Panduan Pengguna

AWS CloudHSM

AWS CloudHSM: Panduan Pengguna

Table of Contents

Apakah AWS CloudHSM itu?

AWS CloudHSMmenggabungkan manfaat AWS cloud dengan keamanan modul keamanan perangkat keras (HSM). Modul keamanan perangkat keras (HSM) adalah perangkat komputasi yang memproses operasi kriptografi dan menyediakan penyimpanan aman untuk kunci kriptografi. DenganAWS CloudHSM, Anda memiliki kendali penuh atas HSM ketersediaan tinggi 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 CloudHSMmenawarkan pelanggan berbagai manfaat:

HSM divalidasi FIPS 140-2 level-3

AWS CloudHSMmenggunakan HSM tujuan umum yang memenuhi standar, penyewa tunggal, dan FIPS 140-2 level-3 divalidasi. Mereka 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.

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 Anda

AWS CloudHSMmemberi Anda kontrol penuh atas algoritma dan kunci yang Anda gunakan. Anda dapat membuat, menyimpan, mengimpor, mengekspor, mengelola, dan menggunakan kunci kriptografi (termasuk, kunci sesi, kunci token, kunci simetris, dan pasangan kunci asimetris). Selain itu, AWS CloudHSM SDK memberi Anda kontrol penuh atas pengembangan aplikasi, bahasa aplikasi, threading, dan tempat 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

Akses ke klaster FIPS dan non-FIPS

Untuk mempelajari lebih lanjut tentang apa yang dapat Anda lakukan dengan AWS CloudHSM, lihat topik berikut. Ketika Anda siap untuk memulai dengan AWS CloudHSM, lihat Mulai.

> **Note**
>
> Jika Anda menginginkan layanan terkelola untuk membuat dan mengendalikan kunci enkripsi Anda tetapi Anda tidak ingin atau perlu mengoperasikan HSM Anda sendiri, pertimbangkan untuk menggunakannya AWS Key Management Service.
> Jika Anda mencari layanan elastis yang mengelola HSM pembayaran dan kunci untuk aplikasi pemrosesan pembayaran di cloud, pertimbangkan untuk menggunakan AWS Payment Cryptography.

Konten

Kasus penggunaan AWS CloudHSM

AWS CloudHSMdapat digunakan untuk mencapai berbagai tujuan. Konten dalam topik ini memberikan gambaran umum tentang apa yang dapat Anda lakukanAWS 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. HSM yang 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 Kepatuhan.

Enkripsi dan dekripsi 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 SDK kriptografi.

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 HMAC dan CMAC

Dalam kriptografi, Kode Otentikasi Pesan Cipher (CMAC) dan Kode Otentikasi Pesan Berbasis Hash (HMAC) digunakan untuk mengautentikasi dan memastikan integritas pesan yang dikirim melalui jaringan yang tidak aman. DenganAWS CloudHSM, Anda dapat dengan aman membuat dan mengelola kunci simetris yang mendukung HMAC dan CMAC.

Memanfaatkan manfaat AWS CloudHSM dan AWS Key Management Service

Pelanggan dapat menggabungkan AWS CloudHSM dan AWS KMSmenyimpan materi utama dalam lingkungan penyewa tunggal yang bersertifikat FIPS 140-2 Level 3 sambil juga mendapatkan manfaat manajemen utama, penskalaan, dan integrasi cloud. AWS KMS Untuk detail tentang cara melakukannya, lihat toko AWS CloudHSM utama di Panduan AWS Key Management Service Pengembang.

Offload pemrosesan SSL/TLS untuk server web

Untuk mengirim data dengan aman melalui internet, server web menggunakan pasangan kunci publik-pribadi dan sertifikat kunci publik SSL/TLS 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 menyiapkan pembongkaran SSL/TLS 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 HSM di Anda. AWS CloudHSM Untuk informasi tentang menyiapkan 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 menggunakan HSM Anda untuk melakukan operasi penandatanganan kriptografi.

Hasilkan angka acak

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

Bagaimana AWS CloudHSM berhasil

Topik ini memberikan gambaran umum tentang konsep dasar dan arsitektur yang Anda gunakan untuk mengenkripsi data dengan aman dan melakukan operasi kriptografi di HSM. AWS CloudHSMberoperasi di Amazon Virtual Private Cloud (VPC) Anda sendiri. Sebelum Anda dapat menggunakanAWS CloudHSM, Anda terlebih dahulu membuat cluster, menambahkan HSM ke dalamnya, membuat pengguna dan kunci, dan kemudian menggunakan SDK Klien untuk mengintegrasikan HSM Anda dengan aplikasi Anda. Setelah ini selesai, Anda menggunakan log SDK KlienAWS 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

- Klaster AWS CloudHSM
- Pengguna HSM
- Kunci HSM
- SDK Klien
- AWS CloudHSMcadangan kluster
- Wilayah

Klaster AWS CloudHSM

Membuat masing-masing HSM bekerja sama dalam cluster yang disinkronkan, berlebihan, dan sangat tersedia bisa jadi sulit, tetapi AWS CloudHSM melakukan pekerjaan berat untuk Anda dengan menyediakan modul keamanan perangkat keras (HSM) dalam kelompok. Sebuah klaster adalah kumpulan HSM individu yang dijaga AWS CloudHSM tetap sinkron. Ketika Anda melakukan tugas atau operasi pada satu HSM dalam sebuah klaster, HSM lain di klaster secara otomatis dimutakhirkan. Untuk memenuhi tujuan ketersediaan, daya tahan, dan skalabilitas, Anda menetapkan jumlah HSM di klaster Anda di beberapa zona ketersediaan.

Anda dapat membuat cluster yang memiliki 1 hingga 28 HSM (batas default adalah 6 HSM per AWS akun per AWSWilayah). Anda dapat menempatkan HSM di Availability Zone yang berbeda di suatu AWS wilayah. Menambahkan lebih banyak HSM ke klaster memberikan performa yang lebih tinggi. Menyebarkan klaster di seluruh Availability Zone menyediakan redundansi dan ketersediaan tinggi.

Untuk informasi selengkapnya tentang klaster, lihat Mengelola AWS CloudHSM cluster.

Untuk membuat klaster, lihat Mulai.

Pengguna HSM

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

Pengguna HSM berbeda dari pengguna IAM. Pengguna IAM yang memiliki kredensi yang benar dapat membuat HSM dengan berinteraksi dengan sumber daya melalui AWS API. Karena enkripsi E2E tidak terlihat oleh AWS, Anda harus menggunakan kredensi 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 kredensi yang Anda tentukan menggunakan CloudHSM CLI.

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

Kunci HSM

AWS CloudHSM memungkinkan Anda membuat, menyimpan, dan mengelola kunci enkripsi dengan aman di HSM penyewa tunggal yang ada di klaster AWS CloudHSM Anda. 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 (HMAC).
- Bungkus dan lindungi kunci lainnya.
- Akses data acak yang aman secara kriptografis.

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 menyediakan banyak jenis dan algoritma utama untuk dipilih. algoritma mendukung berbagai ukuran kunci. Untuk informasi lebih lanjut, lihat atribut dan halaman mekanisme masing-masingAWS CloudHSM SDK Klien.

Bagaimana Anda mengelola kunci

> AWS CloudHSMkunci dikelola melalui SDK dan alat baris perintah. Untuk informasi tentang cara menggunakan alat ini untuk mengelola kunci, lihat Mengelola kunci di AWS CloudHSM danPraktik terbaik untuk AWS CloudHSM.

Siapa yang memiliki kunci

Pada tahunAWS CloudHSM, pengguna kripto (CU) yang membuat kunci memilikinya. Pemilik dapat menggunakan key unshare perintah key share and untuk berbagi dan membatalkan pembagian kunci dengan CU lain. Untuk informasi selengkapnya, lihat Menggunakan CloudHSM CLI untuk berbagi dan membatalkan berbagi kunci.

Akses dan penggunaan dapat dikontrol dengan enkripsi berbasis atribut

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

SDK Klien

Saat menggunakanAWS CloudHSM, Anda melakukan operasi kriptografi dengan Kit Pengembangan Perangkat Lunak AWS CloudHSM Klien (SDK). AWS CloudHSM SDK klien meliputi:

- Standar Kriptografi Kunci Publik #11 (PKCS #11)
- Penyedia JCE
- OpenSSL Dynamic Engine
- API Kriptografi: Generasi Berikutnya (CNG) dan penyedia penyimpanan kunci (KSP) untuk Microsoft Windows

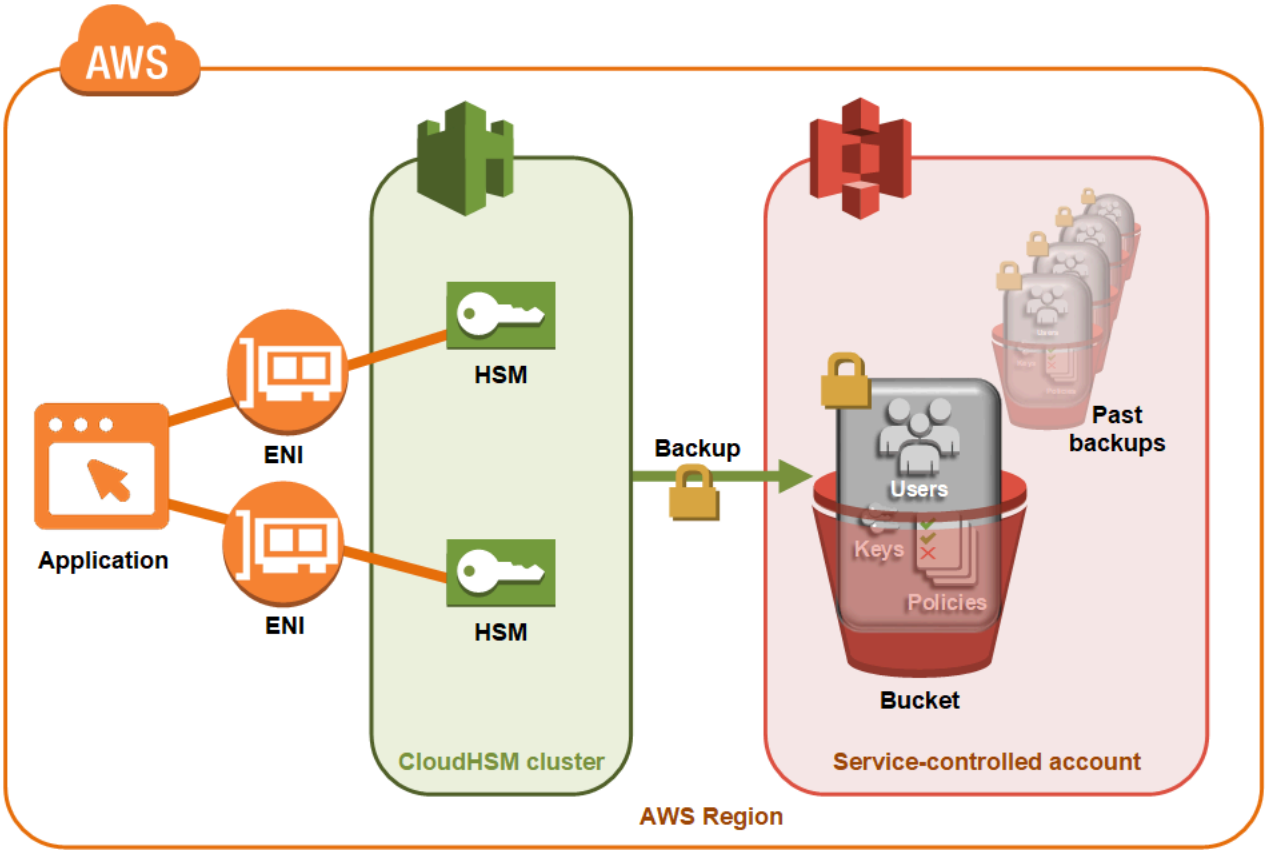
Anda dapat menggunakan salah satu atau semua SDK ini di AWS CloudHSM cluster Anda. Tulis kode aplikasi Anda untuk menggunakan SDK ini untuk melakukan operasi kriptografi di HSM Anda.

Alat utilitas dan baris perintah diperlukan tidak hanya untuk menggunakan SDK tetapi juga untuk mengonfigurasi kredensi, kebijakan, dan pengaturan aplikasi Anda. Untuk informasi lebih lanjut, lihatAWS CloudHSM alat baris perintah.

Untuk informasi lebih lanjut tentang menginstal dan menggunakan SDK Klien atau keamanan sambungan klien, lihat SDK Klien dan nd-to-end Enkripsi E.

AWS CloudHSMcadangan kluster

AWS CloudHSM membuat cadangan periodik pengguna, kunci, dan kebijakan dalam klaster. 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 Mengelola cadangan.

Keamanan

> Saat 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 klaster

Daya tahan

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

Wilayah

Untuk informasi tentang Wilayah yang didukungAWS CloudHSM, lihat AWS CloudHSMWilayah 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 mempengaruhi performa, karena AWS CloudHSM secara otomatis menyeimbangkan beban di semua HSM dalam sebuah klaster.

Seperti sebagian besar sumber daya AWS, klaster dan HSM adalah 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 CloudHSMcadangan kluster.

Harga

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

Memulai dengan AWS CloudHSM

Topik berikut membantu Anda membuat, menginisialisasi, dan mengaktifkan AWS CloudHSM klaster. Setelah menyelesaikan prosedur ini, Anda akan siap mengelola pengguna, mengelola klaster, dan menggunakan pustaka perangkat lunak yang disertakan untuk melakukan operasi kriptografi.

Daftar Isi

Buat grup administratif IAM

Sebagai praktik terbaik, jangan gunakan Anda Pengguna root akun AWS untuk berinteraksiAWS, termasukAWS CloudHSM. Sebagai ganti, gunakan AWS Identity and Access Management (IAM) untuk membuat pengguna IAM, IAM role, atau pengguna gabungan. 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 menciptakan grup administrator AWS CloudHSM yang hanya memiliki izin yang diperlukan untuk menjalankan AWS CloudHSM. Tambahkan pengguna individu ke grup ini

sesuai kebutuhan. Setiap pengguna mewarisi izin terbatas yang dilampirkan ke grup daripada akses AWS penuh. Bagian Kebijakan yang dikelola pelanggan untuk AWS CloudHSM yang berikutnya berisi kebijakan yang harus Anda lampirkan ke grup administrator AWS CloudHSM Anda.

AWS CloudHSMmendefinisikan peran terkait layanan untuk akun Anda. AWS Peran terkait layanan saat ini menentukan izin yang memungkinkan akun Anda untuk 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 Akun AWS

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

Untuk mendaftar Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk secara online.

   Anda akan diminta untuk menerima panggilan telepon dan memasukkan kode verifikasi pada keypad telepon sebagai bagian dari prosedur pendaftaran.

   Saat Anda mendaftar Akun AWS, Pengguna root akun AWS akan dibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya dalam akun. Sebagai praktik terbaik keamanan, tetapkan akses administratif ke pengguna administratif, dan hanya gunakan pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS akan mengirimkan email konfirmasi kepada Anda setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun saat ini dan mengelola akun dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Membuat pengguna administratif

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

Mengamankan Pengguna root akun AWS Anda

1. Masuk ke AWS Management Console sebagai pemilik akun dengan memilih Pengguna root dan memasukkan alamat email Akun AWS Anda. Di halaman berikutnya, masukkan kata sandi Anda.

   Untuk bantuan masuk menggunakan pengguna root, lihat Masuk sebagai pengguna root dalam Panduan Pengguna AWS Sign-In.

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

   Untuk petunjuknya, silakan lihat Mengaktifkan perangkat MFA virtual untuk pengguna root Akun AWS Anda (konsol) dalam Panduan Pengguna IAM.

Membuat pengguna 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 sebuah pengguna administratif.

