TRUE	Benar menunjukkan Anda dapat menggunakan kunci untuk membungkus kunci lain.
CKA_SIGN	TRUE		TRUE	Benar menunjukkan Anda dapat menggunakan kunci untuk menandatangani digest pesan. Hal ini umumnya diatur ke FALSE untuk kunci publik dan kunci privat yang telah Anda arsipkan.
CKA_VERIFY	TRUE	TRUE		Benar menunjukkan Anda dapat menggunakan kunci untuk menandatangani digest pesan. Hal ini umumnya diatur ke FALSE untuk kunci privat.

Atribut	Nilai Default			Catatan
CKA_PRIVATE	TRUE	TRUE	TRUE	Benar menunjukkan bahwa pengguna tidak dapat mengakses kunci sampai pengguna diautentikasi. Untuk kejelasan, pengguna tidak dapat mengakses kunci apa pun di CloudHSM sampai mereka diautentikasi, bahkan jika atribut ini diatur ke FALSE.

 Note

Anda mendapatkan dukungan yang lebih luas untuk atribut di pustaka PKCS #11. Untuk informasi lebih lanjut, lihat [Atribut PKCS #11 yang Didukung](#).

Menyetel atribut untuk kunci

CloudHsmKeyAttributesMap adalah objek seperti [Peta Java](#), yang dapat Anda gunakan untuk mengatur nilai atribut untuk objek kunci. Metode untuk fungsi CloudHsmKeyAttributesMap yang mirip dengan metode yang digunakan untuk manipulasi peta Java.

Untuk mengatur nilai kustom pada atribut, Anda memiliki dua opsi:

- Gunakan metode yang tercantum dalam tabel berikut

- Gunakan pola pembangun yang ditunjukkan kemudian dalam dokumen ini

Atribut peta objek mendukung metode berikut untuk mengatur atribut:

Operasi	Nilai Pengembalian	Metode CloudHSMKeyAttributesMap
Dapatkan nilai atribut kunci untuk kunci yang ada	Obyek (berisi nilai) atau nol	<code>get(keyAttribute)</code>
Isi nilai satu atribut kunci	Nilai sebelumnya terkait dengan atribut kunci, atau nol jika tidak ada pemetaan untuk atribut kunci	<code>put(keyAttribute, value)</code>
Isi nilai untuk beberapa atribut kunci	N/A	<code>putAll () keyAttributesMap</code>
Hapus pasangan nilai kunci dari peta atribut	Nilai sebelumnya terkait dengan atribut kunci, atau nol jika tidak ada pemetaan untuk atribut kunci	<code>remove(keyAttribute)</code>

Note

Setiap atribut yang tidak secara eksplisit Anda tentukan diatur ke default yang tercantum dalam tabel sebelumnya di [the section called “Atribut yang didukung”](#).

Contoh pola pembangun

Developer umumnya akan merasa lebih nyaman untuk memanfaatkan kelas melalui pola Builder. Sebagai contoh:

```
import com.amazonaws.cloudhsm.CloudHsmKeyAttributes;
import com.amazonaws.cloudhsm.CloudHsmKeyAttributesMap;
import com.amazonaws.cloudhsm.CloudHsmKeyPairAttributesMap;
```

```

CloudHsmKeyAttributesMap keyAttributesSessionDecryptionKey =
    new CloudHsmKeyAttributesMap.Builder()
        .put(CloudHsmKeyAttributes.CKA_LABEL, "ExtractableSessionKeyEncryptDecrypt")
        .put(CloudHsmKeyAttributes.CKA_WRAP, false)
        .put(CloudHsmKeyAttributes.CKA_UNWRAP, false)
        .put(CloudHsmKeyAttributes.CKA_SIGN, false)
        .put(CloudHsmKeyAttributes.CKA_VERIFY, false)
        .build();

CloudHsmKeyAttributesMap keyAttributesTokenWrappingKey =
    new CloudHsmKeyAttributesMap.Builder()
        .put(CloudHsmKeyAttributes.CKA_LABEL, "TokenWrappingKey")
        .put(CloudHsmKeyAttributes.CKA_TOKEN, true)
        .put(CloudHsmKeyAttributes.CKA_ENCRYPT, false)
        .put(CloudHsmKeyAttributes.CKA_DECRYPT, false)
        .put(CloudHsmKeyAttributes.CKA_SIGN, false)
        .put(CloudHsmKeyAttributes.CKA_VERIFY, false)
        .build();

```

Developer juga dapat memanfaatkan set atribut yang telah ditetapkan sebagai cara yang mudah untuk menegakkan praktik terbaik dalam templat kunci. Sebagai contoh:

```

//best practice template for wrapping keys

CloudHsmKeyAttributesMap commonKeyAttrs = new CloudHsmKeyAttributesMap.Builder()
    .put(CloudHsmKeyAttributes.CKA_EXTRACTABLE, false)
    .put(CloudHsmKeyAttributes.CKA_DECRYPT, false)
    .build();

// initialize a new instance of CloudHsmKeyAttributesMap by copying commonKeyAttrs
// but with an appropriate label

CloudHsmKeyAttributesMap firstKeyAttrs = new CloudHsmKeyAttributesMap(commonKeyAttrs);
firstKeyAttrs.put(CloudHsmKeyAttributes.CKA_LABEL, "key label");

// alternatively, putAll() will overwrite existing values to enforce conformance

CloudHsmKeyAttributesMap secondKeyAttrs = new CloudHsmKeyAttributesMap();
secondKeyAttrs.put(CloudHsmKeyAttributes.CKA_DECRYPT, true);
secondKeyAttrs.put(CloudHsmKeyAttributes.CKA_ENCRYPT, true);
secondKeyAttrs.put(CloudHsmKeyAttributes.CKA_LABEL, "safe wrapping key");
secondKeyAttrs.putAll(commonKeyAttrs); // will overwrite CKA_DECRYPT to be FALSE

```

Menetapkan atribut untuk key pair

Gunakan kelas Java `CloudHsmKeyPairAttributesMap` untuk menangani atribut kunci untuk pasangan kunci. `CloudHsmKeyPairAttributesMap` merangkum dua objek `CloudHsmKeyAttributesMap`; satu untuk kunci publik dan satu untuk kunci privat.

Untuk mengatur atribut individu untuk kunci publik dan kunci privat secara terpisah, Anda dapat menggunakan metode `put()` pada objek peta `CloudHsmKeyAttributes` yang terkait untuk kunci itu. Gunakan metode `getPublic()` untuk mengambil peta atribut untuk kunci publik, dan gunakan `getPrivate()` untuk mengambil peta atribut untuk kunci privat. Isi nilai dari beberapa atribut kunci bersama-sama untuk kedua pasangan kunci publik dan privat menggunakan `putAll()` dengan peta atribut pasangan kunci sebagai argumen.

Contoh pola pembangun

Developer umumnya akan merasa lebih nyaman untuk mengatur atribut kunci melalui pola Builder. Contoh:

```
import com.amazonaws.cloudhsm.CloudHsmKeyAttributes;
import com.amazonaws.cloudhsm.CloudHsmKeyAttributesMap;
import com.amazonaws.cloudhsm.CloudHsmKeyPairAttributesMap;

//specify attributes up-front
CloudHsmKeyAttributesMap keyAttributes =
    new CloudHsmKeyAttributesMap.Builder()
        .put(CloudHsmKeyAttributes.CKA_SIGN, false)
        .put(CloudHsmKeyAttributes.CKA_LABEL, "PublicCertSerial12345")
        .build();

CloudHsmKeyPairAttributesMap keyPairAttributes =
    new CloudHsmKeyPairAttributesMap.Builder()
        .withPublic(keyAttributes)
        .withPrivate(
            new CloudHsmKeyAttributesMap.Builder() //or specify them inline
                .put(CloudHsmKeyAttributes.CKA_LABEL, "PrivateCertSerial12345")
                .put(CloudHsmKeyAttributes.CKA_WRAP, FALSE)
                .build()
        )
        .build();
```

Note

[Untuk informasi selengkapnya tentang ekstensi kepemilikan ini, lihat arsip Javadoc dan sampelnya.](#) [GitHub](#) Untuk menjelajahi Javadoc, unduh dan perluas arsip.

Menyatukan semuanya

Untuk menentukan atribut kunci dengan operasi kunci Anda, ikuti langkah berikut:

1. Instantiate `CloudHsmKeyAttributesMap` untuk kunci simetris atau `CloudHsmKeyPairAttributesMap` untuk pasangan kunci.
2. Tentukan objek atribut dari langkah 1 dengan atribut kunci dan nilai-nilai yang diperlukan.
3. Instantiate kelas `Cavium*ParameterSpec`, sesuai dengan jenis kunci tertentu Anda, dan masuk ke konstruktor objek atribut yang dikonfigurasi ini.
4. Masukkan objek `Cavium*ParameterSpec` ke kelas atau metode krypto yang sesuai.

Untuk referensi, tabel berikut berisi kelas dan metode `Cavium*ParameterSpec` yang mendukung atribut kunci kustom.

Tipe Kunci	Kelas Spesifikasi Parameter	Contoh Konstruktor
Kelas Dasar	<code>CaviumKeyGenAlgorithmParameterSpec</code>	<code>CaviumKeyGenAlgorithmParameterSpec(CloudHsmKeyAttributesMap keyAttributesMap)</code>
DES	<code>CaviumDESKeyGenParameterSpec</code>	<code>CaviumDESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttributesMap keyAttributesMap)</code>
RSA	<code>CaviumRSAKeyGenParameterSpec</code>	<code>CaviumRSAKeyGenParameterSpec(int</code>

Tipe Kunci	Kelas Spesifikasi Parameter	Contoh Konstruktor
		<code>keysize, BigInteger publicExponent, Clo udHsmKeyPairAttrib utesMap keyPairAt tributesMap)</code>
Rahasia	<code>CaviumGenericSecre tKeyGenParameterSp ec</code>	<code>CaviumGenericSecre tKeyGenParameterSp ec(int size, CloudHsmKeyAttribu tesMap key AttributesMap)</code>
AES	<code>CaviumAESKeyGenPar ameterSpec</code>	<code>CaviumAESKeyGenPar ameterSpec(int keySize, byte[] iv, CloudHsmKeyAttribu tesMap key AttributesMap)</code>
EC	<code>CaviumECGenParamet erSpec</code>	<code>CaviumECGenParamet erSpec(String stdName, CloudHsmK eyPairAttributesMa p keyPairA ttributesMap)</code>

Contoh kode: Buat dan bungkus kunci

Contoh kode singkat ini menunjukkan langkah-langkah untuk dua operasi yang berbeda: Pembuatan Kunci dan Pembungkusan Kunci:

```
// Set up the desired key attributes

KeyGenerator keyGen = KeyGenerator.getInstance("AES", "Cavium");
CaviumAESKeyGenParameterSpec keyAttributes = new CaviumAESKeyGenParameterSpec(
```

```
256,  
new CloudHsmKeyAttributesMap.Builder()  
    .put(CloudHsmKeyAttributes.CKA_LABEL, "MyPersistentAESKey")  
    .put(CloudHsmKeyAttributes.CKA_EXTRACTABLE, true)  
    .put(CloudHsmKeyAttributes.CKA_TOKEN, true)  
    .build()  
);  
  
// Assume we already have a handle to the myWrappingKey  
// Assume we already have the wrappedBytes to unwrap  
  
// Unwrap a key using Custom Key Attributes  
  
CaviumUnwrapParameterSpec unwrapSpec = new  
    CaviumUnwrapParameterSpec(myInitializationVector, keyAttributes);  
  
Cipher unwrapCipher = Cipher.getInstance("AESWrap", "Cavium");  
unwrapCipher.init(Cipher.UNWRAP_MODE, myWrappingKey, unwrapSpec);  
Key unwrappedKey = unwrapCipher.unwrap(wrappedBytes, "AES", Cipher.SECRET_KEY);
```

Contoh kode untuk pustaka AWS CloudHSM perangkat lunak untuk Java untuk Client SDK 3

Topik ini menyediakan sumber daya dan informasi tentang contoh kode Java untuk AWS CloudHSM Client SDK 3.

Prasyarat

Sebelum menjalankan sampel, Anda harus mengatur lingkungan Anda:

- Instal dan konfigurasi [Penyedia Java Cryptographic Extension \(JCE\)](#) dan [paket klien AWS CloudHSM](#).
- Siapkan [nama pengguna HSM dan kata sandi](#) yang valid. Izin pengguna kriptografi (CU) cukup untuk tugas-tugas ini. Aplikasi Anda menggunakan kredensial ini untuk login ke HSM di setiap contoh.
- Tentukan cara memberikan kredensial ke [penyedia JCE](#).

Sampel Kode

Sampel kode berikut menunjukkan cara menggunakan [penyedia JCE AWS CloudHSM](#) untuk melakukan tugas dasar. Lebih banyak contoh kode tersedia di [GitHub](#).

- [Masuk ke HSM](#)
- [Kelola kunci](#)
- [Hasilkan kunci AES](#)
- [Enkripsi dan dekripsi dengan AES-GCM](#)
- [Enkripsi dan dekripsi dengan AES-CTR](#)
- [Enkripsi dan dekripsi dengan D3DES-ECB lihat catatan 1](#)
- [Bungkus dan buka kunci dengan AES-GCM](#)
- [Bungkus dan buka kunci dengan AES](#)
- [Bungkus dan buka kunci dengan RSA](#)
- [Gunakan atribut kunci yang didukung](#)
- [Enumerasi kunci di toko kunci](#)
- [Gunakan toko kunci CloudHSM](#)
- [Menandatangani pesan dalam sampel multi-utas](#)
- [Masuk dan Verifikasi dengan Kunci EC](#)

[1] Sesuai dengan panduan NIST, ini tidak diizinkan untuk cluster dalam mode FIPS setelah 2023. Untuk cluster dalam mode non-FIPS, masih diperbolehkan setelah 2023. Lihat [Kepatuhan FIPS 140: Penutupan Mekanisme 2024](#) untuk rincian selengkapnya.

AWS CloudHSM KeyStore Kelas Java untuk Client SDK 3

AWS CloudHSM KeyStoreKelas ini menyediakan penyimpanan PKCS12 kunci tujuan khusus yang memungkinkan akses ke AWS CloudHSM kunci melalui aplikasi seperti keytool dan jarsigner. Penyimpanan kunci ini dapat menyimpan sertifikat bersama dengan data kunci Anda dan menghubungkannya dengan data kunci yang disimpan di AWS CloudHSM.

Note

Karena sertifikat adalah informasi publik, dan untuk memaksimalkan kapasitas penyimpanan untuk kunci kriptografi, AWS CloudHSM tidak mendukung penyimpanan sertifikat. HSMs

AWS CloudHSM KeyStoreKelas mengimplementasikan KeyStore Service Provider Interface (SPI) dari Java Cryptography Extension (JCE). Untuk informasi selengkapnya tentang penggunaanKeyStore, lihat [Kelas KeyStore](#).

Pilih penyimpanan kunci yang sesuai untuk AWS CloudHSM Client SDK 3

Penyedia AWS CloudHSM Java Cryptographic Extension (JCE) dilengkapi dengan pass-through default, read-only key store yang meneruskan semua transaksi ke HSM. Penyimpanan kunci default ini berbeda dari tujuan khusus AWS CloudHSM KeyStore. Dalam kebanyakan situasi, Anda akan mendapatkan performa waktu aktif yang lebih baik dan throughput dengan menggunakan default. Anda hanya boleh menggunakan AWS CloudHSM KeyStore untuk aplikasi di mana Anda memerlukan dukungan untuk sertifikat dan operasi berbasis sertifikat selain membongkar operasi kunci ke HSM.

Meskipun kedua penyimpanan kunci menggunakan penyedia JCE untuk operasi, keduanya adalah entitas independen dan tidak bertukar informasi satu sama lain.

Muat penyimpanan kunci default untuk aplikasi Java Anda sebagai berikut:

```
KeyStore ks = KeyStore.getInstance("Cavium");
```

Muat CloudHSM tujuan khusus sebagai berikut: KeyStore

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

Inisialisasi AWS CloudHSM KeyStore untuk Client SDK 3

Masuk ke cara AWS CloudHSM KeyStore yang sama seperti Anda masuk ke penyedia JCE. Anda dapat menggunakan variabel lingkungan atau file properti sistem, dan Anda harus masuk sebelum mulai menggunakan CloudHSM KeyStore. Untuk contoh masuk ke HSM menggunakan penyedia JCE, lihat [Masuk ke HSM](#).

Jika diinginkan, Anda dapat menentukan kata sandi untuk mengenkripsi PKCS12 file lokal yang menyimpan data penyimpanan kunci. Saat Anda membuat AWS CloudHSM Keystore, Anda mengatur kata sandi dan memberikannya saat menggunakan metode pemuatan, atur, dan dapatkan.

Buat instance objek CloudHSM baru sebagai berikut: KeyStore

```
ks.load(null, null);
```

Tulis data penyimpanan kunci ke file menggunakan metode store. Sejak saat itu, Anda dapat memuat penyimpanan kunci yang ada menggunakan metode load dengan file sumber dan kata sandi sebagai berikut:

```
ks.load(inputStream, password);
```

Gunakan AWS CloudHSM KeyStore untuk AWS CloudHSM Client SDK 3

[Objek KeyStore CloudHSM umumnya digunakan melalui aplikasi pihak ketiga seperti jarsigner atau keytool](#). Anda juga dapat mengakses objek secara langsung dengan kode.

AWS CloudHSM KeyStore sesuai dengan KeyStore spesifikasi JCE [Class](#) dan menyediakan fungsi-fungsi berikut.

- **load**

Memuat penyimpanan kunci dari pengaliran input yang diberikan. Jika kata sandi ditetapkan saat menyimpan penyimpanan kunci, kata sandi yang sama ini harus disediakan agar pemuatan berhasil. Atur kedua parameter ke null untuk menginisialisasi sebuah penyimpanan kunci kosong baru.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
ks.load(inputStream, password);
```

- **aliases**

Mengembalikan penghitungan nama alias dari semua entri dalam contoh instans penyimpanan kunci yang diberikan. Hasil termasuk objek yang disimpan secara lokal dalam PKCS12 file dan objek yang tinggal di HSM.

Contoh kode:

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
for(Enumeration<String> entry = ks.aliases(); entry.hasMoreElements();)
{
    String label = entry.nextElement();
    System.out.println(label);
}
```

- **ContainsAlias**

Mengembalikan nilai true jika penyimpanan kunci memiliki akses ke setidaknya satu objek dengan alias yang ditentukan. Toko kunci memeriksa objek yang disimpan secara lokal dalam PKCS12 file dan objek yang tinggal di HSM.

- **DeleteEntry**

Menghapus entri sertifikat dari PKCS12 file lokal. Menghapus data kunci yang disimpan dalam HSM tidak didukung menggunakan file. AWS CloudHSM KeyStore Anda dapat menghapus kunci dengan alat [key_mgmt_util](#) CloudHSM.

- **GetCertificate**

Mengembalikan sertifikat yang terkait dengan alias jika tersedia. Jika alias tidak ada atau mereferensikan objek yang bukan sertifikat, fungsi mengembalikan NULL.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
Certificate cert = ks.getCertificate(alias)
```

- **GetCertificateAlias**

Mengembalikan nama (alias) dari entri penyimpanan kunci pertama yang datanya cocok dengan sertifikat yang diberikan.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
String alias = ks.getCertificateAlias(cert)
```

- **GetCertificateChain**

Mengembalikan rantai sertifikat yang terkait dengan alias yang diberikan. Jika alias tidak ada atau mereferensikan objek yang bukan sertifikat, fungsi mengembalikan NULL.

- **GetCreationDate**

Mengembalikan tanggal pembuatan entri yang diidentifikasi oleh alias yang diberikan. Jika tanggal pembuatan tidak tersedia, fungsi mengembalikan tanggal saat sertifikat menjadi valid.

- **GetKey**

GetKey diteruskan ke HSM dan mengembalikan objek kunci yang sesuai dengan label yang diberikan. Seperti yang getKey secara langsung menanyakan HSM, itu dapat digunakan untuk kunci apa pun pada HSM terlepas dari apakah itu dihasilkan oleh. KeyStore

```
Key key = ks.getKey(keyLabel, null);
```

- **IsCertificateEntry**

Memeriksa apakah entri dengan alias yang diberikan merupakan entri sertifikat.

- **IsKeyEntry**

Memeriksa apakah entri dengan alias yang diberikan merupakan entri kunci. Tindakan mencari PKCS12 file dan HSM untuk alias.

- `SetCertificateEntry`

Menetapkan sertifikat yang diberikan untuk alias yang diberikan. Jika alias yang diberikan sudah digunakan untuk mengidentifikasi kunci atau sertifikat, `KeyStoreException` dikeluarkan. Anda dapat menggunakan kode JCE untuk mendapatkan objek kunci dan kemudian menggunakan `KeyStore SetKeyEntry` metode untuk mengaitkan sertifikat ke kunci.

- `SetKeyEntry` dengan `byte[]` kunci

API ini saat ini tidak didukung dengan Client SDK 3.

- `SetKeyEntry` dengan objek `Key`

Menetapkan kunci yang diberikan untuk alias yang diberikan dan menyimpannya di dalam HSM. Jika objek `Key` bukan tipe `CaviumKey`, kunci diimpor ke HSM sebagai kunci sesi yang dapat diekstrak.

Jika objek `Key` adalah tipe `PrivateKey`, objek harus disertai dengan rantai sertifikat yang sesuai.

Jika alias sudah ada, panggilan `SetKeyEntry` mengeluarkan `KeyStoreException` dan mencegah kunci ditimpa. Jika kunci harus ditimpa, gunakan KMU atau JCE untuk tujuan itu.

- `EngineSize`

Mengembalikan jumlah entri dalam penyimpanan kunci.

- `Store`

Menyimpan penyimpanan kunci ke aliran output yang diberikan sebagai PKCS12 file dan mengamankannya dengan kata sandi yang diberikan. Selain itu, semua kunci dimuat tetap ada (yang ditetapkan menggunakan panggilan `setKey`).

API Kriptografi: Generasi Berikutnya (CNG) dan penyedia penyimpanan kunci (KSP) untuk AWS CloudHSM

AWS CloudHSM Klien untuk Windows termasuk penyedia CNG dan KSP.

Penyedia penyimpanan kunci (KSPs) mengaktifkan penyimpanan dan pengambilan kunci. Sebagai contoh, jika Anda menambahkan peran Microsoft Active Directory Certificate Services (AD CS) ke

server Windows dan memilih untuk membuat kunci privat baru untuk otoritas sertifikat (CA), Anda dapat memilih KSP yang akan mengelola penyimpanan kunci. Bila Anda mengonfigurasi peran AD CS, Anda dapat memilih KSP ini. Untuk informasi selengkapnya, lihat [Buat Windows Server CA](#).

API Kriptografi: Next Generation (CNG) adalah API kriptografi khusus untuk sistem operasi Microsoft Windows. CNG mengizinkan developer untuk menggunakan teknik kriptografi untuk mengamankan aplikasi berbasis Windows. Pada tingkat tinggi, AWS CloudHSM implementasi CNG menyediakan fungsionalitas berikut:

- Primitif Kriptografi — memungkinkan Anda melakukan operasi kriptografi mendasar.
- Kunci Impor dan Ekspor - memungkinkan Anda untuk mengimpor dan mengeksport kunci asimetris.
- Data Protection API (CNG DPAPI) — memungkinkan Anda untuk dengan mudah mengenkripsi dan mendekripsi data.
- Key Storage and Retrieval — memungkinkan Anda untuk menyimpan dan mengisolasi kunci pribadi dari symmetric key pair dengan aman.

Topik

- [Verifikasi Penyedia KSP dan CNG untuk AWS CloudHSM](#)
- [Prasyarat untuk menggunakan Klien Windows AWS CloudHSM](#)
- [Kaitkan AWS CloudHSM kunci dengan sertifikat](#)
- [Contoh kode untuk penyedia CNG untuk AWS CloudHSM](#)

Verifikasi Penyedia KSP dan CNG untuk AWS CloudHSM

Penyedia KSP dan CNG diinstal ketika Anda menginstal klien Windows AWS CloudHSM . Anda dapat menginstal klien dengan mengikuti langkah-langkah di [Instal klien \(Windows\)](#).

Gunakan bagian berikut untuk memverifikasi pemasangan penyedia.

Konfigurasi dan jalankan AWS CloudHSM klien Windows

Untuk memulai klien CloudHSM Windows, Anda harus terlebih dahulu memenuhi [Prasyarat](#). Kemudian, perbarui file konfigurasi yang digunakan penyedia dan mulai klien dengan menyelesaikan langkah-langkah di bawah ini. Anda perlu melakukan langkah-langkah ini saat pertama kali Anda menggunakan penyedia KSP dan CNG dan setelah Anda menambah atau menghapus HSMs di cluster Anda. Dengan cara AWS CloudHSM ini, mampu menyinkronkan data dan menjaga konsistensi HSMs di semua cluster.

Langkah 1: Hentikan AWS CloudHSM klien

Sebelum Anda memperbarui file konfigurasi yang digunakan penyedia, hentikan AWS CloudHSM klien. Jika klien sudah berhenti, menjalankan perintah stop tidak berpengaruh.

- Untuk klien Windows 1.1.2+:

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- Untuk klien Windows 1.1.1 dan yang lebih lama:

Gunakan Ctrl+C di jendela perintah tempat Anda memulai klien. AWS CloudHSM

Langkah 2: Perbarui file AWS CloudHSM konfigurasi

Langkah ini menggunakan -a parameter [alat Configure](#) untuk menambahkan alamat IP elastic network interface (ENI) dari salah satu HSMs di cluster ke file konfigurasi.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\configure.exe" -a <HSM ENI IP>
```

Untuk mendapatkan alamat IP ENI dari HSM di cluster Anda, navigasikan ke AWS CloudHSM konsol, pilih cluster, dan pilih cluster yang diinginkan. Anda juga dapat menggunakan [DescribeClusters](#) operasi, [perintah deskripsi-cluster](#), atau cmdlet. [Get-HSM2Cluster](#) PowerShell Ketik hanya satu alamat IP ENI. Apa pun alamat IP ENI yang Anda gunakan.

Langkah 3: Mulai AWS CloudHSM klien

Selanjutnya, mulai atau mulai ulang AWS CloudHSM klien. Ketika AWS CloudHSM klien mulai, ia menggunakan alamat IP ENI dalam file konfigurasi untuk query cluster. Kemudian menambahkan alamat IP ENI dari semua HSMs di cluster ke file informasi cluster.

- Untuk klien Windows 1.1.2+:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Untuk klien Windows 1.1.1 dan yang lebih lama:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Memeriksa penyedia KSP dan CNG

Anda dapat menggunakan salah satu dari perintah berikut untuk menentukan penyedia yang diinstal pada sistem Anda. Perintah daftar penyedia KSP dan CNG terdaftar. Parameter klien AWS CloudHSM tidak perlu berjalan.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\ksp_config.exe" -enum
```

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\cng_config.exe" -enum
```

Untuk memverifikasi bahwa penyedia KSP dan CNG diinstal pada instans EC2 Windows Server, Anda akan melihat entri berikut dalam daftar:

```
Cavium CNG Provider  
Cavium Key Storage Provider
```

Jika penyedia CNG hilang, jalankan perintah berikut.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\cng_config.exe" -register
```

Jika penyedia KSP hilang, jalankan perintah berikut.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\ksp_config.exe" -register
```

Prasyarat untuk menggunakan Klien Windows AWS CloudHSM

Sebelum Anda dapat memulai AWS CloudHSM klien Windows dan menggunakan penyedia KSP dan CNG, Anda harus mengatur kredensi login untuk HSM di sistem Anda. Anda dapat mengatur kredensial melalui Windows Credential Manager atau variabel lingkungan sistem. Kami sarankan Anda menggunakan Windows Credential Manager untuk menyimpan kredensial. Opsi ini tersedia dengan versi AWS CloudHSM klien 2.0.4 dan yang lebih baru. Menggunakan variabel lingkungan lebih mudah untuk mengatur, tapi kurang aman daripada menggunakan Windows Credential Manager.

Pengelola Kredensial Windows

Anda dapat menggunakan utilitas `set_cloudhsm_credentials` atau antarmuka Windows Credentials Manager.

- Menggunakan utilitas **set_cloudhsm_credentials**:

Utilitas `set_cloudhsm_credentials` disertakan dalam penginstal Windows Anda. Anda dapat menggunakan utilitas ini untuk dengan mudah melewati kredensial login HSM ke Windows Credential Manager. Jika Anda ingin mengompilasi utilitas ini dari sumber, Anda dapat menggunakan kode Python yang disertakan dalam installer.

1. Pergi ke folder `C:\Program Files\Amazon\CloudHSM\tools\`.
2. Jalankan file `set_cloudhsm_credentials.exe` dengan parameter nama pengguna dan kata sandi CU.

```
set_cloudhsm_credentials.exe --username <CU USER> --password <CU PASSWORD>
```

- Menggunakan antarmuka Credential Manager:

Anda dapat menggunakan antarmuka Credential Manager untuk mengelola kredensial Anda secara manual.

1. Untuk membuka Credential Manager, ketik `credential manager` di kotak pencarian pada bilah tugas dan pilih Credential Manager.
2. Pilih Kredensial Windows untuk mengelola kredensial Windows.
3. Pilih Tambah kredensial generik dan isi detail sebagai berikut:
 - Di Alamat Internet atau Jaringan, masukkan nama target sebagai `cloudhsm_client`.
 - Di Nama pengguna dan Kata Sandi, masukkan kredensial CU.
 - Klik OK.

Variabel lingkungan sistem

Anda dapat mengatur variabel lingkungan sistem yang mengidentifikasi HSM dan [pengguna kriptografi](#) (CU) untuk aplikasi Windows Anda. Anda dapat menggunakan [perintah setx](#) untuk mengatur variabel lingkungan sistem, atau mengatur variabel lingkungan sistem permanen [secara terprogram](#) atau di tab Lanjutan dari Panel Kontrol Windows Properti Sistem.

Warning

Ketika Anda mengatur kredensial melalui variabel lingkungan sistem, kata sandi tersedia dalam teks terang pada sistem pengguna. Untuk mengatasi masalah ini, gunakan Windows Credential Manager.

Atur variabel lingkungan sistem berikut:

n3fips_password=<CU USERNAME>:<CU PASSWORD>

Mengidentifikasi [pengguna krypto](#) (CU) di HSM dan menyediakan semua informasi login yang diperlukan. Aplikasi Anda mengautentikasi dan berjalan sebagai CU ini. Aplikasi memiliki izin CU ini dan dapat melihat dan mengelola hanya kunci yang CU miliki dan bagikan. Untuk membuat CU baru, gunakan [createUser](#). Untuk menemukan yang ada CUs, gunakan [ListUsers](#).

Contoh:

```
setx /m n3fips_password test_user:password123
```

Kaitkan AWS CloudHSM kunci dengan sertifikat

Sebelum Anda dapat menggunakan AWS CloudHSM kunci dengan alat pihak ketiga, seperti Microsoft [SignTool](#), Anda harus mengimpor metadata kunci ke toko sertifikat lokal dan mengaitkan metadata dengan sertifikat. Untuk mengimpor metadata kunci, gunakan utilitas `import_key.exe` yang termasuk dalam CloudHSM versi 3.0 dan lebih tinggi. Langkah-langkah berikut memberikan informasi tambahan, dan contoh output.

Langkah 1: Impor sertifikat Anda

Pada Windows, Anda akan dapat mengeklik dua kali sertifikat untuk mengimpornya ke penyimpanan sertifikat lokal Anda.

Namun, jika klik dua kali tidak bekerja, gunakan [alat Microsoft Certreq](#) untuk mengimpor sertifikat ke certificate manager. Sebagai contoh:

```
certreq -accept <certificatename>
```

Jika tindakan ini gagal dan Anda menerima kesalahan, `Key not found`, lanjutkan ke Langkah 2. Jika sertifikat muncul di penyimpanan kunci Anda, Anda telah menyelesaikan tugas dan tidak diperlukan tindakan lebih lanjut.

Langkah 2: Kumpulkan informasi pengenalan sertifikat

Jika langkah sebelumnya tidak berhasil, Anda harus mengaitkan kunci privat Anda dengan sertifikat. Namun, sebelum Anda dapat membuat kaitan, Anda harus terlebih dahulu menemukan Nama Kontainer Unik dan Nomor Seri sertifikat. Gunakan utilitas, `septicertutil`, untuk menampilkan

informasi sertifikat yang diperlukan. Output sampel berikut dari `certutil` menunjukkan nama wadah dan nomor seri.

```
===== Certificate 1 ===== Serial Number:
72000000047f7f7a9d41851b4e000000000004Issuer: CN=Enterprise-CANotBefore: 10/8/2019
11:50
AM NotAfter: 11/8/2020 12:00 PMSubject: CN=www.example.com, OU=Certificate
Management,
O=Information Technology, L=Seattle, S=Washington, C=USNon-root CertificateCert
Hash(sha1): 7f d8 5c 00 27 bf 37 74 3d 71 5b 54 4e c0 94 20 45 75 bc 65No key
provider
information Simple container name: CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c
Unique
container name: CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c
```

Langkah 3: Kaitkan kunci AWS CloudHSM pribadi dengan sertifikat

Untuk mengaitkan kunci dengan sertifikat, pertama-tama pastikan untuk [memulai daemon AWS CloudHSM klien](#). Kemudian, gunakan `import_key.exe` (yang termasuk dalam CloudHSM versi 3.0 dan lebih tinggi) untuk mengaitkan kunci privat dengan sertifikat. Saat menentukan sertifikat, gunakan nama kontainernya yang sederhana. Contoh berikut menunjukkan perintah dan respons. Tindakan ini hanya menyalin metadata kunci; kunci tetap pada HSM.

```
$> import_key.exe -RSA CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c

Successfully opened Microsoft Software Key Storage Provider : 0NCryptOpenKey failed :
80090016
```

Langkah 4: Perbarui penyimpanan sertifikat

Pastikan daemon AWS CloudHSM klien masih berjalan. Kemudian, gunakan `certutil` kata kerja, `repairstore`, untuk memperbarui nomor seri sertifikat. Sampel berikut menunjukkan perintah dan output. Lihat dokumentasi Microsoft untuk informasi tentang [repairstore kata kerja](#).

```
C:\Program Files\Amazon\CloudHSM>certutil -f -csp "Cavium Key Storage Provider"-
repairstore my "72000000047f7f7a9d41851b4e000000000004"
my "Personal"
===== Certificate 1 =====
Serial Number: 72000000047f7f7a9d41851b4e000000000004
Issuer: CN=Enterprise-CA
NotBefore: 10/8/2019 11:50 AM
```

```

NotAfter: 11/8/2020 12:00 PM
Subject: CN=www.example.com, OU=Certificate Management, O=Information Technology,
        L=Seattle, S=Washington, C=US
Non-root CertificateCert Hash(sha1): 7f d8 5c 00 27 bf 37 74 3d 71 5b 54 4e c0 94 20 45
        75 bc 65
SDK Version: 3.0
Key Container = CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c
Provider = "Cavium Key Storage Provider"
Private key is NOT exportableEncryption test passedCertUtil: -repairstore command
        completed successfully.

```

Setelah memperbarui nomor seri sertifikat, Anda dapat menggunakan sertifikat ini dan kunci AWS CloudHSM pribadi yang sesuai dengan alat penandatanganan pihak ketiga di Windows.

Contoh kode untuk penyedia CNG untuk AWS CloudHSM



**** Contoh kode saja - Tidak untuk penggunaan produksi****

Kode contoh hanya untuk tujuan ilustrasi. Jangan jalankan kode ini dalam produksi.

Contoh berikut menunjukkan cara menghitung penyedia kriptografi terdaftar pada sistem Anda untuk menemukan penyedia CNG yang diinstal dengan klien CloudHSM untuk Windows. Sampel juga menunjukkan cara membuat pasangan kunci asimetris dan bagaimana menggunakan pasangan kunci untuk menandatangani data.



Important

Sebelum Anda menjalankan contoh ini, Anda harus mengatur kredensial HSM seperti yang dijelaskan dalam prasyarat. Untuk detailnya, lihat [Prasyarat untuk menggunakan Klien Windows AWS CloudHSM](#).

```

// CloudHsmCngExampleConsole.cpp : Console application that demonstrates CNG
// capabilities.
// This example contains the following functions.
//

```

```

// VerifyProvider()          - Enumerate the registered providers and retrieve Cavium
// KSP and CNG providers.
// GenerateKeyPair()         - Create an RSA key pair.
// SignData()                - Sign and verify data.
//

#include "stdafx.h"
#include <Windows.h>

#ifdef NT_SUCCESS
#define NT_SUCCESS(Status) ((NTSTATUS)(Status) >= 0)
#endif

#define CAVIUM_CNG_PROVIDER L"Cavium CNG Provider"
#define CAVIUM_KEYSTORE_PROVIDER L"Cavium Key Storage Provider"

// Enumerate the registered providers and determine whether the Cavium CNG provider
// and the Cavium KSP provider exist.
//
bool VerifyProvider()
{
    NTSTATUS status;
    ULONG cbBuffer = 0;
    PCRYPT_PROVIDERS pBuffer = NULL;
    bool foundCng = false;
    bool foundKeystore = false;

    // Retrieve information about the registered providers.
    // cbBuffer - the size, in bytes, of the buffer pointed to by pBuffer.
    // pBuffer - pointer to a buffer that contains a CRYPT_PROVIDERS structure.
    status = BCryptEnumRegisteredProviders(&cbBuffer, &pBuffer);

    // If registered providers exist, enumerate them and determine whether the
    // Cavium CNG provider and Cavium KSP provider have been registered.
    if (NT_SUCCESS(status))
    {
        if (pBuffer != NULL)
        {
            for (ULONG i = 0; i < pBuffer->cProviders; i++)
            {
                // Determine whether the Cavium CNG provider exists.
                if (wcscmp(CAVIUM_CNG_PROVIDER, pBuffer->rgpszProviders[i]) == 0)
                {
                    printf("Found %S\n", CAVIUM_CNG_PROVIDER);
                }
            }
        }
    }
}

```

```

        foundCng = true;
    }

    // Determine whether the Cavium KSP provider exists.
    else if (wcscmp(CAVIUM_KEYSTORE_PROVIDER, pBuffer->rgpszProviders[i]) == 0)
    {
        printf("Found %S\n", CAVIUM_KEYSTORE_PROVIDER);
        foundKeystore = true;
    }
}
}
else
{
    printf("BCryptEnumRegisteredProviders failed with error code 0x%08x\n", status);
}

// Free memory allocated for the CRYPT_PROVIDERS structure.
if (NULL != pBuffer)
{
    BCryptFreeBuffer(pBuffer);
}

return foundCng == foundKeystore == true;
}

// Generate an asymmetric key pair. As used here, this example generates an RSA key
pair
// and returns a handle. The handle is used in subsequent operations that use the key
pair.
// The key material is not available.
//
// The key pair is used in the SignData function.
//
NTSTATUS GenerateKeyPair(BCRYPT_ALG_HANDLE hAlgorithm, BCRYPT_KEY_HANDLE *hKey)
{
    NTSTATUS status;

    // Generate the key pair.
    status = BCryptGenerateKeyPair(hAlgorithm, hKey, 2048, 0);
    if (!NT_SUCCESS(status))
    {
        printf("BCryptGenerateKeyPair failed with code 0x%08x\n", status);
        return status;
    }
}

```

```
}

// Finalize the key pair. The public/private key pair cannot be used until this
// function is called.
status = BCryptFinalizeKeyPair(*hKey, 0);
if (!NT_SUCCESS(status))
{
    printf("BCryptFinalizeKeyPair failed with code 0x%08x\n", status);
    return status;
}

return status;
}

// Sign and verify data using the RSA key pair. The data in this function is hardcoded
// and is for example purposes only.
//
NTSTATUS SignData(BCRYPT_KEY_HANDLE hKey)
{
    NTSTATUS status;
    PBYTE sig;
    ULONG sigLen;
    ULONG resLen;
    BCRYPT_PKCS1_PADDING_INFO pInfo;

    // Hardcode the data to be signed (for demonstration purposes only).
    PBYTE message = (PBYTE)"d83e7716bed8a20343d8dc6845e57447";
    ULONG messageLen = strlen((char*)message);

    // Retrieve the size of the buffer needed for the signature.
    status = BCryptSignHash(hKey, NULL, message, messageLen, NULL, 0, &sigLen, 0);
    if (!NT_SUCCESS(status))
    {
        printf("BCryptSignHash failed with code 0x%08x\n", status);
        return status;
    }

    // Allocate a buffer for the signature.
    sig = (PBYTE)HeapAlloc(GetProcessHeap(), 0, sigLen);
    if (sig == NULL)
    {
        return -1;
    }
}
```

```
// Use the SHA256 algorithm to create padding information.
pInfo.pszAlgId = BCRYPT_SHA256_ALGORITHM;

// Create a signature.
status = BCryptSignHash(hKey, &pInfo, message, messageLen, sig, sigLen, &resLen,
BCRYPT_PAD_PKCS1);
if (!NT_SUCCESS(status))
{
    printf("BCryptSignHash failed with code 0x%08x\n", status);
    return status;
}

// Verify the signature.
status = BCryptVerifySignature(hKey, &pInfo, message, messageLen, sig, sigLen,
BCRYPT_PAD_PKCS1);
if (!NT_SUCCESS(status))
{
    printf("BCryptVerifySignature failed with code 0x%08x\n", status);
    return status;
}

// Free the memory allocated for the signature.
if (sig != NULL)
{
    HeapFree(GetProcessHeap(), 0, sig);
    sig = NULL;
}

return 0;
}

// Main function.
//
int main()
{
    NTSTATUS status;
    BCRYPT_ALG_HANDLE hRsaAlg;
    BCRYPT_KEY_HANDLE hKey = NULL;

    // Enumerate the registered providers.
    printf("Searching for Cavium providers...\n");
    if (VerifyProvider() == false) {
        printf("Could not find the CNG and Keystore providers\n");
        return 1;
    }
}
```

```
}

// Get the RSA algorithm provider from the Cavium CNG provider.
printf("Opening RSA algorithm\n");
status = BCryptOpenAlgorithmProvider(&hRsaAlg, BCRYPT_RSA_ALGORITHM,
CAVIUM_CNG_PROVIDER, 0);
if (!NT_SUCCESS(status))
{
    printf("BCryptOpenAlgorithmProvider RSA failed with code 0x%08x\n", status);
    return status;
}

// Generate an asymmetric key pair using the RSA algorithm.
printf("Generating RSA Keypair\n");
GenerateKeyPair(hRsaAlg, &hKey);
if (hKey == NULL)
{
    printf("Invalid key handle returned\n");
    return 0;
}
printf("Done!\n");

// Sign and verify [hardcoded] data using the RSA key pair.
printf("Sign/Verify data with key\n");
SignData(hKey);
printf("Done!\n");

// Remove the key handle from memory.
status = BCryptDestroyKey(hKey);
if (!NT_SUCCESS(status))
{
    printf("BCryptDestroyKey failed with code 0x%08x\n", status);
    return status;
}

// Close the RSA algorithm provider.
status = BCryptCloseAlgorithmProvider(hRsaAlg, NULL);
if (!NT_SUCCESS(status))
{
    printf("BCryptCloseAlgorithmProvider RSA failed with code 0x%08x\n", status);
    return status;
}

return 0;
```



```
}
```

Mengintegrasikan aplikasi pihak ketiga dengan AWS CloudHSM

Beberapa [kasus penggunaan](#) untuk AWS CloudHSM melibatkan mengintegrasikan aplikasi perangkat lunak pihak ketiga dengan HSM di cluster Anda AWS CloudHSM. Dengan mengintegrasikan perangkat lunak pihak ketiga AWS CloudHSM, Anda dapat mencapai berbagai tujuan terkait keamanan. Topik berikut menjelaskan cara untuk mencapai beberapa tujuan ini.

Topik

- [Tingkatkan keamanan server web Anda dengan SSL/TLS offload di AWS CloudHSM](#)
- [Konfigurasi Windows Server sebagai otoritas sertifikat \(CA\) dengan AWS CloudHSM](#)
- [Enkripsi data transparan \(TDE\) database Oracle dengan AWS CloudHSM](#)
- [Gunakan Microsoft SignTool AWS CloudHSM untuk menandatangani file](#)
- [Java Keytool dan integrasi Jarsigner dengan AWS CloudHSM](#)
- [Gunakan Microsoft Manifest Generation dan Editing Tool \(Mage.exe\) AWS CloudHSM untuk menandatangani file](#)
- [Integrasi vendor pihak ketiga lainnya dengan AWS CloudHSM](#)

Tingkatkan keamanan server web Anda dengan SSL/TLS offload di AWS CloudHSM

Server web dan klien mereka (browser web) dapat menggunakan protokol Secure Sockets Layer (SSL) atau Transport Layer Security (TLS) untuk mengkonfirmasi identitas server web dan membuat koneksi aman yang mengirim dan menerima halaman web atau data lain melalui internet. Hal ini umumnya dikenal sebagai HTTPS. Server web menggunakan public-private key pair dan SSL/TLS public key certificate untuk membuat sesi HTTPS dengan setiap klien. Proses ini melibatkan banyak perhitungan untuk server web, tetapi Anda dapat menurunkan beberapa ini ke AWS CloudHSM cluster Anda, yang disebut sebagai akselerasi SSL. Pembongkaran mengurangi beban komputasi pada server web Anda dan memberikan keamanan ekstra dengan menyimpan kunci pribadi server. HSMs

Topik berikut memberikan ikhtisar tentang bagaimana SSL/TLS offload dengan AWS CloudHSM karya dan tutorial untuk mengatur SSL/TLS offload dengan AWS CloudHSM pada platform berikut.

Untuk Linux, gunakan OpenSSL Dynamic Engine pada perangkat lunak server web NGINX atau Apache HTTP Server

Untuk Windows, gunakan Internet Information Services (IIS) untuk perangkat lunak server web Windows Server

Topik

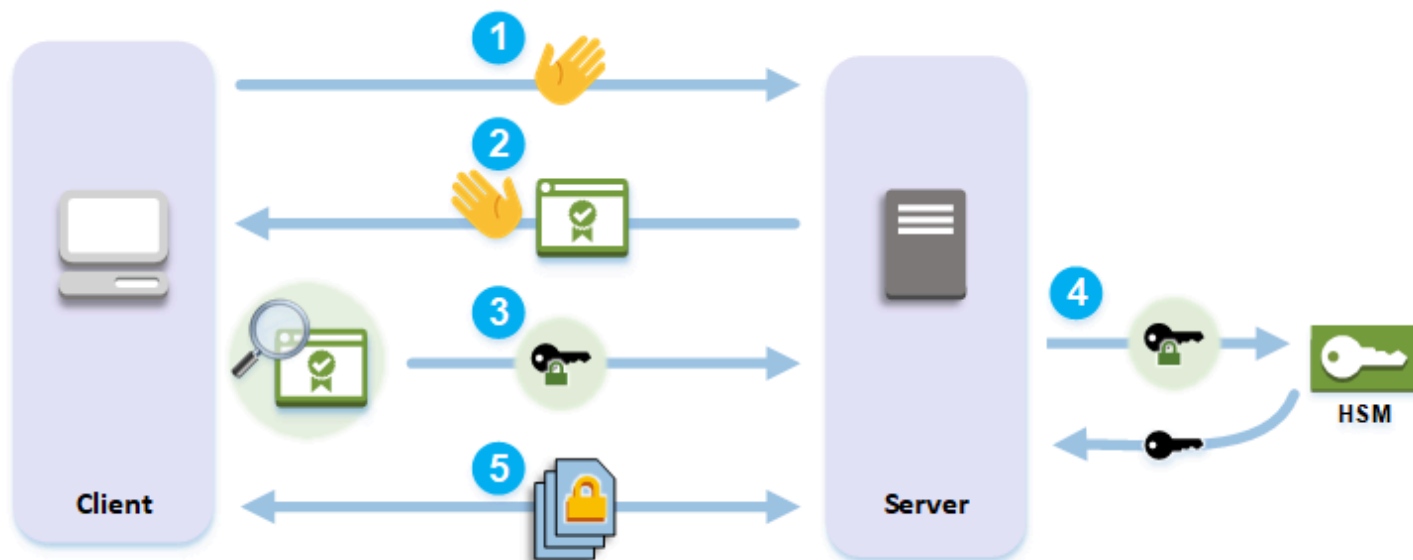
- Bagaimana SSL/TLS offload dengan bekerja AWS CloudHSM
- AWS CloudHSM SSL/TLS offload di Linux menggunakan NGINX atau Apache dengan OpenSSL
- AWS CloudHSM SSL/TLS offload di Linux menggunakan NGINX atau dengan OpenSSL Provider HAProxy
- AWS CloudHSM SSL/TLS offload di Linux menggunakan Tomcat dengan JSSE
- AWS CloudHSM SSL/TLS offload pada Windows menggunakan IIS dengan KSP
- Tambahkan penyeimbang beban dengan Elastic Load Balancing AWS CloudHSM untuk (opsional)

Bagaimana SSL/TLS offload dengan bekerja AWS CloudHSM

Untuk membuat sambungan HTTPS, server web Anda melakukan proses jabat tangan dengan klien. Sebagai bagian dari proses ini, server membongkar beberapa pemrosesan kriptografi ke HSMs dalam AWS CloudHSM cluster, seperti yang ditunjukkan pada gambar berikut. Setiap langkah proses dijelaskan di bawah gambar.

Note

Gambar dan proses berikut mengasumsikan bahwa RSA digunakan untuk verifikasi server dan pertukaran kunci. Proses ini sedikit berbeda ketika Diffie-Hellman digunakan sebagai pengganti RSA.



1. Klien mengirimkan pesan halo ke server.
2. Server merespon dengan pesan halo dan mengirimkan sertifikat server.
3. Klien akan melakukan tindakan berikut:
 - a. Memverifikasi bahwa sertifikat SSL/TLS server ditandatangani oleh sertifikat root yang dipercaya klien.
 - b. Ekstraksi kunci publik dari sertifikat.
 - c. Menghasilkan rahasia pra-master dan mengenkripsi dengan kunci publik server.
 - d. Mengirim rahasia pra-master terenkripsi ke server.
4. Untuk mendekripsi rahasia pra-master klien, server mengirimkannya ke HSM. HSM menggunakan kunci pribadi di HSM untuk mendekripsi rahasia pra-master dan kemudian mengirimkan rahasia pra-master ke server. Secara independen, klien dan server masing-masing menggunakan rahasia pra-master dan beberapa informasi dari pesan halo untuk menghitung rahasia utama.
5. Proses jabat tangan berakhir. Untuk sisa sesi, semua pesan yang dikirim antara klien dan server dienkripsi dengan turunan dari rahasia master.

Untuk mempelajari cara mengonfigurasi SSL/TLS offload dengan AWS CloudHSM, lihat salah satu topik berikut:

- [AWS CloudHSM SSL/TLS offload di Linux menggunakan NGINX atau Apache dengan OpenSSL](#)
- [AWS CloudHSM SSL/TLS offload di Linux menggunakan Tomcat dengan JSSE](#)
- [AWS CloudHSM SSL/TLS offload pada Windows menggunakan IIS dengan KSP](#)

AWS CloudHSM SSL/TLS offload di Linux menggunakan NGINX atau Apache dengan OpenSSL

Topik ini memberikan step-by-step instruksi untuk mengatur SSL/TLS offload dengan AWS CloudHSM pada server web Linux.

Topik

- [Ikhtisar](#)
- [Langkah 1: Siapkan prasyarat](#)
- [Langkah 2: Hasilkan kunci pribadi dan SSL/TLS sertifikat](#)
- [Langkah 3: Konfigurasi server web](#)
- [Langkah 4: Aktifkan lalu lintas HTTPS dan verifikasi sertifikat](#)

Ikhtisar

Di Linux, perangkat lunak server web [NGINX](#) dan [Apache HTTP Server](#) terintegrasi dengan [OpenSSL](#) untuk mendukung HTTPS. [Mesin AWS CloudHSM dinamis untuk OpenSSL](#) menyediakan antarmuka yang memungkinkan perangkat lunak server web untuk menggunakan di cluster Anda untuk HSMs pembongkaran kriptografi dan penyimpanan kunci. Mesin OpenSSL adalah jembatan yang menghubungkan server web ke kluster AWS CloudHSM .

Untuk menyelesaikan tutorial ini, Anda harus terlebih dahulu memilih apakah akan menggunakan perangkat lunak server web NGINX atau Apache di Linux. Kemudian, tutorial menunjukkan cara melakukan hal berikut:

- Instal perangkat lunak server web pada instans Amazon EC2.
- Atur konfigurasi perangkat lunak server web untuk mendukung HTTPS dengan kunci privat yang tersimpan di kluster AWS CloudHSM .
- (Opsional) Gunakan Amazon EC2 untuk membuat instans server web kedua dan Elastic Load Balancing untuk membuat penyeimbang beban. Menggunakan penyeimbang beban dapat meningkatkan performa dengan mendistribusikan beban di beberapa server. Hal ini juga dapat memberikan redundansi dan ketersediaan yang lebih tinggi jika satu atau lebih server gagal.

Saat Anda siap memulai, buka [Langkah 1: Siapkan prasyarat](#).

Langkah 1: Siapkan prasyarat

Platform yang berbeda memerlukan prasyarat yang berbeda. Gunakan bagian prasyarat di bawah ini yang sesuai dengan platform Anda.

Prasyarat untuk Klien SDK 5

Untuk mengatur SSL/TLS offload server web dengan Client SDK 5, Anda memerlukan yang berikut ini:

- AWS CloudHSM Cluster aktif dengan setidaknya dua modul keamanan perangkat keras (HSM)

Note

Anda dapat menggunakan satu HSM klaster, tetapi Anda harus terlebih dahulu menonaktifkan daya tahan kunci klien. Untuk informasi selengkapnya, lihat [Kelola Pengaturan Daya Tahan Kunci Klien](#) dan [Alat Konfigurasi Klien SDK 5](#).

- Instans Amazon EC2 menjalankan sistem operasi Linux dengan perangkat lunak berikut diinstal:
 - Sebuah server web (baik NGINX atau Apache)
 - OpenSSL Dynamic Engine untuk Klien SDK 5
- [Pengguna kriptografi](#) (CU) harus memiliki dan mengelola kunci privat server web pada HSM.

Untuk mengatur sebuah instans server web Linux dan membuat CU pada HSM

1. Instal dan konfigurasi OpenSSL Dynamic Engine untuk AWS CloudHSM Untuk informasi selengkapnya tentang menginstal OpenSSL Dynamic Engine, lihat [OpenSSL Dynamic Engine untuk Klien SDK 5](#).
2. Pada instans EC2 Linux yang memiliki akses ke cluster Anda, instal server web NGINX atau Apache:

Amazon Linux 2

- Untuk informasi tentang cara mengunduh versi terbaru NGINX di Amazon Linux 2, lihat situs web [NGINX](#).

Versi terbaru NGINX yang tersedia untuk Amazon Linux 2 menggunakan versi OpenSSL yang lebih baru dari versi sistem OpenSSL. Setelah menginstal NGINX, Anda perlu

membuat tautan simbolis dari pustaka OpenSSL Dynamic Engine ke lokasi yang diharapkan oleh versi AWS CloudHSM OpenSSL ini

```
$ sudo ln -sf /opt/cloudhsm/lib/libcloudhsm_openssl_engine.so /usr/lib64/engines-1.1/cloudhsm.so
```

- Apache

```
$ sudo yum install httpd mod_ssl
```

Amazon Linux 2023

- NGINX

```
$ sudo yum install nginx
```

- Apache

```
$ sudo yum install httpd mod_ssl
```

Red Hat 8

- NGINX

```
$ sudo yum install nginx
```

- Apache

```
$ sudo yum install httpd mod_ssl
```

Red Hat 9

- NGINX

```
$ sudo yum install nginx
```

- Apache

```
$ sudo yum install httpd mod_ssl
```

Ubuntu 22.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

Ubuntu 24.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

3. [Gunakan CloudHSM CLI untuk membuat pengguna kriptografi](#). Untuk informasi selengkapnya tentang mengelola pengguna HSM, lihat [Mengelola pengguna HSM dengan CloudHSM CLI](#).

Tip

Lacak nama pengguna dan kata sandi CU. Anda akan membutuhkannya nanti ketika Anda membuat atau mengimpor kunci privat HTTPS dan sertifikat untuk server web Anda.

Setelah Anda menyelesaikan langkah ini, buka [Langkah 2: Hasilkan kunci pribadi dan SSL/TLS sertifikat](#).

Catatan

- Untuk menggunakan Security-Enhanced Linux (SELinux) dan server web, Anda harus mengizinkan koneksi TCP keluar pada port 2223, yang merupakan port yang digunakan Client SDK 5 untuk berkomunikasi dengan HSM.
- Untuk membuat dan mengaktifkan klaster dan memberikan akses instans EC2 ke klaster, selesaikan langkah-langkah dalam [Memulai dengan AWS CloudHSM](#). Memulai menawarkan step-by-step instruksi untuk membuat cluster aktif dengan satu HSM dan instans klien Amazon EC2. Anda dapat menggunakan instans klien ini sebagai server web Anda.
- Untuk menghindari menonaktifkan daya tahan kunci klien, tambahkan lebih dari satu HSM ke klaster Anda. Untuk informasi lebih lanjut, lihat [Menambahkan HSM ke cluster AWS CloudHSM](#).
- Untuk terhubung ke instans klien Anda, Anda dapat menggunakan SSH atau PuTTY. Untuk informasi selengkapnya, lihat [Menghubungkan ke Instans Linux Anda Menggunakan SSH](#) atau [Menyambung ke Instans Linux Anda dari Windows Menggunakan PuTTY](#) dalam dokumentasi Amazon EC2.

Langkah 2: Hasilkan kunci pribadi dan SSL/TLS sertifikat

Untuk mengaktifkan HTTPS, aplikasi server web Anda (NGINX atau Apache) memerlukan kunci pribadi dan sertifikat yang sesuai. SSL/TLS Untuk menggunakan server web SSL/TLS offload dengan AWS CloudHSM, Anda harus menyimpan kunci pribadi di HSM di cluster Anda AWS CloudHSM. Pertama-tama Anda akan membuat kunci pribadi dan menggunakan kunci untuk membuat permintaan penandatanganan sertifikat (CSR). Anda kemudian mengekspor kunci pribadi PEM palsu dari HSM, yang merupakan file kunci pribadi dalam format PEM yang berisi referensi ke kunci pribadi yang disimpan di HSM (itu bukan kunci pribadi yang sebenarnya). Server web Anda menggunakan file kunci pribadi PEM palsu untuk mengidentifikasi kunci pribadi pada HSM selama SSL/TLS pembongkaran.

Menghasilkan kunci pribadi dan sertifikat

Menghasilkan kunci pribadi

[Bagian ini menunjukkan cara membuat keypair menggunakan CloudHSM CLI](#). Setelah Anda memiliki key pair yang dihasilkan di dalam HSM, Anda dapat mengekspornya sebagai file PEM palsu dan menghasilkan sertifikat yang sesuai.

Instal dan konfigurasi CloudHSM CLI

1. [Instal dan Konfigurasi](#) CloudHSM CLI.
2. Gunakan perintah berikut untuk memulai CloudHSM CLI.

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

3. Jalankan perintah berikut untuk masuk ke HSM. Ganti *<user name>* dengan nama pengguna pengguna krypto Anda

```
Command: login --username <user name> --role crypto-user
```

Menghasilkan Kunci Pribadi

Tergantung pada kasus penggunaan Anda, Anda dapat menghasilkan RSA atau EC key pair. Lakukan salah satu tindakan berikut:

- Untuk menghasilkan kunci pribadi RSA pada HSM

Gunakan [key generate-asymmetric-pair rsa](#) perintah untuk menghasilkan key pair RSA. Contoh ini menghasilkan key pair RSA dengan modulus 2048, eksponen publik 65537, label kunci publik, dan label kunci pribadi dari *tls_rsa_pub*. *tls_rsa_private*

```
aws-cloudhsm > key generate-asymmetric-pair rsa \  
--public-exponent 65537 \  
--modulus-size-bits 2048 \  
--public-label tls_rsa_pub \  
--private-label tls_rsa_private \  
--private-attributes sign=true  
{  
  "error_code": 0,  
  "data": {  
    "public_key": {  
      "key-reference": "0x0000000000280cc8",  
      "key-info": {  
        "key-owners": [  
          {  
            "username": "cu1",  
            "key-coverage": "full"  
          }  
        ],  
      },  
    },  
  },  
}
```

```

    "shared-users": [],
    "cluster-coverage": "full"
  },
  "attributes": {
    "key-type": "rsa",
    "label": "tls_rsa_pub",
    "id": "",
    "check-value": "0x01fe6e",
    "class": "public-key",
    "encrypt": true,
    "decrypt": false,
    "token": true,
    "always-sensitive": false,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": false,
    "sign": false,
    "trusted": false,
    "unwrap": false,
    "verify": false,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 512,
    "public-exponent": "0x010001",
    "modulus":
      "0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1
73a80fdb457aa7b20cd61e486c326e2cfd5e124a7f6a996437437812b542e3caf85928aa866f0298580f7967ee6aa
f6e6296d6c116d5744c6d60d14d3bf3cb978fe6b75ac67b7089bafd50d8687213b31abc7dc1bad422780d29c851d5
133022653225bd129f8491101725e9ea33e1ded83fb57af35f847e532eb30cd7e726f23910d2671c6364092e83469
ac3160f0ca9725d38318b7",
    "modulus-size-bits": 2048
  }
},
  "private_key": {
    "key-reference": "0x00000000000280cc7",
    "key-info": {
      "key-owners": [
        {
          "username": "cu1",

```

```

        "key-coverage": "full"
    },
    ],
    "shared-users": [],
    "cluster-coverage": "full"
},
"attributes": {
    "key-type": "rsa",
    "label": "tls_rsa_private",
    "id": "",
    "check-value": "0x01fe6e",
    "class": "private-key",
    "encrypt": false,
    "decrypt": true,
    "token": true,
    "always-sensitive": true,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": false,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 1217,
    "public-exponent": "0x010001",
    "modulus":
"0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1
    "modulus-size-bits": 2048
}
}
}
}

```

- Untuk menghasilkan kunci pribadi EC pada HSM

Gunakan [key generate-asymmetric-pair ec](#) perintah untuk menghasilkan EC key pair. Contoh ini menghasilkan key pair EC dengan prime256v1 kurva (sesuai dengan

NID_X9_62_prime256v1 kurva), label kunci publik *tls_ec_pub*, dan label kunci pribadi dari *tls_ec_private*.

```
aws-cloudhsm > key generate-asymmetric-pair ec \
  --curve prime256v1 \
  --public-label tls_ec_pub \
  --private-label tls_ec_private \
  --private-attributes sign=true
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x00000000000012000b",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "ec",
        "label": "tls_ec_pub",
        "id": "",
        "check-value": "0xd7c1a7",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
```

```

        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 57,
        "ec-point":
"0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f3
        "curve": "secp224r1"
    },
    },
    "private_key": {
        "key-reference": "0x0000000000012000c",
        "key-info": {
            "key-owners": [
                {
                    "username": "cu1",
                    "key-coverage": "full"
                }
            ],
            "shared-users": [],
            "cluster-coverage": "session"
        },
        "attributes": {
            "key-type": "ec",
            "label": "tls_ec_private",
            "id": "",
            "check-value": "0xd7c1a7",
            "class": "private-key",
            "encrypt": false,
            "decrypt": false,
            "token": false,
            "always-sensitive": true,
            "derive": false,
            "destroyable": true,
            "extractable": true,
            "local": true,
            "modifiable": true,
            "never-extractable": false,
            "private": true,
            "sensitive": true,
            "sign": true,
            "trusted": false,
            "unwrap": false,
            "verify": false,

```

```

        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 122,
        "ec-point":
"0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f3
        "curve": "secp224r1"
    }
}
}
}

```

Ekspor file kunci pribadi PEM palsu

Setelah Anda memiliki kunci pribadi di HSM, Anda harus mengekspor file kunci pribadi PEM palsu. File ini tidak berisi data kunci yang sebenarnya, tetapi memungkinkan OpenSSL Dynamic Engine untuk mengidentifikasi kunci pribadi pada HSM. Anda kemudian dapat menggunakan kunci pribadi untuk membuat permintaan penandatanganan sertifikat (CSR) dan menandatangani CSR untuk membuat sertifikat.

Gunakan [key generate-file](#) perintah untuk mengekspor kunci pribadi dalam format PEM palsu dan menyimpannya ke file. Ganti nilai berikut dengan nilai Anda sendiri:

- **<private_key_label>**— Label kunci pribadi yang Anda buat pada langkah sebelumnya.
- **<web_server_fake_pem.key>**— Nama file tempat kunci PEM palsu Anda akan ditulis.

```

aws-cloudhsm > key generate-file --encoding reference-pem --
path <web_server_fake_pem.key> --filter attr.label=<private_key_label>
{
  "error_code": 0,
  "data": {
    "message": "Successfully generated key file"
  }
}

```

Keluar dari CloudHSM CLI

Jalankan perintah berikut untuk menghentikan CloudHSM CLI.

```
aws-cloudhsm > quit
```

Anda sekarang harus memiliki file baru di sistem Anda, yang terletak di jalur yang ditentukan oleh `<web_server_fake_pem.key>` dalam perintah sebelumnya. File ini adalah file kunci pribadi PEM palsu.

Menghasilkan sertifikat yang ditandatangani sendiri

Setelah Anda membuat kunci pribadi PEM palsu, Anda dapat menggunakan file ini untuk menghasilkan permintaan penandatanganan sertifikat (CSR) dan sertifikat.

Di lingkungan produksi, Anda biasanya menggunakan sertifikat otoritas (CA) untuk membuat sertifikat dari CSR. CA tidak diperlukan untuk lingkungan pengujian. Jika Anda menggunakan CA, kirim file CSR kepada mereka dan gunakan SSL/TLS sertifikat bertanda tangan yang mereka berikan kepada Anda di server web Anda untuk HTTPS.

Sebagai alternatif untuk menggunakan CA, Anda dapat menggunakan AWS CloudHSM OpenSSL Dynamic Engine untuk membuat sertifikat yang ditandatangani sendiri. Sertifikat yang ditandatangani sendiri tidak dipercaya oleh peramban dan tidak boleh digunakan dalam lingkungan produksi. Sertifikat dapat digunakan dalam lingkungan pengujian.

Warning

Sertifikat yang ditandatangani sendiri hanya boleh digunakan dalam lingkungan pengujian. Untuk lingkungan produksi, gunakan metode yang lebih aman seperti otoritas sertifikat untuk membuat sertifikat.

Instal dan konfigurasi OpenSSL Dynamic Engine

1. Hubungkan ke instans klien Anda.
2. [the section called “Menginstal”](#)

Menghasilkan sertifikat

1. Dapatkan salinan file PEM palsu Anda yang dihasilkan pada langkah sebelumnya.
2. Buat CSR

Jalankan perintah berikut untuk menggunakan AWS CloudHSM OpenSSL Dynamic Engine untuk membuat permintaan penandatanganan sertifikat (CSR). Ganti

`<web_server_fake_pem.key>` dengan nama file yang berisi kunci pribadi PEM palsu Anda. Ganti `<web_server.csr>` dengan nama file yang berisi CSR Anda.

Perintah `req` bersifat interaktif. Tanggapi setiap bidang. Informasi bidang disalin ke SSL/TLS sertifikat Anda.

```
$ openssl req -engine cloudhsm -new -key <web_server_fake_pem.key> -  
out <web_server.csr>
```

3. Buat sertifikat yang ditandatangani sendiri

Jalankan perintah berikut untuk menggunakan AWS CloudHSM OpenSSL Dynamic Engine untuk menandatangani CSR Anda dengan kunci pribadi Anda di HSM Anda. Ini membuat sertifikat yang ditandatangani sendiri. Ganti nilai berikut dalam perintah dengan nilai Anda sendiri.

- `<web_server.csr>`— Nama file yang berisi CSR.
- `<web_server_fake_pem.key>`— Nama file yang berisi kunci pribadi PEM palsu.
- `<web_server.crt>`— Nama file yang akan berisi sertifikat server web Anda.

```
$ openssl x509 -engine cloudhsm -req -days 365 -in <web_server.csr> -  
signkey <web_server_fake_pem.key> -out <web_server.crt>
```

Setelah Anda menyelesaikan langkah ini, buka [Langkah 3: Konfigurasi server web](#).

Langkah 3: Konfigurasi server web

Perbarui konfigurasi perangkat lunak server web Anda untuk menggunakan sertifikat HTTPS dan kunci privat PEM palsu yang Anda buat di [langkah sebelumnya](#). Ingatlah untuk mencadangkan sertifikat dan kunci yang sudah ada sebelum memulai. Ini akan menyelesaikan pengaturan perangkat lunak server web Linux Anda untuk SSL/TLS dibongkar dengan AWS CloudHSM.

Menyelesaikan langkah-langkah dari salah satu bagian berikut.

Topik

- [Konfigurasi server web NGINX](#)
- [Konfigurasi server web Apache](#)

Konfigurasi server web NGINX

Gunakan bagian ini untuk mengatur konfigurasi NGINX pada platform yang didukung.

Untuk memperbarui konfigurasi server web untuk NGINX

1. Hubungkan ke instans klien Anda.
2. Jalankan perintah berikut untuk membuat direktori yang diperlukan untuk sertifikat server web dan kunci privat PEM palsu.

```
$ sudo mkdir -p /etc/pki/nginx/private
```

3. Jalankan perintah berikut untuk menyalin sertifikat server web Anda ke lokasi yang diperlukan. Ganti `<web_server.crt>` dengan nama sertifikat server web Anda.

```
$ sudo cp <web_server.crt> /etc/pki/nginx/server.crt
```

4. Jalankan perintah berikut untuk menyalin kunci privat PEM palsu Anda ke lokasi yang diperlukan. Ganti `<web_server_fake_pem.key>` dengan nama file yang berisi kunci pribadi PEM palsu Anda.

```
$ sudo cp <web_server_fake_pem.key> /etc/pki/nginx/private/server.key
```

5. Jalankan perintah berikut untuk mengubah kepemilikan file sehingga pengguna bernama nginx dapat membacanya.

```
$ sudo chown nginx /etc/pki/nginx/server.crt /etc/pki/nginx/private/server.key
```

6. Jalankan perintah berikut untuk mencadangkan file `/etc/nginx/nginx.conf`.

```
$ sudo cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.backup
```

7. Perbarui konfigurasi NGINX.

Note

Setiap kluster dapat mendukung maksimum 1000 proses pekerja NGINX di semua server web NGINX.

Amazon Linux 2

Gunakan editor teks untuk mengedit file `/etc/nginx/nginx.conf`. Ini memerlukan izin root Linux. Di bagian atas file, tambahkan baris berikut:

```
ssl_engine cloudhsm;  
env CLOUDHSM_PIN;
```

Kemudian tambahkan yang berikut ini ke bagian TLS file:

```
# Settings for a TLS enabled server.  
server {  
    listen      443 ssl http2 default_server;  
    listen      [::]:443 ssl http2 default_server;  
    server_name _;  
    root        /usr/share/nginx/html;  
  
    ssl_certificate "/etc/pki/nginx/server.crt";  
    ssl_certificate_key "/etc/pki/nginx/private/server.key";  
    # It is strongly recommended to generate unique DH parameters  
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048  
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";  
    ssl_session_cache shared:SSL:1m;  
    ssl_session_timeout 10m;  
    ssl_protocols TLSv1.2;  
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";  
    ssl_prefer_server_ciphers on;  
  
    # Load configuration files for the default server block.  
    include /etc/nginx/default.d/*.conf;  
  
    location / {  
    }  
  
    error_page 404 /404.html;  
    location = /40x.html {
```

```

    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}

```

Amazon Linux 2023

Gunakan editor teks untuk mengedit file `/etc/nginx/nginx.conf`. Ini memerlukan izin root Linux. Di bagian atas file, tambahkan baris berikut:

```

ssl_engine cloudhsm;
env CLOUDHSM_PIN;

```

Kemudian tambahkan yang berikut ini ke bagian TLS file:

```

# Settings for a TLS enabled server.
server {
    listen      443 ssl http2 default_server;
    listen      [::]:443 ssl http2 default_server;
    server_name _;
    root        /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout 10m;
    ssl_protocols TLSv1.2;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.

```

```
include /etc/nginx/default.d/*.conf;

location / {

error_page 404 /404.html;
location = /40x.html {

error_page 500 502 503 504 /50x.html;
location = /50x.html {

}
}
```

Red Hat 8

Gunakan editor teks untuk mengedit file `/etc/nginx/nginx.conf`. Ini memerlukan izin root Linux. Di bagian atas file, tambahkan baris berikut:

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Kemudian tambahkan yang berikut ini ke bagian TLS file:

```
# Settings for a TLS enabled server.
server {
    listen      443 ssl http2 default_server;
    listen      [::]:443 ssl http2 default_server;
    server_name _;
    root        /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout 10m;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
```

```

RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {

    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}

```

Red Hat 9

Gunakan editor teks untuk mengedit file `/etc/nginx/nginx.conf`. Ini memerlukan izin root Linux. Di bagian atas file, tambahkan baris berikut:

```

ssl_engine cloudhsm;
env CLOUDHSM_PIN;

```

Kemudian tambahkan yang berikut ini ke bagian TLS file:

```

# Settings for a TLS enabled server.
server {
    listen      443 ssl http2 default_server;
    listen      [::]:443 ssl http2 default_server;
    server_name _;
    root        /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is strongly recommended to generate unique DH parameters

```

```
# Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
#ssl_dhparam "/etc/pki/nginx/dhparams.pem";
ssl_session_cache shared:SSL:1m;
ssl_session_timeout 10m;
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
ssl_prefer_server_ciphers on;

# Load configuration files for the default server block.
include /etc/nginx/default.d/*.conf;

location / {
}

error_page 404 /404.html;
location = /40x.html {
}

error_page 500 502 503 504 /50x.html;
location = /50x.html {
}
}
```

Ubuntu 22.04 LTS

Gunakan editor teks untuk mengedit file `/etc/nginx/nginx.conf`. Ini memerlukan izin root Linux. Di bagian atas file, tambahkan baris berikut:

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Kemudian tambahkan yang berikut ini ke bagian TLS file:

```
# Settings for a TLS enabled server.
server {
    listen      443 ssl http2 default_server;
```

```
listen      [::]:443 ssl http2 default_server;
server_name _;
root        /usr/share/nginx/html;

ssl_certificate "/etc/pki/nginx/server.crt";
ssl_certificate_key "/etc/pki/nginx/private/server.key";
# It is *strongly* recommended to generate unique DH parameters
# Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
#ssl_dhparam "/etc/pki/nginx/dhparams.pem";
ssl_session_cache shared:SSL:1m;
ssl_session_timeout 10m;
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
ssl_prefer_server_ciphers on;

# Load configuration files for the default server block.
include /etc/nginx/default.d/*.conf;

location / {

error_page 404 /404.html;
location = /40x.html {

error_page 500 502 503 504 /50x.html;
location = /50x.html {

}
```

Ubuntu 24.04 LTS

Gunakan editor teks untuk mengedit file `/etc/nginx/nginx.conf`. Ini memerlukan izin root Linux. Di bagian atas file, tambahkan baris berikut:

```
ssl_engine clouhsm;
```



```
env CLOUDHSM_PIN;
```

Kemudian tambahkan yang berikut ini ke bagian TLS file:

```
# Settings for a TLS enabled server.
server {
    listen      443 ssl http2 default_server;
    listen      [::]:443 ssl http2 default_server;
    server_name _;
    root        /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout 10m;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

Simpan file tersebut.

8. Cadangkan file konfigurasi `systemd`, dan kemudian atur jalur `EnvironmentFile`.

Amazon Linux 2

1. pencadangan `nginx.service` file.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Buka file `/lib/systemd/system/nginx.service` dalam teks editor, dan kemudian di bawah bagian `[Layanan]`, tambahkan lintasan berikut:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Amazon Linux 2023

1. pencadangan `nginx.service` file.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Buka `/lib/systemd/system/nginx.service` file di editor teks. dan kemudian di bawah bagian `[Layanan]`, tambahkan jalur berikut:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Red Hat 8

1. pencadangan `nginx.service` file.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Buka file `/lib/systemd/system/nginx.service` dalam teks editor, dan kemudian di bawah bagian `[Layanan]`, tambahkan lintasan berikut:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Red Hat 9

1. pencadangan `nginx.service` file.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Buka file `/lib/systemd/system/nginx.service` dalam teks editor, dan kemudian di bawah bagian [Layanan], tambahkan lintasan berikut:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 22.04 LTS

1. pencadangan `nginx.service` file.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Buka file `/lib/systemd/system/nginx.service` dalam teks editor, dan kemudian di bawah bagian [Layanan], tambahkan lintasan berikut:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 24.04 LTS

1. pencadangan `nginx.service` file.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Buka file `/lib/systemd/system/nginx.service` dalam teks editor, dan kemudian di bawah bagian [Layanan], tambahkan lintasan berikut:

```
EnvironmentFile=/etc/sysconfig/nginx
```

9. Periksa apakah file `/etc/sysconfig/nginx` ada, dan lakukan salah satu hal berikut:

- Jika file ada, buat cadangan file dengan menjalankan perintah berikut:

```
$ sudo cp /etc/sysconfig/nginx /etc/sysconfig/nginx.backup
```

- Jika file tidak ada, buka editor teks, dan kemudian membuat file bernama `nginx` dalam folder `/etc/sysconfig/`.

10. Konfigurasi lingkungan NGINX.

 Note

SDK Klien 5 memperkenalkan `CLOUDHSM_PIN` variabel lingkungan untuk menyimpan kredensial CU.

Amazon Linux 2

Buka file `/etc/sysconfig/nginx` di editor teks. Ini memerlukan izin root Linux. Tambahkan kredensi Pengguna Kriptografi (CU):

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti `<CU user name>` dan `<password>` dengan kredensial CU.

Simpan file tersebut.

Amazon Linux 2023

Buka file `/etc/sysconfig/nginx` di editor teks. Ini memerlukan izin root Linux. Tambahkan kredensi Pengguna Kriptografi (CU):

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti `<CU user name>` dan `<password>` dengan kredensial CU.

Simpan file tersebut.

Red Hat 8

Buka file `/etc/sysconfig/nginx` di editor teks. Ini memerlukan izin root Linux.
Tambahkan kredensi Pengguna Kriptografi (CU):

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU.

Simpan file tersebut.

Red Hat 9

Buka file `/etc/sysconfig/nginx` di editor teks. Ini memerlukan izin root Linux.
Tambahkan kredensi Pengguna Kriptografi (CU):

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU.

Simpan file tersebut.

Ubuntu 22.04 LTS

Buka file `/etc/sysconfig/nginx` di editor teks. Ini memerlukan izin root Linux.
Tambahkan kredensi Pengguna Kriptografi (CU):

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU.

Simpan file tersebut.

Ubuntu 24.04 LTS

Buka file `/etc/sysconfig/nginx` di editor teks. Ini memerlukan izin root Linux.
Tambahkan kredensi Pengguna Kriptografi (CU):

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU.

Simpan file tersebut.

11. Mulai server web NGINX.

Amazon Linux 2

Hentikan proses NGINX yang sedang berjalan

```
$ sudo systemctl stop nginx
```

Muat ulang systemd konfigurasi untuk mengambil perubahan terbaru

```
$ sudo systemctl daemon-reload
```

Mulai proses NGINX

```
$ sudo systemctl start nginx
```

Amazon Linux 2023

Hentikan semua proses NGINX

```
$ sudo systemctl stop nginx
```

Muat ulang systemd konfigurasi untuk mengambil perubahan terbaru

```
$ sudo systemctl daemon-reload
```

Mulai NGINX

```
$ sudo systemctl start nginx
```

Red Hat 8

Hentikan proses NGINX yang sedang berjalan

```
$ sudo systemctl stop nginx
```

Muat ulang systemd konfigurasi untuk mengambil perubahan terbaru

```
$ sudo systemctl daemon-reload
```

Mulai proses NGINX

```
$ sudo systemctl start nginx
```

Red Hat 9

Hentikan proses NGINX yang sedang berjalan

```
$ sudo systemctl stop nginx
```

Muat ulang systemd konfigurasi untuk mengambil perubahan terbaru

```
$ sudo systemctl daemon-reload
```

Mulai proses NGINX

```
$ sudo systemctl start nginx
```

Ubuntu 22.04 LTS

Hentikan proses NGINX yang sedang berjalan

```
$ sudo systemctl stop nginx
```

Muat ulang systemd konfigurasi untuk mengambil perubahan terbaru

```
$ sudo systemctl daemon-reload
```

Mulai proses NGINX

```
$ sudo systemctl start nginx
```

Ubuntu 24.04 LTS

Hentikan proses NGINX yang sedang berjalan

```
$ sudo systemctl stop nginx
```

Muat ulang systemd konfigurasi untuk mengambil perubahan terbaru

```
$ sudo systemctl daemon-reload
```

Mulai proses NGINX

```
$ sudo systemctl start nginx
```

12. (Opsional) Konfigurasikan platform Anda untuk memulai NGINX saat mulai.

Amazon Linux 2

```
$ sudo systemctl enable nginx
```

Amazon Linux 2023

```
$ sudo systemctl enable nginx
```

Red Hat 8

```
$ sudo systemctl enable nginx
```

Red Hat 9

```
$ sudo systemctl enable nginx
```

Ubuntu 22.04 LTS

```
$ sudo systemctl enable nginx
```


Ubuntu 24.04 LTS

```
$ sudo systemctl enable nginx
```

Setelah memperbarui konfigurasi server web, buka [Langkah 4: Aktifkan lalu lintas HTTPS dan verifikasi sertifikat](#).

Konfigurasi server web Apache

Gunakan bagian ini untuk mengatur konfigurasi Apache pada platform yang didukung.

Untuk memperbarui konfigurasi server web untuk Apache

1. Hubungkan ke instans klien Amazon EC2 Anda.
2. Tentukan lokasi default untuk sertifikat dan kunci privat untuk platform Anda.

Amazon Linux 2

Dalam `/etc/httpd/conf.d/ssl.conf` file, pastikan nilai-nilai ini ada:

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt  
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Amazon Linux 2023

Dalam `/etc/httpd/conf.d/ssl.conf` file, pastikan nilai-nilai ini ada:

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt  
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Red Hat 8

Dalam `/etc/httpd/conf.d/ssl.conf` file, pastikan nilai-nilai ini ada:

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt  
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Red Hat 9

Dalam `/etc/httpd/conf.d/ssl.conf` file, pastikan nilai-nilai ini ada:

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt  
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Ubuntu 22.04 LTS

Dalam `/etc/apache2/sites-available/default-ssl.conf` file, pastikan nilai-nilai ini ada:

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt  
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 24.04 LTS

Dalam `/etc/apache2/sites-available/default-ssl.conf` file, pastikan nilai-nilai ini ada:

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt  
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

3. Salin sertifikat server web Anda ke lokasi yang diperlukan untuk platform Anda.

Amazon Linux 2

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Ganti `<web_server.crt>` dengan nama sertifikat server web Anda.

Amazon Linux 2023

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Ganti `<web_server.crt>` dengan nama sertifikat server web Anda.

Red Hat 8

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Ganti `<web_server.crt>` dengan nama sertifikat server web Anda.

Red Hat 9

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Ganti `<web_server.crt>` dengan nama sertifikat server web Anda.

Ubuntu 22.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

Ganti `<web_server.crt>` dengan nama sertifikat server web Anda.

Ubuntu 24.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

Ganti `<web_server.crt>` dengan nama sertifikat server web Anda.

4. Salin kunci privat PEM palsu Anda ke lokasi yang diperlukan untuk platform Anda.

Amazon Linux 2

```
$ sudo cp <web_server_fake_pem.key> /etc/pki/tls/private/localhost.key
```

Ganti `<web_server_fake_pem.key>` dengan nama file yang berisi kunci pribadi PEM palsu Anda.

Amazon Linux 2023

```
$ sudo cp <web_server_fake_pem.key> /etc/pki/tls/private/localhost.key
```

Ganti `<web_server_fake_pem.key>` dengan nama file yang berisi kunci pribadi PEM palsu Anda.

Red Hat 8

```
$ sudo cp <web_server_fake_pem.key> /etc/pki/tls/private/localhost.key
```

Ganti `<web_server_fake_pem.key>` dengan nama file yang berisi kunci pribadi PEM palsu Anda.

Red Hat 9

```
$ sudo cp <web_server_fake_pem.key> /etc/pki/tls/private/localhost.key
```

Ganti `<web_server_fake_pem.key>` dengan nama file yang berisi kunci pribadi PEM palsu Anda.

Ubuntu 22.04 LTS

```
$ sudo cp <web_server_fake_pem.key> /etc/ssl/private/localhost.key
```

Ganti `<web_server_fake_pem.key>` dengan nama file yang berisi kunci pribadi PEM palsu Anda.

Ubuntu 24.04 LTS

```
$ sudo cp <web_server_fake_pem.key> /etc/ssl/private/localhost.key
```

Ganti `<web_server_fake_pem.key>` dengan nama file yang berisi kunci pribadi PEM palsu Anda.

5. Ubah kepemilikan file-file ini jika diperlukan oleh platform Anda.

Amazon Linux 2

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/localhost.key
```

Memberikan izin baca kepada pengguna bernama apache.

Amazon Linux 2023

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/localhost.key
```

Memberikan izin baca kepada pengguna bernama apache.

Red Hat 8

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/localhost.key
```

Memberikan izin baca kepada pengguna bernama apache.

Red Hat 9

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/localhost.key
```

Memberikan izin baca kepada pengguna bernama apache.

Ubuntu 22.04 LTS

Tidak ada tindakan diperlukan.