   Untuk mendapatkan tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, silakan lihat Mengonfigurasi akses pengguna dengan Direktori Pusat Identitas IAM default di Panduan Pengguna AWS IAM Identity Center.

Masuk sebagai pengguna administratif

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

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal akses AWS dalam Panduan Pengguna AWS Sign-In.

Contoh kebijakan untuk AWS CloudHSM yang dapat dilampirkan ke grup pengguna IAM, lihat Identitas dan manajemen akses untuk AWS CloudHSM.

Membuat virtual private cloud (VPC)

Jika Anda belum memiliki virtual private cloud (VPC), ikuti langkah-langkah dalam topik ini untuk buatlah.

> **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 dari Wilayah AWS tempat AWS CloudHSM saat ini didukung.
3. Pilih tombol Buat VPC.
4. Agar Sumber Daya dapat dibuat, pilih VPC dan lainnya.
5. Untuk Nama tanda otomatis, ketik nama yang dapat diidentifikasi seperti**`CloudHSM`**.
6. Biarkan semua pilihan lain diatur ke defaultnya.
7. Pilih Buat VPC.
8. Setelah VPC dibuat, pilih Lihat VPC untuk melihat VPC yang baru saja Anda buat.

Membuat klaster

Cluster adalah kumpulan HSM individu. AWS CloudHSM menyinkronkan HSM di setiap cluster sehingga berfungsi sebagai unit logis.

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

> **Important**
>
> Saat Anda membuat klaster, AWS CloudHSM buat peran terkait layanan bernama. AWSServiceRoleForCloudHSM Jika AWS CloudHSM tidak dapat membuat peran atau peran tersebut belum ada, Anda mungkin tidak dapat membuat klaster. Untuk informasi selengkapnya, lihat Menyelesaikan kegagalan pembuatan cluster. Untuk informasi lebih lanjut tentang peran terkait layanan, lihat Peran terkait layanan untuk AWS CloudHSM.

Anda dapat membuat cluster dari AWS CloudHSM konsol, AWS Command Line Interface (CLI), atau API. AWS CloudHSM

Untuk membuat klaster (konsol)

1. Buka AWS CloudHSM konsol di https://console.aws.amazon.com/cloudhsm/home.
2. Pada bilah navigasi, gunakan pemilih wilayah untuk memilih salah satu AWS Wilayah yang saat AWS CloudHSM ini didukung.
3. Pilih Buat klaster.
4. Di bagian Konfigurasi klaster, lakukan hal berikut:
   a. Untuk VPC, pilih VPC yang Anda buat. Membuat virtual private cloud (VPC)
   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, kinerja tidak akan terpengaruh, karena AWS CloudHSM secara otomatis memuat saldo di semua HSM dalam sebuah cluster. Lihat AWS CloudHSM Wilayah dan Titik Akhir di bagian Referensi Umum AWSuntuk melihat dukungan Availability Zone. AWS CloudHSM
5. Pilih Selanjutnya.
6. Tentukan berapa lama layanan harus mempertahankan cadangan.

> **Note**
>
> 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 Mengkonfigurasi 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.

Untuk membuat cluster (CLI)

- Pada jendela perintah, jalankan perintah create-cluster. Tentukan tipe instans HSM, periode retensi cadangan, dan ID subnet dari subnet tempat Anda berencana untuk membuat HSM. Gunakan ID subnet dari subnet privat yang Anda buat. Tentukan hanya satu subnet per Availability Zone.

```
$ aws cloudhsmv2 create-cluster --hsm-type hsm1.medium \
      --backup-retention-policy Type=DAYS,Value=<number of days> \
      --subnet-ids <subnet ID>

{
    "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",
        "Certificates": {},
        "State": "CREATE_IN_PROGRESS",
        "Hsms": [],
        "ClusterId": "cluster-igklspoyj5v",
        "CreateTimestamp": 1502423370.069
    }
}
```

Untuk membuat cluster (AWS CloudHSM API)

- Kirim permintaan CreateCluster. Tentukan tipe instans HSM, kebijakan retensi cadangan, dan ID subnet dari subnet tempat Anda berencana untuk membuat HSM. Gunakan ID subnet dari subnet privat yang Anda buat. Tentukan hanya satu subnet per Availability Zone.

Jika upaya Anda untuk membuat sebuah klaster gagal, itu mungkin terkait dengan masalah pada peran terkait layanan AWS CloudHSM . Untuk bantuan menyelesaikan kegagalan, lihat Menyelesaikan kegagalan pembuatan cluster.

Tinjau grup keamanan klaster

Saat Anda membuat klaster, AWS CloudHSM membuat grup keamanan dengan nama `cloudhsm-cluster-`*`clusterID`*`-sg`. Grup keamanan ini berisi aturan TCP yang telah dikonfigurasi yang memungkinkan komunikasi masuk dan keluar dalam grup keamanan klaster mellui port 2223-2225. SG ini memungkinkan instans EC2 Anda menggunakan VPC Anda untuk berbicara dengan HSM di cluster Anda.

⚠ 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 HSM Anda.
- Grup keamanan klaster mencegah akses tidak sah ke HSM Anda. Siapa pun yang dapat mengakses instans dalam grup keamanan dapat mengakses HSM Anda. Sebagian besar operasi mewajibkan pengguna untuk login ke HSM. Namun, dimungkinkan untuk mengenolkan HSM tanpa autentikasi, 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.

Pada langkah selanjutnya, Anda bisa meluncurkan instans Amazon EC2dan menghubungkannya ke HSM Anda dengan melampirkan grup keamanan klaster padanya.

Luncurkan instans klien Amazon EC2

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
- AWS 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 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. Pilih Pilih grup keamanan yang ada.
12. Di grup keamanan umum, pilih grup keamanan default dari menu tarik-turun.
13. Di Configure Storage, gunakan menu drop-down untuk memilih konfigurasi penyimpanan.
14. Di jendela Ringkasan, pilih Launch instance.

    > **Note**
    >
    > Menyelesaikan langkah ini akan memulai proses pembuatan 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 Instans 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 , lihatMemulai 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 perintah CLI yang terkandung dalam panduan ini. Jika CLI tidak diinstal, Anda dapat 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.

Konfigurasikan grup keamanan instans Amazon EC2 Klien

Ketika Anda meluncurkan instans Amazon EC2, Anda mengaitkannya dengan grup keamanan Amazon VPC default. Topik ini menjelaskan cara mengaitkan grup keamanan klaster 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 klaster dengan instans tersebut.

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, pilihRDP. 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.

Hubungkan instans Amazon EC2 ke cluster AWS CloudHSM

Anda harus melampirkan grup keamanan klaster ke instans EC2, sehingga instans EC2 dapat berkomunikasi dengan HSM di klaster 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

Setelah Anda membuat klaster, Anda dapat membuat HSM. Namun, sebelum Anda dapat membuat HSM di klaster Anda, klaster harus dalam keadaan tidka diinisialisasi. Untuk menentukan status cluster, lihat halaman cluster di AWS CloudHSM konsol, gunakan CLI untuk menjalankan describe-clusters perintah, atau mengirim DescribeClusterspermintaan di AWS CloudHSM API. Anda dapat membuat HSM dari AWS CloudHSM konsol, CLI, atau AWS CloudHSM API.

Untuk membuat HSM (konsol)

1. Buka AWS CloudHSM konsol di https://console.aws.amazon.com/cloudhsm/home.

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.

Untuk membuat HSM (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",
        "AvailabilityZone": "us-west-2a",
        "ClusterId": "cluster-igklspoyj5v",
        "EniId": "eni-5d7ade72",
        "SubnetId": "subnet-fd54af9b",
        "State": "CREATE_IN_PROGRESS"
    }
}
```

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 klaster Anda (opsional)

Untuk menginisialisasi klaster Anda, Anda menandatangani permintaan penandatanganan sertifikat (CSR) yang dihasilkan oleh HSM klaster pertama. 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 klaster diinisialisasi, Anda tidak dapat menggunakan proses ini untuk mendapatkan sertifikat atau memverifikasi HSM.

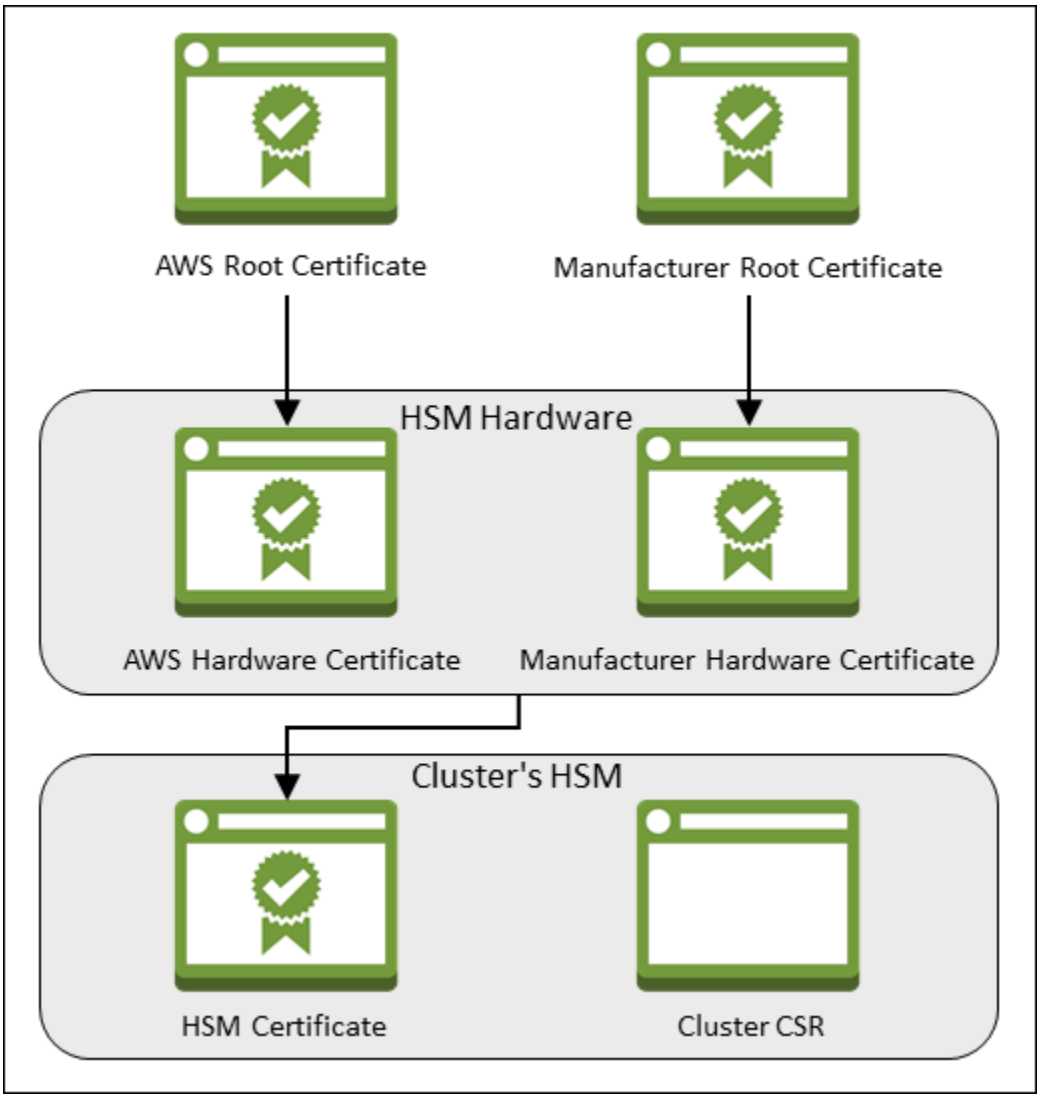
Topik

Gambaran Umum

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

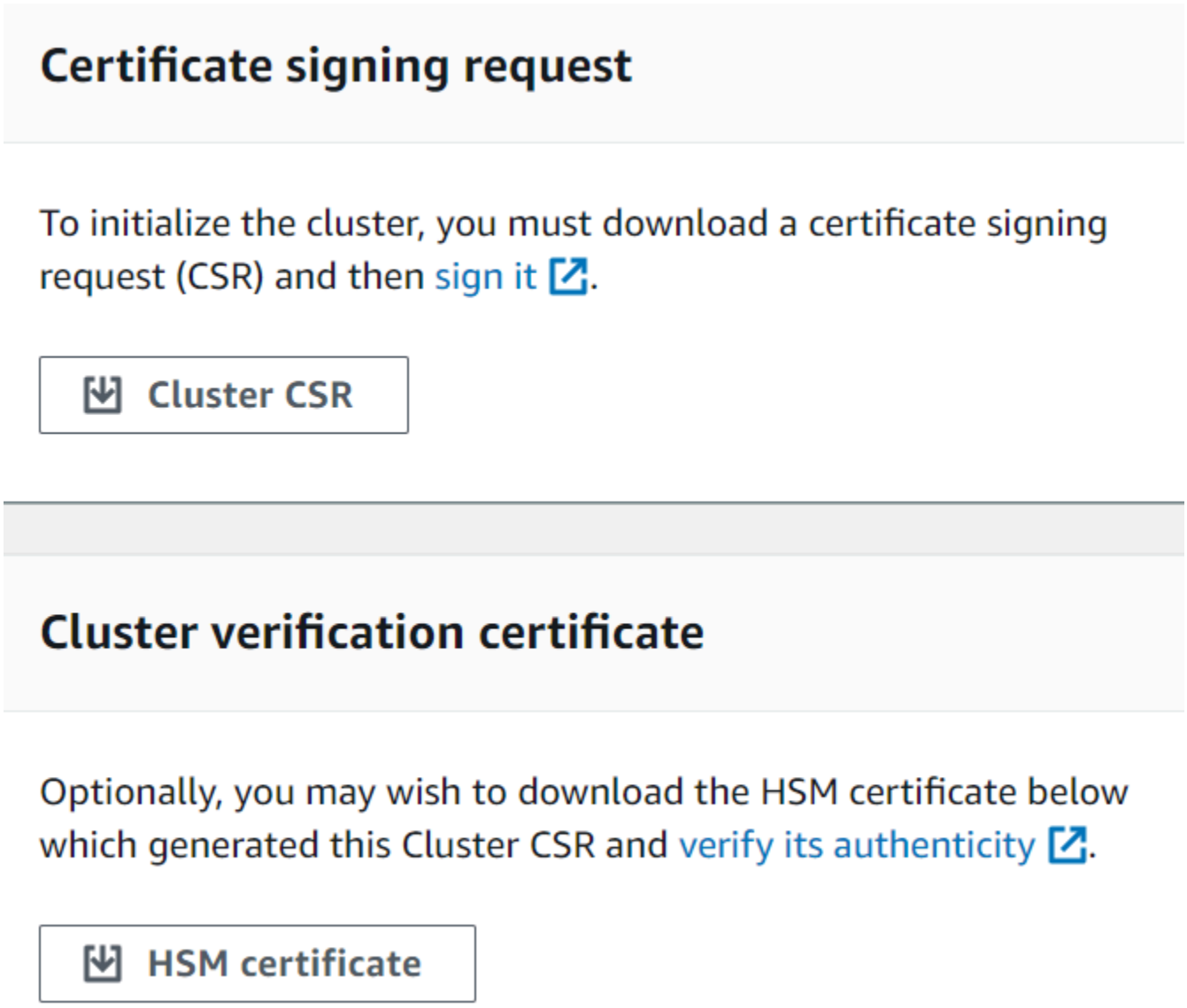
HSM pertama membuat CSR klaster. Saat Anda tandatangani CSR klaster, Anda mengklaim klaster. Kemudian, Anda dapat menggunakan CSR yang ditandatangani untuk menginisialisasi klaster.

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 (CLI), atau AWS CloudHSM API.

Untuk mendapatkan CSR dan sertifikat HSM (konsol)

1. Buka AWS CloudHSM konsol di https://console.aws.amazon.com/cloudhsm/home.
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.



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.

Untuk mendapatkan sertifikat CSR dan HSM (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 *<ID klaster>* dengan ID klaster yang telah 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 *<ID klaster>* dengan ID klaster yang telah 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 *<ID klaster>* dengan ID klaster yang telah 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 *<ID klaster>* dengan ID klaster yang telah Anda buat sebelumnya.

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

Untuk mendapatkan sertifikat CSR dan HSM (API)AWS CloudHSM

- Kirim permintaan DescribeClusters, lalu ekstrak dan simpan CSR dan sertifikat dari respons.

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 produsen: liquid_security_certificate.zip

   Untuk mengunduh sertifikat dari halaman arahan, https://www.marvell.com/products/security-solutions/ liquid-security-hsm-adapters -and-appliances/liquidsecurity-certificate.html, lalu pilih Unduh Sertifikat.

   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.

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 memerlukan CLI untuk langkah ini, dan shell tidak perlu dikaitkan dengan akun Anda AWS .

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 *<ID klaster>* dengan ID klaster yang telah 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 *<ID klaster>* dengan ID klaster yang telah 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 *<ID klaster>* dengan ID klaster yang telah 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 *<ID klaster>* dengan ID klaster yang telah Anda buat sebelumnya.

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

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 memerlukan CLI untuk langkah 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

Selesaikan langkah-langkah dalam topik berikut untuk menginisialisasi AWS CloudHSM cluster Anda.

> **Note**
>
> Sebelum Anda menginisialisasi klaster, tinjau proses yang dengannya Anda dapat memverifikasi identitas dan keaslian HSM. Proses ini opsional dan bekerja hanya sampai klaster diinisialisasi. Setelah klaster diinisialisasi, Anda tidak dapat menggunakan proses ini untuk mendapatkan sertifikat atau memverifikasi HSM.

Topik

- Dapatkan CSR cluster
- Tanda tangani CSR
- Inisialisasi cluster

Dapatkan CSR cluster

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

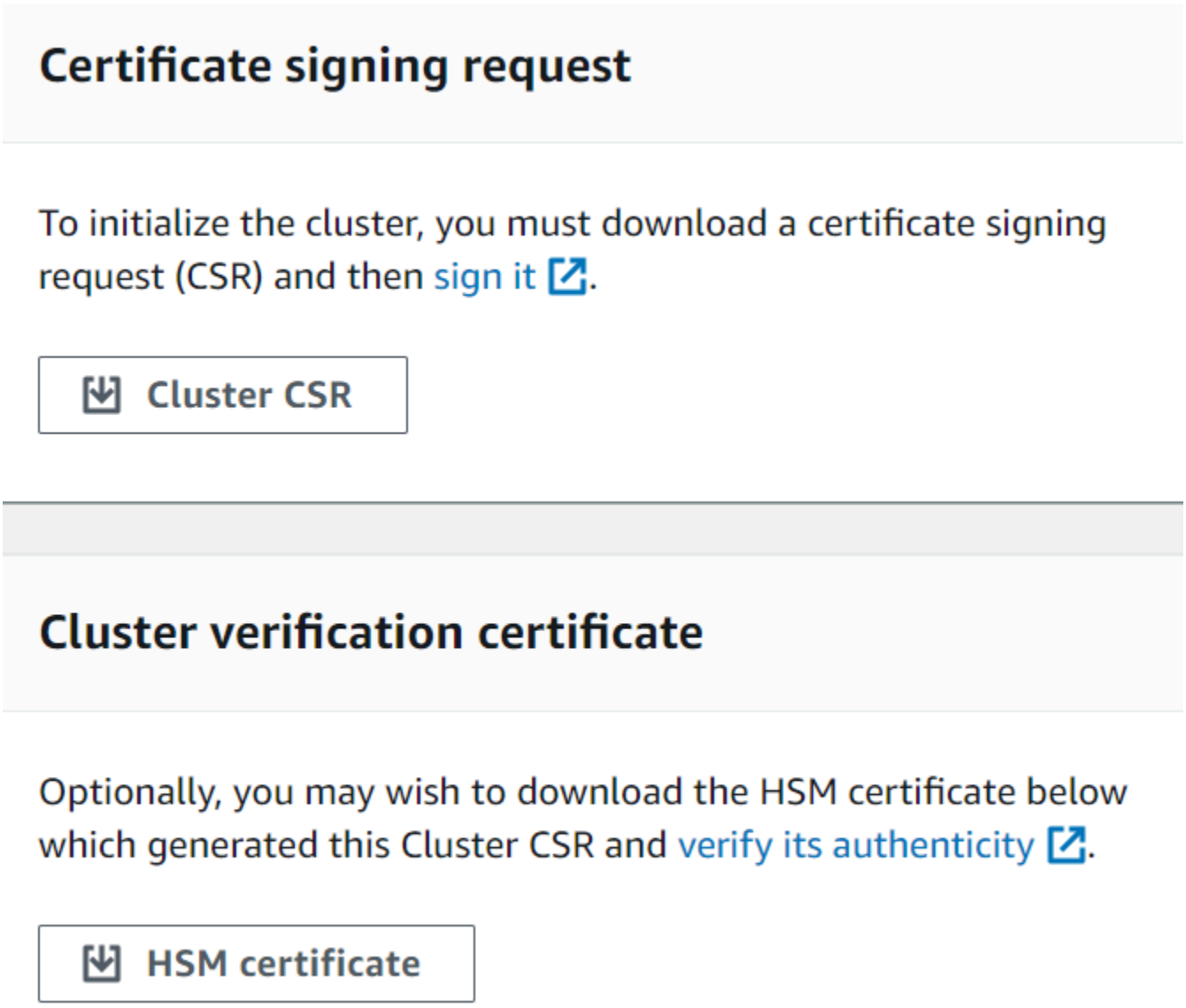
Important

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.

Untuk mendapatkan CSR (konsol)

1. Buka AWS CloudHSM konsol di https://console.aws.amazon.com/cloudhsm/home.
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.



6. Pilih CSR Klaster untuk mengunduh dan menyimpan CSR.

Untuk mendapatkan CSR (CLI)

- Pada prompt perintah, jalankan perintah describe-clusters, yang mengekstraksi CSR dan menyimpannya ke file. Ganti *<ID klaster>* dengan ID klaster yang telah Anda buat sebelumnya.

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

Untuk mendapatkan CSR (AWS CloudHSM API)

1. Kirim permintaan DescribeClusters.
2. Ekstrak dan simpan CSR dari respons.

Tanda tangani CSR

Saat ini, Anda harus membuat sertifikat penandatanganan yang ditandatangani sendiri dan menggunakannya untuk menandatangani CSR untuk klaster Anda. Anda tidak memerlukan CLI untuk langkah ini, dan shell tidak perlu dikaitkan dengan akun Anda AWS . Untuk menandatangani CSR, Anda harus melakukan hal berikut:

1. Lengkapi bagian sebelumnya (lihatDapatkan CSR cluster).
2. Buat kunci privat.
3. Gunakan kunci privat untuk membuat sertifikat penandatanganan.
4. Tanda tangani CSR klaster Anda.

Buat kunci privat

> **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 HSM yang 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.

Gunakan perintah berikut untuk membuat kunci privat. Saat menginisialisasi AWS CloudHSM cluster, Anda harus menggunakan sertifikat RSA 2048 atau sertifikat RSA 4096.

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

Gunakan kunci privat untuk membuat 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 ini menggunakan OpenSSL dan kunci privat yang Anda buat di langkah sebelumnya untuk membuat sertifikat penandatanganan. Sertifikat ini berlaku selama 10 tahun (3652 hari). Baca petunjuk di layar dan ikuti prompt-nya.