Ubuntu 24.04 LTS

Tidak ada tindakan diperlukan.

6. Atur konfigurasi arahan Apache untuk platform Anda.

Amazon Linux 2

Temukan file SSL untuk platform ini:

```
/etc/httpd/conf.d/ssl.conf
```

File ini berisi arahan Apache yang menentukan bagaimana server Anda harus berjalan. Arahan muncul di sebelah kiri, diikuti dengan nilai. Gunakan editor teks untuk mengedit file ini. Ini memerlukan izin root Linux.

Perbarui atau masukkan arahan berikut dengan nilai ini:

```
SSLCryptoDevice cloudhsm  
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
```

```
SHA384:ECDSA-AES256-SHA384:ECDSA-AES128-GCM-SHA256:ECDSA-
AES128-SHA256:ECDSA-AES256-SHA:ECDSA-AES128-SHA
```

Simpan file tersebut.

Amazon Linux 2023

Temukan file SSL untuk platform ini:

```
/etc/httpd/conf.d/ssl.conf
```

File ini berisi arahan Apache yang menentukan bagaimana server Anda harus berjalan. Arahan muncul di sebelah kiri, diikuti dengan nilai. Gunakan editor teks untuk mengedit file ini. Ini memerlukan izin root Linux.

Perbarui atau masukkan arahan berikut dengan nilai ini:

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDSA-RSA-AES128-GCM-SHA256:ECDSA-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDSA-RSA-AES256-SHA384:ECDSA-
RSA-AES128-SHA256:ECDSA-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDSA-AES256-GCM-
SHA384:ECDSA-AES256-SHA384:ECDSA-AES128-GCM-SHA256:ECDSA-
AES128-SHA256:ECDSA-AES256-SHA:ECDSA-AES128-SHA
```

Simpan file tersebut.

Red Hat 8

Temukan file SSL untuk platform ini:

```
/etc/httpd/conf.d/ssl.conf
```

File ini berisi arahan Apache yang menentukan bagaimana server Anda harus berjalan. Arahan muncul di sebelah kiri, diikuti dengan nilai. Gunakan editor teks untuk mengedit file ini. Ini memerlukan izin root Linux.

Perbarui atau masukkan arahan berikut dengan nilai ini:

```
SSLCryptoDevice cloudhsm
SSLProtocol TLSv1.2 TLSv1.3
```

```

SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
SSLProxyCipherSuite HIGH:!aNULL

```

Simpan file tersebut.

Red Hat 9

Temukan file SSL untuk platform ini:

```
/etc/httpd/conf.d/ssl.conf
```

File ini berisi arahan Apache yang menentukan bagaimana server Anda harus berjalan. Arahan muncul di sebelah kiri, diikuti dengan nilai. Gunakan editor teks untuk mengedit file ini. Ini memerlukan izin root Linux.

Perbarui atau masukkan arahan berikut dengan nilai ini:

```

SSLCryptoDevice ccloudhsm
SSLProtocol TLSv1.2 TLSv1.3
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
SSLProxyCipherSuite HIGH:!aNULL

```

Simpan file tersebut.

Ubuntu 22.04 LTS

Temukan file SSL untuk platform ini:

```
/etc/apache2/mods-available/ssl.conf
```

File ini berisi arahan Apache yang menentukan bagaimana server Anda harus berjalan. Arahan muncul di sebelah kiri, diikuti dengan nilai. Gunakan editor teks untuk mengedit file ini. Ini memerlukan izin root Linux.

Perbarui atau masukkan arahan berikut dengan nilai ini:

```
SSLCryptoDevice ccloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
SSLProtocol TLSv1.2 TLSv1.3
```

Simpan file tersebut.

Mengaktifkan modul SSL dan konfigurasi situs SSL default:

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 24.04 LTS

Temukan file SSL untuk platform ini:

```
/etc/apache2/mods-available/ssl.conf
```

File ini berisi arahan Apache yang menentukan bagaimana server Anda harus berjalan. Arahan muncul di sebelah kiri, diikuti dengan nilai. Gunakan editor teks untuk mengedit file ini. Ini memerlukan izin root Linux.

Perbarui atau masukkan arahan berikut dengan nilai ini:

```
SSLCryptoDevice ccloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
```



```
SHA384:ECDSA-AES256-SHA384:ECDSA-AES128-GCM-SHA256:ECDSA-AES128-SHA256:ECDSA-AES256-SHA:ECDSA-AES128-SHA  
SSLProtocol TLSv1.2 TLSv1.3
```

Simpan file tersebut.

Mengaktifkan modul SSL dan konfigurasi situs SSL default:

```
$ sudo a2enmod ssl  
$ sudo a2ensite default-ssl
```

7. Atur konfigurasi file nilai lingkungan untuk platform Anda.

Amazon Linux 2

Buka file layanan httpd:

```
/lib/systemd/system/httpd.service
```

Di bawah bagian [Service], tambahkan berikut ini:

```
EnvironmentFile=/etc/sysconfig/httpd
```

Amazon Linux 2023

Buka file layanan httpd:

```
/lib/systemd/system/httpd.service
```

Di bawah bagian [Service], tambahkan berikut ini:

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 8

Buka file layanan httpd:

```
/lib/systemd/system/httpd.service
```

Di bawah bagian [Service], tambahkan berikut ini:

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 9

Buka file layanan httpd:

```
/lib/systemd/system/httpd.service
```

Di bawah bagian [Service], tambahkan berikut ini:

```
EnvironmentFile=/etc/sysconfig/httpd
```

Ubuntu 22.04 LTS

Tidak ada tindakan diperlukan. Nilai lingkungan masuk /etc/sysconfig/httpd

Ubuntu 24.04 LTS

Tidak ada tindakan diperlukan. Nilai lingkungan masuk /etc/sysconfig/httpd

8. Dalam file yang menyimpan variabel lingkungan untuk platform Anda, tetapkan variabel lingkungan yang berisi kredensial pengguna kriptografi (CU):

Amazon Linux 2

Gunakan editor teks untuk mengedit /etc/sysconfig/httpd.

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti <CU user name> dan <password> dengan kredensial CU.

Amazon Linux 2023

Gunakan editor teks untuk mengedit /etc/sysconfig/httpd.

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti <CU user name> dan <password> dengan kredensial CU.

Red Hat 8

Gunakan editor teks untuk mengedit `/etc/sysconfig/httpd`.

```
CLLOUDHSM_PIN=<CU user name>:<password>
```

Ganti `<CU user name>` dan `<password>` dengan kredensial CU.



Note

SDK Klien 5 memperkenalkan CLOUDHSM_PIN variabel lingkungan untuk menyimpan kredensial CU.

Red Hat 9

Gunakan editor teks untuk mengedit `/etc/sysconfig/httpd`.

```
CLLOUDHSM_PIN=<CU user name>:<password>
```

Ganti `<CU user name>` dan `<password>` dengan kredensial CU.



Note

SDK Klien 5 memperkenalkan CLOUDHSM_PIN variabel lingkungan untuk menyimpan kredensial CU.

Ubuntu 22.04 LTS

Gunakan editor teks untuk mengedit `/etc/apache2/envvars`.

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti `<CU user name>` dan `<password>` dengan kredensial CU.

Note

SDK Klien 5 memperkenalkan CLOUDHSM_PIN variabel lingkungan untuk menyimpan kredensial CU. Dalam Klien SDK 3, Anda menyimpan kredensial CU di variabel lingkungan n3fips_password. Klien SDK 5 mendukung kedua variabel lingkungan, namun sebaiknya gunakan CLOUDHSM_PIN.

Ubuntu 24.04 LTS

Gunakan editor teks untuk mengedit /etc/apache2/envvars.

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti <CU user name> dan <password> dengan kredensial CU.

Note

SDK Klien 5 memperkenalkan CLOUDHSM_PIN variabel lingkungan untuk menyimpan kredensial CU. Dalam Klien SDK 3, Anda menyimpan kredensial CU di variabel lingkungan n3fips_password. Klien SDK 5 mendukung kedua variabel lingkungan, namun sebaiknya gunakan CLOUDHSM_PIN.

9. Mulai server web Apache.

Amazon Linux 2

```
$ sudo systemctl daemon-reload  
$ sudo service httpd start
```

Amazon Linux 2023

```
$ sudo systemctl daemon-reload  
$ sudo service httpd start
```

Red Hat 8

```
$ sudo systemctl daemon-reload
```

```
$ sudo service httpd start
```

Red Hat 9

```
$ sudo systemctl daemon-reload  
$ sudo service httpd start
```

Ubuntu 22.04 LTS

```
$ sudo service apache2 start
```

Ubuntu 24.04 LTS

```
$ sudo service apache2 start
```

10. (Opsional) Atur konfigurasi platform Anda untuk memulai Apache saat mulai.

Amazon Linux 2

```
$ sudo chkconfig httpd on
```

Amazon Linux 2023

```
$ sudo chkconfig httpd on
```

Red Hat 8

```
$ systemctl enable httpd
```

Red Hat 9

```
$ systemctl enable httpd
```

Ubuntu 22.04 LTS

```
$ sudo systemctl enable apache2
```

Ubuntu 24.04 LTS

```
$ sudo systemctl enable apache2
```

Setelah memperbarui konfigurasi server web, buka [Langkah 4: Aktifkan lalu lintas HTTPS dan verifikasi sertifikat](#).

Langkah 4: Aktifkan lalu lintas HTTPS dan verifikasi sertifikat

Setelah Anda mengonfigurasi server web untuk SSL/TLS dibongkar AWS CloudHSM, tambahkan instance server web Anda ke grup keamanan yang memungkinkan lalu lintas HTTPS masuk. Hal ini memungkinkan klien, seperti peramban web, untuk membuat koneksi HTTPS dengan server web Anda. Kemudian buat koneksi HTTPS ke server web Anda dan verifikasi bahwa itu menggunakan sertifikat yang Anda konfigurasikan untuk SSL/TLS dibongkar. AWS CloudHSM

Topik

- [Aktifkan koneksi HTTPS masuk](#)
- [Verifikasi bahwa HTTPS menggunakan sertifikat yang Anda konfigurasikan](#)

Aktifkan koneksi HTTPS masuk

Untuk menyambungkan ke server web Anda dari klien (seperti peramban web), buat grup keamanan yang mengizinkan koneksi HTTPS masuk. Secara khusus, ini harus mengizinkan koneksi TCP masuk pada port 443. Tetapkan grup keamanan ini ke server web Anda.

Untuk membuat grup keamanan untuk HTTPS dan menentukannya ke server web Anda

1. Buka konsol Amazon EC2 di <https://console.aws.amazon.com/ec2/>
2. Pilih Grup keamanan di panel navigasi.
3. Pilih Buat grup keamanan.
4. Untuk Buat Grup Keamanan, lakukan hal berikut:
 - a. Untuk Nama grup keamanan, ketik nama untuk grup keamanan yang Anda buat.
 - b. (Opsional) Ketik deskripsi grup keamanan yang Anda buat.
 - c. Untuk VPC, pilih VPC yang berisi instans Amazon EC2 server web Anda.
 - d. Pilih Tambahkan Aturan.

- e. Untuk Type, pilih HTTPS dari jendela drop-down.
 - f. Untuk Sumber, masukkan lokasi sumber.
 - g. Pilih Buat grup keamanan.
5. Di panel navigasi, pilih Instans.
 6. Pilih kotak centang di samping instans server web Anda.
 7. Pilih menu tarik-turun Tindakan di bagian atas halaman. Pilih Keamanan dan kemudian Ubah Grup Keamanan.
 8. Untuk grup keamanan terkait, pilih kotak pencarian dan pilih grup keamanan yang Anda buat untuk HTTPS. Kemudian pilih Tambahkan Grup Keamanan.
 9. Pilih Simpan.

Verifikasi bahwa HTTPS menggunakan sertifikat yang Anda konfigurasi

Setelah menambahkan server web ke grup keamanan, Anda dapat memverifikasi bahwa SSL/TLS offload menggunakan sertifikat yang ditandatangani sendiri. Anda dapat melakukan ini dengan peramban web atau dengan alat seperti [OpenSSL s_client](#).

Untuk memverifikasi SSL/TLS pembongkaran dengan browser web

1. Gunakan browser web untuk terhubung ke server web Anda menggunakan nama DNS publik atau alamat IP server. Pastikan URL di bilah alamat dimulai dengan https://. Misalnya, **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**.

 Tip

Anda dapat menggunakan layanan DNS seperti Amazon Route 53 untuk merutekan nama domain situs web Anda (misalnya, `https://www.example.com/`) ke server web Anda. Untuk informasi selengkapnya, lihat [Perutean Lalu Lintas ke Instans Amazon EC2](#) dalam Panduan Developer Amazon Route 53 atau dalam dokumentasi untuk layanan DNS Anda.

2. Gunakan peramban web Anda untuk melihat sertifikat server web. Untuk informasi lebih lanjut, lihat hal berikut:
 - Para Mozilla Firefox, lihat [Lihat Sertifikat](#) di situs web Dukung Mozilla.

- Untuk Google Chrome, lihat [Memahami Masalah Keamanan](#) pada Alat Google Tools untuk Pengembang Web.

Peramban web lain mungkin memiliki fitur serupa yang dapat Anda gunakan untuk melihat sertifikat server web.

3. Pastikan bahwa SSL/TLS sertifikat adalah salah satu yang Anda konfigurasi server web Anda untuk digunakan.

Untuk memverifikasi SSL/TLS offload dengan OpenSSL s_client

1. Jalankan perintah OpenSSL berikut untuk terhubung ke server web Anda menggunakan HTTPS. Ganti *<server name>* dengan nama DNS publik atau alamat IP server web Anda.

```
openssl s_client -connect <server name>:443
```

 Tip

Anda dapat menggunakan layanan DNS seperti Amazon Route 53 untuk merutekan nama domain situs web Anda (misalnya, <https://www.example.com/>) ke server web Anda. Untuk informasi selengkapnya, lihat [Perutean Lalu Lintas ke Instans Amazon EC2](#) dalam Panduan Developer Amazon Route 53 atau dalam dokumentasi untuk layanan DNS Anda.

2. Pastikan bahwa SSL/TLS sertifikat adalah salah satu yang Anda konfigurasi server web Anda untuk digunakan.

Anda sekarang memiliki situs web yang diamankan dengan HTTPS. Kunci pribadi untuk server web disimpan dalam HSM di AWS CloudHSM cluster Anda.

Untuk menambahkan penyeimbang beban, lihat [Tambahkan penyeimbang beban dengan Elastic Load Balancing AWS CloudHSM untuk \(opsional\)](#).

AWS CloudHSM SSL/TLS offload di Linux menggunakan NGINX atau dengan OpenSSL Provider HAProxy

Topik ini memberikan step-by-step instruksi untuk mengatur SSL/TLS pembongkaran identitas server dengan AWS CloudHSM pada server web Linux menggunakan NGINX atau dengan Penyedia HAProxy OpenSSL.

Topik

- [Ikhtisar](#)
- [Langkah 1: Siapkan prasyarat](#)
- [Langkah 2: Buat atau impor kunci pribadi dan dapatkan sertifikat](#)
- [Langkah 3: Konfigurasi server web](#)
- [Langkah 4: Aktifkan lalu lintas HTTPS dan verifikasi sertifikat](#)

Ikhtisar

[Di Linux, perangkat lunak NGINX dan server HAProxy web terintegrasi dengan OpenSSL untuk mendukung HTTPS.](#) Penyedia [AWS CloudHSM OpenSSL](#) menyediakan antarmuka yang memungkinkan perangkat lunak server web untuk menggunakan di cluster Anda untuk HSMs pembongkaran kriptografi dan penyimpanan kunci. OpenSSL Provider adalah jembatan yang menghubungkan server web ke cluster Anda. AWS CloudHSM

Untuk menyelesaikan tutorial ini, Anda akan mengkonfigurasi NGINX atau HAProxy menggunakan OpenSSL AWS CloudHSM Provider. Tutorial menunjukkan kepada Anda bagaimana melakukan hal berikut:

- Instal perangkat lunak server web pada instans Amazon EC2.
- Atur konfigurasi perangkat lunak server web untuk mendukung HTTPS dengan kunci privat yang tersimpan di klaster AWS CloudHSM .
- (Opsional) Gunakan Amazon EC2 untuk membuat instans server web kedua dan Elastic Load Balancing untuk membuat penyeimbang beban. Menggunakan penyeimbang beban dapat meningkatkan performa dengan mendistribusikan beban di beberapa server. Hal ini juga dapat memberikan redundansi dan ketersediaan yang lebih tinggi jika satu atau lebih server gagal.

Saat Anda siap memulai, buka [Langkah 1: Siapkan prasyarat](#).

Langkah 1: Siapkan prasyarat

Platform yang berbeda memerlukan prasyarat yang berbeda. Gunakan bagian prasyarat di bawah ini yang sesuai dengan platform Anda.

Prasyarat untuk Penyedia OpenSSL AWS CloudHSM

Untuk mengatur pembongkaran identitas server web SSL/TLS server dengan AWS CloudHSM OpenSSL Provider for Client SDK 5, Anda memerlukan yang berikut ini:

- AWS CloudHSM Cluster aktif dengan setidaknya dua modul keamanan perangkat keras (HSM)

Note

Anda dapat menggunakan satu HSM klaster, tetapi Anda harus terlebih dahulu menonaktifkan daya tahan kunci klien. Untuk informasi selengkapnya, lihat [Kelola Pengaturan Daya Tahan Kunci Klien](#) dan [Alat Konfigurasi Klien SDK 5](#).

- Instans Amazon EC2 menjalankan sistem operasi Linux dengan perangkat lunak berikut diinstal:
 - Server web (baik NGINX atau) HAProxy
 - Penyedia AWS CloudHSM OpenSSL untuk SDK Klien 5
- [Pengguna krypto](#) (CU) harus memiliki dan mengelola kunci privat server web pada HSM.

Untuk mengatur sebuah instans server web Linux dan membuat CU pada HSM

Note

Banyak perintah dalam prosedur ini memerlukan hak istimewa yang tinggi. Anda mungkin perlu menjalankan perintah dengan sudo atau sebagai pengguna root tergantung pada konfigurasi sistem Anda.

1. Instal dan konfigurasi Penyedia AWS CloudHSM OpenSSL untuk Client SDK 5. Untuk informasi selengkapnya tentang menginstal Penyedia OpenSSL, lihat Penyedia [AWS CloudHSM OpenSSL](#) untuk SDK Klien 5.
2. Pada instans EC2 Linux yang memiliki akses ke cluster Anda, instal NGINX atau HAProxy server web:

Amazon Linux 2023

- NGINX

```
$ yum install nginx
```

- HAProxy

```
$ yum install haproxy
```

RHEL 9 (9.2+)

- NGINX

```
$ yum install nginx
```

- HAProxy

```
$ yum install haproxy
```

RHEL 10 (10.0+)

- NGINX

```
$ yum install nginx
```

- HAProxy

```
$ yum install haproxy
```

Ubuntu 24.04

- NGINX

```
$ apt install nginx
```

- HAProxy

```
$ apt install haproxy
```

3. [Gunakan CloudHSM CLI untuk membuat pengguna krypto](#). Untuk informasi selengkapnya tentang mengelola pengguna HSM, lihat [Mengelola pengguna HSM dengan CloudHSM CLI](#).

 Tip

Lacak nama pengguna dan kata sandi CU. Anda akan membutuhkannya nanti ketika Anda membuat atau mengimpor kunci privat HTTPS dan sertifikat untuk server web Anda.

Setelah Anda menyelesaikan langkah ini, buka [Langkah 2: Buat atau impor kunci pribadi dan dapatkan sertifikat](#).

Catatan

- Untuk menggunakan Security-Enhanced Linux (SELinux) dan server web, Anda harus mengizinkan koneksi TCP keluar pada port 2223, yang merupakan port yang digunakan Client SDK 5 untuk berkomunikasi dengan HSM.
- Untuk membuat dan mengaktifkan klaster dan memberikan akses instans EC2 ke klaster, selesaikan langkah-langkah dalam [Memulai dengan AWS CloudHSM](#). Memulai menawarkan step-by-step instruksi untuk membuat cluster aktif dengan satu HSM dan instans klien Amazon EC2. Anda dapat menggunakan instans klien ini sebagai server web Anda.
- Untuk menghindari menonaktifkan daya tahan kunci klien, tambahkan lebih dari satu HSM ke klaster Anda. Untuk informasi lebih lanjut, lihat [Menambahkan HSM ke cluster AWS CloudHSM](#).
- Untuk terhubung ke instans klien Anda, Anda dapat menggunakan SSH atau PuTTY. Untuk informasi selengkapnya, lihat [Menghubungkan ke Instans Linux Anda Menggunakan SSH](#) atau [Menyambung ke Instans Linux Anda dari Windows Menggunakan PuTTY](#) dalam dokumentasi Amazon EC2.

Langkah 2: Buat atau impor kunci pribadi dan dapatkan sertifikat

Untuk mengaktifkan HTTPS, aplikasi server web Anda (NGINX atau HAProxy) memerlukan kunci pribadi dan sertifikat yang sesuai SSL/TLS. Untuk menggunakan identitas SSL/TLS server web offload dengan AWS CloudHSM, Anda harus menyimpan kunci pribadi di HSM di cluster Anda

AWS CloudHSM . Pertama-tama Anda akan membuat kunci pribadi dan menggunakan kunci untuk membuat permintaan penandatanganan sertifikat (CSR). Anda kemudian mengekspor kunci pribadi PEM palsu dari HSM, yang merupakan file kunci pribadi dalam format PEM yang berisi referensi ke kunci pribadi yang disimpan di HSM (itu bukan kunci pribadi yang sebenarnya). Server web Anda menggunakan file kunci pribadi PEM palsu untuk mengidentifikasi kunci pribadi pada HSM selama pembongkaran identitas SSL/TLS server.

Hasilkan kunci pribadi

[Bagian ini menunjukkan cara membuat keypair menggunakan CloudHSM CLI.](#) Setelah Anda memiliki key pair yang dihasilkan di dalam HSM, Anda dapat mengekspornya sebagai file PEM palsu dan menghasilkan sertifikat yang sesuai.

Instal dan konfigurasi CloudHSM CLI

1. [Instal dan Konfigurasi](#) CloudHSM CLI.
2. Gunakan perintah berikut untuk memulai CloudHSM CLI.

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

3. Jalankan perintah berikut untuk masuk ke HSM. Ganti *<user name>* dengan nama pengguna pengguna krypto Anda

```
aws-cloudhsm>login --username <user name> --role crypto-user
```

Menghasilkan Kunci Pribadi

Tergantung pada kasus penggunaan Anda, Anda dapat menghasilkan RSA atau EC key pair. Lakukan salah satu tindakan berikut:

- Untuk menghasilkan kunci pribadi RSA pada HSM

Gunakan [key generate-asymmetric-pair rsa](#) perintah untuk menghasilkan key pair RSA. Contoh ini menghasilkan key pair RSA dengan modulus 2048, eksponen publik 65537, label kunci publik, dan label kunci pribadi. *tls_rsa_pub tls_rsa_private*

```
aws-cloudhsm > key generate-asymmetric-pair rsa \  
--public-exponent 65537 \  
--modulus-size-bits 2048 \  

```

```
--public-label tls_rsa_pub \  
--private-label tls_rsa_private \  
--private-attributes sign=true  
{  
  "error_code": 0,  
  "data": {  
    "public_key": {  
      "key-reference": "0x0000000000280cc8",  
      "key-info": {  
        "key-owners": [  
          {  
            "username": "cu1",  
            "key-coverage": "full"  
          }  
        ],  
        "shared-users": [],  
        "cluster-coverage": "full"  
      },  
      "attributes": {  
        "key-type": "rsa",  
        "label": "tls_rsa_pub",  
        "id": "",  
        "check-value": "0x01fe6e",  
        "class": "public-key",  
        "encrypt": true,  
        "decrypt": false,  
        "token": true,  
        "always-sensitive": false,  
        "derive": false,  
        "destroyable": true,  
        "extractable": true,  
        "local": true,  
        "modifiable": true,  
        "never-extractable": false,  
        "private": true,  
        "sensitive": false,  
        "sign": false,  
        "trusted": false,  
        "unwrap": false,  
        "verify": false,  
        "wrap": false,  
        "wrap-with-trusted": false,  
        "key-length-bytes": 512,  
        "public-exponent": "0x010001",
```

```

    "modulus":
"0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1
    "modulus-size-bits": 2048
  }
},
"private_key": {
  "key-reference": "0x00000000000280cc7",
  "key-info": {
    "key-owners": [
      {
        "username": "cu1",
        "key-coverage": "full"
      }
    ],
    "shared-users": [],
    "cluster-coverage": "full"
  },
  "attributes": {
    "key-type": "rsa",
    "label": "tls_rsa_private",
    "id": "",
    "check-value": "0x01fe6e",
    "class": "private-key",
    "encrypt": false,
    "decrypt": true,
    "token": true,
    "always-sensitive": true,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": false,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 1217,
    "public-exponent": "0x010001",

```

```

    "modulus":
      "0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1"
    "modulus-size-bits": 2048
  }
}
}
}

```

- Untuk menghasilkan kunci pribadi EC pada HSM

Gunakan [key generate-asymmetric-pair ec](#) perintah untuk menghasilkan EC key pair. Contoh ini menghasilkan key pair EC dengan prime256v1 kurva (sesuai dengan NID_X9_62_prime256v1 kurva), label kunci publik *tls_ec_pub*, dan label kunci pribadi dari *tls_ec_private*.

```

aws-cloudhsm > key generate-asymmetric-pair ec \
  --curve prime256v1 \
  --public-label tls_ec_pub \
  --private-label tls_ec_private \
  --private-attributes sign=true
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x0000000000012000b",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "ec",
        "label": "tls_ec_pub",
        "id": "",
        "check-value": "0xd7c1a7",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,

```



```

    "token": false,
    "always-sensitive": false,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": false,
    "sign": false,
    "trusted": false,
    "unwrap": false,
    "verify": false,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 57,
    "ec-point":
"0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f3
    "curve": "secp224r1"
  }
},
"private_key": {
  "key-reference": "0x0000000000012000c",
  "key-info": {
    "key-owners": [
      {
        "username": "cu1",
        "key-coverage": "full"
      }
    ],
    "shared-users": [],
    "cluster-coverage": "session"
  },
  "attributes": {
    "key-type": "ec",
    "label": "tls_ec_private",
    "id": "",
    "check-value": "0xd7c1a7",
    "class": "private-key",
    "encrypt": false,
    "decrypt": false,
    "token": false,
    "always-sensitive": true,

```

```

    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": true,
    "trusted": false,
    "unwrap": false,
    "verify": false,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 122,
    "ec-point":
    "0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f3
    "curve": "secp224r1"
  }
}
}
}
}

```

Ekspor file kunci pribadi PEM palsu

Setelah Anda memiliki kunci pribadi di HSM, Anda harus mengekspor file kunci pribadi PEM palsu. File ini tidak berisi data kunci yang sebenarnya, tetapi memungkinkan OpenSSL Dynamic Engine untuk mengidentifikasi kunci pribadi pada HSM. Anda kemudian dapat menggunakan kunci pribadi untuk membuat permintaan penandatanganan sertifikat (CSR) dan menandatangani CSR untuk membuat sertifikat.

Gunakan [key generate-file](#) perintah untuk mengekspor kunci pribadi dalam format PEM palsu dan menyimpannya ke file. Ganti nilai berikut dengan nilai Anda sendiri:

- *<private_key_label>*— Label kunci pribadi yang Anda buat pada langkah sebelumnya.
- *<web_server_fake_pem.key>*— Nama file tempat kunci PEM palsu Anda akan ditulis.

```

aws-cloudhsm > key generate-file --encoding reference-pem --
path <web_server_fake_pem.key> --filter attr.label=<private_key_label>
{

```

```
"error_code": 0,  
"data": {  
  "message": "Successfully generated key file"  
}  
}
```

Keluar dari CloudHSM CLI

Jalankan perintah berikut untuk menghentikan CloudHSM CLI.

```
aws-cloudhsm > quit
```

Anda sekarang harus memiliki file baru di sistem Anda, yang terletak di jalur yang ditentukan oleh `<web_server_fake_pem.key>` dalam perintah sebelumnya. File ini adalah file kunci pribadi PEM palsu.

Menghasilkan sertifikat yang ditandatangani sendiri

Setelah Anda membuat kunci pribadi PEM palsu, Anda dapat menggunakan file ini untuk menghasilkan permintaan penandatanganan sertifikat (CSR) dan sertifikat.

Di lingkungan produksi, Anda biasanya menggunakan sertifikat otoritas (CA) untuk membuat sertifikat dari CSR. CA tidak diperlukan untuk lingkungan pengujian. Jika Anda menggunakan CA, kirim file CSR kepada mereka dan gunakan SSL/TLS sertifikat bertanda tangan yang mereka berikan kepada Anda di server web Anda untuk HTTPS.

Sebagai alternatif untuk menggunakan CA, Anda dapat menggunakan AWS CloudHSM OpenSSL Dynamic Engine untuk membuat sertifikat yang ditandatangani sendiri. Sertifikat yang ditandatangani sendiri tidak dipercaya oleh peramban dan tidak boleh digunakan dalam lingkungan produksi. Sertifikat dapat digunakan dalam lingkungan pengujian.

Warning

Sertifikat yang ditandatangani sendiri hanya boleh digunakan dalam lingkungan pengujian. Untuk lingkungan produksi, gunakan metode yang lebih aman seperti otoritas sertifikat untuk membuat sertifikat.

Instal dan konfigurasi OpenSSL Dynamic Engine

1. Hubungkan ke instans klien Anda.

2. [the section called “Menginstal”](#)

Menghasilkan sertifikat

1. Dapatkan salinan file PEM palsu Anda yang dihasilkan pada langkah sebelumnya.
2. Buat CSR

Jalankan perintah berikut untuk menggunakan AWS CloudHSM OpenSSL Dynamic Engine untuk membuat permintaan penandatanganan sertifikat (CSR). Ganti `<web_server_fake_pem.key>` dengan nama file yang berisi kunci pribadi PEM palsu Anda. Ganti `<web_server.csr>` dengan nama file yang berisi CSR Anda.

Perintah `req` bersifat interaktif. Tanggapi setiap bidang. Informasi bidang disalin ke SSL/TLS sertifikat Anda.

Note

Pembuatan CSR saat ini tidak didukung dengan Penyedia OpenSSL. Anda harus menggunakan OpenSSL Engine untuk langkah ini, tetapi operasi cipher TLS akan bekerja dengan Provider.

```
$ openssl req -engine cloudhsm -new -key <web_server_fake_pem.key> -  
out <web_server.csr>
```

3. Buat sertifikat yang ditandatangani sendiri

Jalankan perintah berikut untuk menggunakan AWS CloudHSM OpenSSL Dynamic Engine untuk menandatangani CSR Anda dengan kunci pribadi Anda di HSM Anda. Ini membuat sertifikat yang ditandatangani sendiri. Ganti nilai berikut dalam perintah dengan nilai Anda sendiri.

- `<web_server.csr>`— Nama file yang berisi CSR.
- `<web_server_fake_pem.key>`— Nama file yang berisi kunci pribadi PEM palsu.
- `<web_server.crt>`— Nama file yang akan berisi sertifikat server web Anda.

```
$ openssl x509 -engine cloudhsm -req -days 365 -in <web_server.csr> -  
signkey <web_server_fake_pem.key> -out <web_server.crt>
```

Setelah Anda memiliki kunci pribadi dan sertifikat, buka [Langkah 3: Konfigurasi server web](#).

Langkah 3: Konfigurasi server web

Perbarui konfigurasi perangkat lunak server web Anda untuk menggunakan sertifikat HTTPS dan kunci privat PEM palsu yang Anda buat di [langkah sebelumnya](#). Ingatlah untuk mencadangkan sertifikat dan kunci yang sudah ada sebelum memulai. Ini akan menyelesaikan pengaturan perangkat lunak server web Linux Anda untuk SSL/TLS pembongkaran identitas server. AWS CloudHSM

Menyelesaikan langkah-langkah dari salah satu bagian berikut.

Topik

- [Konfigurasi server web NGINX](#)
- [Konfigurasi server HAProxy web](#)

Konfigurasi server web NGINX

Gunakan bagian ini untuk mengkonfigurasi NGINX dengan Penyedia OpenSSL.

Untuk mengkonfigurasi NGINX untuk Penyedia OpenSSL

1. Hubungkan ke instans klien Anda.
2. Jalankan perintah berikut untuk membuat direktori yang diperlukan untuk sertifikat server web dan kunci privat PEM palsu.

```
$ mkdir -p /etc/pki/nginx/private
```

3. Jalankan perintah berikut untuk menyalin sertifikat server web Anda ke lokasi yang diperlukan. Ganti `<web_server.crt>` dengan nama sertifikat server web Anda.

```
$ cp <web_server.crt> /etc/pki/nginx/server.crt
```

4. Jalankan perintah berikut untuk menyalin kunci privat PEM palsu Anda ke lokasi yang diperlukan. Ganti `<web_server_fake_pem.key>` dengan nama file yang berisi kunci pribadi PEM palsu Anda.

```
$ cp <web_server_fake_pem.key> /etc/pki/nginx/private/server.key
```

5. Jalankan perintah berikut untuk mengubah kepemilikan file sehingga pengguna bernama nginx dapat membacanya.

```
$ chown nginx /etc/pki/nginx/server.crt /etc/pki/nginx/private/server.key
```

6. Konfigurasi OpenSSH untuk menggunakan penyedia. AWS CloudHSM Untuk informasi selengkapnya tentang mengonfigurasi Penyedia OpenSSH, lihat Penyedia [AWS CloudHSM OpenSSH untuk SDK Klien](#) 5.

1. Temukan file konfigurasi OpenSSH Anda:

```
$ openssl version -d
```

Anda akan melihat output yang mirip dengan:

```
OPENSSLDIR: "/etc/pki/tls"
```

File konfigurasi ada `openssl.cnf` di direktori ini.

- 2.

Note

Jangan memodifikasi file `openssl.cnf` default sistem Anda secara langsung. Ini mencegah operasi OpenSSH di seluruh sistem (SSH, koneksi TLS, dan layanan lainnya) dari routing secara tidak sengaja melalui penyedia CloudHSM. Menggunakan file konfigurasi terpisah memungkinkan Anda untuk mencakup penggunaan Penyedia CloudHSM hanya ke aplikasi tertentu yang memerlukan operasi kriptografi yang didukung HSM.

Buat file konfigurasi OpenSSH baru dengan konten berikut:

```
$ cat > <example-cloudhsm-openssl.cnf> << 'EOF'
## NOTE: This should point to the system default openssl config file.
```

```
# Replace /etc/pki/tls with the path to your OpenSSL configuration directory
.include </etc/pki/tls>/openssl.cnf

# Override the existing provider_section to include AWS CloudHSM OpenSSL
# Provider as a 3rd party OpenSSL provider
[provider_sect]
default = default_sect
# Include AWS CloudHSM CloudHSM OpenSSL provider
cloudhsm = cloudhsm_sect

[default_sect]
activate = 1

[cloudhsm_sect]
activate = 1

EOF
```

3. Pastikan variabel CLOUDHSM_PIN lingkungan disetel dengan kredensial pengguna kriptografi (CU) Anda:

```
$ export CLOUDHSM_PIN=<username>:<password>
```

4. Setel variabel OPENSSL_CONF lingkungan untuk menunjuk ke file konfigurasi Anda yang diperbarui dan verifikasi penyedia dimuat:

```
$ OPENSSL_CONF=/path/to/example-cloudhsm-openssl.cnf openssl list -providers
```

Anda akan melihat penyedia default dan penyedia CloudHSM terdaftar:

```
OPENSSL_CONF=/path/to/example-cloudhsm-openssl.cnf openssl list -providers
Providers:
  default
    name: OpenSSL Default Provider
    version: 3.2.2
    status: active
  cloudhsm
    name: AWS CloudHSM OpenSSL Provider
    version: 5.17.0
    status: active
```

7. Jalankan perintah berikut untuk mencadangkan file /etc/nginx/nginx.conf.

```
$ cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.backup
```

8. Perbarui konfigurasi NGINX.

Note

Setiap kluster dapat mendukung maksimum 1000 proses pekerja NGINX di semua server web NGINX.

Amazon Linux 2023

Gunakan editor teks untuk mengedit file `/etc/nginx/nginx.conf`. Ini mungkin memerlukan izin root Linux. Di bagian atas file, tambahkan baris berikut:

```
env CLOUDHSM_PIN;
env OPENSSL_CONF;
```

Kemudian tambahkan yang berikut ini ke bagian TLS file:

```
# Settings for a TLS enabled server.
server {
    listen      443 ssl http2 default_server;
    listen      [::]:443 ssl http2 default_server;
    server_name _;
    root        /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is strongly recommended to generate unique DH parameters for DHE
    ciphers
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    # ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout 10m;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-CHACHA20-";
```



```
POLY1305:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-
ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-CHACHA20-
POLY1305:TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256";
    ssl_prefer_server_ciphers off;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

RHEL 9 (9.2+)

Gunakan editor teks untuk mengedit file `/etc/nginx/nginx.conf`. Ini mungkin memerlukan izin root Linux. Di bagian atas file, tambahkan baris berikut:

```
env CLOUDHSM_PIN;
env OPENSSL_CONF;
```

Kemudian tambahkan yang berikut ini ke bagian TLS file:

```
# Settings for a TLS enabled server.
server {
    listen      443 ssl http2 default_server;
    listen      [::]:443 ssl http2 default_server;
    server_name _;
    root        /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is strongly recommended to generate unique DH parameters for DHE
    ciphers
```

```
# Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
# ssl_dhparam "/etc/pki/nginx/dhparams.pem";
ssl_session_cache shared:SSL:1m;
ssl_session_timeout 10m;
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-CHACHA20-POLY1305:TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256";
ssl_prefer_server_ciphers off;

# Load configuration files for the default server block.
include /etc/nginx/default.d/*.conf;

location / {

error_page 404 /404.html;
location = /40x.html {

error_page 500 502 503 504 /50x.html;
location = /50x.html {

}
```

RHEL 10 (10.0+)

Gunakan editor teks untuk mengedit file `/etc/nginx/nginx.conf`. Ini mungkin memerlukan izin root Linux. Di bagian atas file, tambahkan baris berikut:

```
env CLOUDHSM_PIN;
env OPENSSL_CONF;
```

Kemudian tambahkan yang berikut ini ke bagian TLS file:

```
# Settings for a TLS enabled server.
server {
```

```

listen      443 ssl http2 default_server;
listen      [::]:443 ssl http2 default_server;
server_name _;
root        /usr/share/nginx/html;

ssl_certificate "/etc/pki/nginx/server.crt";
ssl_certificate_key "/etc/pki/nginx/private/server.key";
# It is strongly recommended to generate unique DH parameters for DHE
ciphers
# Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
# ssl_dhparam "/etc/pki/nginx/dhparams.pem";
ssl_session_cache shared:SSL:1m;
ssl_session_timeout 10m;
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-
SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-
RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA256:DHE-
RSA-AES256-SHA256:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-CHACHA20-
POLY1305:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-
ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-CHACHA20-
POLY1305:TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256";
ssl_prefer_server_ciphers off;

# Load configuration files for the default server block.
include /etc/nginx/default.d/*.conf;

location / {

error_page 404 /404.html;
location = /40x.html {

error_page 500 502 503 504 /50x.html;
location = /50x.html {

}
}

```

Ubuntu 24.04

Gunakan editor teks untuk mengedit file `/etc/nginx/nginx.conf`. Ini mungkin memerlukan izin root Linux. Di bagian atas file, tambahkan baris berikut:

```
env CLOUDHSM_PIN;
env OPENSSL_CONF;
```

Kemudian tambahkan yang berikut ini ke bagian TLS file:

```
# Settings for a TLS enabled server.
server {
    listen      443 ssl http2 default_server;
    listen      [::]:443 ssl http2 default_server;
    server_name _;
    root        /var/www/html;

    ssl_certificate "/etc/ssl/certs/server.crt";
    ssl_certificate_key "/etc/ssl/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters for DHE
    ciphers
    # Generate them with: openssl dhparam -out /etc/ssl/certs/dhparams.pem 2048
    # ssl_dhparam "/etc/ssl/certs/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout 10m;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-CHACHA20-POLY1305:TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256";
    ssl_prefer_server_ciphers off;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {

    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
```

```
}  
}
```

Simpan file tersebut.

9. Cadangkan file konfigurasi systemd, dan kemudian atur jalur EnvironmentFile.

Amazon Linux 2023

1. Cadangkan nginx.service file:

```
$ cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Buka /lib/systemd/system/nginx.service di editor teks. Di bawah bagian [Layanan], tambahkan:

```
EnvironmentFile=/etc/sysconfig/nginx
```

RHEL 9 (9.2+)

1. Cadangkan nginx.service file:

```
$ cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Buka /lib/systemd/system/nginx.service di editor teks. Di bawah bagian [Layanan], tambahkan:

```
EnvironmentFile=/etc/sysconfig/nginx
```

RHEL 10 (10.0+)

1. Cadangkan nginx.service file:

```
$ cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Buka `/lib/systemd/system/nginx.service` di editor teks. Di bawah bagian [Layanan], tambahkan:

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 24.04

1. Cadangkan `nginx.service` file:

```
$ cp /lib/systemd/system/nginx.service /lib/systemd/system/  
nginx.service.backup
```

2. Buka `/lib/systemd/system/nginx.service` di editor teks. Di bawah bagian [Layanan], tambahkan:

```
EnvironmentFile=/etc/sysconfig/nginx
```

10. Periksa apakah file `/etc/sysconfig/nginx` ada, dan lakukan salah satu hal berikut:

- Jika file ada, buat cadangan file dengan menjalankan perintah berikut:

```
$ cp /etc/sysconfig/nginx /etc/sysconfig/nginx.backup
```

- Jika file tidak ada, buka editor teks, dan kemudian membuat file bernama `nginx` dalam folder `/etc/sysconfig/`.

11. Konfigurasi lingkungan NGINX.

Amazon Linux 2023

Sebagai pengguna root Linux, buka `/etc/sysconfig/nginx` file di editor teks. Misalnya,

```
vi /etc/sysconfig/nginx
```

Tambahkan kredensial Cryptography User (CU) dan path ke file konfigurasi OpenSSL Anda:

```
CLOUDHSM_PIN=<CU user name>:<password>  
OPENSSL_CONF=<path to example-cloudhsm-openssl.cnf>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU. Ganti *<path to example-cloudhsm-openssl.cnf>* dengan path lengkap ke file konfigurasi yang Anda buat [Untuk mengkonfigurasi NGINX untuk Penyedia OpenSSL](#).

Simpan file tersebut.

RHEL 9 (9.2+)

Buka file `/etc/sysconfig/nginx` di editor teks. Ini mungkin memerlukan izin root Linux. Tambahkan kredensial Cryptography User (CU) dan path ke file konfigurasi OpenSSL Anda:

```
CLLOUDHSM_PIN=<CU user name>:<password>  
OPENSSL_CONF=<path to example-cloudhsm-openssl.cnf>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU. Ganti *<path to example-cloudhsm-openssl.cnf>* dengan path lengkap ke file konfigurasi yang Anda buat [Untuk mengkonfigurasi NGINX untuk Penyedia OpenSSL](#).

Simpan file tersebut.

RHEL 10 (10.0+)

Buka file `/etc/sysconfig/nginx` di editor teks. Ini mungkin memerlukan izin root Linux. Tambahkan kredensial Cryptography User (CU) dan path ke file konfigurasi OpenSSL Anda:

```
CLLOUDHSM_PIN=<CU user name>:<password>  
OPENSSL_CONF=<path to example-cloudhsm-openssl.cnf>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU. Ganti *<path to example-cloudhsm-openssl.cnf>* dengan path lengkap ke file konfigurasi yang Anda buat [Untuk mengkonfigurasi NGINX untuk Penyedia OpenSSL](#).

Simpan file tersebut.

Ubuntu 24.04

Buka file `/etc/sysconfig/nginx` di editor teks. Ini mungkin memerlukan izin root Linux. Tambahkan kredensial Cryptography User (CU) dan path ke file konfigurasi OpenSSL Anda:

```
CLLOUDHSM_PIN=<CU user name>:<password>
```

```
OPENSSL_CONF=<path to example-cloudhsm-openssl.cnf>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU. Ganti *<path to example-cloudhsm-openssl.cnf>* dengan path lengkap ke file konfigurasi yang Anda buat [Untuk mengkonfigurasi NGINX untuk Penyedia OpenSSL](#).

Simpan file tersebut.

12. Mulai server web NGINX.

Amazon Linux 2023

Hentikan semua proses NGINX

```
$ systemctl stop nginx
```

Muat ulang systemd konfigurasi untuk mengambil perubahan terbaru

```
$ systemctl daemon-reload
```

Mulai NGINX

```
$ systemctl start nginx
```

RHEL 9 (9.2+)

Hentikan proses NGINX yang sedang berjalan

```
$ systemctl stop nginx
```

Muat ulang systemd konfigurasi untuk mengambil perubahan terbaru

```
$ systemctl daemon-reload
```

Mulai proses NGINX

```
$ systemctl start nginx
```


RHEL 10 (10.0+)

Hentikan proses NGINX yang sedang berjalan

```
$ systemctl stop nginx
```

Muat ulang systemd konfigurasi untuk mengambil perubahan terbaru

```
$ systemctl daemon-reload
```

Mulai proses NGINX

```
$ systemctl start nginx
```

Ubuntu 24.04

Hentikan proses NGINX yang sedang berjalan

```
$ systemctl stop nginx
```

Muat ulang systemd konfigurasi untuk mengambil perubahan terbaru

```
$ systemctl daemon-reload
```

Mulai proses NGINX

```
$ systemctl start nginx
```

Setelah Anda mengkonfigurasi NGINX, buka. [Verifikasi bahwa HTTPS menggunakan sertifikat yang Anda konfigurasikan](#)

Konfigurasi server HAProxy web

Gunakan bagian ini untuk mengkonfigurasi HAProxy dengan Penyedia OpenSSL. Contoh berikut menunjukkan cara mengatur HAProxy dengan sertifikat dan kunci CloudHSM Anda.

Untuk mengkonfigurasi HAProxy untuk Penyedia OpenSSL

1. Cadangkan file sertifikat gabungan yang ada jika ada:

```
$ cp server-combined.pem server-combined.pem.backup
```

2. Buat file sertifikat gabungan untuk HAProxy menggunakan sertifikat dan kunci PEM palsu CloudHSM Anda:

```
$ cat server.crt server.key > server-combined.pem
```

3. Cadangkan HAProxy konfigurasi yang ada:

```
$ cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.backup
```

4. Buat konfigurasi offload CloudHSM TLS baru di: /etc/haproxy/haproxy.cfg

```
global
    daemon
    ssl-provider cloudhsm
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/haproxy/dhparams.pem 2048
    # ssl-dh-param-file /etc/haproxy/dhparams.pem
    ssl-default-bind-ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-
GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-
AES256-SHA384:ECDHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-
SHA256:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES256-
GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-CHACHA20-POLY1305
    ssl-default-bind-ciphersuites
    TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
    ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets

defaults
    mode http
    timeout connect 5000ms
    timeout client 50000ms
    timeout server 50000ms

frontend haproxy_frontend
    bind *:443 ssl crt /path/to/server-combined.pem
    default_backend web_servers
```

```
backend web_servers
    server web1 127.0.0.1:8080 check
```

Perbarui jalur sertifikat agar sesuai dengan lokasi file Anda.

5. Konfigurasi systemd untuk menggunakan file lingkungan untuk HAProxy. Lokasi tergantung pada distribusi Linux Anda.

Amazon Linux and RHEL

Cadangkan dan modifikasi file HAProxy layanan:

```
$ cp /lib/systemd/system/haproxy.service /lib/systemd/system/
haproxy.service.backup
```

Edit `/lib/systemd/system/haproxy.service` dan tambahkan baris berikut di bawah bagian [Layanan]:

```
EnvironmentFile=/etc/sysconfig/haproxy
```

Ubuntu

Cadangkan dan modifikasi file HAProxy layanan:

```
$ cp /lib/systemd/system/haproxy.service /lib/systemd/system/
haproxy.service.backup
```

Edit `/lib/systemd/system/haproxy.service` dan tambahkan baris berikut di bawah bagian [Layanan]:

```
EnvironmentFile=/etc/default/haproxy
```

6. Buat file lingkungan di lokasi yang sesuai untuk sistem Anda.

Amazon Linux and RHEL

Cadangkan file HAProxy lingkungan jika ada:

```
$ cp /etc/sysconfig/haproxy /etc/sysconfig/haproxy.backup
```

Buat file HAProxy lingkungan `/etc/sysconfig/haproxy` dengan konten berikut:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ubuntu

Cadangkan file HAProxy lingkungan jika ada:

```
$ cp /etc/default/haproxy /etc/default/haproxy.backup
```

Buat file HAProxy lingkungan `/etc/default/haproxy` dengan konten berikut:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti `<CU user name>` dan `<password>` dengan kredensial CU Anda.

7. Muat ulang konfigurasi systemd:

```
$ systemctl daemon-reload
```

8. Mulailah HAProxy dengan konfigurasi offload CloudHSM TLS:

```
$ systemctl start haproxy
```

Anda juga dapat menjalankan HAProxy langsung dengan file konfigurasi khusus:

```
$ haproxy -f /path/to/haproxy-cloudhsm.cfg
```

Setelah Anda mengkonfigurasi HAProxy, buka [Verifikasi bahwa HTTPS menggunakan sertifikat yang Anda konfigurasi](#).

Langkah 4: Aktifkan lalu lintas HTTPS dan verifikasi sertifikat

Setelah Anda mengonfigurasi server web untuk SSL/TLS dibongkar AWS CloudHSM, tambahkan instance server web Anda ke grup keamanan yang memungkinkan lalu lintas HTTPS masuk. Hal ini memungkinkan klien, seperti peramban web, untuk membuat koneksi HTTPS dengan server web

Anda. Kemudian buat koneksi HTTPS ke server web Anda dan verifikasi bahwa itu menggunakan sertifikat yang Anda konfigurasi untuk SSL/TLS dibongkar. AWS CloudHSM

Topik

- [Aktifkan koneksi HTTPS masuk](#)
- [Verifikasi bahwa HTTPS menggunakan sertifikat yang Anda konfigurasi](#)

Aktifkan koneksi HTTPS masuk

Untuk menyambungkan ke server web Anda dari klien (seperti peramban web), buat grup keamanan yang mengizinkan koneksi HTTPS masuk. Secara khusus, ini harus mengizinkan koneksi TCP masuk pada port 443. Tetapkan grup keamanan ini ke server web Anda.

Untuk membuat grup keamanan untuk HTTPS dan menentukannya ke server web Anda

1. Buka konsol Amazon EC2 di <https://console.aws.amazon.com/ec2/>
2. Pilih Grup keamanan di panel navigasi.
3. Pilih Buat grup keamanan.
4. Untuk Buat Grup Keamanan, lakukan hal berikut:
 - a. Untuk Nama grup keamanan, ketik nama untuk grup keamanan yang Anda buat.
 - b. (Opsional) Ketik deskripsi grup keamanan yang Anda buat.
 - c. Untuk VPC, pilih VPC yang berisi instans Amazon EC2 server web Anda.
 - d. Pilih Tambahkan Aturan.
 - e. Untuk Type, pilih HTTPS dari jendela drop-down.
 - f. Untuk Sumber, masukkan lokasi sumber.
 - g. Pilih Buat grup keamanan.
5. Di panel navigasi, pilih Instans.
6. Pilih kotak centang di samping instans server web Anda.
7. Pilih menu tarik-turun Tindakan di bagian atas halaman. Pilih Keamanan dan kemudian Ubah Grup Keamanan.
8. Untuk grup keamanan terkait, pilih kotak pencarian dan pilih grup keamanan yang Anda buat untuk HTTPS. Kemudian pilih Tambahkan Grup Keamanan.
9. Pilih Simpan.

Verifikasi bahwa HTTPS menggunakan sertifikat yang Anda konfigurasi

Setelah menambahkan server web ke grup keamanan, Anda dapat memverifikasi bahwa SSL/TLS offload menggunakan sertifikat yang ditandatangani sendiri. Anda dapat melakukan ini dengan peramban web atau dengan alat seperti [OpenSSL s_client](#).

Untuk memverifikasi SSL/TLS offload dengan browser web

1. Gunakan browser web untuk terhubung ke server web Anda menggunakan nama DNS publik atau alamat IP server. Pastikan URL di bilah alamat dimulai dengan `https://`. Misalnya, **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**.

 Tip

Anda dapat menggunakan layanan DNS seperti Amazon Route 53 untuk merutekan nama domain situs web Anda (misalnya, `https://www.example.com/`) ke server web Anda. Untuk informasi selengkapnya, lihat [Perutean Lalu Lintas ke Instans Amazon EC2](#) dalam Panduan Developer Amazon Route 53 atau dalam dokumentasi untuk layanan DNS Anda.

2. Gunakan peramban web Anda untuk melihat sertifikat server web. Untuk informasi lebih lanjut, lihat hal berikut:
 - Para Mozilla Firefox, lihat [Lihat Sertifikat](#) di situs web Dukung Mozilla.
 - Untuk Google Chrome, lihat [Memahami Masalah Keamanan](#) pada Alat Google Tools untuk Pengembang Web.

Peramban web lain mungkin memiliki fitur serupa yang dapat Anda gunakan untuk melihat sertifikat server web.

3. Pastikan bahwa SSL/TLS sertifikat adalah salah satu yang Anda konfigurasi server web Anda untuk digunakan.

Untuk memverifikasi SSL/TLS offload dengan OpenSSL s_client

1. Jalankan perintah OpenSSL berikut untuk terhubung ke server web Anda menggunakan HTTPS. Ganti `<server name>` dengan nama DNS publik atau alamat IP server web Anda.

```
openssl s_client -connect <server name>:443
```

 Tip

Anda dapat menggunakan layanan DNS seperti Amazon Route 53 untuk merutekan nama domain situs web Anda (misalnya, <https://www.example.com/>) ke server web Anda. Untuk informasi selengkapnya, lihat [Perutean Lalu Lintas ke Instans Amazon EC2](#) dalam Panduan Developer Amazon Route 53 atau dalam dokumentasi untuk layanan DNS Anda.

2. Pastikan bahwa SSL/TLS sertifikat adalah salah satu yang Anda konfigurasi server web Anda untuk digunakan.

Anda sekarang memiliki situs web yang diamankan dengan HTTPS. Kunci pribadi untuk server web disimpan dalam HSM di AWS CloudHSM cluster Anda.

Untuk menambahkan penyeimbang beban, lihat [Tambahkan penyeimbang beban dengan Elastic Load Balancing AWS CloudHSM untuk \(opsional\)](#).

AWS CloudHSM SSL/TLS offload di Linux menggunakan Tomcat dengan JSSE

Topik ini memberikan step-by-step petunjuk untuk mengatur SSL/TLS offload menggunakan Java Secure Socket Extension (JSSE) dengan JCE SDK. AWS CloudHSM

Topik

- [Ikhtisar](#)
- [Langkah 1: Siapkan prasyarat](#)
- [Langkah 2: Buat atau impor kunci pribadi dan SSL/TLS sertifikat](#)
- [Langkah 3: Konfigurasi server web Tomcat](#)
- [Langkah 4: Aktifkan lalu lintas HTTPS dan verifikasi sertifikat](#)

Ikhtisar

Pada tahun AWS CloudHSM, server web Tomcat bekerja di Linux untuk mendukung HTTPS. AWS CloudHSM JCE SDK menyediakan antarmuka yang dapat digunakan dengan JSSE (Java Secure Socket Extension) untuk memungkinkan penggunaan HSMs untuk server web tersebut. AWS CloudHSM JCE adalah jembatan yang menghubungkan JSSE ke klaster AWS CloudHSM Anda. JSSE adalah Java API untuk protokol Secure Sockets Layer (SSL) dan Transport Layer Security (TLS).

Langkah 1: Siapkan prasyarat

Ikuti prasyarat ini untuk menggunakan server web Tomcat untuk offload di Linux. AWS CloudHSM SSL/TLS Prasyarat ini harus dipenuhi untuk mengatur SSL/TLS offload server web dengan Client SDK 5 dan server web Tomcat.

Note

Platform yang berbeda memerlukan prasyarat yang berbeda. Selalu ikuti langkah-langkah instalasi yang benar untuk platform Anda.

Prasyarat

- Instans Amazon EC2 yang menjalankan sistem operasi Linux dengan server web tomcat diinstal.
- [Pengguna krypto](#) (CU) harus memiliki dan mengelola kunci privat server web pada HSM.
- AWS CloudHSM Cluster aktif dengan setidaknya dua modul keamanan perangkat keras (HSMs) yang memiliki [JCE untuk Client SDK 5](#) diinstal dan dikonfigurasi.

Note

Anda dapat menggunakan satu HSM klaster, tetapi Anda harus terlebih dahulu menonaktifkan daya tahan kunci klien. Untuk informasi selengkapnya, lihat [Kelola Pengaturan Daya Tahan Kunci Klien](#) dan [Alat Konfigurasi Klien SDK 5](#).

Bagaimana memenuhi prasyarat

1. Instal dan konfigurasi JCE untuk AWS CloudHSM pada AWS CloudHSM cluster aktif dengan setidaknya dua modul keamanan perangkat keras (HSMs). Untuk informasi selengkapnya tentang penginstalan, lihat [JCE for Client SDK 5](#).
2. Pada instans EC2 Linux yang memiliki akses ke AWS CloudHSM cluster Anda, ikuti [instruksi Apache Tomcat](#) untuk mengunduh dan menginstal server web Tomcat.
3. Gunakan [CloudHSM](#) CLI untuk membuat pengguna kriptografi (CU). Untuk informasi selengkapnya tentang mengelola pengguna HSM, lihat [Mengelola pengguna HSM dengan CloudHSM](#) CLI.

Tip

Lacak nama pengguna dan kata sandi CU. Anda akan membutuhkannya nanti ketika Anda membuat atau mengimpor kunci privat HTTPS dan sertifikat untuk server web Anda.

4. Untuk mengatur JCE dengan Java Keytool, ikuti instruksi di [Gunakan Client SDK 5 untuk mengintegrasikan AWS CloudHSM dengan Java Keytool dan Jarsigner](#)

Setelah Anda menyelesaikan langkah ini, buka [Langkah 2: Buat atau impor kunci pribadi dan SSL/TLS sertifikat](#).

Catatan

- Untuk menggunakan Security-Enhanced Linux (SELinux) dan server web, Anda harus mengizinkan koneksi TCP keluar pada port 2223, yang merupakan port yang digunakan Client SDK 5 untuk berkomunikasi dengan HSM.
- Untuk membuat dan mengaktifkan kluster dan memberikan akses instans EC2 ke kluster, selesaikan langkah-langkah dalam [Memulai dengan AWS CloudHSM](#). Bagian ini menawarkan step-by-step petunjuk untuk membuat cluster aktif dengan satu HSM dan instans klien Amazon EC2. Anda dapat menggunakan instans klien ini sebagai server web Anda.
- Untuk menghindari menonaktifkan daya tahan kunci klien, tambahkan lebih dari satu HSM ke kluster Anda. Untuk informasi lebih lanjut, lihat [Menambahkan HSM ke cluster AWS CloudHSM](#).
- Untuk terhubung ke instans klien Anda, Anda dapat menggunakan SSH atau PuTTY. Untuk informasi selengkapnya, lihat [Menghubungkan ke Instans Linux Anda Menggunakan SSH](#) atau [Menyambung ke Instans Linux Anda dari Windows Menggunakan PuTTY](#) dalam dokumentasi Amazon EC2.

Langkah 2: Buat atau impor kunci pribadi dan SSL/TLS sertifikat

Untuk mengaktifkan HTTPS, aplikasi server web Tomcat Anda memerlukan kunci pribadi dan SSL/TLS sertifikat yang sesuai. Untuk menggunakan server web SSL/TLS offload dengan AWS CloudHSM, Anda harus menyimpan kunci pribadi di HSM di cluster Anda AWS CloudHSM .

Note

Jika Anda belum memiliki kunci pribadi dan sertifikat yang sesuai, buat kunci pribadi di HSM. Anda menggunakan kunci pribadi untuk membuat permintaan penandatanganan sertifikat (CSR), yang Anda gunakan untuk membuat SSL/TLS sertifikat.

Anda membuat AWS CloudHSM KeyStore file lokal yang berisi referensi ke kunci pribadi Anda di HSM dan sertifikat terkait. Server web Anda menggunakan AWS CloudHSM KeyStore file untuk mengidentifikasi kunci pribadi pada HSM selama SSL/TLS offload.

Topik

- [Menghasilkan kunci pribadi](#)
- [Menghasilkan sertifikat yang ditandatangani sendiri](#)

Menghasilkan kunci pribadi

Bagian ini menunjukkan kepada Anda cara menghasilkan key pair menggunakan KeyTool from JDK. Setelah Anda memiliki key pair yang dihasilkan di dalam HSM, Anda dapat mengeksportnya sebagai KeyStore file, dan menghasilkan sertifikat yang sesuai.

Tergantung pada kasus penggunaan Anda, Anda dapat menghasilkan RSA atau EC key pair. Langkah-langkah berikut menunjukkan cara menghasilkan key pair RSA.

Gunakan **genkeypair** perintah KeyTool untuk menghasilkan key pair RSA

1. Setelah mengganti di **<VARIABLES>** bawah ini dengan data spesifik Anda, gunakan perintah berikut untuk menghasilkan file keystore bernama `jsse_keystore.keystore`, yang akan memiliki referensi kunci pribadi Anda di HSM.

```
$ keytool -genkeypair -alias <UNIQUE ALIAS FOR KEYS> -keyalg <KEY ALGORITHM> -  
keysize <KEY SIZE> -sigalg <SIGN ALGORITHM> \
```

```
-keystore <PATH>/<JSSE KEYSTORE NAME>.keystore -storetype CLOUDHSM \
-dname CERT_DOMAIN_NAME \
-J-classpath '-J'$JAVA_LIB'/*:/opt/cloudhsm/java/*:./*' \
-provider "com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider" \
-providerpath "$CLOUDHSM_JCE_LOCATION" \
-keypass <KEY PASSWORD> -storepass <KEYSTORE PASSWORD>
```

- **<PATH>**: Jalur yang ingin Anda hasilkan file keystore Anda.
 - **<UNIQUE ALIAS FOR KEYS>**: Ini digunakan untuk mengidentifikasi kunci Anda secara unik pada HSM. Alias ini akan ditetapkan sebagai atribut LABEL untuk kunci.
 - **<KEY PASSWORD>**: Kami menyimpan referensi ke kunci Anda di file keystore lokal, dan kata sandi ini melindungi referensi lokal itu.
 - **<KEYSTORE PASSWORD>**: Ini adalah kata sandi untuk file keystore lokal Anda.
 - **<JSSE KEYSTORE NAME>**: Nama file Keystore.
 - **<CERT DOMAIN NAME>**: X.500 Nama terhormat.
 - **<KEY ALGORITHM>**: Algoritma kunci untuk menghasilkan key pair (Misalnya, RSA dan EC).
 - **<KEY SIZE>** Ukuran kunci untuk menghasilkan key pair (misalnya, 2048, 3072, dan 4096).
 - **<SIGN ALGORITHM>**: Ukuran kunci untuk menghasilkan key pair (misalnya, SHA1with RSA, RSA, SHA224with RSA, SHA256with RSA, dan SHA384with RSA). SHA512with
2. Untuk mengonfirmasi bahwa perintah berhasil, masukkan perintah berikut dan verifikasi bahwa Anda telah berhasil menghasilkan key pair RSA.

```
$ ls <PATH>/<JSSE KEYSTORE NAME>.keystore
```

Menghasilkan sertifikat yang ditandatangani sendiri

Setelah Anda membuat kunci pribadi bersama dengan file keystore, Anda dapat menggunakan file ini untuk menghasilkan permintaan penandatanganan sertifikat (CSR) dan sertifikat.

Di lingkungan produksi, Anda biasanya menggunakan sertifikat otoritas (CA) untuk membuat sertifikat dari CSR. CA tidak diperlukan untuk lingkungan pengujian. Jika Anda menggunakan CA, kirim file CSR kepada mereka dan gunakan SSL/TLS sertifikat bertanda tangan yang mereka berikan kepada Anda di server web Anda untuk HTTPS.

Sebagai alternatif untuk menggunakan CA, Anda dapat menggunakan KeyTool untuk membuat sertifikat yang ditandatangani sendiri. Sertifikat yang ditandatangani sendiri tidak dipercaya oleh

peramban dan tidak boleh digunakan dalam lingkungan produksi. Sertifikat dapat digunakan dalam lingkungan pengujian.

Warning

Sertifikat yang ditandatangani sendiri hanya boleh digunakan dalam lingkungan pengujian. Untuk lingkungan produksi, gunakan metode yang lebih aman, seperti otoritas sertifikat untuk membuat sertifikat.

Menghasilkan sertifikat

1. Dapatkan salinan file keystore Anda yang dihasilkan pada langkah sebelumnya.
2. Jalankan perintah berikut untuk menggunakan KeyTool untuk membuat permintaan penandatanganan sertifikat (CSR).

```
$ keytool -certreq -keyalg RSA -alias unique_alias_for_key -file certreq.csr \  
-keystore <JSSE KEYSTORE NAME>.keystore -storetype CLOUDHSM \  
-J-classpath '-J$JAVA_LIB/*:/opt/cloudhsm/java/*:./*' \  
-keypass <KEY PASSWORD> -storepass <KEYSTORE PASSWORD>
```

Note

File keluaran dari permintaan penandatanganan sertifikat adalah `certreq.csr`.

Menandatangani sertifikat

- Setelah mengganti di *<VARIABLES>* bawah ini dengan data spesifik Anda, jalankan perintah berikut untuk menandatangani CSR Anda dengan kunci pribadi Anda di HSM Anda. Ini membuat sertifikat yang ditandatangani sendiri.

```
$ keytool -gencert -infile certreq.csr -outfile certificate.crt \  
-alias <UNIQUE ALIAS FOR KEYS> -keypass <KEY_PASSWORD> -  
storepass <KEYSTORE_PASSWORD> -sigalg SIG_ALG \  
-storetype CLOUDHSM -J-classpath '-J$JAVA_LIB/*:/opt/cloudhsm/java/*:./*' \  
-keystore jsse_keystore.keystore
```

 Note

`certificate.crt` adalah sertifikat yang ditandatangani yang menggunakan kunci pribadi alias.

Impor sertifikat di Keystore

- Setelah mengganti di **<VARIABLES>** bawah ini dengan data spesifik Anda, jalankan perintah berikut untuk mengimpor sertifikat yang ditandatangani sebagai sertifikat tepercaya. Langkah ini akan menyimpan sertifikat di entri keystore yang diidentifikasi oleh alias.

```
$ keytool -import -alias <UNIQUE ALIAS FOR KEYS> -keystore jsse_keystore.keystore \  
-file certificate.crt -storetype CLOUDHSM \  
-v -J-classpath '-J$JAVA_LIB/*:/opt/cloudhsm/java/*:./*' \  
-keypass <KEY PASSWORD> -storepass <KEYSTORE_PASSWORD>
```

Mengkonversi sertifikat ke PEM

- Jalankan perintah berikut untuk mengonversi file sertifikat yang ditandatangani (`.crt`) ke PEM. File PEM akan digunakan untuk mengirim permintaan dari klien http.

```
$ openssl x509 -inform der -in certificate.crt -out certificate.pem
```

Setelah Anda menyelesaikan langkah-langkah ini, pergi ke [Langkah 3: Konfigurasi server web](#).

Langkah 3: Konfigurasi server web Tomcat

Perbarui konfigurasi perangkat lunak server web Anda untuk menggunakan sertifikat HTTPS dan file PEM terkait yang Anda buat pada langkah sebelumnya. Ingatlah untuk mencadangkan sertifikat dan kunci yang sudah ada sebelum memulai. Ini akan menyelesaikan pengaturan perangkat lunak server web Linux Anda untuk SSL/TLS dibongkar dengan AWS CloudHSM. Untuk informasi lebih lanjut, lihat Referensi [Konfigurasi Apache Tomcat 9](#).

Hentikan server

- Setelah mengganti di **<VARIABLES>** bawah ini dengan data spesifik Anda, jalankan perintah berikut untuk menghentikan Tomcat Server sebelum memperbarui konfigurasi

```
$ /<TOMCAT DIRECTORY>/bin/shutdown.sh
```

- <TOMCAT DIRECTORY>**: Direktori instalasi Tomcat Anda.

Perbarui classpath Tomcat

- Hubungkan ke instans klien Anda.
- Temukan folder instalasi Tomcat.
- Setelah mengganti di **<VARIABLES>** bawah ini dengan data spesifik Anda, gunakan perintah berikut untuk menambahkan perpustakaan Java dan jalur AWS CloudHSM Java di Tomcatclasspath, yang terletak di Tomcat/bin/catalina file.sh.

```
$ sed -i 's@CLASSPATH="$CLASSPATH"$CATALINA_HOME"/bin/bootstrap.jar@CLASSPATH="$CLASSPATH"$CATALINA_HOME"/bin/bootstrap.jar:"  
    <JAVA LIBRARY>"'\/*:/opt/cloudhsm/java/*:.//*' <TOMCAT PATH> /bin/  
catalina.sh
```

- <JAVA LIBRARY>**: Lokasi Perpustakaan Java JRE.
- <TOMCAT PATH>**: Folder instalasi Tomcat.

Tambahkan konektor HTTPS dalam konfigurasi server.

- Buka folder instalasi Tomcat.
- Setelah mengganti di **<VARIABLES>** bawah ini dengan data spesifik Anda, gunakan perintah berikut untuk menambahkan konektor HTTPS untuk menggunakan sertifikat yang dihasilkan dalam prasyarat:

```
$ sed -i '/<Connector port="8080"/i <Connector port="\443" maxThreads="\200"  
scheme="\https" secure="\true" SSLEnabled="\true" keystoreType="\CLOUDHSM"  
keystoreFile="  
    <CUSTOM DIRECTORY>/<JSSE KEYSTORE NAME>.keystore" keystorePass="\<KEYSTORE  
PASSWORD>" keyPass="\<KEY PASSWORD>
```

```
\ " keyAlias=\"<UNIQUE ALIAS FOR KEYS>\" clientAuth=\"false\" sslProtocol=
\"TLS\"/>' <TOMCAT PATH>/conf/server.xml
```

- **<CUSTOM DIRECTORY>**: Direktori tempat file keystore berada.
- **<JSSE KEYSTORE NAME>**: Nama file Keystore.
- **<KEYSTORE PASSWORD>**: Ini adalah kata sandi untuk file keystore lokal Anda.
- **<KEY PASSWORD>**: Kami menyimpan referensi ke kunci Anda di file keystore lokal, dan kata sandi ini melindungi referensi lokal itu.
- **<UNIQUE ALIAS FOR KEYS>**: Ini digunakan untuk mengidentifikasi kunci Anda secara unik pada HSM. Alias ini akan ditetapkan sebagai atribut LABEL untuk kunci.
- **<TOMCAT PATH>**: Jalur ke folder Tomcat Anda.

Mulai Server

- Setelah mengganti di **<VARIABLES>** bawah ini dengan data spesifik Anda, gunakan perintah berikut untuk memulai Tomcat Server:

```
$ /<TOMCAT DIRECTORY>/bin/startup.sh
```

Note

<TOMCAT DIRECTORY> adalah nama direktori instalasi Tomcat Anda.

Setelah memperbarui konfigurasi server web, buka [Langkah 4: Aktifkan lalu lintas HTTPS dan verifikasi sertifikat](#).

Langkah 4: Aktifkan lalu lintas HTTPS dan verifikasi sertifikat

Setelah Anda mengonfigurasi server web untuk SSL/TLS dibongkar AWS CloudHSM, tambahkan instance server web Anda ke grup keamanan yang memungkinkan lalu lintas HTTPS masuk. Hal ini memungkinkan klien, seperti peramban web, untuk membuat koneksi HTTPS dengan server web Anda. Kemudian buat koneksi HTTPS ke server web Anda dan verifikasi bahwa itu menggunakan sertifikat yang Anda konfigurasikan untuk SSL/TLS dibongkar. AWS CloudHSM

Topik

- [Aktifkan koneksi HTTPS masuk](#)

- [Verifikasi bahwa HTTPS menggunakan sertifikat yang Anda konfigurasi](#)

Aktifkan koneksi HTTPS masuk

Untuk menyambungkan ke server web Anda dari klien (seperti peramban web), buat grup keamanan yang mengizinkan koneksi HTTPS masuk. Secara khusus, ini harus mengizinkan koneksi TCP masuk pada port 443. Tetapkan grup keamanan ini ke server web Anda.

Untuk membuat grup keamanan untuk HTTPS dan menentukannya ke server web Anda

1. Buka konsol Amazon EC2 di <https://console.aws.amazon.com/ec2/>
2. Pilih Grup keamanan di panel navigasi.
3. Pilih Buat grup keamanan.
4. Untuk Buat Grup Keamanan, lakukan hal berikut:
 - a. Untuk Nama grup keamanan, ketik nama untuk grup keamanan yang Anda buat.
 - b. (Opsional) Ketik deskripsi grup keamanan yang Anda buat.
 - c. Untuk VPC, pilih VPC yang berisi instans Amazon EC2 server web Anda.
 - d. Pilih Tambahkan Aturan.
 - e. Untuk Type, pilih HTTPS dari jendela drop-down.
 - f. Untuk Sumber, masukkan lokasi sumber.
 - g. Pilih Buat grup keamanan.
5. Di panel navigasi, pilih Instans.
6. Pilih kotak centang di samping instans server web Anda.
7. Pilih menu tarik-turun Tindakan di bagian atas halaman. Pilih Keamanan dan kemudian Ubah Grup Keamanan.
8. Untuk grup keamanan terkait, pilih kotak pencarian dan pilih grup keamanan yang Anda buat untuk HTTPS. Kemudian pilih Tambahkan Grup Keamanan.
9. Pilih Simpan.

Verifikasi bahwa HTTPS menggunakan sertifikat yang Anda konfigurasi

Setelah menambahkan server web ke grup keamanan, Anda dapat memverifikasi bahwa SSL/TLS offload menggunakan sertifikat yang ditandatangani sendiri. Anda dapat melakukan ini dengan peramban web atau dengan alat seperti [OpenSSL s_client](#).

Untuk memverifikasi SSL/TLS offload dengan browser web

1. Gunakan browser web untuk terhubung ke server web Anda menggunakan nama DNS publik atau alamat IP server. Pastikan URL di bilah alamat dimulai dengan `https://`. Misalnya, **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**.

 Tip

Anda dapat menggunakan layanan DNS seperti Amazon Route 53 untuk merutekan nama domain situs web Anda (misalnya, `https://www.example.com/`) ke server web Anda. Untuk informasi selengkapnya, lihat [Perutean Lalu Lintas ke Instans Amazon EC2](#) dalam Panduan Developer Amazon Route 53 atau dalam dokumentasi untuk layanan DNS Anda.

2. Gunakan peramban web Anda untuk melihat sertifikat server web. Untuk informasi lebih lanjut, lihat hal berikut:
 - Para Mozilla Firefox, lihat [Lihat Sertifikat](#) di situs web Dukung Mozilla.
 - Untuk Google Chrome, lihat [Memahami Masalah Keamanan](#) pada Alat Google Tools untuk Pengembang Web.

Peramban web lain mungkin memiliki fitur serupa yang dapat Anda gunakan untuk melihat sertifikat server web.

3. Pastikan bahwa SSL/TLS sertifikat adalah salah satu yang Anda konfigurasi server web Anda untuk digunakan.

Untuk memverifikasi SSL/TLS offload dengan OpenSSL `s_client`

1. Jalankan perintah OpenSSL berikut untuk terhubung ke server web Anda menggunakan HTTPS. Ganti `<server name>` dengan nama DNS publik atau alamat IP server web Anda.

```
openssl s_client -connect <server name>:443
```

 Tip

Anda dapat menggunakan layanan DNS seperti Amazon Route 53 untuk merutekan nama domain situs web Anda (misalnya, `https://www.example.com/`) ke server web

Anda. Untuk informasi selengkapnya, lihat [Perutean Lalu Lintas ke Instans Amazon EC2](#) dalam Panduan Developer Amazon Route 53 atau dalam dokumentasi untuk layanan DNS Anda.

2. Pastikan bahwa SSL/TLS sertifikat adalah salah satu yang Anda konfigurasi server web Anda untuk digunakan.

Anda sekarang memiliki situs web yang diamankan dengan HTTPS. Kunci pribadi untuk server web disimpan dalam HSM di AWS CloudHSM cluster Anda.

Untuk menambahkan penyeimbang beban, lihat [Tambahkan penyeimbang beban dengan Elastic Load Balancing AWS CloudHSM untuk \(opsional\)](#).

AWS CloudHSM SSL/TLS offload pada Windows menggunakan IIS dengan KSP

Tutorial ini memberikan step-by-step instruksi untuk mengatur SSL/TLS offload dengan AWS CloudHSM pada server web Windows.

Topik

- [Ikhtisar](#)
- [Langkah 1: Siapkan prasyarat](#)
- [Langkah 2: Buat permintaan penandatanganan sertifikat \(CSR\) dan sertifikat](#)
- [Langkah 3: Konfigurasi server web](#)
- [Langkah 4: Aktifkan lalu lintas HTTPS dan verifikasi sertifikat](#)

Ikhtisar

Pada Windows, aplikasi server web [Layanan Informasi Internet \(IIS\) untuk Windows Server](#) secara native mendukung HTTPS. [Penyedia penyimpanan AWS CloudHSM kunci \(KSP\) untuk Microsoft Cryptography API: Next Generation \(CNG\)](#) menyediakan antarmuka yang memungkinkan IIS untuk menggunakan HSMs di cluster Anda untuk pembongkaran kriptografi dan penyimpanan kunci. AWS CloudHSM KSP adalah jembatan yang menghubungkan IIS ke cluster Anda AWS CloudHSM .

Tutorial ini menunjukkan kepada Anda cara melakukan hal berikut:

- Instal perangkat lunak server web pada instans Amazon EC2.

- Atur konfigurasi perangkat lunak server web untuk mendukung HTTPS dengan kunci privat yang tersimpan di kluster AWS CloudHSM .
- (Opsional) Gunakan Amazon EC2 untuk membuat instans server web kedua dan Elastic Load Balancing untuk membuat penyeimbang beban. Menggunakan penyeimbang beban dapat meningkatkan performa dengan mendistribusikan beban di beberapa server. Hal ini juga dapat memberikan redundansi dan ketersediaan yang lebih tinggi jika satu atau lebih server gagal.

Saat Anda siap memulai, buka [Langkah 1: Siapkan prasyarat](#).

Langkah 1: Siapkan prasyarat

Platform yang berbeda memerlukan prasyarat yang berbeda. Gunakan bagian prasyarat di bawah ini yang sesuai dengan platform Anda.

Topik

- [Prasyarat untuk Klien SDK 5](#)
- [Prasyarat untuk Klien SDK 3](#)

Prasyarat untuk Klien SDK 5

Untuk mengatur SSL/TLS pembongkaran server web AWS CloudHSM, Anda memerlukan yang berikut ini:

- AWS CloudHSM Cluster aktif dengan setidaknya satu HSM.
- Instans Amazon EC2 menjalankan sistem operasi Windows dengan perangkat lunak berikut diinstal:
 - Perangkat lunak AWS CloudHSM klien untuk Windows.
 - Layanan Informasi Internet (IIS) untuk Windows Server.
- [Pengguna krypto](#) (CU) harus memiliki dan mengelola kunci privat server web pada HSM.

Note

Tutorial ini menggunakan Microsoft Windows Server 2019. Microsoft Windows Server 2016 dan 2022 juga didukung.

Untuk mengatur sebuah instans Windows Server dan membuat CU pada HSM

1. Selesaikan langkah-langkah dalam [Mulai menggunakan](#). Saat Anda meluncurkan klien Amazon EC2, pilih AMI Windows Server 2019. Ketika Anda menyelesaikan langkah-langkah ini, Anda memiliki kluster aktif dengan setidaknya satu HSM. Anda juga memiliki instans klien Amazon EC2 yang menjalankan Windows Server dengan perangkat lunak AWS CloudHSM klien untuk Windows diinstal.
2. (Opsional) Tambahkan lebih banyak HSMs ke cluster Anda. Untuk informasi selengkapnya, lihat [Menambahkan HSM ke cluster AWS CloudHSM](#).
3. Hubungkan ke server Windows Anda. Untuk informasi selengkapnya, lihat [Connect to Your Instance](#) di Panduan Pengguna Amazon EC2.
4. Gunakan CloudHSM CLI untuk membuat pengguna kriptografi (CU). Lacak nama pengguna dan kata sandi CU. Anda akan membutuhkannya untuk menyelesaikan langkah berikutnya.

 Note

Untuk informasi tentang cara membuat pengguna, lihat [Mengelola pengguna HSM dengan CloudHSM CLI](#).

5. [Atur kredensial login untuk HSM](#), menggunakan nama pengguna dan kata sandi CU yang Anda buat di langkah sebelumnya.
6. Pada langkah 5, jika Anda menggunakan Windows Credentials Manager untuk mengatur kredensial HSM, unduh [psexec.exe](#) dari SysInternals untuk menjalankan perintah berikut sebagai NT Authority\SYSTEM:

```
psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"  
--username <USERNAME> --password <PASSWORD>
```

Ganti **<USERNAME>** dan **<PASSWORD>** dengan kredensial HSM.

Untuk menginstal IIS pada Windows Server Anda

1. Jika Anda belum melakukannya, hubungkan ke server Windows server Anda. Untuk informasi selengkapnya, lihat [Connect to Your Instance](#) di Panduan Pengguna Amazon EC2.
2. Pada server Windows Anda, mulai Manager Server.

3. Di dasbor Server Manager, pilih Tambahkan peran dan fitur.
4. Baca informasi Sebelum Anda memulai, kemudian pilih Selanjutnya.
5. Untuk Jenis Instalasi, pilih Instalasi berbasis peran atau berbasis fitur. Lalu, pilih Selanjutnya.
6. Untuk Pemilihan Server, pilih Pilih server dari kolam server. Lalu, pilih Selanjutnya.
7. Untuk Peran Server, lakukan hal berikut:
 - a. Pilih Server Web (IIS).
 - b. Untuk Tambahkan fitur yang diperlukan untuk Web Server (IIS) Pilih Tambahkan fitur.
 - c. Pilih Selanjutnya untuk menyelesaikan pemilihan peran server.
8. Untuk Fitur, terima default. Lalu, pilih Selanjutnya.
9. Baca informasi Peran Server Web (IIS). Lalu, pilih Selanjutnya.
10. Untuk Pilih layanan peran, terima default atau ubah pengaturan sesuai pilihan. Lalu, pilih Selanjutnya.
11. Untuk Konfirmasi, baca informasi konfirmasi. Lalu, pilih Instal.
12. Setelah penginstalan selesai, pilih Tutup.

Setelah Anda menyelesaikan langkah ini, buka [Langkah 2: Buat permintaan penandatanganan sertifikat \(CSR\) dan sertifikat](#).

Prasyarat untuk Klien SDK 3

Untuk mengatur SSL/TLS pembongkaran server web AWS CloudHSM, Anda memerlukan yang berikut ini:

- AWS CloudHSM Cluster aktif dengan setidaknya satu HSM.
- Instans Amazon EC2 menjalankan sistem operasi Windows dengan perangkat lunak berikut diinstal:
 - Perangkat lunak AWS CloudHSM klien untuk Windows.
 - Layanan Informasi Internet (IIS) untuk Windows Server.
- [Pengguna krypto](#) (CU) harus memiliki dan mengelola kunci privat server web pada HSM.

Note

Tutorial ini menggunakan Microsoft Windows Server 2016. Microsoft Windows Server 2012 juga didukung, tetapi Microsoft Windows Server 2012 R2 tidak.

Untuk mengatur sebuah instans Windows Server dan membuat CU pada HSM

1. Selesaikan langkah-langkah dalam [Mulai menggunakan](#). Saat Anda meluncurkan klien Amazon EC2, pilih AMI Windows Server 2016 atau Windows Server 2012. Ketika Anda menyelesaikan langkah-langkah ini, Anda memiliki kluster aktif dengan setidaknya satu HSM. Anda juga memiliki instans klien Amazon EC2 yang menjalankan Windows Server dengan perangkat lunak AWS CloudHSM klien untuk Windows diinstal.
2. (Opsional) Tambahkan lebih banyak HSMs ke cluster Anda. Untuk informasi selengkapnya, lihat [Menambahkan HSM ke cluster AWS CloudHSM](#).
3. Hubungkan ke server Windows Anda. Untuk informasi selengkapnya, lihat [Connect to Your Instance](#) di Panduan Pengguna Amazon EC2.
4. Gunakan CloudHSM CLI untuk membuat pengguna krypto (CU). Lacak nama pengguna dan kata sandi CU. Anda akan membutuhkannya untuk menyelesaikan langkah berikutnya.

Note

Untuk informasi tentang cara membuat pengguna, lihat [Mengelola pengguna HSM dengan CloudHSM CLI](#).

5. [Atur kredensial login untuk HSM](#), menggunakan nama pengguna dan kata sandi CU yang Anda buat di langkah sebelumnya.
6. Pada langkah 5, jika Anda menggunakan Windows Credentials Manager untuk mengatur kredensial HSM, unduh [psexec.exe](#) dari SysInternals untuk menjalankan perintah berikut sebagai NT Authority\SYSTEM:

```
psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"  
--username <USERNAME> --password <PASSWORD>
```

Ganti <USERNAME> dan <PASSWORD> dengan kredensial HSM.

Untuk menginstal IIS pada Windows Server Anda

1. Jika Anda belum melakukannya, hubungkan ke server Windows server Anda. Untuk informasi selengkapnya, lihat [Connect to Your Instance](#) di Panduan Pengguna Amazon EC2.
2. Pada server Windows Anda, mulai Manager Server.
3. Di dasbor Server Manager, pilih Tambahkan peran dan fitur.
4. Baca informasi Sebelum Anda memulai, kemudian pilih Selanjutnya.
5. Untuk Jenis Instalasi, pilih Instalasi berbasis peran atau berbasis fitur. Lalu, pilih Selanjutnya.
6. Untuk Pemilihan Server, pilih Pilih server dari kolam server. Lalu, pilih Selanjutnya.
7. Untuk Peran Server, lakukan hal berikut:
 - a. Pilih Server Web (IIS).
 - b. Untuk Tambahkan fitur yang diperlukan untuk Web Server (IIS) Pilih Tambahkan fitur.
 - c. Pilih Selanjutnya untuk menyelesaikan pemilihan peran server.
8. Untuk Fitur, terima default. Lalu, pilih Selanjutnya.
9. Baca informasi Peran Server Web (IIS). Lalu, pilih Selanjutnya.
10. Untuk Pilih layanan peran, terima default atau ubah pengaturan sesuai pilihan. Lalu, pilih Selanjutnya.
11. Untuk Konfirmasi, baca informasi konfirmasi. Lalu, pilih Instal.
12. Setelah penginstalan selesai, pilih Tutup.

Setelah Anda menyelesaikan langkah ini, buka [Langkah 2: Buat permintaan penandatanganan sertifikat \(CSR\) dan sertifikat](#).

Langkah 2: Buat permintaan penandatanganan sertifikat (CSR) dan sertifikat

Untuk mengaktifkan HTTPS, server web Anda memerlukan SSL/TLS sertifikat dan kunci pribadi yang sesuai. Untuk menggunakan SSL/TLS offload dengan AWS CloudHSM, Anda menyimpan kunci pribadi di HSM di cluster Anda AWS CloudHSM . Untuk melakukannya, Anda menggunakan [penyedia penyimpanan kunci \(KSP\)AWS CloudHSM untuk API Kriptografi Microsoft: Next Generation \(CNG\)](#) untuk membuat permintaan penandatanganan sertifikat (CSR). Kemudian, Anda memberikan CSR ke otoritas sertifikat (CA), yang menandatangani CSR untuk menghasilkan sertifikat.

Topik

- [Membuat CSR dengan Client SDK 5](#)

- [Membuat CSR dengan Client SDK 3](#)
- [Dapatkan sertifikat yang ditandatangani dan impor](#)

Membuat CSR dengan Client SDK 5

1. Pada Windows Server Anda, gunakan editor teks untuk membuat file permintaan sertifikat bernama `IISCertRequest.inf`. Hal berikut menunjukkan isi contoh file `IISCertRequest.inf`. Untuk informasi lebih lanjut tentang bagian, kunci, dan nilai-nilai yang dapat Anda tentukan dalam file, lihat [Dokumentasi Microsoft](#). Jangan ubah nilai `ProviderName`.

```
[Version]
Signature = "$Windows NT$"
[NewRequest]
Subject = "CN=example.com,C=US,ST=Washington,L=Seattle,O=ExampleOrg,OU=WebServer"
HashAlgorithm = SHA256
KeyAlgorithm = RSA
KeyLength = 2048
ProviderName = "CloudHSM Key Storage Provider"
KeyUsage = 0xf0
MachineKeySet = True
[EnhancedKeyUsageExtension]
OID=1.3.6.1.5.5.7.3.1
```

2. Gunakan [certreqperintah Windows](#) untuk membuat CSR dari `IISCertRequest.inf` file yang Anda buat pada langkah sebelumnya. Contoh berikut menyimpan CSR ke file bernama `IISCertRequest.csr`. Jika Anda menggunakan nama file yang berbeda untuk file permintaan sertifikat Anda, ganti *`IISCertRequest.inf`* dengan nama file yang sesuai. Anda dapat mengganti secara opsional *`IISCertRequest.csr`* dengan nama file yang berbeda untuk file CSR Anda.

```
C:\>certreq -new IISCertRequest.inf IISCertRequest.csr

CertReq: Request Created
```

File `IISCertRequest.csr` berisi CSR Anda. Anda memerlukan CSR ini untuk mendapatkan sertifikat yang ditandatangani.

Membuat CSR dengan Client SDK 3

1. Jika Anda belum melakukannya, hubungkan ke server Windows server Anda. Untuk informasi selengkapnya, lihat [Connect to Your Instance](#) di Panduan Pengguna Amazon EC2.
2. Gunakan perintah berikut untuk memulai daemon AWS CloudHSM klien.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Untuk klien Windows 1.1.2+:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Untuk klien Windows 1.1.1 dan yang lebih lama:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe  
C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

3. Pada Windows Server Anda, gunakan editor teks untuk membuat file permintaan sertifikat bernama `IISCertRequest.inf`. Hal berikut menunjukkan isi contoh file `IISCertRequest.inf`. Untuk informasi lebih lanjut tentang bagian, kunci, dan nilai-nilai yang dapat Anda tentukan dalam file, lihat [Dokumentasi Microsoft](#). Jangan ubah nilai `ProviderName`.

```
[Version]  
Signature = "$Windows NT$"  
[NewRequest]  
Subject = "CN=example.com,C=US,ST=Washington,L=Seattle,O=ExampleOrg,OU=WebServer"  
HashAlgorithm = SHA256  
KeyAlgorithm = RSA  
KeyLength = 2048  
ProviderName = "Cavium Key Storage Provider"  
KeyUsage = 0xf0  
MachineKeySet = True  
[EnhancedKeyUsageExtension]  
OID=1.3.6.1.5.5.7.3.1
```

4. Gunakan [certreqperintah Windows](#) untuk membuat CSR dari `IISCertRequest.inf` file yang Anda buat pada langkah sebelumnya. Contoh berikut menyimpan CSR ke file bernama `IISCertRequest.csr`. Jika Anda menggunakan nama file yang berbeda untuk file permintaan sertifikat Anda, ganti *`IISCertRequest.inf`* dengan nama file yang sesuai. Anda dapat mengganti secara opsional *`IISCertRequest.csr`* dengan nama file yang berbeda untuk file CSR Anda.

```
C:\>certreq -new IISCertRequest.inf IISCertRequest.csr  
SDK Version: 2.03  
  
CertReq: Request Created
```

File `IISCertRequest.csr` berisi CSR Anda. Anda memerlukan CSR ini untuk mendapatkan sertifikat yang ditandatangani.

Dapatkan sertifikat yang ditandatangani dan impor

Di lingkungan produksi, Anda biasanya menggunakan sertifikat otoritas (CA) untuk membuat sertifikat dari CSR. CA tidak diperlukan untuk lingkungan pengujian. Jika Anda menggunakan CA, kirim file CSR (`IISCertRequest.csr`) ke sana dan gunakan CA untuk membuat SSL/TLS sertifikat yang ditandatangani.

Sebagai alternatif untuk menggunakan CA, Anda dapat menggunakan alat seperti [OpenSSL](#) untuk membuat sertifikat yang ditandatangani sendiri.

 Warning

Sertifikat yang ditandatangani sendiri tidak dipercaya oleh peramban dan tidak boleh digunakan dalam lingkungan produksi. Sertifikat dapat digunakan dalam lingkungan pengujian.

Prosedur berikut menunjukkan cara membuat sertifikat yang ditandatangani sendiri dan menggunakannya untuk menandatangani CSR server web Anda.

Untuk membuat sertifikat yang ditandatangani sendiri

1. Gunakan perintah OpenSSL berikut untuk membuat kunci privat. Anda dapat secara opsional mengganti `SelfSignedCA.key` dengan nama file untuk memuat kunci pribadi Anda.

```
openssl genrsa -aes256 -out SelfSignedCA.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
Enter pass phrase for SelfSignedCA.key:
Verifying - Enter pass phrase for SelfSignedCA.key:
```

2. Gunakan berikut ini perintah OpenSSL untuk membuat sertifikat yang ditandatangani sendiri menggunakan kunci privat yang Anda buat pada langkah sebelumnya. Ini adalah perintah interaktif. Baca petunjuk di layar dan ikuti prompt-nya. Ganti `SelfSignedCA.key` dengan

nama file yang berisi kunci pribadi Anda (jika berbeda). Anda dapat secara opsional mengganti *SelfSignedCA.crt* dengan nama file untuk berisi sertifikat yang ditandatangani sendiri.

```
openssl req -new -x509 -days 365 -key SelfSignedCA.key -out SelfSignedCA.crt
Enter pass phrase for SelfSignedCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

Untuk menggunakan sertifikat yang ditandatangani sendiri untuk menandatangani CSR server web

- Gunakan perintah OpenSSL berikut untuk menggunakan kunci privat dan sertifikat yang ditandatangani sendiri untuk menandatangani CSR. Ganti yang berikut dengan nama file yang berisi data yang sesuai (jika berbeda).
- *IISCertRequest.csr*— Nama file yang berisi CSR server web Anda
- *SelfSignedCA.crt*— Nama file yang berisi sertifikat yang ditandatangani sendiri
- *SelfSignedCA.key*— Nama file yang berisi kunci pribadi Anda
- *IISCert.crt*— Nama file yang berisi sertifikat yang ditandatangani server web Anda

```
openssl x509 -req -days 365 -in IISCertRequest.csr \
    -CA SelfSignedCA.crt \
    -CAkey SelfSignedCA.key \
    -CAcreateserial \
    -out IISCert.crt

Signature ok
subject=/ST=IIS-HSM/L=IIS-HSM/OU=IIS-HSM/O=IIS-HSM/CN=IIS-HSM/C=IIS-HSM
Getting CA Private Key
```

Enter pass phrase for SelfSignedCA.key:

Setelah Anda menyelesaikan langkah sebelumnya, Anda memiliki sertifikat yang ditandatangani untuk server web Anda (IISCert.crt) dan sertifikat yang ditandatangani sendiri (SelfSignedCA.crt). Bila Anda memiliki file-file ini, buka [Langkah 3: Konfigurasi server web](#).

Langkah 3: Konfigurasi server web

Perbarui konfigurasi situs web IIS Anda untuk menggunakan sertifikat HTTPS yang Anda buat pada akhir [langkah sebelumnya](#). Ini akan menyelesaikan pengaturan perangkat lunak server web Windows (IIS) Anda untuk SSL/TLS dibongkar. AWS CloudHSM

Jika Anda menggunakan sertifikat yang ditandatangani sendiri untuk menandatangani CSR Anda, Anda harus terlebih dahulu mengimpor sertifikat yang ditandatangani sendiri ke Otoritas sertifikasi root tepercaya Windows.

Untuk mengimpor sertifikat ditandatangani sendiri ke Otoritas Sertifikasi Tepercaya Windows

1. Jika Anda belum melakukannya, hubungkan ke server Windows server Anda. Untuk informasi selengkapnya, lihat [Connect to Your Instance](#) di Panduan Pengguna Amazon EC2.
2. Salin sertifikat yang ditandatangani sendiri ke server Windows Anda.
3. Pada Windows Server, buka Panel Kontrol.
4. Untuk Cari Panel Kontrol, ketik **certificates**. Kemudian, pilih Kelola sertifikat komputer.
5. Di jendela Certificates - Local Computer, klik dua kali Otoritas Sertifikasi Root Tepercaya.
6. Klik kanan pada Sertifikat lalu pilih Semua Tugas, Impor.
7. Di Wizzard Impor Sertifikat, Pilih Selanjutnya.
8. Pilih Jelajahi, lalu cari dan pilih sertifikat yang Anda tandatangi sendiri. Jika Anda membuat sertifikat yang ditandatangani sendiri dengan mengikuti petunjuk di [langkah sebelumnya dari tutorial ini](#), sertifikat yang Anda tandatangi sendiri diberi nama SelfSignedCA.crt. Pilih Buka .
9. Pilih Selanjutnya.
10. Untuk Penyimpanan Sertifikat, pilih Tempatkan semua sertifikat di penyimpanan berikut. Kemudian, pastikan bahwa Otoritas Sertifikasi Root Tepercaya dipilih untuk Penyimpanan sertifikat.
11. Pilih Selanjutnya dan kemudian pilih Selesai.

Untuk memperbarui konfigurasi situs web IIS

1. Jika Anda belum melakukannya, hubungkan ke server Windows server Anda. Untuk informasi selengkapnya, lihat [Connect to Your Instance](#) di Panduan Pengguna Amazon EC2.
2. Mulai daemon AWS CloudHSM klien.
3. Salin sertifikat yang ditandatangani oleh server web Anda—sertifikat yang Anda buat di akhir [langkah sebelumnya tutorial ini](#)—ke server Windows Anda.
4. Di Windows Server Anda, gunakan [certreqperintah Windows](#) untuk menerima sertifikat yang ditandatangani, seperti pada contoh berikut. Ganti *IISCert.crt* dengan nama file yang berisi sertifikat yang ditandatangani server web Anda.

```
C:\>certreq -accept IISCert.crt
SDK Version: 2.03
```

5. Pada server Windows Anda, mulai Manager Server.
6. Di dasbor Server Manager, di pojok kanan atas, pilih Alat, Manajer Layanan Informasi Internet (IIS).
7. Di jendela Manajer Layanan Informasi Internet (IIS), klik dua kali nama server Anda. Kemudian, klik dua kali Situs. Pilih situs web Anda.
8. Pilih Pengaturan SSL. Lalu, di sisi kanan jendela, pilih Ikatan.
9. Di jendela Ikatan Situs, pilih Tambahkan.
10. Untuk Jenis, pilih HTTPS. Untuk Sertifikat SSL, pilih sertifikat HTTPS yang Anda buat di akhir [langkah sebelumnya tutorial ini](#).

Note

Jika Anda mengalami kesalahan selama pengikatan sertifikat ini, mulai ulang server Anda dan coba lagi langkah ini.

11. Pilih OK.

Setelah memperbarui konfigurasi server web, buka [Langkah 4: Aktifkan lalu lintas HTTPS dan verifikasi sertifikat](#).

Langkah 4: Aktifkan lalu lintas HTTPS dan verifikasi sertifikat

Setelah Anda mengonfigurasi server web untuk SSL/TLS dibongkar AWS CloudHSM, tambahkan instance server web Anda ke grup keamanan yang memungkinkan lalu lintas HTTPS masuk. Hal ini memungkinkan klien, seperti peramban web, untuk membuat koneksi HTTPS dengan server web Anda. Kemudian buat koneksi HTTPS ke server web Anda dan verifikasi bahwa itu menggunakan sertifikat yang Anda konfigurasikan untuk SSL/TLS dibongkar. AWS CloudHSM

Topik

- [Aktifkan koneksi HTTPS masuk](#)
- [Verifikasi bahwa HTTPS menggunakan sertifikat yang Anda konfigurasikan](#)

Aktifkan koneksi HTTPS masuk

Untuk menyambungkan ke server web Anda dari klien (seperti peramban web), buat grup keamanan yang mengizinkan koneksi HTTPS masuk. Secara khusus, ini harus mengizinkan koneksi TCP masuk pada port 443. Tetapkan grup keamanan ini ke server web Anda.

Untuk membuat grup keamanan untuk HTTPS dan menentukannya ke server web Anda

1. Buka konsol Amazon EC2 di <https://console.aws.amazon.com/ec2/>
2. Pilih Grup keamanan di panel navigasi.
3. Pilih Buat grup keamanan.
4. Untuk Buat Grup Keamanan, lakukan hal berikut:
 - a. Untuk Nama grup keamanan, ketik nama untuk grup keamanan yang Anda buat.
 - b. (Opsional) Ketik deskripsi grup keamanan yang Anda buat.
 - c. Untuk VPC, pilih VPC yang berisi instans Amazon EC2 server web Anda.
 - d. Pilih Tambahkan Aturan.
 - e. Untuk Type, pilih HTTPS dari jendela drop-down.
 - f. Untuk Sumber, masukkan lokasi sumber.
 - g. Pilih Buat grup keamanan.
5. Di panel navigasi, pilih Instans.
6. Pilih kotak centang di samping instans server web Anda.

7. Pilih menu tarik-turun Tindakan di bagian atas halaman. Pilih Keamanan dan kemudian Ubah Grup Keamanan.
8. Untuk grup keamanan terkait, pilih kotak pencarian dan pilih grup keamanan yang Anda buat untuk HTTPS. Kemudian pilih Tambahkan Grup Keamanan.
9. Pilih Simpan.

Verifikasi bahwa HTTPS menggunakan sertifikat yang Anda konfigurasi

Setelah menambahkan server web ke grup keamanan, Anda dapat memverifikasi bahwa SSL/TLS offload menggunakan sertifikat yang ditandatangani sendiri. Anda dapat melakukan ini dengan peramban web atau dengan alat seperti [OpenSSL s_client](#).

Untuk memverifikasi SSL/TLS offload dengan browser web

1. Gunakan browser web untuk terhubung ke server web Anda menggunakan nama DNS publik atau alamat IP server. Pastikan URL di bilah alamat dimulai dengan https://. Misalnya, **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**.

 Tip

Anda dapat menggunakan layanan DNS seperti Amazon Route 53 untuk merutekan nama domain situs web Anda (misalnya, `https://www.example.com/`) ke server web Anda. Untuk informasi selengkapnya, lihat [Perutean Lalu Lintas ke Instans Amazon EC2](#) dalam Panduan Developer Amazon Route 53 atau dalam dokumentasi untuk layanan DNS Anda.

2. Gunakan peramban web Anda untuk melihat sertifikat server web. Untuk informasi lebih lanjut, lihat hal berikut:
 - Para Mozilla Firefox, lihat [Lihat Sertifikat](#) di situs web Dukung Mozilla.
 - Untuk Google Chrome, lihat [Memahami Masalah Keamanan](#) pada Alat Google Tools untuk Pengembang Web.

Peramban web lain mungkin memiliki fitur serupa yang dapat Anda gunakan untuk melihat sertifikat server web.

3. Pastikan bahwa SSL/TLS sertifikat adalah salah satu yang Anda konfigurasi server web Anda untuk digunakan.

Untuk memverifikasi SSL/TLS offload dengan OpenSSL s_client

1. Jalankan perintah OpenSSL berikut untuk terhubung ke server web Anda menggunakan HTTPS. Ganti `<server name>` dengan nama DNS publik atau alamat IP server web Anda.

```
openssl s_client -connect <server name>:443
```

 Tip

Anda dapat menggunakan layanan DNS seperti Amazon Route 53 untuk merutekan nama domain situs web Anda (misalnya, <https://www.example.com/>) ke server web Anda. Untuk informasi selengkapnya, lihat [Perutean Lalu Lintas ke Instans Amazon EC2](#) dalam Panduan Developer Amazon Route 53 atau dalam dokumentasi untuk layanan DNS Anda.

2. Pastikan bahwa SSL/TLS sertifikat adalah salah satu yang Anda konfigurasi server web Anda untuk digunakan.

Anda sekarang memiliki situs web yang diamankan dengan HTTPS. Kunci pribadi untuk server web disimpan dalam HSM di AWS CloudHSM cluster Anda.

Untuk menambahkan penyeimbang beban, lihat [Tambahkan penyeimbang beban dengan Elastic Load Balancing AWS CloudHSM untuk \(opsional\)](#).

Tambahkan penyeimbang beban dengan Elastic Load Balancing AWS CloudHSM untuk (opsional)

Setelah mengatur SSL/TLS offload dengan satu server web, Anda dapat membuat lebih banyak server web dan penyeimbang beban Elastic Load Balancing yang mengarahkan lalu lintas HTTPS ke server web. Penyeimbang beban dapat mengurangi beban pada server web individual Anda dengan menyeimbangkan lalu lintas di dua atau lebih server. Hal ini juga dapat meningkatkan ketersediaan situs web Anda karena penyeimbang beban memantau kondisi server web Anda dan hanya merutekan lalu lintas ke server yang sehat. Jika server web gagal, penyeimbang beban secara otomatis berhenti routing lalu lintas ke sana.

Topik

- [Langkah 1. Buat subnet untuk server web kedua](#)

- [Langkah 2. Buat server web kedua](#)
- [Langkah 3. Buat penyeimbang beban](#)

Langkah 1. Buat subnet untuk server web kedua

Sebelum Anda dapat membuat server web lain, Anda perlu membuat subnet baru di VPC yang sama yang berisi server AWS CloudHSM web dan cluster yang ada.

Untuk membuat subnet baru

1. Buka [bagian Subnet dari konsol VPC Amazon](#).
2. Pilih Buat Subnet.
3. Pada kotak dialog Buat Subnet, lakukan hal berikut:
 - a. Untuk Tanda Nama, ketikkan nama untuk subnet Anda.
 - b. Untuk VPC, pilih AWS CloudHSM VPC yang berisi server web dan cluster yang ada. AWS CloudHSM
 - c. Untuk Availability Zone, pilih Availability Zone yang berbeda dari availability zone yang berisi server web Anda yang sudah ada.
 - d. Untuk IPv4 Blok CIDR, ketik blok CIDR yang akan digunakan untuk subnet. Sebagai contoh, ketik **10.0.10.0/24**.
 - e. Pilih Ya, Buat.
4. Centang kotak di samping subnet publik yang berisi server web Anda yang sudah ada. Ini berbeda dengan subnet publik yang Anda buat pada langkah sebelumnya.
5. Di panel konten, pilih tab Tabel Rute. Lalu, pilih tautan untuk tabel rute.

subnet-1f358d78 | CloudHSM Public subnet

Summary	Route Table	Network ACL
Edit		
Route Table:		rtb-cea112a9
Destination	Target	
10.0.0.0/16	local	
0.0.0.0/0	igw-68ee440c	

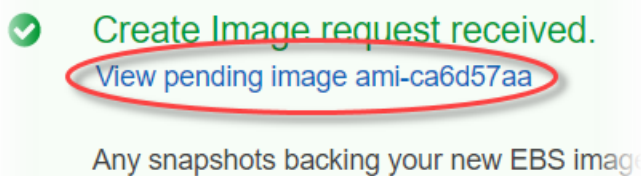
6. Centang kotak di samping tabel rute.
7. Pilih tab Asosiasi Subnet. Lalu, pilih Edit.
8. Centang kotak di samping subnet publik yang telah Anda buat sebelumnya dalam prosedur ini. Lalu, pilih Simpan.

Langkah 2. Buat server web kedua

Selesaikan langkah-langkah berikut untuk membuat server web kedua dengan konfigurasi yang sama seperti server web Anda yang ada.

Untuk membuat server web kedua

1. Buka bagian [Instans](#) pada konsol Amazon EC2 di.
2. Centang kotak di samping instans server web Anda yang sudah ada.
3. Pilih Tindakan, Gambar, dan kemudian Buat Gambar.
4. Dalam kotak dialog Buat Gambar, lakukan hal berikut:
 - a. Untuk Nama gambar, ketikkan nama untuk gambar.
 - b. Untuk Deskripsi gambar, ketikkan deskripsi untuk gambar.
 - c. Pilih Buat Gambar. Tindakan ini me-reboot server web Anda yang ada.
 - d. Pilih **<AMI ID>** tautan Lihat gambar yang tertunda.



Di kolom Status, catat status gambar Anda. Saat status gambar Anda tersedia (ini mungkin memakan waktu beberapa menit), lanjutkan ke langkah berikutnya.

5. Di panel navigasi, pilih Instans.
6. Centang kotak di samping server web Anda yang sudah ada.
7. Pilih Tindakan dan pilih Luncurkan Lebih Banyak Seperti Ini.
8. Pilih Edit AMI.

▼ AMI Details

[Edit AMI](#)**amzn-ami-hvm-2017.09.1.20171120-x86_64-gp2 - ami-a51f27c5**

Amazon Linux AMI 2017.09.1.20171120 x86_64 HVM GP2

Root Device Type: ebs Virtualization type: hvm

9. Di panel navigasi kiri, pilih My AMIs. Lalu, hapus teks di kotak pencarian.
10. Di samping gambar server web Anda, pilih Pilih.
11. Pilih Ya, saya ingin melanjutkan dengan AMI ini (**<image name>**- ami-**<AMI ID>**).
12. Pilih Berikutnya.
13. Pilih tipe instans, lalu pilih Berikutnya: Detail Instans Konfigurasi.
14. Untuk Langkah 3: Konfigurasi Detail Instans, lakukan hal berikut:
 - a. Untuk Jaringan, pilih VPC yang berisi server web Anda yang ada.
 - b. Untuk Subnet, pilih subnet publik yang telah Anda buat untuk server web kedua.
 - c. Untuk Tetapkan Otomatis IP Publik, pilih Aktifkan.
 - d. Ubah detail instans yang tersisa sesuai pilihan. Lalu, pilih Berikutnya: Tambahkan Penyimpanan.
15. Ubah pengaturan penyimpanan sesuai pilihan. Lalu, pilih Berikutnya: Tambahkan Tanda.
16. Tambahkan atau edit tanda sesuai pilihan. Lalu, pilih Berikutnya: Konfigurasikan Grup Keamanan.
17. Untuk Langkah 6: Konfigurasikan Grup Keamanan, lakukan hal berikut:
 - a. Untuk Tetapkan grup keamanan, pilih Pilih grup keamanan yang sudah ada.
 - b. Pilih kotak centang di samping grup keamanan bernama cloudhsm- **<cluster ID>** - sg. AWS CloudHSM membuat grup keamanan ini atas nama Anda saat Anda [membuat klaster](#). Anda harus memilih grup keamanan ini untuk memungkinkan instance server web terhubung ke HSMs dalam cluster.
 - c. Centang kotak di samping grup keamanan yang memungkinkan lalu lintas HTTPS masuk. Anda [membuat grup keamanan ini sebelumnya](#).
 - d. (Opsional) Pilih kotak centang di sebelah grup keamanan yang memungkinkan lalu lintas SSH masuk (untuk Linux) atau RDP (untuk Windows) dari jaringan Anda. Artinya, grup keamanan harus mengizinkan lalu lintas TCP masuk pada port 22 (untuk SSH di Linux) atau port 3389 (untuk RDP pada Windows). Jika tidak, Anda tidak dapat menyambung ke instans

klien Anda. Jika Anda tidak memiliki grup keamanan seperti ini, Anda harus membuat satu dan kemudian menetapkan ke instans klien Anda nanti.

Pilih Tinjau dan Luncurkan.

18. Tinjau detail instans Anda, lalu pilih Luncurkan.
19. Pilih apakah akan meluncurkan instans Anda dengan pasangan kunci yang sudah ada, membuat pasangan kunci baru, atau meluncurkan instans tanpa pasangan kunci.
 - Untuk menggunakan pasangan kunci yang sudah ada, lakukan hal berikut:
 1. Pilih Pasangan kunci yang sudah ada.
 2. Untuk Pilih pasangan kunci, pilih pasangan kunci yang akan digunakan.
 3. Pilih kotak centang di sebelah Saya mengakui bahwa saya memiliki akses ke file kunci pribadi yang dipilih (**<private key file name>**.pem), dan bahwa tanpa file ini, saya tidak akan dapat masuk ke instance saya.
 - Untuk membuat pasangan kunci baru, lakukan hal berikut:
 1. Pilih Buat pasangan kunci baru.
 2. Untuk Nama pasangan kunci, ketik nama pasangan kunci.
 3. Pilih Unduh Pasangan Kunci dan simpan file kunci privat di lokasi yang aman dan mudah diakses.



Warning

Anda tidak dapat mengunduh file kunci privat lagi setelah titik ini. Jika Anda tidak mengunduh file kunci privat sekarang, Anda tidak akan dapat mengakses instans klien.

- Untuk meluncurkan instans tanpa pasangan kunci, lakukan hal berikut:
 1. Pilih Lanjutkan tanpa pasangan kunci.
 2. Centang kotak di samping Saya menyatakan bahwa saya tidak akan dapat terhubung ke contoh ini kecuali saya sudah tahu kata sandi yang ada di AMI ini.

Pilih Luncurkan Instans.

Langkah 3. Buat penyeimbang beban

Selesaikan langkah-langkah berikut untuk membuat penyeimbang beban Elastic Load Balancing yang merutekan lalu lintas HTTPS ke server web Anda.

Untuk membuat penyeimbang beban

1. Buka bagian [Load balancer](#) di konsol Amazon EC2.
2. Pilih Buat Penyeimbang Beban.
3. Di bagian Penyeimbang Beban Jaringan, pilih Buat.
4. Untuk Langkah 1: Konfigurasi Penyeimbang Beban, lakukan hal berikut:
 - a. Untuk Nama, ketik nama untuk penyeimbang beban yang Anda buat.
 - b. Di bagian Listener, untuk Port Penyeimbang Beban, ubah nilai menjadi **443**.
 - c. Di bagian Availability Zone, untuk VPC, pilih VPC yang berisi server web Anda.
 - d. Di bagian Availability Zone, pilih subnet yang berisi server web Anda.
 - e. Pilih Selanjutnya: Konfigurasi Perutean.
5. Untuk Langkah 2: Konfigurasi Perutean, lakukan hal berikut:
 - a. Untuk Nama, ketik nama untuk grup target yang Anda buat.
 - b. Untuk Port, ubah nilai ke **443**.
 - c. Pilih Selanjutnya: Daftarkan Target.
6. Untuk Langkah 3: Daftarkan Target, lakukan hal berikut:
 - a. Di bagian Instans, pilih kotak centang di sebelah instans server web Anda. Lalu, pilih Tambahkan ke terdaftar.
 - b. Pilih Selanjutnya: Tinjau.
7. Tinjau detail penyeimbang beban Anda, lalu pilih Buat.
8. Bila penyeimbang beban telah berhasil dibuat, pilih Tutup.

Setelah Anda menyelesaikan langkah-langkah sebelumnya, konsol Amazon EC2 menunjukkan penyeimbang beban Elastic Load Balancing Anda.

Bila kondisi penyeimbang beban Anda aktif, Anda dapat memverifikasi bahwa penyeimbang beban bekerja. Artinya, Anda dapat memverifikasi bahwa itu mengirim lalu lintas HTTPS ke server

web Anda dengan SSL/TLS offload dengan AWS CloudHSM. Anda dapat melakukan ini dengan peramban web atau alat seperti [OpenSSL s_client](#).

Untuk memverifikasi bahwa penyeimbang beban Anda bekerja dengan peramban web

1. Di konsol Amazon EC2, temukan Nama DNS untuk penyeimbang beban yang baru saja Anda buat. Kemudian, pilih nama DNS dan salin.
2. Gunakan peramban web seperti Mozilla Firefox atau Google Chrome untuk terhubung ke penyeimbang beban Anda menggunakan nama DNS penyeimbang beban. Pastikan URL di bilah alamat dimulai dengan https://.

 Tip

Anda dapat menggunakan layanan DNS seperti Amazon Route 53 untuk merutekan nama domain situs web Anda (misalnya, <https://www.example.com/>) ke server web Anda. Untuk informasi selengkapnya, lihat [Perutean Lalu Lintas ke Instans Amazon EC2](#) dalam Panduan Developer Amazon Route 53 atau dalam dokumentasi untuk layanan DNS Anda.

3. Gunakan peramban web Anda untuk melihat sertifikat server web. Untuk informasi lebih lanjut, lihat hal berikut:
 - Para Mozilla Firefox, lihat [Lihat Sertifikat](#) di situs web Dukung Mozilla.
 - Untuk Google Chrome, lihat [Memahami Masalah Keamanan](#) pada Alat Google Tools untuk Pengembang Web.

Peramban web lain mungkin memiliki fitur serupa yang dapat Anda gunakan untuk melihat sertifikat server web.

4. Pastikan bahwa sertifikat adalah yang telah Anda atur konfigurasi web servernya untuk digunakan.

Untuk memverifikasi bahwa penyeimbang beban Anda bekerja dengan OpenSSL s_client

1. Gunakan perintah OpenSSL berikut untuk terhubung ke penyeimbang beban Anda menggunakan HTTPS. Ganti *<DNS name>* dengan nama DNS penyeimbang beban Anda.

```
openssl s_client -connect <DNS name>:443
```

Tip

Anda dapat menggunakan layanan DNS seperti Amazon Route 53 untuk merutekan nama domain situs web Anda (misalnya, <https://www.example.com/>) ke server web Anda. Untuk informasi selengkapnya, lihat [Perutean Lalu Lintas ke Instans Amazon EC2](#) dalam Panduan Developer Amazon Route 53 atau dalam dokumentasi untuk layanan DNS Anda.

2. Pastikan bahwa sertifikat adalah yang telah Anda atur konfigurasi web servernya untuk digunakan.

Anda sekarang memiliki situs web yang diamankan dengan HTTPS, dengan kunci pribadi server web disimpan dalam HSM di cluster Anda AWS CloudHSM. Situs web Anda memiliki dua server web dan penyeimbang beban untuk membantu meningkatkan efisiensi dan ketersediaan.

Konfigurasi Windows Server sebagai otoritas sertifikat (CA) dengan AWS CloudHSM

AWS CloudHSM menawarkan dukungan untuk mengkonfigurasi Windows Server sebagai otoritas sertifikat (CA) melalui Client SDK 3 dan Client SDK 5. Langkah-langkah untuk menggunakan alat ini akan bervariasi tergantung pada versi SDK klien yang saat ini telah Anda unduh. Bagian berikut memberikan informasi untuk setiap SDK.

Topik

- [Konfigurasi Windows Server sebagai otoritas sertifikat \(CA\) dengan Client SDK 5](#)
- [Konfigurasi Windows Server sebagai otoritas sertifikat \(CA\) dengan Client SDK 3](#)

Konfigurasi Windows Server sebagai otoritas sertifikat (CA) dengan Client SDK 5

Dalam infrastruktur kunci publik (PKI), otoritas sertifikat (CA) adalah entitas tepercaya yang mengeluarkan sertifikat digital. Sertifikat digital ini mengikat kunci publik dengan suatu identitas (orang atau organisasi) dengan cara kriptografi kunci publik dan tanda tangan digital. Untuk mengoperasikan CA, Anda harus mempertahankan kepercayaan dengan melindungi kunci privat

yang menandatangani sertifikat yang dikeluarkan oleh CA Anda. Anda dapat menyimpan kunci privat di HSM di klaster AWS CloudHSM, dan menggunakan HSM untuk melakukan operasi penandatanganan kriptografi.

Dalam tutorial ini, Anda menggunakan Windows Server dan AWS CloudHSM untuk mengkonfigurasi CA. Instal perangkat lunak klien AWS CloudHSM untuk Windows di server Windows, kemudian tambahkan peran Active Directory Certificate Services (AD CS) ke Windows Server Anda. Saat mengonfigurasi peran ini, Anda menggunakan penyedia penyimpanan AWS CloudHSM kunci (KSP) untuk membuat dan menyimpan kunci pribadi CA di AWS CloudHSM klaster Anda. KSP adalah jembatan yang menghubungkan server Windows Anda ke AWS CloudHSM cluster Anda. Pada langkah terakhir, Anda menandatangani permintaan penandatanganan sertifikat (CSR) dengan Windows Server CA.

Untuk informasi selengkapnya, lihat topik berikut:

Topik

- [Langkah 1: Siapkan prasyarat](#)
- [Langkah 2: Buat CA Windows Server dengan AWS CloudHSM](#)
- [Langkah 3: Menandatangani permintaan penandatanganan sertifikat \(CSR\) dengan Windows Server CA Anda AWS CloudHSM](#)

Langkah 1: Siapkan prasyarat

Untuk mengatur Windows Server sebagai otoritas sertifikat (CA) dengan AWS CloudHSM, Anda memerlukan yang berikut ini:

- AWS CloudHSM Cluster aktif dengan setidaknya satu HSM.
- Instans Amazon EC2 yang menjalankan sistem operasi Windows Server dengan perangkat lunak AWS CloudHSM klien untuk Windows diinstal. Tutorial ini menggunakan Microsoft Windows Server 2016.
- Pengguna kriptografi (CU) harus memiliki dan mengelola kunci privat CA pada HSM.

Untuk mengatur prasyarat untuk Windows Server CA dengan AWS CloudHSM

1. Selesaikan langkah-langkah dalam [Mulai menggunakan](#). Ketika Anda meluncurkan klien Amazon EC2, pilih Windows Server AMI. Tutorial ini menggunakan Microsoft Windows Server 2016. Ketika Anda menyelesaikan langkah-langkah ini, Anda memiliki klaster aktif dengan

setidaknya satu HSM. Anda juga memiliki instans klien Amazon EC2 yang menjalankan Windows Server dengan perangkat lunak AWS CloudHSM klien untuk Windows diinstal.

2. (Opsional) Tambahkan lebih banyak HSMs ke cluster Anda. Untuk informasi selengkapnya, lihat [Menambahkan HSM ke cluster AWS CloudHSM](#).
3. Hubungkan ke instans klien Anda. Untuk informasi selengkapnya, lihat [Connect to Your Instance](#) di Panduan Pengguna Amazon EC2.
4. Buat pengguna krypto (CU) menggunakan [Mengelola pengguna HSM dengan CloudHSM CLI](#) atau [Mengelola pengguna HSM dengan CloudHSM Management Utility \(CMU\)](#). Lacak nama pengguna dan kata sandi CU. Anda akan membutuhkannya untuk menyelesaikan langkah berikutnya.
5. [Atur kredensial login untuk HSM](#), menggunakan nama pengguna dan kata sandi CU yang Anda buat di langkah sebelumnya.
6. Pada langkah 5, jika Anda menggunakan Windows Credentials Manager untuk mengatur kredensi HSM, unduh [psexec.exe](#) dari SysInternals untuk menjalankan perintah berikut sebagai NT Authority\ SYSTEM:

```
psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"  
--username <USERNAME> --password <PASSWORD>
```

Ganti **<USERNAME>** dan **<PASSWORD>** dengan kredensi HSM.

Untuk membuat Windows Server CA dengan AWS CloudHSM, buka [Buat Windows Server CA](#).

Langkah 2: Buat CA Windows Server dengan AWS CloudHSM

Untuk membuat Windows Server CA, Anda menambahkan peran Active Directory Certificate Services (AD CS) ke Windows Server. Saat menambahkan peran ini, Anda menggunakan penyedia penyimpanan AWS CloudHSM kunci (KSP) untuk membuat dan menyimpan kunci pribadi CA di AWS CloudHSM klaster Anda.

Note

Ketika Anda membuat Windows Server CA, Anda dapat memilih untuk membuat CA root atau CA bawahan. Anda biasanya membuat keputusan ini berdasarkan desain infrastruktur kunci publik dan kebijakan keamanan organisasi Anda. Tutorial ini menjelaskan cara membuat root CA untuk kesederhanaan.

Untuk menambahkan peran AD CS ke Windows Server dan membuat kunci privat CA

1. Jika Anda belum melakukannya, hubungkan ke server Windows server Anda. Untuk informasi selengkapnya, lihat [Connect to Your Instance](#) di Panduan Pengguna Amazon EC2.
2. Pada server Windows Anda, mulai Manager Server.
3. Di dasbor Server Manager, pilih Tambahkan peran dan fitur.
4. Baca informasi Sebelum Anda memulai, kemudian pilih Selanjutnya.
5. Untuk Jenis Instalasi, pilih Instalasi berbasis peran atau berbasis fitur. Lalu, pilih Selanjutnya.
6. Untuk Pemilihan Server, pilih Pilih server dari kolam server. Lalu, pilih Selanjutnya.
7. Untuk Peran Server, lakukan hal berikut:
 - a. Pilih Layanan Sertifikat Active Directory.
 - b. Untuk Tambahkan fitur yang diperlukan untuk Active Directory Sertifikat Services, pilih Tambah Fitur.
 - c. Pilih Selanjutnya untuk menyelesaikan pemilihan peran server.
8. Untuk Fitur, terima default, dan kemudian pilih Selanjutnya.
9. Untuk AD CS, lakukan hal berikut:
 - a. Pilih Selanjutnya.
 - b. Pilih Otoritas Sertifikasi, lalu pilih Selanjutnya.
10. Untuk Konfirmasi, baca informasi konfirmasi, dan kemudian pilih Instal. Jangan tutup jendela.
11. Pilih tautan yang disorot Atur Konfigurasi Layanan Active Directory Certificate Services di server tujuan.
12. Untuk Kredensial, verifikasi atau ubah kredensial yang ditampilkan. Lalu, pilih Selanjutnya.
13. Untuk Peran Layanan, pilih Otoritas Sertifikasi. Lalu, pilih Selanjutnya.
14. Untuk Jenis Pengaturan, pilih CA Mandiri. Lalu, pilih Selanjutnya.
15. Untuk Jenis CA, pilih CA Root. Lalu, pilih Selanjutnya.

 Note

Anda dapat memilih untuk membuat CA root atau CA bawahan berdasarkan desain infrastruktur kunci publik Anda dan kebijakan keamanan organisasi Anda. Tutorial ini menjelaskan cara membuat root CA untuk kesederhanaan.

16. Untuk Kunci Privat, pilih Buat kunci privat baru. Lalu, pilih Selanjutnya.
17. Untuk Kriptografi, lakukan hal berikut:
 - a. Untuk Pilih penyedia kriptografi, pilih salah satu opsi Penyedia Penyimpanan Kunci CloudHSM dari menu. Ini adalah penyedia penyimpanan kunci AWS CloudHSM . Misalnya, Anda dapat memilih Penyedia Penyimpanan Kunci RSA #CloudHSM.
 - b. Untuk Panjang kunci, pilih salah satu opsi panjang kunci.
 - c. Untuk Pilih algoritme hash untuk menandatangani sertifikat yang dikeluarkan oleh CA ini, pilih salah satu opsi algoritme hash.

Pilih Selanjutnya.

18. Untuk Nama CA, lakukan hal berikut:
 - a. (Opsional) Edit nama umum.
 - b. (Opsional) Ketik akhiran nama berbeda.

Pilih Selanjutnya.

19. Untuk Masa Berlaku, tentukan jangka waktu dalam tahun, bulan, minggu, atau hari. Lalu, pilih Selanjutnya.
20. Untuk Basis Data Sertifikat, Anda dapat menerima nilai default, atau secara opsional mengubah lokasi untuk basis data dan log basis data. Lalu, pilih Selanjutnya.
21. Untuk Konfirmasi, tinjau informasi tentang CA Anda; Kemudian pilih Konfigurasi.
22. Pilih Tutup, lalu pilih Tutup lagi.

Anda sekarang memiliki Windows Server CA dengan AWS CloudHSM. Untuk mempelajari cara menandatangani permintaan penandatanganan sertifikat (CSR) dengan CA Anda, kunjungi [Menandatangani CSR](#).

Langkah 3: Menandatangani permintaan penandatanganan sertifikat (CSR) dengan Windows Server CA Anda AWS CloudHSM

Anda dapat menggunakan CA Windows Server Anda AWS CloudHSM untuk menandatangani permintaan penandatanganan sertifikat (CSR). Untuk menyelesaikan langkah-langkah ini, Anda memerlukan CSR yang valid. Anda dapat membuat CSR dengan beberapa cara, termasuk yang berikut:

- Menggunakan OpenSSL
- Menggunakan Manajer Layanan Informasi Internet (IIS) Windows Server
- Menggunakan sertifikat snap-in di Konsol Manajemen Microsoft
- Menggunakan utilitas baris perintah certreq pada Windows

Langkah-langkah untuk membuat CSR berada di luar cakupan tutorial ini. Bila Anda memiliki CSR, Anda dapat menandatangani dengan Windows Server CA.

Untuk menandatangani CSR dengan Windows Server CA

1. Jika Anda belum melakukannya, hubungkan ke server Windows server Anda. Untuk informasi selengkapnya, lihat [Connect to Your Instance](#) di Panduan Pengguna Amazon EC2.
2. Pada server Windows Anda, mulai Manager Server.
3. Di dasbor Manajer Server, di pojok kanan atas, pilih Alat, Otoritas Sertifikasi.
4. Di jendela Otoritas Sertifikasi, pilih nama komputer Anda.
5. Dari menu Tindakan, pilih Semua Tugas, Kirim permintaan baru.
6. Pilih file CSR Anda, lalu pilih Buka.
7. Di jendela Otoritas Sertifikasi, klik dua kali Permintaan Menunggu Keputusan.
8. Pilih permintaan yang tertunda. Kemudian, dari menu Tindakan, pilih Semua Tugas, Terbitkan.
9. Di jendela Otoritas Sertifikasi, klik dua kali Permintaan Diterbitkan untuk melihat sertifikat ditandatangani.
10. (Opsional) Untuk mengekspor sertifikat yang ditandatangani ke file, selesaikan langkah-langkah berikut:
 - a. Di jendela Otoritas Sertifikasi, klik dua kali sertifikat.
 - b. Pilih tab Detail, dan kemudian pilih Salin ke File.
 - c. Ikuti instruksi di Wizard Ekspor Sertifikat.

Anda sekarang memiliki Windows Server CA dengan AWS CloudHSM, dan sertifikat yang valid ditandatangani oleh Windows Server CA.

Konfigurasi Windows Server sebagai otoritas sertifikat (CA) dengan Client SDK 3

Dalam infrastruktur kunci publik (PKI), otoritas sertifikat (CA) adalah entitas tepercaya yang mengeluarkan sertifikat digital. Sertifikat digital ini mengikat kunci publik dengan suatu identitas (orang atau organisasi) dengan cara kriptografi kunci publik dan tanda tangan digital. Untuk mengoperasikan CA, Anda harus mempertahankan kepercayaan dengan melindungi kunci privat yang menandatangani sertifikat yang dikeluarkan oleh CA Anda. Anda dapat menyimpan kunci privat di HSM di kluster AWS CloudHSM, dan menggunakan HSM untuk melakukan operasi penandatanganan kriptografi.

Dalam tutorial ini, Anda menggunakan Windows Server dan AWS CloudHSM untuk mengkonfigurasi CA. Instal perangkat lunak klien AWS CloudHSM untuk Windows di server Windows, kemudian tambahkan peran Active Directory Certificate Services (AD CS) ke Windows Server Anda. Saat mengonfigurasi peran ini, Anda menggunakan penyedia penyimpanan AWS CloudHSM kunci (KSP) untuk membuat dan menyimpan kunci pribadi CA di AWS CloudHSM kluster Anda. KSP adalah jembatan yang menghubungkan server Windows Anda ke AWS CloudHSM cluster Anda. Pada langkah terakhir, Anda menandatangani permintaan penandatanganan sertifikat (CSR) dengan Windows Server CA.

Untuk informasi selengkapnya, lihat topik berikut:

Topik

- [Langkah 1: Siapkan prasyarat](#)
- [Langkah 2: Buat CA Windows Server dengan AWS CloudHSM](#)
- [Langkah 3: Menandatangani permintaan penandatanganan sertifikat \(CSR\) dengan Windows Server CA Anda AWS CloudHSM](#)

Langkah 1: Siapkan prasyarat

Untuk mengatur Windows Server sebagai otoritas sertifikat (CA) dengan AWS CloudHSM, Anda memerlukan yang berikut ini:

- AWS CloudHSM Cluster aktif dengan setidaknya satu HSM.
- Instans Amazon EC2 yang menjalankan sistem operasi Windows Server dengan perangkat lunak AWS CloudHSM klien untuk Windows diinstal. Tutorial ini menggunakan Microsoft Windows Server 2016.

- Pengguna kriptografi (CU) harus memiliki dan mengelola kunci privat CA pada HSM.

Untuk mengatur prasyarat untuk Windows Server CA dengan AWS CloudHSM

1. Selesaikan langkah-langkah dalam [Mulai menggunakan](#). Ketika Anda meluncurkan klien Amazon EC2, pilih Windows Server AMI. Tutorial ini menggunakan Microsoft Windows Server 2016. Ketika Anda menyelesaikan langkah-langkah ini, Anda memiliki kluster aktif dengan setidaknya satu HSM. Anda juga memiliki instans klien Amazon EC2 yang menjalankan Windows Server dengan perangkat lunak AWS CloudHSM klien untuk Windows diinstal.
2. (Opsional) Tambahkan lebih banyak HSMs ke cluster Anda. Untuk informasi selengkapnya, lihat [Menambahkan HSM ke cluster AWS CloudHSM](#).
3. Hubungkan ke instans klien Anda. Untuk informasi selengkapnya, lihat [Connect to Your Instance](#) di Panduan Pengguna Amazon EC2.
4. Buat pengguna krypto (CU) menggunakan [Mengelola pengguna HSM dengan CloudHSM CLI](#) atau [Mengelola pengguna HSM dengan CloudHSM Management Utility \(CMU\)](#). Lacak nama pengguna dan kata sandi CU. Anda akan membutuhkannya untuk menyelesaikan langkah berikutnya.
5. [Atur kredensial login untuk HSM](#), menggunakan nama pengguna dan kata sandi CU yang Anda buat di langkah sebelumnya.
6. Pada langkah 5, jika Anda menggunakan Windows Credentials Manager untuk mengatur kredensi HSM, unduh [psexec.exe](#) dari SysInternals untuk menjalankan perintah berikut sebagai NT Authority\ SYSTEM:

```
psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools  
\set_cloudhsm_credentials.exe" --username <USERNAME> --password <PASSWORD>
```

Ganti **<USERNAME>** dan **<PASSWORD>** dengan kredensi HSM.

Untuk membuat Windows Server CA dengan AWS CloudHSM, buka [Buat Windows Server CA](#).

Langkah 2: Buat CA Windows Server dengan AWS CloudHSM

Untuk membuat Windows Server CA, Anda menambahkan peran Active Directory Certificate Services (AD CS) ke Windows Server. Saat menambahkan peran ini, Anda menggunakan penyedia

penyimpanan AWS CloudHSM kunci (KSP) untuk membuat dan menyimpan kunci pribadi CA di AWS CloudHSM klaster Anda.

 Note

Ketika Anda membuat Windows Server CA, Anda dapat memilih untuk membuat CA root atau CA bawahan. Anda biasanya membuat keputusan ini berdasarkan desain infrastruktur kunci publik dan kebijakan keamanan organisasi Anda. Tutorial ini menjelaskan cara membuat root CA untuk kesederhanaan.

Untuk menambahkan peran AD CS ke Windows Server dan membuat kunci privat CA

1. Jika Anda belum melakukannya, hubungkan ke server Windows server Anda. Untuk informasi selengkapnya, lihat [Connect to Your Instance](#) di Panduan Pengguna Amazon EC2.
2. Pada server Windows Anda, mulai Manager Server.
3. Di dasbor Server Manager, pilih Tambahkan peran dan fitur.
4. Baca informasi Sebelum Anda memulai, kemudian pilih Selanjutnya.
5. Untuk Jenis Instalasi, pilih Instalasi berbasis peran atau berbasis fitur. Lalu, pilih Selanjutnya.
6. Untuk Pemilihan Server, pilih Pilih server dari kolam server. Lalu, pilih Selanjutnya.
7. Untuk Peran Server, lakukan hal berikut:
 - a. Pilih Layanan Sertifikat Active Directory.
 - b. Untuk Tambahkan fitur yang diperlukan untuk Active Directory Sertifikat Services, pilih Tambah Fitur.
 - c. Pilih Selanjutnya untuk menyelesaikan pemilihan peran server.
8. Untuk Fitur, terima default, dan kemudian pilih Selanjutnya.
9. Untuk AD CS, lakukan hal berikut:
 - a. Pilih Selanjutnya.
 - b. Pilih Otoritas Sertifikasi, lalu pilih Selanjutnya.
10. Untuk Konfirmasi, baca informasi konfirmasi, dan kemudian pilih Instal. Jangan tutup jendela.
11. Pilih tautan yang disorot Atur Konfigurasi Layanan Active Directory Certificate Services di server tujuan.
12. Untuk Kredensial, verifikasi atau ubah kredensial yang ditampilkan. Lalu, pilih Selanjutnya.

13. Untuk Peran Layanan, pilih Otoritas Sertifikasi. Lalu, pilih Selanjutnya.
14. Untuk Jenis Pengaturan, pilih CA Mandiri. Lalu, pilih Selanjutnya.
15. Untuk Jenis CA, pilih CA Root. Lalu, pilih Selanjutnya.

 Note

Anda dapat memilih untuk membuat CA root atau CA bawahan berdasarkan desain infrastruktur kunci publik Anda dan kebijakan keamanan organisasi Anda. Tutorial ini menjelaskan cara membuat root CA untuk kesederhanaan.

16. Untuk Kunci Privat, pilih Buat kunci privat baru. Lalu, pilih Selanjutnya.
17. Untuk Kriptografi, lakukan hal berikut:
 - a. Untuk Pilih penyedia kriptografi, pilih salah satu dari opsi Penyedia Penyimpanan Kunci Cavium dari menu. Ini adalah penyedia penyimpanan kunci AWS CloudHSM . Misalnya, Anda dapat memilih Penyedia Penyimpanan Kunci RSA#Cavium.
 - b. Untuk Panjang kunci, pilih salah satu opsi panjang kunci.
 - c. Untuk Pilih algoritme hash untuk menandatangani sertifikat yang dikeluarkan oleh CA ini, pilih salah satu opsi algoritme hash.

Pilih Selanjutnya.

18. Untuk Nama CA, lakukan hal berikut:
 - a. (Opsional) Edit nama umum.
 - b. (Opsional) Ketik akhiran nama berbeda.

Pilih Selanjutnya.

19. Untuk Masa Berlaku, tentukan jangka waktu dalam tahun, bulan, minggu, atau hari. Lalu, pilih Selanjutnya.
20. Untuk Basis Data Sertifikat, Anda dapat menerima nilai default, atau secara opsional mengubah lokasi untuk basis data dan log basis data. Lalu, pilih Selanjutnya.
21. Untuk Konfirmasi, tinjau informasi tentang CA Anda; Kemudian pilih Konfigurasi.
22. Pilih Tutup, lalu pilih Tutup lagi.

Anda sekarang memiliki Windows Server CA dengan AWS CloudHSM. Untuk mempelajari cara menandatangani permintaan penandatanganan sertifikat (CSR) dengan CA Anda, kunjungi [Menandatangani CSR](#).

Langkah 3: Menandatangani permintaan penandatanganan sertifikat (CSR) dengan Windows Server CA Anda AWS CloudHSM

Anda dapat menggunakan CA Windows Server Anda AWS CloudHSM untuk menandatangani permintaan penandatanganan sertifikat (CSR). Untuk menyelesaikan langkah-langkah ini, Anda memerlukan CSR yang valid. Anda dapat membuat CSR dengan beberapa cara, termasuk yang berikut:

- Menggunakan OpenSSL
- Menggunakan Manajer Layanan Informasi Internet (IIS) Windows Server
- Menggunakan sertifikat snap-in di Konsol Manajemen Microsoft
- Menggunakan utilitas baris perintah certreq pada Windows

Langkah-langkah untuk membuat CSR berada di luar cakupan tutorial ini. Bila Anda memiliki CSR, Anda dapat menandatangani dengan Windows Server CA.

Untuk menandatangani CSR dengan Windows Server CA

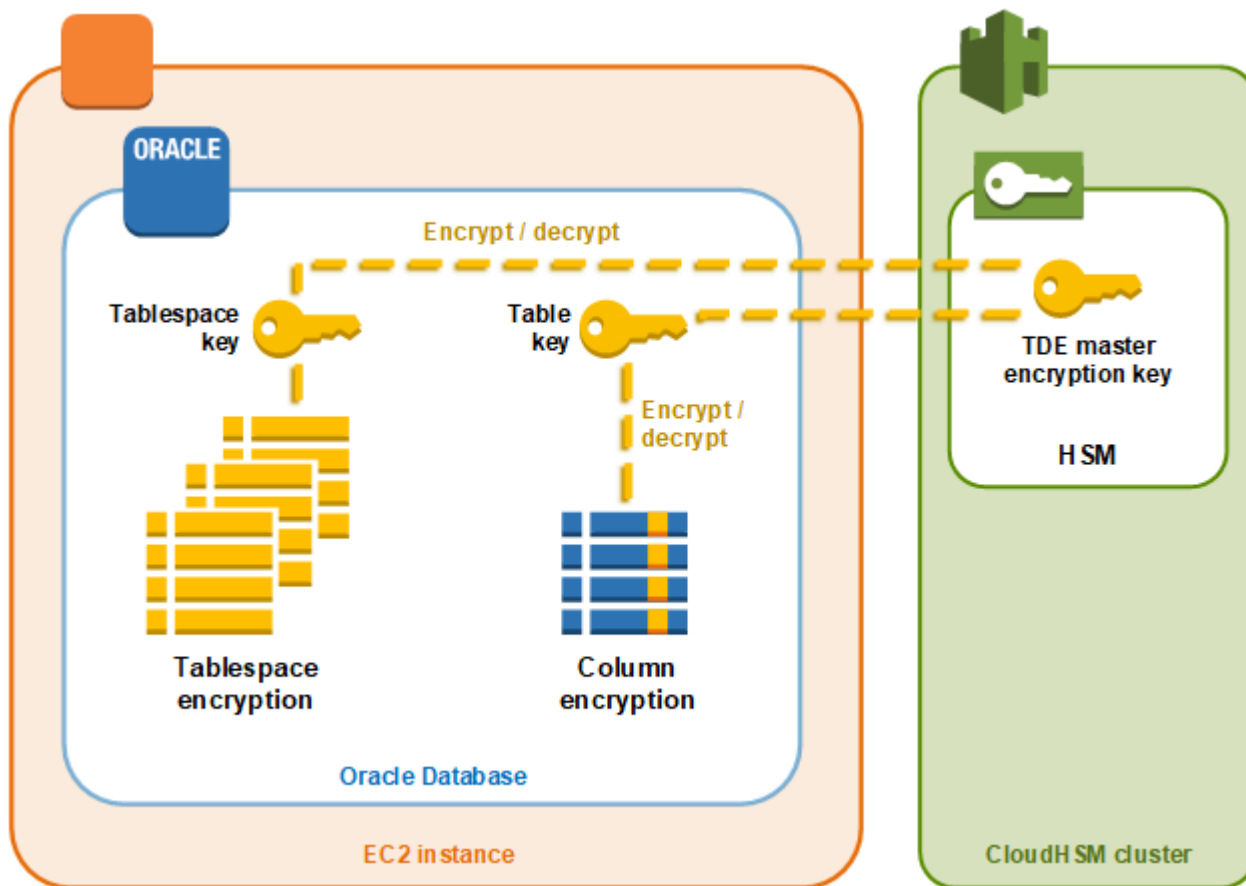
1. Jika Anda belum melakukannya, hubungkan ke server Windows server Anda. Untuk informasi selengkapnya, lihat [Connect to Your Instance](#) di Panduan Pengguna Amazon EC2.
2. Pada server Windows Anda, mulai Manager Server.
3. Di dasbor Manajer Server, di pojok kanan atas, pilih Alat, Otoritas Sertifikasi.
4. Di jendela Otoritas Sertifikasi, pilih nama komputer Anda.
5. Dari menu Tindakan, pilih Semua Tugas, Kirim permintaan baru.
6. Pilih file CSR Anda, lalu pilih Buka.
7. Di jendela Otoritas Sertifikasi, klik dua kali Permintaan Menunggu Keputusan.
8. Pilih permintaan yang tertunda. Kemudian, dari menu Tindakan, pilih Semua Tugas, Terbitkan.
9. Di jendela Otoritas Sertifikasi, klik dua kali Permintaan Diterbitkan untuk melihat sertifikat ditandatangani.
10. (Opsional) Untuk mengekspor sertifikat yang ditandatangani ke file, selesaikan langkah-langkah berikut:

- a. Di jendela Otoritas Sertifikasi, klik dua kali sertifikat.
- b. Pilih tab Detail, dan kemudian pilih Salin ke File.
- c. Ikuti instruksi di Wizard Ekspor Sertifikat.

Anda sekarang memiliki Windows Server CA dengan AWS CloudHSM, dan sertifikat yang valid ditandatangani oleh Windows Server CA.

Enkripsi data transparan (TDE) database Oracle dengan AWS CloudHSM

Transparent Data Encryption (TDE) digunakan untuk mengenkripsi file database. Menggunakan TDE, perangkat lunak database mengenkripsi data sebelum menyimpannya di disk. Data dalam kolom tabel database atau ruang tabel dienkripsi dengan tombol tabel atau tombol tablespace. Beberapa versi perangkat lunak database Oracle menawarkan TDE. Di Oracle TDE, kunci ini dienkripsi dengan kunci enkripsi master TDE. Anda dapat mencapai keamanan yang lebih besar dengan menyimpan kunci enkripsi master TDE HSMs di AWS CloudHSM cluster Anda.



Dalam solusi ini, Anda menggunakan Oracle Database yang diinstal pada instans Amazon EC2. Oracle Database terintegrasi dengan [pustaka AWS CloudHSM perangkat lunak untuk PKCS #11](#) untuk menyimpan kunci master TDE di cluster Anda. HSMs

 Important

- Sebaiknya instal Oracle Database pada instans Amazon EC2.

Selesaikan langkah-langkah berikut untuk menyelesaikan integrasi Oracle TDE dengan AWS CloudHSM.

Topik

- [Langkah 1. Siapkan prasyarat](#)
- [Langkah 2: Perbarui konfigurasi database Oracle](#)
- [Langkah 3: Hasilkan kunci enkripsi master Oracle TDE](#)

Langkah 1. Siapkan prasyarat

Untuk mencapai integrasi Oracle TDE dengan AWS CloudHSM, Anda memerlukan yang berikut:

- AWS CloudHSM Cluster aktif dengan setidaknya satu HSM.
- Instans Amazon EC2 menjalankan sistem operasi Amazon Linux dengan perangkat lunak berikut diinstal:
 - AWS CloudHSM Klien dan alat baris perintah.
 - Pustaka AWS CloudHSM perangkat lunak untuk PKCS #11.
 - Basis Data Oracle. AWS CloudHSM mendukung integrasi Oracle TDE. Klien SDK 5.6 dan lebih tinggi mendukung Oracle TDE untuk Oracle Database 19c. Client SDK 3 mendukung Oracle TDE untuk Oracle Database versi 11g dan 12c.
- Pengguna kriptografi (CU) untuk memiliki dan mengelola kunci enkripsi master TDE HSMs di cluster Anda.

Selesaikan langkah-langkah berikut untuk menyiapkan semua prasyarat.

Untuk mengatur prasyarat untuk integrasi Oracle TDE dengan AWS CloudHSM

1. Selesaikan langkah-langkah dalam [Mulai menggunakan](#). Setelah Anda melakukannya, Anda akan memiliki kluster aktif dengan satu HSM. Anda juga akan memiliki instans Amazon EC2 menjalankan sistem operasi Amazon Linux. AWS CloudHSM Klien dan alat baris perintah juga akan diinstal dan dikonfigurasi.
2. (Opsional) Tambahkan lebih banyak HSMs ke cluster Anda. Untuk informasi selengkapnya, lihat [Menambahkan HSM ke cluster AWS CloudHSM](#).
3. Hubungkan ke instans klien Amazon EC2 Anda dan lakukan hal berikut:
 - a. [Instal pustaka AWS CloudHSM perangkat lunak untuk PKCS #11](#).
 - b. Instal Oracle Database. Lihat informasi yang lebih lengkap dalam [dokumentasi Oracle Database](#). Klien SDK 5.6 dan lebih tinggi mendukung Oracle TDE untuk Oracle Database 19c. Client SDK 3 mendukung Oracle TDE untuk Oracle Database versi 11g dan 12c.
 - c. Gunakan alat baris perintah `cloudhsm_mgmt_util` untuk membuat pengguna kriptografi (CU) di kluster Anda. Untuk informasi selengkapnya tentang cara membuat CU, lihat [Cara Mengelola Pengguna HSM dengan CMU](#) dan [Pengguna HSM](#).

Langkah 2: Perbarui konfigurasi database Oracle

Untuk memperbarui konfigurasi Oracle Database untuk menggunakan HSM di kluster Anda sebagai modul keamanan eksternal, selesaikan langkah-langkah berikut. Untuk informasi tentang modul keamanan eksternal, lihat [Pengenalan Data Transparan](#) di Panduan Keamanan Tingkat Lanjut Oracle Database.

Memperbarui Konfigurasi Oracle Database

1. Hubungkan ke instans klien Amazon EC2 Anda. Ini adalah contoh tempat Anda menginstal Oracle Database.
2. Buat salinan cadangan dari file bernama `sqlnet.ora`. Untuk lokasi file ini, lihat dokumentasi Oracle.
3. Gunakan editor teks untuk mengedit file bernama `sqlnet.ora`. Tambahkan baris berikut. Jika baris yang ada dalam file dimulai dengan `encryption_wallet_location`, ganti baris yang ada dengan yang berikut.

```
encryption_wallet_location=(source=(method=hsm))
```

Simpan file tersebut.

4. Jalankan perintah berikut untuk membuat direktori tempat Oracle Database mengharapkan untuk menemukan file perpustakaan untuk pustaka perangkat lunak AWS CloudHSM PKCS #11.

```
sudo mkdir -p /opt/oracle/extapi/64/hsm
```

5. Jalankan perintah berikut untuk menyalin pustaka AWS CloudHSM perangkat lunak untuk file PKCS #11 ke direktori yang Anda buat pada langkah sebelumnya.

```
sudo cp /opt/cloudhsm/lib/libcloudhsm_pkcs11.so /opt/oracle/extapi/64/hsm/
```

 Note

Direktori `/opt/oracle/extapi/64/hsm` harus berisi hanya satu file pustaka. Hapus file lain yang ada di direktori tersebut.

6. Jalankan perintah berikut untuk mengubah kepemilikan direktori `/opt/oracle` dan segala sesuatu di dalamnya.

```
sudo chown -R oracle:dba /opt/oracle
```

7. Mulai Oracle Database.

Langkah 3: Hasilkan kunci enkripsi master Oracle TDE

Untuk menghasilkan kunci master Oracle TDE HSMs di cluster Anda, selesaikan langkah-langkah dalam prosedur berikut.

Untuk menghasilkan kunci utama

1. Gunakan perintah berikut untuk membuka Oracle SQL* Plus. Saat diminta, ketik sandi sistem yang Anda tetapkan saat Anda menginstal Oracle Database.

```
sqlplus / as sysdba
```

 Note

Untuk Client SDK 3, Anda harus mengatur variabel `CLOUDHSM_IGNORE_CKA_MODIFIABLE_FALSE` lingkungan setiap kali Anda membuat kunci master. Variabel ini hanya diperlukan untuk pembuatan kunci utama. Untuk informasi lebih lanjut, lihat “Masalah: Oracle menetapkan atribut PKCS #11 CKA_MODIFIABLE selama pembuatan kunci utama, tapi HSM tidak mendukungnya” di [Masalah yang Diketahui untuk Mengintegrasikan Aplikasi Pihak Ketiga](#).

2. Jalankan pernyataan SQL yang membuat kunci enkripsi master, seperti yang ditunjukkan dalam contoh berikut. Gunakan pernyataan yang sesuai dengan versi Oracle Database. Ganti `<CU user name>` dengan nama pengguna kriptografi (CU). Ganti `<password>` dengan kata sandi CU.

 Important

Jalankan perintah berikut hanya sekali. Setiap kali perintah dijalankan, itu membuat kunci enkripsi utama baru.

- Untuk Oracle Database versi 11, jalankan pernyataan SQL berikut.

```
SQL> alter system set encryption key identified by "<CU user name>:<password>";
```

- Untuk Oracle Database versi 12 dan versi 19c, jalankan pernyataan SQL berikut.

```
SQL> administer key management set key identified by "<CU user name>:<password>";
```

Jika respons adalah `System altered` atau `keystore altered`, maka Anda berhasil dihasilkan dan mengatur kunci utama untuk Oracle TDE.

3. (Opsional) Jalankan perintah berikut untuk memverifikasi status dompet Oracle.

```
SQL> select * from v$encryption_wallet;
```

Jika dompet tidak terbuka, gunakan salah satu perintah berikut untuk membukanya. Ganti *<CU user name>* dengan nama pengguna kriptografi (CU). Ganti *<password>* dengan kata sandi CU.

- Untuk Oracle 11, jalankan perintah berikut untuk membuka dompet.

```
SQL> alter system set encryption wallet open identified by "<CU user name>:<password>";
```

Untuk menutup dompet secara manual, jalankan perintah berikut.

```
SQL> alter system set encryption wallet close identified by "<CU user name>:<password>";
```

- Untuk Oracle 12 dan Oracle 19c, jalankan perintah berikut untuk membuka dompet.

```
SQL> administer key management set keystore open identified by "<CU user name>:<password>";
```

Untuk menutup dompet secara manual, jalankan perintah berikut.

```
SQL> administer key management set keystore close identified by "<CU user name>:<password>";
```

Gunakan Microsoft SignTool AWS CloudHSM untuk menandatangani file

AWS CloudHSM menawarkan dukungan untuk menggunakan Microsoft Signtool untuk menandatangani file melalui Client SDK 3 dan Client SDK 5. Langkah-langkah untuk menggunakan alat ini akan bervariasi tergantung pada versi SDK klien yang saat ini telah Anda unduh. Bagian berikut memberikan informasi untuk setiap SDK.

Topik

- [Gunakan Microsoft SignTool dengan Client SDK 5 untuk menandatangani file](#)
- [Gunakan Microsoft SignTool dengan Client SDK 3 untuk menandatangani file](#)

Gunakan Microsoft SignTool dengan Client SDK 5 untuk menandatangani file

Dalam kriptografi dan infrastruktur kunci publik (PKI), tanda tangan digital digunakan untuk mengonfirmasi bahwa data telah dikirim oleh entitas tepercaya. Tanda tangan juga menunjukkan bahwa data belum dirusak saat transit. Tanda tangan adalah hash terenkripsi yang dihasilkan dengan kunci privat pengirim. Penerima dapat memverifikasi integritas data dengan mendekripsi tanda tangan hash dengan kunci publik pengirim. Pada gilirannya, itu adalah tanggung jawab pengirim untuk mempertahankan sertifikat digital. Sertifikat digital menunjukkan kepemilikan pengirim kunci privat dan menyediakan penerima dengan kunci publik yang diperlukan untuk dekripsi. Selama kunci pribadi dimiliki oleh pengirim, tanda tangan dapat dipercaya. AWS CloudHSM menyediakan perangkat keras tervalidasi FIPS 140-2 level 3 yang aman bagi Anda untuk mengamankan kunci ini dengan akses penyewa tunggal eksklusif.

Banyak organisasi menggunakan Microsoft SignTool, alat baris perintah yang menandatangani, memverifikasi, dan stempel waktu file untuk menyederhanakan proses penandatanganan kode. Anda dapat menggunakan AWS CloudHSM untuk menyimpan pasangan kunci Anda dengan aman hingga dibutuhkan SignTool, sehingga menciptakan alur kerja yang mudah diotomatisasi untuk menandatangani data.

Topik berikut memberikan ikhtisar tentang cara menggunakannya SignTool AWS CloudHSM.

Topik

- [Langkah 1: Siapkan prasyarat](#)
- [Langkah 2: Buat sertifikat penandatanganan](#)
- [Langkah 3: Tanda tangani file](#)

Langkah 1: Siapkan prasyarat

Untuk menggunakan Microsoft SignTool dengan AWS CloudHSM, Anda memerlukan yang berikut ini:

- Instans klien Amazon EC2 yang menjalankan sistem operasi Windows.
- Otoritas sertifikat (CA), dikelola sendiri atau didirikan oleh penyedia pihak ketiga.
- AWS CloudHSM Cluster aktif di cloud publik virtual (VPC) yang sama dengan instans EC2 Anda. Klaster harus berisi setidaknya satu HSM.
- Pengguna crypto (CU) untuk memiliki dan mengelola kunci di AWS CloudHSM cluster.

- File tanpa tanda tangan atau dapat dijalankan.
- Kit Pengembangan Perangkat Lunak (SDK) Microsoft Windows.

Untuk mengatur prasyarat untuk digunakan dengan Windows AWS CloudHSM SignTool

1. Ikuti instruksi di bagian [Memulai](#) dari panduan ini untuk meluncurkan instans EC2 Windows dan klaster AWS CloudHSM .
2. Jika Anda ingin meng-host Windows Server CA Anda sendiri, ikuti langkah 1 dan 2 dalam [Mengkonfigurasi Windows Server sebagai Otoritas Sertifikat dengan AWS CloudHSM](#). Jika tidak, terus gunakan CA pihak ketiga tepercaya publik Anda.
3. Unduh dan instal salah satu versi SDK Microsoft Windows berikut pada instans EC2 Windows Anda:
 - [Microsoft Windows SDK 10](#)
 - [Microsoft Windows SDK 8.1](#)
 - [Microsoft Windows SDK 7](#)

SignTool yang dapat dijalankan adalah bagian dari Alat Tanda Tangan SDK Windows untuk fitur instalasi Aplikasi Desktop. Anda dapat menghilangkan fitur lain yang akan diinstal jika Anda tidak membutuhkannya. Lokasi instalasi default adalah:

```
C:\Program Files (x86)\Windows Kits\<SDK version>\bin\<version number>\<CPU architecture>\signtool.exe
```

Sekarang Anda dapat menggunakan Microsoft Windows SDK, AWS CloudHSM klaster, dan CA Anda untuk [Membuat Sertifikat Penandatanganan](#).

Langkah 2: Buat sertifikat penandatanganan

Sekarang karena Anda telah mengunduh SDK Windows ke instans EC2 Anda, Anda dapat menggunakannya untuk menghasilkan permintaan penandatanganan sertifikat (CSR). CSR adalah sertifikat tanpa tanda tangan yang akhirnya diteruskan ke CA Anda untuk ditandatangani. Dalam contoh ini, kami menggunakan `certreq` yang dapat dijalankan yang disertakan dengan SDK Windows untuk menghasilkan CSR.

Untuk menghasilkan CSR menggunakan **certreq** yang dapat dijalankan

1. Jika Anda belum melakukannya, sambungkan ke instans EC2 Windows Anda. Untuk informasi selengkapnya, lihat [Connect to Your Instance](#) di Panduan Pengguna Amazon EC2.
2. Buat file bernama `request.inf` yang berisi baris di bawah ini. Ganti informasi Subject dinformasi engan organisasi Anda. Untuk penjelasan tentang setiap parameter, lihat [Dokumentasi Microsoft](#).

```
[Version]
Signature= $Windows NT$
[NewRequest]
Subject = "C=<Country>,CN=<www.website.com>,O=<Organization>,OU=<Organizational-Unit>,L=<City>,S=<State>"
RequestType=PKCS10
HashAlgorithm = SHA256
KeyAlgorithm = RSA
KeyLength = 2048
ProviderName = "CloudHSM Key Storage Provider"
KeyUsage = "CERT_DIGITAL_SIGNATURE_KEY_USAGE"
MachineKeySet = True
Exportable = False
```

3. Jalankan `certreq.exe`. Untuk contoh ini, kami menyimpan CSR sebagai `request.csr`.

```
certreq.exe -new request.inf request.csr
```

Secara internal, key pair baru dihasilkan di AWS CloudHSM cluster Anda, dan kunci pribadi pasangan digunakan untuk membuat CSR.

4. Kirimkan CSR ke CA Anda. Jika Anda menggunakan CA Windows Server, ikuti langkah-langkah berikut:
 - a. Masukkan perintah berikut untuk menjalankan alat .

```
certsrv.msc
```

- b. Di jendela baru, klik kanan nama server CA. Pilih Semua Tugas, lalu pilih Kirim permintaan baru.
- c. Arahkan ke `request.csr` dan pilih Buka.

- d. Arahkan ke folder Permintaan menunggu keputusan dengan memperluas menu CA Server. Klik kanan permintaan yang baru Anda buat, dan di bawah Semua Tugas pilih Terbitkan.
- e. Sekarang arahkan ke folder Sertifikat Diterbitkan (di atas folder Permintaan menunggu keputusan).
- f. Pilih Buka untuk melihat sertifikat, dan kemudian memilih tab Detail.
- g. Pilih Salin ke File untuk memulai Wizard Ekspor Sertifikat. Simpan file Der-encoded X.509 ke lokasi aman sebagai `signedCertificate.cer`.
- h. Keluar dari alat CA dan gunakan perintah berikut, yang memindahkan file sertifikat ke Penyimpanan Sertifikat Pribadi di Windows. Ini kemudian dapat digunakan oleh aplikasi lain.

```
certreq.exe -accept signedCertificate.cer
```

Sekarang Anda dapat menggunakan sertifikat yang diimpor untuk [Tanda tangani File](#).

Langkah 3: Tanda tangani file

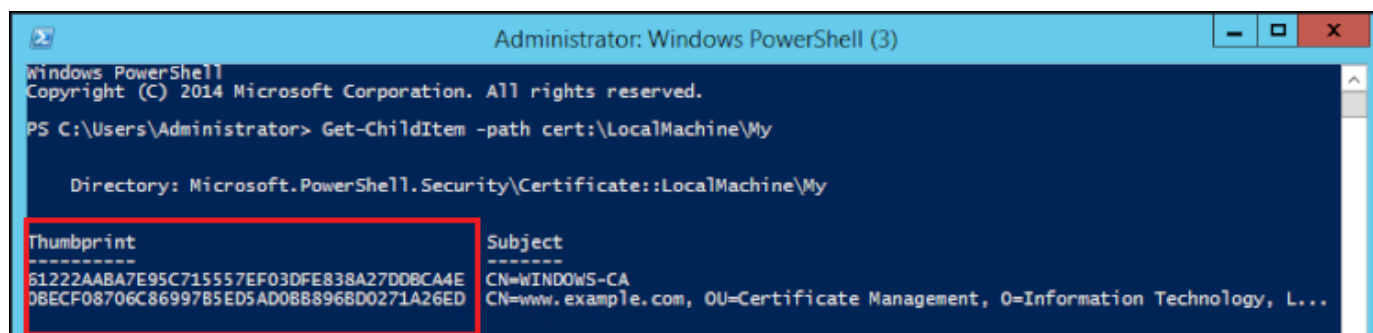
Anda sekarang siap untuk menggunakan SignTool dan sertifikat impor Anda untuk menandatangani file contoh Anda. Untuk melakukannya, Anda perlu mengetahui SHA-1 hash atau cap jempol. Sidik jari digunakan untuk memastikan bahwa SignTool hanya menggunakan sertifikat yang diverifikasi oleh AWS CloudHSM. Dalam contoh ini, kita gunakan PowerShell untuk mendapatkan hash sertifikat. Anda juga dapat menggunakan GUI CA atau `certutil` SDK Windows yang dapat dieksekusi.

Untuk mendapatkan cap jempol sertifikat dan menggunakannya untuk menandatangani file

1. Buka PowerShell sebagai administrator dan jalankan perintah berikut:

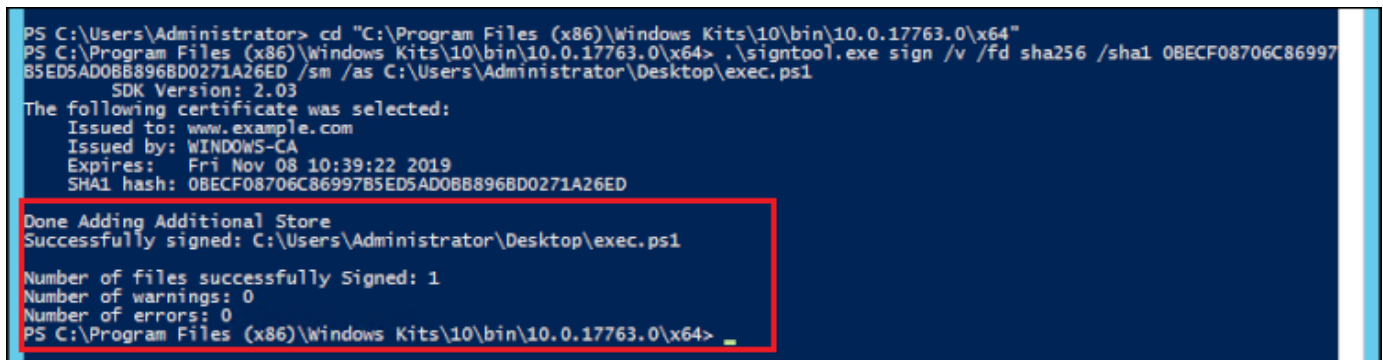
```
Get-ChildItem -path cert:\LocalMachine\My
```

Salin Thumbprint yang dikembalikan.



2. Arahkan ke direktori di dalamnya PowerShell berisi `SignTool.exe`. Lokasi default adalah `C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64`.
3. Terakhir, tanda tangani file Anda dengan menjalankan perintah berikut. Jika perintah berhasil, PowerShell mengembalikan pesan sukses.

```
signtool.exe sign /v /fd sha256 /sha1 <thumbprint> /sm C:\Users\Administrator\
\Desktop\<test>.ps1
```



```
PS C:\Users\Administrator> cd "C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64"
PS C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64> .\signtool.exe sign /v /fd sha256 /sha1 0BECF08706C86997
B5ED5AD0BB896BD0271A26ED /sm /as C:\Users\Administrator\Desktop\exec.ps1
    SDK Version: 2.03
The following certificate was selected:
    Issued to: www.example.com
    Issued by: WINDOWS-CA
    Expires:   Fri Nov 08 10:39:22 2019
    SHA1 hash: 0BECF08706C86997B5ED5AD0BB896BD0271A26ED
Done Adding Additional Store
Successfully signed: C:\Users\Administrator\Desktop\exec.ps1
Number of files successfully Signed: 1
Number of warnings: 0
Number of errors: 0
PS C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64>
```

4. (Opsional) Untuk memverifikasi tanda tangan pada file, gunakan perintah berikut:

```
signtool.exe verify /v /pa C:\Users\Administrator\Desktop\<test>.ps1
```

Gunakan Microsoft SignTool dengan Client SDK 3 untuk menandatangani file

Dalam kriptografi dan infrastruktur kunci publik (PKI), tanda tangan digital digunakan untuk mengonfirmasi bahwa data telah dikirim oleh entitas tepercaya. Tanda tangan juga menunjukkan bahwa data belum dirusak saat transit. Tanda tangan adalah hash terenkripsi yang dihasilkan dengan kunci privat pengirim. Penerima dapat memverifikasi integritas data dengan mendekripsi tanda tangan hash dengan kunci publik pengirim. Pada gilirannya, itu adalah tanggung jawab pengirim untuk mempertahankan sertifikat digital. Sertifikat digital menunjukkan kepemilikan pengirim kunci privat dan menyediakan penerima dengan kunci publik yang diperlukan untuk dekripsi. Selama kunci pribadi dimiliki oleh pengirim, tanda tangan dapat dipercaya. AWS CloudHSM menyediakan perangkat keras tervalidasi FIPS 140-2 level 3 yang aman bagi Anda untuk mengamankan kunci ini dengan akses penyewa tunggal eksklusif.

Banyak organisasi menggunakan Microsoft SignTool, alat baris perintah yang menandatangani, memverifikasi, dan stempel waktu file untuk menyederhanakan proses penandatanganan kode.

Anda dapat menggunakan AWS CloudHSM untuk menyimpan pasangan kunci Anda dengan aman hingga dibutuhkan SignTool, sehingga menciptakan alur kerja yang mudah diotomatisasi untuk menandatangani data.

Topik berikut memberikan ikhtisar tentang cara menggunakannya SignTool AWS CloudHSM.

Topik

- [Langkah 1: Siapkan prasyarat](#)
- [Langkah 2: Buat sertifikat penandatanganan](#)
- [Langkah 3: Tanda tangani file](#)

Langkah 1: Siapkan prasyarat

Untuk menggunakan Microsoft SignTool dengan AWS CloudHSM, Anda memerlukan yang berikut ini:

- Instans klien Amazon EC2 yang menjalankan sistem operasi Windows.
- Otoritas sertifikat (CA), dikelola sendiri atau didirikan oleh penyedia pihak ketiga.
- AWS CloudHSM Cluster aktif di cloud publik virtual (VPC) yang sama dengan instans EC2 Anda. Kluster harus berisi setidaknya satu HSM.
- Pengguna crypto (CU) untuk memiliki dan mengelola kunci di AWS CloudHSM cluster.
- File tanpa tanda tangan atau dapat dijalankan.
- Kit Pengembangan Perangkat Lunak (SDK) Microsoft Windows.

Untuk mengatur prasyarat untuk digunakan dengan Windows AWS CloudHSM SignTool

1. Ikuti instruksi di bagian [Memulai](#) dari panduan ini untuk meluncurkan instans EC2 Windows dan kluster AWS CloudHSM .
2. Jika Anda ingin meng-host Windows Server CA Anda sendiri, ikuti langkah 1 dan 2 dalam [Mengkonfigurasi Windows Server sebagai Otoritas Sertifikat dengan AWS CloudHSM](#). Jika tidak, terus gunakan CA pihak ketiga tepercaya publik Anda.
3. Unduh dan instal salah satu versi SDK Microsoft Windows berikut pada instans EC2 Windows Anda:
 - [Microsoft Windows SDK 10](#)
 - [Microsoft Windows SDK 8.1](#)
 - [Microsoft Windows SDK 7](#)

SignTool yang dapat dijalankan adalah bagian dari Alat Tanda Tangan SDK Windows untuk fitur instalasi Aplikasi Desktop. Anda dapat menghilangkan fitur lain yang akan diinstal jika Anda tidak membutuhkannya. Lokasi instalasi default adalah:

```
C:\Program Files (x86)\Windows Kits\<SDK version>\bin\<version number>\<CPU architecture>\signtool.exe
```

Sekarang Anda dapat menggunakan Microsoft Windows SDK, AWS CloudHSM klaster, dan CA Anda untuk [Membuat Sertifikat Penandatanganan](#).

Langkah 2: Buat sertifikat penandatanganan

Sekarang karena Anda telah mengunduh SDK Windows ke instans EC2 Anda, Anda dapat menggunakannya untuk menghasilkan permintaan penandatanganan sertifikat (CSR). CSR adalah sertifikat tanpa tanda tangan yang akhirnya diteruskan ke CA Anda untuk ditandatangani. Dalam contoh ini, kami menggunakan `certreq` yang dapat dijalankan yang disertakan dengan SDK Windows untuk menghasilkan CSR.