```
$ openssl req -new -x509 -days 3652 -key customerCA.key -out customerCA.crt
Enter pass phrase for customerCA.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 `customerCA.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 cluster

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 menggunakan kunci privat Anda dan sertifikat yang ditandatangani sendiri yang Anda buat di langkah sebelumnya.

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

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

Inisialisasi cluster

Gunakan sertifikat HSM yang ditandatangani dan sertifikat penandatanganan Anda untuk menginisialisasi klaster Anda. Anda dapat menggunakan AWS CloudHSM konsol, CLI, atau API. AWS CloudHSM

Untuk menginisialisasi klaster (konsol)

1. Buka AWS CloudHSM konsol di https://console.aws.amazon.com/cloudhsm/home.
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 klaster, pilih Unggah file. Kemudia, 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. Jika Anda menyelesaikan langkah-langkah di bagian sebelumnya, pilih file bernama customerCA.crt.

c. Pilih Unggah dan inisialisasi.

Untuk menginisialisasi cluster (CLI)

- Pada prompt perintah, jalankan perintah initialize-cluster. Berikan yang berikut ini:
  - ID klaster 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 bertanda tangan Anda. Jika Anda menyelesaikan langkah-langkah di bagian sebelumnya, sertifikat penandatanganan tersimpan dalam file dengan nama customerCA.crt.

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

Untuk menginisialisasi cluster (AWS CloudHSM API)

- Kirim permintaan InitializeCluster dengan yang berikut ini:
  - ID klaster yang Anda buat sebelumnya.
  - Sertifikat HSM yang Anda tandatangani sebelumnya.

- Sertifikat bertanda tangan Anda.

Instal dan konfigurasikan CloudHSM CLI

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

Tugas

- Instal alat baris AWS CloudHSM perintah

Instal alat baris AWS CloudHSM perintah

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.

Gunakan perintah berikut untuk mengunduh dan menginstal CloudHSM CLI.

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 pada arsitektur ARM64:

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

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

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 pada arsitektur ARM64:

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

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

CentOS 7 (7.8+)

CentOS 7 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
```

RHEL 7 (7.8+)

RHEL 7 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
```

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 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 pada arsitektur ARM64:

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

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

Ubuntu 20.04 LTS

Ubuntu 20.04 LTS pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Focal/cloudhsm-cli_latest_u20.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u20.04_amd64.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 arsitektur ARM64:

```
$ 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 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 msiexec.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 msiexec.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 IP 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.

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

Mengaktifkan klaster

Saat Anda mengaktifkan klaster AWS CloudHSM, kondisi klaster berubah dari diinisialisasi menjadi aktif. Anda kemudian dapat mengelola pengguna modul keamanan perangkat keras (HSM) dan menggunakan HSM.

Important

Sebelum Anda dapat mengaktifkan klaster, Anda harus menyalin sertifikat penerbitan ke lokasi default untuk platform pada setiap instans EC2 yang menghubungkan ke klaster (Anda membuat sertifikat penerbitan ketika Anda menginisialisasi klaster).

Linux

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

Windows

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

Setelah menempatkan sertifikat penerbitan, instal CloudHSM CLI dan jalankan cluster activateperintah 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 klaster. Saat Anda mengaktifkan klaster, peran admin yang tidak aktif akan berubah menjadi admin.

Mengaktifkan klaster

1. Connect ke instans klien yang Anda luncurkan sebelumnya. Untuk informasi selengkapnya, lihat Luncurkan instans klien Amazon EC2. Anda dapat meluncurkan instans Linux atau Windows Server.

2. Jalankan CloudHSM CLI dalam mode interaktif.

   Linux

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

   Windows

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

3. (Opsional) 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. Gunakancluster 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) Gunakanuser list perintah untuk memverifikasi bahwa jenis 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. Gunakanquit perintah untuk menghentikan alat CloudHSM CLI.

```
aws-cloudhsm > quit
```

Untuk informasi lebih lanjut tentang bekerja dengan CloudHSM CLI atau CMU, lihat Memahami Pengguna HSM dan Memahami Manajemen Pengguna HSM dengan CMU.

Konfigurasi ulang SSL dengan sertifikat baru dan kunci pribadi (opsional)

AWS CloudHSM menggunakan sertifikat SSL untuk membuat sambungan ke HSM. Kunci default dan sertifikat SSL disertakan ketika Anda menginstal klien. Namun, Anda dapat membuat dan menggunakan milik Anda sendiri. Perhatikan bahwa Anda memerlukan sertifikat yang ditandatangani sendiri (*customerCA.crt*) yang Anda buat saat Anda mengisialisasi klaster Anda.

Pada tingkat tinggi, ini adalah proses dua langkah:

1. Pertama, Anda membuat kunci privat, kemudian gunakan kunci tersebut untuk membuat permintaan penandatanganan sertifikat (CSR). Gunakan sertifikat penerbitan, sertifikat yang Anda buat saat Anda menginisialisasi klaster, untuk menandatangani CSR.
2. Selanjutnya, Anda menggunakan alat konfigurasi untuk menyalin kunci dan sertifikat ke direktori yang sesuai.

Buat kunci, CSR, lalu tandatangani CSR

Langkah-langkah yang sama untuk SDK Klien 3 atau SDK Klien 5.

Untuk konfigurasi ulang SSL dengan sertifikat baru dan kunci privat

1. Buat kunci privat menggunakan perintah OpenSSL berikut:

```
openssl genrsa -out ssl-client.key 2048
Generating RSA private key, 2048 bit long modulus
........+++
............+++
e is 65537 (0x10001)
```

2. Gunakan perintah OpenSSL berikut untuk membuat permintaan penandatanganan sertifikat (CSR). Anda akan ditanya serangkaian pertanyaan untuk sertifikat Anda.

```
openssl req -new -sha256 -key ssl-client.key -out ssl-client.csr
Enter pass phrase for ssl-client.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) [XX]:
State or Province Name (full name) []:
Locality Name (eg, city) [Default City]:
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

3. Tanda tangani CSR dengan sertifikat *customerCA.crt* yang Anda buat ketika Anda menginisialisasi klaster Anda.

```
openssl x509 -req -days 3652 -in ssl-client.csr \
        -CA customerCA.crt \
        -CAkey customerCA.key \
        -CAcreateserial \
        -out ssl-client.crt
Signature ok
subject=/C=US/ST=WA/L=Seattle/O=Example Company/OU=sales
Getting CA Private Key
```

Aktifkan SSL kustom untuk AWS CloudHSM

Langkahnya berbeda untuk SDK Klien 3 atau SDK Klien 5. Untuk informasi selengkapnya tentang bekerja dengan alat baris perintah konfigurasi, lihat ???.

Topik

- SSL Kustom untuk SDK Klien 3
- SSL Kustom untuk SDK Klien 5

SSL Kustom untuk SDK Klien 3

Gunakan alat konfigurasi untuk SDK Klien 3 untuk mengaktifkan SSL kustom. Untuk informasi lebih lanjut tentang alat konfigurasi untuk SDK Klien 3, lihat ???.

Untuk menggunakan sertifikat kustom dan kunci untuk autentikasi mutual server-klien TLS dengan SDK Klien 3 di Linux

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

```
sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

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

```
sudo /opt/cloudhsm/bin/configure --ssl \
 --pkey /opt/cloudhsm/etc/ssl-client.key \
 --cert /opt/cloudhsm/etc/ssl-client.crt
```

3. Tambahkan sertifikat `customerCA.crt` ke penyimpanan kepercayaan. Membuat hash dari nama subjek sertifikat. Hal ini membuat indeks untuk memungkinkan sertifikat dicari berdasarkan nama itu.

```
openssl x509 -in /opt/cloudhsm/etc/customerCA.crt -hash | head -n 1
1234abcd
```

   Buatlah sebuah direktori.

```
mkdir /opt/cloudhsm/etc/certs
```

Buat file yang berisi sertifikat dengan nama hash.