Untuk menghasilkan CSR menggunakan **certreq** yang dapat dijalankan

1. Jika Anda belum melakukannya, sambungkan ke instans EC2 Windows Anda. Untuk informasi selengkapnya, lihat [Connect to Your Instance](#) di Panduan Pengguna Amazon EC2.
2. Buat file bernama `request.inf` yang berisi baris di bawah ini. Ganti informasi Subject dinformasi engan organisasi Anda. Untuk penjelasan tentang setiap parameter, lihat [Dokumentasi Microsoft](#).

```
[Version]
Signature= $Windows NT$
[NewRequest]
Subject = "C=<Country>,CN=<www.website.com>,O=<Organization>,OU=<Organizational-Unit>,L=<City>,S=<State>"
RequestType=PKCS10
HashAlgorithm = SHA256
KeyAlgorithm = RSA
KeyLength = 2048
ProviderName = "Cavium Key Storage Provider"
KeyUsage = "CERT_DIGITAL_SIGNATURE_KEY_USAGE"
MachineKeySet = True
```

```
Exportable = False
```

3. Jalankan `certreq.exe`. Untuk contoh ini, kami menyimpan CSR sebagai `request.csr`.

```
certreq.exe -new request.inf request.csr
```

Secara internal, key pair baru dihasilkan di AWS CloudHSM cluster Anda, dan kunci pribadi pasangan digunakan untuk membuat CSR.

4. Kirimkan CSR ke CA Anda. Jika Anda menggunakan CA Windows Server, ikuti langkah-langkah berikut:

- a. Masukkan perintah berikut untuk menjalankan alat .

```
certsrv.msc
```

- b. Di jendela baru, klik kanan nama server CA. Pilih Semua Tugas, lalu pilih Kirim permintaan baru.
- c. Arahkan ke `request.csr` dan pilih Buka.
- d. Arahkan ke folder Permintaan menunggu keputusan dengan memperluas menu CA Server. Klik kanan permintaan yang baru Anda buat, dan di bawah Semua Tugas pilih Terbitkan.
- e. Sekarang arahkan ke folder Sertifikat Diterbitkan (di atas folder Permintaan menunggu keputusan).
- f. Pilih Buka untuk melihat sertifikat, dan kemudian memilih tab Detail.
- g. Pilih Salin ke File untuk memulai Wizard Ekspor Sertifikat. Simpan file Der-encoded X.509 ke lokasi aman sebagai `signedCertificate.cer`.
- h. Keluar dari alat CA dan gunakan perintah berikut, yang memindahkan file sertifikat ke Penyimpanan Sertifikat Pribadi di Windows. Ini kemudian dapat digunakan oleh aplikasi lain.

```
certreq.exe -accept signedCertificate.cer
```

Sekarang Anda dapat menggunakan sertifikat yang diimpor untuk [Tanda tangani File](#).

Langkah 3: Tanda tangani file

Anda sekarang siap untuk menggunakan SignTool dan sertifikat impor Anda untuk menandatangani file contoh Anda. Untuk melakukannya, Anda perlu mengetahui SHA-1 hash atau cap jempol. Sidik jari digunakan untuk memastikan bahwa SignTool hanya menggunakan sertifikat yang diverifikasi

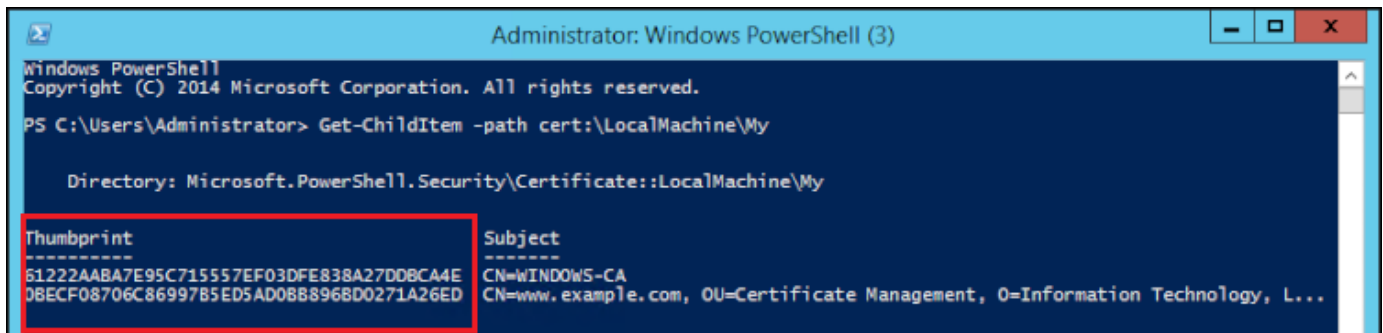
oleh. AWS CloudHSM Dalam contoh ini, kita gunakan PowerShell untuk mendapatkan hash sertifikat. Anda juga dapat menggunakan GUI CA atau certutil SDK Windows yang dapat dieksekusi.

Untuk mendapatkan cap jempol sertifikat dan menggunakannya untuk menandatangani file

1. Buka PowerShell sebagai administrator dan jalankan perintah berikut:

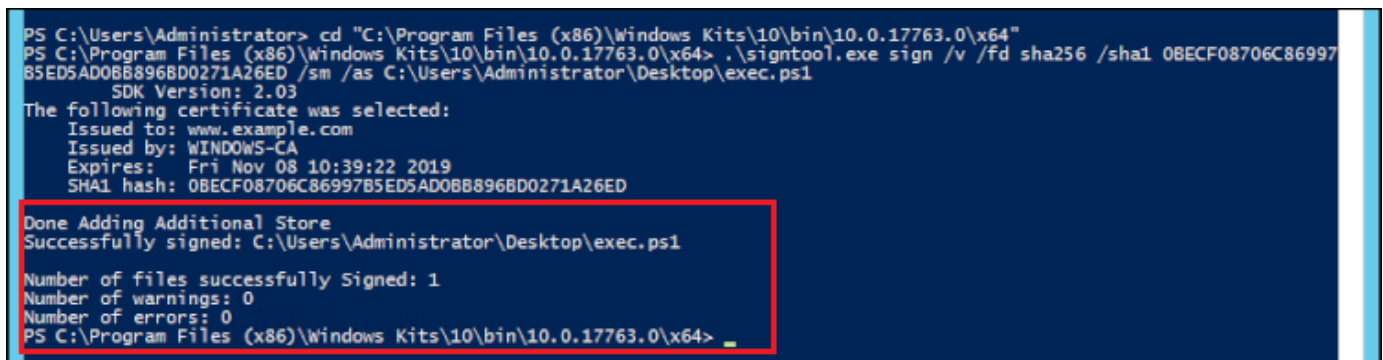
```
Get-ChildItem -path cert:\LocalMachine\My
```

Salin Thumbprint yang dikembalikan.



2. Arahkan ke direktori di dalamnya PowerShell berisi SignTool.exe. Lokasi default adalah C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64.
3. Terakhir, tanda tangani file Anda dengan menjalankan perintah berikut. Jika perintah berhasil, PowerShell mengembalikan pesan sukses.

```
signtool.exe sign /v /fd sha256 /sha1 <thumbprint> /sm C:\Users\Administrator\
\Desktop\<test>.ps1
```



4. (Opsional) Untuk memverifikasi tanda tangan pada file, gunakan perintah berikut:

```
signtool.exe verify /v /pa C:\Users\Administrator\Desktop\<test>.ps1
```

Java Keytool dan integrasi Jarsigner dengan AWS CloudHSM

AWS CloudHSM menawarkan integrasi dengan utilitas Java Keytool dan Jarsigner melalui Client SDK 3 dan Client SDK 5. Langkah-langkah untuk menggunakan alat ini akan bervariasi tergantung pada versi SDK klien yang saat ini telah Anda unduh. Bagian berikut memberikan informasi untuk setiap SDK.

Topik

- [Gunakan Client SDK 5 untuk mengintegrasikan AWS CloudHSM dengan Java Keytool dan Jarsigner](#)
- [Gunakan Client SDK 3 untuk berintegrasi AWS CloudHSM dengan Java Keytool dan Jarsigner](#)

Gunakan Client SDK 5 untuk mengintegrasikan AWS CloudHSM dengan Java Keytool dan Jarsigner

AWS CloudHSM key store adalah toko kunci JCE tujuan khusus yang menggunakan sertifikat yang terkait dengan kunci pada modul keamanan perangkat keras (HSM) Anda melalui alat pihak ketiga seperti `keytool` `jarsigner`. AWS CloudHSM tidak menyimpan sertifikat di HSM, karena sertifikat bersifat publik, data non-rahasia. Toko AWS CloudHSM kunci menyimpan sertifikat dalam file lokal dan memetakan sertifikat ke kunci yang sesuai di HSM Anda.

Saat Anda menggunakan toko AWS CloudHSM kunci untuk menghasilkan kunci baru, tidak ada entri yang dihasilkan di file penyimpanan kunci lokal — kunci dibuat di HSM. Demikian pula, ketika Anda menggunakan penyimpanan kunci AWS CloudHSM untuk mencari kunci, pencarian diteruskan ke HSM. Ketika Anda menyimpan sertifikat di toko AWS CloudHSM kunci, penyedia memverifikasi bahwa key pair dengan alias yang sesuai ada di HSM, dan kemudian mengaitkan sertifikat yang disediakan dengan key pair yang sesuai.

Topik

- [Prasyarat untuk mengintegrasikan AWS CloudHSM dengan Java Keytool dan Jarsigner menggunakan Client SDK 5](#)
- [Gunakan AWS CloudHSM key store dengan keytool menggunakan Client SDK 5](#)
- [Gunakan toko AWS CloudHSM kunci dengan Jarsigner menggunakan Client SDK 5](#)
- [Masalah yang diketahui untuk AWS CloudHSM integrasi Java Keytool dan Jarsigner menggunakan Client SDK 5](#)

Prasyarat untuk mengintegrasikan AWS CloudHSM dengan Java Keytool dan Jarsigner menggunakan Client SDK 5

Untuk menggunakan penyimpanan AWS CloudHSM kunci, Anda harus terlebih dahulu menginisialisasi dan mengkonfigurasi AWS CloudHSM JCE SDK. Gunakan langkah-langkah berikut untuk melakukannya.

Langkah 1: Instal JCE

[Untuk menginstal JCE, termasuk prasyarat AWS CloudHSM klien, ikuti langkah-langkah untuk menginstal perpustakaan Java.](#)

Langkah 2: Tambahkan kredensial login HSM ke variabel lingkungan

Atur variabel lingkungan berisi kredensial login HSM Anda.

Linux

```
$ export HSM_USER=<HSM user name>
```

```
$ export HSM_PASSWORD=<HSM password>
```

Windows

```
PS C:\> $Env:HSM_USER=<HSM user name>
```

```
PS C:\> $Env:HSM_PASSWORD=<HSM password>
```

Note

AWS CloudHSM JCE menawarkan berbagai opsi login. Untuk menggunakan penyimpanan AWS CloudHSM kunci dengan aplikasi pihak ketiga, Anda harus menggunakan login implisit dengan variabel lingkungan. Jika Anda ingin menggunakan login eksplisit melalui kode aplikasi, Anda harus membangun aplikasi Anda sendiri menggunakan toko AWS CloudHSM kunci. Untuk informasi tambahan, lihat artikel tentang [Menggunakan AWS CloudHSM Key Store](#).

Langkah 3: Mendaftarkan penyedia JCE

Untuk mendaftar penyedia JCE dalam CloudProvider konfigurasi Java, ikuti langkah-langkah berikut:

1. Buka file `java.security` konfigurasi di instalasi Java Anda untuk diedit.
2. Dalam file `java.security` konfigurasi, tambahkan `com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider` sebagai penyedia terakhir. Misalnya, jika ada sembilan penyedia dalam `java.security` file, tambahkan penyedia berikut sebagai penyedia terakhir di bagian:

```
security.provider.10=com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider
```

Note

Menambahkan AWS CloudHSM penyedia sebagai prioritas yang lebih tinggi dapat berdampak negatif pada kinerja sistem Anda karena AWS CloudHSM penyedia akan diprioritaskan untuk operasi yang dapat diturunkan dengan aman ke perangkat lunak. Sebagai praktik terbaik, selalu tentukan penyedia yang ingin Anda gunakan untuk suatu operasi, apakah itu AWS CloudHSM atau penyedia berbasis perangkat lunak.

Note

Menentukan `-providerName`, `-providerclass`, dan opsi baris `-providerpath` perintah saat membuat kunci yang digunakan `keytool` dengan penyimpanan kunci AWS CloudHSM dapat menyebabkan kesalahan.

Gunakan AWS CloudHSM key store dengan `keytool` menggunakan Client SDK 5

[Keytool](#) adalah utilitas baris perintah populer untuk tugas kunci dan sertifikat umum. Tutorial lengkap tentang `keytool` ada di luar lingkup dokumentasi AWS CloudHSM. Artikel ini menjelaskan parameter spesifik yang harus Anda gunakan dengan berbagai fungsi `keytool` saat memanfaatkan AWS CloudHSM sebagai akar kepercayaan melalui toko AWS CloudHSM kunci.

Saat menggunakan `keytool` dengan penyimpanan AWS CloudHSM kunci, tentukan argumen berikut ke perintah `keytool` apa pun:

Linux

```
-storetype CLOUDHSM -J-classpath< '-J/opt/cloudhsm/java/*'>
```

Windows

```
-storetype CLOUDHSM -J-classpath<'-J"C:\Program Files\Amazon\CloudHSM\java\*"'>
```

Jika Anda ingin membuat file penyimpanan kunci baru menggunakan toko AWS CloudHSM kunci, lihat [Gunakan AWS CloudHSM KeyStore untuk AWS CloudHSM Client SDK 3](#). Untuk menggunakan penyimpanan kunci yang ada, tentukan namanya (termasuk jalur) menggunakan argumen penyimpanan kunci pada keytool. Jika Anda menentukan file penyimpanan kunci yang tidak ada dalam perintah keytool, toko AWS CloudHSM kunci akan membuat file penyimpanan kunci baru.

Buat AWS CloudHSM kunci baru dengan keytool

Anda dapat menggunakan keytool untuk menghasilkan RSA, AES, dan DESede jenis kunci yang didukung oleh JCE SDK AWS CloudHSM .

Important

Kunci yang dihasilkan melalui keytool dihasilkan dalam perangkat lunak, dan kemudian diimpor AWS CloudHSM sebagai kunci persisten yang dapat diekstraksi.

Kami sangat menyarankan untuk membuat bukti kunci non-ekspor di luar keytool, dan kemudian mengimpor sertifikat yang sesuai ke penyimpanan kunci. Jika Anda menggunakan kunci RSA atau EC yang dapat diekstraksi melalui keytool dan Jarsigner, penyedia mengeksport kunci dari AWS CloudHSM dan kemudian menggunakan kunci secara lokal untuk operasi penandatanganan.

Jika Anda memiliki beberapa instance klien yang terhubung ke AWS CloudHSM klaster Anda, ketahuilah bahwa mengimpor sertifikat di penyimpanan kunci satu instans klien tidak akan secara otomatis membuat sertifikat tersedia pada instance klien lainnya. Untuk mendaftarkan kunci dan sertifikat terkait pada setiap instance klien, Anda perlu menjalankan aplikasi Java seperti yang dijelaskan dalam [the section called “Hasilkan CSR menggunakan keytool”](#). Atau, Anda dapat membuat perubahan yang diperlukan pada satu klien dan menyalin file penyimpanan kunci yang dihasilkan untuk setiap instans klien lainnya.

Contoh 1: Untuk menghasilkan kunci AES-256 simetris dan menyimpannya dalam file penyimpanan kunci bernama, “example_keystore.store”, di direktori kerja. Ganti *<secret label>* dengan label unik.

Linux

```
$ keytool -genseckey -alias <secret label> -keyalg aes \
-keysize 256 -keystore example_keystore.store \
-storetype CloudHSM -J-classpath '-J/opt/cloudhsm/java/*' \
```

Windows

```
PS C:\> keytool -genseckey -alias <secret label> -keyalg aes `
-keysize 256 -keystore example_keystore.store `
-storetype CloudHSM -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Contoh 2: Untuk menghasilkan key pair RSA 2048 dan menyimpannya dalam file key store bernama, “example_keystore.store” di direktori kerja. Ganti *<RSA key pair label>* dengan label unik.

Linux

```
$ keytool -genkeypair -alias <RSA key pair label> \
-keyalg rsa -keysize 2048 \
-sigalg sha512withrsa \
-keystore example_keystore.store \
-storetype CLOUDHSM \
-J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -genkeypair -alias <RSA key pair label> `
-keyalg rsa -keysize 2048 `
-sigalg sha512withrsa `
-keystore example_keystore.store `
-storetype CLOUDHSM `
-J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Anda dapat menemukan daftar [algoritme tanda tangan yang didukung](#) di pustaka Java.

Hapus AWS CloudHSM kunci menggunakan keytool

Toko AWS CloudHSM kunci tidak mendukung penghapusan kunci. Anda dapat menghapus kunci menggunakan metode penghancuran antarmuka [Destroyable](#).

```
((Destroyable) key).destroy();
```

Menghasilkan AWS CloudHSM CSR menggunakan keytool

Anda menerima fleksibilitas terbesar dalam membuat permintaan penandatanganan sertifikat (CSR) jika menggunakan [Mesin AWS CloudHSM Dinamis OpenSSL untuk SDK Klien 5](#). Perintah berikut menggunakan keytool untuk menghasilkan CSR untuk pasangan kunci dengan alias, `example-key-pair`.

Linux

```
$ keytool -certreq -alias <key pair label> \  
-file my_csr.csr \  
-keystore example_keystore.store \  
-storetype CLOUDHSM \  
-J-classpath '-J/opt/cloudhsm/java/'
```

Windows

```
PS C:\> keytool -certreq -alias <key pair label> \  
-file my_csr.csr \  
-keystore example_keystore.store \  
-storetype CLOUDHSM \  
-J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Note

Untuk menggunakan pasangan kunci dari keytool, pasangan kunci itu harus memiliki entri dalam file penyimpanan kunci tertentu. Jika Anda ingin menggunakan pasangan kunci yang dihasilkan di luar keytool, Anda harus mengimpor kunci dan metadata sertifikat ke penyimpanan kunci. Untuk petunjuk tentang mengimpor data keystore lihat. [the section called “Gunakan keytool untuk mengimpor sertifikat ke toko kunci”](#)

Gunakan keytool untuk mengimpor sertifikat perantara dan root ke dalam penyimpanan AWS CloudHSM kunci

Untuk mengimpor sertifikat CA AWS CloudHSM, Anda harus mengaktifkan verifikasi rantai sertifikat lengkap pada sertifikat yang baru diimpor. Perintah berikut menunjukkan sebuah contoh.

Linux

```
$ keytool -import -trustcacerts -alias rootCAcert \  
-file rootCAcert.cert -keystore example_keystore.store \  
-storetype CLOUDHSM \  
-J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -import -trustcacerts -alias rootCAcert `\  
-file rootCAcert.cert -keystore example_keystore.store `\  
-storetype CLOUDHSM `\  
-J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Jika Anda menghubungkan beberapa instance klien ke AWS CloudHSM klaster Anda, mengimpor sertifikat pada penyimpanan kunci satu instans klien tidak akan secara otomatis membuat sertifikat tersedia pada instance klien lainnya. Anda harus mengimpor sertifikat pada setiap instans klien.

Gunakan keytool untuk menghapus sertifikat dari toko AWS CloudHSM kunci

Perintah berikut menunjukkan contoh cara menghapus AWS CloudHSM sertifikat dari toko kunci keytool Java.

Linux

```
$ keytool -delete -alias mydomain \  
-keystore example_keystore.store \  
-storetype CLOUDHSM \  
-J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -delete -alias mydomain `\  
-keystore example_keystore.store `\  
-storetype CLOUDHSM `
```



```
-J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Jika Anda menghubungkan beberapa instance klien ke AWS CloudHSM klaster Anda, menghapus sertifikat pada penyimpanan kunci satu instans klien tidak akan secara otomatis menghapus sertifikat dari instance klien lainnya. Anda harus menghapus sertifikat pada setiap instans klien.

Impor sertifikat kerja ke toko AWS CloudHSM kunci menggunakan keytool

Setelah permintaan penandatanganan sertifikat (CSR) ditandatangani, Anda dapat mengimpornya ke penyimpanan kunci AWS CloudHSM dan mengaitkannya dengan pasangan kunci yang sesuai. Perintah berikut memberikan sebuah contoh.

Linux

```
$ keytool -importcert -noprompt -alias <key pair label> \  
-file my_certificate.crt \  
-keystore example_keystore.store \  
-storetype CLOUDHSM \  
-J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -importcert -noprompt -alias <key pair label> `  
-file my_certificate.crt `  
-keystore example_keystore.store `  
-storetype CLOUDHSM `  
-J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Alias harus menjadi pasangan kunci dengan sertifikat terkait di penyimpanan kunci. Jika kunci yang dihasilkan di luar keytool, atau dihasilkan pada instans klien yang berbeda, Anda harus terlebih dahulu mengimpor kunci dan metadata sertifikat ke penyimpanan kunci.

Rantai sertifikat harus dapat diverifikasi. Jika Anda tidak dapat memverifikasi sertifikat, Anda mungkin perlu mengimpor sertifikat tanda tangan (otoritas sertifikat) ke penyimpanan kunci sehingga rantai dapat diverifikasi.

Ekspor sertifikat dari AWS CloudHSM menggunakan keytool

Contoh berikut menghasilkan sertifikat dalam format X.509 biner. Untuk mengekspor sertifikat yang dapat dibaca manusia dari AWS CloudHSM, tambahkan `-rfc` ke `-exportcert` perintah.

Linux

```
$ keytool -exportcert -alias <key pair label> \  
-file my_exported_certificate.crt \  
-keystore example_keystore.store \  
-storetype CLOUDHSM \  
-J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -exportcert -alias <key pair label> \  
-file my_exported_certificate.crt \  
-keystore example_keystore.store \  
-storetype CLOUDHSM \  
-J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Gunakan toko AWS CloudHSM kunci dengan Jarsigner menggunakan Client SDK 5

Jarsigner adalah utilitas baris perintah populer untuk menandatangani file JAR menggunakan kunci yang disimpan dengan aman pada modul keamanan perangkat keras (HSM). Tutorial lengkap tentang Jarsigner berada di luar cakupan dokumentasi. AWS CloudHSM Bagian ini menjelaskan parameter Jarsigner yang harus Anda gunakan untuk menandatangani dan memverifikasi tanda tangan AWS CloudHSM sebagai akar kepercayaan melalui penyimpanan kunci. AWS CloudHSM

Siapkan AWS CloudHSM kunci dan sertifikat dengan Jarsigner

Sebelum Anda dapat menandatangani file AWS CloudHSM JAR dengan Jarsigner, pastikan Anda telah mengatur atau menyelesaikan langkah-langkah berikut:

1. Ikuti panduan di [prasyarat toko AWS CloudHSM utama](#).
2. Siapkan kunci penandatanganan Anda dan sertifikat terkait serta rantai sertifikat yang harus disimpan di penyimpanan AWS CloudHSM kunci server atau instance klien saat ini. Buat kunci pada AWS CloudHSM dan kemudian impor metadata terkait ke toko AWS CloudHSM kunci Anda. Jika Anda ingin menggunakan keytool untuk mengatur kunci dan sertifikat, lihat [the section called "Buat kunci baru dengan keytool"](#). Jika Anda menggunakan beberapa instance klien untuk menandatangani JARs, buat kunci dan impor rantai sertifikat. Kemudian, salin file penyimpanan kunci yang dihasilkan ke setiap instans klien. Jika Anda sering membuat kunci baru, Anda mungkin merasa lebih mudah untuk secara individual mengimpor sertifikat ke setiap instans klien.

3. Seluruh rantai sertifikat harus dapat diverifikasi. Agar rantai sertifikat dapat diverifikasi, Anda mungkin perlu menambahkan sertifikat CA dan sertifikat perantara ke toko AWS CloudHSM kunci. Lihat cuplikan kode [the section called “Tanda tangani file JAR”](#) untuk instruksi tentang penggunaan kode Java untuk memverifikasi rantai sertifikat. Jika mau, Anda dapat menggunakan keytool untuk mengimpor sertifikat. Untuk petunjuk tentang penggunaan keytool, lihat [the section called “Gunakan keytool untuk mengimpor sertifikat ke toko kunci”](#).

Menandatangani file JAR menggunakan AWS CloudHSM dan Jarsigner

Gunakan perintah berikut untuk menandatangani file JAR menggunakan AWS CloudHSM dan Jarsigner:

Linux;

Untuk OpenJDK 8

```
jarsigner -keystore example_keystore.store \  
-signedjar signthisclass_signed.jar \  
-sigalg sha512withrsa \  
-storetype CloudHSM \  
-J-classpath '-J/opt/cloudhsm/java/*:/usr/lib/jvm/java-1.8.0/lib/tools.jar' \  
-J-Djava.library.path=/opt/cloudhsm/lib \  
signthisclass.jar <key pair label>
```

Untuk OpenJDK 11, OpenJDK 17, dan OpenJDK 21

```
jarsigner -keystore example_keystore.store \  
-signedjar signthisclass_signed.jar \  
-sigalg sha512withrsa \  
-storetype CloudHSM \  
-J-classpath '-J/opt/cloudhsm/java/*' \  
-J-Djava.library.path=/opt/cloudhsm/lib \  
signthisclass.jar <key pair label>
```

Windows

Untuk Buka JDK8

```
jarsigner -keystore example_keystore.store `
```

```
-signedjar signthisclass_signed.jar `
-sialg sha512withrsa `
-storetype CloudHSM `
-J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*;C:\Program Files\Java
\jdk1.8.0_331\lib\tools.jar' `
"-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
signthisclass.jar <key pair label>
```

Untuk OpenJDK 11, OpenJDK 17, dan OpenJDK 21

```
jarsigner -keystore example_keystore.store `
-signedjar signthisclass_signed.jar `
-sialg sha512withrsa `
-storetype CloudHSM `
-J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*' `
"-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
signthisclass.jar <key pair label>
```

Gunakan perintah berikut untuk memverifikasi JAR yang ditandatangani:

Linux

Untuk Buka JDK8

```
jarsigner -verify \
-keystore example_keystore.store \
-sialg sha512withrsa \
-storetype CloudHSM \
-J-classpath '-J/opt/cloudhsm/java/*:/usr/lib/jvm/java-1.8.0/lib/tools.jar' \
-J-Djava.library.path=/opt/cloudhsm/lib \
signthisclass_signed.jar <key pair label>
```

Untuk OpenJDK 11, OpenJDK 17, dan OpenJDK 21

```
jarsigner -verify \
-keystore example_keystore.store \
-sialg sha512withrsa \
-storetype CloudHSM \
```

```
-J-classpath '-J/opt/cloudhsm/java/*' \
-J-Djava.library.path=/opt/cloudhsm/lib \
signthisclass_signed.jar <key pair label>
```

Windows

Untuk OpenJDK 8

```
jarsigner -verify `
-keystore example_keystore.store `
-sigalg sha512withrsa `
-storetype CloudHSM `
-J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*;C:\Program Files\Java
\jdk1.8.0_331\lib\tools.jar' `
"-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
signthisclass_signed.jar <key pair label>
```

Untuk OpenJDK 11, OpenJDK 17, dan OpenJDK 21

```
jarsigner -verify `
-keystore example_keystore.store `
-sigalg sha512withrsa `
-storetype CloudHSM `
-J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*' `
"-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
signthisclass_signed.jar <key pair label>
```

Masalah yang diketahui untuk AWS CloudHSM integrasi Java Keytool dan Jarsigner menggunakan Client SDK 5

Daftar berikut menyediakan daftar masalah yang diketahui saat ini untuk integrasi dengan AWS CloudHSM dan Java Keytool dan Jarsigner menggunakan Client SDK 5.

1. Kami tidak mendukung kunci EC dengan Keytool dan Jarsigner.

Gunakan Client SDK 3 untuk berintegrasi AWS CloudHSM dengan Java Keytool dan Jarsigner

AWS CloudHSM key store adalah toko kunci JCE tujuan khusus yang menggunakan sertifikat yang terkait dengan kunci pada modul keamanan perangkat keras (HSM) Anda melalui alat pihak ketiga seperti `keytool` dan `jarsigner`. AWS CloudHSM tidak menyimpan sertifikat di HSM, karena sertifikat bersifat publik, data non-rahasia. Toko AWS CloudHSM kunci menyimpan sertifikat dalam file lokal dan memetakan sertifikat ke kunci yang sesuai di HSM Anda.

Saat Anda menggunakan toko AWS CloudHSM kunci untuk menghasilkan kunci baru, tidak ada entri yang dihasilkan di file penyimpanan kunci lokal — kunci dibuat di HSM. Demikian pula, ketika Anda menggunakan penyimpanan kunci AWS CloudHSM untuk mencari kunci, pencarian diteruskan ke HSM. Ketika Anda menyimpan sertifikat di toko AWS CloudHSM kunci, penyedia memverifikasi bahwa key pair dengan alias yang sesuai ada di HSM, dan kemudian mengaitkan sertifikat yang disediakan dengan key pair yang sesuai.

Topik

- [Prasyarat untuk mengintegrasikan AWS CloudHSM dengan Java Keytool dan Jarsigner menggunakan Client SDK 3](#)
- [Gunakan AWS CloudHSM key store dengan keytool menggunakan Client SDK 3](#)
- [Gunakan toko AWS CloudHSM kunci dengan Jarsigner menggunakan Client SDK 3](#)
- [Masalah yang diketahui untuk AWS CloudHSM integrasi Java Keytool dan Jarsigner menggunakan Client SDK 3](#)
- [Daftarkan kunci yang sudah ada sebelumnya dengan toko AWS CloudHSM kunci](#)

Prasyarat untuk mengintegrasikan AWS CloudHSM dengan Java Keytool dan Jarsigner menggunakan Client SDK 3

Untuk menggunakan penyimpanan AWS CloudHSM kunci, Anda harus terlebih dahulu menginisialisasi dan mengkonfigurasi AWS CloudHSM JCE SDK. Gunakan langkah-langkah berikut untuk melakukannya.

Langkah 1: Instal JCE

[Untuk menginstal JCE, termasuk prasyarat AWS CloudHSM klien, ikuti langkah-langkah untuk menginstal perpustakaan Java.](#)

Langkah 2: Tambahkan kredensial login HSM ke variabel lingkungan

Atur variabel lingkungan berisi kredensial login HSM Anda.

```
export HSM_PARTITION=PARTITION_1
export HSM_USER=<HSM user name>
export HSM_PASSWORD=<HSM password>
```

Note

CloudHSM JCE menawarkan berbagai pilihan login. Untuk menggunakan penyimpanan AWS CloudHSM kunci dengan aplikasi pihak ketiga, Anda harus menggunakan login implisit dengan variabel lingkungan. Jika Anda ingin menggunakan login eksplisit melalui kode aplikasi, Anda harus membangun aplikasi Anda sendiri menggunakan toko AWS CloudHSM kunci. Untuk informasi tambahan, lihat artikel tentang [Menggunakan AWS CloudHSM Key Store](#).

Langkah 3: Daftarkan penyedia JCE

Untuk mendaftarkan penyedia JCE, dalam CloudProvider konfigurasi Java.

1. Buka file konfigurasi java.security di instalasi Java Anda, untuk mengedit.
2. Dalam file konfigurasi java.security, tambahkan `com.cavium.provider.CaviumProvider` sebagai penyedia terakhir. Misalnya, jika ada sembilan penyedia dalam file.security, tambahkan penyedia berikut sebagai penyedia terakhir di bagian tersebut. Menambahkan penyedia Cavium sebagai prioritas yang lebih tinggi dapat berdampak negatif terhadap performa sistem Anda.

```
security.provider.10=com.cavium.provider.CaviumProvider
```

Note

Power user mungkin terbiasa untuk menentukan opsi baris perintah `-providerName`, `-providerclass`, dan `-providerpath` saat menggunakan `keytool`, alih-alih memperbarui file konfigurasi keamanan. Jika Anda mencoba menentukan opsi baris perintah saat membuat kunci dengan penyimpanan AWS CloudHSM kunci, itu akan menyebabkan kesalahan.

Gunakan AWS CloudHSM key store dengan keytool menggunakan Client SDK 3

[Keytool](#) adalah utilitas baris perintah populer untuk kunci umum dan sertifikat tugas pada sistem Linux. Tutorial lengkap tentang keytool ada di luar lingkup dokumentasi AWS CloudHSM . Artikel ini menjelaskan parameter spesifik yang harus Anda gunakan dengan berbagai fungsi keytool saat memanfaatkan AWS CloudHSM sebagai akar kepercayaan melalui toko AWS CloudHSM kunci.

Saat menggunakan keytool dengan penyimpanan AWS CloudHSM kunci, tentukan argumen berikut ke perintah keytool apa pun:

```
-storetype CLOUDHSM \  
-J-classpath '-J/opt/cloudhsm/java/*' \  
-J-Djava.library.path=/opt/cloudhsm/lib
```

Jika Anda ingin membuat file penyimpanan kunci baru menggunakan toko AWS CloudHSM kunci, lihat [Gunakan AWS CloudHSM KeyStore untuk AWS CloudHSM Client SDK 3](#). Untuk menggunakan penyimpanan kunci yang ada, tentukan namanya (termasuk jalur) menggunakan argumen penyimpanan kunci pada keytool. Jika Anda menentukan file penyimpanan kunci yang tidak ada dalam perintah keytool, toko AWS CloudHSM kunci akan membuat file penyimpanan kunci baru.

Buat AWS CloudHSM kunci baru dengan keytool

Anda dapat menggunakan keytool untuk menghasilkan semua jenis kunci yang didukung oleh AWS CloudHSM JCE SDK. Lihat daftar lengkap kunci dan panjang di artikel [Kunci yang Didukung](#) di Pustaka Java.

Important

Kunci yang dihasilkan melalui keytool dihasilkan dalam perangkat lunak, dan kemudian diimpor AWS CloudHSM sebagai kunci persisten yang dapat diekstraksi.

Petunjuk untuk membuat kunci yang tidak dapat diekstraksi langsung pada modul keamanan perangkat keras (HSM), dan kemudian menggunakannya dengan keytool atau Jarsigner, ditampilkan dalam contoh kode di [Mendaftarkan](#) Kunci yang Sudah Ada dengan Key Store. AWS CloudHSM Kami sangat menyarankan untuk membuat bukti kunci non-ekspor di luar keytool, dan kemudian mengimpor sertifikat yang sesuai ke penyimpanan kunci. Jika Anda menggunakan kunci RSA atau EC yang dapat diekstrak melalui keytool dan jarsigner, penyedia mengeksport kunci dari AWS CloudHSM dan kemudian menggunakan kunci secara lokal untuk operasi penandatanganan.

Jika Anda memiliki beberapa instans klien yang tersambung ke klaster CloudHSM Anda, perhatikan bahwa mengimpor sertifikat di penyimpanan kunci satu instans klien tidak akan secara otomatis membuat sertifikat tersebut tersedia pada instans klien lainnya. Untuk mendaftarkan kunci dan sertifikat terkait pada setiap instans klien, Anda perlu menjalankan aplikasi Java seperti yang dijelaskan dalam [Hasilkan CSR menggunakan Keytool](#). Atau, Anda dapat membuat perubahan yang diperlukan pada satu klien dan menyalin file penyimpanan kunci yang dihasilkan untuk setiap instans klien lainnya.

Contoh 1: Untuk menghasilkan kunci AES-256 simetris dan menyimpannya dalam file penyimpanan kunci bernama, “example_keystore.store”, di direktori kerja. Ganti *<secret label>* dengan label unik.

```
keytool -genseckey -alias <secret label> -keyalg aes \
-keysize 256 -keystore example_keystore.store \
-storetype CloudHSM -J-classpath '-J/opt/cloudhsm/java/*' \
-J-Djava.library.path=/opt/cloudhsm/lib/
```

Contoh 2: Untuk menghasilkan key pair RSA 2048 dan menyimpannya dalam file key store bernama, “example_keystore.store” di direktori kerja. Ganti *<RSA key pair label>* dengan label unik.

```
keytool -genkeypair -alias <RSA key pair label> \
-keyalg rsa -keysize 2048 \
-sigalg sha512withrsa \
-keystore example_keystore.store \
-storetype CLOUDHSM \
-J-classpath '-J/opt/cloudhsm/java/*' \
-J-Djava.library.path=/opt/cloudhsm/lib/
```

Contoh 3: Untuk menghasilkan kunci ED p256 dan menyimpannya dalam file penyimpanan kunci bernama, “example_keystore.store” di direktori kerja. Ganti *<ec key pair label>* dengan label unik.

```
keytool -genkeypair -alias <ec key pair label> \
-keyalg ec -keysize 256 \
-sigalg SHA512withECDSA \
-keystore example_keystore.store \
-storetype CLOUDHSM \
-J-classpath '-J/opt/cloudhsm/java/*' \
-J-Djava.library.path=/opt/cloudhsm/lib/
```

Anda dapat menemukan daftar [algoritme tanda tangan yang didukung](#) di pustaka Java.

Hapus AWS CloudHSM kunci menggunakan keytool

Toko AWS CloudHSM kunci tidak mendukung penghapusan kunci. Untuk menghapus kunci, Anda harus menggunakan deleteKey fungsi AWS CloudHSM alat baris perintah, [Hapus AWS CloudHSM kunci menggunakan KMU](#).

Menghasilkan AWS CloudHSM CSR menggunakan keytool

Anda menerima fleksibilitas terbesar dalam membuat permintaan penandatanganan sertifikat (CSR) jika menggunakan [Mesin AWS CloudHSM Dinamis OpenSSL untuk SDK Klien 5](#). Perintah berikut menggunakan keytool untuk menghasilkan CSR untuk pasangan kunci dengan alias, example-key-pair.

```
keytool -certreq -alias <key pair label> \
  -file example_csr.csr \
  -keystore example_keystore.store \
  -storetype CLOUDHSM \
  -J-classpath '-J/opt/cloudhsm/java/*' \
  -J-Djava.library.path=/opt/cloudhsm/lib/
```

Note

Untuk menggunakan pasangan kunci dari keytool, pasangan kunci itu harus memiliki entri dalam file penyimpanan kunci tertentu. Jika Anda ingin menggunakan pasangan kunci yang dihasilkan di luar keytool, Anda harus mengimpor kunci dan metadata sertifikat ke penyimpanan kunci. Untuk petunjuk tentang mengimpor data keystore, lihat [Mengimpor sertifikat Intermediate dan root ke AWS CloudHSM Key Store](#) menggunakan Keytool.

Gunakan keytool untuk mengimpor sertifikat perantara dan root ke dalam penyimpanan AWS CloudHSM kunci

Untuk mengimpor sertifikat CA AWS CloudHSM, Anda harus mengaktifkan verifikasi rantai sertifikat lengkap pada sertifikat yang baru diimpor. Perintah berikut menunjukkan sebuah contoh.

```
keytool -import -trustcacerts -alias rootCAcert \
  -file rootCAcert.cert -keystore example_keystore.store \
  -storetype CLOUDHSM \
```

```
-J-classpath '-J/opt/cloudhsm/java/*' \  
-J-Djava.library.path=/opt/cloudhsm/lib/
```

Jika Anda menghubungkan beberapa instance klien ke AWS CloudHSM klaster Anda, mengimpor sertifikat pada penyimpanan kunci satu instans klien tidak akan secara otomatis membuat sertifikat tersedia pada instance klien lainnya. Anda harus mengimpor sertifikat pada setiap instans klien.

Gunakan keytool untuk menghapus sertifikat dari toko AWS CloudHSM kunci

Perintah berikut menunjukkan contoh cara menghapus AWS CloudHSM sertifikat dari toko kunci keytool Java.

```
keytool -delete -alias mydomain -keystore \  
-keystore example_keystore.store \  
-storetype CLOUDHSM \  
-J-classpath '-J/opt/cloudhsm/java/*' \  
-J-Djava.library.path=/opt/cloudhsm/lib/
```

Jika Anda menghubungkan beberapa instance klien ke AWS CloudHSM klaster Anda, menghapus sertifikat pada penyimpanan kunci satu instans klien tidak akan secara otomatis menghapus sertifikat dari instance klien lainnya. Anda harus menghapus sertifikat pada setiap instans klien.

Impor sertifikat kerja ke toko AWS CloudHSM kunci menggunakan keytool

Setelah permintaan penandatanganan sertifikat (CSR) ditandatangani, Anda dapat mengimpornya ke penyimpanan kunci AWS CloudHSM dan mengaitkannya dengan pasangan kunci yang sesuai. Perintah berikut memberikan sebuah contoh.

```
keytool -importcert -noprompt -alias <key pair label> \  
-file example_certificate.crt \  
-keystore example_keystore.store \  
-storetype CLOUDHSM \  
-J-classpath '-J/opt/cloudhsm/java/*' \  
-J-Djava.library.path=/opt/cloudhsm/lib/
```

Alias harus menjadi pasangan kunci dengan sertifikat terkait di penyimpanan kunci. Jika kunci yang dihasilkan di luar keytool, atau dihasilkan pada instans klien yang berbeda, Anda harus terlebih dahulu mengimpor kunci dan metadata sertifikat ke penyimpanan kunci. Untuk petunjuk tentang mengimpor metadata sertifikat, lihat contoh kode di [Mendaftarkan Kunci yang sudah ada sebelumnya](#) dengan Key Store. AWS CloudHSM

Rantai sertifikat harus dapat diverifikasi. Jika Anda tidak dapat memverifikasi sertifikat, Anda mungkin perlu mengimpor sertifikat tanda tangan (otoritas sertifikat) ke penyimpanan kunci sehingga rantai dapat diverifikasi.

Ekspor sertifikat dari AWS CloudHSM menggunakan keytool

Contoh berikut menghasilkan sertifikat dalam format X.509 biner. Untuk mengekspor sertifikat yang dapat dibaca manusia dari AWS CloudHSM, tambahkan `-rfc` ke `-exportcert` perintah.

```
keytool -exportcert -alias <key pair label> \
        -file example_exported_certificate.crt \
        -keystore example_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

Gunakan toko AWS CloudHSM kunci dengan Jarsigner menggunakan Client SDK 3

Jarsigner adalah utilitas baris perintah populer untuk menandatangani file JAR menggunakan kunci yang disimpan dengan aman pada modul keamanan perangkat keras (HSM). Tutorial lengkap tentang Jarsigner berada di luar cakupan dokumentasi. AWS CloudHSM Bagian ini menjelaskan parameter Jarsigner yang harus Anda gunakan untuk menandatangani dan memverifikasi tanda tangan AWS CloudHSM sebagai akar kepercayaan melalui penyimpanan kunci. AWS CloudHSM

Siapkan AWS CloudHSM kunci dan sertifikat dengan Jarsigner

Sebelum Anda dapat menandatangani file AWS CloudHSM JAR dengan Jarsigner, pastikan Anda telah mengatur atau menyelesaikan langkah-langkah berikut:

1. Ikuti petunjuk di [Prasyarat penyimpanan kunci AWS CloudHSM](#).
2. Siapkan kunci penandatanganan Anda dan sertifikat terkait serta rantai sertifikat yang harus disimpan di penyimpanan AWS CloudHSM kunci server atau instance klien saat ini. Buat kunci pada AWS CloudHSM dan kemudian impor metadata terkait ke toko AWS CloudHSM kunci Anda. Gunakan contoh kode dalam [Mendaftarkan Kunci yang sudah ada sebelumnya dengan AWS CloudHSM Key Store](#) untuk mengimpor metadata ke toko kunci. Jika Anda ingin menggunakan keytool untuk mengatur kunci dan sertifikat, lihat [Buat AWS CloudHSM kunci baru dengan keytool](#). Jika Anda menggunakan beberapa instance klien untuk menandatangani JARs, buat kunci dan impor rantai sertifikat. Kemudian, salin file penyimpanan kunci yang dihasilkan ke setiap instans klien. Jika Anda sering membuat kunci baru, Anda mungkin merasa lebih mudah untuk secara individual mengimpor sertifikat ke setiap instans klien.

3. Seluruh rantai sertifikat harus dapat diverifikasi. Agar rantai sertifikat dapat diverifikasi, Anda mungkin perlu menambahkan sertifikat CA dan sertifikat perantara ke toko AWS CloudHSM kunci. Lihat cuplikan kode di [Menandatangani file JAR menggunakan AWS CloudHSM dan Jarsigner](#) untuk instruksi tentang penggunaan kode Java untuk memverifikasi rantai sertifikat. Jika mau, Anda dapat menggunakan keytool untuk mengimpor sertifikat. Untuk petunjuk tentang penggunaan keytool, lihat [Menggunakan Keytool untuk mengimpor sertifikat perantara dan root ke AWS CloudHSM Key Store](#).

Menandatangani file JAR menggunakan AWS CloudHSM dan Jarsigner

Gunakan perintah berikut untuk menandatangani file JAR menggunakan AWS CloudHSM dan jarsigner:

```
jarsigner -keystore example_keystore.store \  
    -signedjar signthisclass_signed.jar \  
    -sigalg sha512withrsa \  
    -storetype CloudHSM \  
    -J-classpath '-J/opt/cloudhsm/java/*:/usr/lib/jvm/java-1.8.0/lib/tools.jar' \  
    -J-Djava.library.path=/opt/cloudhsm/lib \  
    signthisclass.jar <key pair label>
```

Gunakan perintah berikut untuk memverifikasi JAR yang ditandatangani:

```
jarsigner -verify \  
    -keystore example_keystore.store \  
    -sigalg sha512withrsa \  
    -storetype CloudHSM \  
    -J-classpath '-J/opt/cloudhsm/java/*:/usr/lib/jvm/java-1.8.0/lib/tools.jar' \  
    -J-Djava.library.path=/opt/cloudhsm/lib \  
    signthisclass_signed.jar <key pair label>
```

Masalah yang diketahui untuk AWS CloudHSM integrasi Java Keytool dan Jarsigner menggunakan Client SDK 3

Daftar berikut menyediakan daftar masalah yang diketahui saat ini untuk integrasi dengan AWS CloudHSM dan Java Keytool dan Jarsigner menggunakan Client SDK 3.

- Saat membuat kunci menggunakan keytool, penyedia pertama dalam konfigurasi penyedia tidak bisa CaviumProvider.

- Ketika menghasilkan kunci menggunakan keytool, penyedia pertama (yang didukung) dalam file konfigurasi keamanan digunakan untuk menghasilkan kunci. Ini umumnya adalah penyedia perangkat lunak. Kunci yang dihasilkan kemudian diberi alias dan diimpor ke AWS CloudHSM HSM sebagai kunci persisten (token) selama proses penambahan kunci.
- Saat menggunakan keytool dengan AWS CloudHSM key store, jangan tentukan -providerName, -providerclass, atau -providerpath opsi pada baris perintah. Tentukan opsi ini dalam file penyedia keamanan seperti yang dijelaskan dalam [Prasyarat penyimpanan kunci](#).
- Saat menggunakan kunci EC yang tidak dapat diekstraksi melalui keytool dan Jarsigner, penyedia SunEc removed/disabled harus dari daftar penyedia dalam file java.security. Jika Anda menggunakan kunci EC yang dapat diekstrak melalui keytool dan Jarsigner, penyedia mengekspor bit kunci dari AWS CloudHSM HSM dan menggunakan kunci secara lokal untuk operasi penandatanganan. Kami tidak menyarankan Anda menggunakan kunci yang dapat diekspor dengan keytool atau Jarsigner.

Daftarkan kunci yang sudah ada sebelumnya dengan toko AWS CloudHSM kunci

Untuk keamanan dan fleksibilitas maksimum dalam atribut dan pelabelan, kami sarankan Anda membuat kunci AWS CloudHSM penandatanganan menggunakan [key_mgmt_util](#). Anda juga dapat menggunakan aplikasi Java untuk menghasilkan kunci dalam AWS CloudHSM.

Bagian berikut menyediakan contoh kode yang menunjukkan cara menghasilkan key pair baru pada HSM dan mendaftarkannya menggunakan kunci yang ada yang diimpor ke toko AWS CloudHSM kunci. Kunci yang diimpor tersedia untuk digunakan dengan alat pihak ketiga seperti keytool dan Jarsigner.

Untuk menggunakan kunci yang sudah ada sebelumnya, modifikasi sampel kode untuk mencari kunci dengan label bukannya menghasilkan kunci baru. Contoh kode untuk mencari kunci berdasarkan label tersedia di [KeyUtilitiesRunnersampel.java](#) pada GitHub.

Important

Mendaftarkan kunci yang disimpan AWS CloudHSM dengan toko kunci lokal tidak mengekspor kunci. Ketika kunci terdaftar, penyimpanan kunci mendaftarkan alias kunci (atau label) dan menghubungkan objek sertifikat yang disimpan secara lokal dengan pasangan kunci pada AWS CloudHSM. Selama pasangan kunci dibuat sebagai tidak dapat diekspor, bit kunci tidak akan meninggalkan HSM.

```
//  
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
//  
// Permission is hereby granted, free of charge, to any person obtaining a copy of  
// this  
// software and associated documentation files (the "Software"), to deal in the  
// Software  
// without restriction, including without limitation the rights to use, copy, modify,  
// merge, publish, distribute, sublicense, and/or sell copies of the Software, and to  
// permit persons to whom the Software is furnished to do so.  
//  
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,  
// INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A  
// PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT  
// HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION  
// OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE  
// SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.  
//  
  
package com.amazonaws.cloudhsm.examples;  
  
import com.cavium.key.CaviumKey;  
import com.cavium.key.parameter.CaviumAESKeyGenParameterSpec;  
import com.cavium.key.parameter.CaviumRSAKeyGenParameterSpec;  
import com.cavium.asn1.Encoder;  
import com.cavium.cfm2.Util;  
  
import javax.crypto.KeyGenerator;  
  
import java.io.ByteArrayInputStream;  
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.FileNotFoundException;  
  
import java.math.BigInteger;  
  
import java.security.*;  
import java.security.cert.Certificate;  
import java.security.cert.CertificateException;  
import java.security.cert.CertificateFactory;
```

```

import java.security.cert.X509Certificate;
import java.security.interfaces.RSAPrivateKey;
import java.security.interfaces.RSAPublicKey;
import java.security.KeyStore.PasswordProtection;
import java.security.KeyStore.PrivateKeyEntry;
import java.security.KeyStore.Entry;

import java.util.Calendar;
import java.util.Date;
import java.util.Enumuration;

//
// KeyStoreExampleRunner demonstrates how to load a keystore, and associate a
// certificate with a
// key in that keystore.
//
// This example relies on implicit credentials, so you must setup your environment
// correctly.
//
// https://docs.aws.amazon.com/cloudhsm/latest/userguide/java-library-
// install.html#java-library-credentials
//

public class KeyStoreExampleRunner {

    private static byte[] COMMON_NAME_OID = new byte[] { (byte) 0x55, (byte) 0x04,
(byte) 0x03 };
    private static byte[] COUNTRY_NAME_OID = new byte[] { (byte) 0x55, (byte) 0x04,
(byte) 0x06 };
    private static byte[] LOCALITY_NAME_OID = new byte[] { (byte) 0x55, (byte) 0x04,
(byte) 0x07 };
    private static byte[] STATE_OR_PROVINCE_NAME_OID = new byte[] { (byte) 0x55,
(byte) 0x04, (byte) 0x08 };
    private static byte[] ORGANIZATION_NAME_OID = new byte[] { (byte) 0x55, (byte)
0x04, (byte) 0x0A };
    private static byte[] ORGANIZATION_UNIT_OID = new byte[] { (byte) 0x55, (byte)
0x04, (byte) 0x0B };

    private static String helpString = "KeyStoreExampleRunner%n" +
        "This sample demonstrates how to load and store keys using a keystore.%n%n"
+
        "Options%n" +
        "\t--help\t\t\tDisplay this message.%n" +
        "\t--store <filename>\t\tPath of the keystore.%n" +

```



```
"\t--password <password>\t\tPassword for the keystore (not your CU
password).%n" +
"\t--label <label>\t\tLabel to store the key and certificate under.%n" +
"\t--list\t\tList all the keys in the keystore.%n%n";

public static void main(String[] args) throws Exception {
    Security.addProvider(new com.cavium.provider.CaviumProvider());
    KeyStore keyStore = KeyStore.getInstance("CloudHSM");

    String keystoreFile = null;
    String password = null;
    String label = null;
    boolean list = false;
    for (int i = 0; i < args.length; i++) {
        String arg = args[i];
        switch (args[i]) {
            case "--store":
                keystoreFile = args[++i];
                break;
            case "--password":
                password = args[++i];
                break;
            case "--label":
                label = args[++i];
                break;
            case "--list":
                list = true;
                break;
            case "--help":
                help();
                return;
        }
    }

    if (null == keystoreFile || null == password) {
        help();
        return;
    }

    if (list) {
        listKeys(keystoreFile, password);
        return;
    }
}
```

```

    if (null == label) {
        label = "Keystore Example Keypair";
    }

    //
    // This call to keyStore.load() will open the pkcs12 keystore with the supplied
    // password and connect to the HSM. The CU credentials must be specified using
    // standard CloudHSM login methods.
    //
    try {
        FileInputStream instream = new FileInputStream(keystoreFile);
        keyStore.load(instream, password.toCharArray());
    } catch (FileNotFoundException ex) {
        System.err.println("Keystore not found, loading an empty store");
        keyStore.load(null, null);
    }

    PasswordProtection passwd = new PasswordProtection(password.toCharArray());
    System.out.println("Searching for example key and certificate...");

    PrivateKeyEntry keyEntry = (PrivateKeyEntry) keyStore.getEntry(label, passwd);
    if (null == keyEntry) {
        //
        // No entry was found, so we need to create a key pair and associate a
        certificate.
        // The private key will get the label passed on the command line. The
        keystore alias
        // needs to be the same as the private key label. The public key will have
        ":public"
        // appended to it. The alias used in the keystore will We associate the
        certificate
        // with the private key.
        //
        System.out.println("No entry found, creating...");
        KeyPair kp = generateRSAKeyPair(2048, label + ":public", label);
        System.out.printf("Created a key pair with the handles %d/%d%n",
            ((CaviumKey) kp.getPrivate()).getHandle(), ((CaviumKey) kp.getPublic()).getHandle());

        //
        // Generate a certificate and associate the chain with the private key.
        //
        Certificate self_signed_cert = generateCert(kp);
        Certificate[] chain = new Certificate[1];
        chain[0] = self_signed_cert;

```

```

        PrivateKeyEntry entry = new PrivateKeyEntry(kp.getPrivate(), chain);

        //
        // Set the entry using the label as the alias and save the store.
        // The alias must match the private key label.
        //
        keyStore.setEntry(label, entry, passwd);

        FileOutputStream outstream = new FileOutputStream(keystoreFile);
        keyStore.store(outstream, password.toCharArray());
        outstream.close();

        keyEntry = (PrivateKeyEntry) keyStore.getEntry(label, passwd);
    }

    long handle = ((CaviumKey) keyEntry.getPrivateKey()).getHandle();
    String name = keyEntry.getCertificate().toString();
    System.out.printf("Found private key %d with certificate %s\n", handle, name);
}

private static void help() {
    System.out.println(helpString);
}

//
// Generate a non-extractable / non-persistent RSA keypair.
// This method allows us to specify the public and private labels, which
// will make KeyStore aliases easier to understand.
//
public static KeyPair generateRSAKeyPair(int keySizeInBits, String publicLabel,
String privateLabel)
    throws InvalidAlgorithmParameterException, NoSuchAlgorithmException,
NoSuchProviderException {

    boolean isExtractable = false;
    boolean isPersistent = false;
    KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("rsa", "Cavium");
    CaviumRSAKeyGenParameterSpec spec = new
CaviumRSAKeyGenParameterSpec(keySizeInBits, new BigInteger("65537"), publicLabel,
privateLabel, isExtractable, isPersistent);

    keyPairGen.initialize(spec);

    return keyPairGen.generateKeyPair();
}

```

```

}

//
// Generate a certificate signed by a given keypair.
//
private static Certificate generateCert(KeyPair kp) throws CertificateException {
    CertificateFactory cf = CertificateFactory.getInstance("X509");
    PublicKey publicKey = kp.getPublic();
    PrivateKey privateKey = kp.getPrivate();
    byte[] version = Encoder.encodeConstructed((byte) 0,
Encoder.encodePositiveBigInteger(new BigInteger("2"))); // version 1
    byte[] serialNo = Encoder.encodePositiveBigInteger(new BigInteger(1,
Util.computeKCV(publicKey.getEncoded())));

    // Use the SHA512 OID and algorithm.
    byte[] signatureOid = new byte[] {
        (byte) 0x2A, (byte) 0x86, (byte) 0x48, (byte) 0x86, (byte) 0xF7, (byte)
0x0D, (byte) 0x01, (byte) 0x01, (byte) 0x0D };
    String sigAlgoName = "SHA512WithRSA";

    byte[] signatureId = Encoder.encodeSequence(
        Encoder.encodeOid(signatureOid),
        Encoder.encodeNull());

    byte[] issuer = Encoder.encodeSequence(
        encodeName(COUNTRY_NAME_OID, "<Country>"),
        encodeName(STATE_OR_PROVINCE_NAME_OID, "<State>"),
        encodeName(LOCALITY_NAME_OID, "<City>"),
        encodeName(ORGANIZATION_NAME_OID,
"<Organization>"),
        encodeName(ORGANIZATION_UNIT_OID, "<Unit>"),
        encodeName(COMMON_NAME_OID, "<CN>")
    );

    Calendar c = Calendar.getInstance();
    c.add(Calendar.DAY_OF_YEAR, -1);
    Date notBefore = c.getTime();
    c.add(Calendar.YEAR, 1);
    Date notAfter = c.getTime();
    byte[] validity = Encoder.encodeSequence(
        Encoder.encodeUTCTime(notBefore),
        Encoder.encodeUTCTime(notAfter)
    );

    byte[] key = publicKey.getEncoded();

```

```

        byte[] certificate = Encoder.encodeSequence(
            version,
            serialNo,
            signatureId,
            issuer,
            validity,
            issuer,
            key);

        Signature sig;
        byte[] signature = null;
        try {
            sig = Signature.getInstance(sigAlgoName, "Cavium");
            sig.initSign(privateKey);
            sig.update(certificate);
            signature = Encoder.encodeBitstring(sig.sign());

        } catch (Exception e) {
            System.err.println(e.getMessage());
            return null;
        }

        byte [] x509 = Encoder.encodeSequence(
            certificate,
            signatureId,
            signature
        );

        return cf.generateCertificate(new ByteArrayInputStream(x509));
    }

    //
    // Simple OID encoder.
    // Encode a value with OID in ASN.1 format
    //
    private static byte[] encodeName(byte[] nameOid, String value) {
        byte[] name = null;
        name = Encoder.encodeSet(
            Encoder.encodeSequence(
                Encoder.encodeOid(nameOid),
                Encoder.encodePrintableString(value)
            )
        );
        return name;
    }
}

```

```
//  
// List all the keys in the keystore.  
//  
private static void listKeys(String keystoreFile, String password) throws Exception  
{  
    KeyStore keyStore = KeyStore.getInstance("CloudHSM");  
  
    try {  
        FileInputStream instream = new FileInputStream(keystoreFile);  
        keyStore.load(instream, password.toCharArray());  
    } catch (FileNotFoundException ex) {  
        System.err.println("Keystore not found, loading an empty store");  
        keyStore.load(null, null);  
    }  
  
    for(Enumeration<String> entry = keyStore.aliases(); entry.hasMoreElements();) {  
        System.out.println(entry.nextElement());  
    }  
}
```

Gunakan Microsoft Manifest Generation dan Editing Tool (Mage.exe) AWS CloudHSM untuk menandatangani file

Note

AWS CloudHSM hanya mendukung alat Mage 64-bit yang disertakan dalam Windows SDK untuk .NET Framework 4.8.1 dan yang lebih baru.

Topik berikut memberikan ikhtisar tentang cara menggunakan [Mage.exe](#) dengan AWS CloudHSM.

Topik

- [Langkah 1: Siapkan prasyarat](#)
- [Langkah 2: Buat sertifikat penandatanganan](#)

- [Langkah 3: Tanda tangani file](#)

Langkah 1: Siapkan prasyarat

Untuk menggunakan Microsoft Mage.exe dengan AWS CloudHSM, Anda memerlukan yang berikut:

- EC2 Instans Amazon yang menjalankan sistem operasi Windows
- Otoritas sertifikat (CA), baik yang dikelola sendiri atau dari penyedia pihak ketiga
- AWS CloudHSM Cluster aktif di virtual private cloud (VPC) yang sama dengan EC2 instans Anda, dengan setidaknya satu HSM
- Pengguna kriptografi (CU) untuk memiliki dan mengelola kunci di AWS CloudHSM cluster
- File yang tidak ditandatangani atau dapat dieksekusi
- Kit Pengembangan Perangkat Lunak Microsoft Windows (SDK)

Untuk mengatur prasyarat untuk menggunakan dengan Mage.exe AWS CloudHSM

1. Luncurkan EC2 instance Windows dan AWS CloudHSM cluster dengan mengikuti petunjuk di bagian [Memulai](#) panduan ini.
2. Jika Anda ingin meng-host Windows Server CA Anda sendiri, selesaikan langkah 1 dan 2 dalam [Mengkonfigurasi Windows Server sebagai Otoritas Sertifikat dengan AWS CloudHSM](#). Jika tidak, gunakan CA pihak ketiga tepercaya publik Anda.
3. Unduh dan instal Microsoft Windows SDK for .NET Framework 4.8.1 atau yang lebih baru pada contoh Windows Anda EC2 :
 - [Microsoft Windows SDK 10](#)

mage.exeExecutable adalah bagian dari Windows SDK Tools. Lokasi instalasi default adalah:

```
C:\Program Files (x86)\Windows Kits\<SDK version>\bin\<version number>\x64\Mage.exe
```

Setelah menyelesaikan langkah-langkah ini, Anda dapat menggunakan Microsoft Windows SDK, AWS CloudHSM klaster, dan CA untuk [membuat sertifikat penandatanganan](#).

Langkah 2: Buat sertifikat penandatanganan

Sekarang setelah Anda menginstal Windows SDK pada EC2 instans Anda, Anda dapat menggunakannya untuk menghasilkan permintaan penandatanganan sertifikat (CSR). CSR adalah sertifikat yang tidak ditandatangani yang Anda kirimkan ke CA untuk ditandatangani. Dalam contoh ini, kami menggunakan `certreq` executable yang disertakan dengan Windows SDK untuk menghasilkan CSR.

Untuk menghasilkan CSR menggunakan `certreq` yang dapat dieksekusi

1. Connect ke EC2 instans Windows Anda. Untuk informasi selengkapnya, lihat [Connect to Your Instance](#) di Panduan EC2 Pengguna Amazon.
2. Buat file dengan nama `request.inf` dengan konten berikut ini. Ganti Subject informasi dengan detail organisasi Anda:

```
[Version]
Signature= $Windows NT$
[NewRequest]
Subject = "C=<Country>,CN=<www.website.com>,O=<Organization>,OU=<Organizational-Unit>,L=<City>,S=<State>"
RequestType=PKCS10
HashAlgorithm = SHA256
KeyAlgorithm = RSA
KeyLength = 2048
ProviderName = "CloudHSM Key Storage Provider"
KeyUsage = "CERT_DIGITAL_SIGNATURE_KEY_USAGE"
MachineKeySet = True
Exportable = False
```

Untuk penjelasan tentang setiap parameter, lihat [Dokumentasi Microsoft](#).

3. Jalankan `certreq.exe` untuk menghasilkan CSR:

```
certreq.exe -new request.inf request.csr
```

Perintah ini menghasilkan key pair baru di AWS CloudHSM cluster Anda dan menggunakan kunci pribadi untuk membuat CSR.

4. Kirimkan CSR ke CA Anda. Jika Anda menggunakan Windows Server CA, ikuti langkah-langkah berikut:

a. Buka alat CA:

```
certsrv.msc
```

- b. Di jendela baru, klik kanan nama server CA. Pilih Semua Tugas, lalu pilih Kirim permintaan baru.
- c. Arahkan ke lokasi `request.csr` dan pilih Buka.
- d. Perluas menu Server CA dan arahkan ke folder Permintaan Tertunda. Klik kanan permintaan yang baru saja Anda buat, pilih Semua Tugas, lalu pilih Masalah.
- e. Arahkan ke folder Sertifikat yang Diterbitkan.
- f. Pilih Buka untuk melihat sertifikat, dan kemudian memilih tab Detail.
- g. Pilih Salin ke File untuk memulai Wizard Ekspor Sertifikat. Simpan file Der-encoded X.509 ke lokasi aman sebagai `signedCertificate.cer`.
- h. Keluar dari alat CA dan jalankan perintah berikut untuk memindahkan file sertifikat ke Personal Certificate Store di Windows:

```
certreq.exe -accept signedCertificate.cer
```

Anda sekarang dapat menggunakan sertifikat yang diimpor untuk [menandatangani file](#).

Langkah 3: Tanda tangani file

Sekarang Anda memiliki Mage.exe dan sertifikat impor Anda, Anda dapat menandatangani file. Anda perlu mengetahui hash SHA-1 sertifikat, atau cap jempol. Cetak jempol memastikan bahwa Mage.exe hanya menggunakan sertifikat yang diverifikasi oleh AWS CloudHSM. Dalam contoh ini, kita gunakan PowerShell untuk mendapatkan hash sertifikat.

Untuk mendapatkan cap jempol sertifikat dan menggunakannya untuk menandatangani file

1. Arahkan ke direktori yang berisi `image.exe`. Lokasi defaultnya adalah:

```
C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.8.1 Tools\x64
```

2. Untuk membuat contoh file aplikasi menggunakan Mage.exe, jalankan perintah berikut:

```
mage.exe -New Application -ToFile C:\Users\Administrator\Desktop\sample.application
```

3. Buka PowerShell sebagai administrator dan jalankan perintah berikut:

```
Get-ChildItem -path cert:\LocalMachine\My
```

Salin Thumbprint, Key Container, dan Provider nilai dari output.

```
===== Certificate 1 =====
Serial Number: 7ad8df4f5bfaf5a3405d32aa4a01b85a
Issuer: CN=EC2AMAZ-0BVI3HH-CA
NotBefore: 2/4/2025 7:55 PM
NotAfter: 2/4/2030 8:05 PM
Subject: CN=EC2AMAZ-0BVI3HH-CA
CA Version: V0.0
Signature matches Public Key
Root Certificate: Subject matches Issuer...
CertHash(sha1): 08f295934325ee2c37ea01ab605cee81610d3be thumbprint
Key-Container = EC2AMAZ-0BVI3HH-CA key-container
Provider = CloudHSM Key Storage Provider provider
Signature test passed
CertUtil: -store command completed successfully.
```

4. Tanda tangani file Anda dengan menjalankan perintah berikut:

```
mage.exe -Sign -CertHash <thumbprint> -KeyContainer <keycontainer> -
CryptoProvider <CloudHSM Key Storage Provider/Cavium Key Storage Provider> C:\Users
\Administrator\Desktop\<sample.application>
```

Jika perintah berhasil, PowerShell mengembalikan pesan sukses.

5. Untuk memverifikasi tanda tangan pada file, gunakan perintah berikut:

```
mage.exe -Verify -CryptoProvider <CloudHSM Key Storage Provider/Cavium Key Storage
Provider> C:\Users\Administrator\Desktop\<sample.application>
```

Integrasi vendor pihak ketiga lainnya dengan AWS CloudHSM

Beberapa vendor pihak ketiga mendukung AWS CloudHSM sebagai akar kepercayaan. Ini berarti bahwa Anda dapat memanfaatkan solusi perangkat lunak pilihan Anda saat membuat dan menyimpan kunci yang mendasari dalam kluster CloudHSM Anda. Akibatnya, beban kerja Anda AWS dapat mengandalkan manfaat latensi, ketersediaan, keandalan, dan elastisitas CloudHSM. Daftar berikut mencakup vendor pihak ketiga yang mendukung CloudHSM.

 Note

AWS tidak mendukung atau menjamin vendor pihak ketiga mana pun.

- [Hashicorp Vault](#) adalah alat manajemen rahasia yang dirancang untuk memungkinkan kolaborasi dan tata kelola di seluruh organisasi. Ini mendukung AWS Key Management Service dan AWS CloudHSM sebagai akar kepercayaan untuk perlindungan tambahan.
- [Thycotic Secrets Server](#) membantu pelanggan mengelola kredensial sensitif di seluruh akun istimewa. Ini mendukung AWS CloudHSM sebagai akar kepercayaan.
- [Adaptor KMIP P6R](#) memungkinkan Anda memanfaatkan AWS CloudHSM instans Anda melalui antarmuka KMIP standar.
- [PrimeKey EJBCA](#) adalah solusi open source populer untuk PKI. Ini memungkinkan Anda untuk membuat dan menyimpan pasangan kunci dengan AWS CloudHSM aman.
- [Box KeySafe](#) menyediakan manajemen kunci enkripsi untuk konten cloud ke banyak organisasi dengan persyaratan keamanan, privasi, dan kepatuhan peraturan yang ketat. Pelanggan dapat lebih lanjut mengamankan KeySafe kunci langsung di dalam AWS Key Management Service atau tidak langsung AWS CloudHSM melalui AWS KMS Custom Key Store.
- [Insyde Software](#) mendukung AWS CloudHSM sebagai akar kepercayaan untuk penandatanganan firmware.
- [F5 BIG-IP LTM](#) mendukung AWS CloudHSM sebagai akar kepercayaan.
- [Cloudera Navigator Key HSM](#) memungkinkan Anda untuk menggunakan kluster CloudHSM Anda untuk membuat dan menyimpan kunci untuk Cloudera Navigator Key Trustee Server.
- [Venafi Trust Protection Platform](#) menyediakan manajemen identitas alat berat yang komprehensif untuk TLS, SSH, dan penandatanganan kode dengan pembuatan dan perlindungan kunci AWS CloudHSM.

Pemantauan AWS CloudHSM

Selain fitur logging yang dibangun ke dalam Client SDK, Anda juga dapat menggunakan AWS CloudTrail, Amazon CloudWatch Logs, dan Amazon CloudWatch untuk memantau AWS CloudHSM.

Log SDK Klien

Gunakan pencatatan SDK Klien untuk memantau informasi diagnostik dan pemecahan masalah dari aplikasi yang Anda buat.

CloudTrail

Gunakan CloudTrail untuk memantau semua panggilan API di AWS akun Anda, termasuk panggilan yang Anda buat untuk membuat dan menghapus klaster, modul keamanan perangkat keras (HSM), dan tag sumber daya.

CloudWatch Log

Gunakan CloudWatch Log untuk memantau log dari instans HSM Anda, yang mencakup peristiwa untuk membuat dan menghapus pengguna HSM, mengubah kata sandi pengguna, membuat dan menghapus kunci, dan banyak lagi.

CloudWatch

Gunakan CloudWatch untuk memantau kesehatan cluster Anda secara real time.

Topik

- [Bekerja dengan log SDK AWS CloudHSM klien](#)
- [Bekerja dengan AWS CloudTrail dan AWS CloudHSM](#)
- [Bekerja dengan CloudWatch Log Amazon dan Log AWS CloudHSM Audit](#)
- [Mendapatkan CloudWatch metrik untuk AWS CloudHSM](#)

Bekerja dengan log SDK AWS CloudHSM klien

Anda dapat mengambil log yang dihasilkan oleh SDK Klien. AWS CloudHSM menawarkan implementasi logging dengan Client SDK 3 dan Client SDK 5.

Topik

- [Pencatatan SDK 5 klien](#)
- [Pencatatan SDK 3 klien](#)

Pencatatan SDK 5 klien

Log SDK Klien 5 berisi informasi untuk setiap komponen dalam file bernama untuk komponen. Anda dapat menggunakan alat konfigurasi untuk SDK Klien 5 untuk mengatur setiap komponen.

Jika Anda tidak menentukan lokasi untuk file, sistem menulis log ke lokasi default:

AWS CloudHSM CLI

- Linux

```
/opt/cloudhsm/run/cloudhsm-cli.log
```

Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-cli.log
```

PKCS #11 library

- Linux

```
/opt/cloudhsm/run/cloudhsm-pkcs11.log
```

Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-pkcs11.log
```

OpenSSL Dynamic Engine

- Linux

```
stderr
```

JCE provider

- Linux

```
/opt/cloudhsm/run/cloudhsm-jce.log
```

Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-jce.log
```

Key Storage Provider

- Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-ksp.log
```

OpenSSL Dynamic Engine Provider

- Linux

```
stderr
```

Untuk informasi tentang cara mengonfigurasi logging untuk Client SDK 5, lihat alat Konfigurasi [SDK Klien 5](#)

Pencatatan SDK 3 klien

Log SDK 3 klien berisi informasi rinci dari daemon AWS CloudHSM klien. Lokasi log tergantung pada sistem operasi instance EC2 klien Amazon tempat Anda menjalankan daemon klien.

Amazon Linux

Di Amazon Linux, log AWS CloudHSM klien ditulis ke file bernama `/opt/cloudhsm/run/cloudhsm_client.log`. Anda dapat menggunakan `logrotate` atau alat serupa untuk memutar dan mengelola log ini.

Amazon Linux 2

Di Amazon Linux 2, log AWS CloudHSM Klien dikumpulkan dan disimpan dalam jurnal. Anda dapat menggunakan `journalctl` untuk melihat dan mengelola log ini. Misalnya, gunakan perintah berikut untuk melihat log AWS CloudHSM Klien.

```
journalctl -f -u cloudhsm-client
```

CentOS 7

Di CentOS 7, log AWS CloudHSM Klien dikumpulkan dan disimpan dalam jurnal. Anda dapat menggunakan `journalctl` untuk melihat dan mengelola log ini. Misalnya, gunakan perintah berikut untuk melihat log AWS CloudHSM Klien.

```
journalctl -f -u cloudhsm-client
```

CentOS 8

Di CentOS 8, log AWS CloudHSM Klien dikumpulkan dan disimpan dalam jurnal. Anda dapat menggunakan `journalctl` untuk melihat dan mengelola log ini. Misalnya, gunakan perintah berikut untuk melihat log AWS CloudHSM Klien.

```
journalctl -f -u cloudhsm-client
```

RHEL 7

Di Red Hat Enterprise Linux 7, log AWS CloudHSM Klien dikumpulkan dan disimpan dalam jurnal. Anda dapat menggunakan `journalctl` untuk melihat dan mengelola log ini. Misalnya, gunakan perintah berikut untuk melihat log AWS CloudHSM Klien.

```
journalctl -f -u cloudhsm-client
```

RHEL 8

Di Red Hat Enterprise Linux 8, log AWS CloudHSM Klien dikumpulkan dan disimpan dalam jurnal. Anda dapat menggunakan `journalctl` untuk melihat dan mengelola log ini. Misalnya, gunakan perintah berikut untuk melihat log AWS CloudHSM Klien.

```
journalctl -f -u cloudhsm-client
```

Ubuntu 16.04

Di Ubuntu 16.04, log AWS CloudHSM Klien dikumpulkan dan disimpan dalam jurnal. Anda dapat menggunakan `journalctl` untuk melihat dan mengelola log ini. Misalnya, gunakan perintah berikut untuk melihat log AWS CloudHSM Klien.

```
journalctl -f -u cloudhsm-client
```

Ubuntu 18.04

Di Ubuntu 18.04, log AWS CloudHSM Klien dikumpulkan dan disimpan dalam jurnal. Anda dapat menggunakan `journalctl` untuk melihat dan mengelola log ini. Misalnya, gunakan perintah berikut untuk melihat log AWS CloudHSM Klien.

```
journalctl -f -u cloudhsm-client
```

Windows

- Untuk klien Windows 1.1.2+:

AWS CloudHSM log klien ditulis ke `cloudhsm-kps.log` file di folder file AWS CloudHSM program (`C:\Program Files\Amazon\CloudHSM\`). Setiap nama file log diberi akhiran dengan stempel waktu yang menunjukkan kapan klien dimulai. AWS CloudHSM

- Untuk klien Windows 1.1.1 dan yang lebih lama:

Log klien tidak ditulis ke file. Log ditampilkan di prompt perintah atau di PowerShell jendela tempat Anda memulai AWS CloudHSM klien.

Bekerja dengan AWS CloudTrail dan AWS CloudHSM

AWS CloudHSM terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di AWS CloudHSM. CloudTrail menangkap semua panggilan API untuk AWS CloudHSM sebagai peristiwa. Panggilan yang diambil termasuk panggilan dari AWS CloudHSM konsol dan panggilan kode ke operasi AWS CloudHSM API. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara berkelanjutan ke bucket Amazon S3, termasuk acara untuk. AWS CloudHSM Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan

yang dibuat AWS CloudHSM, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat [Panduan AWS CloudTrail Pengguna](#). Untuk daftar lengkap operasi AWS CloudHSM API, lihat [Tindakan](#) di Referensi AWS CloudHSM API.

AWS CloudHSM informasi di CloudTrail

CloudTrail diaktifkan di AWS akun Anda saat Anda membuat akun. Ketika aktivitas terjadi di AWS CloudHSM, aktivitas tersebut dicatat dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di akun AWS. Untuk informasi selengkapnya, lihat [Melihat Acara dengan Riwayat CloudTrail Acara](#).

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk acara untuk AWS CloudHSM, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, ketika Anda membuat jejak di konsol tersebut, jejak tersebut diterapkan ke semua Wilayah AWS. Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat berikut:

- [Gambaran Umum untuk Membuat Jejak](#)
- [CloudTrail Layanan dan Integrasi yang Didukung](#)
- [Mengkonfigurasi Notifikasi Amazon SNS untuk CloudTrail](#)
- [Menerima File CloudTrail Log dari Beberapa Wilayah](#) dan [Menerima File CloudTrail Log dari Beberapa Akun](#)

CloudTrail mencatat semua AWS CloudHSM operasi, termasuk operasi hanya-baca, seperti `DescribeClusters` dan `ListTags`, dan operasi manajemen, seperti, `InitializeClusterCreatHsm`, dan `DeleteBackup`

Setiap entri peristiwa atau log berisi informasi tentang entitas yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut ini:

- Apakah permintaan itu dibuat dengan kredensial pengguna root atau AWS Identity and Access Management (IAM).

- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna gabungan.
- Apakah permintaan itu dibuat oleh AWS layanan lain.

Untuk informasi lain, lihat [Elemen userIdentity CloudTrail](#).

Memahami entri file AWS CloudHSM log

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, jadi file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan AWS CloudHSM CreateHsm tindakan.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AR0AJZVM5NEGZSTCITAMM:ExampleSession",
    "arn": "arn:aws:sts::111122223333:assumed-role/AdminRole/ExampleSession",
    "accountId": "111122223333",
    "accessKeyId": "ASIAIY22AX6VRYNBJSA",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2017-07-11T03:48:44Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AR0AJZVM5NEGZSTCITAMM",
        "arn": "arn:aws:iam::111122223333:role/AdminRole",
        "accountId": "111122223333",
        "userName": "AdminRole"
      }
    }
  },
  "eventTime": "2017-07-11T03:50:45Z",
```

```

    "eventSource": "cloudhsm.amazonaws.com",
    "eventName": "CreateHsm",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "205.251.233.179",
    "userAgent": "aws-internal/3",
    "requestParameters": {
        "availabilityZone": "us-west-2b",
        "clusterId": "cluster-fw7mh6mayb5"
    },
    "responseElements": {
        "hsm": {
            "eniId": "eni-65338b5a",
            "clusterId": "cluster-fw7mh6mayb5",
            "state": "CREATE_IN_PROGRESS",
            "eniIp": "10.0.2.7",
            "hsmId": "hsm-6lz2hfmnzb5",
            "subnetId": "subnet-02c28c4b",
            "availabilityZone": "us-west-2b"
        }
    },
    "requestID": "1dae0370-65ec-11e7-a770-6578d63de907",
    "eventID": "b73a5617-8508-4c3d-900d-aa8ac9b31d08",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}

```

Bekerja dengan CloudWatch Log Amazon dan Log AWS CloudHSM Audit

Ketika HSM di akun Anda menerima perintah dari [alat baris AWS CloudHSM perintah](#) atau [pustaka perangkat lunak](#), ia mencatat eksekusi perintahnya dalam bentuk log audit. Log audit HSM mencakup semua [perintah manajemen](#) yang dimulai klien, termasuk yang membuat dan menghapus HSM, masuk dan keluar dari HSM, dan serta mengelola pengguna dan kunci. Log ini memberikan catatan tindakan yang dapat diandalkan yang telah mengubah keadaan HSM.

AWS CloudHSM mengumpulkan log audit HSM Anda dan mengirimkannya ke [Amazon CloudWatch Logs atas nama](#) Anda. Anda dapat menggunakan fitur CloudWatch Log untuk mengelola Log AWS CloudHSM Audit Anda, termasuk mencari dan memfilter log dan mengeksport data log ke Amazon S3. Anda dapat bekerja dengan log audit HSM Anda di [CloudWatch konsol Amazon](#) atau menggunakan perintah CloudWatch Log di [AWS CLI](#) dan [CloudWatch Log SDKs](#).

Topik

- [Cara kerja pencatatan audit HSM](#)
- [Melihat log AWS CloudHSM audit di CloudWatch Log](#)
- [Menafsirkan log AWS CloudHSM audit](#)
- [AWS CloudHSM referensi log audit](#)

Cara kerja pencatatan audit HSM

Pencatatan audit diaktifkan secara otomatis di semua AWS CloudHSM cluster. Itu tidak dapat dinonaktifkan atau dimatikan, dan tidak ada pengaturan yang dapat AWS CloudHSM mencegah mengeksport CloudWatch log ke Log. Setiap peristiwa log memiliki stempel waktu dan nomor urut yang menunjukkan urutan peristiwa dan membantu Anda mendeteksi setiap gangguan log.

Setiap instans HSM menghasilkan log sendiri. Log audit dari berbagai HSMs, bahkan yang berada di cluster yang sama, cenderung berbeda. Misalnya, hanya HSM pertama di setiap klaster mencatat inisialisasi HSM. Peristiwa inisialisasi tidak muncul di log HSMs yang dikloning dari cadangan. Demikian pula, ketika Anda membuat kunci, HSM yang menghasilkan kunci mencatat peristiwa pembuatan kunci. Yang lain HSMs di cluster merekam peristiwa ketika mereka menerima kunci melalui sinkronisasi.

AWS CloudHSM mengumpulkan log dan mempostingnya ke CloudWatch Log di akun Anda. Untuk berkomunikasi dengan layanan CloudWatch Log atas nama Anda, AWS CloudHSM gunakan peran [terkait layanan](#). Kebijakan IAM yang terkait dengan peran memungkinkan AWS CloudHSM untuk melakukan hanya tugas yang diperlukan untuk mengirim log audit ke CloudWatch Log.

Important

Jika Anda membuat klaster sebelum 20 Januari 2018, dan belum membuat peran terkait layanan terlampir, Anda harus membuatnya secara manual. Hal ini diperlukan CloudWatch untuk menerima log audit dari AWS CloudHSM klaster Anda. Untuk informasi lebih lanjut tentang pembuatan peran terkait layanan, lihat [Memahami Peran Terkait Layanan](#), serta [Membuat Peran Terkait Layanan](#) dalam Panduan Pengguna IAM.

Melihat log AWS CloudHSM audit di CloudWatch Log

Amazon CloudWatch Logs mengatur log audit ke dalam grup log dan, dalam grup log, ke dalam aliran log. Setiap entri log adalah acara. AWS CloudHSM membuat satu grup log untuk setiap cluster dan satu aliran log untuk setiap HSM di cluster. Anda tidak perlu membuat komponen CloudWatch Log atau mengubah pengaturan apa pun.

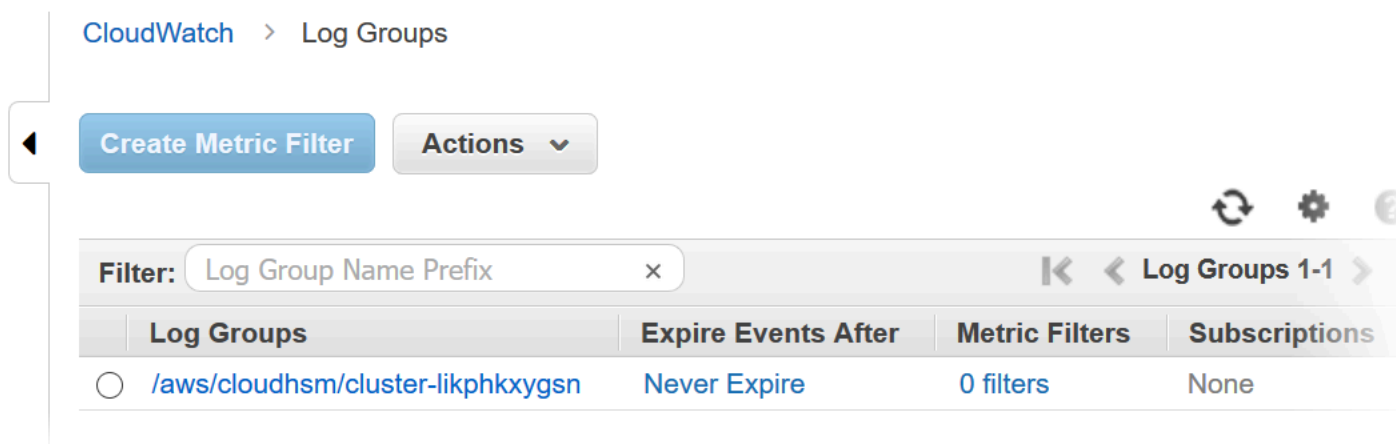
- Nama grup log adalah `/aws/cloudhsm/<cluster ID>`; misalnya `/aws/cloudhsm/cluster-likphkxygsn`. Saat Anda menggunakan nama grup log dalam PowerShell perintah AWS CLI atau, pastikan untuk melampirkannya dalam tanda kutip ganda.
- Nama pengaliran log adalah ID HSM; sebagai contohnya, `hsm-nwbbiqbj4jk`.

Secara umum, terdapat satu pengaliran log untuk setiap HSM. Namun, setiap tindakan yang mengubah ID HSM, seperti ketika HSM gagal dan diganti, membuat pengaliran log baru.

Untuk informasi selengkapnya tentang konsep CloudWatch Log, lihat [Konsep](#) di Panduan Pengguna CloudWatch Log Amazon.

[Anda dapat melihat log audit untuk HSM dari halaman CloudWatch Log di Konsol Manajemen AWS, perintah CloudWatch Log di, PowerShellcmdlet CloudWatch Log AWS CLI, atau Log. CloudWatch SDKs](#) Untuk petunjuknya, [lihat Melihat Data Log](#) di Panduan Pengguna Amazon CloudWatch Logs.

Sebagai contoh, gambar berikut menunjukkan grup log untuk klaster `cluster-likphkxygsn` di Konsol Manajemen AWS.



Ketika Anda memilih nama grup log cluster, Anda dapat melihat aliran log untuk masing-masing HSMs di cluster. Gambar berikut menunjukkan aliran log untuk HSMs di `cluster-likphkxygsn` cluster.

CloudWatch > Log Groups > Streams for /aws/cloudhsm/cluster-likphkxygsn

Search Log Group Create Log Stream Delete Log Stream

Filter: Log Stream Name Prefix x Log Streams 1-2

☐	Log Streams	Last Event Time
☐	hsm-aht4p3sgs3c	2017-12-28 06:12 UTC-8
☐	hsm-xkvjp4wk5o3	2017-12-28 06:12 UTC-8

Bila Anda memilih nama pengaliran log HSM, Anda dapat melihat peristiwa di log audit. Sebagai contoh, peristiwa ini, yang memiliki nomor urut 0x0 dan Opcode dari CN_INIT_TOKEN, biasanya merupakan peristiwa pertama untuk HSM pertama di setiap klaster. Peristiwa ini mencatat inisialisasi HSM di klaster.

Filter events

Time (UTC +00:00)	Message
2017-12-19	Time: 12/19/17 21:01:16.962174, usecs:1513717276962174 Sequence No : 0x0 Reboot counter : 0xe8 Command Type(hex) : CN_MGMT_CMD (0x0) Opcode : CN_INIT_TOKEN (0x1) Session Handle : 0x1004001 Response : 0:HSM Return: SUCCESS Log type : MINIMAL_LOG_ENTRY (0)

Anda dapat menggunakan semua banyak fitur di CloudWatch Log untuk mengelola log audit Anda. Misalnya, Anda dapat menggunakan fitur Filter peristiwa untuk menemukan teks tertentu dalam suatu peristiwa, seperti CN_CREATE_USER Opcode.

Untuk menemukan semua peristiwa yang tidak termasuk teks yang ditentukan, tambahkan tanda minus (-) sebelum teks. Sebagai contoh, untuk mencari peristiwa yang tidak termasuk CN_CREATE_USER, masukkan -CN_CREATE_USER.

CN_CREATE_USER	
Time (UTC +00:00)	Message
2017-12-20	
No older events	
▼ 00:04:53	Time: 12/20/17 00:04:53.635826, u
Time: 12/20/17 00:04:53.635826, usecs:1513728293635826 Sequence No : 0x13a Reboot counter : 0xe8 Command Type(hex) : CN_MGMT_CMD (0x0) Opcode : CN_CREATE_USER (0x3) Session Handle : 0x1014006 Response : 0:HSM Return: SUCCESS Log type : MGMT_USER_DETAILS_LOG (2) User Name : testuser User Type : CN_CRYPTO_USER (1)	

Menafsirkan log AWS CloudHSM audit

Peristiwa dalam log audit HSM memiliki bidang standar. Beberapa jenis peristiwa memiliki bidang tambahan yang menangkap informasi yang berguna tentang peristiwa tersebut. Misalnya, peristiwa login pengguna dan manajemen pengguna termasuk nama pengguna dan jenis pengguna dari pengguna. Perintah manajemen kunci termasuk handel kunci.

Beberapa bidang memberikan informasi yang sangat penting. Opcode mengidentifikasi perintah manajemen yang sedang dicatat. Sequence No mengidentifikasi peristiwa dalam pengaliran log dan menunjukkan urutan pencatatannya.

Sebagai contoh, contoh peristiwa berikut ini adalah peristiwa kedua (Sequence No: 0x1) dalam pengaliran log untuk HSM. Peristiwa ini menunjukkan HSM menghasilkan kunci enkripsi kata sandi, yang merupakan bagian dari rutinitas awal.