```
sudo cp /opt/cloudhsm/etc/customerCA.crt /opt/cloudhsm/etc/certs/1234abcd.0
```

SSL Kustom untuk SDK Klien 5

Gunakan salah satu alat konfigurasi SDK Klien 5 untuk mengaktifkan SSL kustom. Untuk informasi lebih lanjut tentang alat konfigurasi untuk SDK Klien 5, lihat ???.

PKCS #11 library

Untuk menggunakan sertifikat kustom dan kunci untuk autentikasi mutual server-klien TLS dengan Klien SDK 5 di Linux

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

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

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

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

Untuk menggunakan sertifikat kustom dan kunci untuk autentikasi mutual klien-server TLS dengan Klien SDK 5 pada Windows

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

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

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

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" `
```

```
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

OpenSSL Dynamic Engine

Untuk menggunakan sertifikat kustom dan kunci untuk autentikasi mutual server-klien TLS dengan Klien SDK 5 di Linux

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

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

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

```
$ sudo /opt/cloudhsm/bin/configure-dyn \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

JCE provider

Untuk menggunakan sertifikat kustom dan kunci untuk autentikasi mutual server-klien TLS dengan Klien SDK 5 di Linux

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

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

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

```
$ sudo /opt/cloudhsm/bin/configure-jce \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

Untuk menggunakan sertifikat kustom dan kunci untuk autentikasi mutual klien-server TLS dengan Klien SDK 5 pada Windows

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

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

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

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

CloudHSM CLI

Untuk menggunakan sertifikat kustom dan kunci untuk autentikasi mutual server-klien TLS dengan Klien SDK 5 di Linux

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

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

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

```
$ sudo /opt/cloudhsm/bin/configure-cli \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

Untuk menggunakan sertifikat kustom dan kunci untuk autentikasi mutual klien-server TLS dengan Klien SDK 5 pada Windows

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

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
```

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

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

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

Bangun Aplikasi

Bangun aplikasi dan kerja dengan kunci menggunakan AWS CloudHSM.

Untuk mulai membuat dan menggunakan kunci di klaster baru, Anda harus terlebih dahulu membuat pengguna modul keamanan perangkat keras (HSM) dengan CloudHSM Management Utility (CMU). Untuk informasi lebih lanjut, lihat Memahami Tugas Manajemen Pengguna HSM, Memulai AWS CloudHSM Command Line Interface (CLI), dan Cara Mengelola Pengguna HSM.

> **Note**
>
> Jika menggunakan SDK Klien 3, gunakan CloudHSM Management Utility (CMU), bukan CloudHSM CLI.

Dengan memasang pengguna HSM, Anda dapat masuk ke HSM dan membuat serta menggunakan kunci dengan salah satu opsi berikut:

- 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 denganserver web NGINX dan Apache
- Gunakan penyedia CNG dan KSP untuk menggunakan AWS CloudHSM dengan Otoritas Sertifikasi (CA) Microsoft Windows Server

- Gunakan penyedia CNG dan KSP untuk menggunakan AWS CloudHSM dengan Alat Tanda Tangan Microsoft
- Gunakan penyedia CNG dan KSP untuk pengosongan TLS 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

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 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 HSM agar layanan Anda 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.

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

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

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

Akses aman ke klaster Anda

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

Gunakan titik akhir VPC untuk mengakses API: Pesawat AWS CloudHSM data dirancang untuk beroperasi tanpa memerlukan akses ke internet atau AWS API. 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.

Konfigurasi ulang SSL untuk mengamankan komunikasi client-server: AWS CloudHSM menggunakan TLS untuk membuat koneksi ke HSM Anda. Setelah Anda menginisialisasi cluster Anda, Anda dapat mengganti sertifikat TLS default dan kunci yang digunakan untuk membuat koneksi TLS luar. Untuk informasi selengkapnya, lihat Tingkatkan keamanan server web Anda dengan pembongkaran SSL/TLS di AWS CloudHSM.

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) HSM Anda menjadi nol ketika tidak diperlukan. Ketika HSM diperlukan lagi, Anda dapat memulihkan HSM Anda 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

HSM Anda 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 HSM di 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:

- Ikhtisar umum cadangan
- Kebijakan retensi cadangan
- Menyalin cadangan di seluruh Wilayah AWS

Manajemen pengguna HSM

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 HSM dengan berinteraksi dengan sumber daya melalui AWS API. Setelah HSM dibuat, Anda harus menggunakan kredensyal pengguna HSM untuk mengautentikasi operasi pada HSM. Untuk panduan rinci tentang pengguna HSM, lihatMengelola pengguna HSM di AWS CloudHSM.

Lindungi kredensi pengguna HSM Anda

Sangat penting untuk menjaga kredensyal 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 kredensyal 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 (CO) 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 kripto 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 kripto lain yang kemudian menggunakannya dalam aplikasi Anda).

Sumber daya terkait:

- pembagian kunci
- kunci unshare

Manajemen kunci HSM

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. HSM ekstra di cluster Anda tidak akan meningkatkan throughput permintaan untuk kunci itu. Jika Anda menggunakan kunci token untuk aplikasi yang sama, permintaan Anda akan diseimbangkan di semua HSM yang tersedia di klaster Anda. Untuk informasi selengkapnya, lihat Sinkronisasi kunci dan pengaturan daya tahan di AWS CloudHSM.

Kelola batas penyimpanan utama

HSM memiliki batasan jumlah maksimum token dan kunci sesi yang dapat disimpan pada HSM pada satu waktu. Untuk informasi tentang batas penyimpanan utama, lihatKuota AWS CloudHSM. 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 mengarahkan permintaan klien Anda ke klaster dengan kunci yang diperlukan. Untuk informasi tentang cara menyambung ke beberapa cluster dari aplikasi klien yang sama, lihat topik berikut:

- Menghubungkan ke beberapa klaster dengan penyedia JCE
- Menghubungkan ke beberapa slot dengan PKCS #11

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 untuk hanya membungkus/membuka kunci yang telah secara eksplisit ditandai sebagai dipercaya oleh admin. Untuk informasi selengkapnya, lihat topik tentang pembungkus kunci tepercaya. Mengelola kunci di AWS CloudHSM

Sumber daya terkait

- Fungsi Bungkus dan Buka Bungkus
- Fungsi cipher untuk JCE
- Atribut kunci Java yang didukung
- Atribut kunci untuk CloudHSM CLI

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 login 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, lihatMode Perintah Tunggal. AWS CloudHSM merekomendasikan untuk menyimpan kredensyal 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

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 kredensyal untuk aplikasi Anda ketika menggunakan integrasi yang tidak memungkinkan Anda untuk memiliki kontrol atas kode aplikasi Anda. Untuk selengkapnya tentang metode login, lihatMemberikan kredensi kepada penyedia JCE.

Otentikasi dengan OpenSSL: Dengan OpenSSL Dynamic Engine, Anda memberikan kredensyal melalui variabel lingkungan. AWS CloudHSM merekomendasikan untuk menyimpan kredensyal 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. Memasang OpenSSL Dynamic Engine

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 SDK Klien berikut:

- 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 findKey dandaftar kunci) 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 HSM yang tersedia di klaster 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 rekomendasikan 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 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:

- Menghubungkan ke beberapa slot dengan PKCS #11
- Menghubungkan ke beberapa klaster dengan penyedia JCE
- Menghubungkan ke beberapa cluster dengan CLI

Memulihkan klaster dari cadangan: Anda dapat membuat Cluster baru dari cadangan Cluster yang ada. Untuk informasi selengkapnya, lihat Mengelola AWS CloudHSM cadangan.

Memantau

Bagian ini menjelaskan beberapa mekanisme yang dapat Anda gunakan untuk memantau klaster dan aplikasi Anda. Untuk detail tambahan tentang pemantauan, lihatAWS CloudHSM Pemantauan.

Pantau log klien

Setiap SDK Klien menulis log yang dapat Anda pantau. Untuk informasi tentang pencatatan klien, lihatBekerja dengan log SDK 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 pencatatan di aplikasi Anda. 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 Bagaimana HSM audit logging bekerja 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 denganAWS CloudTraildanAWS CloudHSM untuk detail.

Pantau CloudWatch metrik Amazon

Anda dapat menggunakan CloudWatch metrik Amazon untuk memantau AWS CloudHSM klaster 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. Kami merekomendasikan untuk mengonfigurasi 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
- HSM yang tidak sehat 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.

Mengelola AWS CloudHSM cluster

Anda dapat mengelola AWS CloudHSM cluster Anda dari AWS CloudHSM konsol atau salah satu AWS SDK atau alat baris perintah. Untuk informasi lebih lanjut, lihat topik berikut.

Untuk membuat klaster, lihat Mulai.

Arsitektur cluster

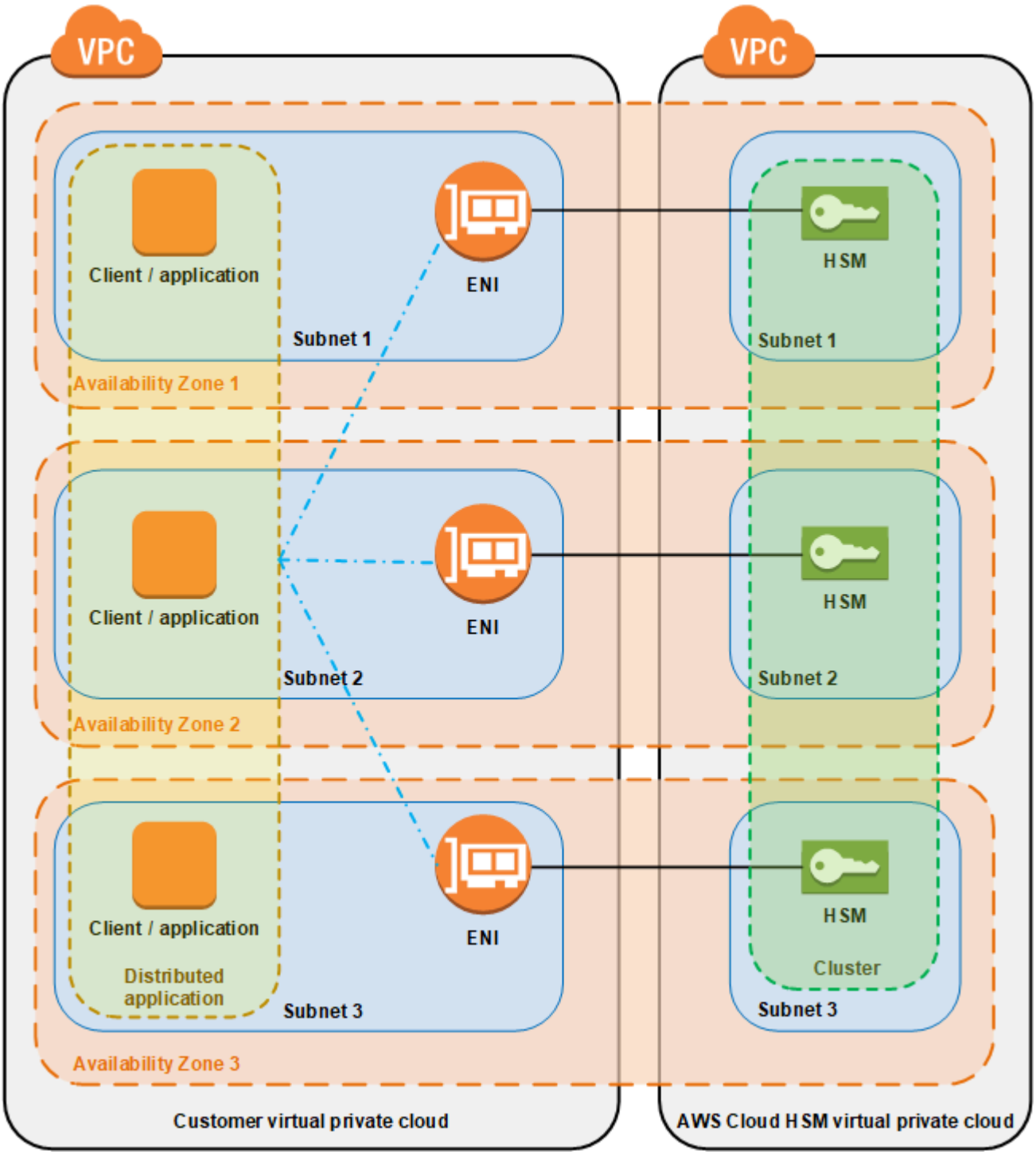
Saat membuat klaster, Anda menentukan Amazon Virtual Private Cloud (VPC) di AWS akun Anda dan satu atau beberapa subnet di VPC tersebut. Kami menyarankan Anda membuat satu subnet di setiap Availability Zone (AZ) di AWS Wilayah pilihan Anda. Anda dapat membuat subnet pribadi saat membuat VPC. Untuk mempelajari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

Setiap kali Anda membuat HSM, Anda menentukan klaster dan Availability Zone untuk HSM. Dengan menempatkan HSM 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 HSM dalam sebuah cluster, Anda memerlukan perangkat lunak AWS CloudHSM klien. Biasanya Anda menginstal klien pada instans Amazon EC2, yang dikenal sebagai instans klien, yang berada di VPC yang sama dengan ENI HSM, seperti yang ditunjukkan pada gambar berikut. Itu tidak secara teknis diperlukan; Anda dapat menginstal klien pada setiap komputer yang kompatibel, selama dapat terhubung ke ENI HSM. Klien berkomunikasi dengan HSM individu dalam klaster Anda melalui ENI mereka.

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



Sinkronisasi cluster

Dalam sebuah AWS CloudHSM cluster, AWS CloudHSM jaga agar kunci pada masing-masing HSM tetap sinkron. Anda tidak perlu melakukan apa pun untuk menyinkronkan kunci di HSM Anda. 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 Menjaga agar pengguna HSM tetap sinkron.

Saat Anda menambahkan HSM baru ke klaster, AWS CloudHSM buat cadangan semua kunci, pengguna, dan kebijakan pada HSM yang ada. Ini kemudian mengembalikan cadangan ke HSM baru. Hal ini membuat dua HSM tetap sinkron.

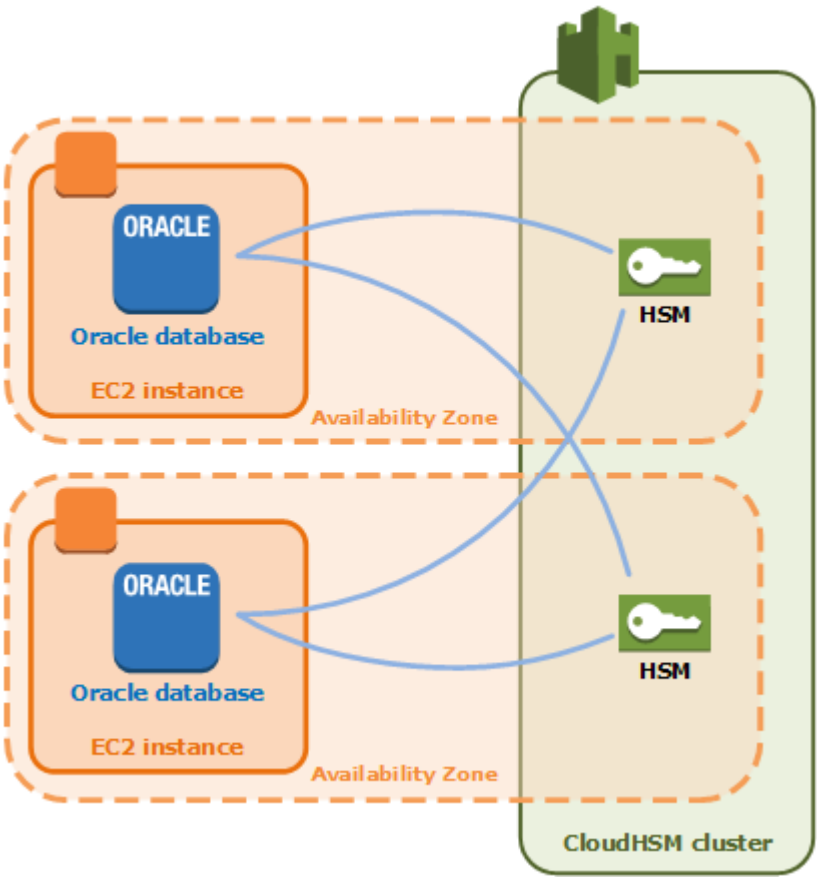
Jika HSM dalam cluster tidak sinkronisasi, AWS CloudHSM secara otomatis menyinkronkannya kembali. Untuk mengaktifkan ini, AWS CloudHSM gunakan kredensi pengguna alat. Pengguna ini ada di semua HSM yang disediakan oleh AWS CloudHSM dan memiliki izin terbatas. Hal ini bisa mendapatkan hash dari objek pada 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.

Ketersediaan klaster tinggi dan penyeimbangan beban

Saat Anda membuat AWS CloudHSM cluster dengan lebih dari satu HSM, Anda secara otomatis mendapatkan load balancing. Penyeimbangan beban berarti bahwa klien AWS CloudHSM mendistribusikan operasi kriptografi di semua HSM di klaster berdasarkan kapasitas masing-masing HSM untuk pemrosesan tambahan.

Saat Anda membuat HSM 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 HSM 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 kedua HSM sehingga mereka segera dapat diakses dan berlebihan.



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.