```
Time: 12/19/17 21:01:17.140812, usecs:1513717277140812
Sequence No : 0x1
Reboot counter : 0xe8
```

```
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_GEN_PSWD_ENC_KEY (0x1d)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MINIMAL_LOG_ENTRY (0)
```

Bidang berikut ini umum untuk setiap AWS CloudHSM peristiwa di log audit.

Waktu

Waktu terjadinya peristiwa di zona waktu UTC. Waktu ditampilkan sebagai waktu yang dapat dibaca manusia dan waktu Unix dalam mikrodetik.

Penghitung reboot

Penghitung ordinal persisten 32-bit yang bertambah saat perangkat keras HSM reboot.

Semua peristiwa dalam pengaliran log memiliki nilai penghitung reboot. Namun, penghitung reboot counter mungkin tidak unik untuk pengaliran log, karena dapat berbeda di berbagai instans HSM yang berbeda di klaster yang sama.

No Urut

Penghitung ordinal 64-bit yang bertambah untuk setiap peristiwa log. Peristiwa pertama di setiap pengaliran log memiliki nomor urut 0x0. Seharusnya tidak ada celah di nilai Sequence No. Nomor urut unik hanya dalam pengaliran log.

Jenis perintah

Nilai heksadesimal yang mewakili kategori perintah. Perintah dalam pengaliran log AWS CloudHSM memiliki jenis perintah CN_MGMT_CMD (0x0) atau CN_CERT_AUTH_CMD (0x9).

Opcode

Mengidentifikasi perintah manajemen yang dieksekusi. Untuk daftar Opcode nilai dalam log AWS CloudHSM audit, lihat [AWS CloudHSM referensi log audit](#).

Handel sesi

Mengidentifikasi sesi di mana perintah dijalankan dan peristiwa dicatat.

Jawaban

Mencatat respons terhadap perintah manajemen. Anda dapat mencari bidang Response untuk SUCCESS dan nilai ERROR.

Jenis log

Menunjukkan jenis log AWS CloudHSM log yang merekam perintah.

- MINIMAL_LOG_ENTRY (0)
- MGMT_KEY_DETAILS_LOG (1)
- MGMT_USER_DETAILS_LOG (2)
- GENERIC_LOG

Contoh peristiwa log audit

Peristiwa dalam pengaliran log mencatat riwayat HSM dari pembuatan hingga penghapusan. Anda dapat menggunakan log untuk meninjau siklus hidup Anda HSMs dan mendapatkan wawasan tentang operasinya. Ketika Anda menafsirkan peristiwa, catat Opcode, yang menunjukkan perintah atau tindakan manajemen, dan Sequence No, yang menunjukkan urutan peristiwa.

Topik

- [Contoh: Inisialisasi HSM pertama dalam sebuah cluster](#)
- [Login dan logout acara](#)
- [Contoh: Membuat dan menghapus pengguna](#)
- [Contoh: Membuat dan menghapus key pair](#)
- [Contoh: Menghasilkan dan menyinkronkan kunci](#)
- [Contoh: Ekspor kunci](#)
- [Contoh: Impor kunci](#)
- [Contoh: Bagikan dan batalkan pembagian kunci](#)

Contoh: Inisialisasi HSM pertama dalam sebuah cluster

Aliran log audit untuk HSM pertama di setiap cluster berbeda secara signifikan dari aliran log lainnya HSMs di cluster. Log audit untuk HSM pertama di setiap klaster mencatat pembuatan dan inisialisasinya. Log tambahan HSMs di cluster, yang dihasilkan dari cadangan, dimulai dengan peristiwa login.

⚠ Important

Entri inisialisasi berikut tidak akan muncul di CloudWatch log kluster yang diinisialisasi sebelum rilis fitur pencatatan audit CloudHSM (30 Agustus 2018). Untuk informasi lebih lanjut, lihat [Riwayat Dokumentasi](#).

Contoh peristiwa berikut muncul dalam pengaliran log untuk HSM pertama dalam sebuah kluster. Peristiwa pertama dalam log — peristiwa dengan Sequence No 0x0 — mewakili perintah untuk menginisialisasi HSM (CN_INIT_TOKEN). Tanggapan menunjukkan bahwa perintah berhasil (Response : 0: HSM Return: SUCCESS).

```
Time: 12/19/17 21:01:16.962174, usecs:1513717276962174
Sequence No : 0x0
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_INIT_TOKEN (0x1)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MINIMAL_LOG_ENTRY (0)
```

Peristiwa kedua dalam contoh ini, pengaliran log (Sequence No 0x1), mencatat perintah untuk membuat kunci enkripsi kata sandi yang menggunakan HSM (CN_GEN_PSWD_ENC_KEY).

Ini adalah urutan awalan khas untuk HSM pertama di setiap kluster. Karena selanjutnya HSMs dalam cluster yang sama adalah klon dari yang pertama, mereka menggunakan kunci enkripsi kata sandi yang sama.

```
Time: 12/19/17 21:01:17.140812, usecs:1513717277140812
Sequence No : 0x1
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_GEN_PSWD_ENC_KEY (0x1d)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MINIMAL_LOG_ENTRY (0)
```

Peristiwa ketiga dalam contoh pengaliran log ini (Sequence No 0x2) adalah pembuatan [pengguna peralatan \(AU\)](#), yang merupakan layanan AWS CloudHSM . Peristiwa yang melibatkan pengguna HSM termasuk bidang tambahan untuk nama pengguna dan jenis pengguna.

```
Time: 12/19/17 21:01:17.174902, usecs:1513717277174902
Sequence No : 0x2
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_CREATE_APPLIANCE_USER (0xfc)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : app_user
User Type : CN_APPLIANCE_USER (5)
```

Peristiwa keempat dalam contoh ini pengaliran log (Sequence No 0x3) mencatat peristiwa CN_INIT_DONE, yang melengkapi inisialisasi HSM.

```
Time: 12/19/17 21:01:17.298914, usecs:1513717277298914
Sequence No : 0x3
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_INIT_DONE (0x95)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MINIMAL_LOG_ENTRY (0)
```

Anda dapat mengikuti peristiwa yang tersisa dalam urutan memulai. Peristiwa ini mungkin termasuk beberapa peristiwa login dan logout, dan pembuatan kunci enkripsi kunci (KEK). Peristiwa berikut mencatat perintah yang mengubah kata sandi [petugas precrypto \(PRECO\)](#). Perintah ini mengaktifkan kluster.

```
Time: 12/13/17 23:04:33.846554, usecs:1513206273846554
Sequence No: 0x1d
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_CHANGE_PSWD (0x9)
Session Handle: 0x2010003
Response: 0:HSM Return: SUCCESS
Log type: MGMT_USER_DETAILS_LOG (2)
User Name: admin
User Type: CN_CRYPT0_PRE_OFFICER (6)
```

Login dan logout acara

Saat menafsirkan log audit Anda, catat peristiwa yang mencatat pengguna masuk dan keluar dari HSM. Peristiwa ini membantu Anda menentukan pengguna yang bertanggung jawab atas perintah manajemen yang muncul secara berurutan antara perintah login dan logout.

Misalnya, entri log ini mencatat login oleh petugas kripto (CO) bernama admin. Nomor urut, 0x0, menunjukkan bahwa ini adalah peristiwa pertama dalam pengaliran log.

Ketika pengguna log ke HSM, yang lain HSMs di cluster juga merekam peristiwa login untuk pengguna. Anda dapat menemukan peristiwa login yang sesuai di aliran log lain HSMs di cluster tak lama setelah peristiwa login awal.

```
Time: 01/16/18 01:48:49.824999, usecs:1516067329824999
Sequence No : 0x0
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7014006
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : admin
User Type : CN_CRYPT0_OFFICER (2)
```

Contoh peristiwa berikut ini mencatat petugas kripto (CO) admin keluar. Nomor urut, 0x2, menunjukkan bahwa ini adalah peristiwa ketiga dalam pengaliran log.

Jika pengguna yang login menutup sesi tanpa keluar, pengaliran log termasuk CN_APP_FINALIZE atau peristiwa tutup sesi (CN_SESSION_CLOSE), bukan peristiwa CN_LOGOUT. Berbeda dengan peristiwa login, peristiwa keluar ini biasanya dicatat hanya oleh HSM yang mengeksekusi perintah.

```
Time: 01/16/18 01:49:55.993404, usecs:1516067395993404
Sequence No : 0x2
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGOUT (0xe)
Session Handle : 0x7014000
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : admin
User Type : CN_CRYPT0_OFFICER (2)
```

Jika upaya login gagal karena nama pengguna tidak valid, HSM mencatat peristiwa CN_LOGIN dengan nama pengguna dan jenis yang disediakan dalam perintah login. Tanggapan menampilkan pesan kesalahan 157, yang menjelaskan bahwa nama pengguna tidak ada.

```
Time: 01/24/18 17:41:39.037255, usecs:1516815699037255
Sequence No : 0x4
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0xc008002
Response : 157:HSM Error: user isn't initialized or user with this name doesn't exist
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : ExampleUser
User Type : CN_CRYPT0_USER (1)
```

Jika upaya login gagal karena kata sandi tidak valid, HSM mencatat peristiwa CN_LOGIN dengan nama pengguna dan jenis yang disediakan dalam perintah login. Tanggapan menampilkan pesan kesalahan dengan kode kesalahan RET_USER_LOGIN_FAILURE.

```
Time: 01/24/18 17:44:25.013218, usecs:1516815865013218
Sequence No : 0x5
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0xc008002
Response : 163:HSM Error: RET_USER_LOGIN_FAILURE
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPT0_USER (1)
```

Contoh: Membuat dan menghapus pengguna

Contoh ini menunjukkan peristiwa log yang dicatat ketika seorang petugas krypto (CO) membuat dan menghapus pengguna.

Peristiwa pertama mencatat CO, admin, masuk ke HSM. Nomor urut 0x0 menunjukkan bahwa ini adalah peristiwa pertama dalam pengaliran log. Nama dan jenis pengguna yang masuk termasuk dalam acara tersebut.

```
Time: 01/16/18 01:48:49.824999, usecs:1516067329824999
```

```
Sequence No : 0x0
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7014006
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : admin
User Type : CN_CRYPT0_OFFICER (2)
```

Peristiwa berikutnya dalam pengaliran log (urutan 0x1) mencatat CO yang membuat pengguna kriptografi baru (CU). Nama dan jenis pengguna baru disertakan dalam peristiwa tersebut.

```
Time: 01/16/18 01:49:39.437708, usecs:1516067379437708
Sequence No : 0x1
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_CREATE_USER (0x3)
Session Handle : 0x7014006
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : bob
User Type : CN_CRYPT0_USER (1)
```

Kemudian, CO membuat petugas kriptografi (CO) lain, *alice*. Nomor urut menunjukkan bahwa tindakan ini diikuti yang sebelumnya tanpa tindakan intervensi.

```
Time: 01/16/18 01:49:55.993404, usecs:1516067395993404
Sequence No : 0x2
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_CREATE_CO (0x4)
Session Handle : 0x7014007
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : alice
User Type : CN_CRYPT0_OFFICER (2)
```

Kemudian, CO bernama *admin* masuk dan menghapus petugas kriptografi (CO) bernama *alice*. HSM mencatat peristiwa *CN_DELETE_USER*. Nama dan jenis pengguna yang dihapus disertakan dalam peristiwa tersebut.

```
Time: 01/23/18 19:58:23.451420, usecs:1516737503451420
Sequence No : 0xb
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_DELETE_USER (0xa1)
Session Handle : 0x7014007
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : alice
User Type : CN_CRYPT0_OFFICER (2)
```

Contoh: Membuat dan menghapus key pair

Contoh ini menunjukkan peristiwa yang dicatat dalam log audit HSM ketika Anda membuat dan menghapus pasangan kunci.

Peristiwa berikut mencatat pengguna krypto (CU) bernama `crypto_user` masuk ke HSM.

```
Time: 12/13/17 23:09:04.648952, usecs:1513206544648952
Sequence No: 0x28
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_LOGIN (0xd)
Session Handle: 0x2014005
Response: 0:HSM Return: SUCCESS
Log type: MGMT_USER_DETAILS_LOG (2)
User Name: crypto_user
User Type: CN_CRYPT0_USER (1)
```

Selanjutnya, CU menghasilkan pasangan kunci (`CN_GENERATE_KEY_PAIR`). Kunci privat memiliki handel kunci 131079. Kunci publik memiliki handel kunci 131078.

```
Time: 12/13/17 23:09:04.761594, usecs:1513206544761594
Sequence No: 0x29
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_GENERATE_KEY_PAIR (0x19)
Session Handle: 0x2014004
Response: 0:HSM Return: SUCCESS
Log type: MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle: 131079
```

```
Public Key Handle: 131078
```

CU segera menghapus pasangan kunci. Peristiwa CN_DESTROY_OBJECT mencatat penghapusan kunci publik (131078).

```
Time: 12/13/17 23:09:04.813977, usecs:1513206544813977
Sequence No: 0x2a
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_DESTROY_OBJECT (0x11)
Session Handle: 0x2014004
Response: 0:HSM Return: SUCCESS
Log type: MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle: 131078
Public Key Handle: 0
```

Kemudian, peristiwa CN_DESTROY_OBJECT kedua mencatat penghapusan kunci privat (131079).

```
Time: 12/13/17 23:09:04.815530, usecs:1513206544815530
Sequence No: 0x2b
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_DESTROY_OBJECT (0x11)
Session Handle: 0x2014004
Response: 0:HSM Return: SUCCESS
Log type: MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle: 131079
Public Key Handle: 0
```

Terakhir, CU log out.

```
Time: 12/13/17 23:09:04.817222, usecs:1513206544817222
Sequence No: 0x2c
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_LOGOUT (0xe)
Session Handle: 0x2014004
Response: 0:HSM Return: SUCCESS
Log type: MGMT_USER_DETAILS_LOG (2)
User Name: crypto_user
User Type: CN_CRYPT0_USER (1)
```


Contoh: Menghasilkan dan menyinkronkan kunci

Contoh ini menunjukkan efek membuat kunci dalam cluster dengan beberapa HSMs. Kunci dihasilkan pada satu HSM, diekstraksi dari HSM sebagai objek bertopeng, dan dimasukkan di yang lain HSMs sebagai objek bertopeng.

Note

Alat klien mungkin gagal untuk menyinkronkan kunci. Atau perintah mungkin termasuk `min_srv` parameter, yang menyinkronkan kunci hanya untuk jumlah HSMs yang ditentukan. Dalam kedua kasus, AWS CloudHSM layanan menyinkronkan kunci ke yang lain HSMs di cluster. Karena HSMs catatan hanya perintah manajemen sisi klien di log mereka, sinkronisasi sisi server tidak dicatat dalam log HSM.

Pertama, pertimbangkan pengaliran log dari HSM yang menerima dan mengeksekusi perintah. Pengaliran log dinamai ID HSM, `hsm-abcde123456`, tetapi ID HSM tidak muncul dalam peristiwa log.

Pertama, pengguna kripto (CU) `testuser` masuk ke HSM `hsm-abcde123456`.

```
Time: 01/24/18 00:39:23.172777, usecs:1516754363172777
Sequence No : 0x0
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0xc008002
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPT0_USER (1)
```

CU menjalankan [exSymKey](#) perintah untuk menghasilkan kunci simetris. HSM `hsm-abcde123456` menghasilkan kunci simetris dengan handel kunci 262152. HSM mencatat peristiwa `CN_GENERATE_KEY` di log-nya.

```
Time: 01/24/18 00:39:30.328334, usecs:1516754370328334
Sequence No : 0x1
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
```

```
Opcode : CN_GENERATE_KEY (0x17)
Session Handle : 0xc008004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 262152
Public Key Handle : 0
```

Peristiwa berikutnya dalam pengaliran log untuk hsm-abcde123456 mencatat langkah pertama dalam proses sinkronisasi kunci. Kunci baru (handel kunci 262152) diekstraksi dari HSM sebagai objek tertutup.

```
Time: 01/24/18 00:39:30.330956, usecs:1516754370330956
Sequence No : 0x2
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_EXTRACT_MASKED_OBJECT_USER (0xf0)
Session Handle : 0xc008004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 262152
Public Key Handle : 0
```

Sekarang pertimbangkan pengaliran log untuk HSM hsm-zyxwv987654, HSM lain di klaster yang sama. Pengaliran log juga mencakup peristiwa login untuk CU testuser. Nilai waktu menunjukkan hal itu terjadi tak lama setelah pengguna masuk ke HSM hsm-abcde123456.

```
Time: 01/24/18 00:39:23.199740, usecs:1516754363199740
Sequence No : 0xd
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7004004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPT0_USER (1)
```

Pengaliran log untuk HSM ini tidak memiliki peristiwa CN_GENERATE_KEY. Namun, nilai tidak memiliki peristiwa yang mencatat sinkronisasi kunci untuk HSM ini. Peristiwa CN_INSERT_MASKED_OBJECT_USER mencatat penerimaan kunci 262152 sebagai objek tertutup. Sekarang kunci 262152 ada pada keduanya HSMs di cluster.

```
Time: 01/24/18 00:39:30.408950, usecs:1516754370408950
Sequence No : 0xe
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_INSERT_MASKED_OBJECT_USER (0xf1)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 262152
Public Key Handle : 0
```

Ketika pengguna CU keluar, peristiwa CN_LOGOUT ini muncul hanya dalam pengaliran log HSM yang menerima perintah.

Contoh: Ekspor kunci

Contoh ini menunjukkan peristiwa log audit yang direkam saat pengguna krypto (CU) mengekspor kunci dari klaster dengan beberapa HSMs.

Peristiwa berikut mencatat CU (testuser) masuk ke [key_mgmt_util](#).

```
Time: 01/24/18 19:42:22.695884, usecs:1516822942695884
Sequence No : 0x26
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7004004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPT0_USER (1)
```

CU menjalankan [exSymKey](#) perintah untuk mengekspor kunci7, kunci AES 256-bit. Perintah menggunakan kunci6, kunci AES 256-bit pada HSMs, sebagai kunci pembungkus.

HSM yang menerima catatan perintah CN_WRAP_KEY peristiwa untuk kunci 7, kunci yang sedang diekspor.

```
Time: 01/24/18 19:51:12.860123, usecs:1516823472860123
Sequence No : 0x27
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
```

```
Opcode : CN_WRAP_KEY (0x1a)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 7
Public Key Handle : 0
```

Kemudian, HSM mencatat peristiwa CN_NIST_AES_WRAP untuk kunci pembungkus, kunci 6. Kuncinya dibungkus dan kemudian segera dibuka, tapi HSM mencatat hanya satu peristiwa.

```
Time: 01/24/18 19:51:12.905257, usecs:1516823472905257
Sequence No : 0x28
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_NIST_AES_WRAP (0x1e)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 6
Public Key Handle : 0
```

Perintah `exSymKey` menulis kunci yang diekspor ke file, tetapi tidak mengubah kunci pada HSM. Akibatnya, tidak ada peristiwa yang sesuai di log lain HSMs di cluster.

Contoh: Impor kunci

Contoh ini menunjukkan peristiwa log audit yang direkam saat Anda mengimpor kunci ke HSMs dalam klaster. Dalam contoh ini, pengguna kriptografi (CU) menggunakan `imSymKey` perintah untuk mengimpor kunci AES ke dalam HSMs. Perintah menggunakan kunci 6 sebagai kunci pembungkus.

HSM yang menerima perintah pertama mencatat CN_NIST_AES_WRAP peristiwa untuk 6 kunci, kunci pembungkus.

```
Time: 01/24/18 19:58:23.170518, usecs:1516823903170518
Sequence No : 0x29
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_NIST_AES_WRAP (0x1e)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 6
```

```
Public Key Handle : 0
```

Kemudian, HSM mencatat peristiwa CN_UNWRAP_KEY yang mewakili operasi impor. Kunci yang diimpor mendapat handel kunci dari 11.

```
Time: 01/24/18 19:58:23.200711, usecs:1516823903200711
Sequence No : 0x2a
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_UNWRAP_KEY (0x1b)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 11
Public Key Handle : 0
```

Ketika kunci baru dihasilkan atau diimpor, alat klien secara otomatis mencoba untuk menyinkronkan kunci baru ke yang lain HSMs di cluster. Dalam hal ini, HSM mencatat peristiwa CN_EXTRACT_MASKED_OBJECT_USER ketika kunci 11 diekstrakdisi dari HSM sebagai objek tertutup.

```
Time: 01/24/18 19:58:23.203350, usecs:1516823903203350
Sequence No : 0x2b
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_EXTRACT_MASKED_OBJECT_USER (0xf0)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 11
Public Key Handle : 0
```

Aliran log lain HSMs di cluster mencerminkan kedatangan kunci yang baru diimpor.

Sebagai contoh, acara ini tercatat dalam pengaliran log HSM yang berbeda di klaster yang sama. Peristiwa CN_INSERT_MASKED_OBJECT_USER mencatat kedatangan objek tertutup yang mewakili kunci 11.

```
Time: 01/24/18 19:58:23.286793, usecs:1516823903286793
Sequence No : 0xb
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
```

```

Opcode : CN_INSERT_MASKED_OBJECT_USER (0xf1)
Session Handle : 0xc008004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 11
Public Key Handle : 0

```

Contoh: Bagikan dan batalkan pembagian kunci

Contoh ini menunjukkan peristiwa log audit yang dicatat ketika pengguna kriptografi (CU) berbagi atau batal berbagi kunci privat ECC dengan pengguna kriptografi lainnya. CU menggunakan perintah [shareKey](#) dan menyediakan handle kunci, ID pengguna, dan nilai 1 untuk berbagi atau memberi nilai 0 untuk membatalkan berbagi kunci.

Pada contoh berikut, HSM yang menerima perintah, catatan CM_SHARE_OBJECT acara yang mewakili operasi berbagi.

```

Time: 02/08/19 19:35:39.480168, usecs:1549654539480168
Sequence No : 0x3f
Reboot counter : 0x38
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_SHARE_OBJECT (0x12)
Session Handle : 0x3014007
Response : 0:HSM Return: SUCCESS
Log type : UNKNOWN_LOG_TYPE (5)

```

AWS CloudHSM referensi log audit

AWS CloudHSM mencatat perintah manajemen HSM dalam peristiwa log audit. Setiap peristiwa memiliki kode nilai operasi (Opcode) yang mengidentifikasi tindakan yang terjadi dan responsnya. Anda dapat menggunakan nilai Opcode untuk menelusuri, menyortir, dan memfilter log.

Tabel berikut mendefinisikan Opcode nilai-nilai dalam log AWS CloudHSM audit.

Kode Operasi (Opcode)	Deskripsi
User Login:	Peristiwa ini termasuk nama pengguna dan tipe pengguna
CN_LOGIN (0xd)	Login pengguna
CN_LOGOUT (0xe)	Logout pengguna

Kode Operasi (Opcode)	Deskripsi
CN_APP_FINALIZE	Koneksi dengan HSM ditutup. Kunci sesi atau token kuorum apa pun dari koneksi ini telah dihapus.
CN_CLOSE_SESSION	Sesi dengan HSM ditutup. Kunci sesi atau token kuorum apa pun dari sesi ini telah dihapus.

Manajemen Pengguna: Peristiwa ini mencakup nama pengguna dan jenis pengguna

CN_CREATE_USER (0x3)	Buat pengguna kriptografi (CU)
CN_CREATE_CO	Buat petugas kriptografi (CO)
CN_DELETE_USER	Hapus pengguna
CN_CHANGE_PSWD	Mengubah kata sandi pengguna
CN_SET_M_VALUE	Tetapkan otentikasi kuorum (M dari N) untuk tindakan pengguna
CN_APPROVE_TOKEN	Menyetujui token otentikasi kuorum untuk tindakan pengguna
CN_DELETE_TOKEN	Hapus satu atau beberapa token kuorum
CN_GET_TOKEN	Meminta token penandatanganan untuk memulai operasi kuorum

Manajemen Kunci: Peristiwa ini termasuk pegangan kunci

CN_GENERATE_KEY	Hasilkan kunci simetris
CN_GENERATE_KEY_PAIR (0x19)	Menghasilkan key pair asimetris
CN_CREATE_OBJECT	Impor kunci publik (tanpa membungkus)
CN_MODIFY_OBJECT	Tetapkan atribut kunci

Kode Operasi (Opcode)	Deskripsi
CN_DESTROY_OBJECT (0x11)	Penghapusan kunci sesi
CN_TOMBSTONE_OBJECT	Penghapusan kunci token
CN_SHARE_OBJECT	Bagikan atau batalkan pembagian kunci
CN_WRAP_KEY	Ekspor salinan kunci terenkripsi (WrapKey)
CN_UNWRAP_KEY	Impor salinan kunci terenkripsi (UnwrapKey)
CN_DERIVE_KEY	Turunkan kunci simetris dari kunci yang ada
CN_NIST_AES_WRAP	Enkripsi atau dekripsi kunci dengan kunci AES
CN_INSERT_MASKED_OBJECT_USER	Masukkan kunci terenkripsi dengan atribut dari HSM lain di cluster.
CN_EXTRACT_MASKED_OBJECT_USER	Membungkus/mengenkripsi kunci dengan atribut dari HSM untuk dikirim ke HSM lain di cluster.
Session Management	
CN_ENCRYPT_SESSION_V2 (0x107)	Menetapkan sesi end-to-end terenkripsi yang diautentikasi.
END_MARKER_OPCODE (0xffff)	Menyisipkan penanda akhir dalam buffer log audit yang menunjukkan tidak ada lagi perintah yang dapat dicatat yang diizinkan di HSM
Back up HSMs	
CN_BACKUP_BEGIN	Mulailah proses pencadangan
CN_BACKUP_END	Menyelesaikan proses backup
CN_RESTORE_BEGIN	Mulailah memulihkan dari cadangan
CN_RESTORE_END	Menyelesaikan proses restorasi dari cadangan

Kode Operasi (Opcode)	Deskripsi
Certificate-Based Authentication	
CN_CERT_AUTH_STORE_CERT	Menyimpan sertifikat cluster
HSM Instance Commands	
CN_INIT_TOKEN (0x1)	Mulai proses inisialisasi HSM
CN_INIT_DONE	Proses inisialisasi HSM telah selesai
CN_GEN_KEY_ENC_KEY	Hasilkan kunci enkripsi kunci (KEK)
CN_GEN_PSWD_ENC_KEY (0x1d)	Hasilkan kunci enkripsi kata sandi (PEK)
HSM crypto commands	
CN_FIPS_RAND	Hasilkan nomor acak yang sesuai dengan FIPS 1

[1] Hanya dicatat untuk klaster hsm1.medium.

Mendapatkan CloudWatch metrik untuk AWS CloudHSM

Gunakan CloudWatch untuk memantau AWS CloudHSM cluster Anda secara real time. Metrik dapat dikelompokkan berdasarkan wilayah, ID cluster, atau ID cluster dan ID HSM.

AWS/CloudHSMNamespace mencakup metrik berikut:

Metrik	Deskripsi
HsmUnhealthy	Instans HSM tidak berfungsi dengan baik. AWS CloudHSM secara otomatis menggantikan contoh yang tidak sehat untuk Anda. Anda dapat memilih untuk secara proaktif memperluas ukuran klaster untuk mengurangi dampak performa sementara kita mengganti HSM.
HsmTemperature 1	Suhu persimpangan prosesor perangkat keras. Sistem mati jika suhu mencapai 110 derajat Centigrade.

Metrik	Deskripsi
HsmKeysSessionOccupied	Jumlah kunci sesi yang digunakan oleh instance HSM.
HsmKeysTokenOccupied	Jumlah kunci token yang digunakan oleh instance HSM dan cluster.
HsmSslCtxsOccupied ¹	Jumlah saluran end-to-end terenkripsi yang saat ini ditetapkan untuk instance HSM. Hingga 2.048 saluran diperbolehkan.
HsmSessionCount	Jumlah koneksi terbuka ke instance HSM. Hingga 2.048 diperbolehkan. Secara default, daemon klien dikonfigurasi untuk membuka dua sesi dengan setiap instance HSM di bawah satu end-to-end saluran terenkripsi. AWS CloudHSM mungkin juga memiliki hingga 2 koneksi terbuka dengan HSM untuk memantau kesehatan. HSMs
HsmUsersAvailable	Jumlah pengguna tambahan yang dapat dibuat. Ini sama dengan jumlah maksimum pengguna (tercantum dalam HsmUsersMax) dikurangi pengguna yang dibuat hingga saat ini.
HsmUsersMax ¹	Jumlah maksimum pengguna yang dapat dibuat pada instance HSM.
InterfaceEth2OctetsInput ¹	Jumlah kumulatif lalu lintas masuk ke HSM hingga saat ini.
InterfaceEth2OctetsOutput ¹	Jumlah kumulatif lalu lintas keluar ke HSM hingga saat ini.

- [1] Metrik ini tidak tersedia untuk hsm2m.medium.

AWS CloudHSM informasi kinerja

Untuk AWS CloudHSM cluster produksi, Anda harus memiliki setidaknya dua instance modul keamanan perangkat keras (HSM) yang tersebar di berbagai zona ketersediaan di suatu wilayah. Kami merekomendasikan pengujian beban klaster Anda untuk menentukan beban puncak yang harus Anda antisipasi, dan kemudian menambahkan satu HSM lagi untuk memastikan ketersediaan tinggi. Untuk aplikasi yang membutuhkan daya tahan kunci yang baru dibuat, kami merekomendasikan setidaknya tiga instans HSM yang tersebar di berbagai zona ketersediaan di suatu wilayah.

Data kinerja

Kinerja AWS CloudHSM cluster bervariasi berdasarkan beban kerja tertentu. Untuk meningkatkan kinerja, Anda dapat menambahkan instance HSM tambahan ke cluster Anda. Kinerja dapat bervariasi berdasarkan konfigurasi, ukuran data, dan pemuatan aplikasi tambahan pada instans EC2 Anda. Kami mendorong pengujian beban aplikasi Anda untuk menentukan kebutuhan penskalaan.

Tabel berikut menunjukkan perkiraan kinerja untuk algoritma kriptografi umum yang berjalan pada instans EC2 dengan instance hsm1.medium.

Data kinerja untuk hsm1.medium

Operasi	Cluster dua HSM ¹	Kluster tiga HSM ²	Cluster enam-HSM ³
Tanda RSA 2048-bit	2.000 ops/detik	3.000 ops/detik	5.000 ops/detik
Tanda EC P256	500 ops/detik	750 ops/detik	1.500 ops/detik

Tabel berikut menunjukkan perkiraan kinerja untuk algoritma kriptografi umum yang berjalan pada instance EC2 dengan hsm2m.medium.

Data kinerja untuk hsm2m.medium

Operasi	Cluster dua HSM ¹	Kluster tiga HSM ²	Cluster enam-HSM ³
Tanda RSA 2048-bit	2000 ops/detik	3000 ops/detik	5000 ops/detik
Tanda EC P256	3000 ops/detik	4500 ops/detik	7000 ops/detik

- [1] Cluster dua-HSM dengan aplikasi multi-threaded Java yang berjalan pada satu [c4.large EC2 instance dengan satu HSM di AZ yang sama dengan instans EC2](#).
- [2] Sebuah cluster tiga-HSM dengan aplikasi multi-threaded Java berjalan pada satu [c4.large EC2 instance dengan satu HSM di AZ yang sama dengan instans EC2](#).
- [3] Cluster enam HSM dengan aplikasi multi-threaded Java yang berjalan pada satu [c4.large EC2 instance dengan dua HSM di AZ yang sama dengan instans EC2](#).

Pelambatan HSM

Ketika beban kerja Anda melebihi kapasitas HSM klaster Anda, Anda akan menerima pesan kesalahan yang menyatakan sibuk atau HSMs terhambat. Untuk detail tentang apa yang harus dilakukan ketika ini terjadi, lihat [Pelambatan HSM](#)

Keamanan di AWS CloudHSM

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. [Model tanggung jawab bersama](#) menjelaskan hal ini sebagai keamanan dari cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari [Program AWS Kepatuhan Program AWS Kepatuhan](#) . Untuk mempelajari tentang program kepatuhan yang berlaku AWS CloudHSM, lihat [AWS Layanan dalam Lingkup oleh AWS Layanan Program Kepatuhan](#) .
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan AWS CloudHSM. Topik berikut menunjukkan cara mengonfigurasi AWS CloudHSM untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan layanan AWS lain yang membantu Anda memantau dan mengamankan AWS CloudHSM sumber daya Anda.

Daftar Isi

- [Kontrol akses API dengan kebijakan IAM](#)
- [Perlindungan data di AWS CloudHSM](#)
- [Identitas dan manajemen akses untuk AWS CloudHSM](#)
- [Validasi kepatuhan untuk AWS CloudHSM](#)
- [Ketahanan di AWS CloudHSM](#)
- [Keamanan infrastruktur di AWS CloudHSM](#)
- [AWS CloudHSM dan titik akhir VPC](#)
- [Perbarui manajemen di AWS CloudHSM](#)

Kontrol akses API dengan kebijakan IAM

Tingkatkan kebijakan IAM ke IPv6

AWS CloudHSM pelanggan menggunakan kebijakan IAM untuk mengontrol akses ke AWS CloudHSM APIs dan mencegah alamat IP apa pun di luar rentang yang dikonfigurasi agar tidak dapat mengakses AWS CloudHSM APIs.

Cloudhsmv2. *<region>*.api.aws dual-stack endpoint di mana AWS CloudHSM APIs dukungan yang di-host selain. IPv6 IPv4

Pelanggan yang perlu mendukung keduanya IPv4 dan IPv6 harus memperbarui kebijakan penyaringan alamat IP mereka untuk menangani IPv6 alamat, jika tidak maka akan memengaruhi kemampuan mereka untuk terhubung ke AWS CloudHSM over IPv6.

Siapa yang harus meng-upgrade?

Pelanggan yang menggunakan pengalamatan ganda dengan kebijakan yang mengandung AWS: SourceIP terpengaruh oleh peningkatan ini. Pengalamatan ganda berarti bahwa jaringan mendukung keduanya IPv4 dan IPv6.

Jika Anda menggunakan pengalamatan ganda, Anda harus memperbarui kebijakan IAM Anda yang saat ini dikonfigurasi dengan alamat IPv4 format untuk menyertakan alamat IPv6 format.

Untuk bantuan terkait masalah akses, hubungi [Dukungan](#).

Note

Pelanggan berikut tidak terpengaruh oleh peningkatan ini:

- Pelanggan yang hanya berada di IPv4 jaringan.

Apa itu IPv6?

IPv6 adalah standar IP generasi berikutnya yang dimaksudkan untuk akhirnya menggantikan IPv4. Versi sebelumnya, IPv4, menggunakan skema pengalamatan 32-bit untuk mendukung 4,3 miliar perangkat. IPv6 Sebaliknya menggunakan pengalamatan 128-bit untuk mendukung sekitar 340 triliun triliun (atau 2 hingga daya 128) perangkat.

Untuk detail selengkapnya, lihat halaman [web VPC IPv6](#).

```
2001:cdba:0000:0000:0000:0000:3257:9652
2001:cdba:0:0:0:0:3257:9652
2001:cdba::3257:965
```

Memperbarui kebijakan IAM untuk IPv6

Kebijakan IAM saat ini digunakan untuk mengatur rentang alamat IP yang diizinkan menggunakan `aws:SourceIp` filter.

Pengalamatan ganda mendukung keduanya IPv4 dan IPv6 lalu lintas. Jika jaringan Anda menggunakan pengalamatan ganda, Anda harus memperbarui kebijakan IAM yang digunakan untuk pemfilteran alamat IP untuk menyertakan rentang alamat IPv6

Misalnya, kebijakan di bawah ini mengidentifikasi rentang IPv4 alamat yang diizinkan `192.0.2.0.*` dan `203.0.113.0.*` dalam `Condition` elemen.

```
# https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_policies_examples_aws_deny-ip.html
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
      "NotIpAddress": {
        "aws:SourceIp": [
          "192.0.2.0/24",
          "203.0.113.0/24"
        ]
      },
      "Bool": {
        "aws:ViaAWSService": "false"
      }
    }
  }
}
```

Untuk memperbarui kebijakan ini, ubah `Condition` elemen untuk menyertakan rentang IPv6 alamat `2001:DB8:1234:5678::/64` dan `2001:cdba:3257:8593::/64`.

Note

JANGAN HAPUS IPv4 alamat yang ada karena diperlukan untuk kompatibilitas mundur.

```
"Condition": {
  "NotIpAddress": {
    "*aws:SourceIp*": [
      "*192.0.2.0/24*", <<DO NOT REMOVE existing IPv4 address>>
      "*203.0.113.0/24*", <<DO NOT REMOVE existing IPv4 address>>
      "*2001:DB8:1234:5678::/64*", <<New IPv6 IP address>>
      "*2001:cdba:3257:8593::/64*" <<New IPv6 IP address>>
    ]
  },
  "Bool": {
    "aws:ViaAWSService": "false"
  }
}
```

Verifikasi dukungan klien Anda IPv6

Pelanggan menggunakan cloudhsmv2. Titik akhir {region} .api.aws disarankan untuk memverifikasi apakah mereka dapat terhubung dengannya. Langkah-langkah berikut menjelaskan cara melakukan verifikasi.

Contoh ini menggunakan Linux dan curl versi 8.6.0 dan menggunakan [titik akhir AWS CloudHSM layanan yang telah IPv6 mengaktifkan titik akhir](#) yang terletak di titik akhir api.aws.

Note

Beralih AWS Region ke Wilayah yang sama di mana klien berada. Dalam contoh ini, kita menggunakan US East (N. Virginia) — us-east-1 endpoint.

1. Tentukan apakah titik akhir diselesaikan dengan IPv6 alamat menggunakan perintah berikut.
dig

```
dig +short AAAA cloudhsmv2.us-east-1.api.aws
2600:1f18:e2f:4e05:1a8a:948e:7c08:c1c3
```


2. Tentukan apakah jaringan klien dapat membuat IPv6 koneksi menggunakan `curl` perintah berikut. Kode respons 404 berarti koneksi berhasil, sedangkan kode respons 0 berarti koneksi gagal.

```
curl --ipv6 -o /dev/null --silent -w "\nremote ip: %{remote_ip}\nresponse code:
%{response_code}\n" https://cloudhsmv2.us-east-1.api.aws

remote ip: 2600:1f18:e2f:4e05:1a8a:948e:7c08:c1c3
response code: 404
```

Jika IP jarak jauh diidentifikasi dan kode respons tidak 0, koneksi jaringan berhasil dibuat ke titik akhir menggunakan IPv6. IP jarak jauh harus menjadi IPv6 alamat karena sistem operasi harus memilih protokol yang valid untuk klien. Jika IP jarak jauh bukan IPv6 alamat, gunakan perintah berikut `curl` untuk memaksa penggunaan IPv4.

```
curl --ipv4 -o /dev/null --silent -w "\nremote ip: %{remote_ip}\nresponse code:
%{response_code}\n" https://cloudhsmv2.us-east-1.api.aws

remote ip: 3.123.154.250
response code: 404
```

Jika IP jarak jauh kosong atau kode responsnya 0, jaringan klien atau jalur jaringan ke titik akhir adalah IPv4 -only. Anda dapat memverifikasi konfigurasi ini dengan `curl` perintah berikut.

```
curl -o /dev/null --silent -w "\nremote ip: %{remote_ip}\nresponse code:
%{response_code}\n" https://cloudhsmv2.us-east-1.api.aws

remote ip: 3.123.154.250
response code: 404
```

Perlindungan data di AWS CloudHSM

[Model tanggung jawab AWS bersama model](#) berlaku untuk perlindungan data di AWS CloudHSM. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam [Pertanyaan Umum Privasi Data](#). Lihat

informasi tentang perlindungan data di Eropa di pos blog [Model Tanggung Jawab Bersama dan GDPR AWS](#) di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensial dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan AWS sumber daya. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat [Bekerja dengan CloudTrail jejak](#) di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-3 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi selengkapnya tentang titik akhir FIPS yang tersedia di [Standar Pemrosesan Informasi Federal \(FIPS\) 140-3](#).

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan AWS CloudHSM atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau AWS SDKs. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Enkripsi saat diam

Saat AWS CloudHSM membuat cadangan dari HSM, HSM mengenkripsi datanya sebelum mengirimnya ke. AWS CloudHSM Data dienkripsi menggunakan kunci enkripsi fana yang unik, kunci enkripsi sementara. Untuk informasi selengkapnya, lihat [AWS CloudHSM cadangan klaster](#).

Enkripsi saat bergerak

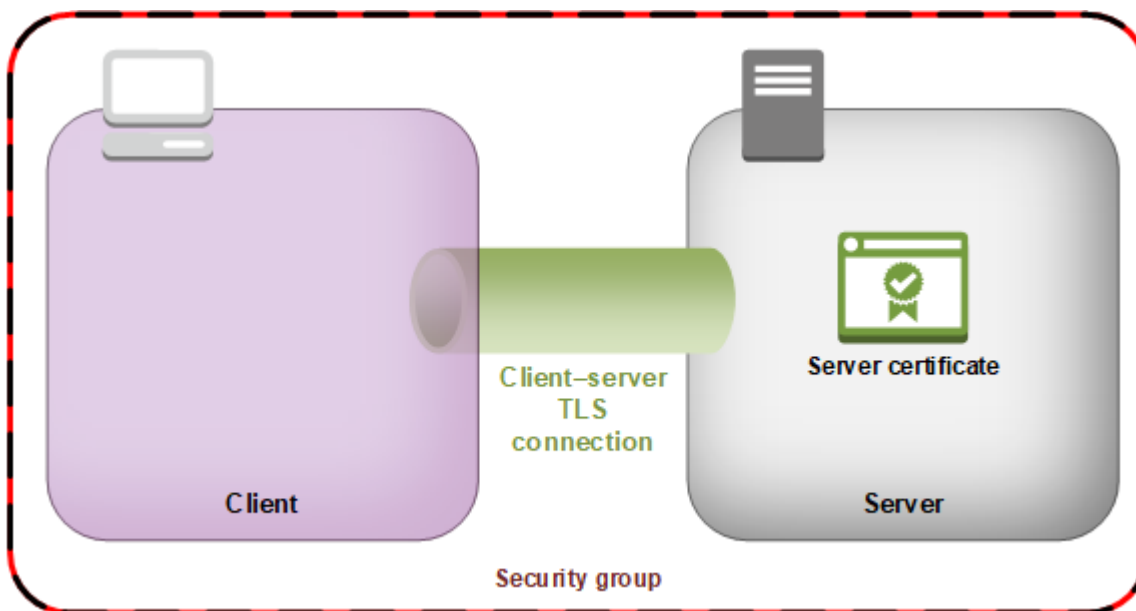
Komunikasi antara AWS CloudHSM klien dan HSM di cluster Anda dienkripsi dari ujung ke ujung. Komunikasi ini dapat didekripsi hanya oleh klien Anda dan HSM Anda. Untuk informasi selengkapnya, lihat [End-to-end enkripsi](#).

AWS CloudHSM end-to-endenkripsi klien

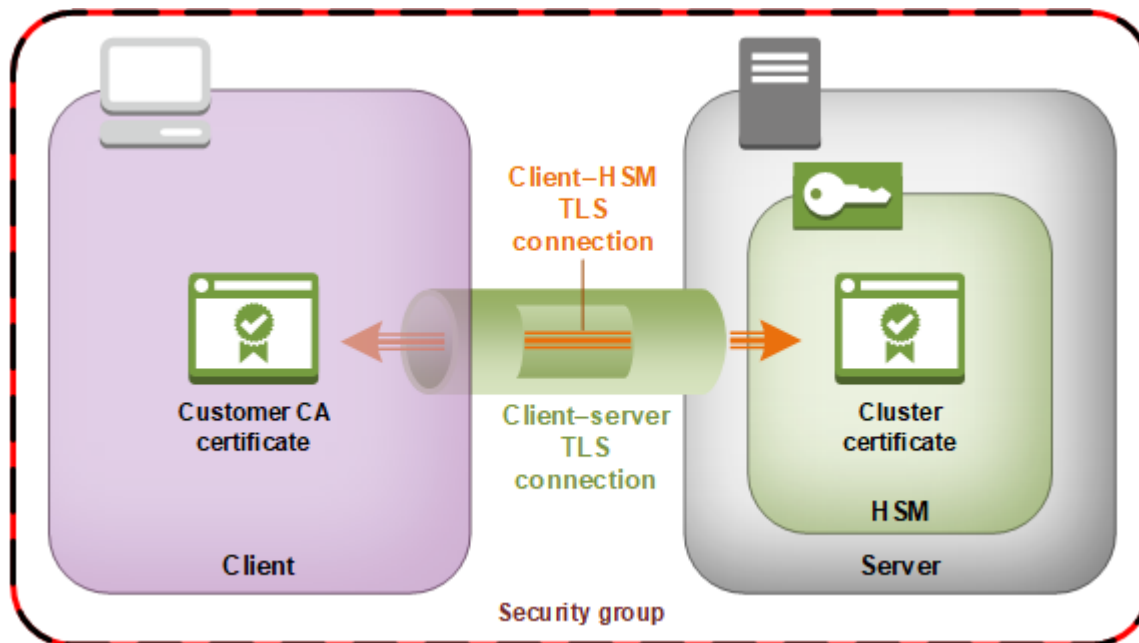
Komunikasi antara instance klien dan HSMs di cluster Anda dienkripsi dari ujung ke ujung. Hanya klien Anda dan Anda yang HSMs dapat mendekripsi komunikasi.

Proses berikut menjelaskan bagaimana klien membangun komunikasi end-to-end terenkripsi dengan HSM.

1. Klien Anda menetapkan sambungan Keamanan Lapisan Pengangkutan (TLS) dengan server yang meng-host perangkat keras HSM Anda. Grup keamanan kluster Anda mengizinkan lalu lintas masuk ke server hanya dari instans klien dalam grup keamanan. Klien juga memeriksa sertifikat server untuk memastikan bahwa itu adalah server tepercaya.



2. Selanjutnya, klien membuat koneksi terenkripsi dengan perangkat keras HSM. HSM memiliki sertifikat kluster yang Anda tandatangi dengan sertifikat otoritas (CA) Anda sendiri, dan klien memiliki sertifikat root CA. Sebelum sambungan terenkripsi klien-HSM didirikan, klien memverifikasi sertifikat kluster HSM terhadap sertifikat root. Sambungan dibuat hanya ketika klien berhasil memverifikasi bahwa HSM tepercaya.



Keamanan cadangan klaster

Ketika AWS CloudHSM membuat cadangan dari HSM, HSM mengenkripsi semua datanya sebelum mengirimnya ke. AWS CloudHSM Data tidak pernah meninggalkan HSM dalam bentuk plaintext. Selain itu, cadangan tidak dapat didekripsi oleh AWS karena AWS tidak memiliki akses ke kunci yang digunakan untuk mendekripsi cadangan.

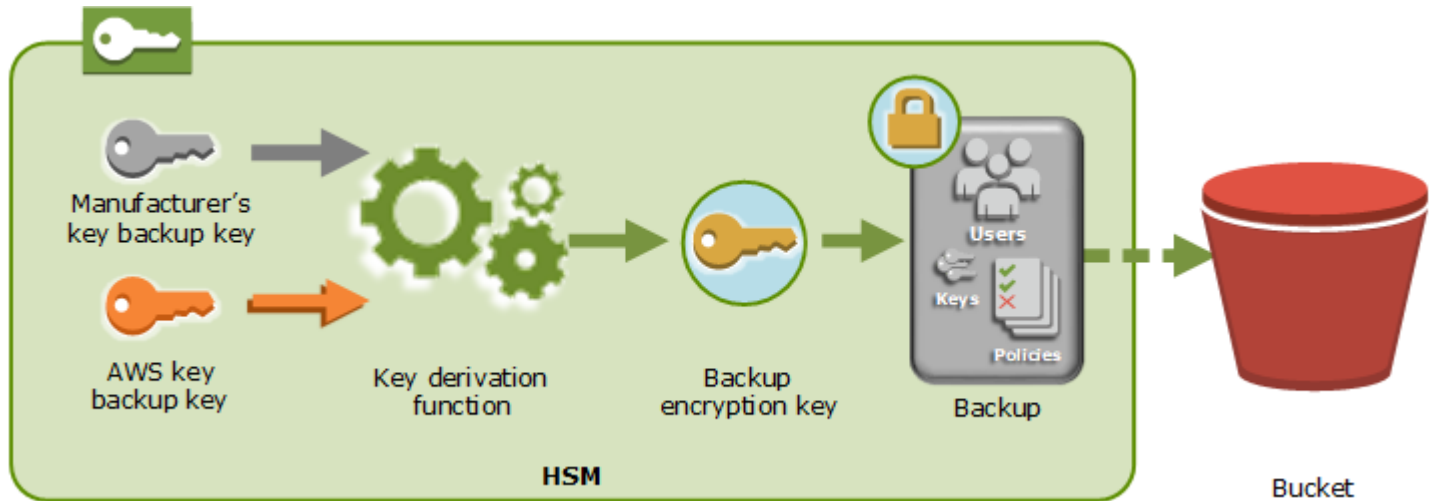
Untuk mengenkripsi data, HSM menggunakan kunci enkripsi unik, sementara yang dikenal sebagai kunci cadangan sementara (EBK). EBK adalah kunci enkripsi AES 256-bit yang dihasilkan di dalam HSM saat membuat cadangan. AWS CloudHSM HSM menghasilkan EBK, kemudian menggunakannya untuk mengenkripsi data HSM dengan metode pembungkus kunci AES yang disetujui FIPS yang sesuai dengan [publikasi khusus NIST 800-38F](#). Kemudian HSM memberikan data terenkripsi ke. AWS CloudHSM Data yang dienkripsi mencakup salinan EBK yang dienkripsi.

Untuk mengenkripsi EBK, HSM menggunakan kunci enkripsi lain yang dikenal sebagai kunci cadangan persisten (PBK). PBK juga merupakan kunci enkripsi AES 256-bit. Untuk menghasilkan PBK, HSM menggunakan fungsi derivasi kunci (KDF) yang disetujui FIPS dalam modus penghitung yang sesuai dengan [publikasi khusus NIST 800-108](#). Input ke KDF ini meliputi:

- Kunci cadangan kunci produsen (MKBK), tertanam secara permanen dalam perangkat keras HSM oleh produsen perangkat keras.

- AWS Kunci cadangan kunci (AKBK), dipasang dengan aman di HSM saat awalnya dikonfigurasi oleh. AWS CloudHSM

Proses enkripsi dirangkum dalam gambar berikut. Kunci enkripsi cadangan mewakili kunci cadangan persisten (PBK) dan kunci cadangan sementara (EBK).



AWS CloudHSM dapat mengembalikan cadangan ke hanya AWS milik yang HSMs dibuat oleh produsen yang sama. Karena setiap cadangan berisi semua pengguna, kunci, dan konfigurasi dari HSM asli, HSM yang dipulihkan berisi perlindungan dan kontrol akses yang sama seperti aslinya. Data yang dipulihkan akan menimpa semua data lain yang mungkin ada di HSM sebelum pemulihan.

Cadangan hanya terdiri dari data terenripsi. Sebelum layanan menyimpan cadangan di Amazon S3, layanan mengenkripsi cadangan lagi menggunakan (). AWS Key Management Service AWS KMS

Identitas dan manajemen akses untuk AWS CloudHSM

AWS menggunakan kredensial keamanan untuk mengidentifikasi Anda dan memberikan Anda akses ke sumber daya AWS Anda. Anda dapat menggunakan fitur AWS Identity and Access Management (IAM) untuk memungkinkan pengguna, layanan, dan aplikasi lain menggunakan sumber daya AWS Anda sepenuhnya atau dengan cara yang terbatas. Anda dapat melakukan ini tanpa perlu berbagi kredensial keamanan Anda.

Secara default, pengguna IAM tidak memiliki izin untuk membuat, melihat, atau memodifikasi sumber daya AWS. Untuk mengizinkan pengguna IAM mengakses sumber daya seperti penyeimbang beban, dan untuk melakukan tugas, Anda:

1. Membuat kebijakan IAM yang memberikan izin pengguna IAM untuk menggunakan sumber daya tertentu dan tindakan API yang mereka butuhkan.
2. Melampirkan kebijakan ke pengguna IAM atau grup yang dimiliki oleh pengguna IAM.

Saat Anda melampirkan kebijakan ke pengguna atau grup pengguna, kebijakan itu mengizinkan atau menolak izin pengguna untuk melakukan tugas yang ditentukan pada sumber daya yang ditentukan.

Misalnya, Anda dapat menggunakan IAM untuk membuat pengguna dan grup di bawah akun AWS Anda. Pengguna IAM dapat berupa orang, sistem, atau aplikasi. Kemudian, Anda memberikan izin kepada pengguna dan grup untuk melakukan tindakan tertentu pada sumber daya tertentu menggunakan kebijakan IAM.

Memberikan izin menggunakan kebijakan IAM

Saat Anda melampirkan kebijakan ke pengguna atau grup pengguna, kebijakan itu mengizinkan atau menolak izin pengguna untuk melakukan tugas yang ditentukan pada sumber daya yang ditentukan.

Kebijakan IAM adalah sebuah dokumen JSON yang terdiri dari satu pernyataan atau lebih.

Contoh kebijakan IAM berikut memungkinkan pengguna untuk menggambarkan AWS CloudHSM cadangan di AS Timur (Virginia N.). Hanya menjelaskan permintaan yang berasal dari US East (Virginia N.) yang diizinkan.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cloudhsm:DescribeBackups",
      "Resource": "arn:aws:cloudhsm:us-east-1:123456789012:backup/*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": "us-east-1"
        }
      }
    }
  ]
}
```

```
}
```

- **Efek**— Efek dapat berupa Allow atau Deny. Secara default, pengguna IAM tidak memiliki izin untuk menggunakan sumber daya dan tindakan API, sehingga semua permintaan akan ditolak. izin eksplisit akan menggantikan izin default. Penolakan eksplisit akan mengabaikan izin apa pun.
- **Tindakan**— Tindakan adalah tindakan API tertentu yang Anda berikan atau tolak izinnya. Untuk informasi lebih lanjut tentang menentukan tindakan, lihat [Tindakan API untuk AWS CloudHSM](#).
- **Sumber daya** — Sumber daya yang dipengaruhi oleh tindakan. AWS CloudHSM tidak mendukung izin tingkat sumber daya. Anda harus menggunakan wildcard * untuk menentukan semua AWS CloudHSM sumber daya.
- **Syarat**— Anda dapat secara opsional menggunakan syarat untuk mengontrol kapan kebijakan Anda berlaku. Untuk informasi lebih lanjut, lihat [Kunci kondisi untuk AWS CloudHSM](#).

Untuk informasi lebih lanjut, lihat [Panduan Pengguna IAM](#).

Tindakan API untuk AWS CloudHSM

Dalam elemen Tindakan pernyataan kebijakan IAM Anda, Anda dapat menentukan tindakan API apa pun yang AWS CloudHSM ditawarkan. Anda harus memberi prefiks pada nama tindakan dengan string huruf kecil `cloudhsm:`, seperti yang ditunjukkan dalam contoh berikut.

```
"Action": "cloudhsm:DescribeClusters"
```

Untuk menetapkan beberapa tindakan dalam satu pernyataan, batasi dengan kurung perseggi dan pisahkan dengan koma, seperti ditunjukkan dalam contoh berikut.

```
"Action": [  
    "cloudhsm:DescribeClusters",  
    "cloudhsm:DescribeHsm"  
]
```

Anda juga dapat menentukan beberapa tindakan menggunakan wildcard *. Contoh berikut menentukan semua nama tindakan API untuk memulai dengan AWS CloudHSM List itu.

```
"Action": "cloudhsm:List*"
```

Untuk menentukan semua tindakan API AWS CloudHSM, gunakan wildcard *, seperti yang ditunjukkan pada contoh berikut.

```
"Action": "cloudhsm:*"
```

Untuk daftar tindakan API AWS CloudHSM, lihat [AWS CloudHSM Tindakan](#).

Kunci kondisi untuk AWS CloudHSM

Saat membuat kebijakan, Anda dapat menentukan syarat yang mengontrol kapan kebijakan berlaku. Setiap syarat mengandung satu atau lebih pasangan nilai-kunci. Terdapat kunci syarat global dan kunci syarat khusus layanan.

AWS CloudHSM tidak memiliki kunci konteks khusus layanan.

Untuk informasi lebih lanjut tentang kunci syarat global, lihat [Kunci Konteks Syarat Global AWS](#) dalam Panduan Pengguna IAM.

Kebijakan terkelola AWS yang telah ditentukan sebelumnya untuk AWS CloudHSM

Kebijakan terkelola yang dibuat oleh AWS memberikan izin yang diperlukan untuk kasus penggunaan umum. Anda dapat melampirkan kebijakan ini ke pengguna IAM Anda, berdasarkan akses ke AWS CloudHSM yang mereka butuhkan:

- **AWSCloudHSMFullAkses** — Memberikan akses penuh yang diperlukan untuk menggunakan AWS CloudHSM fitur.
- **AWSCloudHSMReadOnlyAccess**— Memberikan akses hanya-baca ke fitur. AWS CloudHSM

Kebijakan yang dikelola pelanggan untuk AWS CloudHSM

Kami menyarankan Anda membuat grup administrator IAM yang hanya berisi izin AWS CloudHSM yang diperlukan untuk menjalankan. AWS CloudHSM Lampirkan kebijakan dengan izin yang sesuai untuk grup ini. Tambahkan pengguna IAM ke grup sesuai kebutuhan. Setiap pengguna yang Anda tambahkan mewarisi kebijakan dari grup administrator.

Selain itu, sebaiknya Anda membuat grup pengguna tambahan berdasarkan izin yang dibutuhkan pengguna. Hal ini memastikan bahwa hanya pengguna tepercaya yang memiliki akses ke

tindakan API kritis. Misalnya, Anda dapat membuat grup pengguna yang Anda gunakan untuk memberikan akses hanya-baca ke kluster dan. HSMs Karena grup ini tidak mengizinkan pengguna untuk menghapus cluster atau HSMs, pengguna yang tidak dipercaya tidak dapat memengaruhi ketersediaan beban kerja produksi.

Karena fitur AWS CloudHSM manajemen baru ditambahkan dari waktu ke waktu, Anda dapat memastikan bahwa hanya pengguna tepercaya yang diberikan akses langsung. Dengan menetapkan izin terbatas untuk kebijakan saat pembuatan, Anda dapat secara manual menetapkan izin fitur baru untuk mereka nanti.

Berikut ini adalah contoh kebijakan untuk AWS CloudHSM. Untuk informasi tentang cara membuat kebijakan dan melampirkan ke grup pengguna IAM, lihat [Membuat Kebijakan di Tab JSON](#) dalam Panduan Pengguna IAM.

Contoh

- [Hanya Baca Izin](#)
- [Izin Pengguna Daya](#)
- [Izin Admin](#)

Example Contoh: Izin baca-saja

Kebijakan ini mengizinkan akses ke tindakan API DescribeClusters dan DescribeBackups. Ini juga mencakup izin tambahan untuk tindakan Amazon EC2 API tertentu. Itu tidak memungkinkan pengguna untuk menghapus cluster atau HSMs.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "cloudhsm:DescribeClusters",
      "cloudhsm:DescribeBackups",
      "cloudhsm:ListTags"
    ],
    "Resource": "*"
  }
}
```

```
}
```

Example Contoh: Izin pengguna daya

Kebijakan ini memungkinkan akses ke subset tindakan AWS CloudHSM API. Ini juga mencakup izin tambahan untuk tindakan Amazon EC2 tertentu. Itu tidak memungkinkan pengguna untuk menghapus cluster atau HSMs. Anda harus menyertakan `iam:CreateServiceLinkedRole` tindakan AWS CloudHSM untuk memungkinkan secara otomatis membuat peran terkait layanan `AWSServiceRoleForCloudHSM` di akun Anda. Peran ini memungkinkan AWS CloudHSM untuk mencatat peristiwa. Untuk informasi selengkapnya, lihat [Peran terkait layanan untuk AWS CloudHSM](#).

Note

Untuk melihat izin khusus untuk setiap API, lihat [kunci Tindakan, sumber daya, dan kondisi AWS CloudHSM di Referensi Otorisasi Layanan](#).

JSON

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "cloudhsm:DescribeClusters",
      "cloudhsm:DescribeBackups",
      "cloudhsm:CreateCluster",
      "cloudhsm:CreateHsm",
      "cloudhsm:RestoreBackup",
      "cloudhsm:CopyBackupToRegion",
      "cloudhsm:InitializeCluster",
      "cloudhsm:ListTags",
      "cloudhsm:TagResource",
      "cloudhsm:UntagResource",
      "ec2:CreateNetworkInterface",
      "ec2:DescribeNetworkInterfaces",
      "ec2:DescribeNetworkInterfaceAttribute",
      "ec2:DetachNetworkInterface",
      "ec2>DeleteNetworkInterface",
    ]
  }
}
```

```

        "ec2:CreateSecurityGroup",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:DescribeSecurityGroups",
        "ec2>DeleteSecurityGroup",
        "ec2:CreateTags",
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "iam:CreateServiceLinkedRole"
    ],
    "Resource": "*"
}
}

```

Example Contoh: Izin admin

Kebijakan ini memungkinkan akses ke semua tindakan AWS CloudHSM API, termasuk tindakan untuk menghapus HSM dan cluster. Ini juga mencakup izin tambahan untuk tindakan Amazon EC2 tertentu. Anda harus menyertakan `iam:CreateServiceLinkedRole` tindakan AWS CloudHSM untuk memungkinkan secara otomatis membuat peran terkait layanan `AWSServiceRoleForCloudHSM` di akun Anda. Peran ini memungkinkan AWS CloudHSM untuk mencatat peristiwa. Untuk informasi selengkapnya, lihat [Peran terkait layanan untuk AWS CloudHSM](#).

JSON

```

{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "cloudhsm:*",
      "ec2:CreateNetworkInterface",
      "ec2:DescribeNetworkInterfaces",
      "ec2:DescribeNetworkInterfaceAttribute",
      "ec2:DetachNetworkInterface",
      "ec2>DeleteNetworkInterface",
      "ec2:CreateSecurityGroup",
      "ec2:AuthorizeSecurityGroupIngress",
      "ec2:AuthorizeSecurityGroupEgress",

```

```

        "ec2:RevokeSecurityGroupEgress",
        "ec2:DescribeSecurityGroups",
        "ec2>DeleteSecurityGroup",
        "ec2:CreateTags",
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "iam:CreateServiceLinkedRole"
    ],
    "Resource": "*"
}
}

```

Peran terkait layanan untuk AWS CloudHSM

Kebijakan IAM yang Anda buat sebelumnya untuk [Kebijakan yang dikelola pelanggan untuk AWS CloudHSM](#) menyertakan `iam:CreateServiceLinkedRole` tindakan. AWS CloudHSM mendefinisikan [peran terkait layanan bernama](#) HSM. AWSService RoleForCloud Peran tersebut telah ditentukan sebelumnya oleh AWS CloudHSM dan termasuk izin yang AWS CloudHSM mengharuskan untuk memanggil AWS layanan lain atas nama Anda. Peran tersebut memudahkan pengaturan layanan karena Anda tidak perlu menambahkan kebijakan peran dan izin kebijakan kepercayaan secara manual.

Kebijakan peran memungkinkan Anda AWS CloudHSM membuat grup CloudWatch log Amazon Log dan aliran log serta menulis peristiwa log atas nama Anda. Anda dapat melihatnya di bawah dan di konsol IAM.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ]
    }
  ]
}

```

```

        "Resource": [
            "arn:aws:logs:*:*:*"
        ]
    }
]
}

```

Kebijakan kepercayaan untuk peran `AWSServiceRoleForCloudHSM` memungkinkan AWS CloudHSM untuk mengambil peran tersebut.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudhsm.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

Membuat peran terkait layanan (otomatis)

AWS CloudHSM membuat peran `AWSServiceRoleForCloudHSM` saat Anda membuat kluster jika Anda menyertakan `iam:CreateServiceLinkedRole` tindakan dalam izin yang Anda tentukan saat membuat grup AWS CloudHSM administrator. Lihat [Kebijakan yang dikelola pelanggan untuk AWS CloudHSM](#).

Jika Anda sudah memiliki satu atau beberapa cluster dan hanya ingin menambahkan peran `AWSServiceRoleForCloudHSM`, Anda dapat menggunakan konsol, perintah [create-cluster](#), atau operasi [CreateCluster](#) API untuk membuat cluster. Kemudian gunakan konsol, perintah [delete-cluster](#), atau operasi [DeleteCluster](#) API untuk menghapusnya. Membuat kluster baru menciptakan peran terkait layanan dan berlaku untuk semua kluster di akun Anda. Atau, Anda dapat membuat peran secara manual. Lihat bagian berikut untuk informasi selengkapnya.

 Note

Anda tidak perlu melakukan semua langkah yang diuraikan [Memulai dengan AWS CloudHSM](#) untuk membuat cluster jika Anda hanya membuatnya untuk menambahkan peran AWSServiceRoleForCloudHSM.

Membuat peran terkait layanan (manual)

Anda dapat menggunakan konsol IAM, AWS CLI, atau API untuk membuat peran AWSServiceRoleForCloudHSM. Untuk informasi lebih lanjut, lihat [Membuat Peran yang Terhubung dengan Layanan](#) di Panduan Pengguna IAM.

Mengedit peran terkait layanan

AWS CloudHSM tidak memungkinkan Anda untuk mengedit peran AWSServiceRoleForCloudHSM. Setelah membuat peran, misalnya, Anda tidak dapat mengubah namanya karena berbagai entitas mungkin mereferensikan peran tersebut. Selain itu, Anda tidak dapat mengubah kebijakan peran. Namun, Anda dapat menggunakan IAM untuk mengedit deskripsi peran. Untuk informasi lebih lanjut, lihat [Mengedit Peran Terkait Layanan](#) di Panduan Pengguna IAM.

Menghapus peran tertaut layanan

Anda tidak dapat menghapus peran terkait layanan selama kluster yang telah diterapkan masih ada. Untuk menghapus peran, Anda harus terlebih dahulu menghapus setiap HSM di kluster Anda dan kemudian menghapus kluster. Setiap kluster di akun Anda harus dihapus. Anda kemudian dapat menggunakan konsol IAM, AWS CLI, atau API untuk menghapus peran. Untuk informasi lebih lanjut tentang penghapusan kluster, lihat [Menghapus sebuah cluster AWS CloudHSM](#). Untuk informasi lebih lanjut, lihat [Menghapus Peran Tertaut Layanan](#) dalam Panduan Pengguna IAM.

Validasi kepatuhan untuk AWS CloudHSM

Untuk cluster dalam mode FIPS, AWS CloudHSM menyediakan FIPS yang disetujui yang memenuhi persyaratan PCI-PIN HSMs, PCI-3DS, dan kepatuhan. SOC2 AWS CloudHSM juga memberi pelanggan pilihan untuk memilih cluster yang mode non-FIPS. Untuk detail tentang persyaratan sertifikasi dan kepatuhan yang berlaku untuk masing-masing, lihat [AWS CloudHSM mode cluster](#).

Mengandalkan HSM yang divalidasi FIPS dapat membantu Anda memenuhi persyaratan kepatuhan perusahaan, kontrak, dan peraturan untuk keamanan data di Cloud. AWS

Kepatuhan FIPS 140-2

Publikasi Standar Pengolahan Informasi Federal (FIPS) 140-2 adalah standar keamanan pemerintah AS yang menentukan persyaratan keamanan untuk modul kriptografi yang melindungi informasi sensitif. [Jenis instans AWS CloudHSM hsm1.medium adalah FIPS 140-2 level 3 bersertifikat \(Sertifikat #4218\)](#). Pada 4 Januari 2026 sertifikat untuk hsm1.medium pindah ke daftar historis. [Kami menyarankan pelanggan bermigrasi ke hsm2m.medium, yang bersertifikat FIPS 140-3 \(Sertifikat #4703\)](#). Untuk informasi lebih lanjut, lihat [validasi FIPS untuk](#) perangkat keras.

Kepatuhan FIPS 140-3

Federal Information Processing Standard (FIPS) Publication 140-3 adalah standar keamanan pemerintah AS yang menetapkan persyaratan keamanan untuk modul kriptografi yang melindungi informasi sensitif. [Jenis hsm2m.medium yang HSMs disediakan oleh FIPS 140-3 level 3 AWS CloudHSM bersertifikat \(Sertifikat #4703\)](#). Untuk informasi lebih lanjut, lihat [validasi FIPS untuk](#) perangkat keras.

[Kepatuhan PCI DSS](#)

Standar Keamanan Data Industri Kartu Pembayaran (PCI DSS) adalah standar keamanan informasi eksklusif yang dikelola oleh [Dewan Standar Keamanan PCI](#). Yang HSMs disediakan oleh AWS CloudHSM mematuhi PCI DSS.

[Kepatuhan PIN PCI](#)

PIN PCI menyediakan persyaratan keamanan dan standar penilaian untuk mentransmisikan, memproses, dan mengelola data nomor identifikasi pribadi (PIN), informasi yang digunakan untuk transaksi di ATMs dan terminal point-of-sale (POS). Hsm1.medium dan hsm2m.medium HSMs yang disediakan oleh keduanya sesuai dengan PIN PCI. AWS CloudHSM Untuk informasi selengkapnya, lihat artikel [AWS CloudHSM sekarang disertifikasi PIN PCI](#).

Kepatuhan PCI-3DS

PCI 3DS (atau Three Domain Secure, 3-D Secure) menyediakan keamanan data untuk pembayaran e-commerce aman EMV 3D. PCI 3DS menyediakan lapisan keamanan lain untuk belanja online. Hsm1.medium dan hsm2m.medium yang disediakan oleh keduanya sesuai dengan HSMs PCI-3DS. AWS CloudHSM

SOC2

SOC2 adalah kerangka kerja untuk membantu organisasi layanan mendemonstrasikan kontrol keamanan cloud dan pusat data mereka. AWS CloudHSM telah SOC2 menerapkan kontrol di

area kritis untuk mematuhi prinsip layanan tepercaya. Untuk informasi lebih lanjut, lihat [FAQs halaman AWS SOC](#).

AWS CloudHSM Kepatuhan PCI-PIN FAQs

PIN PCI menyediakan persyaratan keamanan dan standar penilaian untuk mentransmisikan, memproses, dan mengelola data nomor identifikasi pribadi (PIN), informasi yang digunakan untuk transaksi di ATMs dan terminal point-of-sale (POS).

Ringkasan Pengesahan Kepatuhan (AOC) dan Tanggung Jawab PCI-PIN tersedia bagi pelanggan melalui AWS Artifact, portal layanan mandiri untuk akses sesuai permintaan ke laporan kepatuhan AWS. Untuk informasi selengkapnya, masuk ke [AWS Artifact di AWS Management Console](#), atau pelajari selengkapnya di [Memulai AWS Artifact](#).

FAQs

T: Apa itu Ringkasan Pengesahan Kepatuhan dan Tanggung Jawab?

Pengesahan Kepatuhan (AOC) diproduksi oleh Penilai PIN Berkualitas (QPA) yang membuktikan AWS CloudHSM memenuhi kontrol yang berlaku dalam standar PCI-PIN. Matriks ringkasan tanggung jawab menjelaskan kontrol yang merupakan tanggung jawab masing-masing AWS CloudHSM dan pelanggannya.

T: Bagaimana cara mendapatkan AWS CloudHSM Pengesahan Kepatuhan?

Pengesahan Kepatuhan PCI-PIN (AOC) tersedia untuk pelanggan melalui AWS Artifact, portal layanan mandiri untuk akses sesuai permintaan ke laporan kepatuhan AWS. Untuk informasi selengkapnya, masuk ke [AWS Artifact di AWS Management Console](#), atau pelajari selengkapnya di [Memulai AWS Artifact](#).

T: Bagaimana saya bisa mempelajari kontrol PIN PCI mana yang menjadi tanggung jawab saya?

Untuk informasi selengkapnya, silakan lihat "Ringkasan Tanggung Jawab PIN AWS CloudHSM PCI" dari Paket Kepatuhan PIN AWS PCI, yang tersedia bagi pelanggan melalui AWS Artifact, portal layanan mandiri untuk akses sesuai permintaan ke laporan kepatuhan AWS. Untuk informasi selengkapnya, masuk ke [AWS Artifact di AWS Management Console](#), atau pelajari selengkapnya di [Memulai AWS Artifact](#).

T: Sebagai AWS CloudHSM pelanggan, dapatkah saya mengandalkan Pengesahan Kepatuhan PCI-PIN (AOC)?

Pelanggan harus mengelola kepatuhan PCI-PIN mereka sendiri. Anda harus melalui proses pengesahan PCI-PIN formal melalui Qualified PIN Assessor (QPA) untuk memverifikasi bahwa beban kerja pembayaran Anda memenuhi semua kontrol/persyaratan PCI-PIN. Namun, untuk kontrol yang menjadi tanggung jawab AWS, QPA Anda dapat mengandalkan AWS CloudHSM Pengesahan Kepatuhan (AOC) tanpa pengujian lebih lanjut.

T: Apakah AWS CloudHSM bertanggung jawab atas persyaratan PCI-PIN yang terkait dengan siklus hidup manajemen kunci?

AWS CloudHSM bertanggung jawab atas siklus hidup perangkat fisik. HSMs Pelanggan bertanggung jawab atas persyaratan siklus hidup manajemen utama dalam standar PCI-PIN.

T: AWS CloudHSM Kontrol mana yang sesuai dengan PCI-PIN?

AOC merangkum AWS CloudHSM kontrol yang dinilai oleh QPA. Ringkasan Tanggung Jawab PCI-PIN tersedia bagi pelanggan melalui AWS Artifact, portal layanan mandiri untuk akses sesuai permintaan ke laporan kepatuhan AWS.

Q: Apakah AWS CloudHSM mendukung fungsi pembayaran seperti terjemahan PIN dan DUKPT?

Tidak, AWS CloudHSM memberikan tujuan umum HSMs. Seiring waktu kami dapat menyediakan fungsi pembayaran. Meskipun layanan tidak melakukan fungsi pembayaran secara langsung, pengesahan kepatuhan PIN AWS CloudHSM PCI memungkinkan pelanggan untuk mencapai kepatuhan PCI mereka sendiri untuk layanan mereka yang sedang berjalan. AWS CloudHSM Jika Anda tertarik untuk menggunakan layanan AWS Payment Cryptography untuk beban kerja Anda, silakan merujuk ke blog [“Pindahkan Pemrosesan Pembayaran ke Cloud dengan Kriptografi Pembayaran AWS.”](#)

Pemberitahuan Pengakhiran

Dari waktu ke waktu, AWS CloudHSM dapat menghentikan fungsionalitas agar tetap sesuai dengan persyaratan FIPS 140, PCI-DSS, PCI-PIN, PCI-3DS, atau karena perangkat keras. SOC2 end-of-support Halaman ini mencantumkan perubahan yang saat ini berlaku.

HSM1 penghentian

Jenis instans AWS CloudHSM hsm1.medium akan mencapai akhir dukungannya pada 31 Maret 2026. Untuk memastikan layanan berkelanjutan, kami memperkenalkan perubahan berikut:

- Mulai April 2025, Anda tidak akan dapat membuat cluster hsm1.medium baru.

- Mulai Januari 2026, kami akan mulai secara otomatis memigrasikan cluster hsm1.medium yang ada ke tipe instance hsm2m.medium yang baru.

Tipe instans hsm2m.medium kompatibel dengan tipe instans Anda saat ini dan AWS CloudHSM menawarkan peningkatan kinerja. Untuk menghindari gangguan pada aplikasi Anda, Anda harus meningkatkan ke versi terbaru SDK klien. Untuk petunjuk peningkatan, lihat [???](#).

Anda memiliki dua opsi untuk migrasi:

1. Ikut serta dalam migrasi yang dikelola CloudHSM saat Anda siap. Untuk informasi selengkapnya, lihat [???](#).
2. Buat cluster hsm2m.medium baru dari cadangan cluster hsm1 Anda dan arahkan aplikasi Anda ke cluster baru. Sebaiknya gunakan strategi blue/green penyebaran untuk pendekatan ini. Untuk informasi selengkapnya, lihat [???](#).

Kepatuhan FIPS 140: Penutupan Mekanisme 2024

Institut Standar dan Teknologi Nasional (NIST) ¹menyarankan bahwa dukungan untuk enkripsi Triple DES (DESede, 3DES, DES3) dan pembungkus kunci RSA dan membuka bungkusnya dengan padding PKCS #1 v1.5 tidak diizinkan setelah 31 Desember 2023. Oleh karena itu, dukungan untuk akhir ini pada 1 Januari 2024 di klaster mode Standar Pemrosesan Informasi Federal (FIPS) kami. Support untuk ini tetap untuk cluster dalam FIPs mode non-mode.

Panduan ini berlaku untuk operasi kriptografi berikut:

- Generasi kunci Triple DES
 - CKM_DES3_KEY_GEN untuk Perpustakaan PKCS #11
 - DESedeKeygen untuk Penyedia JCE
 - genSymKey dengan -t=21 untuk KMU
- Enkripsi dengan kunci Triple DES (catatan: operasi dekripsi diperbolehkan)
 - Untuk Perpustakaan PKCS #11: CKM_DES3_CBC mengenkripsi, mengenkripsi, dan CKM_DES3_CBC_PAD mengenkripsi CKM_DES3_ECB
 - Untuk Penyedia JCE: DESede/CBC/PKCS5Padding mengenkripsi, mengenkripsi, DESede/CBC/NoPadding mengenkripsi, dan DESede/ECB/Padding mengenkripsi DESede/ECB/NoPadding
- Pembungkus kunci RSA, buka bungkus, enkripsi, dan dekripsi dengan padding PKCS #1 v1.5

- CKM_RSA_PKCSbungkus, buka bungkus, enkripsi, dan dekripsi untuk PKCS #11 SDK
- RSA/ECB/PKCS1Paddingbungkus, buka bungkus, enkripsi, dan dekripsi untuk JCE SDK
- wrapKeydan unWrapKey dengan -m 12 untuk KMU (catatan 12 adalah nilai untuk mekanismeRSA_PKCS)

[1] Untuk detail tentang perubahan ini, lihat Tabel 1 dan Tabel 5 dalam [Transisi Penggunaan Algoritma Kriptografi](#) dan Panjang Kunci.

Ketahanan di AWS CloudHSM

Infrastruktur AWS global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang AWS Wilayah dan Availability Zone, lihat [Infrastruktur AWS Global](#). Untuk informasi lebih lanjut tentang fitur AWS CloudHSM untuk mendukung ketahanan, lihat [AWS CloudHSM cluster ketersediaan tinggi dan load balancing](#).

Keamanan infrastruktur di AWS CloudHSM

Sebagai layanan terkelola, AWS CloudHSM dilindungi oleh prosedur keamanan jaringan AWS global yang dijelaskan dalam whitepaper [Amazon Web Services: Tinjauan Proses Keamanan](#).

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses AWS CloudHSM melalui jaringan. Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan principal IAM. Atau, Anda bisa menggunakan [AWS Security Token Service](#) (AWS STS) untuk membuat kredensial keamanan sementara untuk menandatangani permintaan.

Isolasi jaringan

Virtual private cloud (VPC) adalah jaringan virtual di area yang diisolasi secara logis di AWS cloud. Anda dapat membuat klaster di subnet privat di VPC. Anda dapat membuat subnet pribadi saat

membuat VPC. Untuk informasi selengkapnya, lihat [Buat virtual private cloud \(VPC\) untuk AWS CloudHSM](#).

Saat Anda membuat HSM, AWS CloudHSM letakkan elastic network interface (ENI) di subnet Anda sehingga Anda dapat berinteraksi dengan Anda. HSMs Untuk informasi selengkapnya, lihat [AWS CloudHSM arsitektur cluster](#).

AWS CloudHSM membuat grup keamanan yang memungkinkan komunikasi masuk dan keluar antara HSM di cluster Anda. Anda dapat menggunakan grup keamanan ini untuk mengaktifkan instans EC2 Anda untuk berkomunikasi dengan HSMs di cluster Anda. Untuk informasi selengkapnya, lihat [Konfigurasi grup keamanan instans Amazon EC2 Klien untuk AWS CloudHSM](#).

Otorisasi pengguna

Dengan AWS CloudHSM, operasi yang dilakukan pada HSM memerlukan kredensial dari pengguna HSM yang diautentikasi. Untuk informasi selengkapnya, lihat [the section called “Jenis pengguna”](#).

AWS CloudHSM dan titik akhir VPC

Anda dapat membuat koneksi pribadi antara VPC Anda dan AWS CloudHSM dengan membuat antarmuka VPC endpoint. Titik akhir antarmuka didukung oleh [AWS PrivateLink](#), teknologi yang memungkinkan Anda mengakses secara pribadi AWS CloudHSM APIs tanpa gateway internet, perangkat NAT, koneksi VPN, atau koneksi AWS Direct Connect. Instans di VPC Anda tidak memerlukan alamat IP publik untuk berkomunikasi. AWS CloudHSM APIs Lalu lintas antara VPC Anda dan AWS CloudHSM tidak meninggalkan jaringan Amazon.

Setiap titik akhir antarmuka diwakili oleh satu atau lebih [Antarmuka Jaringan Elastis](#) dalam subnet Anda.

Untuk informasi selengkapnya, lihat [Titik akhir VPC Antarmuka \(AWS PrivateLink\) di Panduan Pengguna Amazon VPC](#).

Pertimbangan untuk titik akhir AWS CloudHSM VPC

Sebelum menyiapkan titik akhir VPC antarmuka AWS CloudHSM, pastikan Anda meninjau [properti dan batasan titik akhir Antarmuka di](#) Panduan Pengguna Amazon VPC.

- AWS CloudHSM mendukung panggilan ke semua tindakan API-nya dari VPC Anda.

Membuat titik akhir VPC antarmuka untuk AWS CloudHSM

Anda dapat membuat titik akhir VPC untuk AWS CloudHSM layanan menggunakan konsol VPC Amazon atau (). AWS Command Line Interface AWS CLI Untuk informasi selengkapnya, lihat [Membuat titik akhir antarmuka](#) dalam Panduan Pengguna Amazon VPC.

Untuk membuat titik akhir VPC AWS CloudHSM, gunakan nama layanan berikut:

```
com.amazonaws.<region>.cloudhsmv2
```

Sebagai contoh, di Wilayah US West (Oregon) (us-west-2), nama layanan akan menjadi:

```
com.amazonaws.us-west-2.cloudhsmv2
```

Untuk mempermudah penggunaan VPC endpoint, Anda dapat mengaktifkan [nama host DNS privat](#) untuk VPC endpoint Anda. Jika Anda memilih opsi Aktifkan Nama DNS Pribadi, nama host AWS CloudHSM DNS standar (<https://cloudhsmv2.<region>.amazonaws.com>) dan <https://cloudhsmv2.<region>.api.aws>) akan diselesaikan ke titik akhir VPC Anda.

Opsi ini mempermudah untuk menggunakan VPC endpoint. Itu AWS SDKs dan AWS CLI gunakan nama host AWS CloudHSM DNS standar secara default, sehingga Anda tidak perlu menentukan URL titik akhir VPC dalam aplikasi dan perintah.

Untuk informasi selengkapnya, lihat [Mengakses layanan melalui titik akhir antarmuka](#) dalam Panduan Pengguna Amazon VPC.

Membuat kebijakan titik akhir VPC untuk AWS CloudHSM

Anda dapat melampirkan kebijakan titik akhir ke VPC endpoint yang mengendalikan akses ke AWS CloudHSM. Kebijakan menentukan informasi berikut ini:

- Prinsipal yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan.
- Sumber daya yang menjadi target tindakan.

Untuk informasi selengkapnya, lihat [Mengontrol Akses ke Layanan dengan titik akhir VPC](#) dalam Panduan Pengguna Amazon VPC.

Contoh: Kebijakan titik akhir VPC untuk tindakan AWS CloudHSM

Berikut ini adalah contoh kebijakan endpoint untuk AWS CloudHSM. Saat dilampirkan ke titik akhir, kebijakan ini memberikan akses ke AWS CloudHSM tindakan yang tercantum untuk semua prinsipal di semua sumber daya. Lihat [Identitas dan manajemen akses untuk AWS CloudHSM](#) AWS CloudHSM tindakan lain dan izin IAM terkait.