```
"C:\Program Files\Amazon\CloudHSM\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>
```

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.

```
"C:\Program Files\Amazon\CloudHSM\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.

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

Untuk informasi selengkapnya, lihat Alat Konfigurasi.

Untuk informasi selengkapnya tentang menginisialisasi klaster 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 klaster. Anda dapat menggunakan alamat IP dari setiap HSM yang terlampir pada klaster Anda. Setelah terhubung, Klien SDK mengambil alamat IP dari setiap HSM tambahan dan melakukan penyeimbangan beban dan operasi sinkronisasi kunci sisi klien.

Untuk mendapatkan alamat IP untuk klaster

Untuk mendapatkan alamat IP untuk HSM (konsol)

1. Buka AWS CloudHSM konsol di https://console.aws.amazon.com/cloudhsm/home.
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, pada tab HSM, pilih salah satu alamat IP yang tercantum di bawah alamat IP ENI.

Untuk mendapatkan alamat IP untuk HSM (CLI)

- Dapatkan alamat IP HSM dengan menggunakan describe-clusters perintah dari CLI. Dalam output dari perintah, alamat IP dari HSM adalah nilai-nilai dari `EniIp`.

```
$  aws cloudhsmv2 describe-clusters


{
    "Clusters": [
        { ... }
```

```
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
...
```

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.

```
"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>
```

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.

```
"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 IP 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.

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

> **Note**
>
> Anda dapat menggunakan `--cluster-id` parameter sebagai pengganti`-a <HSM_IP_ADDRESSES>`. Untuk melihat persyaratan penggunaan`--cluster-id`, lihatAlat 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.

```
C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe -a <HSM IP address>
```

Untuk informasi lebih lanjut tentang konfigurasi, lihat ???.

Menambahkan atau menghapus HSM dalam sebuah cluster AWS CloudHSM

Untuk meningkatkan atau menurunkan AWS CloudHSM klaster Anda, tambahkan atau hapus HSM dengan menggunakan AWS CloudHSM konsol atau salah satu AWS SDK atau alat baris perintah. Kami merekomendasikan pengujian beban klaster Anda untuk menentukan beban puncak yang

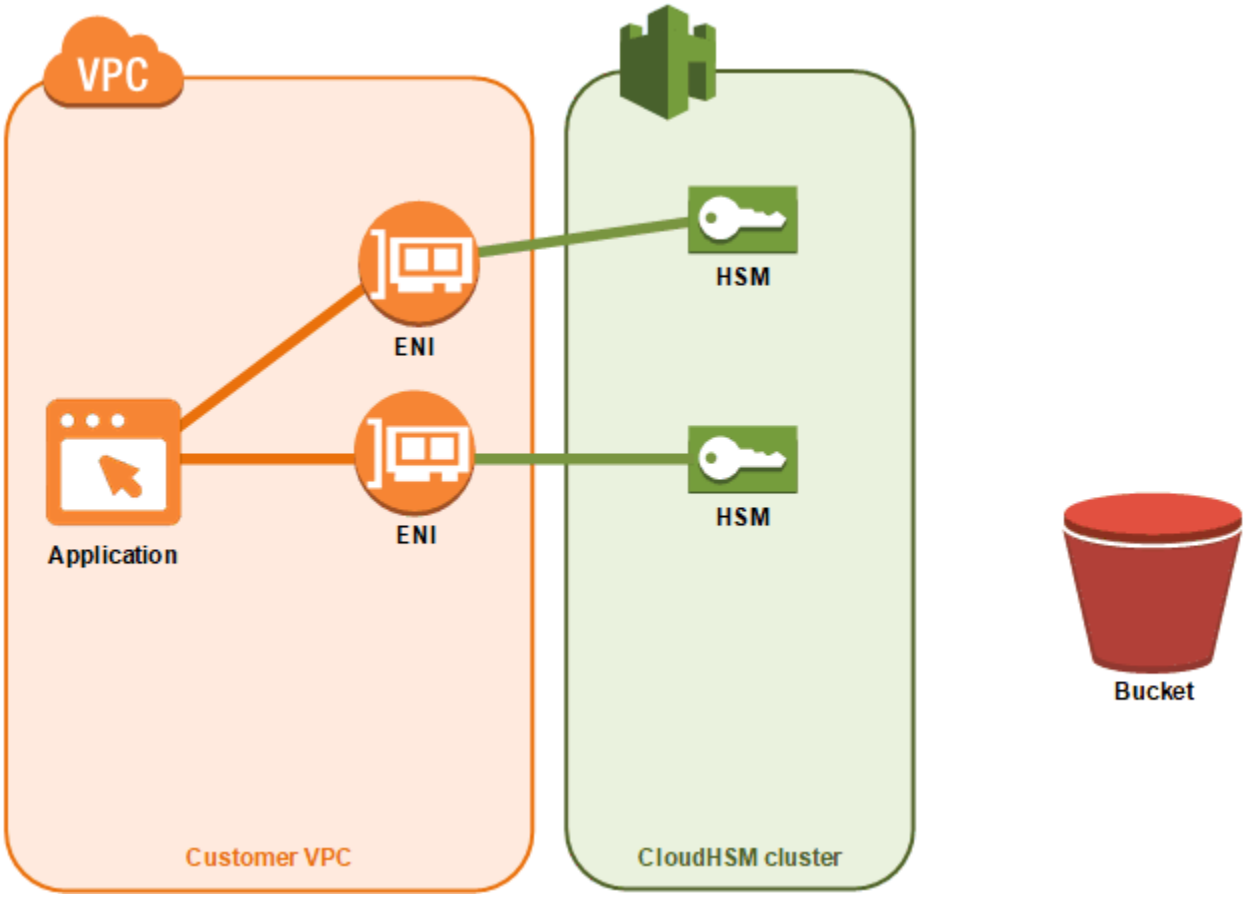
harus Anda antisipasi, dan kemudian menambahkan satu HSM lagi untuk memastikan ketersediaan tinggi.

Topik

- Menambahkan HSM
- Hapus HSM

Menambahkan HSM

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



1. Anda menambahkan HSM baru ke sebuah klaster. Prosedur berikut menjelaskan cara melakukan ini dari AWS CloudHSM konsol, AWS Command Line Interface (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 Cadangan.

3. AWS CloudHSM mengembalikan cadangan ke HSM baru. Hal ini memastikan bahwa HSM sinkron dengan yang lain di klaster.
4. HSM yang ada 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/home.
2. Pilih sebuah klaster untuk HSM yang Anda tambahkan.
3. Pada tab HSM, pilih Buat HSM.
4. Pilih Availability Zone (AZ) untuk HSM yang Anda buat. Lalu, pilih Buat.

Untuk menambahkan HSM (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, keluarka 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"
    }
}
```

Untuk menambahkan HSM (AWS CloudHSM API)

- Kirim permintaan CreateHsm, menentukan ID klaster dan Availability Zone untuk HSM yang Anda buat.

Hapus HSM

Anda dapat menghapus HSM dengan menggunakan AWS CloudHSM konsol, CLI, atau AWS CloudHSM API.

Untuk menghapus HSM (konsol)

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

Untuk menghapus HSM (CLI)

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

  Jika Anda tidak tahu nilai untuk pengenal 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 pengenal untuk HSM yang Anda hapus.

Menghapus sebuah cluster AWS CloudHSM

Sebelum dapat menghapus klaster, Anda harus menghapus semua HSM dari klaster. Untuk informasi selengkapnya, lihat Hapus HSM.

Setelah Anda menghapus semua HSM, Anda dapat menghapus cluster dengan menggunakan AWS CloudHSM konsol, AWS Command Line Interface (CLI), atau AWS CloudHSM API.

Untuk menghapus sebuah klaster (konsol)

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

Untuk menghapus sebuah klaster (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 Mengelola cadangan.

Buat cluster dari cadangan (konsol)

1. Buka AWS CloudHSM konsol di https://console.aws.amazon.com/cloudhsm/home.
2. Pilih Buat klaster.
3. Di bagian Konfigurasi klaster, lakukan hal berikut:
   a. UntukVPC, pilih VPC untuk klaster yang Anda buat.
   b. UntukAZ, pilih subnet privat untuk setiap Availability Zone yang Anda tambahkan ke klaster.
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 pilihBuat 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 Mengkonfigurasi 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 (CLI)

Untuk menentukan ID cadangan, keluarkan perintah describe-backups.

- Pada jendela perintah, keluarkan perintah create-cluster. Tentukan tipe instans HSM, subnet ID dari subnet tempat Anda berencana untuk membuat HSM, dan ID cadangan dari cadangan yang Anda pulihkan.

```
$ aws cloudhsmv2 create-cluster --hsm-type hsm1.medium \
                                --subnet-ids <subnet ID 1> <subnet ID 2> <subnet ID
 N> \
                                --source-backup-id <backup ID>
{
    "Cluster": {
        "HsmType": "hsm1.medium",
        "VpcId": "vpc-641d3c0d",
        "Hsms": [],
        "State": "CREATE_IN_PROGRESS",
        "SourceBackupId": "backup-rtq2dwi2gq6",
        "BackupPolicy": "DEFAULT",
        "BackupRetentionPolicy": {
            "Type": "DAYS",
            "Value": 90
         },
        "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

Mengelola AWS CloudHSM cadangan

AWS CloudHSM membuat backup berkala dari cluster Anda setidaknya sekali setiap 24 jam. Setiap cadangan berisi salinan terenkripsi dari data berikut:

- Pengguna (CO, CU, dan AU)
- 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 kluster. Untuk informasi tentang pengelolaan kebijakan retensi cadangan, lihat Mengkonfigurasi retensi cadangan.

Topik

Menggunakan cadangan

Ketika Anda menambahkan HSM ke cluster yang sebelumnya berisi satu atau lebih HSM aktif, layanan mengembalikan cadangan terbaru ke HSM baru. Gunakan backup untuk mengelola HSM yang jarang 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 “Menghapus dan memulihkan 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.

Menghapus dan memulihkan cadangan

Setelah Anda menghapus cadangan, layanan menyimpan cadangan selama tujuh hari, yang selama itu Anda dapat memulihkan cadangan. Setelah periode tujuh hari, Anda tidak bisa lagi memulihkan cadangan. Untuk informasi selengkapnya tentang mengelola cadangan, lihat Mengelola cadangan.

Hapus dan pulihkan cadangan (konsol)

Untuk menghapus cadangan (konsol)

1. Buka AWS CloudHSM konsol di https://console.aws.amazon.com/cloudhsm/home.
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, pilihTindakan, Hapus.

   Kotak dialog Hapus cadangan akan muncul.
6. Pilih Hapus.

   Keadaan kembali berubah menjadi `PENDING_DELETE`. Anda dapat memulihkan cadangan yaitu penghapusan tunda hingga 7 hari setelah Anda meminta penghapusan.

Untuk memulihkan cadangan (konsol)

1. Buka AWS CloudHSM konsol di https://console.aws.amazon.com/cloudhsm/home.
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.

Hapus dan pulihkan cadangan (CLI)

Periksa status cadangan atau temukan ID-nya dengan menggunakan describe-backups perintah dari CLI.

Untuk menghapus cadangan (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
    }
}
```

Untuk memulihkan cadangan (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"
    }
}
```

Untuk membuat daftar cadangan (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",
            "CreateTimestamp": 1534461854.64,
            "ClusterId": "cluster-dygnwhmscg5",
            "DeleteTimestap": 1536339805.522,
        }
}
```

Hapus dan pulihkan cadangan (API)AWS CloudHSM

Rujuk topik berikut untuk mempelajari cara menyalin cadangan ke berbagai wilayah dengan menggunakan API.

- DeleteBackup
- RestoreBackup

Mengkonfigurasi kebijakan retensi AWS CloudHSM cadangan

Dengan pengecualian cluster yang dibuat sebelum 18 November 2020, kebijakan retensi cadangan default untuk cluster adalah 90 hari. Anda dapat mengatur periode ini ke angka apa pun antara 7 dan 379 hari. AWS CloudHSM tidak menghapus cadangan terakhir cluster. Untuk informasi selengkapnya tentang mengelola cadangan, lihat Mengelola cadangan.

Memahami kebijakan retensi cadangan

AWS CloudHSM membersihkan cadangan berdasarkan kebijakan retensi cadangan yang Anda tetapkan saat membuat klaster. Kebijakan penyimpanan cadangan berlaku untuk klaster. Jika Anda memindahkan cadangan ke wilayah yang berbeda, cadangan tidak lagi terkait dengan klaster 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.

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 klaster 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 klaster.

Pengecualian klaster yang ada

AWS CloudHSM meluncurkan retensi cadangan terkelola pada 18 November 2020. Klaster yang dibuat sebelum 18 November 2020 mempunyai kebijakan penyimpaan cadangan 90 hari ditambah umur klaster. Misalnya, jika Anda membuat sebuah klaster pada 18 November 2019, layanan akan menetapkan untuk klaster Anda kebijakan penyimpanan cadangan satu tahun ditambah 90 hari (455 hari).

> **Note**
>
> Anda dapat berhenti dari penyimpanan cadangan terkelola sama sekali dengan menghubungi dukungan (https://aws.amazon.com/support).

Konfigurasikan retensi cadangan (konsol)

Untuk mengatur konfigurasi kebijakan retensi cadangan (konsol)

1. Buka AWS CloudHSM konsol di https://console.aws.amazon.com/cloudhsm/home.
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/home.
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.

Konfigurasikan retensi cadangan (CLI)

Periksa status cadangan atau temukan ID-nya dengan menggunakan describe-backups perintah dari CLI.

Untuk mengonfigurasi kebijakan retensi cadangan (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 (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 (CLI)

- Pada jendela 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
   }
}
```

Konfigurasikan retensi cadangan (AWS CloudHSM API)

Lihat topik berikut untuk mempelajari cara mengelola penyimpanan cadangan dengan menggunakan API.

- ModifyCluster
- ModifyBackupAttributes

Menyalin cadangan di seluruh Wilayah AWS

Anda dapat menyalin cadangan 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 Mengelola cadangan 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 Mengelola klaster.
- Cadangan tidak dapat disalin antara wilayah AWS terbatas dan wilayah standar. Cadangan dapat disalin antara wilayah AWS GovCloud (AS-Timur) dan AWS GovCloud (AS-Barat).

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/home.
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 (CLI)

Untuk menentukan ID cadangan, jalankan describe-backups perintah.

Untuk menyalin cadangan ke berbagai wilayah (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

Sumber daya penandaan AWS CloudHSM

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 konsol AWS CloudHSM atau salah satu SDK AWS atau alat baris perintah untuk menambahkan, memperbarui, membuat daftar, dan menghapus tanda.

Topik

- Menambahkan atau memperbarui tag
- Mencantumkan tanda
- Menghapus tanda

Menambahkan atau memperbarui tag

Anda dapat menambahkan atau memperbarui tag dari AWS CloudHSM konsol, AWS Command Line Interface (CLI), atau API. AWS CloudHSM

Untuk menambahkan atau memperbarui tanda (konsol)

1. Buka AWS CloudHSM konsol di https://console.aws.amazon.com/cloudhsm/home.
2. Pilih klaster yang Anda tandai.
3. PilihTanda.

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 menambah atau memperbarui tag (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.

Mencantumkan tanda

Anda dapat mencantumkan tag untuk klaster dari AWS CloudHSM konsol, CLI, atau API. AWS CloudHSM

Untuk membuat daftar tanda (konsol)

1. Buka AWS CloudHSM konsol di https://console.aws.amazon.com/cloudhsm/home.
2. Pilih klaster yang tandanya Anda cantumkan.
3. PilihTanda.

Untuk daftar tag (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.

Menghapus tanda

Anda dapat menghapus tag dari cluster dengan menggunakan AWS CloudHSM konsol, CLI, atau API. AWS CloudHSM

Untuk menghapus tanda (konsol)

1. Buka AWS CloudHSM konsol di https://console.aws.amazon.com/cloudhsm/home.
2. Pilih klaster yang tandanya Anda hapus.
3. PilihTanda.
4. Pilih Edit Tanda lalu pilih Hapus tanda untuk tanda yang ingin Anda hapus.
5. Pilih Simpan.

Untuk menghapus tag (CLI)

- Pada prompt perintah, keluarkan perintah untag-resource, menentukan kunci tanda dari tanda yang Anda hapus dan ID klaster yang tandanya Anda hapus. Saat Anda menggunakan 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 UntagResourcepermintaan di AWS CloudHSM API, tentukan ID cluster dan tag yang Anda hapus.

Mengelola pengguna HSM dan kunci diAWS CloudHSM

Sebelum Anda dapat menggunakan klaster AWS CloudHSM untuk pemrosesan kripto, Anda harus membuat pengguna dan kunci pada HSM di klaster Anda. Lihat topik berikut untuk informasi selengkapnya mengenai pengelolaan pengguna HSM dan kunci HSM di AWS CloudHSM. Anda juga dapat mempelajari cara menggunakan autentikasi kuorum (juga dikenal sebagai M dari N kontrol akses).

Topik

- Mengelola pengguna HSM di AWS CloudHSM
- Mengelola kunci di AWS CloudHSM
- Mengelola kloning kloning

Mengelola pengguna HSM di AWS CloudHSM

Di dalamnyaAWS 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.

Topik

- Mengelola pengguna HSM dengan CloudHSM CLI
- Mengelola pengguna HSM dengan CloudHSM Management Utility (CMU)

Mengelola pengguna HSM dengan CloudHSM CLI

Gunakan alat baris perintah CloudHSM CLI untuk membuat dan mengelola pengguna di HSM Anda dengan SDK terbaru.

Topik

- Memahami pengguna HSM
- Tabel izin pengguna HSM
- Menggunakan CloudHSM CLI untuk mengelola pengguna
- Menggunakan CloudHSM CLI untuk mengelola MFA

- Menggunakan CloudHSM CLI untuk mengelola autentikasi kuorum (M dari N kontrol akses)

Memahami pengguna HSM

Sebagian besar operasi yang dilakukan pada HSM memerlukan kredensial dari pengguna HSM. 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 kredensi yang benar dapat membuat HSM 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 kripto (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

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.

Pengguna alat (AU)

Pengguna alat (AU) dapat melakukan operasi kloning dan sinkronisasi pada HSM cluster Anda. AWS CloudHSM menggunakan AU untuk menyinkronkan HSM dalam sebuah AWS CloudHSM cluster. AU ada di semua HSM yang disediakan oleh AWS CloudHSM, dan memiliki izin terbatas. Lihat informasi yang lebih lengkap di Tabel izin pengguna HSM.

AWS tidak dapat melakukan operasi apa pun pada HSM Anda. 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

Tabel berikut berisi daftar operasi HSM diurutkan berdasarkan jenis pengguna HSM atau sesi yang dapat melakukan operasi.

	Admin	Pengguna Kripto (CU)	Pengguna Peralatan (AU)	Sesi Tidak Diautentikasi
Dapatkan informasi klaster dasar[1]	Y	Y	Y	Ya
Ubah kata sandi sendiri	Y	Y	Y	Tidak berlaku

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

- [1] Informasi klaster dasar mencakup jumlah HSM di klaster 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. Sebuah aplikasi dapat membandingkan set digest ini untuk memahami status sinkronisasi HSM dalam sebuah klaster.
- [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 klaster yang sama dengan HSM asalnya diekstraksi. Sebuah aplikasi dapat mengekstraksi dan memasukkan objek tertutup untuk menyinkronkan HSM dalam sebuah klaster.
- [4] Fungsi manajemen kunci termasuk membuat, menghapus, membungkus, membuka bungkusan, dan memodifikasi atribut kunci.

Menggunakan CloudHSM CLI untuk mengelola pengguna

Topik ini memberikan step-by-step instruksi tentang mengelola pengguna modul keamanan perangkat keras (HSM) dengan CloudHSM CLI. Untuk informasi selengkapnya tentang CloudHSM CLI atau pengguna HSM, lihat dan. CloudHSM CLI Menggunakan CloudHSM CLI

Bagian-bagian

- Memahami manajemen pengguna HSM dengan CloudHSM CLI
- Unduh CloudHSM CLI
- Cara mengelola pengguna HSM dengan CloudHSM CLI

Memahami manajemen pengguna HSM dengan CloudHSM CLI

Untuk mengelola pengguna HSM, Anda harus masuk ke HSM dengan nama pengguna dan kata sandi admin. Hanya admin yang dapat mengelola pengguna. HSM berisi admin default bernamaadmin. Anda mengatur kata sandi untuk admin ketika Anda mengaktifkan klaster.

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 CloudHSM Command Line Interface (CLI) Parameter `-a` mengharuskan Anda untuk menambahkan alamat IP HSM di klaster Anda. Jika Anda memiliki beberapa HSM, Anda dapat menggunakan alamat IP yang mana saja. 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 klaster 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/home.
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, pada tab HSM, pilih salah satu alamat IP yang tercantum di bawah alamat IP ENI.

Untuk mendapatkan alamat IP untuk HSM (CLI)

- Dapatkan alamat IP HSM dengan menggunakan describe-clusters perintah dari CLI. Dalam output dari perintah, alamat IP dari HSM adalah nilai-nilai dari `EniIp`.

```
$  aws cloudhsmv2 describe-clusters


{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
...
```

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 konfigurasikan CloudHSM CLI.

Cara mengelola pengguna HSM dengan CloudHSM CLI

Bagian ini mencakup perintah dasar untuk mengelola pengguna HSM dengan CloudHSM CLI.

> **Note**
>
> Catatan: Perintah pengguna CloudHSM CLI tercantum dalam referensi perintah pengguna CloudHSM CLI

Topik

Untuk membuat admin

Ikuti langkah-langkah ini untuk membuat admin.

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

   Linux

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

   Windows

   ```
   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": "admin",
    "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.

Untuk membuat pengguna kripto

Ikuti langkah-langkah ini untuk membuat pengguna.

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

   Linux

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

   Windows

   ```
   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": "admin",
    "role": "admin"
  }
}
```

4. Masukkan perintah berikut untuk membuat pengguna kripto:

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

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

Untuk daftar semua pengguna HSM di cluster

Gunakan user list perintah untuk daftar semua pengguna di cluster. Anda tidak perlu masuk untuk menjalankanuser 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

   ```
   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 selengkapnyauser list, lihat daftar pengguna.

Untuk mengubah kata sandi pengguna HSM

Gunakan user change-password perintah untuk mengubah kata sandi.

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

Admin, pengguna kripto (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

   ```
   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": "admin1",
       "role": "admin"
     }
   }
   ```

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

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

2. Gunakan login perintah dan masuk sebagai admin.

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

3. Masukkan kata sandi admin.

   ```
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "admin1",
       "role": "admin"
     }
   }
   ```

4. 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>
   ```

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

Untuk informasi selengkapnyauser change-password, lihat perubahan kata sandi pengguna.

Untuk menghapus pengguna HSM

Gunakan user delete untuk menghapus pengguna. Anda harus masuk sebagai admin untuk menghapus pengguna lain.

> **Tip**
>
> Anda tidak dapat menghapus pengguna kripto (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

   ```
   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": "admin",
       "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.

Menggunakan CloudHSM CLI untuk mengelola MFA

Untuk meningkatkan keamanan, Anda dapat mengonfigurasi otentikasi multi-faktor (MFA) untuk membantu melindungi klaster. Untuk informasi selengkapnya, lihat topik di bawah.

Topik

Memahami MFA untuk pengguna HSM

Saat Anda masuk ke klaster dengan akun pengguna 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 lebih lanjut tentang pengaturan MFA untuk pengguna, silakan lihat Mengatur MFA untuk CloudHSM CLI

Kuorum otentikasi dan MFA

Cluster menggunakan kunci yang sama untuk otentikasi kuorum dan untuk MFA. Ini berarti pengguna dengan MFA diaktifkan secara efektif terdaftar untuk kontrol akses MoFN atau Quroum. 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 Kuroum (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 kuorum (M dari N).

Bekerja dengan MFA untuk pengguna HSM

Topik ini memberikan informasi dan instruksi untuk menggunakan CloudHSM CLI untuk mengelola Otentikasi Multifaktor (MFA). Untuk informasi selengkapnya tentang CloudHSM CLI, lihat. Antarmuka Baris Perintah CloudHSM (CLI)

Topik

- Persyaratan Pasangan Kunci MFA
- Mengatur MFA untuk CloudHSM CLI
- Buat pengguna dengan MFA diaktifkan
- Masuk pengguna dengan MFA diaktifkan
- Putar tombol untuk pengguna dengan MFA diaktifkan
- Membatalkan pendaftaran kunci publik MFA untuk pengguna admin saat kunci publik MFA terdaftar
- Referensi file token

Untuk informasi lebih lanjut tentang bekerja dengan pengguna HSM, lihat Antarmuka Baris Perintah CloudHSM (CLI)

Persyaratan Pasangan Kunci MFA

Untuk mengaktifkan MFA bagi pengguna HSM, 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: SHA256withRSAEncryption

> **Note**
>
> Jika Anda menggunakan otentikasi kuorum atau berencana menggunakan otentikasi kuorum, lihat Kuorum otentikasi dan MFA

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 mengatur 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. Menggunakan CloudHSM CLI, masuk ke akun pengguna Anda.

```
$ cloudhsm-cli interactive
aws-cloudhsm > login --username admin --role admin --cluster-id <cluster ID>
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin",
    "role": "admin"
  }
}
```

3. 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:
```

4. Anda sekarang memiliki file dengan token yang tidak ditandatangani yang perlu ditandatangani:`unsigned-tokens.json`. Jumlah token dalam file ini tergantung pada jumlah HSM 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/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=",
      "signed": ""
    },
    {
      "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
      "signed": ""
    },
    {
      "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
      "signed": ""
    }
  ]
}
```

5. 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/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=" > 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
```

6. 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
```

7. Sekarang, Anda memiliki tanda tangan biner dari token. Anda harus menyandikannya menggunakan base64, dan 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
```

8. Terakhir, Anda dapat menyalin dan menempelkan nilai base64 kembali ke file token Anda:

```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwxb9bJOUUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2tlF9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGBOj4yB9recpOBB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
```

```
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASPnvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+HjmrO53OQpZaJ/VXGIgApD0myuu/
ZGNKQTCSkkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAxORTLlmwyUOYvPYOvUhc
+s83hx36QpGwGcD7RA0bPT5OrTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93SbOUyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnswduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkYlftr5EOjqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCiOYDzUCd43GftGq2LfxH3qSD51oFHglHQVOY0jyVzzlAvub5HQdtOQ
    }
  ]
}
```


9. 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

Ikuti langkah-langkah ini untuk membuat pengguna dengan MFA diaktifkan.

1. Gunakan CloudHSM CLI untuk masuk ke HSM sebagai admin.
2. Gunakan user createperintah untuk membuat pengguna pilihan Anda. Kemudian ikuti langkah-langkah Mengatur MFA untuk CloudHSM CLI untuk mengatur MFA untuk pengguna.

Masuk pengguna dengan MFA diaktifkan

Ikuti langkah-langkah ini untuk login pengguna dengan MFA diaktifkan.

1. Gunakan login mfa-token-signperintah 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 yang tidak ditandatangani, di mana 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 HSM 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/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=",
      "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/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=" > 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": "1jqwxb9bJOUUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2tlF9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGBOj4yB9recpOBB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASPnvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+HjmrO53OQpZaJ/VXGIgApD0myuu/
ZGNKQTCSkkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAxORTLlmwyUOYvPYOvUhc
+s83hx36QpGwGcD7RA0bPT5OrTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
```

```
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93SbOUyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnswduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkYlftr5EOjqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCiOYDzUCd43GftGq2LfxH3qSD51oFHglHQVOY0jyVzzlAvub5HQdtOQ
    }
  ]
}
```

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

Ikuti langkah-langkah ini untuk memutar tombol untuk pengguna dengan 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 HSM 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 RSA public/private key pair baru yang ingin Anda gunakan untuk memutar kunci publik yang saat ini terdaftar.

```
$cat unsigned-tokens.json
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=",
      "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/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=" > 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": "1jqwxb9bJOUUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2tlF9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
```

```
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGBOj4yB9recpOBB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASPnvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+HjmrO53OQpZaJ/VXGIgApD0myuu/
ZGNKQTCSkkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAxORTLlmwyUOYvPYOvUhc
+s83hx36QpGwGcD7RA0bPT5OrTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93SbOUyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnswduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkYlftr5EOjqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCiOYDzUCd43GftGq2LfxH3qSD51oFHglHQVOY0jyVzzlAvub5HQdtOQ
    }
  ]
}
```


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 untuk pengguna admin saat kunci publik MFA terdaftar

Ikuti langkah-langkah ini untuk membatalkan pendaftaran kunci publik MFA untuk 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

File token yang dihasilkan saat mendaftarkan kunci publik MFA atau ketika mencoba masuk menggunakan MFA terdiri dari yang berikut:

- Token: Sebuah array base64 menyandikan pasangan token yang tidak ditandatangani atau ditandatangani dalam bentuk literal objek JSON.
- Tidak ditandatangani: Token yang dikodekan base64 dan hashed 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": "1jqwxb9bJOUUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2tlF9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/TK0PVaxLN42X
+pebV4juwMhN4mK4CzdFAJgM+UGBOj4yB9recpOBB9K8QFSpJZALSEdDgUc/mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37+j/
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASPnvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+HjmrO53OQpZaJ/VXGIgApD0myuu/
ZGNKQTCSkkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAxORTLlmwyUOYvPYOvUhc
+s83hx36QpGwGcD7RA0bPT5OrTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93SbOUyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnswduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkYlftr5EOjqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCiOYDzUCd43GftGq2LfxH3qSD51oFHglHQVOY0jyVzzlAvub5HQdtOQdErI
    }
  ]
}
```


Menggunakan CloudHSM CLI untuk mengelola autentikasi kuorum (M dari N kontrol akses)

HSM di klaster AWS CloudHSM mendukung autentikasi kuorum, yang juga dikenal sebagai M dari N kontrol akses. 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 admin — Membuat dan menghapus pengguna HSM, dan mengubah kata sandi pengguna HSM yang berbeda. Untuk informasi selengkapnya, lihat Menggunakan autentikasi kuorum untuk admin.

Topik berikut ini menyediakan informasi lebih lanjut tentang autentikasi kuorum di AWS CloudHSM.