```
{
  "Statement": [
    {
      "Principal": "*",
      "Effect": "Allow",
      "Action": [
        "<cloudhsm>:<DescribeBackups>",
        "<cloudhsm>:<DescribeClusters>",
        "<cloudhsm>:<ListTags>",
      ],
      "Resource": "*"
    }
  ]
}
```

Perbarui manajemen di AWS CloudHSM

AWS mengelola firmware. Firmware dikelola oleh pihak ketiga, dan harus dievaluasi oleh NIST untuk kepatuhan FIPS 140-2 Level 3 atau FIPS 140-3 Level 3 tergantung pada jenis hsm. Hanya firmware yang telah ditandatangani secara kriptografi oleh kunci FIPS, yang AWS tidak memiliki akses ke sana, dapat diinstal.

Pemecahan masalah AWS CloudHSM

Jika Anda mengalami masalah AWS CloudHSM, topik berikut dapat membantu Anda menyelesaikannya.

Topik

- [AWS CloudHSM masalah yang diketahui](#)
- [AWS CloudHSM Kegagalan sinkronisasi kunci SDK 3 klien](#)
- [AWS CloudHSM Klien SDK 3 memverifikasi kinerja HSM dengan alat pkpspeed](#)
- [AWS CloudHSM Pengguna SDK 5 klien berisi nilai yang tidak konsisten](#)
- [AWS CloudHSM Kegagalan replikasi pengguna SDK 5 klien](#)
- [AWS CloudHSM Kegagalan replikasi kunci SDK 5 klien](#)
- [AWS CloudHSM kesalahan terlihat selama pemeriksaan ketersediaan kunci](#)
- [AWS CloudHSM mengekstraksi kunci menggunakan JCE](#)
- [Pelambatan HSM](#)
- [Jaga agar pengguna HSM tetap sinkron HSMs di kluster AWS CloudHSM](#)
- [Kehilangan koneksi ke AWS CloudHSM cluster](#)
- [Log AWS CloudHSM audit tidak ada di CloudWatch](#)
- [Kustom IVs dengan panjang yang tidak sesuai untuk pembungkus kunci AES AWS CloudHSM](#)
- [Menyelesaikan AWS CloudHSM kegagalan pembuatan cluster](#)
- [Mengambil log konfigurasi AWS CloudHSM klien](#)

AWS CloudHSM masalah yang diketahui

AWS CloudHSM memiliki masalah yang diketahui berikut. Pilih topik untuk mempelajari selengkapnya.

Topik

- [Masalah yang diketahui untuk semua instance HSM](#)
- [Masalah yang diketahui untuk instance AWS CloudHSM hsm1.medium](#)
- [Masalah yang diketahui untuk instance AWS CloudHSM hsm2m.medium](#)

- [Masalah yang diketahui untuk pustaka PKCS #11 untuk AWS CloudHSM](#)
- [Masalah yang diketahui untuk JCE SDK untuk AWS CloudHSM](#)
- [Masalah yang diketahui untuk OpenSSL Dynamic Engine untuk AWS CloudHSM](#)
- [Masalah yang diketahui untuk Penyedia Penyimpanan Kunci \(KSP\) untuk AWS CloudHSM](#)
- [Masalah yang diketahui untuk Penyedia OpenSSL untuk AWS CloudHSM](#)
- [Masalah yang diketahui untuk instans Amazon EC2 yang menjalankan Amazon Linux 2 dengan AWS CloudHSM](#)
- [Masalah yang diketahui untuk mengintegrasikan aplikasi pihak ketiga dengan AWS CloudHSM](#)
- [Masalah yang diketahui untuk modifikasi AWS CloudHSM cluster](#)
- [Masalah kegagalan operasi yang diketahui menggunakan AWS CloudHSM klien versi 5.12.0 di `hsm2.medium`](#)

Masalah yang diketahui untuk semua instance HSM

Masalah berikut memengaruhi semua AWS CloudHSM pengguna terlepas dari apakah mereka menggunakan alat baris perintah `key_mgmt_util`, PKCS #11 SDK, JCE SDK, atau OpenSSL SDK.

Topik

- [Masalah: Pembungkus kunci AES menggunakan bantalan PKCS #5 alih-alih menyediakan implementasi patuh standar pembungkus kunci dengan bantalan `no1`](#)
- [Masalah: Daemon klien memerlukan setidaknya satu alamat IP yang valid dalam file konfigurasi untuk berhasil menyambung ke kluster](#)
- [Masalah: Ada batas atas 16 KB pada data yang dapat di-hash dan ditandatangani dengan AWS CloudHSM menggunakan Client SDK 3](#)
- [Masalah: Kunci yang diimpor tidak dapat ditentukan sebagai tidak dapat diekspor](#)
- [Masalah: Mekanisme default untuk `WrapKey` dan `unWrapKey` perintah di `key\_mgmt\_util` telah dihapus](#)
- [Masalah: Jika Anda memiliki satu HSM di kluster Anda, failover HSM tidak bekerja dengan benar](#)
- [Masalah: Jika Anda melebihi kapasitas kunci di cluster Anda HSMs dalam waktu singkat, klien memasuki status kesalahan yang tidak tertangani](#)
- [Masalah: Operasi digest dengan kunci HMAC ukuran lebih besar dari 800 byte tidak didukung](#)
- [Masalah: Alat `client\_info`, didistribusikan dengan SDK Klien 3, menghapus isi jalur yang ditentukan oleh argumen output opsional](#)

- [Masalah: Anda menerima kesalahan saat menjalankan alat konfigurasi SDK 5 menggunakan --cluster-id argumen di lingkungan kontainer](#)
- [Masalah: Anda menerima kesalahan “Gagal membuat cert/key dari file pfx yang disediakan. Kesalahan: NotPkcs 8”](#)
- [Masalah: Penandatanganan ECDSA gagal dengan kesalahan “mekanisme tidak valid” yang dimulai dengan SDK 5.16](#)
- [Masalah: Operasi penandatanganan dengan data pra-hash tidak menghapus token sesi dengan benar dalam mode interaktif](#)
- [Masalah: Sertifikat klien default perpustakaan klien CloudHSM kedaluwarsa pada 31 Januari 2026](#)

Masalah: Pembungkus kunci AES menggunakan bantalan PKCS #5 alih-alih menyediakan implementasi patuh standar pembungkus kunci dengan bantalan nol

Selain itu, pembungkus kunci tanpa bantalan dan bantalan nol tidak didukung.

- Dampak: Tidak ada dampak jika Anda membungkus dan membuka bungkusnya menggunakan algoritme ini di dalamnya AWS CloudHSM. Namun, kunci yang dibungkus AWS CloudHSM tidak dapat dibuka dalam perangkat lunak lain HSMs atau yang mengharapkan kepatuhan terhadap spesifikasi tanpa padding. Hal ini karena delapan byte bantalan data mungkin ditambahkan ke akhir data kunci Anda selama pembukaan patuh standar. Kunci yang dibungkus secara eksternal tidak dapat dibuka dengan benar ke dalam sebuah instance. AWS CloudHSM
- Pemecahan masalah: Untuk membuka kunci eksternal yang dibungkus dengan Pembungkus Kunci AES dengan Bantalan PKCS #5 pada instans AWS CloudHSM, strip bantalan ekstra sebelum Anda mencoba untuk menggunakan kunci. Anda dapat melakukan ini dengan pemangkasan byte tambahan dalam editor file atau menyalin hanya byte kunci ke penyangga baru dalam kode Anda.
- Status resolusi: Dengan rilis perangkat lunak dan klien 3.1.0, AWS CloudHSM menyediakan opsi yang sesuai standar untuk pembungkus kunci AES. Untuk informasi selengkapnya, lihat [Pembungkus Kunci AES](#).

Masalah: Daemon klien memerlukan setidaknya satu alamat IP yang valid dalam file konfigurasi untuk berhasil menyambung ke klaster

- Dampak: Jika Anda menghapus setiap HSM di cluster Anda dan kemudian menambahkan HSM lain, yang mendapat alamat IP baru, daemon klien terus mencari alamat IP asli Anda HSMs .

- Solusi: Jika Anda menjalankan beban kerja intermiten, kami sarankan Anda menggunakan `IpAddress` argumen dalam [CreateHsm](#) fungsi untuk mengatur elastic network interface (ENI) ke nilai aslinya. Perhatikan bahwa ENI khusus untuk Availability Zone (AZ). Alternatifnya adalah menghapus file `/opt/cloudhsm/daemon/1/cluster.info` dan kemudian mengatur ulang konfigurasi klien ke alamat IP HSM baru Anda. Anda dapat menggunakan perintah `client -a <IP address>`. Untuk informasi selengkapnya, lihat [Menginstal dan Mengkonfigurasi AWS CloudHSM Klien \(Linux\)](#) atau [Menginstal dan Mengkonfigurasi AWS CloudHSM Klien \(Windows\)](#).

Masalah: Ada batas atas 16 KB pada data yang dapat di-hash dan ditandatangani dengan AWS CloudHSM menggunakan Client SDK 3

- Status resolusi: Data kurang dalam ukuran dari 16 KB terus dikirim ke HSM untuk hashing. Kami telah menambahkan kemampuan untuk hash lokal, dalam perangkat lunak, data dalam ukuran antara 16 KB dan 64 KB. Klien SDK 5 secara eksplisit akan gagal jika buffer data lebih besar dari 64KB. Anda harus memperbarui klien dan SDK ke versi yang lebih besar dari 5.0.0 atau lebih tinggi untuk mendapatkan keuntungan dari perbaikan.

Masalah: Kunci yang diimpor tidak dapat ditentukan sebagai tidak dapat diekspor

- Status Resolusi: Masalah ini telah diperbaiki. Tidak ada tindakan yang diperlukan dari pihak Anda untuk mendapatkan manfaat dari perbaikan.

Masalah: Mekanisme default untuk `WrapKey` dan `unWrapKey` perintah di `key_mgmt_util` telah dihapus

- Resolusi: Saat menggunakan `WrapKey` atau `unWrapKey` perintah, Anda harus menggunakan `-m` opsi untuk menentukan mekanisme. Lihat contoh di [WrapKey](#) atau [unWrapKey](#) artikel untuk informasi selengkapnya.

Masalah: Jika Anda memiliki satu HSM di kluster Anda, failover HSM tidak bekerja dengan benar

- Dampak: Jika instans HSM tunggal di kluster Anda kehilangan konektivitas, klien tidak akan menyambung kembali dengan itu bahkan jika instans HSM kemudian dipulihkan.

- Pemecahan masalah: Kami merekomendasikan setidaknya dua instans HSM di setiap kluster produksi. Jika Anda menggunakan konfigurasi ini, Anda tidak akan terpengaruh oleh masalah ini. Untuk kluster HSM tunggal, pentalkan daemon klien untuk memulihkan konektivitas.
- Status resolusi: Masalah ini telah diselesaikan dalam rilis 1.1.2 AWS CloudHSM klien. Anda harus meningkatkan ke klien ini untuk mendapatkan manfaat dari perbaikan.

Masalah: Jika Anda melebihi kapasitas kunci di cluster Anda HSMs dalam waktu singkat, klien memasuki status kesalahan yang tidak tertangani

- Dampak: Ketika klien bertemu status galat tidak tertangani, klien akan membeku dan harus dimulai ulang.
- Pemecahan masalah: Uji throughput Anda untuk memastikan Anda tidak membuat kunci sesi pada tingkat yang klien tidak mampu menanganinya. Anda dapat menurunkan tingkat Anda dengan menambahkan HSM ke kluster atau memperlambat pembuatan kunci sesi.
- Status resolusi: Masalah ini telah diselesaikan dalam rilis 1.1.2 AWS CloudHSM klien. Anda harus meningkatkan ke klien ini untuk mendapatkan manfaat dari perbaikan.

Masalah: Operasi digest dengan kunci HMAC ukuran lebih besar dari 800 byte tidak didukung

- Dampak: Kunci HMAC lebih besar dari 800 byte dapat dihasilkan pada atau diimpor ke HSM. Namun, jika Anda menggunakan kunci yang lebih besar ini dalam operasi digest melalui JCE atau `key_mgmt_util`, operasi akan gagal. Perhatikan bahwa jika Anda menggunakan PKCS11, kunci HMAC terbatas pada ukuran 64 byte.
- Pemecahan masalah: Jika Anda akan menggunakan kunci HMAC untuk operasi digest pada HSM, pastikan ukurannya lebih kecil dari 800 byte.
- Status resolusi: Tidak ada saat ini.

Masalah: Alat `client_info`, didistribusikan dengan SDK Klien 3, menghapus isi jalur yang ditentukan oleh argumen output opsional

- Dampak: Semua file yang ada dan sub-direktori di bawah jalur output yang ditentukan mungkin hilang secara permanen.

- Pemecahan masalah: Jangan gunakan argumen opsional `-output` *path* saat menggunakan alat `client_info`.
- Status resolusi: Masalah ini telah diatasi di [rilis SDK Klien 3.3.2](#). Anda harus meningkatkan ke klien ini untuk mendapatkan manfaat dari perbaikan.

Masalah: Anda menerima kesalahan saat menjalankan alat konfigurasi SDK 5 menggunakan `--cluster-id` argumen di lingkungan kontainer

Anda menerima kesalahan berikut saat menggunakan argumen `--cluster-id` dengan Configure Tool:

No credentials in the property bag

Kesalahan ini disebabkan oleh pembaruan ke Layanan Metadata Instans Versi 2 (IMDSv2). Untuk informasi lebih lanjut, lihat [IMDSv2](#) dokumentasi.

- Dampak: Masalah ini akan memengaruhi pengguna yang menjalankan alat konfigurasi pada SDK versi 5.5.0 dan yang lebih baru di lingkungan kontainer dan memanfaatkan metadata instans EC2 untuk memberikan kredensial.
- Solusi: Setel batas hop respons PUT menjadi setidaknya dua. Untuk panduan tentang cara melakukannya, lihat [Mengonfigurasi opsi metadata instance](#).

Masalah: Anda menerima kesalahan “Gagal membuat cert/key dari file pfx yang disediakan. Kesalahan: NotPkcs 8”

- Solusi: Anda dapat mengonversi kunci pribadi SSL khusus ke PKCS8 format dengan perintah `openssl: openssl pkcs8 -topk8 -inform PEM -outform PEM -in ssl_private_key -out ssl_private_key_pkcs8`
- Status resolusi: Masalah ini telah diselesaikan dalam rilis [SDK 5.12.0 klien](#). Anda harus meningkatkan ke versi klien ini atau yang lebih baru untuk mendapatkan keuntungan dari perbaikan.

Masalah: Penandatanganan ECDSA gagal dengan kesalahan “mekanisme tidak valid” yang dimulai dengan SDK 5.16

- Dampak: Operasi penandatanganan ECDSA gagal saat menggunakan fungsi hash yang lebih lemah dari kekuatan kunci. Kegagalan ini terjadi karena [FIPS 186-5](#) membutuhkan fungsi hash setidaknya sekuat kekuatan kunci.

Anda mungkin melihat kesalahan yang mirip dengan ini di log klien Anda:

```
[cloudhsm_provider::hsm1::session::ecdsa::sign::common][[]] Digest security strength (80) is weaker than the key security strength (128)
```

- Solusi: Jika Anda tidak dapat memperbarui fungsi hash Anda, Anda dapat bermigrasi ke kluster non-FIPS, yang tidak menerapkan persyaratan kekuatan hash. Namun, kami sarankan untuk memperbarui fungsi hash Anda untuk mempertahankan kepatuhan FIPS.

Sebagai solusi tambahan, kami telah menambahkan opsi konfigurasi untuk melewati persyaratan ini. Harap dicatat, opsi ini tidak disarankan, karena menggunakan ECDSA dengan fungsi hash yang lebih lemah tidak mengikuti praktik terbaik keamanan. Untuk menggunakan opsi ini, jalankan perintah berikut (ganti `configure-cli` dengan alat konfigurasi untuk SDK yang digunakan: [the section called “Sintaksis”](#)):

```
sudo /opt/cloudhsm/bin/configure-cli --enable-ecdsa-with-weak-hash-function
```

- Resolusi: Gunakan fungsi hash yang setidaknya sekuat kunci ECDSA Anda. Untuk informasi tentang fungsi hash dan kekuatan kunci ECDSA, lihat Tabel 2 dan 3 di [NIST SP 800-57](#) Bagian 1 Rev 5.

Masalah: Operasi penandatanganan dengan data pra-hash tidak menghapus token sesi dengan benar dalam mode interaktif

- Dampak: Saat menggunakan CloudHSM CLI dalam mode interaktif dengan SDK versi 5.16.1, operasi penandatanganan dengan data pra-hash gagal menghapus token sesi dengan benar.
- Solusi: Gunakan mode perintah tunggal alih-alih mode interaktif saat melakukan operasi penandatanganan dengan data pra-hash. Ini memastikan pembersihan token yang tepat setelah setiap operasi.
- Status resolusi: Masalah ini telah diselesaikan di CloudHSM SDK 5.16.2. Tingkatkan ke versi 5.16.2 atau yang lebih baru untuk mendapatkan manfaat dari perbaikan.

Masalah: Sertifikat klien default perpustakaan klien CloudHSM kedaluwarsa pada 31 Januari 2026

- Dampak: Tidak ada dampak bagi pelanggan setelah 31 Januari 2026. [Semua komunikasi antara klien dan HSM dijamin oleh klien-HSM TLS seperti yang dijelaskan di sini](#). Client-HSM TLS menggunakan owned/managed sertifikat pelanggan yang dikonfigurasi oleh pelanggan selama inisialisasi cluster.
- Status resolusi: terselesaikan.

Masalah yang diketahui untuk instance AWS CloudHSM hsm1.medium

Masalah berikut memengaruhi semua instance AWS CloudHSM hsm1.medium.

Topik

- [Masalah: HSM tidak dapat membuat lebih dari 250 pengguna](#)

Masalah: HSM tidak dapat membuat lebih dari 250 pengguna

- Solusi: Masalah ini diselesaikan pada tipe instance AWS CloudHSM hsm2m.medium.
- Status resolusi: Tidak ada saat ini.

Masalah yang diketahui untuk instance AWS CloudHSM hsm2m.medium

Masalah berikut memengaruhi semua instance AWS CloudHSM hsm2m.medium.

Topik

- [Masalah: Peningkatan latensi login di hsm2m.medium](#)
- [Masalah: Peningkatan latensi kunci temukan di hsm2m.medium](#)
- [Masalah: CO yang mencoba menyetel atribut tepercaya dari kunci akan gagal dengan Client SDK 5.12.0 dan yang lebih lama](#)
- [Masalah: Verifikasi ECDSA akan gagal dengan Client SDK 5.12.0 dan sebelumnya untuk cluster dalam mode FIPS](#)
- [Masalah: Hanya sertifikat berformat PEM yang dapat didaftarkan sebagai jangkar kepercayaan mtls dengan CloudHSM CLI](#)

- [Masalah: Aplikasi pelanggan akan berhenti memproses semua permintaan saat menggunakan mTL dengan kunci pribadi klien yang dilindungi frasa sandi.](#)
- [Masalah: Replikasi pengguna gagal saat menggunakan CloudHSM CLI](#)
- [Masalah: Operasi dapat gagal selama pembuatan cadangan](#)
- [Masalah: Klien SDK 5.8 dan di atasnya tidak melakukan percobaan ulang otomatis untuk operasi terhambat HSM dalam beberapa skenario di hsm2m.medium](#)
- [Masalah: AES/CBC membuka operasi dengan semua nol IV gagal pada hsm2m.medium](#)
- [Masalah: Kegagalan untuk menginisialisasi koneksi HSM selama aplikasi dingin dimulai pada hsm2m.medium](#)

Masalah: Peningkatan latensi login di hsm2m.medium

- Dampak: Masuk ke hsm2m.medium mengikuti interpretasi persyaratan kepatuhan yang terlalu ketat, yang menghasilkan peningkatan latensi.
- Resolusi: Jika Anda membuat instance hsm2m.medium baru atau bermigrasi ke hsm2m.medium dari hsm1.medium sebelum 20 Desember 2025, Anda perlu mengatur ulang kata sandi Anda untuk memanfaatkan peningkatan kinerja yang telah kami terapkan untuk operasi login. Lihat [perubahan-kata sandi untuk instruksi](#).

Masalah: Peningkatan latensi kunci temukan di hsm2m.medium

- Dampak: Instans HSM hsm2m.medium telah meningkatkan arsitektur fair share yang menghasilkan kinerja yang dapat diprediksi lebih konsisten dibandingkan dengan hsm1.medium. Dengan hsm1.medium, pelanggan dapat mengamati kinerja kunci pencarian yang lebih tinggi karena penggunaan sumber daya HSM yang tidak teratur. Namun, hsm1.medium find key performance akan menurun ketika instance HSM ditambah atau diperbarui dengan firmware baru. Masalah ini memengaruhi operasi seperti `KeyStore.getKey()` di JCE.
- Resolusi: Masalah ini telah diselesaikan. Sebagai praktik terbaik, cache hasil dari operasi find key. Caching akan mengurangi jumlah total operasi find key karena merupakan operasi intensif sumber daya di HSM. Selain itu, terapkan percobaan ulang sisi klien dengan backoff eksponensial dan jitter untuk mengurangi kegagalan pelambatan HSM.

Masalah: CO yang mencoba menyetel atribut tepercaya dari kunci akan gagal dengan Client SDK 5.12.0 dan yang lebih lama

- Dampak: Setiap pengguna CO yang mencoba menyetel atribut tepercaya dari kunci akan menerima kesalahan yang menunjukkan hal itu `User type should be CO or CU`.
- Resolusi: Versi SDK Klien yang akan datang akan menyelesaikan masalah ini. Pembaruan akan diumumkan di panduan pengguna kami [Riwayat dokumen](#).

Masalah: Verifikasi ECDSA akan gagal dengan Client SDK 5.12.0 dan sebelumnya untuk cluster dalam mode FIPS

- Dampak: ECDSA memverifikasi operasi yang dilakukan untuk HSMs dalam mode FIPS akan gagal.
- Status resolusi: Masalah ini telah diselesaikan dalam rilis [SDK 5.13.0 klien](#). Anda harus meningkatkan ke versi klien ini atau yang lebih baru untuk mendapatkan keuntungan dari perbaikan.

Masalah: Hanya sertifikat berformat PEM yang dapat didaftarkan sebagai jangkar kepercayaan mtls dengan CloudHSM CLI

- Dampak: Sertifikat dalam format DER tidak dapat didaftarkan sebagai jangkar kepercayaan mTLS dengan CloudHSM CLI.
- Solusi: Anda dapat mengonversi sertifikat dalam format DER ke format PEM dengan perintah `openssl x509 -inform DER -outform PEM -in certificate.der -out certificate.pem`

Masalah: Aplikasi pelanggan akan berhenti memproses semua permintaan saat menggunakan mTL dengan kunci pribadi klien yang dilindungi frasa sandi.

- Dampak: Semua operasi yang dilakukan oleh aplikasi akan dihentikan dan pengguna akan diminta untuk frasa sandi pada input standar beberapa kali sepanjang masa aplikasi. Operasi akan habis waktu dan gagal jika frasa sandi tidak diberikan sebelum durasi batas waktu operasi.
- Solusi: Kunci pribadi terenkripsi frasa sandi tidak didukung untuk mTL. Hapus enkripsi frasa sandi dari kunci pribadi klien

Masalah: Replikasi pengguna gagal saat menggunakan CloudHSM CLI

- Dampak: Replikasi pengguna gagal pada instance hsm2m.medium saat menggunakan CloudHSM CLI. `user replicate` Perintah berfungsi seperti yang diharapkan pada instance hsm1.medium.
- Resolusi: Masalah ini telah diselesaikan.

Masalah: Operasi dapat gagal selama pembuatan cadangan

- Dampak: Operasi seperti menghasilkan angka acak dapat gagal pada instans hsm2m.medium sementara AWS CloudHSM membuat cadangan.
- Resolusi: Untuk meminimalkan gangguan layanan, terapkan praktik terbaik ini:
 - Buat cluster multi-HSM
 - Konfigurasi aplikasi Anda untuk mencoba lagi operasi kluster

Untuk informasi lebih lanjut tentang praktik terbaik, lihat [Praktik terbaik untuk AWS CloudHSM](#).

Masalah: Klien SDK 5.8 dan di atasnya tidak melakukan percobaan ulang otomatis untuk operasi terhambat HSM dalam beberapa skenario di hsm2m.medium

- Dampak: Klien SDK 5.8 dan di atasnya tidak akan mencoba lagi beberapa operasi terbatas HSM
- Solusi: Ikuti praktik terbaik untuk merancang kluster Anda guna menangani pemuatan dan implementasi percobaan ulang tingkat aplikasi. Saat ini kami sedang mengerjakan perbaikan. Pembaruan akan diumumkan di panduan pengguna kami [Riwayat dokumen](#).
- Status resolusi: Masalah ini telah diselesaikan di SDK AWS CloudHSM Klien 5.16.2. Anda harus meningkatkan ke versi klien ini atau yang lebih baru untuk mendapatkan keuntungan dari perbaikan.

Masalah: AES/CBC membuka operasi dengan semua nol IV gagal pada hsm2m.medium

- Dampak: Saat menggunakan AES/CBC mekanisme untuk membuka kunci menggunakan penyedia AWS CloudHSM JCE, operasi dengan IV terisi nol 16 byte gagal pada instance hsm2m.medium, karena pemeriksaan validasi tambahan yang tidak ada dalam instance hsm1.medium.

- Status Resolusi: Kami sedang mengerjakan perbaikan yang memungkinkan zero-byte diterima selama IVs AES/CBC operasi unwrap.

Masalah: Kegagalan untuk menginisialisasi koneksi HSM selama aplikasi dingin dimulai pada hsm2m.medium

- Dampak: Masalah ini memengaruhi start dingin seperti penerapan aplikasi klien atau restart. Instans HSM hsm2m.medium telah meningkatkan arsitektur fair share yang memastikan kinerja, throughput, dan latensi yang lebih konsisten untuk semua pelanggan. Saat ini di hsm1.medium, Anda dapat mengamati kinerja yang lebih tinggi dari yang dimaksudkan untuk inisialisasi koneksi HSM bersamaan. Namun, kinerja inisialisasi koneksi hsm1.medium akan bervariasi berdasarkan pembaruan sistem yang mendasarinya.
- Resolusi: Ikuti [praktik terbaik](#) dan terhuyung-huyung penerapan aplikasi klien dan restart untuk membatasi jumlah aplikasi klien yang menginisialisasi koneksi HSM secara bersamaan. Kami juga menyarankan Anda menerapkan percobaan ulang tingkat aplikasi untuk inisialisasi aplikasi klien. Selain itu, bootstrap menggunakan alat konfigurasi dengan `--cluster-id <cluster ID>` untuk menambahkan semua IP HSM ke file konfigurasi klien. Perilaku ini telah ditingkatkan di AWS CloudHSM Client SDK versi 5.17.1 dan yang lebih baru. Kami merekomendasikan untuk meningkatkan ke versi SDK terbaru untuk mendapatkan manfaat dari peningkatan ini.

Masalah yang diketahui untuk pustaka PKCS #11 untuk AWS CloudHSM

Isu-isu berikut berdampak pada pustaka PKCS #11 untuk AWS CloudHSM

Topik

- [Masalah: Pembungkus kunci AES di versi 3.0.0 dari pustaka PKCS #11 tidak memvalidasi sebelum digunakan IVs](#)
- [Masalah: PKCS #11 SDK 2.0.4 dan versi sebelumnya selalu menggunakan IV default 0xA6A6A6A6A6A6A6A6 untuk membungkus dan membuka kunci AES](#)
- [Masalah: atribut CKA_DERIVE tidak didukung dan tidak ditangani](#)
- [Masalah: atribut CKA_SENSITIVE tidak didukung dan tidak ditangani](#)
- [Masalah: Hashing multibagian dan penandatanganan tidak didukung](#)
- [Masalah: C_GenerateKeyPair tidak menangani CKA_MODULUS_BITS atau CKA_PUBLIC_EXPONENT dalam templat privat dengan cara yang patuh dengan standar](#)

- [Masalah: Penyangga untuk operasi API C_Encrypt dan C_Decrypt tidak dapat melebihi 16 KB saat menggunakan mekanisme CKM_AES_GCM](#)
- [Masalah: Derivasi kunci Elliptic-curve Diffie-Hellman \(ECDH\) dijalankan sebagian dalam HSM](#)
- [Masalah: Verifikasi tanda tangan secp256k1 gagal pada EL6 platform seperti Cent dan RHEL 6 OS6](#)
- [Masalah: Urutan panggilan fungsi yang salah memberikan hasil yang tidak terdefinisi, bukannya gagal](#)
- [Masalah: Sesi Hanya Baca tidak didukung di SDK 5](#)
- [Masalah: file cryptoki.h header hanya untuk Windows](#)

Masalah: Pembungkus kunci AES di versi 3.0.0 dari pustaka PKCS #11 tidak memvalidasi sebelum digunakan IVs

Jika Anda menentukan IV lebih pendek dari 8 byte panjangnya, IV diberi bantalan dengan byte tak terduga sebelum digunakan.

 Note

Ini memengaruhi C_WrapKey dengan mekanisme CKM_AES_KEY_WRAP saja.

- Dampak: Jika Anda memberikan IV yang lebih pendek dari 8 byte di pustaka PKCS #11 versi 3.0.0, Anda mungkin tidak dapat membuka kunci.
- Solusi:
 - Kami sangat menyarankan Anda meningkatkan ke versi 3.0.1 atau lebih tinggi dari pustaka PKCS #11, yang dengan benar memberlakukan panjang IV selama pembungkus kunci AES. Ubah kode pembungkus Anda untuk melewati NULL IV, atau tentukan IV default dari 0xA6A6A6A6A6A6A6A6. Untuk informasi selengkapnya, lihat [Custom IVs with Non-Compliant Length for AES](#) Key Wrap.
 - Jika Anda membungkus kunci dengan versi 3.0.0 dari pustaka PKCS #11 menggunakan IV yang lebih pendek dari 8 byte, hubungi kami untuk [dukungan](#).
- Status resolusi: Masalah ini telah diatasi di pustaka PKCS #11 versi 3.0.1. Untuk membungkus kunci menggunakan pembungkus kunci AES, tentukan IV yang NULL atau 8 byte panjangnya.

Masalah: PKCS #11 SDK 2.0.4 dan versi sebelumnya selalu menggunakan IV default **0xA6A6A6A6A6A6A6A6** untuk membungkus dan membuka kunci AES

Pengguna yang disediakan diam-diam IVs diabaikan.

 Note

Ini memengaruhi C_WrapKey dengan mekanisme CKM_AES_KEY_WRAP saja.

- Dampak:
 - Jika Anda menggunakan PKCS #11 SDK 2.0.4 atau versi sebelumnya dan IV yang disediakan pengguna, kunci Anda dibungkus dengan IV default 0xA6A6A6A6A6A6A6A6.
 - Jika Anda menggunakan PKCS #11 SDK 3.0.0 atau versi lebih baru dan IV yang disediakan pengguna, kunci Anda dibungkus dengan IV yang disediakan pengguna..
- Solusi:
 - Untuk membuka kunci yang dibungkus dengan PKCS #11 SDK 2.0.4 atau sebelumnya, gunakan IV default dari 0xA6A6A6A6A6A6A6A6.
 - Untuk membuka kunci yang dibungkus dengan PKCS #11 SDK 3.0.0 atau yang lebih baru, gunakan IV yang disediakan pengguna.
- Status resolusi: Kami sangat menyarankan Anda untuk mengubah kode pembungkus dan pembuka untuk melewati NULL IV, atau menentukan IV default 0xA6A6A6A6A6A6A6A6.

Masalah: atribut **CKA_DERIVE** tidak didukung dan tidak ditangani

- Status resolusi: Kami telah menerapkan perbaikan untuk menerima CKA_DERIVE jika diatur ke FALSE. CKA_DERIVE diatur ke TRUE tidak akan didukung sampai kita mulai menambahkan dukungan fungsi derivasi kunci ke AWS CloudHSM. Anda harus memperbarui klien dan SDK ke versi 1.1.1 atau lebih tinggi untuk mendapatkan manfaat dari perbaikan.

Masalah: atribut **CKA_SENSITIVE** tidak didukung dan tidak ditangani

- Status resolusi: Kami telah menerapkan perbaikan untuk menerima dan menghormati atribut CKA_SENSITIVE dengan benar. Anda harus memperbarui klien dan SDK ke versi 1.1.1 atau lebih tinggi untuk mendapatkan manfaat dari perbaikan.

Masalah: Hashing multibagian dan penandatanganan tidak didukung

- Dampak: `C_DigestUpdate` dan `C_DigestFinal` tidak diimplementasikan. `C_SignFinal` juga tidak diimplementasikan dan akan gagal dengan `CKR_ARGUMENTS_BAD` untuk selain penyangga `NULL`.
- Solusi: Hash data Anda dalam aplikasi Anda dan gunakan AWS CloudHSM hanya untuk menandatangani hash.
- Status resolusi: Kami memperbaiki klien dan menerapkan hashing multipart dengan benar. SDKs Pembaruan akan diumumkan di forum AWS CloudHSM dan pada halaman riwayat versi.

Masalah: `C_GenerateKeyPair` tidak menangani `CKA_MODULUS_BITS` atau `CKA_PUBLIC_EXPONENT` dalam templat privat dengan cara yang patuh dengan standar

- Dampak: `C_GenerateKeyPair` harus mengembalikan `CKA_TEMPLATE_INCONSISTENT` ketika templat privat berisi `CKA_MODULUS_BITS` atau `CKA_PUBLIC_EXPONENT`. Ini malah menghasilkan kunci privat yang semua bidang penggunaannya diatur ke `FALSE`. Kunci tidak dapat digunakan.
- Pemecahan masalah: Kami merekomendasikan bahwa aplikasi Anda memeriksa nilai bidang penggunaan selain kode kesalahan.
- Status resolusi: Kami menerapkan perbaikan untuk mengembalikan pesan kesalahan yang tepat ketika templat kunci privat yang salah digunakan. Pembaruan pustaka PKCS #11 akan diumumkan di halaman riwayat versi.

Masalah: Penyangga untuk operasi API `C_Encrypt` dan `C_Decrypt` tidak dapat melebihi 16 KB saat menggunakan mekanisme `CKM_AES_GCM`

AWS CloudHSM tidak mendukung enkripsi AES-GCM multibagian.

- Dampak: Anda tidak dapat menggunakan mekanisme `CKM_AES_GCM` untuk mengenkripsi data yang lebih besar dari 16 KB.
- Solusi: Anda dapat menggunakan mekanisme alternatif seperti `CKM_AES_CBC`, `CKM_AES_CBC_PAD`, atau Anda dapat membagi data Anda menjadi beberapa bagian dan mengenkripsi setiap bagian menggunakan satu per satu. `AES_GCM` Jika Anda menggunakan `AES_GCM`, Anda harus mengelola pembagian data Anda dan enkripsi berikutnya. AWS CloudHSM tidak melakukan enkripsi AES-GCM multibagian untuk Anda. Perhatikan bahwa FIPS mengharuskan vektor inisialisasi (IV) untuk

AES-GCM dihasilkan pada HSM. Oleh karena itu, IV untuk setiap bagian data terenkripsi AES-GCM Anda akan berbeda.

- Status resolusi: Kami memperbaiki SDK gagal secara eksplisit jika penyangga data terlalu besar. Kami mengembalikan CKR_MECHANISM_INVALID untuk operasi API C_EncryptUpdate dan C_DecryptUpdate. Kami sedang mengevaluasi alternatif untuk mendukung penyangga yang lebih besar tanpa mengandalkan enkripsi multibagian. Pembaruan akan diumumkan di AWS CloudHSM forum dan di halaman riwayat versi.

Masalah: Derivasi kunci Elliptic-curve Diffie-Hellman (ECDH) dijalankan sebagian dalam HSM

Kunci privat EC Anda tetap berada dalam HSM setiap saat, tetapi proses derivasi kunci dilakukan dalam beberapa langkah. Akibatnya, hasil menengah dari setiap langkah tersedia pada klien.

- Dampak: Di Client SDK 3, kunci yang diturunkan menggunakan CKM_ECDH1_DERIVE mekanisme pertama kali tersedia pada klien dan kemudian diimpor ke HSM. Sebuah handel kunci kemudian kembali ke aplikasi Anda.
- Solusi: Jika Anda menerapkan SSL/TLS Offload di AWS CloudHSM, batasan ini mungkin tidak menjadi masalah. Jika aplikasi Anda memerlukan kunci Anda untuk tetap berada dalam batas FIPS setiap saat, pertimbangkan untuk menggunakan protokol alternatif yang tidak bergantung pada derivasi kunci ECDH.
- Status resolusi: SDK 5.16 sekarang mendukung ECDH dengan Derivasi Kunci yang dilakukan sepenuhnya dalam HSM.

Masalah: Verifikasi tanda tangan secp256k1 gagal pada EL6 platform seperti Cent dan RHEL 6 OS6

Hal ini terjadi karena pustaka PKCS #11 CloudHSM menghindari panggilan jaringan selama inisialisasi operasi verifikasi dengan menggunakan OpenSSL untuk memverifikasi data kurva EC. Karena Secp256k1 tidak didukung oleh paket OpenSSL default pada platform, inisialisasi gagal. EL6

- Dampak: Verifikasi tanda tangan Secp256k1 akan gagal pada platform. EL6 Panggilan verifikasi akan gagal dengan kesalahan CKR_HOST_MEMORY.
- Solusi: Sebaiknya gunakan Amazon Linux 1 atau EL7 platform apa pun jika aplikasi PKCS #11 Anda perlu memverifikasi tanda tangan secp256k1. Atau, tingkatkan ke versi paket OpenSSL yang mendukung kurva secp256k1.

- Status resolusi: Kami menerapkan perbaikan untuk kembali ke HSM jika validasi kurva lokal tidak tersedia. Pembaruan pustaka PKCS #11 akan diumumkan di halaman [riwayat versi](#).

Masalah: Urutan panggilan fungsi yang salah memberikan hasil yang tidak terdefinisi, bukannya gagal

- Dampak: Jika Anda memanggil urutan fungsi yang salah, hasil akhir tidak benar meskipun fungsi individu panggilan kembali sukses. Misalnya, data yang didekripsi mungkin tidak cocok dengan teks terang asli atau tanda tangan mungkin gagal untuk diverifikasi. Masalah ini memengaruhi operasi satu bagian dan multibagian.

Contoh urutan fungsi yang salah:

- C_EncryptInit/C_EncryptUpdate diikuti oleh C_Encrypt
- C_DecryptInit/C_DecryptUpdate diikuti oleh C_Decrypt
- C_SignInit/C_SignUpdate diikuti oleh C_Sign
- C_VerifyInit/C_VerifyUpdate diikuti oleh C_Verify
- C_FindObjectsInit diikuti oleh C_FindObjectsInit
- Pemecahan masalah: Aplikasi Anda harus, sesuai dengan spesifikasi PKCS #11, menggunakan urutan fungsi panggilan yang tepat untuk operasi tunggal dan multi-bagian. Aplikasi Anda seharusnya tidak bergantung pada pustaka CloudHSM PKCS #11 untuk mengembalikan kesalahan dalam keadaan ini.

Masalah: Sesi Hanya Baca tidak didukung di SDK 5

- Masalah: SDK 5 tidak mendukung pembukaan sesi Read-Only dengan C_OpenSession
- Dampak: Jika Anda mencoba menelepon C_OpenSession tanpa memberikan CKF_RW_SESSION, panggilan akan gagal dengan kesalahan CKR_FUNCTION_FAILED.
- Solusi: Saat membuka sesi, Anda harus meneruskan CKF_SERIAL_SESSION | CKF_RW_SESSION bendera ke panggilan fungsi C_OpenSession

Masalah: file **cryptoki.h** header hanya untuk Windows

- Masalah: Dengan AWS CloudHSM Client SDK 5 versi 5.0.0 hingga 5.4.0 di Linux, file header hanya /opt/cloudhsm/include/pkcs11/cryptoki.h kompatibel dengan sistem operasi Windows.
- Dampak: Anda mungkin mengalami masalah ketika mencoba memasukkan file header ini dalam aplikasi Anda pada sistem operasi berbasis Linux.
- Status resolusi: Upgrade ke AWS CloudHSM Client SDK 5 versi 5.4.1 atau lebih tinggi, yang mencakup versi Linux yang kompatibel dari file header ini.

Masalah yang diketahui untuk JCE SDK untuk AWS CloudHSM

Masalah berikut memengaruhi JCE SDK untuk AWS CloudHSM

Topik

- [Masalah: Ketika bekerja dengan pasangan kunci asimetris, Anda melihat kapasitas kunci ditempati bahkan ketika Anda tidak secara eksplisit membuat atau mengimpor kunci](#)
- [Masalah: JCE KeyStore hanya dibaca](#)
- [Masalah: Penyangga untuk enkripsi AES-GCM tidak dapat melebihi 16.000 byte](#)
- [Masalah: Derivasi kunci Elliptic-curve Diffie-Hellman \(ECDH\) dijalankan sebagian dalam HSM](#)
- [Masalah: KeyGenerator dan KeyAttribute salah menafsirkan parameter ukuran kunci sebagai jumlah byte, bukan bit](#)
- [Masalah: Klien SDK 5 melontarkan peringatan "Operasi akses reflektif ilegal telah terjadi"](#)
- [Masalah: Kumpulan sesi JCE habis](#)
- [Masalah: Kebocoran memori SDK 5 klien dengan operasi GetKey](#)

Masalah: Ketika bekerja dengan pasangan kunci asimetris, Anda melihat kapasitas kunci ditempati bahkan ketika Anda tidak secara eksplisit membuat atau mengimpor kunci

- Dampak: Masalah ini dapat HSMs menyebabkan Anda tiba-tiba kehabisan ruang kunci dan terjadi ketika aplikasi Anda menggunakan objek kunci JCE standar untuk operasi krypto, bukan objek CaviumKey. Bila Anda menggunakan objek kunci JCE standar, CaviumProvider secara implisit mengimpor kunci tersebut ke HSM sebagai kunci sesi dan tidak menghapus kunci ini sampai

aplikasi keluar. Akibatnya, kunci menumpuk saat aplikasi berjalan dan dapat HSMs menyebabkan Anda kehabisan ruang kunci kosong, sehingga membekukan aplikasi Anda.

- Pemecahan masalah: Saat menggunakan kelas `CaviumSignature`, kelas `CaviumCipher`, kelas `CaviumMac`, atau kelas `CaviumKeyAgreement`, Anda harus menyediakan kunci sebagai `CaviumKey` bukan objek kunci JCE standar.

Anda dapat secara manual mengonversi kunci normal ke `CaviumKey` menggunakan kelas [ImportKey](#), dan kemudian dapat secara manual menghapus kunci setelah operasi selesai.

- Status resolusi: Kami memperbarui `CaviumProvider` untuk benar mengelola impor implisit. Pembaruan akan diumumkan di halaman riwayat versi setelah tersedia.

Masalah: JCE KeyStore hanya dibaca

- Dampak: Anda tidak dapat menyimpan jenis objek yang tidak didukung oleh HSM di penyimpanan kunci JCE hari ini. Secara khusus, Anda tidak dapat menyimpan sertifikat di penyimpanan kunci. Hal ini menghalangi interoperabilitas dengan alat-alat seperti jarsigner, yang mengharapkan untuk menemukan sertifikat di penyimpanan kunci.
- Pemecahan masalah: Anda dapat mengerjakan ulang kode Anda untuk memuat sertifikat dari file lokal atau dari lokasi bucket S3, bukan dari penyimpanan kuncinya.
- Status resolusi: Anda dapat menggunakan AWS CloudHSM keystore untuk menyimpan sertifikat.

Masalah: Penyangga untuk enkripsi AES-GCM tidak dapat melebihi 16.000 byte

Enkripsi AES-GCM multi-bagian tidak didukung.

- Dampak: Anda tidak dapat menggunakan AES-GCM untuk mengenkripsi data yang lebih besar dari 16.000 byte.
- Pemecahan masalah: Anda bisa menggunakan mekanisme alternatif seperti AES-CBC atau Anda dapat membagi data menjadi beberapa potong dan mengenkripsi setiap potongnya secara terpisah. Jika Anda membagi data, Anda harus mengelola ciphertext yang dibagi dan dekripsinya. Karena FIPS mensyaratkan bahwa vektor inisialisasi (IV) untuk AES-GCM dihasilkan pada HSM, IV untuk setiap AES-GCM-encrypted bagian data akan berbeda.
- Status resolusi: Kami memperbaiki SDK gagal secara eksplisit jika penyangga data terlalu besar. Kami sedang mengevaluasi alternatif yang mendukung penyangga yang lebih besar tanpa mengandalkan enkripsi multi-bagian. Pembaruan akan diumumkan di forum AWS CloudHSM dan pada halaman riwayat versi.

Masalah: Derivasi kunci Eliptic-curve Diffie-Hellman (ECDH) dijalankan sebagian dalam HSM

Kunci privat EC Anda tetap berada dalam HSM setiap saat, tetapi proses derivasi kunci dilakukan dalam beberapa langkah. Akibatnya, hasil menengah dari setiap langkah tersedia pada klien.

Derivasi kunci ECDH sampel tersedia di [Sampel kode Java](#).

- Dampak: Klien SDK 3 menambahkan fungsionalitas ECDH ke JCE. Ketika Anda menggunakan `KeyAgreement` kelas untuk mendapatkan a `SecretKey`, pertama kali tersedia pada klien dan kemudian diimpor ke HSM. Sebuah handel kunci kemudian kembali ke aplikasi Anda.
- Solusi: Jika Anda menerapkan SSL/TLS Offload di AWS CloudHSM, batasan ini mungkin tidak menjadi masalah. Jika aplikasi Anda memerlukan kunci Anda untuk tetap berada dalam batas FIPS setiap saat, pertimbangkan untuk menggunakan protokol alternatif yang tidak bergantung pada derivasi kunci ECDH.
- Status resolusi: SDK 5.16 sekarang mendukung ECDH dengan Derivasi Kunci yang dilakukan sepenuhnya dalam HSM.

Masalah: `KeyGenerator` dan `KeyAttribute` salah menafsirkan parameter ukuran kunci sebagai jumlah byte, bukan bit

Saat membuat kunci menggunakan `init` fungsi [KeyGenerator kelas](#) atau `SIZE` atribut [AWS CloudHSM KeyAttribute enum](#), API salah mengharapakan argumen menjadi jumlah byte kunci, padahal seharusnya jumlah bit kunci.

- Dampak: Client SDK versi 5.4.0 hingga 5.4.2 salah mengharapakan ukuran kunci diberikan ke yang ditentukan sebagai byte. APIs
- Solusi: Konversi ukuran kunci dari bit ke byte sebelum menggunakan `KeyGenerator` kelas atau `KeyAttribute` enum untuk menghasilkan kunci menggunakan penyedia AWS CloudHSM JCE jika menggunakan Client SDK versi 5.4.0 hingga 5.4.2.
- Status resolusi: Tingkatkan versi SDK klien Anda ke 5.5.0 atau yang lebih baru, yang mencakup perbaikan untuk mengharapakan ukuran kunci dalam bit dengan benar saat menggunakan `KeyGenerator` kelas atau `KeyAttribute` enum untuk menghasilkan kunci.

Masalah: Klien SDK 5 melontarkan peringatan “Operasi akses reflektif ilegal telah terjadi”

Saat menggunakan Client SDK 5 dengan Java 11, CloudHSM menampilkan peringatan Java berikut:

```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by
  com.amazonaws.cloudhsm.jce.provider.CloudHsmKeyStore (file:/opt/cloudhsm/java/
cloudhsm-jce-5.6.0.jar) to field java.security .KeyStore.keyStoreSpi
WARNING: Please consider reporting this to the maintainers of
  com.amazonaws.cloudhsm.jce.provider.CloudHsmKeyStore
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective
  access operations
WARNING: All illegal access operations will be denied in a future release
```

Masalah ini diperbaiki di Client SDK versi 5.8 dan yang lebih baru.

Masalah: Kumpulan sesi JCE habis

Dampak: Anda mungkin tidak dapat melakukan operasi di JCE setelah melihat pesan berikut:

```
com.amazonaws.cloudhsm.jce.jni.exception.InternalException: There are too many
operations
happening at the same time: Reached max number of sessions in session pool: 1000
```

Solusi:

- Mulai ulang aplikasi JCE Anda jika Anda mengalami dampak.
- Saat melakukan operasi, Anda mungkin perlu menyelesaikan operasi JCE sebelum kehilangan referensi ke operasi.

Note

Tergantung pada operasi, metode penyelesaian mungkin diperlukan.

Operasi	Metode penyelesaian
Cipher	<code>doFinal()</code> dalam mode enkripsi atau dekripsi <code>wrap()</code> dalam mode bungkus <code>unwrap()</code> dalam mode buka bungkus
KeyAgreement	<code>generateSecret()</code> atau <code>generateSecret(String)</code>
KeyPairGenerator	<code>generateKeyPair()</code> , <code>genKeyPair()</code> , atau <code>reset()</code>
KeyStore	Tidak ada metode yang dibutuhkan
MAC	<code>doFinal()</code> atau <code>reset()</code>
MessageDigest	<code>digest()</code> atau <code>reset()</code>
SecretKeyFactory	Tidak ada metode yang dibutuhkan
SecureRandom	Tidak ada metode yang dibutuhkan
Tanda tangan	<code>sign()</code> dalam mode tanda <code>verify()</code> dalam mode verifikasi

Status resolusi: Kami telah menyelesaikan masalah ini di Client SDK 5.9.0 dan yang lebih baru. Untuk memperbaiki masalah ini, tingkatkan SDK Klien Anda ke salah satu versi ini.

Masalah: Kebocoran memori SDK 5 klien dengan operasi GetKey

- Dampak: `getKey` Operasi API memiliki kebocoran memori di JCE di Client SDK versi 5.10.0 dan sebelumnya. Jika Anda menggunakan `getKey` API beberapa kali dalam aplikasi Anda, itu akan menyebabkan peningkatan pertumbuhan memori dan akibatnya meningkatkan jejak memori dalam

aplikasi Anda. Seiring waktu ini dapat menyebabkan kesalahan pelambatan atau mengharuskan aplikasi dimulai ulang.

- Solusi: Kami merekomendasikan untuk meningkatkan ke Client SDK 5.11.0. Jika ini tidak dapat dilakukan, kami sarankan untuk tidak memanggil `getKey` API beberapa kali dalam aplikasi Anda. Sebaliknya, gunakan kembali kunci yang dikembalikan sebelumnya dari `getKey` operasi sebelumnya sebanyak mungkin.
- Status resolusi: Tingkatkan versi SDK klien Anda ke 5.11.0 atau yang lebih baru, yang mencakup perbaikan untuk masalah ini.

Masalah yang diketahui untuk OpenSSL Dynamic Engine untuk AWS CloudHSM

Ini adalah masalah yang diketahui untuk OpenSSL Dynamic Engine untuk AWS CloudHSM

Topik

- [Masalah: Anda tidak dapat menginstal AWS CloudHSM OpenSSL Dynamic Engine di RHEL 6 dan Cent OS6](#)
- [Masalah: Hanya pembongkaran RSA ke HSM yang didukung secara default](#)
- [Masalah: Enkripsi dan dekripsi RSA dengan bantalan OAEP menggunakan kunci pada HSM tidak didukung](#)
- [Masalah: Hanya pembuatan kunci privat RSA dan kunci ECC yang dibongkar ke HSM](#)
- [Masalah: Anda tidak dapat menginstal OpenSSL Dynamic Engine untuk SDK Klien 3 pada RHEL 8, CentOS 8, atau Ubuntu 18.04 LTS](#)
- [Masalah: SHA-1 Menandatangani dan Verifikasi penghentian pada RHEL 9 \(9.2+\)](#)
- [Masalah: AWS CloudHSM OpenSSL Dynamic Engine tidak kompatibel dengan penyedia FIPS untuk OpenSSL v3.x](#)
- [Masalah: SSL/TLS pembongkaran gagal dengan rangkaian sandi ECDSA di TLS 1.0 dan TLS 1.1 dimulai dengan SDK 5.16](#)

Masalah: Anda tidak dapat menginstal AWS CloudHSM OpenSSL Dynamic Engine di RHEL 6 dan Cent OS6

- Dampak: OpenSSL Dynamic Engine hanya [mendukung OpenSSL 1.0.2 \[f+\]](#). Secara default, RHEL 6 dan CentOS 6 hadir dengan OpenSSL 1.0.1.

- Pemecahan masalah: Tingkatkan pustaka OpenSSL pada RHEL 6 dan CentOS 6 ke versi 1.0.2[f+].

Masalah: Hanya pembongkaran RSA ke HSM yang didukung secara default

- Dampak: Untuk memaksimalkan performa, SDK tidak dikonfigurasi untuk membongkar fungsi tambahan seperti pembuatan nomor acak atau operasi EC-DH.
- Pemecahan masalah: Silakan hubungi kami melalui kasus dukungan jika Anda perlu membongkar operasi tambahan.
- Status resolusi: Kami menambahkan dukungan ke SDK untuk mengatur konfigurasi opsi pembongkaran melalui file konfigurasi. Pembaruan akan diumumkan di halaman riwayat versi setelah tersedia.

Masalah: Enkripsi dan dekripsi RSA dengan bantalan OAEP menggunakan kunci pada HSM tidak didukung

- Dampak: Setiap panggilan ke enkripsi dan dekripsi RSA dengan padding OAEP gagal dengan kesalahan. divide-by-zero Hal ini terjadi karena mesin dinamis OpenSSL memanggil operasi secara lokal menggunakan file PEM palsu, bukan membongkar operasi ke HSM.
- Pemecahan masalah: Anda dapat melakukan prosedur ini dengan menggunakan [Pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5](#) atau [Penyedia JCE untuk AWS CloudHSM Klien SDK 5](#).
- Status resolusi: Kami menambahkan dukungan ke SDK untuk membongkar dengan benar operasi ini. Pembaruan akan diumumkan di halaman riwayat versi setelah tersedia.

Masalah: Hanya pembuatan kunci privat RSA dan kunci ECC yang dibongkar ke HSM

Untuk jenis kunci lainnya, mesin AWS CloudHSM OpenSSL tidak digunakan untuk pemrosesan panggilan. Mesin OpenSSL lokal digunakan sebagai gantinya. Ini menghasilkan kunci secara lokal dalam perangkat lunak.

- Dampak: Karena failover diam, ada tidak ada indikasi bahwa Anda belum menerima kunci yang aman dihasilkan pada HSM. Anda akan melihat jejak output yang berisi string " ++ ++++" jika kunci dihasilkan secara lokal oleh OpenSSL dalam perangkat lunak. Jejak ini tidak ada saat operasi dibongkar ke HSM. Karena kunci tidak dihasilkan atau disimpan di HSM, kunci akan tidak tersedia untuk penggunaan di masa mendatang.

- Pemecahan masalah: Hanya gunakan mesin OpenSSL untuk jenis kunci yang didukungnya. Untuk semua jenis kunci lainnya, gunakan PKCS #11 atau JCE dalam aplikasi, atau gunakan `key_mgmt_util` di CLI.

Masalah: Anda tidak dapat menginstal OpenSSL Dynamic Engine untuk SDK Klien 3 pada RHEL 8, CentOS 8, atau Ubuntu 18.04 LTS

- Dampak: Secara default, RHEL 8, CentOS 8, dan Ubuntu 18.04 LTS mengirimkan versi OpenSSL yang tidak kompatibel dengan OpenSSL Dynamic Engine untuk SDK Klien 3.
- Pemecahan masalah: Gunakan platform Linux yang menyediakan dukungan untuk OpenSSL Dynamic Engine. Untuk informasi selengkapnya tentang platform yang didukung, lihat [Platform yang Didukung](#).
- Status resolusi: AWS CloudHSM mendukung platform ini dengan OpenSSL Dynamic Engine untuk Client SDK 5. Untuk informasi selengkapnya, lihat [Platform yang Didukung](#) dan [OpenSSL Dynamic Engine](#).

Masalah: SHA-1 Menandatangani dan Verifikasi penghentian pada RHEL 9 (9.2+)

- Dampak: Penggunaan intisari pesan SHA-1 untuk tujuan kriptografi tidak digunakan lagi di RHEL 9 (9.2+). Akibatnya, tanda tangani dan verifikasi operasi dengan SHA-1 menggunakan OpenSSL Dynamic Engine akan gagal.
- Solusi: [Jika skenario Anda memerlukan penggunaan SHA-1 untuk tanda tangan kriptografi yang signing/verifying ada atau pihak ketiga, lihat Meningkatkan Keamanan RHEL: Memahami penghentian SHA-1 pada RHEL 9 \(9.2+\) dan RHEL 9 \(9.2+\) Catatan Rilis untuk detail lebih lanjut.](#)

Masalah: AWS CloudHSM OpenSSL Dynamic Engine tidak kompatibel dengan penyedia FIPS untuk OpenSSL v3.x

- Dampak: Anda akan menerima kesalahan jika Anda mencoba menggunakan AWS CloudHSM OpenSSL Dynamic Engine ketika penyedia FIPS diaktifkan untuk OpenSSL versi 3.x.
- Solusi: Untuk menggunakan OpenSSL Dynamic Engine dengan AWS CloudHSM OpenSSL versi 3.x, pastikan bahwa penyedia “default” dikonfigurasi. Baca selengkapnya tentang penyedia default di Situs Web [OpenSSL](#).

Masalah: SSL/TLS pembongkaran gagal dengan rangkaian sandi ECDSA di TLS 1.0 dan TLS 1.1 dimulai dengan SDK 5.16

- Dampak: Upaya koneksi menggunakan TLS 1.0 atau TLS 1.1 gagal karena versi ini menggunakan SHA-1 untuk penandatanganan, yang tidak memenuhi persyaratan [FIPS](#) 186-5.
- Solusi: Jika Anda tidak dapat segera memutakhirkan versi TLS, Anda dapat bermigrasi ke kluster non-FIPS, yang tidak menerapkan persyaratan kekuatan hash. Namun, kami menyarankan untuk meningkatkan ke TLS 1.2 atau TLS 1.3 untuk mempertahankan kepatuhan FIPS dan praktik terbaik keamanan.
- Resolusi: Tingkatkan implementasi Anda untuk menggunakan TLS 1.2 atau TLS 1.3. Internet Engineering Task Force (IETF) telah [menghentikan TLS](#) 1.0 dan TLS 1.1 karena masalah keamanan.

Masalah yang diketahui untuk Penyedia Penyimpanan Kunci (KSP) untuk AWS CloudHSM

Ini adalah masalah yang diketahui untuk Penyedia Penyimpanan Kunci (KSP) untuk AWS CloudHSM.

Topik

- [Masalah: Verifikasi penyimpanan sertifikat gagal](#)
- [Masalah: Inkonsistensi nama kontainer di penyimpanan sertifikat saat menggunakan mode SDK3 kompatibilitas untuk Client SDK 5](#)

Masalah: Verifikasi penyimpanan sertifikat gagal

Saat menggunakan Client SDK versi 5.14 dan 5.15, panggilan akan `certutil -store my CERTIFICATE_SERIAL_NUMBER` menimbulkan kesalahan berikut:

```
ERROR: Could not verify certificate public key against private key
```

- Dampak: Anda tidak dapat menggunakan `certutil` untuk memvalidasi penyimpanan sertifikat yang dibuat dengan Client SDK 5.
- Solusi: Validasi key pair yang terkait dengan sertifikat dengan menandatangani file menggunakan kunci pribadi dan memverifikasi tanda tangan menggunakan kunci publik. Ini dapat dilakukan

dengan menggunakan Microsoft SignTool dengan mengikuti langkah-langkah yang disediakan [di sini](#).

- Status Resolusi: Kami sedang berupaya menambahkan dukungan untuk memverifikasi sertifikat menggunakan `certutil`. Pembaruan akan diumumkan di halaman riwayat versi setelah tersedia.

Masalah: Inkonsistensi nama kontainer di penyimpanan sertifikat saat menggunakan mode SDK3 kompatibilitas untuk Client SDK 5

Saat menggunakan `certutil -store my CERTIFICATE_SERIAL_NUMBER` perintah untuk melihat sertifikat yang file referensi kunci-dihasilkan menggunakan perintah [generate-file](#) di AWS CLI 5.16.0, kesalahan berikut terjadi:

```
ERROR: Container name inconsistent: CONTAINER_NAME
```

Kesalahan ini terjadi karena ada ketidakcocokan antara nama kontainer yang disimpan dalam sertifikat dan nama file referensi kunci yang dihasilkan oleh CloudHSM CLI.

- Dampak: Meskipun kesalahan ini, sertifikat dan kunci terkait tetap berfungsi penuh. Semua aplikasi yang menggunakan sertifikat ini akan terus bekerja secara normal.
- Solusi: Untuk mengatasi kesalahan ini, ganti nama file referensi kunci menjadi nama wadah Sederhana atau Unik. Lihat contoh output perintah berikut `certutil -store my`

```
Subject: CN=www.website.com, OU=Organizational-Unit, O=Organization, L=City, S=State, C=US
Non-root Certificate
Cert Hash(sha1): 1add52
Key Container = 7e3c-b2f5
Simple container name: tq-3daacd89
Unique container name: tq-3daacd89
ERROR: Container name inconsistent: 7e3c-b2f5
```

Secara default, file referensi kunci akan disimpan di `C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP\GlobalPartition`

1. Ubah nama file referensi kunci menjadi nama kontainer sederhana.
2. Perbaiki toko sertifikat dengan nama wadah kunci baru. Lihat langkah 12 hingga 14 di [Migrasi KSP](#) untuk detail selengkapnya.

- Status resolusi: Masalah ini telah diperbaiki di Client SDK versi 5.16.1. Untuk mengatasi masalah ini, tingkatkan SDK Klien Anda ke versi 5.16.1 atau yang lebih baru.

Masalah yang diketahui untuk Penyedia OpenSSL untuk AWS CloudHSM

Ini adalah masalah yang diketahui untuk Penyedia OpenSSL untuk AWS CloudHSM

Topik

- [Masalah: Kesalahan dalam OpenSSL CLI saat digunakan dengan Penyedia OpenSSL](#)

Masalah: Kesalahan dalam OpenSSL CLI saat digunakan dengan Penyedia OpenSSL

- Dampak: Integrasi dengan OpenSSL CLI saat ini tidak didukung oleh Penyedia OpenSSL. AWS CloudHSM Lihat [the section called “Offload di Linux dengan Penyedia OpenSSL”](#) untuk integrasi yang didukung.
- Resolusi: Gunakan integrasi yang didukung yang tercantum dalam [the section called “Offload di Linux dengan Penyedia OpenSSL”](#). Setiap pembaruan untuk dukungan OpenSSL CLI akan diumumkan di halaman riwayat versi.

Masalah yang diketahui untuk instans Amazon EC2 yang menjalankan Amazon Linux 2 dengan AWS CloudHSM

Masalah berikut berdampak AWS CloudHSM dan instans Amazon EC2 yang berjalan di Amazon Linux 2.

Masalah: Amazon Linux 2 versi 2018.07 menggunakan **ncurses** paket yang diperbarui (versi 6) yang saat ini tidak kompatibel dengan AWS CloudHSM SDKs

[Anda melihat kesalahan berikut dikembalikan saat menjalankan AWS CloudHSMcloudhsm_mgmt_util atau key_mgmt_util:](#)

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util: error while loading shared libraries:  
libncurses.so.5: cannot open shared object file: No such file or directory
```

- Dampak: Instans yang berjalan di Amazon Linux 2 versi 2018.07 tidak akan dapat menggunakan semua utilitas. AWS CloudHSM

- Pemecahan masalah: Keluarkan perintah berikut pada instans EC2 Amazon Linux 2 Anda untuk menginstal paket ncurses (versi 5):

```
sudo yum update && yum install ncurses-compat-libs
```

- Status resolusi: Masalah ini telah diselesaikan dalam rilis 1.1.2 AWS CloudHSM klien. Anda harus meningkatkan ke klien ini untuk mendapatkan manfaat dari perbaikan.

Masalah yang diketahui untuk mengintegrasikan aplikasi pihak ketiga dengan AWS CloudHSM

Masalah berikut berdampak AWS CloudHSM saat mengintegrasikan dengan aplikasi pihak ketiga.

Masalah: Klien SDK 3 tidak mendukung pengaturan Oracle atribut **CKA_MODIFIABLE** PKCS #11 selama pembuatan kunci master

Batas ini didefinisikan dalam pustaka PKCS #11. Untuk informasi selengkapnya, lihat anotasi 1 pada [Atribut PKCS #11 yang Didukung](#).

- Dampak: Pembuatan kunci utama Oracle gagal.
- Pemecahan masalah: Atur variabel lingkungan khusus `CLOUDHSM_IGNORE_CKA_MODIFIABLE_FALSE` ke `TRUE` saat membuat kunci utama baru. Variabel lingkungan ini hanya diperlukan untuk pembuatan kunci utama dan Anda tidak perlu menggunakan variabel lingkungan ini untuk hal lain. Misalnya, Anda akan menggunakan variabel ini untuk kunci utama pertama yang Anda buat dan kemudian Anda hanya akan menggunakan variabel lingkungan ini lagi jika Anda ingin memutar edisi kunci utama Anda. Untuk informasi selengkapnya, lihat [Hasilkan Kunci Enkripsi Utama TDE Oracle](#).
- Status resolusi: Kami meningkatkan firmware HSM untuk sepenuhnya mendukung atribut `CKA_MODIFIABLE`. Pembaruan akan diumumkan di AWS CloudHSM forum dan di halaman riwayat versi

Masalah yang diketahui untuk modifikasi AWS CloudHSM cluster

Masalah berikut berdampak pada pelanggan yang mencoba menggunakan API `modify-cluster` untuk mengubah tipe HSM cluster.

Topik

- [Masalah: Latensi login meningkat karena peningkatan PBKDF2 iterasi](#)
- [Masalah: Tidak dapat mengubah tipe HSM karena pembuatan kunci token](#)

Masalah: Latensi login meningkat karena peningkatan PBKDF2 iterasi

- Dampak: Cluster dengan sejumlah besar pengguna akan mengalami periode migrasi yang diperpanjang. Hal ini disebabkan perubahan dalam proses pemulihan cadangan yang melakukan PBKDF2 operasi per pengguna saat memulihkan cadangan hsm1.medium ke hsm2m.medium untuk pertama kalinya.
- Status resolusi: Jika Anda membuat instance hsm2m.medium baru atau bermigrasi ke hsm2m.medium dari hsm1.medium sebelum 20 Desember 2025, Anda perlu mengatur ulang kata sandi Anda untuk memanfaatkan peningkatan kinerja yang telah kami terapkan untuk operasi login. Lihat [perubahan-kata sandi untuk instruksi](#).

Masalah: Tidak dapat mengubah tipe HSM karena pembuatan kunci token

- Dampak: Pelanggan yang melakukan beban kerja berbasis kunci token tidak akan dapat memulai migrasi mereka. Hal ini dilakukan karena HSM akan ditempatkan ke dalam mode penulisan terbatas untuk mencegah skenario dataloss selama modifikasi tipe HSM.
- Solusi: Berhenti membuat dan menghapus kunci token dan kemudian tunggu 7 hari. Atau, silakan hubungi dukungan jika Anda
 - Tidak dapat menangani pemblokiran migrasi kunci token dan tidak dapat melakukan blue/green penerapan.
 - Dapat menangani pemblokiran operasi kunci token selama migrasi, tetapi tidak dapat menunggu periode 7 hari penuh.
- Status resolusi: Masalah ini telah diselesaikan. Pelanggan yang melakukan beban kerja berbasis kunci token sekarang dapat memulai migrasi. Pembuatan dan penghapusan kunci token akan diblokir selama migrasi.

Masalah kegagalan operasi yang diketahui menggunakan AWS CloudHSM klien versi 5.12.0 di hsm2.medium

Masalah berikut berdampak AWS CloudHSM saat menggunakan AWS CloudHSM klien versi 5.12.0

Masalah: Kesalahan selama operasi get-attribute

Jika Anda bermigrasi dari hsm1.medium ke hsm2m.medium dan menggunakan CloudHSM Client SDK 5.12.0, Anda dapat melihat kesalahan yang terkait dengan penanganan atribut.

Anda mungkin melihat pesan galat berikut di log klien: `Error in deserialization of data: Invalid integer conversion`

Dampak: Operasi di bawah ini akan gagal menggunakan versi klien 5.12.0

- Di PKCS #11 SDK, panggilan ke `C_` gagal `GetAttributeValue`
- Di CloudHSM CLI, perintah daftar kunci tidak menunjukkan atribut dalam output
- Di CloudHSM CLI, file penghasil kunci mungkin gagal untuk kunci yang dihasilkan menggunakan hsm1.medium

Resolusi: Kami menyarankan untuk memutakhirkan ke versi terbaru SDK yang menyelesaikan masalah ini.

AWS CloudHSM Kegagalan sinkronisasi kunci SDK 3 klien

Di Client SDK 3, jika sinkronisasi sisi klien gagal, AWS CloudHSM lakukan respons upaya terbaik untuk membersihkan kunci yang tidak diinginkan yang mungkin telah dibuat (dan sekarang tidak diinginkan). Proses ini melibatkan penghapusan materi kunci yang tidak diinginkan dengan segera atau menandai materi yang tidak diinginkan untuk penghapusan nanti. Dalam kedua kasus ini, resolusi tidak memerlukan tindakan apa pun dari Anda. Dalam kasus yang jarang terjadi yang AWS CloudHSM tidak dapat menghapus dan tidak dapat menandai materi kunci yang tidak diinginkan, Anda harus menghapus materi kunci.

Masalah: Anda mencoba pembuatan kunci token, mengimpor, atau membuka operasi dan melihat kesalahan yang menentukan kegagalan pada tombstone.

```
2018-12-24T18:28:54Z liquidSecurity ERR: print_node_ts_status:
[create_object_min_nodes]Key: 264617 failed to tombstone on node:1
```

Penyebab: tidak AWS CloudHSM berhasil menghapus dan menandai materi kunci yang tidak diinginkan.

Resolusi: HSM di kluster Anda berisi materi kunci yang tidak diinginkan yang tidak ditandai sebagai tidak diinginkan. Anda harus menghapus materi kunci secara manual. Untuk menghapus materi kunci

yang tidak diinginkan secara manual, gunakan `key_mgmt_util` (KMU) atau API dari pustaka PKCS #11 atau penyedia JCE. Untuk informasi selengkapnya, lihat [deleteKey](#) atau [Klien SDKs](#).

Untuk membuat kunci token lebih tahan lama, AWS CloudHSM gagal operasi pembuatan kunci yang tidak berhasil pada jumlah minimum yang HSMs ditentukan dalam pengaturan sinkronisasi sisi klien. Untuk informasi selengkapnya, lihat [Sinkronisasi Kunci dalam AWS CloudHSM](#).

AWS CloudHSM Klien SDK 3 memverifikasi kinerja HSM dengan alat pkpspeed

Topik ini menjelaskan cara memverifikasi kinerja modul keamanan AWS CloudHSM perangkat keras (HSM) dengan Client SDK 3.

Untuk memverifikasi kinerja HSMs di AWS CloudHSM cluster Anda, Anda dapat menggunakan alat `pkpspeed` (Linux) atau `pkpspeed_blocking` (Windows) yang disertakan dengan Client SDK 3. Alat `pkpspeed` mengeksekusi dalam kondisi ideal dan langsung memanggil HSM untuk menjalankan operasi tanpa melalui SDK seperti PKCS11. Kami merekomendasikan pengujian beban aplikasi Anda secara independen untuk menentukan kebutuhan penskalaan Anda. Kami tidak menyarankan menjalankan tes berikut: Acak (I), ModExp (R), dan titik EC mul (Y).

Untuk informasi selengkapnya tentang menginstal klien di instans EC2 Linux, lihat [Instal dan konfigurasi AWS CloudHSM klien untuk CMU \(Linux\)](#). Untuk informasi selengkapnya tentang menginstal klien di instans Windows, lihat [Instal dan konfigurasi AWS CloudHSM klien untuk CMU \(Windows\)](#).

Setelah Anda menginstal dan mengkonfigurasi AWS CloudHSM klien, jalankan perintah berikut untuk memulainya.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Untuk klien Windows 1.1.2+:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Untuk klien Windows 1.1.1 dan yang lebih lama:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Jika Anda telah menginstal perangkat lunak klien, Anda mungkin perlu mengunduh dan menginstal versi terbaru untuk mendapatkan pkpspeed. Anda dapat menemukan alat pkpspeed di /opt/cloudhsm/bin/pkpspeed di Linux atau C:\Program Files\Amazon\CloudHSM\ di Windows.

Untuk menggunakan pkpspeed, jalankan perintah pkpspeed atau pkpspeed_blocking.exe, menentukan nama pengguna dan kata sandi pengguna krypto (CU) pada HSM. Kemudian tetapkan pilihan untuk digunakan sambil mempertimbangkan rekomendasi berikut.

Rekomendasi tes

- Untuk menguji performa tanda tangan RSA dan memverifikasi operasi, pilih cipher RSA_CRT di Linux atau opsi B di Windows. Jangan pilih RSA (opsi A di Windows). Cipher setara, tapi RSA_CRT dioptimalkan untuk performa.
- Mulailah dengan sejumlah kecil thread. Untuk pengujian performa AES, satu thread biasanya cukup untuk menunjukkan performa maksimum. Untuk menguji performa RSA (RSA_CRT), tiga atau empat thread biasanya cukup.

Opsi yang dapat dikonfigurasi untuk alat pkpspeed

- Mode FIPS: AWS CloudHSM selalu dalam mode FIPS (Lihat [AWS CloudHSM FAQs](#) untuk detailnya). Ini dapat diverifikasi dengan menggunakan alat CLI seperti yang didokumentasikan dalam Panduan AWS CloudHSM Pengguna dan menjalankan [Dapatkan informasi perangkat keras untuk setiap HSM dalam AWS CloudHSM cluster dengan CMU](#) perintah yang akan menunjukkan status mode FIPS.
- Jenis pengujian (pemblokiran versus non-pemblokiran): Ini menentukan bagaimana operasi dilakukan dengan cara berulir. Anda kemungkinan besar akan mendapatkan angka yang lebih baik menggunakan non-blocking. Ini karena mereka menggunakan thread dan konkurensi.
- Jumlah utas: Jumlah utas untuk menjalankan pengujian.
- Waktu dalam detik untuk menjalankan tes (maks = 600): pkpspeed menghasilkan hasil yang diukur dalam “operasi/detik” dan melaporkan nilai ini untuk setiap detik pengujian dijalankan. Misalnya, jika pengujian dijalankan selama 5 detik, output mungkin terlihat seperti nilai sampel berikut:
 - OPERATIONS/second 821/1
 - OPERATIONS/second 833/1
 - OPERATIONS/second 845/1
 - OPERATIONS/second 835/1
 - OPERATIONS/second 837/1

Tes yang dapat dijalankan dengan alat pkpspeed

- AES GCM: Menguji enkripsi mode AES GCM.
- Dasar 3DES CBC: Menguji enkripsi mode CBC 3DES. Lihat catatan [1](#) di bawah untuk perubahan yang akan datang.

- AES Dasar: Menguji enkripsi AES. CBC/ECB
- Digest: Menguji intisari hash.
- Tanda ECDSA: Menguji tanda ECDSA.
- ECDSA Verifikasi: Menguji verifikasi ECDSA.
- FIPS Random: Menguji pembuatan nomor acak yang sesuai dengan FIPS (Catatan: ini hanya dapat digunakan dalam mode pemblokiran).
- HMAC: Menguji HMAC.
- Acak: Tes ini tidak relevan karena kami menggunakan FIPS 140-2 HSM.
- RSA non-CRT versus RSA_CRT: Menguji tanda RSA dan memverifikasi operasi.
- RSA OAEP Enc: Menguji enkripsi RSA OAEP.
- RSA OAEP Des: Menguji dekripsi RSA OAEP.
- RSA private dec non-CRT: Menguji enkripsi kunci pribadi RSA (tidak dioptimalkan).
- Kunci pribadi RSA dec CRT: Menguji enkripsi kunci pribadi RSA (dioptimalkan).
- Tanda PSS RSA: Menguji tanda PSS RSA.
- RSA PSS Verifikasi: Tes RSA PSS memverifikasi.
- Enc kunci publik RSA: Menguji enkripsi kunci publik RSA.

Enkripsi kunci publik RSA, dekripsi pribadi RSA non-CRT, dan dekripsi kunci pribadi RSA CRT juga akan meminta pengguna untuk menjawab yang berikut:

```
Do you want to use static key [y/n]
```

Jika y dimasukkan, kunci pra-komputasi diimpor ke HSM.

Jika n dimasukkan, kunci baru dihasilkan.

[1] Sesuai dengan panduan NIST, ini tidak diizinkan untuk cluster dalam mode FIPS setelah 2023. Untuk cluster dalam mode non-FIPS, masih diperbolehkan setelah 2023. Lihat [Kepatuhan FIPS 140: Penutupan Mekanisme 2024](#) untuk detail.

Contoh

Contoh berikut menunjukkan pilihan yang dapat Anda pilih dengan `pkpspeed` (Linux) atau `pkpspeed_blocking` (Windows) untuk menguji performa HSM untuk operasi RSA dan AES.

Example— Menggunakan pkpspeed untuk menguji performa RSA

Anda dapat menjalankan contoh ini pada Windows, Linux, dan sistem operasi yang kompatibel.

Linux

Gunakan petunjuk ini untuk Linux dan sistem operasi yang kompatibel.

```
/opt/cloudhsm/bin/pkpspeed -s CU user name -p password

SDK Version: 2.03

    Available Ciphers:
        AES_128
        AES_256
        3DES
        RSA (non-CRT. modulus size can be 2048/3072)
        RSA_CRT (same as RSA)
For RSA, Exponent will be 65537

Current FIPS mode is: 00002
Enter the number of thread [1-10]: 3
Enter the cipher: RSA_CRT
Enter modulus length: 2048
Enter time duration in Secs: 60
Starting non-blocking speed test using data length of 245 bytes...
[Test duration is 60 seconds]

Do you want to use static key[y/n] (Make sure that KEK is available)?n
```

Windows

```
c:\Program Files\Amazon\CloudHSM>pkpspeed_blocking.exe -s CU user name -p password

Please select the test you want to run

RSA non-CRT----->A
RSA CRT----->B
Basic 3DES CBC----->C
Basic AES----->D
FIPS Random----->H
Random----->I
AES GCM ----->K
```

```

eXit----->X
B

Running 4 threads for 25 sec

Enter mod size(2048/3072):2048
Do you want to use Token key[y/n]n
Do you want to use static key[y/n] (Make sure that KEK is available)? n
OPERATIONS/second      821/1
OPERATIONS/second      833/1
OPERATIONS/second      845/1
OPERATIONS/second      835/1
OPERATIONS/second      837/1
OPERATIONS/second      836/1
OPERATIONS/second      837/1
OPERATIONS/second      849/1
OPERATIONS/second      841/1
OPERATIONS/second      856/1
OPERATIONS/second      841/1
OPERATIONS/second      847/1
OPERATIONS/second      838/1
OPERATIONS/second      843/1
OPERATIONS/second      852/1
OPERATIONS/second      837/

```

Example— Menggunakan pkpspeed untuk menguji performa AES

Linux

Gunakan petunjuk ini untuk Linux dan sistem operasi yang kompatibel.

```
/opt/cloudhsm/bin/pkpspeed -s <CU user name> -p <password>
```

```
SDK Version: 2.03
```

```
Available Ciphers:
```

```
AES_128
```

```
AES_256
```

```
3DES
```

```
RSA (non-CRT. modulus size can be 2048/3072)
```

```
RSA_CRT (same as RSA)
```

```
For RSA, Exponent will be 65537
```

```

Current FIPS mode is: 00000002
Enter the number of thread [1-10]: 1
Enter the cipher: AES_256
Enter the data size [1-16200]: 8192
Enter time duration in Secs: 60
Starting non-blocking speed test using data length of 8192 bytes...

```

Windows

```

c:\Program Files\Amazon\CloudHSM>pkpspeed_blocking.exe -s CU user name -p password
login as USER
Initializing Cfm2 library
    SDK Version: 2.03

Current FIPS mode is: 00000002
Please enter the number of threads [MAX=400] : 1
Please enter the time in seconds to run the test [MAX=600]: 20

Please select the test you want to run

RSA non-CRT----->A
RSA CRT----->B
Basic 3DES CBC----->C
Basic AES----->D
FIPS Random----->H
Random----->I
AES GCM ----->K

eXit----->X
D

Running 1 threads for 20 sec

Enter the key size(128/192/256):256
Enter the size of the packet in bytes[1-16200]:8192
OPERATIONS/second          9/1
OPERATIONS/second          10/1
OPERATIONS/second          11/1
OPERATIONS/second          10/1
OPERATIONS/second          10/1
OPERATIONS/second          10/...

```

AWS CloudHSM Pengguna SDK 5 klien berisi nilai yang tidak konsisten

`user list` perintah di AWS CloudHSM Client SDK 5 mengembalikan daftar semua pengguna, dan properti pengguna, di cluster Anda. Jika salah satu properti pengguna memiliki nilai "tidak konsisten", pengguna ini tidak disinkronkan di seluruh cluster Anda. Ini berarti bahwa pengguna ada dengan properti yang berbeda pada yang berbeda HSMs di cluster. Berdasarkan properti mana yang tidak konsisten, langkah-langkah perbaikan yang berbeda dapat diambil.

Tabel berikut mencakup langkah-langkah untuk mengatasi ketidakkonsistenan untuk satu pengguna. Jika satu pengguna memiliki beberapa inkonsistensi, selesaikan dengan mengikuti langkah-langkah ini dari atas ke bawah. Jika ada beberapa pengguna dengan inkonsistensi, kerjakan daftar ini untuk setiap pengguna, selesaikan sepenuhnya inkonsistensi untuk pengguna tersebut sebelum melanjutkan ke yang berikutnya.

Note

Untuk melakukan langkah-langkah ini, idealnya Anda harus masuk sebagai admin. Jika akun admin Anda tidak konsisten, lakukan langkah-langkah ini masuk dengan admin dan ulangi langkah-langkahnya hingga semua properti konsisten. Setelah akun admin Anda konsisten, Anda dapat melanjutkan untuk menggunakan admin tersebut untuk menyinkronkan pengguna lain di cluster.