Topik

Ikhtisar otentikasi kuorum dengan strategi tanda token

Langkah-langkah berikut merangkum proses autentikasi kuorum. Untuk langkah-langkah dan alat tertentu, lihat Menggunakan autentikasi kuorum untuk admin.

1. Setiap pengguna HSM membuat kunci asimetris untuk penandatanganan. Pengguna melakukan ini di luar HSM, mengurus untuk melindungi kunci dengan tepat.
2. Setiap pengguna HSM masuk ke HSM dan mendaftarkan bagian publik kunci penandatanganannya (kunci publik) dengan HSM.
3. Ketika pengguna HSM ingin melakukan operasi terkontrol 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 terkontrol kuorum dengan--approval argumen, memasok

file token kuorum yang ditandatangani, yang berisi semua persetujuan yang diperlukan (tanda tangan).

7. HSM menggunakan kunci publik terdaftar dari setiap penandatangan untuk memverifikasi tanda tangan. Jika tanda tangan valid, HSM menyetujui token dan operasi terkontrol kuorum dilakukan.

Detail tambahan tentang otentikasi kuorum

Perhatikan informasi tambahan berikut tentang menggunakan autentikasi kuorum di AWS CloudHSM.

- Pengguna HSM dapat menandatangani token kuorum miliknya sendiri—yaitu, pengguna yang meminta dapat memberikan salah satu persetujuan yang diperlukan untuk autentikasi kuorum.
- Anda memilih jumlah minimum pemberi persetujuan kuorum untuk operasi yang dikontrol kuorum. Jumlah terkecil yang dapat Anda pilih adalah dua (2), dan jumlah 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.
- Jika MFA diaktifkan, klaster menggunakan kunci yang sama untuk autentikasi kuorum dan autentikasi multi-faktor (MFA). Untuk informasi selengkapnya tentang penggunaan autentikasi kuorum dan 2FA, lihat Menggunakan CloudHSM untuk mengelola MFA.
- Setiap HSM hanya dapat berisi satu token per layanan pada satu waktu.

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.

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
pengguna	Admin	• pengguna membuat • menghapus pengguna

Nama layanan	Jenis layanan	Operasi layanan
		• perubahan kata sandi pengguna • perubahan pengguna-mfa
kuorum	Admin	• tanda token kuorum set-quorum-value

Menggunakan otentikasi kuorum untuk admin: penyiapan pertama kali

Topik berikut menjelaskan langkah-langkah yang harus Anda selesaikan untuk mengonfigurasi modul keamanan perangkat keras (HSM) Anda sehingga 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 Menggunakan autentikasi kuorum untuk admin.

Topik

- Prasyarat
- Buat dan daftarkan kunci untuk penandatanganan
- Tetapkan nilai minimum kuorum pada HSM

Prasyarat

Untuk memahami contoh ini, Anda harus terbiasa dengan CloudHSM CLI. Dalam contoh ini, AWS CloudHSM cluster memiliki dua HSM, masing-masing dengan admin yang sama, seperti yang ditunjukkan pada output berikut dari perintah. user list Untuk informasi selengkapnya tentang membuat pengguna, lihat Menggunakan CloudHSM CLI.

```
aws-cloudhsm>user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
```

```
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "admin2",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "admin3",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "admin4",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

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 yang dilindungi oleh frase sandi. Untuk menggunakan contoh ini, ganti <admin.key>dengan nama file tempat Anda ingin menyimpan kunci.

```
$ openssl genrsa -out <admin.key> -aes256 2048
Generating RSA private key, 2048 bit long modulus
.....................................+++
.+++
e is 65537 (0x10001)
Enter pass phrase for admin.key:
Verifying - Enter pass phrase for admin.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 admin.key -outform PEM -pubout -out admin1.pub
Enter pass phrase for admin.key:
writing RSA key
```

Membuat dan menandatangani token pendaftaran

Anda membuat token dan menandatanganinya 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

   ```
   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 di-hash SHA256 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 kuorum 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 '6BMUj6mUjjko6ZLCEdzGlWpR5sILhFJfqhW1ej3Oq1g=' | 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

   ```
   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 daftar tanda token ubah-kuorum pengguna 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: Filepath 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 kuorum, yang menyatakan strategi kuorum 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"
      }
    ]
  }
}
```

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 lebih lanjut, 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

   ```
   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 tanda token kuorum set-quorum-value untuk menetapkan nilai minimum kuorum. Untuk informasi selengkapnya, lihat contoh berikut atau gunakan perintah help quorum token-sign set-quorum-value.

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, layanan mengidentifikasi layanan HSM yang nilai minimum kuorumnya Anda tetapkan. tanda token kuorum list-quorum-valuesPerintah mencantumkan jenis layanan HSM, nama, dan deskripsi yang disertakan dalam layanan.

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.

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
pengguna	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

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 Menggunakan kuorum (M dari N).

Menggunakan autentikasi kuorum untuk admin

Admin pada HSM dapat mengkonfigurasi otentikasi kuorum untuk operasi berikut dalam klaster: AWS CloudHSM

- pengguna membuat
- menghapus pengguna
- perubahan kata sandi pengguna
- perubahan pengguna-mfa

Setelah AWS CloudHSM klaster dikonfigurasi untuk otentikasi kuorum, admin tidak dapat melakukan operasi manajemen pengguna HSM sendiri. Contoh berikut menunjukkan output ketika admin mencoba untuk membuat pengguna baru pada HSM. Perintah gagal dengan kesalahan, 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

- Dapatkan token kuorum
- Mendapatkan tanda tangan dari menyetujui admin
- Menyetujui token pada AWS CloudHSM cluster dan menjalankan operasi manajemen pengguna

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

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

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

   ```
   aws-cloudhsm>login --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 - Menghasilkan token kuorum

Contoh ini mendapat 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 menjadi admin yang sama yang login ke HSM dan menjalankan perintah ini.
- **<admin.token>**- Nama file yang akan digunakan untuk menyimpan token kuorum.

Dalam perintah berikut, `user` mengidentifikasi nama layanan yang dapat Anda gunakan token yang Anda hasilkan. Dalam hal ini, token adalah untuk operasi manajemen pengguna HSM (`user`layanan).

```
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 layanan pengguna kuorum token di path file yang ditentukan. File token dapat diperiksa:

```
$cat </path/admin.token>
{
  "version": "2.0",
  "approval_data": "AAEAAwAAABgAAAAAAAAAAJ9eFkfcP3mNzJAlfK
+OWbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAAQADAAAAFQAAAAAAAAAW/
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAAABAAMAAAAUA
+b23gAAAAAAAAA",
```

```
  "token": "0l2LZkmAHZyAc1hPhyckOoVW33aGrgG77qmDHWQ3CJ8=",
  "signatures": []
}
```

File token terdiri dari yang berikut:

- approval_data: Token data mentah yang dikodekan base64 yang dihasilkan oleh HSM.
- token: Sebuah base64 dikodekan dan SHA-256 hash token approval_data
- tanda tangan: Array token bertanda tangan yang dikodekan base64 (tanda tangan) dari token yang tidak ditandatangani, di mana setiap tanda tangan pemberi persetujuan dalam bentuk objek JSON literal:

  ```
  {
        "username": "<APPROVER_USERNAME>",
        "signature": "<APPROVER_RSA2048_BIT_SIGNATURE>"
  }
  ```

  Setiap tanda tangan dibuat dari hasil pemberi persetujuan menggunakan kunci privat RSA 2048-bit yang sesuai yang kunci publiknya terdaftar di HSM..

Token kuorum layanan pengguna yang dihasilkan dapat dikonfirmasi ada di cluster 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.

Mendapatkan tanda tangan dari menyetujui admin

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 SHA-256 digest token.

Example - Dapatkan tanda tangan dari menyetujui admin

Dalam contoh ini, admin yang memiliki token (`admin`) memerlukan setidaknya dua (2) persetujuan. Contoh perintah berikut menunjukkan bagaimana dua (2) admin dapat menggunakan OpenSSL untuk menandatangani token secara kriptografis.

1. Dekode token unsigned yang dikodekan base64 dan tempatkan ke dalam file biner:

   ```
   $echo -n '0l2LZkmAHZyAc1hPhyckOoVW33aGrgG77qmDHWQ3CJ8=' | base64 -d > admin.bin
   ```

2. Gunakan OpenSSL dan kunci pribadi masing-masing pemberi persetujuan (`admin3`) untuk menandatangani token unsigned kuorum biner sekarang 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. Encode 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": "AAEAAwAAABgAAAAAAAAAAJ9eFkfcP3mNzJAlfK
+OWbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAAQADAAAAFQAAAAAAAAAW
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAAABAAMAA
+b23gAAAAAAAAAA",
  "token": "0l2LZkmAHZyAc1hPhyckOoVW33aGrgG77qmDHWQ3CJ8=",
  "signatures": [
    {
      "username": "admin2",
      "signature": "O6qx7/mUaVkYYVr1PW7l8JJko+Kh3e8zBIqdk3tAiNy+1rW
+OsDtvYujhEU4aOFVLcrUFmyB/CX9OQmgJLgx/pyK+ZPEH+GoJGqk9YZ7X1nOXwZRP9g7hKV
+7XCtg9TuDFtHYWDpBfz2jWiu2fXfX4/
jTs4f2xIfFPIDKcSP8fhxjQ63xEcCf1jzGha6rDQMu4xUWWdtDgfT7um7EJ9dXNoHqLB7cTzphaubNaEFbFPXQ1siGm
ssktwyruGFLpXs1n0tJOEglGhx2qbYTs+omKWZdORl5WIWEXW3IXw/
Dg5vVObrNpvG0eZKO8nSMc27+cyPySc+ZbNw=="
    },
    {
      "username": "admin3",
      "signature": "O6qx7/mUaVkYYVr1PW7l8JJko+Kh3e8zBIqdk3tAiNy+1rW
+OsDtvYujhEU4aOFVLcrUFmyB/CX9OQmgJLgx/pyK+ZPEH+GoJGqk9YZ7X1nOXwZRP9g7hKV
+7XCtg9TuDFtHYWDpBfz2jWiu2fXfX4/
jTs4f2xIfFPIDKcSP8fhxjQ63xEcCf1jzGha6rDQMu4xUWWdtDgfT7um7EJ9dXNoHqLB7cTzphaubNaEFbFPXQ1siGm
ssktwyruGFLpXs1n0tJOEglGhx2qbYTs+omKWZdORl5WIWEXW3IXw/
Dg5vVObrNpvG0eZKO8nSMc27+cyPySc+ZbNw=="
    }
  ]
}
```

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 klaster bersama dengan salah satu operasi manajemen pengguna berikut:

- buat

- hapus
- ubah-kata sandi
- user change-mfa

Untuk informasi selengkapnya tentang menggunakan perintah ini, lihat Menggunakan 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 menghasilkan token kuorum baru, mendapatkan tanda tangan baru dari pemberi persetujuan, dan kemudian menyetujui dan mengkonsumsi token baru pada 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. Hal ini tidak dapat digunakan untuk mengotentikasi dalam aplikasi yang berbeda.

Example Membuat pengguna baru sebagai admin

Pada contoh berikut, admin login 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 mengkonfirmasi 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 untuk 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 menghasilkan token kuorum baru, mendapatkan tanda tangan baru dari pemberi persetujuan, dan menjalankan operasi manajemen pengguna yang diinginkan dengan argumen - persetujuan untuk memasok token kuorum untuk disetujui dan dikonsumsi selama pelaksanaan operasi manajemen pengguna.

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": []
  }
}
```

Ubah nilai minimum kuorum untuk admin

Setelah Anda tetapkan nilai minimum kuorum sehingga admin dapat menggunakan autentikasi kuorum, Anda mungkin perlu 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. Sebagai contoh, jika nilai minimum kuorum adalah dua (2), setidaknya dua (2) admin harus menyetujui untuk mengubah nilai minimum kuorum.

> **Note**
>
> Nilai kuorum layanan pengguna harus selalu kurang dari nilai kuorum layanan kuorum. Untuk informasi tentang nama layanan, seperti layanan kuorum dan layanan pengguna, lihat. Nama dan jenis layanan yang mendukung otentikasi kuorum

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 quorum token-sign set-quorum-value perintah quorum service penggunaan, 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.

Ubah nilai minimum kuorum untuk admin

1. Gunakan perintah berikut untuk memulai mode interaktif CloudHSM.

   Linux

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

   Windows

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

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

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

3. Gunakan quorum token-sign list-quorum-values perintah untuk mendapatkan nilai minimum kuorum untuk semua nama layanan. Untuk informasi lebih lanjut, lihat contoh di bawah ini.
4. Jika nilai minimum kuorum untuk layanan kuorum lebih rendah dari nilai untuk layanan pengguna, gunakan quorum token-sign set-quorum-value perintah untuk mengubah nilai untuk layanan kuorum. Ubah nilai untuk layanan kuorum menjadi satu (1) yang sama atau lebih tinggi dari nilai untuk layanan pengguna. Untuk informasi selengkapnya, lihat contoh berikut.
5. Buat token kuorum, berhati-hatilah untuk menentukan layanan kuorum sebagai layanan yang dapat Anda gunakan dengan token.
6. Dapatkan persetujuan (tanda tangan) dari admin lainnya.
7. Menyetujui token pada AWS CloudHSM cluster dan menjalankan operasi manajemen pengguna. .
8. Gunakan quorum token-sign set-quorum-value perintah untuk mengubah nilai minimum kuorum untuk layanan pengguna.

Example — Dapatkan nilai minimum kuorum dan ubah nilai untuk layanan kuorum

Contoh perintah berikut 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
  }
}
```

Untuk mengubah nilai minimum kuorum untuk layanan kuorum, gunakan quorum token-sign set-quorum-value perintah, menetapkan nilai yang sama atau lebih tinggi dari nilai untuk layanan pengguna. Contoh berikut menetapkan nilai minimum kuorum untuk layanan kuorum untuk dua (2), nilai yang sama yang ditetapkan untuk layanan pengguna.

```
aws-cloudhsm > quorum token-sign set-quorum-value --service quorum --value 2{
  "error_code": 0,
  "data": "Set quorum value successful"
}
```

Perintah berikut menunjukkan bahwa nilai minimum kuorum sekarang adalah dua (2) untuk layanan pengguna dan layanan kuorum.

```
aws-cloudhsm > quorum token-sign list-quorum-values{
  "error_code": 0,
  "data": {
    "user": 2,
    "quorum": 2
  }
}
```

Mengelola pengguna HSM dengan CloudHSM Management Utility (CMU)

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 SDK terbaru, sedangkan CMU dirancang untuk digunakan dengan SDK sebelumnya.

Topik

- Memahami pengguna HSM
- Tabel izin pengguna HSM
- Menggunakan CloudHSM Management Utility (CMU) untuk mengelola pengguna
- Menggunakan CloudHSM Management Utility (CMU) untuk mengelola otentikasi dua faktor (2FA) untuk petugas kripto

- Menggunakan CMU Manajemen CMU CMU untuk mengelola autentikasi kuorum autentikasi kuorum kontrol akses M dari N kontrol kuorum untuk mengelola autentikasi kuorum autentikasi kuorum

Memahami pengguna HSM

Sebagian besar operasi yang dilakukan pada HSM memerlukan kredensial dari pengguna HSM. 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 kredensil yang benar dapat membuat HSM 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 cluster activate perintah. 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 crypto (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. 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.

Pengguna alat (AU)

Pengguna alat (AU) dapat melakukan operasi kloning dan sinkronisasi pada HSM cluster Anda. AWS CloudHSM menggunakan AU untuk menyinkronkan HSM dalam sebuah AWS CloudHSM cluster. AU ada di semua HSM yang disediakan oleh AWS CloudHSM, dan memiliki izin terbatas. Lihat informasi yang lebih lengkap di Tabel izin pengguna HSM.