Properti yang tidak konsisten	Contoh output dari daftar pengguna	Implikasi	Metode pemulihan
"Peran" pengguna "tidak konsisten"	<pre>{ "username": "test_user", "role": "inconsistent ", "locked": "false", "mfa": [],</pre>	Pengguna ini adalah CryptoUser pada beberapa HSMs, dan Admin di sisi lain HSMs. Ini dapat terjadi jika dua SDKs upaya untuk membuat pengguna yang sama, pada saat yang sama,	1. Login sebagai admin 2. Hapus pengguna pada semua HSMs: <pre>user delete --username <user's name> -- role admin</pre>

Properti yang tidak konsisten	Contoh output dari daftar pengguna	Implikasi	Metode pemulihan
	<pre>"cluster-coverage": "full" }</pre>	dengan peran yang berbeda. Anda harus menghapus pengguna ini, dan membuatnya kembali dengan peran yang diinginkan.	user delete --username <user's name> -- role crypto-user 3. Buat pengguna dengan peran yang diinginkan: user create --username <user's name> --role <desired role>

Properti yang tidak konsisten	Contoh output dari daftar pengguna	Implikasi	Metode pemulihan
"Cakupan cluster" pengguna "tidak konsisten"	<pre>{ "username": "test_user", "role": "crypto-user", "locked": "false", "mfa": [], "cluster-coverage": "inconsistent " }</pre>	Pengguna ini ada pada subset HSMs di cluster. Hal ini dapat terjadi jika sebagian berhasil, atau jika user create sebagian berhasil. user delete Anda harus menyelesaikan operasi sebelumnya, baik membuat atau menghapus pengguna ini dari cluster Anda.	Jika pengguna seharusnya tidak ada, ikuti langkah-langkah ini: 1. Login sebagai admin 2. Jalankan perintah ini: <pre>user delete -- username<user's name> --role admin</pre> 3. Sekarang, jalankan perintah berikut: <pre>user delete -- username<user's name> --role crypto-user</pre> Jika pengguna harus ada, ikuti langkah-langkah ini: 1. Login sebagai admin 2. Jalankan perintah berikut: <pre>user create --username <user's name></pre>

Properti yang tidak konsisten	Contoh output dari daftar pengguna	Implikasi	Metode pemulihan
			--role < desired role >

Properti yang tidak konsisten	Contoh output dari daftar pengguna	Implikasi	Metode pemulihan
Parameter “terkunci” pengguna “tidak konsisten” atau “benar”	<pre>{ "username": "test_user", "role": "crypto-user", "locked" : inconsistent , "mfa": [], "cluster-coverage": "full" }</pre>	Pengguna ini dikunci pada subset. HSMs Ini bisa terjadi jika pengguna menggunakan kata sandi yang salah dan hanya terhubung ke subset HSMs di cluster. Anda harus mengubah kredensi pengguna agar konsisten di seluruh cluster.	Jika pengguna mengaktifkan MFA, ikuti langkah-langkah ini: 1. Login sebagai admin 2. Jalankan perintah berikut untuk menonaktifkan MFA sementara: <pre>user change-mfa token-sig n --username <user's name> --role <desired role> --disable</pre> 3. Ubah kata sandi pengguna sehingga mereka dapat masuk ke semua HSMs: <pre>user change-password --username <user's name> --role <desired role></pre> Jika MFA harus aktif untuk pengguna, ikuti langkah-langkah ini:

Properti yang tidak konsisten	Contoh output dari daftar pengguna	Implikasi	Metode pemulihan
			1. Minta pengguna masuk dan aktifkan kembali MFA (ini akan mengharuskan mereka untuk menandatangani token dan memberikan kunci publik mereka dalam file PEM): user change-mfa token-sign --username <user's name> --role <desired role> —token <File>

Properti yang tidak konsisten	Contoh output dari daftar pengguna	Implikasi	Metode pemulihan
Status MFA “tidak konsisten”	<pre>{ "username": "test_user", "role": "crypto-u ser", "locked": "false", "mfa": [{ "strategy": "token-sign", "status": "inconsistent " }], "cluster- coverage": "full" }</pre>	Pengguna ini memiliki bendera MFA yang berbeda pada yang berbeda HSMs di cluster. Hal ini dapat terjadi jika operasi MFA hanya selesai pada subset. HSMs Anda harus mengatur ulang kata sandi pengguna, dan memungkinkan mereka untuk mengaktifkan kembali MFA.	Jika pengguna mengaktifkan MFA, ikuti langkah-langkah ini: 1. Login sebagai admin 2. Jalankan perintah berikut untuk menonaktifkan MFA sementara: <pre>user change- mfa token-sig n --username <user's name> --role <desired role> --disable</pre> 3. Anda juga perlu mengubah kata sandi pengguna sehingga mereka dapat masuk ke semua HSMs: <pre>user change-pa ssword --usernam e <user's name> --role <desired role></pre> Jika MFA harus aktif untuk pengguna, ikuti langkah-langkah ini:

Properti yang tidak konsisten	Contoh output dari daftar pengguna	Implikasi	Metode pemulihan
			1. Minta pengguna masuk dan aktifkan kembali MFA (ini akan mengharuskan mereka untuk menandatangani token dan memberikan kunci publik mereka dalam file PEM): <pre> user change- mfa token-sign --username <user's name> --role <desired role> --token <File> </pre>

AWS CloudHSM Kegagalan replikasi pengguna SDK 5 klien

`user replicate` Perintah di CloudHSM CLI mereplikasi pengguna di antara kluster CloudHSM yang dikloning. AWS Panduan ini membahas kegagalan karena ketidakkonsistenan pengguna dalam kluster sumber atau antara kluster sumber dan tujuan. Replikasi pengguna memverifikasi bahwa pengguna konsisten dengan memeriksa atribut berikut:

- Peran Pengguna
- Status Kunci Akun
- Status Kuorum
- Status Multi-Factor Authentication (MFA)

Masalah: Pengguna yang dipilih tidak disinkronkan di seluruh cluster

Proses replikasi pengguna memeriksa sinkronisasi pengguna di seluruh cluster sumber. Jika atribut pengguna memiliki nilai “tidak konsisten”, ini berarti pengguna tidak disinkronkan di seluruh cluster. Replikasi pengguna gagal dengan pesan kesalahan berikut:

```
{
  "error_code": 1,
  "data": "Specified user is inconsistent across the cluster"
}
```

Untuk memeriksa desinkronisasi pengguna di cluster sumber:

- Jalankan `user list` perintah di CloudHSM CLI.

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "example-inconsistent-user",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "inconsistent"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
```

```

    "quorum": [],
    "cluster-coverage": "full"
  }
]
}
}

```

Resolusi: Sinkronisasi atribut pengguna di seluruh cluster sumber

- Untuk menyinkronkan informasi pengguna di seluruh cluster sumber, lihat yang berikut ini: [AWS CloudHSM Pengguna SDK 5 klien berisi nilai yang tidak konsisten](#)

Masalah: Pengguna ada di cluster tujuan dengan atribut yang berbeda

Jika pengguna sudah ada dengan referensi yang sama ada di satu atau lebih HSMs di cluster tujuan tetapi memiliki atribut pengguna yang berbeda, kesalahan berikut dapat terjadi:

```

{
  "error_code": 1,
  "data": "User replicate failed on 1 of 3 connections"
}

```

Resolusi

1. Tentukan versi pengguna mana yang harus disimpan.
2. Hapus pengguna yang tidak diinginkan di cluster appropriate dengan menjalankan perintah. `user delete` Untuk informasi selengkapnya, lihat [Hapus AWS CloudHSM pengguna dengan CloudHSM CLI](#).
3. Replikasi pengguna dengan menjalankan `user replicate` perintah.

Masalah: Replikasi pengguna dari hsm2m.medium ke hsm1.medium gagal

Replikasi pengguna dari hsm2m.medium ke hsm1.medium tidak didukung. Jika mereplikasi pengguna dari cluster sumber hsm2m.medium ke cluster tujuan hsm1.medium, kesalahan berikut akan terjadi:

```

{

```

```
"error_code": 1,
"data": "User replicate failed on 1 of 1 connections"
}
```

Resolusi

- Gunakan [manajemen pengguna](#) dengan CloudHSM CLI untuk membuat ulang pengguna yang hilang secara manual.

AWS CloudHSM Kegagalan replikasi kunci SDK 5 klien

`key replicate` Perintah di CloudHSM CLI mereplikasi kunci dari AWS CloudHSM cluster sumber ke cluster tujuan. AWS CloudHSM Panduan ini membahas kegagalan yang disebabkan oleh ketidakkonsistenan dalam cluster sumber atau antara kluster sumber dan tujuan.

Masalah: Kunci yang dipilih tidak disinkronkan di seluruh cluster

Proses replikasi kunci memeriksa sinkronisasi kunci di seluruh cluster sumber. Jika ada informasi atau atribut kunci yang memiliki nilai “tidak konsisten”, ini berarti kunci tidak disinkronkan di seluruh cluster. Replikasi kunci gagal dengan pesan kesalahan berikut:

```
{
  "error_code": 1,
  "data": "The selected key is not synchronized throughout the cluster"
}
```

Untuk memeriksa desinkronisasi kunci di cluster sumber:

- Jalankan `key list` perintah di CloudHSM CLI.
- Gunakan `--filter` bendera untuk menentukan kunci.
- Tambahkan `--verbose` bendera untuk melihat output lengkap dengan informasi cakupan utama.

```
aws-cloudhsm > key list --filter attr.label=example-desynchronized-key-label --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
```

```

{
  "key-reference": "0x0000000000048000f",
  "key-info": {
    "key-owners": [
      {
        "username": "cu1",
        "key-coverage": "full"
      }
    ],
    "shared-users": [],
    "key-quorum-values": {
      "manage-key-quorum-value": 0,
      "use-key-quorum-value": 0
    },
    "cluster-coverage": "full"
  },
  "attributes": {
    "key-type": "aes",
    "label": "example-desynchronized-key-label",
    "id": "0x",
    "check-value": "0xbe79db",
    "class": "secret-key",
    "encrypt": false,
    "decrypt": false,
    "token": true,
    "always-sensitive": true,
    "derive": false,
    "destroyable": true,
    "extractable": true,
    "local": true,
    "modifiable": true,
    "never-extractable": false,
    "private": true,
    "sensitive": true,
    "sign": "inconsistent",
    "trusted": false,
    "unwrap": false,
    "verify": true,
    "wrap": false,
    "wrap-with-trusted": false,
    "key-length-bytes": 16
  }
}
],

```



```
"total_key_count": 1,  
"returned_key_count": 1  
}  
}
```

Resolusi: Sinkronisasi informasi dan atribut utama di seluruh cluster sumber

Untuk menyinkronkan informasi kunci dan atribut di seluruh cluster sumber:

1. Untuk atribut kunci yang tidak konsisten: Gunakan `key set-attribute` perintah untuk mengatur atribut yang diinginkan untuk kunci tertentu.
2. Untuk cakupan pengguna bersama yang tidak konsisten: Gunakan `key unshare` perintah `key share` atau untuk menyesuaikan berbagi kunci dengan pengguna yang diinginkan.

Masalah: Kunci dengan referensi yang sama ada di cluster tujuan dengan informasi atau atribut yang berbeda

Jika kunci dengan referensi yang sama ada di cluster tujuan tetapi memiliki informasi atau atribut yang berbeda, kesalahan berikut dapat terjadi:

```
{  
  "error_code": 1,  
  "data": "Key replicate failed on 1 of 3 connections"  
}
```

Resolusi

1. Tentukan versi kunci mana yang harus disimpan.
2. Hapus versi kunci yang tidak diinginkan menggunakan `key delete` perintah di cluster yang sesuai.
3. Replikasi kunci dari cluster yang memiliki versi yang benar.

AWS CloudHSM kesalahan terlihat selama pemeriksaan ketersediaan kunci

Masalah: Modul keamanan AWS CloudHSM perangkat keras (HSM) mengembalikan kesalahan berikut:

```
Key <KEY HANDLE> does not meet the availability requirements - The key must be available on at least 2 HSMs before being used.
```

Penyebab: Pemeriksaan ketersediaan kunci mencari kunci yang, dalam kondisi langka tetapi mungkin, bisa hilang. Kesalahan ini biasanya terjadi dalam cluster dengan hanya satu HSM atau dalam cluster dengan dua HSMs selama periode di mana salah satu dari mereka sedang diganti. Dalam situasi ini, operasi pelanggan berikut kemungkinan menyebabkan kesalahan di atas:

- Kunci baru dihasilkan menggunakan perintah seperti [Kategori generate-simetris di CloudHSM CLI](#) atau [generate-asymmetric-pairKategori di CloudHSM CLI](#).
- Sebuah [Daftar kunci untuk pengguna dengan CloudHSM CLI](#) operasi dimulai.
- Contoh baru SDK dimulai.

Note

OpenSSL sering melakukan fork instance baru SDK.

Resolusi/rekomendasi: Pilih dari tindakan berikut untuk mencegah kesalahan ini terjadi:

- Gunakan `--disable-key-availability-check` parameter untuk mengatur ketersediaan kunci ke false dalam file konfigurasi [alat konfigurasi](#) Anda. Untuk informasi selengkapnya, lihat [AWS CloudHSM Parameter konfigurasi SDK 5 klien](#) bagian alat Konfigurasi.
- Jika menggunakan cluster dengan dua HSMs, hindari menggunakan operasi yang memicu kesalahan, kecuali selama kode inisialisasi.
- Tingkatkan jumlah HSMs di cluster Anda menjadi setidaknya tiga.

AWS CloudHSM mengekstraksi kunci menggunakan JCE

Gunakan bagian berikut untuk memecahkan masalah mengekstrak AWS CloudHSM kunci menggunakan JCE.

getEncoded,, atau GETS mengembalikan getPrivateExponent null

getEncoded,getPrivateExponent, dan getS akan mengembalikan null karena secara default dinonaktifkan. Untuk mengaktifkannya, lihat[Ekstraksi kunci menggunakan JCE untuk AWS CloudHSM](#).

Jika getEncoded,getPrivateExponent, dan getS mengembalikan null setelah diaktifkan, kunci Anda tidak memenuhi prasyarat yang tepat. Untuk informasi lebih lanjut, lihat[Ekstraksi kunci menggunakan JCE untuk AWS CloudHSM](#).

getEncoded getPrivateExponent,, atau GETS mengembalikan byte kunci di luar HSM

Anda atau seseorang yang memiliki akses ke sistem Anda telah mengaktifkan ekstraksi kunci yang jelas. Lihat halaman berikut untuk informasi selengkapnya, termasuk cara mengatur ulang konfigurasi ini ke status default dinonaktifkan.

- [Ekstraksi kunci menggunakan JCE untuk AWS CloudHSM](#)
- [Melindungi dan mengekstraksi kunci dari HSM](#)

Pelambatan HSM

Ketika beban kerja Anda melebihi kapasitas modul keamanan perangkat keras (HSM) AWS CloudHSM cluster Anda, Anda akan menerima pesan kesalahan yang menyatakan sibuk atau HSMs terhambat. Ketika ini terjadi, Anda mungkin melihat pengurangan throughput atau peningkatan tingkat permintaan penolakan dari HSMs. Selain itu, HSMs dapat mengirim kesalahan sibuk berikut.

Untuk Klien SDK 5

- Di PKCS11, kesalahan sibuk dipetakan keCKR_FUNCTION_FAILED. Kesalahan ini dapat terjadi karena beberapa alasan, tetapi jika pelambatan HSM menyebabkan kesalahan ini, baris log berikut akan muncul di log Anda:

- `[cloudhsm_provider::hsm1::hsm_connection::e2e_encryption::error]` Failed to prepare E2E response. Error: Received error response code from Server. Response Code: 187
- `[cloudhsm_pkcs11::decryption::aes_gcm]` Received error from the server. Error: This operation is already in progress. Internal error code: `0x000000BB`
- Di JCE, kesalahan sibuk dipetakan ke `com.amazonaws.cloudhsm.jce.jni.exception.InternalException: Unexpected error with the Provider: The HSM could not queue the request for processing.`
- Kesalahan SDKs 'sibuk lainnya' mencetak pesan berikut: Received error response code from Server. Response Code: 187.

Untuk Klien SDK 3

- Di PKCS11, kesalahan sibuk memetakan `CKR_OPERATION_ACTIVE` kesalahan.
- Di JCE, kesalahan sibuk dipetakan `CFM2Exception` dengan status. `0xBB` (187) Aplikasi dapat menggunakan `getStatus()` fungsi `CFM2Exception` untuk memeriksa status apa yang dikembalikan oleh HSM.
- Kesalahan SDKs sibuk lainnya akan mencetak pesan berikut: HSM Error: HSM is already busy generating the keys(or random bytes) for another request.

Resolusi

Anda dapat mengatasi masalah ini dengan menyelesaikan satu atau beberapa tindakan berikut:

- Tambahkan perintah coba lagi untuk operasi HSM yang ditolak di lapisan aplikasi Anda. Sebelum mengaktifkan perintah coba lagi, pastikan cluster Anda berukuran cukup untuk memenuhi beban puncak.

Note

Untuk Client SDK 5.8.0 dan di atasnya, perintah coba lagi diaktifkan secara default. Untuk detail tentang konfigurasi perintah coba ulang setiap SDK, lihat. [Konfigurasi lanjutan untuk alat konfigurasi Client SDK 5](#)

- Tambahkan lebih banyak HSMs ke cluster Anda dengan mengikuti instruksi di [Penskalaan HSMs dalam sebuah cluster AWS CloudHSM](#).

Important

Kami merekomendasikan pengujian beban klaster Anda untuk menentukan beban puncak yang harus Anda antisipasi, dan kemudian menambahkan satu HSM lagi untuk memastikan ketersediaan tinggi.

Jaga agar pengguna HSM tetap sinkron HSMs di klaster AWS CloudHSM

Untuk [mengelola pengguna HSM Anda](#), Anda menggunakan alat baris AWS CloudHSM perintah yang dikenal sebagai `cloudhsm_mgmt_util`. Ini berkomunikasi hanya dengan HSMs yang ada di file konfigurasi alat. Itu tidak mengetahui yang lain HSMs di cluster yang tidak ada dalam file konfigurasi.

AWS CloudHSM menyinkronkan kunci pada Anda HSMs di semua yang lain HSMs di cluster, tetapi tidak menyinkronkan pengguna atau kebijakan HSM. Saat Anda menggunakan `cloudhsm_mgmt_util` untuk [mengelola pengguna HSM, perubahan pengguna](#) ini mungkin hanya memengaruhi beberapa klaster —yang ada di file konfigurasi `cloudhsm_mgmt_util`. HSMs Ini dapat menyebabkan masalah saat AWS CloudHSM menyinkronkan kunci HSMs di cluster, karena pengguna yang memiliki kunci mungkin tidak ada di semua HSMs di cluster.

Untuk menghindari masalah ini, edit file konfigurasi `cloudhsm_mgmt_util` sebelum mengelola pengguna. Untuk informasi lebih lanjut, lihat [???](#).

Kehilangan koneksi ke AWS CloudHSM cluster

Ketika Anda [mengkonfigurasi AWS CloudHSM klien](#), Anda memberikan alamat IP dari HSM pertama di cluster Anda. Alamat IP ini disimpan dalam file konfigurasi untuk AWS CloudHSM klien. Ketika klien dimulai, klien mencoba untuk menyambung ke alamat IP ini. Jika tidak dapat—misalnya, karena HSM gagal atau Anda menghapusnya — Anda mungkin melihat kesalahan seperti berikut:

```
LIQUIDSECURITY: Daemon socket connection error
```

```
LIQUIDSECURITY: Invalid Operation
```

Untuk mengatasi kesalahan ini, perbarui file konfigurasi dengan alamat IP dari HSM aktif, terjangkau dalam klaster.

Untuk memperbarui file konfigurasi untuk AWS CloudHSM klien

1. Gunakan salah satu cara berikut untuk menemukan alamat IP dari HSM aktif di klaster Anda.

- Lihat HSMstab pada halaman detail cluster di [AWS CloudHSM konsol](#).
- Gunakan AWS Command Line Interface (AWS CLI) untuk mengeluarkan [describe-clusters](#) perintah.

Anda perlu alamat IP ini dalam langkah berikutnya.

2. Gunakan perintah berikut untuk menghentikan klien.

Amazon Linux

```
$ sudo stop cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client stop
```

CentOS 7

```
$ sudo service cloudhsm-client stop
```

CentOS 8

```
$ sudo service cloudhsm-client stop
```

RHEL 7

```
$ sudo service cloudhsm-client stop
```

RHEL 8

```
$ sudo service cloudhsm-client stop
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client stop
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client stop
```

Windows

- Untuk klien Windows 1.1.2+:

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- Untuk klien Windows 1.1.1 dan yang lebih lama:

Gunakan Ctrl+C di jendela perintah tempat Anda memulai klien. AWS CloudHSM

3. Gunakan perintah berikut untuk memperbarui file konfigurasi klien, menyediakan alamat IP yang Anda temukan di langkah sebelumnya.

```
$ sudo /opt/cloudhsm/bin/configure -a <IP address>
```

4. Gunakan perintah berikut untuk memulai klien.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Untuk klien Windows 1.1.2+:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Untuk klien Windows 1.1.1 dan yang lebih lama:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe  
C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Log AWS CloudHSM audit tidak ada di CloudWatch

Jika Anda membuat AWS CloudHSM kluster sebelum 20 Januari 2018, Anda perlu mengonfigurasi [peran terkait layanan](#) secara manual untuk mengaktifkan pengiriman log audit kluster tersebut. Untuk petunjuk tentang cara mengaktifkan peran terkait layanan di kluster HSM, lihat [Memahami Peran Terkait Layanan](#), serta [Membuat Peran Terkait Layanan](#) di Panduan Pengguna IAM.

Kustom IVs dengan panjang yang tidak sesuai untuk pembungkus kunci AES AWS CloudHSM

Topik pemecahan masalah ini membantu Anda menentukan apakah aplikasi Anda menghasilkan kunci dibungkus yang tidak dapat dipulihkan. Jika Anda terpengaruh oleh masalah ini, gunakan topik ini untuk mengatasi masalah.

Topik

- [Tentukan apakah kode Anda menghasilkan kunci terbungkus yang tidak dapat dipulihkan](#)
- [Tindakan yang harus Anda ambil jika kode Anda menghasilkan kunci terbungkus yang tidak dapat dipulihkan](#)

Tentukan apakah kode Anda menghasilkan kunci terbungkus yang tidak dapat dipulihkan

Anda akan terkena dampak hanya jika Anda bertemu semua kondisi di bawah ini:

Kondisi	Bagaimana saya tahu?
Aplikasi Anda menggunakan pustaka PKCS #11	Pustaka PKCS #11 diinstal sebagai file <code>libpkcs11.so</code> di folder <code>/opt/cloudhsm/lib</code> . Aplikasi yang ditulis dalam bahasa C umumnya menggunakan pustaka PKCS #11 secara langsung, sementara aplikasi yang ditulis di Java mungkin menggunakan pustaka secara tidak langsung melalui lapisan abstraksi Java. Jika Anda menggunakan Windows, Anda TIDAK terpengaruh, karena pustaka PKCS #11 saat ini tidak tersedia untuk Windows.
Aplikasi Anda secara khusus menggunakan versi 3.0.0 dari pustaka PKCS #11	Jika Anda menerima email dari AWS CloudHSM tim, kemungkinan besar Anda menggunakan pustaka PKCS #11 versi 3.0.0. Untuk memeriksa versi perangkat lunak pada instans aplikasi Anda, gunakan perintah ini:

Kondisi	Bagaimana saya tahu?
	<pre>rpm -qa grep ^cloudhsm</pre>
Anda membungkus kunci menggunakan pembungkus kunci AES	Pembungkus kunci AES berarti Anda menggunakan kunci AES untuk membungkus beberapa kunci lainnya. Nama mekanisme yang sesuai adalah CKM_AES_KEY_WRAP . Hal ini digunakan dengan fungsi C_WrapKey . Mekanisme pembungkus berbasis AES lainnya yang menggunakan vektor inisialisasi (IVs), seperti CKM_AES_GCM dan CKM_CLOUDHSM_AES_GCM , tidak terpengaruh oleh masalah ini. Pelajari lebih lanjut tentang fungsi dan mekanisme.
Anda menentukan IV kustom ketika memanggil pembungkus kunci AES, dan panjang IV ini lebih pendek dari 8	Bungkus kunci AES umumnya diinisialisasi menggunakan struktur CK_MECHANISM sebagai berikut: <pre>CK_MECHANISM mech = {CKM_AES_KEY_WRAP, IV_POINTER, IV_LENGTH};</pre> Masalah ini hanya berlaku untuk Anda jika: • IV_POINTER bukan NULL • IV_LENGTH kurang dari 8 byte

Jika Anda tidak memenuhi semua kondisi di atas, Anda dapat berhenti membaca sekarang. Kunci Anda yang dibungkus dapat dibuka dengan benar, dan masalah ini tidak berdampak pada Anda. Jika tidak, lihat [the section called “Tindakan yang harus Anda ambil jika kode Anda menghasilkan kunci terbungkus yang tidak dapat dipulihkan”](#).

Tindakan yang harus Anda ambil jika kode Anda menghasilkan kunci terbungkus yang tidak dapat dipulihkan

Anda harus mengambil tiga langkah berikut:

1. Segera tingkatkan pustaka PKCS #11 Anda ke versi yang lebih baru
 - [Pustaka PKCS #11 terbaru untuk Amazon Linux, CentOS 6 dan RHEL 6](#)
 - [Pustaka PKCS #11 terbaru untuk Amazon Linux 2, CentOS 7 dan RHEL 7](#)
 - [Pustaka PKCS #11 terbaru untuk Ubuntu 16.04 LTS](#)
2. Perbarui perangkat lunak Anda untuk menggunakan IV yang sesuai standar

Kami sangat menyarankan Anda mengikuti kode contoh kami dan hanya menentukan NULL IV, yang menyebabkan HSM untuk memanfaatkan IV default patuh standar. Atau, Anda dapat secara eksplisit menentukan IV sebagai `0xA6A6A6A6A6A6A6A6` dengan panjang IV yang sesuai 8. Kami tidak merekomendasikan penggunaan IV lain untuk pembungkus kunci AES, dan secara eksplisit akan menonaktifkan kustom IVs untuk pembungkus kunci AES di versi future dari pustaka PKCS #11.

Contoh kode untuk menentukan IV dengan benar muncul di [aes_wrapping.c](#) pada. GitHub

3. Identifikasi dan pulihkan kunci terbungkus yang ada

Anda harus mengidentifikasi kunci apa pun yang Anda bungkus menggunakan pustaka PKCS #11 versi 3.0.0, lalu hubungi support for assistance (<https://aws.amazon.com/support>) dalam memulihkan kunci ini.

Important

Masalah ini hanya memengaruhi kunci yang dibungkus dengan pustaka PKCS #11 versi 3.0.0. Anda dapat membungkus kunci menggunakan versi sebelumnya (paket bernomor 2.0.4 dan lebih rendah) atau versi yang lebih baru (paket bernomor 3.0.1 dan lebih tinggi) dari pustaka PKCS #11.

Menyelesaikan AWS CloudHSM kegagalan pembuatan cluster

Saat Anda membuat klaster, AWS CloudHSM buat peran terkait layanan AWSService RoleForCloud HSM, jika peran tersebut belum ada. Jika AWS CloudHSM tidak dapat membuat peran terkait layanan, upaya Anda untuk membuat klaster mungkin gagal.

Topik ini menjelaskan cara menyelesaikan masalah yang paling umum, sehingga Anda dapat berhasil membuat klaster. Anda perlu membuat peran ini hanya satu kali. Setelah peran terkait layanan dibuat di akun Anda, Anda dapat menggunakan salah satu metode yang didukung untuk membuat klaster tambahan dan mengelolanya.

Bagian berikut menawarkan saran untuk memecahkan masalah kegagalan pembuatan klaster yang berkaitan dengan peran terkait layanan. Jika Anda mencobanya tetapi masih tidak dapat membuat sebuah klaster, hubungi [Dukungan](#). Untuk informasi selengkapnya tentang peran terkait layanan AWSService RoleForCloud HSM, lihat [Peran terkait layanan untuk AWS CloudHSM](#)

Topik

- [Tambahkan izin yang hilang](#)
- [Buat peran terkait layanan secara manual](#)
- [Gunakan pengguna non-federasi](#)

Tambahkan izin yang hilang

Untuk membuat peran terkait layanan, pengguna harus memiliki izin `iam:CreateServiceLinkedRole`. Jika pengguna IAM yang membuat klaster tidak memiliki izin ini, proses pembuatan klaster gagal saat mencoba membuat peran terkait layanan di akun Anda. AWS

Ketika izin hilang menyebabkan kegagalan, pesan galat mencakup teks berikut.

This operation requires that the caller have permission to call `iam:CreateServiceLinkedRole` to create the CloudHSM Service Linked Role.

Untuk mengatasi kesalahan ini, berikan pengguna IAM yang membuat klaster izin `AdministratorAccess` atau tambahkan izin `iam:CreateServiceLinkedRole` untuk kebijakan IAM pengguna. Untuk instruksi, lihat [Menambahkan Izin untuk Pengguna Baru Atau Yang Sudah Ada](#).

Kemudian, cobalah untuk [membuat klaster](#) lagi.

Buat peran terkait layanan secara manual

Anda dapat menggunakan konsol IAM, CLI, atau API untuk membuat peran terkait layanan AWSServiceRoleForCloudHSM. Untuk informasi selengkapnya, silakan lihat [Membuat Peran Terkait Layanan](#) dalam Panduan Pengguna IAM.

Gunakan pengguna non-federasi

Pengguna federasi, yang kredensialnya berasal dari luar AWS, dapat melakukan banyak tugas dari pengguna non-federasi. Namun, AWS tidak mengizinkan pengguna untuk membuat panggilan API untuk membuat peran terkait layanan dari titik akhir gabungan.

Untuk mengatasi masalah ini, [buat pengguna non-gabungan](#) dengan izin `iam:CreateServiceLinkedRole`, atau berikan kepada pengguna non-gabungan yang ada izin `iam:CreateServiceLinkedRole`. Kemudian, minta pengguna itu [membuat sebuah klaster](#) dari AWS CLI. Tindakan ini membuat peran terkait layanan di akun Anda.

Setelah peran terkait layanan dibuat, jika mau, Anda dapat menghapus klaster yang dibuat oleh pengguna non-federasi. Menghapus klaster tidak memengaruhi peran. Setelah itu, setiap pengguna dengan izin yang diperlukan, termasuk pengguna federasi, dapat membuat AWS CloudHSM cluster di akun Anda.

Untuk memverifikasi bahwa peran telah dibuat, buka konsol IAM di <https://console.aws.amazon.com/iam/> dan pilih Peran. Atau, gunakan perintah IAM [get-role](#) di AWS CLI.

```
$ aws iam get-role --role-name AWSServiceRoleForCloudHSM
{
  "Role": {
    "Description": "Role for CloudHSM service operations",
    role policy statement
    "RoleId": "AR0AJ4I6WN5QVGG5G7CBY",
    "CreateDate": "2017-12-19T20:53:12Z",
    "RoleName": "AWSServiceRoleForCloudHSM",
    "Path": "/aws-service-role/cloudhsm.amazonaws.com/",
    "Arn": "arn:aws:iam::111122223333:role/aws-service-role/
cloudhsm.amazonaws.com/AWSServiceRoleForCloudHSM"
  }
}
```

Mengambil log konfigurasi AWS CloudHSM klien

AWS CloudHSM menawarkan alat untuk Client SDK 3 dan Client SDK 5 untuk mengumpulkan informasi tentang lingkungan Anda untuk AWS Support untuk memecahkan masalah.

Topik

- [AWS CloudHSM Alat dukungan SDK 5 klien](#)
- [AWS CloudHSM Alat dukungan SDK 3 klien](#)

AWS CloudHSM Alat dukungan SDK 5 klien

Skrip untuk AWS CloudHSM Client SDK 5 mengekstrak informasi berikut:

- File konfigurasi untuk komponen SDK Klien 5
- Berkas log tersedia
- Versi sistem operasi saat ini
- Informasi paket

Menjalankan alat info untuk Client SDK 5

SDK Klien 5 termasuk alat dukungan klien untuk setiap komponen, tetapi semua alat berfungsi sama. Jalankan alat untuk membuat file output dengan semua informasi yang dikumpulkan.

Alat menggunakan sintaks seperti ini:

```
[ pkcs11 | dyn | jce | ksp | cli ]_info
```

Misalnya, untuk mengumpulkan informasi untuk dukungan dari host Linux yang menjalankan pustaka PKCS #11 dan memiliki sistem yang menulis ke direktori default, Anda akan menjalankan perintah ini:

```
/opt/cloudhsm/bin/pkcs11_info
```

Alat ini membuat file output di dalam /tmp direktori.

AWS CloudHSM CLI

Untuk mengumpulkan data dukungan untuk AWS CloudHSM CLI di Linux

- Gunakan alat dukungan untuk mengumpulkan data.

```
/opt/cloudhsm/bin/cli_info
```

Untuk mengumpulkan data dukungan untuk AWS CloudHSM Windows

- Gunakan alat dukungan untuk mengumpulkan data.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\cli_info.exe"
```

PKCS #11 library

Untuk mengumpulkan data dukungan untuk pustaka PKCS #11 di Linux

- Gunakan alat dukungan untuk mengumpulkan data.

```
/opt/cloudhsm/bin/pkcs11_info
```

Untuk mengumpulkan data dukungan untuk pustaka PKCS #11 di Windows

- Gunakan alat dukungan untuk mengumpulkan data.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\pkcs11_info.exe"
```

OpenSSL Dynamic Engine

Untuk mengumpulkan data dukungan untuk OpenSSL Dynamic Engine di Linux

- Gunakan alat dukungan untuk mengumpulkan data.

```
/opt/cloudhsm/bin/dyn_info
```

JCE provider

Untuk mengumpulkan data dukungan untuk penyedia JCE di Linux

- Gunakan alat dukungan untuk mengumpulkan data.

```
/opt/cloudhsm/bin/jce_info
```

Untuk mengumpulkan data dukungan untuk penyedia JCE di Windows

- Gunakan alat dukungan untuk mengumpulkan data.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\jce_info.exe"
```

Key Storage Provider

Untuk mengumpulkan data dukungan untuk Penyedia Penyimpanan Kunci di Windows

- Gunakan alat dukungan untuk mengumpulkan data.

```
PS C:\> & "C:\Program Files\Amazon\CloudHSM\bin\ksp_info.exe"
```

Mengambil log dari lingkungan tanpa server

Untuk mengonfigurasi lingkungan tanpa server, seperti Fargate atau Lambda, kami sarankan Anda mengonfigurasi jenis log Anda. AWS CloudHSM term Setelah dikonfigurasi ket erm, lingkungan tanpa server akan dapat menghasilkan output ke. CloudWatch

Untuk mendapatkan log klien CloudWatch, lihat [Bekerja dengan grup log dan aliran log](#) di Panduan Pengguna Amazon CloudWatch Logs.

AWS CloudHSM Alat dukungan SDK 3 klien

Skrip untuk AWS CloudHSM Client SDK 3 mengekstrak informasi berikut:

- Sistem operasi dan versi saat ini
- Informasi konfigurasi klien dari file `cloudhsm_client.cfg`, `cloudhsm_mgmt_util.cfg`, dan `application.cfg`

- Log klien dari lokasi khusus untuk platform
- Informasi klaster dan HSM dengan menggunakan `cloudhsm_mgmt_util`
- Informasi OpenSSL
- Versi klien dan build saat ini
- Versi penginstal

Menjalankan alat info untuk Client SDK 3

Skrip membuat file output dengan semua informasi yang dikumpulkan. Script membuat file output di dalam direktori `/tmp`.

Linux: `/opt/cloudhsm/bin/client_info`

Windows: `C:\Program Files\Amazon\CloudHSM\client_info`

Warning

Skrip ini memiliki masalah yang diketahui untuk SDK Klien 3 versi 3.1.0 hingga 3.3.1. Kami sangat menyarankan Anda meningkatkan ke versi 3.3.2 yang mencakup perbaikan untuk masalah ini. Silakan merujuk ke [Masalah Diketahui](#) untuk informasi lebih lanjut sebelum menggunakan alat ini.

AWS CloudHSM kuota

Kuota, sebelumnya dikenal sebagai batas, adalah nilai yang ditetapkan untuk sumber daya. AWS Kuota berikut berlaku untuk AWS CloudHSM sumber daya Anda per AWS Wilayah dan AWS akun. Kuota default adalah nilai awal yang diterapkan oleh AWS, dan nilai-nilai ini tercantum dalam tabel di bawah ini. Kuota yang dapat disesuaikan dapat ditingkatkan di atas kuota default.

Kuota layanan

Sumber Daya	Quotas Default	Dapat disesuaikan?
Klaster	4	Ya
HSMs	6	Ya
HSMs per cluster	28	Tidak

Cara yang disarankan untuk meminta kenaikan kuota adalah dengan membuka [konsol Service Quotas](#). Di konsol, pilih layanan dan kuota Anda, dan kirimkan permintaan Anda. Untuk informasi lebih lanjut, lihat [dokumentasi Service Quotas](#).

Kuota dalam tabel System Quotas berikut ini tidak dapat disesuaikan.

Kuota sistem

Sumber daya	Kuota untuk hsm1.medium	Kuota untuk hsm2m.medium
Tombol maksimum per cluster	3.300	16.666 total kunci, dengan tombol asimetris memiliki maksimum 3.333
Maksimum pengguna per cluster	250	1,024
Panjang maksimum nama pengguna	31 karakter	31 karakter
Panjang sandi yang diperlukan	8 hingga 32 karakter	8 hingga 32 karakter

Sumber daya	Kuota untuk hsm1.medium	Kuota untuk hsm2m.medium
Jumlah maksimum koneksi klien bersamaan per cluster ¹	900	900
Jumlah maksimum sesi PKCS #11 per aplikasi	1,024	1,024

[1] Koneksi klien untuk Client SDK 3 adalah daemon klien. Untuk Client SDK 5, koneksi klien adalah aplikasi.

Unduh untuk AWS CloudHSM Client SDK

Topik berikut menyediakan unduhan untuk AWS CloudHSM klien SDKs.

Note

Untuk informasi tentang platform apa yang didukung oleh setiap SDK Klien, lihat [AWS CloudHSM Platform yang didukung SDK Klien 5](#) dan [AWS CloudHSM Platform yang didukung SDK Klien 3](#).

Topik

- [AWS CloudHSM rilis SDK Klien terbaru](#)
- [AWS CloudHSM rilis SDK Klien sebelumnya](#)
- [AWS CloudHSM rilis SDK Klien yang tidak digunakan lagi](#)
- [AWS CloudHSM end-of-life Rilis SDK klien](#)

AWS CloudHSM rilis SDK Klien terbaru

Pada bulan Maret 2021, AWS CloudHSM merilis Client SDK versi 5.0.0, yang memperkenalkan Client SDK yang serba baru dengan persyaratan, kemampuan, dan dukungan platform yang berbeda.

Client SDK 5 didukung penuh untuk lingkungan produksi, dan menawarkan komponen dan tingkat dukungan yang sama dengan Client SDK 3. Untuk informasi selengkapnya, lihat [Bandingkan AWS CloudHSM dukungan komponen SDK Klien](#).

Note

Mulai CloudHSM SDK 5.17, kami memperbarui perjanjian tingkat layanan kami untuk mendukung hingga 3 versi minor sebelumnya dan satu tahun sejak tanggal rilis versi SDK. Kami akan merilis versi baru CloudHSM SDK untuk menghadirkan fitur baru dan untuk keamanan, stabilitas, perbaikan kinerja. Kami akan menonaktifkan tautan unduhan untuk versi CloudHSM SDK yang lebih lama dan tidak didukung saat versi baru tersedia.

Bagian ini mencakup versi terbaru SDK Klien. Untuk informasi tentang rilis sebelumnya, lihat [Rilis SDK Sebelumnya](#). Untuk versi yang tidak digunakan lagi, lihat Rilis SDK yang Tidak Digunakan Lagi.

Rilis Klien SDK 5: Versi 5.17.1

Amazon Linux 2023

Unduh perangkat lunak versi 5.17.1 untuk Amazon Linux 2023 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum e5d37660312c8d5b9c36a9b74555222a755276e2c1f360418c34125b2b715b97)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 9b84d756e56f19b830adbd46aa5ac5096ae28944e34dd8dac3bd534e487d7aee)
- [Penyedia OpenSSL](#) (SHA256 checksum 5df0647ea5fe6da453bbf9a9578de9674ba93b722ec17dfe79825006211dc7a9)
- [Penyedia JCE](#) (SHA256 checksum ac820e95618bf6cecf76b6ae0c5fb59198041f1d583178105a506d255d886fc9)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum 3b2f966c05690c14c325ea37cf9799bad477b2573f1077a5319d92933af9e590)

Unduh perangkat lunak versi 5.17.1 untuk Amazon Linux 2023 tentang arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum 8cf4f7e9bed86ed15198e0870075fa01cb8243d3646ae9a1417ee3eba0cb3111)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 831c7565caaea18c9c239b060347ec1b0e5121ec94d5bfe4cecdad0fd4b1aa9ab)
- [Penyedia OpenSSL](#) (SHA256 checksum 73e91db43c98fa1eac6ce4d39f211d0dafa8445941b9f3adf77876cbeb443561)
- [Penyedia JCE](#) (SHA256 checksum 1cc6bd9226ac98ebf898498f3b51b41b3ed58304f0290bd2045ab472e3bf3e14)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum d716108dfc8f0e89e082603dbd5c95379b9f593628ab30bc611c37c169fae8d0)

Amazon Linux 2

Unduh perangkat lunak versi 5.17.1 untuk Amazon Linux 2 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum c22875d9a81b2a5814d949b06d1467d4fbb111e67a6b29d83e961d95bdebd9e9)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum f54d264ede8624867c3010d6b3979c51cd05a34d90a4169208400013c8de76c5)
- [Penyedia JCE](#) (SHA256 checksum c46297760a1d306541498e6a0295d7143273286dbb798c1c65bee38799b68e6b)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum 87162212c93d8536d9247632221d855e841637a786c405eb47bc1410d62987d9)

Unduh perangkat lunak versi 5.17.1 untuk Amazon Linux 2 tentang ARM64 arsitektur:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 0abe258732376b20730c2d2e089458e83eff4e751df8e60c0c70d3d92e75d7a4)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 4f8101debd678327a26246e50df10fe79663fc264fef49deef63e036221fc0b)
- [Penyedia JCE](#) (SHA256 checksum dfc83b8f2db9e97eab98bff431c3374181a0051884abe7e9424621fcdba9c8c5)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum bc43b221610d1c220ed402ee14bb0275ad79fb21a342639d1266579881da953a)

RHEL 10 (10.0+)

Unduh perangkat lunak versi 5.17.1 untuk RHEL 10 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 11076479568454ad1083bb6a81953a73f9175f6d8626e247e1e16439a5c105ec)
- [Penyedia OpenSSL](#) (SHA256 checksum 6c3d7a58b38ef39c7b5310958eb684feed20edf6731603a8b727b4bd978f465f)
- [Penyedia JCE](#) (SHA256 checksum 921883948344a06716d6b3b30f67708dedff550978efce6d5817b36c83e83b35)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum 4d69481cc4e725e65811e031fe344cb178016cd268759bb910acdb14df9be413)

Unduh versi 5.17.1 perangkat lunak untuk RHEL 10 pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum 7a20af9f7a9874132c1314841ef2ead0daef92ced3ba2289938a88d0ee929e)
- [Penyedia OpenSSL](#) (SHA256 checksum e762bb21086131725e9ca13b7d25e6fcb474888ae49317e1b045ddf826a8338c)
- [Penyedia JCE](#) (SHA256 checksum c8bb96ded1abdfb54cd710fade3c91ce6030f5dc9ddeea639cc0a0cac5c26bb1)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum d48ccaec9d4247f0cf4c81a6f092704630933dbc89e0b0ba9a75629fd863b7)

RHEL 9 (9.2+)

Unduh perangkat lunak versi 5.17.1 untuk RHEL 9 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum df4f1de8e68e363b351770dd74782d204f04d4db646c554fa96182b429006618)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 06d2e65adb076e65b81b51bcea34b64f718b6aa29c173eebcaf277de2c8d03ec)
- [Penyedia OpenSSL](#) (SHA256 checksum 802e608484ab748eccdce8da0854dc668e90578d4c2bc6ede4caf4121d10171f)
- [Penyedia JCE](#) (SHA256 checksum 1a4fdfe82cc79a4de85a19d6645916b6b388c919a71a22efae7bafaba195db0f)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum daab5b758c61f49b34b6f7e288debb121adc3d63b9652db086855d160c988570)

Unduh perangkat lunak versi 5.17.1 untuk RHEL 9 tentang arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum 2fe6549339b743200c6be91b8abf8017c3f67c90f7db25ffd0ab1915b91d0f78)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum a0a964dcb817251faafcd432e6260dff1d6e1bcdd7e7ffbd54a691cbe366b58)
- [Penyedia OpenSSL](#) (SHA256 checksum bbcc9b3a2455328e6e7b11ff64b7e7653757960a74bfd9d6b3cc998b2a955fb2)
- [Penyedia JCE](#) (SHA256 checksum f9c74e2ededb9d2fdb7076789223c4cde7a0845b6f999b006fa741e9428195c7)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum 0c4c1c7f4e06ba3b5b5dfe619abb0fa9d4729c87c668361fdb99b2a62f4b00fc)

RHEL 8 (8.3+)

Unduh perangkat lunak versi 5.17.1 untuk RHEL 8 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 0f8465ab44f7388980b22bc1f8c6c3c3e09e809ba2712170c85db86918817803)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 8fe4f9fb6088d2a32e210696775d395b918ee867dede8b2152ff37d0d99cc8f3)
- [Penyedia JCE](#) (SHA256 checksum 948d37f435e7b03ec8409ee48f89c265009c7e7a76e7a18921c5ef88f3504391)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bffffb66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum 2168377badf708047ee14d2bc57b9234f01ce2c9f42568dbd1185723a63eeeb8)

Unduh versi 5.17.1 perangkat lunak untuk RHEL 8 pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum 69db8b832edba06fe055ce9772bfad645a86e5e530d20e8f14738fc7cb06a5c7)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum ac7ad7b11e63d1fb7af57ff1e61b421aa0fccbca99ed15b0ddec1602fea2acd3)
- [Penyedia JCE](#) (SHA256 checksum 7520a95076a7a794219677489d2c764502a6dcfeea484f10059e283f1aee6975)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bffffb66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum 596acc3164695a262b15edbb92eda89ffd4b8d06b91617ce91058c2d99ae2a53)

Ubuntu 24.04 LTS

Unduh perangkat lunak versi 5.17.1 untuk Ubuntu 24.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum bc3146d5dc7aabe5359c41d2bd0fd97ed4bc2f389be6a4b94423065af9d60bd0)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 05d6afa19949b69af4325f55d285a89255ab2c7487aba898c74cbd4ee8e30fdf)
- [Penyedia OpenSSL](#) (SHA256 checksum 3deba48c356478fdbb0236ad435cee8f7f8cfe3cec1a894aea3f79d80d398cce)
- [Penyedia JCE](#) (SHA256 checksum ebacb95d8f92528e236a5342cb5c1ecbf74c228428fb4a080c0c972b697292b5)

- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bffffb66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum 575f54610d26561bb98b69548de2f61a17b5b93ed49be13821be536d76ffdb75)

Unduh perangkat lunak versi 5.17.1 untuk Ubuntu 24.04 LTS pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum b0c918575e611f486902ebbcc3ee84e15217247f5577b06764cfc620660c6ec4)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 57f0afb66c34a91e96bef4964dd3fb487fe5b9b06d0818ee9a2c90ad60068321)
- [Penyedia OpenSSL](#) (SHA256 checksum dcf5128c6913e43e0a304d0221c243fd5228f235b78fac5ffd43478bb0d96062)
- [Penyedia JCE](#) (SHA256 checksum 21d43992ace3aec9ec3711e7bd64ec716d3281f627e78a6f7a328a8e533beec4)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bffffb66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum 096844ef80a7d573843682cfb3797a0c45608d004639ab0e671065f3964c06e3)

Ubuntu 22.04 LTS

Unduh perangkat lunak versi 5.17.1 untuk Ubuntu 22.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum ec8c275832f9936e698106454e930268d7f2f310faffa2b7d03c3c03c6e297e1)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 316c3929363ad1d764f356bbf56dff121b3bccefa5060b94c5d9d7955f828ff4)
- [Penyedia JCE](#) (SHA256 checksum 4c53b61bcdada0a684b605c0630f52b6b3bf6e9ed64f4c623f35fa4283e9770d)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bffffb66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum 6b4ba5d3d06e5234b64b7c60310d5b477c2d8732fbbd3e9c60dcb56f2ef69a6e)

Unduh perangkat lunak versi 5.17.1 untuk Ubuntu 22.04 LTS pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum d26d10425572f0624d2205e1409b88ccc5d793bcfeef6cd26fdbcb5c41f5c8129)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum e3b6cd3e93f2645d9a23add1a514e314ad3d48a93b253e6b984577b8e502e1f0)

- [Penyedia JCE](#) (SHA256 checksum 7ab3ea04d34f92c585121b110f620b632616a5585a4ed153fbb61fedef667b06)
- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum 9cf4a05ebf2bce4a0c9e57dd2eef87ba47ac05c4a303d4465cbf41536736c685)

Windows Server 2025

Unduh perangkat lunak versi 5.17.1 untuk Windows Server 2025 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 145ea74ff3ab70c0f5f1ccd8ced90b86f0b4f3ee029d2d9452e9b0724ae2f64e)
- [Penyedia JCE](#) (SHA256 checksum 074bc0a990d51debea16e8691171c1cfc4e164795eb839be1d311858b157b0a6)
- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum ddc11eee7b884ddaa203ded85f36553a9a636d35bf9dd4b6557c23a167ed7c4a)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum 5bef4732d23861795a8463c3a3511e0a235a29972d584c2f65384e87ba4bb48c)

Windows Server 2022

Unduh versi 5.17.1 perangkat lunak untuk Windows Server 2022 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 145ea74ff3ab70c0f5f1ccd8ced90b86f0b4f3ee029d2d9452e9b0724ae2f64e)
- [Penyedia JCE](#) (SHA256 checksum 074bc0a990d51debea16e8691171c1cfc4e164795eb839be1d311858b157b0a6)
- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bfff66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum ddc11eee7b884ddaa203ded85f36553a9a636d35bf9dd4b6557c23a167ed7c4a)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum 5bef4732d23861795a8463c3a3511e0a235a29972d584c2f65384e87ba4bb48c)

Windows Server 2019

Unduh perangkat lunak versi 5.17.1 untuk Windows Server 2019 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 145ea74ff3ab70c0f5f1ccd8ced90b86f0b4f3ee029d2d9452e9b0724ae2f64e)
- [Penyedia JCE](#) (SHA256 checksum 074bc0a990d51debea16e8691171c1cfc4e164795eb839be1d311858b157b0a6)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bffffb66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum ddc11eee7b884ddaa203ded85f36553a9a636d35bf9dd4b6557c23a167ed7c4a)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum 5bef4732d23861795a8463c3a3511e0a235a29972d584c2f65384e87ba4bb48c)

Windows Server 2016

Unduh perangkat lunak versi 5.17.1 untuk Windows Server 2016 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 145ea74ff3ab70c0f5f1ccd8ced90b86f0b4f3ee029d2d9452e9b0724ae2f64e)
- [Penyedia JCE](#) (SHA256 checksum 074bc0a990d51debea16e8691171c1cfc4e164795eb839be1d311858b157b0a6)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 89b5c2232c1739d61813d16f6d3177984bffffb66754367883e2d7e997a363f30)
- [CloudHSM CLI](#) (SHA256 checksum ddc11eee7b884ddaa203ded85f36553a9a636d35bf9dd4b6557c23a167ed7c4a)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum 5bef4732d23861795a8463c3a3511e0a235a29972d584c2f65384e87ba4bb48c)

Client SDK 5.17.1 mencakup perbaikan bug dan peningkatan.

AWS CloudHSM rilis SDK Klien sebelumnya

Bagian ini mencantumkan rilis SDK Klien sebelumnya.

Versi 5.17.0

Amazon Linux 2023

Unduh perangkat lunak versi 5.17.0 untuk Amazon Linux 2023 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum e0656d606c1f75b8ecf11df79f2d4aa17c0b68983bb20d271dabe02f7b97ca47)

- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
b8917abe48799035018bc044fbe2e9db8d5568ebf29663ecc12d53ec2be5efd2)
- [Penyedia OpenSSL](#) (SHA256 checksum 61e49cb0f1f9eb7bcf76f5ef36f01cf1fadfa56bf5eab94b05efdd93214c9cf7)
- [Penyedia JCE](#) (SHA256 checksum 4ba2548fa27f3662c56fc140386366cfb40cde899695412c6d4731a7d0a70e43)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM CLI](#) (SHA256 checksum 0a9de06b3f15d38f3aa02dd0288d3a9ad41f525f692f539664168b9e6b95e18d)

Unduh perangkat lunak versi 5.17.0 untuk Amazon Linux 2023 tentang arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum
9bd40e13bf777a4651721ebe5613cf2e4f0f339679d50f06b81f7f1b73c1fef5)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
e592419bc0449664f0707281a46ebb775091386ac5b297e35ff0e4160468cbe0)
- [Penyedia OpenSSL](#) (SHA256 checksum 46358ef68388928702d780564811814aa826c143ccc0624a2a45d8995226d33e)
- [Penyedia JCE](#) (SHA256 checksum db5bd1494c742a64ca3f2bc57253b17655ad38ecc27d45422965466c1caf740f)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM CLI](#) (SHA256 checksum e3d21b4460a421058f65a597b0bb30a9fdda890363ebee107b124a1212d5a042)

Amazon Linux 2

Unduh perangkat lunak versi 5.17.0 untuk Amazon Linux 2 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
17d96d5f4ad25de74fadcc7272fac825d43c496b0b7912e4d32b1ccfb67250bc)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
3d3de61bd16ef849e53827709b8c8bd9eaf89bd1b7d56b1a02455bca38fe1eb6)
- [Penyedia JCE](#) (SHA256 checksum 5c268313ebf16cf762619a1e75f1d44efdbd6eaf5ba4bbce67ecfe96bbbed9775)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM CLI](#) (SHA256 checksum 43b678eff0225590cc59c02645332f4459c8161a6742e562eb95902d1bec9134)

Unduh perangkat lunak versi 5.17.0 untuk Amazon Linux 2 tentang ARM64 arsitektur:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 7910ec55f8b106d804290b77e6d13cefb85ef79cbc728f69bb21eebeb7245c6b)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 399c16e0fbfa372710a83836517cfeaae0895513931427a458e7994ce91a321d)
- [Penyedia JCE](#) (SHA256 checksum 1678581e31fcfe58625ca1e3c0762c4b96095065a868a5b7f0872214e5ab9e8)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM CLI](#) (SHA256 checksum 52257b580d73d902e9fd29f2c334f88cabee484c7eec8af28f7b2aa7864466e7)

RHEL 10 (10.0+)

Unduh perangkat lunak versi 5.17.0 untuk RHEL 10 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 037102ee2b7f9c244d647cfb06da282b99349e208d27b7b0aa80bd32e382a2d4)
- [Penyedia OpenSSL](#) (SHA256 checksum 3d6d97df03a003bae579dabbf9abeefc1bd2fa6d799a6db01c3ef98d02145393)
- [Penyedia JCE](#) (SHA256 checksum 4e69ec89acfefc19f922c68988d4674718d49d5cda2e0a288f7f0343ebfc270c)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM CLI](#) (SHA256 checksum 38a67191949cc8cb0c69ef2fe4694514b2ca6af4cf5f8763bd7f6860c501675a)

Unduh perangkat lunak versi 5.17.0 untuk RHEL 10 pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum bc383aff236d4b2a0fdbc5fd81907cf9212ad8f3cfc91fadeb6cdc14b9d3d581)
- [Penyedia OpenSSL](#) (SHA256 checksum d9ead84f8d2e6a09044646d62fed293ca082431e5e0338bcbfbcf8acd023ecc0)
- [Penyedia JCE](#) (SHA256 checksum c61fdb5f05cf486619deb3e23ecb171ce4485bb64c891bf4480fbe6a4276c3bc)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM CLI](#) (SHA256 checksum fbd4387ef3fab177de8c888d98d1b24bc042d23ea2f93834dfb5ee918518f406)

RHEL 9 (9.2+)

Unduh perangkat lunak versi 5.17.0 untuk RHEL 9 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
7fc68025493fc4a0bc4d3b2fd11b17da4481b6bee51fc86a40f95fe4c206aa60)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
3e786e9218f657c93e1bf1b54c81f0f573e39d10147cf99485da73677b0d67ae)
- [Penyedia OpenSSL](#) (SHA256 checksum c52b366f9c6c960d697794bd7840888bc51e1ee5aa654b8d2c83ec1626797c56)
- [Penyedia JCE](#) (SHA256 checksum 427e9c68815188b59a0f9b2fd14793791d9a4ce4d3e5ddc34fce916af18a45f1)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM CLI](#) (SHA256 checksum 49d128639eea0e588c95281635a33b0e62837cf5a9d992e0bacb5b598a0037ed)

Unduh perangkat lunak versi 5.17.0 untuk RHEL 9 tentang arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum
3ae8f5068abb3d9ca75ac21b160acc7ee84c7ded6107a889602448af51d5bd82)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
64c5feaf86fc40e28616156f17942cc46d8ac93babdd8af59dd98a1f2e73117d)
- [Penyedia OpenSSL](#) (SHA256 checksum 24df4296ba0ae15f6b7060ccffeca0d1a24f58d9e8da8eb6a4afa3a6a50770fd)
- [Penyedia JCE](#) (SHA256 checksum 648d7948ab8a1271d7cf0eddc2ca32e960979be481b99a1ce2ce8b9fa3a2b2ed)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM CLI](#) (SHA256 checksum edd1f3707aeea285a55cf24d2e2385489458a69e5d7dab4014783f5cdc232d77)

RHEL 8 (8.3+)

Unduh perangkat lunak versi 5.17.0 untuk RHEL 8 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
85e070ed3273ead9698711dfc2c3e590dda1a01f99074943459ddc5ac052872d)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
d0c8e5603f4c99d0ba547c0ba40361c15977f3349ac120d514afed59abf32c83)
- [Penyedia JCE](#) (SHA256 checksum 4567b9a236d0dfd0410863beaced6d5d4e71acb06b2b136c7c40ce6e704f3698)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM CLI](#) (SHA256 checksum f8a090752849fac9b99ef67b59f571e476908dccec1285b466c25e6cc71793c)

Unduh perangkat lunak versi 5.17.0 untuk RHEL 8 pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum 71ade413435947eb2093727cb0e7a5c5f084ca859cb38d728a8c3d23f3b49bd0)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 02210da06b3fd22813fce8b89b753d150c8f321df405c40dd9f0172dc03968a7)
- [Penyedia JCE](#) (SHA256 checksum 80ac5dbaad1f644d6b9551e6a1d95d4b4a51ae82d34e45593be77b1068f7c910)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM CLI](#) (SHA256 checksum c3c56c3b9fed1dc51761d9412007c727d3b608377a089bd637a44bc374744d5c)

Ubuntu 24.04 LTS

Unduh perangkat lunak versi 5.17.0 untuk Ubuntu 24.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum b3869a6fe9f2031cb5f36ac8d2a5a5280451159e7a68f0d6f91f98b5f0100939)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 3e46443a3f8425f62cec8d8ca414c4b5446b86b42012877ebe44a17303403413)
- [Penyedia OpenSSL](#) (SHA256 checksum 69c2e61894e954ac1d0a785025902af5aa1922a7f6011a847f2111e153db706e)
- [Penyedia JCE](#) (SHA256 checksum 663da1a7b13a1055bb90fd7b168b69fcfb000eb91c1c3ce6062d8cc8db607fe4)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM CLI](#) (SHA256 checksum 0d57b4f08bd768ed595978d2a6d3fe6bbbf8398a784c66c4e30c19ecf6730b5d)

Unduh perangkat lunak versi 5.17.0 untuk Ubuntu 24.04 LTS pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum 94d9a36a3d3f2f04c10539713b98ddad68654c739572064e4c123005a9caf575)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum a442950055a7051f50b4cdfa7ec347c1b207b0bbf7e159b31a3bf96896b66146)
- [Penyedia OpenSSL](#) (SHA256 checksum dcc4b5a370da176a86e47683a78703a5604d6b7b49ba687377460f48c3ddff26)
- [Penyedia JCE](#) (SHA256 checksum 1fe1fa59a3e5c32edd5668604e43f7ce152473514b25ee736dfb0e9cf87c92c9)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)

- [CloudHSM CLI](#) (SHA256 checksum 2bf2c13bf1520eaf2a45eb55aaba3334fa54da686d22d1a03005ec8f62ea8895)

Ubuntu 22.04 LTS

Unduh perangkat lunak versi 5.17.0 untuk Ubuntu 22.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 2dc1f00952f69b59b0498e85c2f2009f80c9a2f9f28104995b0291da5f45d805)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum df0e7ff8852a93fb4c5807317d3438ab96ffe69f19ba144d872991cbcb4172f6)
- [Penyedia JCE](#) (SHA256 checksum 77a3cc37e066ae0ad5655ddc3dd38a0513bdcdb3c9d3d780e2973ac42a75035e)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM CLI](#) (SHA256 checksum 88ca1614178771991649b8117ddf27ffb99c6842ee89f68e77c6df3470e6ba78)

Unduh perangkat lunak versi 5.17.0 untuk Ubuntu 22.04 LTS pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum 0e1f62783c16deb9188c9cf0085fe972671e5ea9bf0fc2fc3255f908885949c)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 109d25583e2d5ed0fe821a77098d8f2c815011f7873168e0110aeb857dc903ab)
- [Penyedia JCE](#) (SHA256 checksum 9151661a39f6394aa3ace682bd7964095ed5300244582d3f6b1f81255ebf22cf)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM CLI](#) (SHA256 checksum 66e408cea800a93becf378b8611595ad8cd1bf51e0ae7feebc548fe3099b477c)

Windows Server 2025

Unduh perangkat lunak versi 5.17.0 untuk Windows Server 2025 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 77dd4ad5da0d4a21c82ff2ea03c09a6b8c2005e27b6aba6724e4bec75606b0e7)
- [Penyedia JCE](#) (SHA256 checksum c2c4766b67f02a93cca22b166e63fe6f4c71bfd9f7b45aa2da6a603d825e5c1a)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)

- [CloudHSM CLI](#) (SHA256 checksum 4feb0aeb7554ddd919eb7df301a4da537fffb0577a89a053fba802a2d3657a2a)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum 3766d4d4ab1bd7afa6b72e74bc80d8e5054481c8535f5a5c841cc320617c4e1e)

Windows Server 2022

Unduh perangkat lunak versi 5.17.0 untuk Windows Server 2022 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 77dd4ad5da0d4a21c82ff2ea03c09a6b8c2005e27b6aba6724e4bec75606b0e7)
- [Penyedia JCE](#) (SHA256 checksum c2c4766b67f02a93cca22b166e63fe6f4c71bfd9f7b45aa2da6a603d825e5c1a)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM CLI](#) (SHA256 checksum 4feb0aeb7554ddd919eb7df301a4da537fffb0577a89a053fba802a2d3657a2a)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum 3766d4d4ab1bd7afa6b72e74bc80d8e5054481c8535f5a5c841cc320617c4e1e)

Windows Server 2019

Unduh perangkat lunak versi 5.17.0 untuk Windows Server 2019 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 77dd4ad5da0d4a21c82ff2ea03c09a6b8c2005e27b6aba6724e4bec75606b0e7)
- [Penyedia JCE](#) (SHA256 checksum c2c4766b67f02a93cca22b166e63fe6f4c71bfd9f7b45aa2da6a603d825e5c1a)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM CLI](#) (SHA256 checksum 4feb0aeb7554ddd919eb7df301a4da537fffb0577a89a053fba802a2d3657a2a)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum 3766d4d4ab1bd7afa6b72e74bc80d8e5054481c8535f5a5c841cc320617c4e1e)

Windows Server 2016

Unduh perangkat lunak versi 5.17.0 untuk Windows Server 2016 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 77dd4ad5da0d4a21c82ff2ea03c09a6b8c2005e27b6aba6724e4bec75606b0e7)

- [Penyedia JCE](#) (SHA256 checksum c2c4766b67f02a93cca22b166e63fe6f4c71bfd9f7b45aa2da6a603d825e5c1a)
- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum a89e6d7e4e1f4049fefa64c765e97928fc54c366249f315ac6a839704e1ffba8)
- [CloudHSM CLI](#) (SHA256 checksum 4feb0aeb7554ddd919eb7df301a4da537fffb0577a89a053fba802a2d3657a2a)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum 3766d4d4ab1bd7afa6b72e74bc80d8e5054481c8535f5a5c841cc320617c4e1e)

Client SDK 5.17.0 memperkenalkan fitur dan peningkatan baru di beberapa komponen.

Dukungan platform

- Menambahkan dukungan RHEL 8 untuk ARM64 arsitektur.

JCE

- Menambahkan dukungan untuk OpenJDK 25 di semua platform
- Client SDK 5.17.0 adalah rilis terakhir yang mendukung OpenJDK 8.

Fitur Baru

- Memperkenalkan OpenSSL Provider SDK baru untuk meningkatkan kompatibilitas OpenSSL 3.2+ dan operasi kriptografi tambahan.
- Menambahkan dukungan kurva Ed25519 ke generasi kunci EC.
- Menambahkan dukungan untuk penandatanganan dan verifikasi eDDSA (Edwards-Curve Digital Signature Algorithm) di CloudHSM CLI.

CloudHSM CLI

- Menambahkan dukungan kurva Ed25519 ke generasi kunci EC untuk tipe hsm2m.medium non-FIPS.
- Menambahkan dukungan untuk penandatanganan dan verifikasi eDDSA (Edwards-Curve Digital Signature Algorithm) di CloudHSM CLI. Untuk informasi selengkapnya, lihat [Hasilkan tanda tangan dengan mekanisme HashEd DSA di CloudHSM CLI](#) dan [Verifikasi tanda tangan yang ditandatangani dengan mekanisme HashEd DSA di CloudHSM CLI](#).

Penyedia OpenSSL

- SDK Penyedia OpenSSL baru menyediakan integrasi dengan arsitektur Penyedia OpenSSL 3.2+ untuk tipe hsm2m.medium.

Perbaikan bug/Perbaikan

- Mekanisme coba lagi yang ditingkatkan untuk operasi kriptografi.

Versi 5.16.2

Amazon Linux 2023

Unduh perangkat lunak versi 5.16.2 untuk Amazon Linux 2023 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum e223649cbf689afaeca445b962cceaab3c143b901ee7974a7a5360b6424c409b)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 9b51e8b1429f22705399299c111385e9434873a1dd26dda2c70498646590148c)
- [Penyedia JCE](#) (SHA256 checksum 45ec10c2e3d592fb1fe56cc9dee1d8653e3a6dfa15b8185618fd3cf66c567314)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM CLI](#) (SHA256 checksum 8134a0e316286310eda1606b6a14e1c9fd6f0e27934a43e55e24d3094ef870a6)

Unduh perangkat lunak versi 5.16.2 untuk Amazon Linux 2023 tentang arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum 7556c69294d0c67b359cf9995ec7047cdc90e21542d2c9d5e110e35f13c329c5)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 6ab4f3b9731a6f26891658f5e3a0ff74255096390c42aa7f702ba07e0bd924b1)
- [Penyedia JCE](#) (SHA256 checksum 98b80696762a73f3f9fbe6c3c7af0f8f9db23ba63d66d4d941690b6698019702)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM CLI](#) (SHA256 checksum 649397606f527f330049cf7adb5ad667126dfd7f63f878f1f46b469e2f127814)

Amazon Linux 2

Unduh perangkat lunak versi 5.16.2 untuk Amazon Linux 2 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
073ee736ad9c7de6e6451afdecc4d77e6ed1a63f314a7fc09373ff35986fa0fc)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
b174fb24a15f1e22dd520ea9ac4ac14229c2adce14ac6df8901f9d3c0e7e24f6)
- [Penyedia JCE](#) (SHA256 checksum f18e42f2679ab0fb51c421965a5dd6c8fe1361442d92403d1d230bedb003b27a)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM CLI](#) (SHA256 checksum 5a6c83833b60017dd494c31fd4abf3896f57177be7951674c85d0d4e909d2041)

Unduh perangkat lunak versi 5.16.2 untuk Amazon Linux 2 tentang ARM64 arsitektur:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
df2c97a27855916983f242eba05e0741335158432ac46ebbeae47c4a3b7b0bef)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
d25456e3da39c63d9c0d4a928815f995239b146f4f0b26a8c50c742905543207)
- [Penyedia JCE](#) (SHA256 checksum 7e31734efbaae49203e57d7ad4d1e291faa5b7e16c469a3596da82a7380bceeb)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM CLI](#) (SHA256 checksum b77eb1f9ff58ccc864854d7f261e331ce50836cd770efc8e8eaeabb33d78c23e2)

RHEL 10 (10.0+)

Unduh perangkat lunak versi 5.16.2 untuk RHEL 10 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
8044cb40c9c65ae944900c740308e68eeb0156d3b93051d60463cd3f6095ef67)
- [Penyedia JCE](#) (SHA256 checksum a1b7f110c2b15ba7ce9adee74baf1f108d85146269321991314f9153a074ca27)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM CLI](#) (SHA256 checksum 3ded29fdb74537b284cf871af91dd64fce8483770141a3e4470e94fb69524621)

Unduh versi 5.16.2 perangkat lunak untuk RHEL 10 pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum 643fa23ffea47dfb21fd326cacf99e0e45f44d767ca6198ab06023239eda5d2c)
- [Penyedia JCE](#) (SHA256 checksum e6b5c2760c60f5c82a248d36f3b47cea9b1bbb743cf1c02a1c13554b70ab87f8)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM CLI](#) (SHA256 checksum a6af281ff9f429f964ac9d2babc251b4423748d2b2e6da22fc51b9fd5c59714c)

RHEL 9 (9.2+)

Unduh perangkat lunak versi 5.16.2 untuk RHEL 9 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum e0a44849c9cb562cf803ff182e54a35e7eee682efa2a99dfac51a996822d126)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 97463a8a88239d4a81854f2a4da7e0371dbcb375a662c7eae7753c371ef3c39f)
- [Penyedia JCE](#) (SHA256 checksum 10cd4b9826451b7e9a11390cabcc7df548f5b3197e8bc87f5efe04605e503dc1)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM CLI](#) (SHA256 checksum 6c7fc59dcce4d7bc0c4460ae7c793f13fcb306924942bc6e1457f729602f76f)

Unduh versi 5.16.2 perangkat lunak untuk RHEL 9 tentang arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum 7b8d3f5e2bd09f30ddc1599619c63aa4452b16e1037bf10af070c055ae48cbd7)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 4edb0315c050b3b0699cb244fe36b987c9fd2921999cf7519fb669354d2efe2d)
- [Penyedia JCE](#) (SHA256 checksum eae2064d9523df9f819ef4c7950a0ad15ef8bb01d8dcec062ee2558f5bebf029)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM CLI](#) (SHA256 checksum f9f4073ebcde2b654b706efb1c0f99c08fe24903576e760029097b0e3c783f6a)

RHEL 8 (8.3+)

Unduh perangkat lunak versi 5.16.2 untuk RHEL 8 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
9a3e023fd5ac8c444ef07bb28b96e81b467e066c9dc2da19f648f38eb04d05c9)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
6fc6ca58f16fe2f486f4730cd0a6029eeebae98b3aed41fd8c605fe5c36c7a0)
- [Penyedia JCE](#) (SHA256 checksum cb6e9eccfeebca7377b1fb815c2ae306b39b271513ce73a6cd5f1dcc55eac4c)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM CLI](#) (SHA256 checksum e848f3f2221e7f6b984bd2001331f9e90f20e6d7351484e4e2f12431c00d87e5)

Ubuntu 24.04 LTS

Unduh perangkat lunak versi 5.16.2 untuk Ubuntu 24.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
d993f8c84d15cf93432b73cb37693c4ab68b3e765c35742adef3f947744b9893)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
055f0e4cc74ec7ead77fff3970422b4f1729e26a2cf3cc2be530f0e6a06f887e)
- [Penyedia JCE](#) (SHA256 checksum 5bd62f8f5cb3ff4ae9430020aa4f5c8a29cb635674d2400d989f512c370c755c)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM CLI](#) (SHA256 checksum de263f6035cf7280a982434448aa079164050e4a09508816fd06bf000cf15f0c)

Unduh perangkat lunak versi 5.16.2 untuk Ubuntu 24.04 LTS pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum
a6496f28f82f3e92b0acc8e3ae1fbe230a6d3ed82586e8ca6fe6d2f82c275e7c)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
82275d6ac2628b4291162e3609d6788b4cf01d6488a9b6a84ed8787a5a96e873)
- [Penyedia JCE](#) (SHA256 checksum 67800957c69efca0fe01623364cb3b152f2780d15624955130d507a3f591e8e5)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)

- [CloudHSM CLI](#) (SHA256 checksum 088f53d75250aa0ce3db7a7ba3f06ad7804b3dbeed6226683a3252890a867ac6)

Ubuntu 22.04 LTS

Unduh perangkat lunak versi 5.16.2 untuk Ubuntu 22.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum b9799ba75fb8885782283abae7f49aa76a72d27f692654dd3a415157f0ea9043)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 3c7675e9b7fa4bb692865559f09fc1f9fd78dadb97d8bd6b5d1fc03367d75b86)
- [Penyedia JCE](#) (SHA256 checksum 37c21821ad4d9fde9551112e8a6d7ec6b7bb62a540c0c08b6edd650d5c87b5cf)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM CLI](#) (SHA256 checksum 7d3a724892e993aebfb06ad5c3d1ef8fc650d65dff0a7de3cb9c17c089badc58)

Unduh perangkat lunak versi 5.16.2 untuk Ubuntu 22.04 LTS pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum 8a2b392e48b7f9021966f6c457444ddd0964f9bb85b7ecb988bc0d372bbef6c9)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum cca0736d208d59d070ced8b82181e24919f21fbf0497e5a8a2f40436e06a8dd9)
- [Penyedia JCE](#) (SHA256 checksum c12e1b9bfe1cc78ecf35892050dd06904c6a29736cc628ac1481dca4ec3e3bfe)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM CLI](#) (SHA256 checksum 586ee7663d54e76652e9f073772fcbda218dc3c6432a221d664ffbabee3590cd)

Windows Server 2025

Unduh versi 5.16.2 perangkat lunak untuk Windows Server 2025 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum b5f55bc72fc994244f09e0252c7fc4a3ddf9607b20860c8718ff1d8a3db0a177)
- [Penyedia JCE](#) (SHA256 checksum c28ea8164f277bf067f4630d433ab7090b370225116a2d594dd563dfb5f9f991)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)

- [CloudHSM CLI](#) (SHA256 checksum db708fc33bdc91b9ad554aa63de86a572d3ccff4ba3ab10feac9c5fd9ed79b94)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum c060be59673f325240bd0f1ccc9d9d97639eb7937fa323d71e223451bc0d2e19)

Windows Server 2022

Unduh versi 5.16.2 perangkat lunak untuk Windows Server 2022 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum b5f55bc72fc994244f09e0252c7fc4a3ddf9607b20860c8718ff1d8a3db0a177)
- [Penyedia JCE](#) (SHA256 checksum c28ea8164f277bf067f4630d433ab7090b370225116a2d594dd563dfb5f9f991)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM CLI](#) (SHA256 checksum db708fc33bdc91b9ad554aa63de86a572d3ccff4ba3ab10feac9c5fd9ed79b94)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum c060be59673f325240bd0f1ccc9d9d97639eb7937fa323d71e223451bc0d2e19)

Windows Server 2019

Unduh perangkat lunak versi 5.16.2 untuk Windows Server 2019 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum b5f55bc72fc994244f09e0252c7fc4a3ddf9607b20860c8718ff1d8a3db0a177)
- [Penyedia JCE](#) (SHA256 checksum c28ea8164f277bf067f4630d433ab7090b370225116a2d594dd563dfb5f9f991)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM CLI](#) (SHA256 checksum db708fc33bdc91b9ad554aa63de86a572d3ccff4ba3ab10feac9c5fd9ed79b94)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum c060be59673f325240bd0f1ccc9d9d97639eb7937fa323d71e223451bc0d2e19)

Windows Server 2016

Unduh versi 5.16.2 perangkat lunak untuk Windows Server 2016 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum b5f55bc72fc994244f09e0252c7fc4a3ddf9607b20860c8718ff1d8a3db0a177)

- [Penyedia JCE](#) (SHA256 checksum c28ea8164f277bf067f4630d433ab7090b370225116a2d594dd563dfb5f9f991)
- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 17029e68946b83d47c8512f824599338e7315a99317e4fc33a2058cdc5512d0d)
- [CloudHSM CLI](#) (SHA256 checksum db708fc33bdc91b9ad554aa63de86a572d3ccff4ba3ab10feac9c5fd9ed79b94)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum c060be59673f325240bd0f1ccc9d9d97639eb7937fa323d71e223451bc0d2e19)

Client SDK 5.16.2 menambahkan dukungan untuk platform RHEL 10 dan menghapus dukungan untuk Ubuntu 20.04 LTS. Client SDK 5.16.2 juga menambahkan dukungan untuk JCE Provider dan KSP dengan fitur-fitur baru, sambil memberikan perbaikan bug dan peningkatan di lainnya. SDKs

Dukungan platform

- Menambahkan dukungan RHEL 10 untuk arsitektur x86 dan ARM.
- OpenSSL Engine tidak didukung di RHEL 10 sebagai bagian dari transisi Red Hat ke arsitektur berbasis Provider yang lebih baru yang diperkenalkan di OpenSSL 3.0.

CloudHSM CLI

- Memperbaiki bug di mana penandatanganan dengan data pra-hash tidak menghapus token dalam mode interaktif.
- Memperbaiki bug quorum token-sign list yang sebelumnya menyebabkan kesalahan ketika tiga atau lebih token hadir.

Penyedia JCE

- Menambahkan dukungan untuk menemukan tombol Elliptic Curve (EC) menggunakan ECParameters dan ECPoint untuk tipe hsm2m.medium.
- Ditambahkan dukungan sharedInfo parameter untuk ECDH dengan X9.63 KDF.

Pustaka PKCS #11

- Menambahkan dukungan untuk menemukan tombol Elliptic Curve (EC) menggunakan ECParameters dan ECPoint untuk tipe hsm2m.medium.

Perbaikan bug/Perbaikan

- Fungsionalitas daftar token yang ditingkatkan untuk menampilkan lebih banyak token dalam satu operasi.
- Memperbaiki masalah pembersihan token penggunaan kunci dalam mode interaktif CloudHSM CLI saat menandatangani dengan data pra-hash.
- Menyelesaikan masalah yang diketahui di mana beberapa operasi pembatasan HSM pada hsm2m.medium tidak akan dicoba ulang secara otomatis. Lihat [Masalah: Operasi dapat gagal selama pembuatan cadangan](#) untuk detailnya.

Versi 5.16.1

Amazon Linux 2023

Unduh perangkat lunak versi 5.16.1 untuk Amazon Linux 2023 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
342e81846436708cfc1fb459a7ef1c7b065d8a68b50a5e63653c24918d0338a0)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
4e83bebcd20201c04629a03b70136df8d225d3056346789054c4e2ce8c9b3cba)
- [Penyedia JCE](#) (SHA256 checksum 014e3c804a56087f8855ed30e480f9c193def0e1132a4769eae3ecd235d76c4f)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM CLI](#) (SHA256 checksum 854c5c21f9fbef4f53a710340f213c272365c2bb1233106266eca5a5aa547e11)

Unduh perangkat lunak versi 5.16.1 untuk Amazon Linux 2023 tentang arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum
dbe9498d91e1e2e8f80fc870a4a6576b0dc801dc20cb7a82bfdded79b3a362ac0)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
76cff941f36275163db146e05bf5fe64440248643ceb5dfcb3b64103f4f016a7)
- [Penyedia JCE](#) (SHA256 checksum 8c38082797172b5630c74a8fc0c9e2652f0898bd2232b72911fbbaaa0260b5a6)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM CLI](#) (SHA256 checksum 2a0640e32405a7db138afb2444a542eab11c0a7d9610d5406478c7ac4602a14b)

Amazon Linux 2

Unduh perangkat lunak versi 5.16.1 untuk Amazon Linux 2 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
cfa59458ec239553c86ac55773bebc69a5dc7cdc08a3927917fd5918e10abe93)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
49e50bece7d73f3b7dc95eed1df856a6dee40e27f24f40f015c6a4a2e8dee839)
- [Penyedia JCE](#) (SHA256 checksum 85c8853860aa36a6c54e75c94d607a334d64dd34683ba70430ff5be61eedcc56)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM CLI](#) (SHA256 checksum 0efc3783e8429331aa446e45776fef46c7f9726a4768763ba4881dc2b05e090e)

Unduh perangkat lunak versi 5.16.1 untuk Amazon Linux 2 tentang ARM64 arsitektur:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
dad60a0380ef0ea9c469cae4de10dc47124bbecedafb6956d009734cd49abceb)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
7ee303421d94544cbe9df03022c48af327833e631d8f1ec59a466673b6e9395d)
- [Penyedia JCE](#) (SHA256 checksum 5ebe8157abb042ee92a7edfbfcb98bf79fc3a9907684565176fd28c387c08e88)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM CLI](#) (SHA256 checksum 5e76b4a9021a3c92e59fc608c8263af731013835738977f376fb8ad9189add56)

RHEL 9 (9.2+)

Unduh perangkat lunak versi 5.16.1 untuk RHEL 9 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
696e58f74d89bd8e39f85bddc547f8c94fa2fbca498318a7bb94f22e8be7a668)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
86197cd8bdd70db91331bb8380ea094352b4087c95a04768d2cefc3bba18dffa)
- [Penyedia JCE](#) (SHA256 checksum 37ccc2df3e8aadda74dc060f7f5bbe63e4769311f2ccbf313a8ab5d8831f206)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)

- [CloudHSM CLI](#) (SHA256 checksum 45a6db22db9ae64c239b49ea9281d29d2b4364dd63ee2408f463de17c4da1877)

Unduh perangkat lunak versi 5.16.1 untuk RHEL 9 tentang arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum fd79fbccf504f53a6e70497135fcea4e20982fca75b628124c7510dcbaddf56c)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 23fe7407be9c5f8da0ae64e560a9741887bc31936f0b88c8d1490e3c6893a8a8)
- [Penyedia JCE](#) (SHA256 checksum bd1c43e3e28c2f71e78a41342c064ff873fd8e052f89f90227fe00eaa28f089e)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM CLI](#) (SHA256 checksum 3e877a232303052b8a6b2b869dcf228edbb0da913d3d41393f622831f4455a27)

RHEL 8 (8.3+)

Unduh perangkat lunak versi 5.16.1 untuk RHEL 8 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum d396c53c229b1eaa7cb30d4fcd17addc9170c7942795d1c82b23a157ec379686)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum c01f2cb66a6c5be839906b25c6a7f7990507b1b8ec3da34c128c1a90838e48df)
- [Penyedia JCE](#) (SHA256 checksum 1745ab78de0712111659e91301f80e771e1b7ebc2ff8c4cc5e37bce591058520)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM CLI](#) (SHA256 checksum b3ca40ba66062856ef63848c71f6e9dfa0a46a2b18d44c2d96a798fc4a4fb9cf)

Ubuntu 24.04 LTS

Unduh perangkat lunak versi 5.16.1 untuk Ubuntu 24.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum ade113ee72547cb6a8fe91a4f3ac403f462d0acd0a306f01d16ac90699f9b59c)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 9e00cb32b16fbb286a853f65c5c5154c3e805ad67820d829a0c5343f12cc9e00)
- [Penyedia JCE](#) (SHA256 checksum 4fab322a138e14372aba394de29f444af640479dd338539be84e0cf659c8993b)

- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM CLI](#) (SHA256 checksum f8c1e369885eb77ba4517388dab47c067a617e860fb67a028a1ebde0f96acaef)

Unduh perangkat lunak versi 5.16.1 untuk Ubuntu 24.04 LTS pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum
5e15a69f8fb1429ed5c10429c91323fe82a63c0fe7bcc2f968820733d1737449)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
f5136ee61bd34b74d59a01d37964b70a0f80ec5981f732b265af1c5466309e6f)
- [Penyedia JCE](#) (SHA256 checksum acf5bebe7009461cdc1e31b95caadcbf80da09ec78d8f69142072da9baf61b79)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM CLI](#) (SHA256 checksum 511376d2faf9f991f2193d08f80fc1edd53bf06daaa6fd8bd7a3fba0d6563ad1)

Ubuntu 22.04 LTS

Unduh perangkat lunak versi 5.16.1 untuk Ubuntu 22.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
e1d787f10ee51d94732732811e4d2110f1b8e448e67fd47df8b53a2f7e56e3c8)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
29a9c41379754ce098a025feadc026f2f75a8638981f5b95ed07007d5b3d8510)
- [Penyedia JCE](#) (SHA256 checksum 390d21e499ba181dc2774c3140433f6e60b49d9de3a892a89038c27e1aff157b)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM CLI](#) (SHA256 checksum cadadafaa9464f944e5c2446df92fb1777a9e1e5116cab1b359a8aa101cf7ae8)

Unduh perangkat lunak versi 5.16.1 untuk Ubuntu 22.04 LTS pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum
3b6a3d50b4a55c150f12308ee3914451897bc4cc4c8420accf1d2d706316fccc)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
c366a3c5faf4de32a6dedc4613234ba3d331b0abac3241bed7d25e0109a44f64)
- [Penyedia JCE](#) (SHA256 checksum 705416b47bc62781b5a6dfc78dd02dadb4f18f842b569c3e210b9535813b9e70)

- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM CLI](#) (SHA256 checksum 811b501615cf34665b70f103905094ca84ed7c126f72d91b040aacda99dbf22b)

Ubuntu 20.04 LTS

Unduh perangkat lunak versi 5.16.1 untuk Ubuntu 20.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 6a2c33c78ada33fb435bbf8939b7cde3efe968e9f03250083dc6024b7ebd45b8)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 6ee178454e78d88e0ab92cd7c5b056d0c04cd6de192aea731d7ebcbd4c5ed761)
- [Penyedia JCE](#) (SHA256 checksum 1e61ccd8d2b37c64467051d26e8d8bd592465ecef04282d6f4a9491707ba059d)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM CLI](#) (SHA256 checksum 645693f2ede4fa0d7879eedaa41ae23120d902f24c9a637ba088f277e703cb96)

Windows Server 2025

Unduh perangkat lunak versi 5.16.1 untuk Windows Server 2025 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 25beadaf28323412c60ed6695d89b6261d34da9cccf08869923b979854aa0329)
- [Penyedia JCE](#) (SHA256 checksum 65f487f22c0786b80d81b387f5f4c8a6c9395c8d31a020c91fc0105829b8ec2c)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM CLI](#) (SHA256 checksum d4dc9f5efd2b5e2eb07a59fb237f7f8dfd5476c859cdb9f91841354ce88783c4)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum 80f70eb3ba22d34e49b5b5da3fa183c86c751bff46229ae16fb83a5fc69a4d0c)

Windows Server 2022

Unduh versi 5.16.1 perangkat lunak untuk Windows Server 2022 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 25beadaf28323412c60ed6695d89b6261d34da9cccf08869923b979854aa0329)

- [Penyedia JCE](#) (SHA256 checksum 65f487f22c0786b80d81b387f5f4c8a6c9395c8d31a020c91fc0105829b8ec2c)
- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM CLI](#) (SHA256 checksum d4dc9f5efd2b5e2eb07a59fb237f7f8dfd5476c859cdb9f91841354ce88783c4)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum 80f70eb3ba22d34e49b5b5da3fa183c86c751bff46229ae16fb83a5fc69a4d0c)

Windows Server 2019

Unduh perangkat lunak versi 5.16.1 untuk Windows Server 2019 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 25beadaf28323412c60ed6695d89b6261d34da9cccf08869923b979854aa0329)
- [Penyedia JCE](#) (SHA256 checksum 65f487f22c0786b80d81b387f5f4c8a6c9395c8d31a020c91fc0105829b8ec2c)
- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM CLI](#) (SHA256 checksum d4dc9f5efd2b5e2eb07a59fb237f7f8dfd5476c859cdb9f91841354ce88783c4)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum 80f70eb3ba22d34e49b5b5da3fa183c86c751bff46229ae16fb83a5fc69a4d0c)

Windows Server 2016

Unduh versi 5.16.1 perangkat lunak untuk Windows Server 2016 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 25beadaf28323412c60ed6695d89b6261d34da9cccf08869923b979854aa0329)
- [Penyedia JCE](#) (SHA256 checksum 65f487f22c0786b80d81b387f5f4c8a6c9395c8d31a020c91fc0105829b8ec2c)
- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 70320f9aee48a250f9384f626fc16db00ea35c32af4c29cd63e9c82065a0fe1b)
- [CloudHSM CLI](#) (SHA256 checksum d4dc9f5efd2b5e2eb07a59fb237f7f8dfd5476c859cdb9f91841354ce88783c4)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum 80f70eb3ba22d34e49b5b5da3fa183c86c751bff46229ae16fb83a5fc69a4d0c)

Client SDK 5.16.1 menambahkan dukungan untuk menandatangani dan memverifikasi data pra-hash di CloudHSM CLI.

Dukungan platform

- SDK 5.16.1 adalah rilis terakhir yang menyediakan dukungan platform Ubuntu 20.04 LTS. Untuk informasi lebih lanjut, lihat situs [web Ubuntu](#).

CloudHSM CLI

- Menambahkan dukungan untuk menandatangani dan memverifikasi data pra-hash di CloudHSM CLI. Untuk informasi selengkapnya, lihat [Kategori tanda krypto di CloudHSM CLI](#) dan [Kategori verifikasi krypto di CloudHSM CLI](#).

Penyedia JCE

- Mode AES/CBC/Pkcs5Padding Enkripsi yang diperbarui untuk secara otomatis menghasilkan Vektor Inisialisasi acak (IV) saat null IV disediakan. Sebelumnya, null IV akan mengakibatkan kegagalan operasi. Eksplisit IVs tetap wajib untuk operasi dekripsi.

Perbaikan bug/Perbaikan

- Mengurangi latensi untuk operasi pembaruan intisari berulang di JCE.
- Memperbarui perintah `generate kunci-reference` untuk memberi nama file referensi kunci KSP dengan benar berdasarkan nilai ID Atributnya saat berjalan di Windows Server. Untuk informasi selengkapnya, lihat [Menghasilkan referensi kunci KSP \(Windows\)](#).

Versi 5.16.0

Amazon Linux 2023

Unduh perangkat lunak versi 5.16.0 untuk Amazon Linux 2023 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
d63271304f32f49838390a58d94a2140ae8a744ac64efcf0e6a65983c858d862)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
25d78c1df82355601ed6887bb47d64a06380d001da15a070cfa89dce65417fb6)
- [Penyedia JCE](#) (SHA256 checksum fcd876813e5d114e2b042765ff98b256410e988a4e373ea1aae6f260aebdd554)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)

- [CloudHSM CLI](#) (SHA256 checksum 7f1441eb08daa7ceab91161900d89974ee86e0a3c54d878a29e43db775bd04f5)

Unduh perangkat lunak versi 5.16.0 untuk Amazon Linux 2023 tentang arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum 0b42e4e89cddda10c5be5f295a52d7a541bba057a3a07d3d9b192cf9eb49c776)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum e8ee949d44b9f9ba64ed36eb7944acacbd1d76d7cf09d5492daae5928a9953e9)
- [Penyedia JCE](#) (SHA256 checksum 67a6fd8f7b5dfc60ec6d6d380c31035675373b542205ec7e5a687175191cd275)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM CLI](#) (SHA256 checksum 1e6c3053b19aded54045a3baa9fe2545f81a35332366cf2d77dca92ee888e654)

Amazon Linux 2

Unduh perangkat lunak versi 5.16.0 untuk Amazon Linux 2 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 5a89ff9801f89f51e27e70869f2713cfe9bbd87d1198246cfaafe74aff9e809)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 991b6288289d07972915aa0be6dfd6b3c33c1d6312bc304225715e30832e688f)
- [Penyedia JCE](#) (SHA256 checksum c5e673f79c2efd83195a288a5c8860c4bf74dba5bc8e422741dd5cc1be230fb3)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM CLI](#) (SHA256 checksum b9021b0b348c08f433e7cce247ab1710e2a79f33a82ac36e1d6a4672171028db)

Unduh perangkat lunak versi 5.16.0 untuk Amazon Linux 2 tentang ARM64 arsitektur:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 97677a7f9759637b548205d31c9a08a3e7b27dfa5a38e9a4e8e2398ae9869a5a)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum dde369dc4efd9f59e5c9a5459a532ad5d273f87839fba6dd5dcde2c3a2b61517)
- [Penyedia JCE](#) (SHA256 checksum b7d96e7453e26ff07ccd9437b102f4c21857c79daa6315054728fc7ac66150bc)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)

- [CloudHSM CLI](#) (SHA256 checksum c7386ac016b14ea8cb96a6ed655a67f46826cdb925bd87902277b5ce9b593d51)

RHEL 9 (9.2+)

Unduh perangkat lunak versi 5.16.0 untuk RHEL 9 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 178d3e1ff0ec8cdc7f8c8be9aab772b4195602bb1c48dea692b628689be5ea3d)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 0488ad66cd825fafee70d1a1ccb1c8045b089af7b7d6ccadbec75f535376048c)
- [Penyedia JCE](#) (SHA256 checksum cc9a60fa981a46c674de1e715d92ec1b8cd5c801394bfb3f0d101880589d4202)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM CLI](#) (SHA256 checksum cb7e3ce6704f015d5b9462f12e6ad5489366fb3e80f554a2bdf7f04e688be1a9)

Unduh perangkat lunak versi 5.16.0 untuk RHEL 9 tentang arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum 9567fb016b641821869919d4d6e9754ae08fd549381446640ff135a1e225b809)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 8c46a9dd908e77a0d8412e7050621d0c570e091845480815c230e26d1917227b)
- [Penyedia JCE](#) (SHA256 checksum 78a7c714aedd068e5e2f12f5044a6ce1618e6dfacbcdd91b4a9b505842dcc6c9)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM CLI](#) (SHA256 checksum d2340ef4092ca6c1fa1ac8b484550508a5d9d8670e663bef6e595a07f8151be3)

RHEL 8 (8.3+)

Unduh perangkat lunak versi 5.16.0 untuk RHEL 8 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 47bf628a4a2663b8e8c7207b43e5af1d05ed2247e85e1371773596c132c9a4ec)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 49dd44243863121f7781e0f4ac556828c746d85db723864ccf16ea5fb042846a)
- [Penyedia JCE](#) (SHA256 checksum c5e8a6664e6a8e5338a98b3845bed029f5936b517e6043aeb65ad58c3364eed8)

- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM CLI](#) (SHA256 checksum 71f2cd6bd8892cca8a845bfe575cdf20016dd7176e5216b76290e499c822a2f2)

Ubuntu 24.04 LTS

Unduh perangkat lunak versi 5.16.0 untuk Ubuntu 24.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
fd7e78050aba017b7dc859f0b1e6f354f56b22091cb262b328aad204064a2960)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
9d2a9e8049caa12c8027ad2963fa30f22e1032d4025e39a4e2d5985073c283f5)
- [Penyedia JCE](#) (SHA256 checksum ae19c13f55e01538e989e8693aaf3a5d998c60f1ed71e721a34404a228947c20)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM CLI](#) (SHA256 checksum 7ebdb3e8c4b8e01cc1e5dfe32ae18c9542d9c2940c77bd71ee102093aaf1d1c)

Unduh perangkat lunak versi 5.16.0 untuk Ubuntu 24.04 LTS pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum
cfd9526cf495e81a2477b464ca3c124a89d07b1a366a14b39eeb27d6b5e2404b)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
02a5cc2cd9442b7bf982bb3ae44c9b134758b46e68e46fe4729b4adf1320d75c)
- [Penyedia JCE](#) (SHA256 checksum 124d04f667a7222d624f528404a289405669f4e4a1983b8a7926758bf67e35ec)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM CLI](#) (SHA256 checksum b874556b34ff5f0d082ca19f62822cf97d6e7074ce3861d7577090f99cba0cbd)

Ubuntu 22.04 LTS

Unduh perangkat lunak versi 5.16.0 untuk Ubuntu 22.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
e36d9bf900e195e417db261034d820cb935bec5135a0aa332d46019c70257cac)

- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 2c39129a5acce96b8693ce8c4b4aa174d9c13bc8ea3c2652fa505b4e0a933842)
- [Penyedia JCE](#) (SHA256 checksum 03b70d834985a2500b73eecd831bff4f30389eb67460bf1d799466e59a9f9cfd)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM CLI](#) (SHA256 checksum 3fc4fc662188ebbd430d9b5e01a57774567b926f3478d0a455fee4968c47460d)

Unduh perangkat lunak versi 5.16.0 untuk Ubuntu 22.04 LTS pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum c835d9bb604b5c47a702fd45267ffb9dfa37d68dd0734391f060835ea680a169)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 943213a0f14b50fbdf3a90286e7f7ab219ab77750698f5062013873419484a04)
- [Penyedia JCE](#) (SHA256 checksum 0ecc3e65a3b94c2dd3913f44aba58764e5558a8bedd3b742a42e4769533f3734)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM CLI](#) (SHA256 checksum 8558328b224b850cf9e1b3c4f8773565a90e38ab43e2c9e90bc5b841f98898d2)

Ubuntu 20.04 LTS

Unduh perangkat lunak versi 5.16.0 untuk Ubuntu 20.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 64a01e84d898aca3cd03f22022cf8dbbc806a1735a84df5f820ba5bfa3339b0e)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 53a73cfc60c3fddd3ad9173bf6e602faf289f20935b3c469a137847148442954)
- [Penyedia JCE](#) (SHA256 checksum 09544cb8129af4d06a241e1fbc0b6e6c207e1a518e5cb202f7b6c9802507d1b8)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM CLI](#) (SHA256 checksum e1fdf1e014541f57d3ba4688bad88e7557e32ba974e020b4da76f85f1fc6aa29)

Windows Server 2025

Unduh perangkat lunak versi 5.16.0 untuk Windows Server 2025 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 6d3142d178127372de9ab40e7af5dc81fa7c627a18286dd521ad786d0947f17d)
- [Penyedia JCE](#) (SHA256 checksum 9f5c28583b5127b90e753dd7ce6d081bbe4810c0c30c424192704efa6ad1be34)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM CLI](#) (SHA256 checksum 22cfed533f528e0975f452af51d334baa1b5c143e722182dca107622e703b1ca)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum 9045c4f3e81093c4b49b93f3ea3f5caf2a4e6980628e0db2d971c437dc203bfc)

Windows Server 2022

Unduh perangkat lunak versi 5.16.0 untuk Windows Server 2022 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 6d3142d178127372de9ab40e7af5dc81fa7c627a18286dd521ad786d0947f17d)
- [Penyedia JCE](#) (SHA256 checksum 9f5c28583b5127b90e753dd7ce6d081bbe4810c0c30c424192704efa6ad1be34)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM CLI](#) (SHA256 checksum 22cfed533f528e0975f452af51d334baa1b5c143e722182dca107622e703b1ca)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum 9045c4f3e81093c4b49b93f3ea3f5caf2a4e6980628e0db2d971c437dc203bfc)

Windows Server 2019

Unduh perangkat lunak versi 5.16.0 untuk Windows Server 2019 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 6d3142d178127372de9ab40e7af5dc81fa7c627a18286dd521ad786d0947f17d)
- [Penyedia JCE](#) (SHA256 checksum 9f5c28583b5127b90e753dd7ce6d081bbe4810c0c30c424192704efa6ad1be34)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM CLI](#) (SHA256 checksum 22cfed533f528e0975f452af51d334baa1b5c143e722182dca107622e703b1ca)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum 9045c4f3e81093c4b49b93f3ea3f5caf2a4e6980628e0db2d971c437dc203bfc)

Windows Server 2016

Unduh perangkat lunak versi 5.16.0 untuk Windows Server 2016 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 6d3142d178127372de9ab40e7af5dc81fa7c627a18286dd521ad786d0947f17d)
- [Penyedia JCE](#) (SHA256 checksum 9f5c28583b5127b90e753dd7ce6d081bbe4810c0c30c424192704efa6ad1be34)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum bbbbe99e427b7b4d51d7018a97c4d44372db020e44fdbb76fd3954fb104010e3)
- [CloudHSM CLI](#) (SHA256 checksum 22cfed533f528e0975f452af51d334baa1b5c143e722182dca107622e703b1ca)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum 9045c4f3e81093c4b49b93f3ea3f5caf2a4e6980628e0db2d971c437dc203bfc)

Client SDK 5.16 menambahkan ECDH dengan dukungan X963 KDF pada tipe cluster hsm2m.medium untuk penyedia JCE dan pustaka PKCS #11. Client SDK 5.16 juga menambahkan dukungan untuk menghasilkan file referensi kunci KSP di Windows Server dengan CloudHSM CLI.

CloudHSM CLI

- Menambahkan dukungan untuk menghasilkan file referensi kunci KSP di Windows Server, untuk informasi lebih lanjut, lihat [Menghasilkan referensi kunci KSP \(Windows\)](#).

Penyedia JCE

- Menambahkan ECDH dengan dukungan X963 KDF pada tipe cluster hsm2m.medium, lihat. [Mekanisme yang didukung untuk penyedia JCE untuk AWS CloudHSM Client SDK 5](#)

Pustaka PKCS #11

- Menambahkan ECDH dengan dukungan X963 KDF pada tipe cluster hsm2m.medium, lihat. [Mekanisme yang didukung untuk pustaka PKCS #11 untuk SDK AWS CloudHSM Klien 5](#)

Perbaikan bug/Perbaikan

- Memperbaiki bug di mana kunci sesi tidak dibersihkan dengan benar selama pemutusan.
- Memperbaiki masalah di mana operasi logout mTLS yang berhasil di hsm2m.medium salah mengembalikan respons kesalahan.

- Memperbaiki masalah dengan kegagalan koneksi selama startup windows.
- Memperbaiki masalah saat alat info SDK menampilkan keluaran yang salah pada sistem Ubuntu.
- Memperbaiki masalah yang sebelumnya memungkinkan ECDSA dengan hash yang lebih lemah dalam mode FIPS.

Versi 5.15.0

Amazon Linux 2023

Unduh perangkat lunak versi 5.15.0 untuk Amazon Linux 2023 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 41ef3178811df1dbb03b2cbac83fe0f4768bdc9b17005c409f1c0229f93ef11c)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum afa1f9f8bd99f54866dea1b8928c00b951a6e492f5f36d0d6c7c38fff341d609)
- [Penyedia JCE](#) (SHA256 checksum 6ea775e05570ef3497a4df5c35a6ec1c682aea73c48e7fec3e541af995759e)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM CLI](#) (SHA256 checksum 78c10bb213dd14fcfc5836e358de6aedac61db05125fd61137b0082214fdecbe)

Unduh perangkat lunak versi 5.15.0 untuk Amazon Linux 2023 tentang arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum cd0617d29b6d64c00c8e14fd9a92f604e14acda4746f4f0b86b9de42367192fb)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum bc9acfd04eb1246eb3d5b0a8f3736ec017c0d1699d5395f85868d4a1722cd83)
- [Penyedia JCE](#) (SHA256 checksum 96d05301486206577b7be05aac561649167b57272c9c06ff839fd8ff2b5d96d5)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM CLI](#) (SHA256 checksum b870eadb27736a2cde98022d57e9704c67ae15878cf0b910738859cdabaa35a2)

Amazon Linux 2

Unduh perangkat lunak versi 5.15.0 untuk Amazon Linux 2 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
c70ae4f0181a8187c9380481c51c1d03e12236dd86863ec818ed3f210b294c8e)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
08e9fd1dd80efa637f9a1727bb0de205ba124a3776b2e8bc21008ee458063a42)
- [Penyedia JCE](#) (SHA256 checksum 7932ed060e72c53b2556f30694b0ffe5342b244b6628c7a9dc03966aa49c8fe6)
- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM CLI](#) (SHA256 checksum 0a939e4d5d0a2ff308c7a1d9e73ebc865e426214d556bde1bd29dbe807fbb583)

Unduh perangkat lunak versi 5.15.0 untuk Amazon Linux 2 pada ARM64 arsitektur:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
bcf907a8fb4722b09c54b8e5785fe2b8cffcedd6ee3fdda2d21879a012138077)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
8f70edc3a6a4a1bf0264c6567b1dda1ac69055f206753f88eeadbb8bf3bf9f38)
- [Penyedia JCE](#) (SHA256 checksum 75dd67736bb08fe7e46e113af10803a255ba8edee3016ca963c1ee94fe59d43b)
- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM CLI](#) (SHA256 checksum 1c2ecf90c955281d99bcd8d1956d63debb15bbc8419744c83f88821ef8b78aee)

RHEL 9 (9.2+)

Unduh perangkat lunak versi 5.15.0 untuk RHEL 9 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
65bd0b815eebc806674a7bf7c54e9f884595881547f5fffd08ff6a38aabdcbe)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
f2af9f5882ab2e5a11defecc660f8af5c4d9d6e2e2b89873e6833fc2976f44ac)
- [Penyedia JCE](#) (SHA256 checksum 5124bace5d1544775c891f13a0e309b30dd73699116c46ecc9a77bba5f9cf633)
- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM CLI](#) (SHA256 checksum 1eec31fb2c0ab3c2839d9bdb37874cf8dbc74a383279068fce9e8613966d06e0)

Unduh versi 5.15.0 perangkat lunak untuk RHEL 9 pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum f498697519afa04b6b0362a4388a6a38e2cb6813b781d6ce3d97a7de89c5cbfe)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 5410da63108a1b209e567e9bccf8bd7e4035af88b0d58b9d78b10917be1b40c1)
- [Penyedia JCE](#) (SHA256 checksum b46f233e6994d2c0ed505dc5c717ee3009daaeb2063d260aa78e06273770bffa)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM CLI](#) (SHA256 checksum e9f93eaa58db2f7ea1174164ef96ab219700933d353243c3c6ab1aebac5ccffe)

RHEL 8 (8.3+)

Unduh perangkat lunak versi 5.15.0 untuk RHEL 8 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 87131e179d0e60ade302ec07b22803cfb39294bf060b786c41f154d95791ac94)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum f412a2f5cd761db5940288bb252ce060d44735c6b436bb6d4fa7d3687a44a026)
- [Penyedia JCE](#) (SHA256 checksum d0844f55c08f9ff9c393138a9041efe1b59dc3dce20c0b2c23406efe6acc43db)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM CLI](#) (SHA256 checksum 9fd8033e478ce6d7d640c063c4f007359cb04c19d519826a745ad0885f96a0f8)

Ubuntu 24.04 LTS

Unduh perangkat lunak versi 5.15.0 untuk Ubuntu 24.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum ca5f2f80ae921cfdbdc5c8bc35c39d2b19cfabfd5981932409eaf2e7c00a9097)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum e44cd7b678a421957c84e4fc0f70280360fd4e1e66f4cabdd1b20b955ee5fcca)
- [Penyedia JCE](#) (SHA256 checksum bc6382f12769c3d87c34522d0d616b7a8c108574c003814e1300938c386655ac)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM CLI](#) (SHA256 checksum f786be86c204680e670f7817376ff376733ebd96b51c31e9c98213078596637f)

Unduh perangkat lunak versi 5.15.0 untuk Ubuntu 24.04 LTS pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum
1ecf057d67137c0e9e1bb0dda57d581690c647b0360c3a617a8dc668919d08de)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
4f3d23e6b798f88be587108d8e6a225d796a3d080aef61ea384eb74a1270612a)
- [Penyedia JCE](#) (SHA256 checksum b9fd16bdcc1fcf59fde0d3e0debee500b0b7edfdff69209e84d14393097fe9d2)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM CLI](#) (SHA256 checksum a95922d9b44e64a64723db0c21ac89566515a5a6c87de990af4a9e1f40c7424f)

Ubuntu 22.04 LTS

Unduh perangkat lunak versi 5.15.0 untuk Ubuntu 22.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
966be12eb32de813ca07e766abf7b5616c0d2e105e9296d920aadaca10e5afdf)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
f2840151d87b7f9cbff68993c25397afd48a16f054abf0f2fd4624662d3087d6)
- [Penyedia JCE](#) (SHA256 checksum 01aebdda05640e50a82cae04c5cb6d33ab909dadd917bd834957d4f75ad8c577)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM CLI](#) (SHA256 checksum 2e5eba9409abc429828505779c512cd2424719766c0e488f50aee288966cf61d)

Unduh perangkat lunak versi 5.15.0 untuk Ubuntu 22.04 LTS pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum
614f74f101c9c64fff515128c5a59fa23de5047494fe248e8aabdb441092b3e)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
c15d6db77b76bce690749b73b947567eb5f2d76669887843116d0ce56c1f8ea7)
- [Penyedia JCE](#) (SHA256 checksum bca8511d5c0a173b0fef326016ce5091b6e6829fa2b3cb45ed5621290ae3e42a)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM CLI](#) (SHA256 checksum 2f5b673148b682d7e34619c51b5e8799abe7dc7fd4f046158a0d05320ba24dc1)

Ubuntu 20.04 LTS

Unduh perangkat lunak versi 5.15.0 untuk Ubuntu 20.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
86c8394b5ddff91a71194fb87c327efde36baa2380e559c04f9d543a6e74d61b)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
0a4227389fea61e6e7ac7cfa715eb341f7a4eeae9ed10e4c96da2c0dd4a18f9e)
- [Penyedia JCE](#) (SHA256 checksum 65b2d926ff9dfbe6c7864bc3a41b3da2383bc731dd199bcef8805a0543fbe612)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM CLI](#) (SHA256 checksum 09c2e55bcf72f9e530717950d8c5fd48574ae6ccb09a049526ca5b2a3b8aa9)

Windows Server 2025

Unduh perangkat lunak versi 5.15.0 untuk Windows Server 2025 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
a903b63fe286f15bf669c0555b1fa4d86b33592ed05af0809acac28c0d3ace16)
- [Penyedia JCE](#) (SHA256 checksum fdef6251f06d77d51fddbc2184d3eec87ddec4fe35b3ac620343eb66c95ddf64)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM CLI](#) (SHA256 checksum 6cc9205fe1fc514ddd9774f824e512940d6b1bc4cc0e251265bc46ab99746c28)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum
52ed9b08cd0ce100b8dcd3d8e8f411b6201f9f1b27872b19d1136c0bf36a29b8)

Windows Server 2022

Unduh versi 5.15.0 perangkat lunak untuk Windows Server 2022 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
a903b63fe286f15bf669c0555b1fa4d86b33592ed05af0809acac28c0d3ace16)
- [Penyedia JCE](#) (SHA256 checksum fdef6251f06d77d51fddbc2184d3eec87ddec4fe35b3ac620343eb66c95ddf64)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM CLI](#) (SHA256 checksum 6cc9205fe1fc514ddd9774f824e512940d6b1bc4cc0e251265bc46ab99746c28)

- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum
52ed9b08cd0ce100b8dcd3d8e8f411b6201f9f1b27872b19d1136c0bf36a29b8)

Windows Server 2019

Unduh perangkat lunak versi 5.15.0 untuk Windows Server 2019 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
a903b63fe286f15bf669c0555b1fa4d86b33592ed05af0809acac28c0d3ace16)
- [Penyedia JCE](#) (SHA256 checksum fdef6251f06d77d51fddbc2184d3eec87ddec4fe35b3ac620343eb66c95ddf64)
- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM CLI](#) (SHA256 checksum 6cc9205fe1fc514ddd9774f824e512940d6b1bc4cc0e251265bc46ab99746c28)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum
52ed9b08cd0ce100b8dcd3d8e8f411b6201f9f1b27872b19d1136c0bf36a29b8)

Windows Server 2016

Unduh versi 5.15.0 perangkat lunak untuk Windows Server 2016 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
a903b63fe286f15bf669c0555b1fa4d86b33592ed05af0809acac28c0d3ace16)
- [Penyedia JCE](#) (SHA256 checksum fdef6251f06d77d51fddbc2184d3eec87ddec4fe35b3ac620343eb66c95ddf64)
- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
ce5d92731f2a9c7f46c92bba92f3d0adff4ceb06b9d653597994100dfc352fbe)
- [CloudHSM CLI](#) (SHA256 checksum 6cc9205fe1fc514ddd9774f824e512940d6b1bc4cc0e251265bc46ab99746c28)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum
52ed9b08cd0ce100b8dcd3d8e8f411b6201f9f1b27872b19d1136c0bf36a29b8)

Client SDK 5.15 menambahkan dukungan untuk mereplikasi pengguna di seluruh kloning cluster dengan CloudHSM CLI. Client SDK 5.15 juga menambahkan paket instalasi untuk pustaka PKCS #11, penyedia JCE, CloudHSM CLI, dan Key Storage Provider (KSP) untuk Windows Server 2025.

Dukungan platform

- Menambahkan dukungan untuk Windows Server 2025 untuk pustaka PKCS #11, penyedia JCE, CloudHSM CLI, dan Penyedia Penyimpanan Kunci (KSP).

CloudHSM CLI

- Ditambahkan perintah berikut:
 - Replikasi pengguna dengan CloudHSM CLI

Versi 5.14.0

Amazon Linux 2023

Unduh perangkat lunak versi 5.14.0 untuk Amazon Linux 2023 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 05e7a3882166c694a7a09bc735f08f91c8145a4215176665eacacdf3e509abe8)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum f4dd9966988418e100c276dc0d521f91afdfc0e6c008dbf8eda446ebaca83c14)
- [Penyedia JCE](#) (SHA256 checksum c4dee5c1173f6a1c7683aedb58e61d329c36933435c416f288d94bc9a68a6b31)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM CLI](#) (SHA256 checksum 1fad069fde305450254287e43750d597db9af0cb8fd168300cd5eaed9e2af33a)

Unduh perangkat lunak versi 5.14.0 untuk Amazon Linux 2023 tentang arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum 9f34163d02bce26c1280e589310cda891d27995c50cec0a7fe083100ecff2b69)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 7444a4daad6e4715c82d6c39a7b03a07ee0201a13fe0f98da96acbf9d24abf6c)
- [Penyedia JCE](#) (SHA256 checksum 3393fe3a0f5c3a9c92106d74b7de074c818e095f97d2cfd600dbd47779b90b37)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM CLI](#) (SHA256 checksum d9a2edce48c5f6646d5a351cb431712f2d2fc62d21f8318e7bb1ce579819d7f4)

Amazon Linux 2

Unduh perangkat lunak versi 5.14.0 untuk Amazon Linux 2 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum 9c47b90bfa0ad51627cdb0dd8f148a56090fbdeb2490f1ab4009170c7b9c1120)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 215f9768331565085a317585b3dbe0514b251fdc428c96ed32491c4abb9fea56)
- [Penyedia JCE](#) (SHA256 checksum a802f941e95fcbf0ef37775fc096c0d6ae4c916fa08330a9d56defc5f99ff2b7)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM CLI](#) (SHA256 checksum ac10f23dd81264b1d10a0760f62b5006d2b2b66bf1c6378248ca9326afb65a83)

Unduh perangkat lunak versi 5.14.0 untuk Amazon Linux 2 tentang ARM64 arsitektur:

- [Perpustakaan PKCS #11](#) (SHA256 checksum f2d1550f043c655ef5d51e2ca14a0416886d99902e0702bd7f30f93b2c563d4d)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 2c8e2c81af53ba3646d1f947894d84b9b54780bbca79b58f354125e4ac9427c0)
- [Penyedia JCE](#) (SHA256 checksum 3f9c881056e6905d46358585db72143b59971958a708fa4ac75cb53994487213)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM CLI](#) (SHA256 checksum 6d5985e44c9852409dd3d342239fbbf7d0f9ad43d445e0980153c9b9eafe2b6f)

RHEL 9 (9.2+)

Unduh perangkat lunak versi 5.14.0 untuk RHEL 9 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum c4812210421ff2fa5dac8477a2e6b10552aabd88b1f03d717e044e7293823158)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 3f6aeaa6ae1faae7d8bba1596f358cad1eec9e562cf08aaab9ded92cabe94719)
- [Penyedia JCE](#) (SHA256 checksum bd8f120b08f738ad4d1534b0e32aad903758d5e75d3aba7cb9ff6a77dec533db)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)

- [CloudHSM CLI](#) (SHA256 checksum 84f13a9babd767edf90dabb0b14034ec5b2208898123a13966dd6b8519961c27)

Unduh perangkat lunak versi 5.14.0 untuk RHEL 9 tentang arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum 3faa1de563aa9773939cab7b39907dcb7b981ed7225019f9070a9dd52db7ae70)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 13b41ac47ef7ee7bf78f585b8347ca4da9ebc296e4fc1e6a0c2ff5b333354ca6)
- [Penyedia JCE](#) (SHA256 checksum 06803655ebe54d59c180bb17ac6fe56337bdd58ad0f6fe87c50ff8df32f70258)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM CLI](#) (SHA256 checksum e6b4e9688d0db9d72bbee3450fe19736d640c9931adbd4f4ef73cb7ac2a08cf4)

RHEL 8 (8.3+)

Unduh perangkat lunak versi 5.14.0 untuk RHEL 8 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum e400aeea6dbf7721f97e71c643a0db4f5f1094fb197fc46dc0ab293de9d16f2d)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum bfcc27d251e62f9eba0fd508e7d08dc62126642d4cdd0b5566183957768b8c54)
- [Penyedia JCE](#) (SHA256 checksum b5500031b572c918a8df4a0347e01c8ea00a7366b865b310bd92427fa1ed53e3)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM CLI](#) (SHA256 checksum 000712d0a691efc64a6c5d54bfeb1ab48b315ebb5dd6926b0502e326bf700291)

Ubuntu 24.04 LTS

Unduh perangkat lunak versi 5.14.0 untuk Ubuntu 24.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum bbec70a198a4b173620b4018accc297ce6a6a80d372706e0101997d6bca35bca)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum fcb77f75bd465b22401a09a20c410985833340295101263b7171cdcc4ac9f980)
- [Penyedia JCE](#) (SHA256 checksum e1fa16aae2f6095c89a8bf392ef2cb9ca4db8853c858904ff90abf4bb491d74b)

- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM CLI](#) (SHA256 checksum af32e0ac1c5f8c7f8cb40d255abe7c63ce9d981293187a3fb452562fa05756f4)

Unduh perangkat lunak versi 5.14.0 untuk Ubuntu 24.04 LTS pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum
1232318b084347889f92136536537ca519373683ce39bcec20f6ac3fa2f42d7c)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
1745ee3a33d8e6ea72644e903c55c6a206204cf0a8bea200bc4a7b15736ed801)
- [Penyedia JCE](#) (SHA256 checksum 7a44acabbc90c996594ed53661937f9242850823347f7c386f02fc041a97471a)
- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM CLI](#) (SHA256 checksum e6778bd12c55fd152b50033833531fe569472f4f2bd9927a345eb126e8305739)

Ubuntu 22.04 LTS

Unduh perangkat lunak versi 5.14.0 untuk Ubuntu 22.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum
6b4b1620e9a85267950633b171dd188b7ac7094e371e188fab1bef7a911a16f)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
fee5f0a65fab0f46ad58689af5dc510721581f31364d3be5cbbf79f5d9a60db8)
- [Penyedia JCE](#) (SHA256 checksum 5883a3d15e160d65f8e26f185e1ee30f68becad5f6cd16abc1c4586689800b5)
- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum
173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM CLI](#) (SHA256 checksum 7b1a8cd2962f4be7325154c080dabfb820beac5629e272c3a3ebc0e6cab11d27)

Unduh perangkat lunak versi 5.14.0 untuk Ubuntu 22.04 LTS pada arsitektur: ARM64

- [Perpustakaan PKCS #11](#) (SHA256 checksum
8680f63fd74272ea0e4d4c32dfb96b3eb9b60e03483c202d4e9b65ee101a178f)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum
c757304e8fc5f38be3bab7ac6d37a35dcb56d31e62f7194da62b3a176593d1d8)
- [Penyedia JCE](#) (SHA256 checksum a78ee59341da56af315d126ec7ed9f0dfe6e99649f659db2436be3863cb035b)

- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM CLI](#) (SHA256 checksum 18931bff869a0bd54846b3296d870fa19beba5652f979352be8fba6307e6d1aa)

Ubuntu 20.04 LTS

Unduh perangkat lunak versi 5.14.0 untuk Ubuntu 20.04 LTS pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum f1461c16b135ebcc17deec46aab88bd113ea122b8942fc188d4f05cd03e919a8)
- [Mesin Dinamis OpenSSL](#) (SHA256 checksum 89211a7a7ed50eda2dc385c31ea76f1fbabd389ca691204873531d983c3eb0f7)
- [Penyedia JCE](#) (SHA256 checksum f098c32d61a53b073459a75d88b68e377a9b16335874fae060cb10df0da00df0)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM CLI](#) (SHA256 checksum ff91fb930717c917344af2ba344dc6e02bd5abc004dcb6147e9412b67e2aa7ab)

Windows Server 2022

Unduh perangkat lunak versi 5.14.0 untuk Windows Server 2022 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum ee5a8e8e85fff7acd0bbafa23740e3981f7dc52e708972b600c2b26603786838)
- [Penyedia JCE](#) (SHA256 checksum 2ae0274f09f66981c03fd1e3c264e896ba7cd211168ea31369335db1b3ea2e77)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM CLI](#) (SHA256 checksum 3938654f88ce010042e48909e23991e178e411e2bd9e3c9ec25fcb8157c0cd55)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum b026e4d8c11e9ff6f22a7b9e10b8bb29e7572665f0d7978a3cef7d2354b7693f)

Windows Server 2019

Unduh perangkat lunak versi 5.14.0 untuk Windows Server 2019 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum ee5a8e8e85fff7acd0bbafa23740e3981f7dc52e708972b600c2b26603786838)

- [Penyedia JCE](#) (SHA256 checksum 2ae0274f09f66981c03fd1e3c264e896ba7cd211168ea31369335db1b3ea2e77)
- [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM CLI](#) (SHA256 checksum 3938654f88ce010042e48909e23991e178e411e2bd9e3c9ec25fcb8157c0cd55)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum b026e4d8c11e9ff6f22a7b9e10b8bb29e7572665f0d7978a3cef7d2354b7693f)

Windows Server 2016

Unduh perangkat lunak versi 5.14.0 untuk Windows Server 2016 pada arsitektur x86_64:

- [Perpustakaan PKCS #11](#) (SHA256 checksum ee5a8e8e85fff7acd0bbafa23740e3981f7dc52e708972b600c2b26603786838)
- [Penyedia JCE](#) (SHA256 checksum 2ae0274f09f66981c03fd1e3c264e896ba7cd211168ea31369335db1b3ea2e77)
 - [Javadocs untuk AWS CloudHSM](#) (SHA256 checksum 173087a8f286dc4f88dc915ecc00aa515bec0ae2faf219654df9f3422d8e83bb)
- [CloudHSM CLI](#) (SHA256 checksum 3938654f88ce010042e48909e23991e178e411e2bd9e3c9ec25fcb8157c0cd55)
- [Penyedia Penyimpanan Kunci \(KSP\)](#) (SHA256 checksum b026e4d8c11e9ff6f22a7b9e10b8bb29e7572665f0d7978a3cef7d2354b7693f)

Client SDK 5.14 menambahkan dukungan untuk penggunaan kunci yang dikontrol kuorum dan operasi manajemen kunci menggunakan CloudHSM CLI. Client SDK 5.14 juga menambahkan dukungan [Penyedia penyimpanan kunci \(KSP\) untuk AWS CloudHSM Klien SDK 5](#) untuk platform windows. Selain itu, Client SDK 5.14 menambahkan paket instalasi untuk pustaka PKCS #11, penyedia JCE, CloudHSM CLI, dan Penyedia Penyimpanan Kunci (KSP) untuk Windows Server 2022.

Dukungan platform

- Menambahkan dukungan untuk Windows Server 2022 untuk pustaka PKCS #11, penyedia JCE, CloudHSM CLI, dan Penyedia Penyimpanan Kunci (KSP).

CloudHSM CLI

- Menambahkan dukungan untuk penggunaan kunci yang dikendalikan kuorum dan operasi manajemen kunci.

Penyedia Penyimpanan Kunci (KSP)

- Menambahkan dukungan untuk Key Storage Provider (KSP), API kriptografi khusus untuk sistem operasi Microsoft Windows. Untuk informasi selengkapnya, lihat [Penyedia penyimpanan kunci \(KSP\) untuk AWS CloudHSM Klien SDK 5](#)

Versi 5.13.0

Client SDK 5.13 menambahkan dukungan untuk mengonfigurasi TLS timbal balik pada tipe cluster hsm2m.medium. Untuk informasi tentang penggunaan TLS timbal balik dengan CloudHSM, lihat. [Siapkan TLS timbal balik antara klien dan AWS CloudHSM \(disarankan\)](#) Client SDK 5.13 juga menambahkan paket instalasi untuk Ubuntu 24.04 LTS.

Dukungan platform

- Menambahkan dukungan untuk Ubuntu 24.04 LTS pada x86_64 dan arsitektur untuk semua. ARM64 SDKs

CloudHSM CLI

- Ditambahkan dukungan untuk pengguna admin untuk menjalankan [Replikasi kunci dengan CloudHSM CLI](#) perintah. Client SDK 5.12 memperkenalkan perintah replika kunci untuk digunakan oleh pengguna crypto.
- Ditambahkan perintah berikut:
 - [Kategori cluster mtls di CloudHSM CLI](#)

Perbaikan bug/Perbaikan

- Memperbaiki masalah untuk mengurangi waktu yang diperlukan klien untuk mendeteksi koneksi HSM yang tidak sehat, yang membantu mencegah kesalahan pemutusan koneksi selama lambda warm start.

Versi 5.12.0

Client SDK 5.12.0 menambahkan dukungan ARM ke beberapa platform dan peningkatan kinerja untuk semua. SDKs Fitur baru telah ditambahkan ke penyedia CloudHSM CLI dan JCE.

Dukungan platform

- Menambahkan dukungan untuk Amazon Linux 2023 pada ARM64 arsitektur untuk semua SDKs.
- Menambahkan dukungan untuk Red Hat Enterprise Linux 9 (9.2+) pada ARM64 arsitektur untuk semua. SDKs
- Menambahkan dukungan untuk Ubuntu 22.04 LTS pada ARM64 arsitektur untuk semua. SDKs

CloudHSM CLI

- Ditambahkan perintah berikut:
 - [Replikasi kunci dengan CloudHSM CLI](#)
- Menambahkan dukungan untuk menghubungkan ke beberapa cluster. Untuk informasi selengkapnya, lihat [Menghubungkan ke beberapa cluster dengan CloudHSM CLI](#).

Penyedia JCE

- Ditambahkan KeyReferenceSpec untuk mengambil kunci menggunakan KeyStoreWithAttributes.
- Ditambahkan getKeys untuk mengambil beberapa kunci sekaligus menggunakan KeyStoreWithAttributes.

Peningkatan kinerja

- Peningkatan kinerja untuk NoPadding operasi AES CBC untuk semua. SDKs

Versi 5.11.0

Client SDK 5.11.0 menambahkan fitur baru, meningkatkan stabilitas, dan menyertakan perbaikan bug untuk semua. SDKs

Dukungan platform

- Menambahkan dukungan untuk Amazon Linux 2023 dan RHEL 9 (9.2+) untuk semua. SDKs
- Dukungan yang dihapus untuk Ubuntu 18.04 LTS karena akhir masa pakainya baru-baru ini.
- Dukungan yang dihapus untuk Amazon Linux karena akhir hidupnya baru-baru ini.

CloudHSM CLI

- Ditambahkan perintah berikut:
 - [Kategori tanda krypto di CloudHSM CLI](#)
 - [Kategori verifikasi krypto di CloudHSM CLI](#)
 - [Impor kunci format PEM dengan CloudHSM CLI](#)
 - [Perintah buka kunci di CloudHSM CLI](#)
 - [Perintah bungkus kunci di CloudHSM CLI](#)
- [Ekspor kunci asimetris dengan CloudHSM CLI](#) sekarang mendukung ekspor kunci publik.

OpenSSL Dynamic Engine

- AWS CloudHSM OpenSSL Dynamic Engine sekarang didukung pada platform yang diinstal dengan perpustakaan OpenSSL versi 3.x. Ini termasuk Amazon Linux 2023, RHEL 9 (9.2+), dan Ubuntu 22.04.

JCE

- Menambahkan dukungan untuk JDK 17 dan JDK 21.
- Menambahkan dukungan untuk kunci AES yang akan digunakan untuk operasi HMAC.
- Ditambahkan atribut kunci baru ID.
- Memperkenalkan `DataExceptionCause` varian baru untuk kelelahan utama: `DataExceptionCause.KEY_EXHAUSTED`

Perbaikan bug/Perbaikan

- Meningkatkan panjang maksimal untuk `label` atribut dari 126 menjadi 127 karakter.
- Memperbaiki bug yang mencegah pembukaan kunci EC dengan mekanisme `RsaOaep`
- Menyelesaikan masalah yang diketahui untuk operasi `GetKey` di penyedia JCE. Lihat [Masalah: Kebocoran memori SDK 5 klien dengan operasi GetKey](#) untuk detailnya.
- Peningkatan login di semua SDKs untuk kunci Triple DES yang telah mencapai batas blok enkripsi maksimum mereka, per FIPS 140-2.
- Menambahkan masalah yang diketahui untuk OpenSSL Dynamic Engine. Lihat [Masalah yang diketahui untuk OpenSSL Dynamic Engine untuk AWS CloudHSM](#) untuk detail.

Versi 5.10.0

Client SDK 5.10.0 meningkatkan stabilitas dan menyertakan perbaikan bug untuk semua. SDKs

CloudHSM CLI

- Menambahkan perintah baru yang memungkinkan pelanggan mengelola kunci menggunakan CloudHSM CLI, termasuk:
 - Buat kunci simetris dan pasangan kunci asimetris
 - Bagikan dan batalkan berbagi kunci
 - Daftar dan filter kunci menggunakan atribut kunci
 - Tetapkan atribut kunci
 - Hasilkan file referensi kunci
 - Hapus kunci
- Pencatatan kesalahan yang ditingkatkan.
- Menambahkan dukungan untuk perintah unicode multi-baris dalam mode interaktif.

Perbaikan bug/Perbaikan

- Peningkatan kinerja untuk mengimpor, membuka bungkus, menurunkan, dan membuat kunci sesi untuk semua. SDKs
- Memperbaiki bug di Penyedia JCE yang mencegah file temp dihapus saat keluar.
- Memperbaiki bug yang menyebabkan kesalahan koneksi dalam kondisi tertentu setelah HSMs di cluster diganti.
- Format `getVersion` output JCE yang dimodifikasi untuk menangani nomor versi minor besar dan menyertakan nomor tambalan.

Dukungan platform

- Menambahkan dukungan untuk Ubuntu 22.04 dengan JCE, PKCS #11, dan CloudHSM CLI (dukungan untuk OpenSSL Dynamic Engine belum tersedia).

Versi 5.9.0

Client SDK 5.9.0 meningkatkan stabilitas dan menyertakan perbaikan bug untuk semua. SDKs Optimalisasi telah dibuat untuk semua orang SDKs untuk menginformasikan aplikasi kegagalan operasi segera ketika HSM ditentukan tidak tersedia. Rilis ini mencakup peningkatan kinerja untuk JCE.

Penyedia JCE

- Peningkatan kinerja
- Memperbaiki [masalah yang diketahui](#) untuk kelelahan kumpulan sesi

AWS CloudHSM rilis SDK Klien yang tidak digunakan lagi

Versi 5.8.0 dan sebelumnya tidak digunakan lagi. Kami tidak menyarankan menggunakan rilis usang dalam beban kerja produksi. Kami tidak menyediakan pembaruan kompatibel mundur untuk rilis usang, kami juga tidak meng-hosting rilis usang untuk diunduh. Jika Anda mengalami dampak produksi saat menggunakan rilis usang, Anda harus meningkatkan untuk mendapatkan perbaikan perangkat lunak.

Rilis Client SDK 5 yang tidak digunakan lagi

Bagian ini mencantumkan rilis Client SDK 5 yang tidak digunakan lagi.

Versi 5.8.0

Versi 5.8.0 memperkenalkan otentikasi kuorum untuk CloudHSM SSL/TLS offload with JSSE, multi-slot support for PKCS #11, multi-cluster/multi CLI, -dukungan pengguna untuk JCE, ekstraksi kunci dengan JCE, mendukung KeyFactory untuk JCE, konfigurasi coba lagi baru untuk kode pengembalian non-terminal, dan mencakup peningkatan stabilitas dan perbaikan bug untuk semua. SDKs

Pustaka PKCS #11

- Menambahkan dukungan untuk konfigurasi multi-slot.

Penyedia JCE

- Ditambahkan konfigurasi berbasis ekstraksi kunci.

- Menambahkan dukungan untuk konfigurasi multi-cluster dan multi-pengguna.
- Menambahkan dukungan untuk SSL dan TLS offload dengan JSSE.
- Menambahkan dukungan buka bungkus untuk AES/CBC/NoPadding.
- Menambahkan jenis baru pabrik utama: SecretKeyFactory dan KeyFactory.

CloudHSM CLI

- Menambahkan dukungan untuk otentikasi kuorum

Versi 5.7.0

Versi 5.7.0 memperkenalkan CloudHSM CLI dan menyertakan algoritma kode otentikasi pesan berbasis sandi (CMAC) baru. Rilis ini menambahkan arsitektur ARM di Amazon Linux 2. Penyedia JCE Javadocs sekarang tersedia untuk AWS CloudHSM

Pustaka PKCS #11

- Peningkatan stabilitas dan perbaikan bug.
- Sekarang didukung pada arsitektur ARM dengan Amazon Linux 2.
- Algoritma
 - CKM_AES_CMAC (tanda tangani dan verifikasi)

OpenSSL Dynamic Engine

- Peningkatan stabilitas dan perbaikan bug.
- Sekarang didukung pada arsitektur ARM dengan Amazon Linux 2.

Penyedia JCE

- Peningkatan stabilitas dan perbaikan bug.
- Algoritma
 - AESCMAC

Versi 5.6.0

Versi 5.6.0 mencakup dukungan mekanisme baru untuk perpustakaan PKCS #11 dan penyedia JCE. Selain itu, versi 5.6 mendukung Ubuntu 20.04.

Pustaka PKCS #11

- Peningkatan stabilitas dan perbaikan bug.
- Mekanisme
 - CKM_RSA_X_509, untuk mengenkripsi, mendekripsi, menandatangani, dan memverifikasi mode

OpenSSL Dynamic Engine

- Peningkatan stabilitas dan perbaikan bug.

Penyedia JCE

- Peningkatan stabilitas dan perbaikan bug.
- Cipher
 - RSA/ECB/NoPadding, untuk mengenkripsi dan mendekripsi mode

Kunci yang didukung

- EC dengan kurva secp224r1 dan secp521r1

Dukungan platform

- Menambahkan dukungan untuk Ubuntu 20.04.

Versi 5.5.0

Versi 5.5.0 menambahkan dukungan untuk OpenJDK 11, integrasi Keytool dan Jarsigner, dan mekanisme tambahan ke penyedia JCE. Menyelesaikan [masalah yang diketahui](#) mengenai KeyGenerator kelas yang salah menafsirkan parameter ukuran kunci sebagai jumlah byte, bukan bit.

Pustaka PKCS #11

- Peningkatan stabilitas dan perbaikan bug.

OpenSSL Dynamic Engine

- Peningkatan stabilitas dan perbaikan bug.

Penyedia JCE

- Support untuk utilitas Keytool dan Jarsigner
- Support for OpenJDK 11 di semua platform
- Cipher
 - AES/CBC/NoPaddingEnkripsi dan Dekripsi mode
 - AES/ECB/PKCS5Padding Enkripsi dan Dekripsi mode
 - AES/CTR/NoPaddingEnkripsi dan Dekripsi mode
 - AES/GCM/NoPaddingMode Bungkus dan Buka Bungkus
 - DESede/ECB/PKCS5Padding Enkripsi dan Dekripsi mode
 - DESede/CBC/NoPaddingEnkripsi dan Dekripsi mode
 - AESWrap/ECB/NoPaddingMode Bungkus dan Buka Bungkus
 - AESWrap/ECB/PKCS5Padding Mode Bungkus dan Buka Bungkus
 - AESWrap/ECB/ZeroPaddingMode Bungkus dan Buka Bungkus
 - RSA/ECB/PKCS1Padding Mode Bungkus dan Buka Bungkus
 - RSA/ECB/OAEPPaddingMode Bungkus dan Buka Bungkus
 - RSA/ECB/OAEPWithSHA-1 Mode ANDMGF1 Padding Wrap dan Unwrap
 - RSA/ECB/OAEPWithSHA-224 Mode ANDMGF1 Padding Wrap dan Unwrap
 - RSA/ECB/OAEPWithSHA-256 Mode ANDMGF1 Padding Wrap dan Unwrap
 - RSA/ECB/OAEPWithSHA-384 Mode ANDMGF1 Padding Wrap dan Unwrap
 - RSA/ECB/OAEPWithSHA-512 Mode ANDMGF1 Padding Wrap dan Unwrap
 - RSAAESWrap/ECB/OAEPPaddingMode Bungkus dan Buka Bungkus
 - RSAAESWrap/ECB/OAEPWithSHA-1 Mode ANDMGF1 Padding Wrap dan Unwrap
 - RSAAESWrap/ECB/OAEPWithSHA-224 Mode ANDMGF1 Padding Wrap dan Unwrap
 - RSAAESWrap/ECB/OAEPWithSHA-256 Mode ANDMGF1 Padding Wrap dan Unwrap
 - RSAAESWrap/ECB/OAEPWithSHA-384 Mode ANDMGF1 Padding Wrap dan Unwrap
 - RSAAESWrap/ECB/OAEPWithSHA-512 Mode ANDMGF1 Padding Wrap dan Unwrap

- RSA - Kunci RSA 2048-bit hingga 4096-bit, dengan penambahan 256 bit
- Kunci AES - 128, 192, dan 256-bit AES
- Pasangan kunci EC untuk kurva NIST secp256r1 (P-256), secp384r1 (P-384), dan secp256k1
- DESede (3DES)
- GenericSecret
- HMAC - dengan SHA1,, SHA224, SHA256 SHA384, dukungan SHA512 hash
- Tanda/Verifikasi
 - RSASSA-PSS
 - SHA1withRSA/PSS
 - SHA224withRSA/PSS
 - SHA256withRSA/PSS
 - SHA384withRSA/PSS
 - SHA512withRSA/PSS
 - SHA1withRSAandMGF1
 - SHA224withRSAandMGF1
 - SHA256withRSAandMGF1
 - SHA384withRSAandMGF1
 - SHA512withRSAandMGF1

Versi 5.4.2

Versi 5.4.2 mencakup peningkatan stabilitas dan perbaikan bug untuk semua. SDKs Ini juga merupakan rilis terakhir untuk platform CentOS 8. Untuk informasi lebih lanjut, lihat situs web [CentOS](#).

Pustaka PKCS #11

- Peningkatan stabilitas dan perbaikan bug.

OpenSSL Dynamic Engine

- Peningkatan stabilitas dan perbaikan bug.

Penyedia JCE

- Peningkatan stabilitas dan perbaikan bug.

Versi 5.4.1

Versi 5.4.1 menyelesaikan [masalah yang diketahui](#) dengan pustaka PKCS #11. Ini juga merupakan rilis terakhir untuk platform CentOS 8. Untuk informasi lebih lanjut, lihat situs web [CentOS](#).

Pustaka PKCS #11

- Peningkatan stabilitas dan perbaikan bug.

OpenSSL Dynamic Engine

- Peningkatan stabilitas dan perbaikan bug.

Penyedia JCE

- Peningkatan stabilitas dan perbaikan bug.

Versi 5.4.0

Versi 5.4.0 menambahkan dukungan awal untuk penyedia JCE untuk semua platform. Penyedia JCE kompatibel dengan OpenJDK 8.

Pustaka PKCS #11

- Peningkatan stabilitas dan perbaikan bug.

OpenSSL Dynamic Engine

- Peningkatan stabilitas dan perbaikan bug.

Penyedia JCE

- Tipe kunci
 - RSA — 2048-bit sampai 4096-bit kunci RSA, dengan penambahan 256 bit.

- AES — 128, 192, dan 256-bit kunci AES.
- Pasangan kunci ECC untuk kurva NIST secp256r1 (P-256), secp384r1 (P-384), dan secp256k1.
- DESede (3DES)
- HMAC — dengan SHA1,, SHA224, SHA256 SHA384, dukungan SHA512 hash.
- Cipher (hanya mengenkripsi dan mendekripsi)
 - AES/GCM/NoPadding
 - AES/ECB/NoPadding
 - AES/CBC/PKCS5Padding
 - DESede/ECB/NoPadding
 - DESede/CBC/PKCS5Padding
 - AES/CTR/NoPadding
 - RSA/ECB/PKCS1Padding
 - RSA/ECB/OAEPPadding
 - RSA/ECB/OAEPWithSHA-1 ANDMGF1 Bantalan
 - RSA/ECB/OAEPWithSHA-224 Bantalan ANDMGF1
 - RSA/ECB/OAEPWithSHA-256 Bantalan ANDMGF1
 - RSA/ECB/OAEPWithSHA-384 Bantalan ANDMGF1
 - RSA/ECB/OAEPWithSHA-512 Bantalan ANDMGF1
- Intisari
 - SHA-1
 - SHA-224
 - SHA-256
 - SHA-384
 - SHA-512
- Tanda/Verifikasi
 - NONEwithRSA
 - SHA1withRSA
 - SHA224withRSA
 - SHA256withRSA
 - SHA384withRSA

- SHA512withRSA
- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA
- Integrasi dengan Java KeyStore

Versi 5.3.0

Perpustakaan PKCS #11

- Peningkatan stabilitas dan perbaikan bug.

Mesin Dinamis OpenSSL

- Tambahkan dukungan untuk ECDSA sign/verify dengan kurva P-256, P-384, dan secp256k1.
- Tambahkan dukungan untuk platform: Amazon Linux, Amazon Linux 2, CentOS 7.8+, RHEL 7 (7.8+).
- Tambahkan dukungan untuk OpenSSL versi 1.0.2.
- Peningkatan stabilitas dan perbaikan bug.

Penyedia JCE

- Tipe kunci
 - RSA — 2048-bit sampai 4096-bit kunci RSA, dengan penambahan 256 bit.
 - AES — 128, 192, dan 256-bit kunci AES.
 - Pasangan kunci EC untuk kurva NIST secp256r1 (P-256), secp384r1 (P-384), dan secp256k1.
 - DESede (3DES)
 - HMAC — dengan SHA1,, SHA224, SHA256 SHA384, dukungan SHA512 hash.
- Cipher (hanya mengenkripsi dan mendekripsi)
 - AES/GCM/NoPadding

- AES/ECB/NoPadding
- AES/CBC/PKCS5Padding
- DESede/ECB/NoPadding
- DESede/CBC/PKCS5Padding
- AES/CTR/NoPadding
- RSA/ECB/PKCS1Padding
- RSA/ECB/OAEPPadding
- RSA/ECB/OAEPWithSHA-1 ANDMGF1 Bantalan
- RSA/ECB/OAEPWithSHA-224 Bantalan ANDMGF1
- RSA/ECB/OAEPWithSHA-256 Bantalan ANDMGF1
- RSA/ECB/OAEPWithSHA-384 Bantalan ANDMGF1
- RSA/ECB/OAEPWithSHA-512 Bantalan ANDMGF1
- Intisari
 - SHA-1
 - SHA-224
 - SHA-256
 - SHA-384
 - SHA-512
- Tanda/Verifikasi
 - NONEwithRSA
 - SHA1withRSA
 - SHA224withRSA
 - SHA256withRSA
 - SHA384withRSA
 - SHA512withRSA
 - NONEwithECDSA
 - SHA1withECDSA
 - SHA224withECDSA
 - SHA256withECDSA
 - SHA384withECDSA

- SHA512withECDSA
- Integrasi dengan Java KeyStore

Versi 5.2.1

Perpustakaan PKCS #11

- Peningkatan stabilitas dan perbaikan bug.

Mesin Dinamis OpenSSL

- Peningkatan stabilitas dan perbaikan bug.

Versi 5.2.0

Versi 5.2.0 menambahkan dukungan tipe dan mekanisme kunci tambahan ke pustaka PKCS #11.

Perpustakaan PKCS #11

Tipe Kunci

- ECDSA— Kurva P-224, P-256, P-384, P-521 dan secp256k1
- Triple DES (3DES)

Mekanisme

- CKM_EC_KEY_PAIR_GEN
- CKM_DES3_KEY_GEN
- CKM_CBC_DES3
- CKM_CBC_PAD_DES3
- CKM_ECB_DES3
- CKM_ECDSA
- CKM_ECDSA_SHA1
- CKM_ECDSA_SHA224
- CKM_ECDSA_SHA256

- CKM_ECDSA_SHA384
- CKM_ECDSA_SHA512
- CKM_RSA_PKCS untuk Enkripsi/Dekripsi

Mesin Dinamis OpenSSL

- Peningkatan stabilitas dan perbaikan bug.

Versi 5.1.0

Versi 5.1.0 menambahkan dukungan untuk mekanisme tambahan ke pustaka PKCS #11.

Perpustakaan PKCS #11

Mekanisme

- CKM_RSA_PKCS untuk Membungkus/Membuka
- CKM_RSA_PKCS_PSS
- CKM_SHA1_RSA_PKCS_PSS
- CKM_SHA224_RSA_PKCS_PSS
- CKM_SHA256_RSA_PKCS_PSS
- CKM_SHA384_RSA_PKCS_PSS
- CKM_SHA512_RSA_PKCS_PSS
- CKM_AES_ECB
- CKM_AES_CTR
- CKM_AES_CBC
- CKM_AES_CBC_PAD
- CKM_108_COUNTER_KDF SP800
- CKM_GENERIC_SECRET_KEY_GEN
- CKM_SHA_1_HMAC
- CKM_1_HMAC SHA224
- CKM_1_HMAC SHA256

- CKM_ _HMAC SHA384
- CKM_ _HMAC SHA512
- CKM_RSA_PKCS_OAEP saja Wrap/Unwrap
- CKM_RSA_AES_KEY_WRAP
- CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD
- CKM_CLOUDHSM_AES_KEY_WRAP_ _PAD PKCS5
- CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD

Operasi API

- C_ CreateObject
- C_ DeriveKey
- C_ WrapKey
- C_ UnWrapKey

Mesin Dinamis OpenSSL

- Peningkatan stabilitas dan perbaikan bug.

Versi 5.0.1

Versi 5.0.1 menambahkan dukungan awal untuk OpenSSL Dynamic Engine.

Perpustakaan PKCS #11

- Peningkatan stabilitas dan perbaikan bug.

Mesin Dinamis OpenSSL

- Rilis awal dari OpenSSL Dynamic Engine.
- Rilis ini menawarkan dukungan pengantar untuk tipe kunci dan OpenSSL APIs:
 - Pembuatan kunci RSA untuk tombol 2048, 3072, dan 4096 bit
 - OpenSSL APIs:
 - [Tanda RSA menggunakan RSA](#) PKCS dengan/224/256/384/512 & RSA PSS SHA1

- [Generasi Kunci RSA](#)

Untuk informasi lebih lanjut, lihat [OpenSSL Dynamic Engine](#).

- Platform yang didukung: CentOS 8.3+, Red Hat Enterprise Linux (RHEL) 8.3+, dan Ubuntu 18.04 LTS
- Membutuhkan: OpenSSL 1.1.1

Untuk informasi lebih lanjut, lihat [Platform yang Didukung](#).

- Support untuk SSL/TLS Offload pada CentOS 8.3+, Red Hat Enterprise Linux (RHEL) 8.3, dan Ubuntu 18.04 LTS, termasuk NGINX 1.19 (untuk suite cipher tertentu).

Untuk informasi selengkapnya, lihat [SSL/TLS Offload di Linux menggunakan Tomcat atau SSL/TLS Offload di Linux menggunakan](#) NGINX atau Apache.

Versi 5.0.0

Versi 5.0.0 adalah rilis pertama.

Perpustakaan PKCS #11

- Ini adalah rilis awal.

Dukungan pustaka PKCS #11 pengantar di SDK klien versi 5.0.0

Bagian ini memperinci dukungan untuk kunci jenis, mekanisme, operasi API dan atribut Klien SDK versi 5.0.0.

Tipe Kunci:

- AES— 128, 192, dan 256-bit kunci AES
- RSA— 2048-bit ke 4096-bit kunci RSA, dengan penambahan 256 bit

Mekanisme:

- CKM_AES_GCM
- CKM_AES_KEY_GEN
- CKM_CLOUDHSM_AES_GCM

- CKM_RSA_PKCS
- CKM_RSA_X9_31_KEY_PAIR_GEN
- CKM_SHA1
- CKM__RSA_PKCS_SHA1
- CKM_SHA224
- CKM__RSA_PKCS_SHA224
- CKM_SHA256
- CKM__RSA_PKCS_SHA256
- CKM_SHA384
- CKM__RSA_PKCS_SHA384
- CKM_SHA512
- CKM__RSA_PKCS_SHA512

Operasi API:

- C_CloseAllSessions
- C_CloseSession
- C_Dekripsi
- C_DecryptFinal
- C_DecryptInit
- C_DecryptUpdate
- C_DestroyObject
- C_Digest
- C_DigestFinal
- C_DigestInit
- C_DigestUpdate
- C_Enkripsi
- C_EncryptFinal
- C_EncryptInit
- C_EncryptUpdate

- C_Finalisasi
- C_FindObjects
- C_FindObjectsFinal
- C_FindObjectsInit
- C_GenerateKey
- C_GenerateKeyPair
- C_GenerateRandom
- C_GetAttributeValue
- C_GetFunctionList
- C_GetInfo
- C_GetMechanismInfo
- C_GetMechanismList
- C_GetSessionInfo
- C_GetSlotInfo
- C_GetSlotList
- C_GetTokenInfo
- C_Inisialisasi
- C_Login
- C_Keluar
- C_OpenSession
- Tanda C_
- C_SignFinal
- C_SignInit
- C_SignUpdate
- C_Verifikasi
- C_VerifyFinal
- C_VerifyInit
- C_VerifyUpdate

Atribut:

- `GenerateKeyPair`
 - Semua atribut Kunci RSA
- `GenerateKey`
 - Semua atribut AES Key
- `GetAttributeValue`
 - Semua atribut Kunci RSA
 - Semua atribut AES Key

Sampel:

- [Hasilkan kunci \(AES, RSA, EC\)](#)
- [Daftar atribut kunci](#)
- [Enkripsi dan dekripsi data dengan AES GCM](#)
- [Tanda tangani dan verifikasi data dengan RSA](#)

Rilis Client SDK 3 yang tidak digunakan lagi

Bagian ini mencantumkan rilis Client SDK 3 yang tidak digunakan lagi.

Versi 3.4.4

Versi 3.4.4 menambahkan pembaruan ke penyedia JCE.

AWS CloudHSM Perangkat Lunak Klien

- Memperbarui versi untuk konsistensi.

Perpustakaan PKCS #11

- Memperbarui versi untuk konsistensi.

Mesin Dinamis OpenSSL

- Memperbarui versi untuk konsistensi.

Penyedia JCE

- Perbarui log4j ke versi 2.17.1.

Windows (penyedia CNG dan KSP)

- Memperbarui versi untuk konsistensi.

Versi 3.4.3

Versi 3.4.3 menambahkan pembaruan ke penyedia JCE.

AWS CloudHSM Perangkat Lunak Klien

- Memperbarui versi untuk konsistensi.

Perpustakaan PKCS #11

- Memperbarui versi untuk konsistensi.

Mesin Dinamis OpenSSL

- Memperbarui versi untuk konsistensi.

Penyedia JCE

- Perbarui log4j ke versi 2.17.0.

Windows (penyedia CNG dan KSP)

- Memperbarui versi untuk konsistensi.

Versi 3.4.2

Versi 3.4.2 menambahkan pembaruan ke penyedia JCE.

AWS CloudHSM Perangkat Lunak Klien

- Memperbarui versi untuk konsistensi.

Perpustakaan PKCS #11

- Memperbarui versi untuk konsistensi.

Mesin Dinamis OpenSSL

- Memperbarui versi untuk konsistensi.

Penyedia JCE

- Perbarui log4j ke versi 2.16.0.

Windows (penyedia CNG dan KSP)

- Memperbarui versi untuk konsistensi.

Versi 3.4.1

Versi 3.4.1 menambahkan pembaruan ke penyedia JCE.

AWS CloudHSM Perangkat Lunak Klien

- Memperbarui versi untuk konsistensi.

Perpustakaan PKCS #11

- Memperbarui versi untuk konsistensi.

Mesin Dinamis OpenSSL

- Memperbarui versi untuk konsistensi.

Penyedia JCE

- Perbarui log4j ke versi 2.15.0.

Windows (penyedia CNG dan KSP)

- Memperbarui versi untuk konsistensi.

Versi 3.4.0

Versi 3.4.0 menambahkan pembaruan ke semua komponen.

AWS CloudHSM Perangkat Lunak Klien

- Peningkatan stabilitas dan perbaikan bug.

Perpustakaan PKCS #11

- Peningkatan stabilitas dan perbaikan bug.

Mesin Dinamis OpenSSL

- Peningkatan stabilitas dan perbaikan bug.

Penyedia JCE

- Peningkatan stabilitas dan perbaikan bug.

Windows (penyedia CNG dan KSP)

- Peningkatan stabilitas dan perbaikan bug.

Versi 3.3.2

Versi 3.3.2 menyelesaikan [masalah](#) dengan skrip client_info.

AWS CloudHSM Perangkat Lunak Klien

- Memperbarui versi untuk konsistensi.

Perpustakaan PKCS #11

- Memperbarui versi untuk konsistensi.

Mesin Dinamis OpenSSL

- Memperbarui versi untuk konsistensi.

Penyedia JCE

- Memperbarui versi untuk konsistensi.

Windows (penyedia CNG dan KSP)

- Memperbarui versi untuk konsistensi.

Versi 3.3.1

Versi 3.3.1 menambahkan pembaruan ke semua komponen.

AWS CloudHSM Perangkat Lunak Klien

- Peningkatan stabilitas dan perbaikan bug.

Perpustakaan PKCS #11

- Peningkatan stabilitas dan perbaikan bug.

Mesin Dinamis OpenSSL

- Peningkatan stabilitas dan perbaikan bug.

Penyedia JCE

- Peningkatan stabilitas dan perbaikan bug.

Windows (penyedia CNG dan KSP)

- Peningkatan stabilitas dan perbaikan bug.

Versi 3.3.0

Versi 3.3.0 menambahkan autentikasi dua faktor (2FA) dan perbaikan lainnya.

AWS CloudHSM Perangkat Lunak Klien

- Ditambahkan autentikasi 2FA untuk petugas krypto (CO). Untuk informasi lebih lanjut, lihat [Mengelola Autentikasi Dua Faktor untuk Petugas Crypto](#).
- Dukungan platform yang dihapus untuk RedHat Enterprise Linux 6 dan CentOS 6. Untuk informasi lebih lanjut, lihat [Dukungan Linux](#).
- Menambahkan versi mandiri CMU untuk digunakan dengan SDK Klien 5 atau SDK Klien 3. Ini adalah versi CMU yang sama yang disertakan dengan daemon klien versi 3.3.0, dan sekarang Anda dapat mengunduh CMU tanpa mengunduh daemon klien.

Perpustakaan PKCS #11

- Peningkatan stabilitas dan perbaikan bug.
- Dukungan platform yang dihapus untuk RedHat Enterprise Linux 6 dan CentOS 6. Untuk informasi lebih lanjut, lihat [Dukungan Linux](#).

Mesin Dinamis OpenSSL

- Memperbarui versi untuk konsistensi
- Dukungan platform yang dihapus untuk RedHat Enterprise Linux 6 dan CentOS 6. Untuk informasi lebih lanjut, lihat [Dukungan Linux](#).

Penyedia JCE

- Peningkatan stabilitas dan perbaikan bug.
- Dukungan platform yang dihapus untuk RedHat Enterprise Linux 6 dan CentOS 6. Untuk informasi lebih lanjut, lihat [Dukungan Linux](#).

Windows (penyedia CNG dan KSP)

- Memperbarui versi untuk konsistensi

Versi 3.2.1

Versi 3.2.1 menambahkan analisis kepatuhan antara AWS CloudHSM implementasi pustaka PKCS #11 dan standar PKCS #11, platform baru, dan perbaikan lainnya.

AWS CloudHSM Perangkat Lunak Klien

- Tambahkan dukungan platform untuk CentOS 8, RHEL 8, dan Ubuntu 18.04 LTS. Untuk informasi selengkapnya, lihat [???](#).

Perpustakaan PKCS #11

- [Laporan kepatuhan pustaka PKCS #11 untuk SDK klien 3.2.1](#)
- Tambahkan dukungan platform untuk CentOS 8, RHEL 8, dan Ubuntu 18.04 LTS. Untuk informasi selengkapnya, lihat [???](#).

Mesin Dinamis OpenSSL

- Tidak ada dukungan untuk CentOS 8, RHEL 8, dan Ubuntu 18.04 LTS. Untuk informasi lebih lanjut, lihat [???](#).

Penyedia JCE

- Tambahkan dukungan platform untuk CentOS 8, RHEL 8, dan Ubuntu 18.04 LTS. Untuk informasi selengkapnya, lihat [???](#).

Windows (penyedia CNG dan KSP)

- Peningkatan stabilitas dan perbaikan bug.

Versi 3.2.0

Versi 3.2.0 menambahkan dukungan untuk menutupi kata sandi dan perbaikan lainnya.

AWS CloudHSM Perangkat Lunak Klien

- Menambahkan dukungan untuk menyembunyikan kata sandi Anda saat menggunakan alat baris perintah. Untuk informasi lebih lanjut, lihat [loginHSM dan logoutHSM](#)(cloudhsm_mgmt_util) dan [loginHSM dan logoutHSM](#) (key_mgmt_util).

Perpustakaan PKCS #11

- Menambahkan dukungan untuk hashing data besar dalam perangkat lunak untuk beberapa mekanisme PKCS #11 yang sebelumnya tidak didukung. Untuk informasi lebih lanjut, lihat [Mekanisme yang Didukung](#).

Mesin Dinamis OpenSSL

- Peningkatan stabilitas dan perbaikan bug.

Penyedia JCE

- Memperbarui versi untuk konsistensi.

Windows (penyedia CNG dan KSP)

- Peningkatan stabilitas dan perbaikan bug.

Versi 3.1.2

Versi 3.1.2 menambahkan pembaruan ke penyedia JCE.

AWS CloudHSM Perangkat Lunak Klien

- Memperbarui versi untuk konsistensi

Perpustakaan PKCS #11

- Memperbarui versi untuk konsistensi

Mesin Dinamis OpenSSL

- Memperbarui versi untuk konsistensi

Penyedia JCE

- Perbarui log4j ke versi 2.13.3

Windows (penyedia CNG dan KSP)

- Memperbarui versi untuk konsistensi

Versi 3.1.1

AWS CloudHSM Perangkat Lunak Klien

- Memperbarui versi untuk konsistensi.

Perpustakaan PKCS #11

- Memperbarui versi untuk konsistensi.

Mesin Dinamis OpenSSL

- Memperbarui versi untuk konsistensi.

Penyedia JCE

- Perbaiki bug dan peningkatan performa.

Jendela (CNG, KSP)

- Memperbarui versi untuk konsistensi.

Versi 3.1.0

Versi 3.1.0 menambahkan [pembungkus kunci AES yang sesuai standar](#).

AWS CloudHSM Perangkat Lunak Klien

- Persyaratan baru untuk peningkatan: versi klien Anda harus sesuai dengan versi pustaka perangkat lunak yang Anda gunakan. Untuk meningkatkan, Anda harus menggunakan perintah

batch yang meningkatkan klien dan semua pustaka pada waktu yang sama. Untuk informasi selengkapnya, lihat [Peningkatan SDK 3 Klien](#).

- Key_mgmt_util (KMU) mencakup pembaruan berikut:
 - Menambahkan dua metode pembungkus kunci AES baru - bungkus kunci AES yang sesuai standar dengan bantalan nol dan bungkus kunci AES tanpa bantalan. Untuk informasi lebih lanjut, lihat [wrapKey](#) dan [unwrapKey](#).
 - Kemampuan dinonaktifkan untuk menentukan IV khusus saat membungkus kunci menggunakan PKCS5 AES_KEY_WRAP_PAD_. Untuk informasi lebih lanjut, lihat [Pembungkus Kunci AES](#).

Perpustakaan PKCS #11

- Menambahkan dua metode pembungkus kunci AES baru - bungkus kunci AES yang sesuai standar dengan bantalan nol dan bungkus kunci AES tanpa bantalan. Untuk informasi lebih lanjut, lihat [Pembungkus Kunci AES](#).
- Anda dapat mengatur konfigurasi panjang salt untuk tanda tangan RSA-PSS. Untuk mempelajari cara menggunakan fitur ini, lihat [Panjang garam yang dapat dikonfigurasi untuk tanda tangan RSA-PSS aktif](#). GitHub

Mesin Dinamis OpenSSL

- BREAKING CHANGE: TLS 1.0 dan 1.2 cipher suite dengan tidak SHA1 tersedia di OpenSSL Engine 3.1.0. Masalah ini akan segera diatasi.
- Jika Anda berniat untuk menginstal pustaka OpenSSL Dynamic Engine pada RHEL 6 atau CentOS 6, lihat [masalah yang diketahui](#) tentang versi OpenSSL default yang diinstal pada sistem operasi tersebut.
- Peningkatan stabilitas dan perbaikan bug

Penyedia JCE

- BREAKING CHANGE: Untuk mengatasi masalah dengan kepatuhan Java Cryptography Extension (JCE), AES wrap and unwrap sekarang menggunakan algoritma dengan benar alih-alih AESWrap algoritma AES. Ini berarti Cipher.WRAP_MODE dan Cipher.UNWRAP_MODE tidak lagi berhasil untuk AES/ECB dan mekanisme AES/CBC.

Untuk meningkatkan ke klien versi 3.1.0, Anda harus memperbarui kode Anda. Jika Anda masih memiliki kunci terbungkus, Anda harus memberi perhatian khusus untuk mekanisme yang Anda

gunakan untuk membukanya dan bagaimana default IV telah berubah. Jika Anda membungkus kunci dengan versi klien 3.0.0 atau sebelumnya, maka di 3.1.1 Anda harus menggunakan AESWrap/ECB/PKCS5Padding untuk membuka kunci yang ada. Untuk informasi lebih lanjut, lihat [Pembungkus Kunci AES](#).

- Anda dapat membuat daftar beberapa kunci dengan label yang sama dari penyedia JCE. Untuk mempelajari cara mengulang semua kunci yang tersedia, lihat [Temukan semua kunci aktif](#). GitHub
- Anda dapat mengatur nilai yang lebih ketat untuk atribut selama pembuatan kunci, termasuk menentukan label yang berbeda untuk kunci publik dan privat. Untuk informasi lebih lanjut, lihat [Atribut Java yang Didukung](#).

Jendela (CNG, KSP)

- Peningkatan stabilitas dan perbaikan bug.

AWS CloudHSM end-of-life Rilis SDK klien

Versi AWS CloudHSM klien berikut telah mencapai akhir dukungan. Versi AWS CloudHSM klien ini tidak lagi kompatibel dengan layanan dan tidak akan menerima pembaruan. Untuk menjaga keamanan aplikasi Anda, AWS CloudHSM mungkin menolak koneksi dari rilis yang telah mencapai akhir dukungan.

- SDK versi 3.4.4 dan sebelumnya telah mencapai akhir dukungan.
- SDK versi 5.8.0 dan sebelumnya telah mencapai akhir dukungan.

Riwayat dokumen

Topik ini menjelaskan pembaruan yang signifikan untuk Panduan Pengguna AWS CloudHSM .

Topik

- [Pembaruan terkini](#)
- [Pembaruan lebih awal](#)

Pembaruan terkini

Tabel berikut menjelaskan perubahan signifikan pada dokumentasi ini sejak April 2018. Selain perubahan besar yang tercantum di sini, kami juga sering memperbarui dokumentasi untuk meningkatkan deskripsi dan contoh, dan untuk mengatasi umpan balik yang Anda kirimkan kepada kami. Agar menerima pemberitahuan tentang perubahan signifikan, gunakan tautan di sudut kanan atas untuk berlangganan umpan RSS.

Untuk detail tentang rilis baru, lihat [Unduh untuk AWS CloudHSM Client SDK](#)

Perubahan	Deskripsi	Tanggal
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.17.1.	Maret 19, 2026
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.17.0.	Januari 6, 2026
Menambahkan tautan unduhan Penyedia OpenSSL	Menambahkan tautan unduhan Penyedia OpenSSL AWS CloudHSM ke klien versi 5.17.	Januari 6, 2026
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.16.2.	September 16, 2025
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.16.1.	Juni 25, 2025

Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.16.0.	1 Mei 2025
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.15.0.	Februari 3, 2025
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.14.0.	November 26, 2024
Tipe HSM baru dan mode cluster	Menambahkan dukungan untuk membuat (hsm2m.medium) di cluster mode FIPS.	Agustus 20, 2024
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.13.0.	Agustus 13, 2024
Tipe HSM baru dan mode cluster	Meluncurkan tipe HSM baru (hsm2m.medium) dan mode cluster baru (non-FIPS).	Juni 10, 2024
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.12.0.	Maret 20, 2024
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.11.0.	Januari 17, 2024
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.10.0.	28 Juli 2023
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.9.0.	23 Mei 2023
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.8.0.	16 Maret 2023
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.7.0.	16 November 2022
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.6.0.	September 1, 2022

Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.5.0.	Mei 13, 2022
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.4.2.	18 Maret 2022
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.4.1.	Februari 10, 2022
Menambahkan rilis baru	Merilis penyedia AWS CloudHSM JCE versi 5.4.0 untuk platform Windows.	1 Februari 2022
Menambahkan rilis baru	Merilis versi AWS CloudHSM klien 5.4.0, yang menambahkan dukungan awal untuk penyedia JCE untuk semua platform Linux.	28 Januari 2022
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.3.0.	Januari 3, 2022
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 3.4.4.	Januari 3, 2022
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 3.4.3.	Desember 20, 2021
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 3.4.2.	Desember 15, 2021
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 3.4.1.	Desember 10, 2021
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.2.1.	4 Oktober 2021
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 3.4.0.	25 Agustus 2021

Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.2.0.	Agustus 3, 2021
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 3.3.2.	2 Juli 2021
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.1.0.	1 Juni 2021
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 3.3.1.	26 April 2021
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.0.1.	8 April 2021
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 5.0.0.	12 Maret 2021
Menambahkan konten baru	Menambahkan antarmuka VPC endpoint, fitur AWS yang memungkinkan Anda membuat koneksi pribadi antara VPC Anda dan AWS CloudHSM tanpa memerlukan akses melalui internet atau melalui perangkat NAT, koneksi VPN, atau koneksi. Direct Connect	10 Februari 2021
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 3.3.0.	3 Februari 2021
Tambahkan konten baru	Penambahan retensi cadangan terkelola, fitur yang secara otomatis menghapus cadangan lama.	18 November 2020

Tambahkan konten baru	Menambahkan laporan kepatuhan yang menganalisis implementasi AWS CloudHSM Client SDK 3.2.1 dari pustaka PKCS #11 dengan standar PKCS #11.	29 Oktober 2020
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 3.2.1.	8 Oktober 2020
Menambahkan konten baru	Penambahan dokumentasi yang menjelaskan pengaturan sinkronisasi kunci di AWS CloudHSM.	1 September 2020
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 3.2.0.	31 Agustus 2020
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 3.1.2.	30 Juli 2020
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 3.1.1.	3 Juni 2020
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 3.1.0.	21 Mei 2020
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 3.0.1.	20 April 2020
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 3.0.0 untuk platform Windows Server.	30 Oktober 2019
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 3.0.0 untuk semua platform, kecuali Windows.	22 Oktober 2019

Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 2.0.4.	26 Agustus 2019
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 2.0.3.	13 Mei 2019
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 2.0.1.	21 Maret 2019
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 2.0.0.	6 Februari 2019
Ditambahkan dukungan wilayah	Menambahkan AWS CloudHSM dukungan untuk wilayah UE (Stockholm) dan AWS GovCloud (AS-Timur).	19 Desember 2018
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 1.1.2 untuk Windows.	20 November 2018
Masalah yang diketahui diperbarui	Konten baru ditambahkan ke panduan Pemecahan Masalah.	8 November 2018
Menambahkan rilis baru	Dirilis versi AWS CloudHSM klien 1.1.2 untuk platform Linux.	8 November 2018
Ditambahkan dukungan wilayah	Menambahkan AWS CloudHSM dukungan untuk wilayah UE (Paris) dan Asia Pasifik (Seoul).	24 Oktober 2018
Menambahkan konten baru	Menambahkan kemampuan untuk menghapus dan memulihkan AWS CloudHSM cadangan.	10 September 2018

Menambahkan konten baru	Menambahkan pengiriman log audit otomatis ke Amazon CloudWatch Logs.	13 Agustus 2018
Menambahkan konten baru	Menambahkan kemampuan untuk menyalin cadangan AWS CloudHSM cluster di seluruh wilayah.	30 Juli 2018
Ditambahkan dukungan wilayah	Menambahkan AWS CloudHSM dukungan untuk wilayah UE (London).	13 Juni 2018
Menambahkan konten baru	Menambahkan dukungan AWS CloudHSM klien dan perpustakaan untuk Amazon Linux 2, Red Hat Enterprise Linux (RHEL) 6, Red Hat Enterprise Linux (RHEL) 7, CentOS 6, CentOS 7, dan Ubuntu 16.04 LTS.	10 Mei 2018
Menambahkan rilis baru	Menambahkan AWS CloudHSM klien Windows.	30 April 2018

Pembaruan lebih awal

Tabel berikut menjelaskan perubahan penting AWS CloudHSM sebelum 2018.

Ubah	Deskripsi	Tanggal
Konten baru	Menambahkan otentikasi kuorum (M of N access control) untuk petugas crypto (). COs Untuk informasi selengkapnya, lihat Menggunakan CloudHSM	9 November 2017

Ubah	Deskripsi	Tanggal
	Management Utility (CMU) untuk mengelola otentikasi kuorum (M of N access control).	
Perbarui	Penambahan dokumentasi tentang menggunakan alat baris perintah key_mgmt_util. Untuk informasi lebih lanjut, lihat Referensi untuk perintah AWS CloudHSM Key Management Utility .	9 November 2017
Konten baru	Tambahan Enkripsi Data Transparan Oracle. Untuk informasi lebih lanjut, lihat Enkripsi basis data Oracle .	25 Oktober 2017
Konten baru	Penambahan SSL Offload. Untuk informasi lebih lanjut, lihat SSL/TLS pembongkaran .	12 Oktober 2017
Panduan baru	Rilis ini memperkenalkan AWS CloudHSM	14 Agustus, 2017

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.