AWS tidak dapat melakukan operasi apa pun pada HSM Anda. 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

Tabel berikut berisi daftar operasi HSM diurutkan berdasarkan jenis pengguna HSM atau sesi yang dapat melakukan operasi.

	Petugas kripto (CO)	Pengguna Kripto (CU)	Pengguna Peralatan (AU)	Sesi Tidak Diautentikasi
Dapatkan informasi klaster dasar[1]	Y	Y	Y	Ya

	Petugas kripto (CO)	Pengguna Kripto (CU)	Pengguna Peralatan (AU)	Sesi Tidak Diautentikasi
Ubah kata sandi sendiri	 Y	 Y	 Y	Tidak berlaku
Ubah kata sandi pengguna	 Y	 T	 T	 Tidak
Tambah, hapus pengguna	 Y	 T	 T	 Tidak
Dapatkan status sinkronis[2]	 Y	 Y	 Y	 Tidak
Ekstrak, masukkan benda bertopeng [3]	 Y	 Y	 Y	 Tidak
Fungsi manajemen kunci	 T	 Y	 T	 Tidak
Enkripsi, dekripsi	 T	 Y	 T	 Tidak
Tanda tangan, verifikasi	 T	 Y	 T	 Tidak

	Petugas kripto (CO)	Pengguna Kripto (CU)	Pengguna Peralatan (AU)	Sesi Tidak Diautentikasi
Hasilkan digests dan HMAC	 T	 Y	 T	 Tidak

- [1] Informasi klaster dasar mencakup jumlah HSM di klaster 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. Sebuah aplikasi dapat membandingkan set digest ini untuk memahami status sinkronisasi HSM dalam sebuah klaster.
- [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 klaster yang sama dengan HSM asalnya diekstraksi. Sebuah aplikasi dapat mengekstraksi dan memasukkan objek tertutup untuk menyinkronkan HSM dalam sebuah klaster.
- [4] Fungsi manajemen kunci termasuk membuat, menghapus, membungkus, membuka bungkusan, dan memodifikasi atribut kunci.

Menggunakan CloudHSM Management Utility (CMU) untuk mengelola pengguna

Topik ini memberikan step-by-step instruksi tentang mengelola pengguna modul keamanan perangkat keras (HSM) dengan CloudHSM Management Utility (CMU), alat baris perintah yang disertakan dengan Client SDK. Untuk informasi selengkapnya tentang CMU atau pengguna HSM, lihat Utilitas Manajemen CloudHSM dan Memahami pengguna HSM.

Bagian

- Memahami manajemen pengguna HSM dengan CMU
- Unduh Utilitas Manajemen CloudHSM
- Cara mengelola pengguna HSM dengan CMU

Memahami manajemen pengguna HSM dengan CMU

Untuk mengelola pengguna HSM, Anda harus login ke HSM dengan nama pengguna dan kata sandi petugas kriptografi (CO). Hanya CO yang dapat mengelola pengguna. HSM berisi CO default bernama admin. Anda mengatur kata sandi untuk admin ketika Anda mengaktifkan klaster.

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 HSM, Anda dapat menggunakan alamat IP yang mana saja. 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/home.
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, pada tab HSM, pilih salah satu alamat IP yang tercantum di bawah alamat IP ENI.

Untuk mendapatkan alamat IP untuk HSM (CLI)

- Dapatkan alamat IP HSM dengan menggunakan describe-clusters perintah dari CLI. Dalam output dari perintah, alamat IP dari HSM adalah nilai-nilai dari `EniIp`.

```
$  aws cloudhsmv2 describe-clusters
```

```
{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
...
```

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 (AWSCloudHSMManagementUtil-latest.msi) dengan hak administratif Windows.

Windows Server 2012 R2

1. Unduh Utilitas Manajemen CloudHSM
2. Jalankan penginstal CMU (AWSCloudHSMManagementUtil-latest.msi) dengan hak administratif Windows.

Windows Server 2016

1. Unduh Utilitas Manajemen CloudHSM
2. Jalankan penginstal CMU (AWSCloudHSMManagementUtil-latest.msi) dengan hak administratif Windows.

Cara mengelola pengguna HSM dengan CMU

Bagian ini mencakup perintah dasar untuk mengelola pengguna HSM dengan CMU.

Untuk membuat pengguna HSM

Gunakan createUser untuk membuat pengguna baru di 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

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
   ```

2. Mulai CMU.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   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 daftar CMU koneksi cocok dengan jumlah HSM di klaster. Jika tidak, keluar dan mulai dari awal.

4. 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)?
```

5. Ketik **y**.

Untuk membuat pengguna baru

1. Gunakan alat konfigurasi untuk memperbarui konfigurasi CMU.

   Linux

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

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
   ```

2. Mulai CMU.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   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 daftar CMU koneksi cocok dengan jumlah HSM di klaster. 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.

Untuk daftar semua pengguna HSM di klaster

Gunakan perintah listUsers untuk daftar semua pengguna di klaster. Anda tidak perlu masuk untuk menjalankan listUsers dan semua jenis pengguna dapat mendaftar 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

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
   ```

2. Mulai CMU.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   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                                NO
                  0               NO
            2              CO              example_officer                         NO
                  0               NO
            3              CU              example_user                            NO
                  0               NO
   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                                NO
              0               NO
        2               CO              example_officer                         NO
              0               NO
        3               CU              example_user                            NO
              0               NO
```

Untuk informasi lebih lanjut tentang listUsers, lihat listUsers.

Untuk mengubah kata sandi pengguna HSM

Gunakan changePswd untuk mengubah kata sandi.

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

CO, Pengguna kripto (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

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
   ```

2. Mulai CMU.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

Windows

```
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 CO admin co12345
```

Pastikan jumlah daftar CMU koneksi cocok dengan jumlah HSM di klaster. Jika tidak, keluar dan mulai dari awal.

4. Gunakan changePswd untuk mengubah kata sandi Anda sendiri.

```
aws-cloudhsm>changePswd CO 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(CO) 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

```
C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
```

2. Mulai CMU.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   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 daftar CMU koneksi cocok dengan jumlah HSM di klaster. 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.

Untuk menghapus pengguna HSM

Gunakan deleteUser untuk menghapus pengguna. Anda harus masuk sebagai CO untuk menghapus pengguna lain.

> **Tip**
>
> Anda tidak dapat menghapus pengguna kripto (CU) yang memiliki kunci.

Untuk menghapus klaster

1. Gunakan alat konfigurasi untuk memperbarui konfigurasi CMU.

   Linux

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

   Windows

```
C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
```

2. Mulai CMU.

   Linux

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

Windows

```
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 daftar CMU koneksi cocok dengan jumlah HSM di klaster. Jika tidak, keluar dan mulai dari awal.

4. Gunakan deleteUser untuk menghapus pengguna.

```
aws-cloudhsm>deleteUser CO example_officer
```

CMU menghapus pengguna.

```
Deleting user example_officer(CO) 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.

Menggunakan CloudHSM Management Utility (CMU) untuk mengelola otentikasi dua faktor (2FA) untuk petugas kripto

Untuk peningkatan keamanan, Anda dapat mengatur konfigurasi Dua Factor Authentication (2FA) untuk membantu melindungi klaster. Anda hanya dapat mengaktifkan 2FA untuk petugas kripto (CO).

> **Note**
>
> Anda tidak dapat mengaktifkan 2FA untuk pengguna kripto (CU) atau aplikasi. Autentikasi dua faktor (2FA) hanya untuk pengguna CO.

Topik

- Memahami 2FA untuk pengguna HSM
- Bekerja dengan 2FA untuk pengguna HSM

Memahami 2FA untuk pengguna HSM

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.

Otentikasi kuorum dan 2FA

Klaster menggunakan kunci yang sama untuk autentikasi kuorum dan 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 autentikasi kuorum autentikasi kuorum.

Bekerja dengan 2FA untuk pengguna HSM

Bagian ini menjelaskan cara bekerja dengan 2FA untuk pengguna HSM, termasuk membuat pengguna HSM 2FA, memutar kunci, dan masuk ke HSM sebagai pengguna dengan 2FA diaktifkan. Untuk informasi lebih lanjut tentang bekerja dengan pengguna HSM, lihat ???, ???, ???, ???, dan ???.

Membuat pengguna 2FA

Untuk mengaktifkan 2FA untuk pengguna HSM, gunakan kunci yang memenuhi persyaratan berikut.

Persyaratan key pair 2FA

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 dan 2FA”.

Gunakan CMU dan key pair untuk membuat pengguna CO baru dengan 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 NAME> -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)?

yCreating 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 setelahBEGIN 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 menandatangani.

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": "Kkdl ... 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 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: >>>>> Press Enter here

createUser success on server 0(10.0.1.11)
```

Mengelola 2FA untuk pengguna HSM

Gunakan perubahan kata sandi untuk mengubah kata sandi untuk pengguna 2FA, atau untuk mengaktifkan atau menonaktifkan 2FA, atau untuk memutar kunci 2FA. Setiap kali Anda mengaktifkan 2FA, Anda harus menyediakan kunci publik untuk login 2FA.

Mengubah kata sandi akan 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 klaster.

   ```
   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.
3. 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 dan 2FA”.

Untuk menonaktifkan 2FA untuk pengguna CO dengan 2FA diaktifkan

1. Gunakan CMU untuk masuk ke HSM sebagai CO dengan 2FA diaktifkan.
2. 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 dan 2FA”.

3. Ketik **y**.

   CMU mengonfirmasi operasi perubahan kata sandi.

Referensi konfigurasi

Berikut ini adalah contoh dari properti 2FA di file `authdata` untuk kedua permintaan yang dihasilkan CMU dan tanggapan Anda.

```
{
    "Version": "1.0",
    "PublicKey": "-----BEGIN PUBLIC KEY----- ... -----END PUBLIC KEY-----",
    "Data": [
        {
            "HsmId": "hsm-lgavqitns2a",
            "Digest": "k5O1p3f6foQRVQH7S8Rrjcau6h3TYqsSdr16A54+qG8=",
            "Signature": "Kkdl ... 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 CMU Manajemen CMU CMU untuk mengelola autentikasi kuorum autentikasi kuorum kontrol akses M dari N kontrol kuorum untuk mengelola autentikasi kuorum autentikasi kuorum

HSM di klaster AWS CloudHSM mendukung autentikasi kuorum, yang juga dikenal sebagai M dari N kontrol akses. 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 kripto (CO)— Membuat dan menghapus pengguna HSM, dan mengubah kata sandi pengguna HSM yang berbeda. Untuk informasi lebih lanjut, lihat Menggunakan otentikasi kuorum untuk petugas kripto.

Topik berikut ini menyediakan informasi lebih lanjut tentang autentikasi kuorum di AWS CloudHSM.

Topik

- Gambaran umum autentikasi kuorum
- Detail tambahan tentang otentikasi kuorum
- Menggunakan otentikasi kuorum untuk petugas kripto: pengaturan pertama kali
- Menggunakan otentikasi kuorum untuk petugas kripto
- Ubah Nilai Minimum Kuorum untuk petugas kripto

Gambaran umum autentikasi kuorum

Langkah-langkah berikut merangkum proses autentikasi kuorum. Untuk langkah-langkah dan alat tertentu, lihat Menggunakan otentikasi kuorum untuk petugas kripto.

1. Setiap pengguna HSM membuat kunci asimetris untuk penandatanganan. Pengguna melakukan ini di luar HSM, mengurus untuk melindungi kunci dengan tepat.
2. Setiap pengguna HSM masuk ke HSM dan mendaftarkan bagian publik kunci penandatanganannya (kunci publik) dengan HSM.
3. Ketika pengguna HSM ingin melakukan operasi terkontrol 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) untuk HSM.
7. HSM menggunakan kunci publik terdaftar dari setiap penandatangan untuk memverifikasi tanda tangan. Jika tanda tangan valid, HSM menyetujui token.
8. Pengguna HSM sekarang dapat melakukan operasi dikendalikan kuorum.

Detail tambahan tentang otentikasi kuorum

Perhatikan informasi tambahan berikut tentang menggunakan autentikasi kuorum di AWS CloudHSM.

- Pengguna HSM dapat menandatangani token kuorum miliknya sendiri—yaitu, pengguna yang meminta dapat memberikan salah satu persetujuan yang diperlukan untuk autentikasi kuorum autentikasi kuorum.
- Anda memilih jumlah minimum pemberi persetujuan kuorum untuk operasi yang dikontrol kuorum. Jumlah terkecil yang dapat Anda pilih adalah dua (2), dan jumlah terbesar yang dapat Anda pilih adalah delapan (8), dan jumlah terbesar yang dapat Anda pilih adalah delapan (8), dan jumlah terbesar yang dapat Anda pilih adalah delapan (8), jumlah terbesar
- 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.

Menggunakan otentikasi kuorum untuk petugas kripto: pengaturan pertama kali

Topik berikut menjelaskan langkah-langkah yang harus Anda selesaikan untuk mengatur konfigurasi modul keamanan perangkat keras (HSM), sehingga petugas kripto (CO)dapat menggunakan autentikasi kuorum. Anda perlu melakukan langkah-langkah ini hanya sekali ketika Anda pertama kali mengatur konfigurasi autentikasi kuorum untuk CO. Setelah Anda menyelesaikan langkah ini, lihat Menggunakan otentikasi kuorum untuk petugas kripto.

Topik

- Prasyarat
- Buat dan daftarkan kunci untuk penandatanganan
- 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, klaster AWS CloudHSM memiliki dua HSM, masing-masing dengan CO yang sama, seperti yang ditunjukkan dalam output berikut dari perintah listUsers. Untuk informasi selengkapnya tentang membuat pengguna, lihat Mengelola 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               PRECO           admin                               NO
          0               NO
        2               AU              app_user                            NO
          0               NO
        3               CO              officer1                            NO
          0               NO
        4               CO              officer2                            NO
          0               NO
```

```
         5              CO             officer3                                NO
           0               NO
         6              CO             officer4                                NO
           0               NO
         7              CO             officer5                                NO
           0               NO
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                                   NO
           0               NO
         2              AU             app_user                                NO
           0               NO
         3              CO             officer1                                NO
           0               NO
         4              CO             officer2                                NO
           0               NO
         5              CO             officer3                                NO
           0               NO
         6              CO             officer4                                NO
           0               NO
         7              CO             officer5                                NO
           0               NO
```

Buat dan daftarkan kunci untuk penandatanganan

Untuk menggunakan autentikasi kuorum, setiap CO harus melakukan semualangkah berikut:

Topik

- Buat pasangan kunci RSA
- Membuat dan menandatangani token pendaftaran
- Daftarkan kunci publik dengan HSM

Buat pasangan kunci 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 menandatanganinya 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 mendaftar 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>

Jalur ke file yang berisi mekanisme SHA256_PKCS ditandatangani hash 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 CO mendaftarkan kunci publik mereka, output dari perintah listUsers menunjukkan ini di kolom `MofnPubKey`, seperti yang ditunjukkan dalam contoh berikut.

```
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              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             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         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
```

Tetapkan nilai minimum kuorum pada HSM

Untuk menggunakan autentikasi kuorum untuk CO, CO harus log in ke HSM dan kemudian mengatur 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 mengatur nilai minimum kuorum, termasuk CO yang belum mendaftarkan kunci untuk ditandatangani. Anda dapat mengubah nilai minimum kuorum kapan saja; untuk informasi lebih lanjut, lihatUbah 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 apa pun dari dua (2) hingga delapan (8), hingga jumlah total CO pada HSM. Dalam contoh ini, HSM memiliki enam CO, sehingga nilai maksimum yang mungkin adalah enam.

Untuk menggunakan perintah contoh berikut, ganti nomor akhir (*2*) dengan nilai minimum kuorum pilihan.

```
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 pengenal layanan HSM bersama dengan nama, deskripsi, dan perintah yang disertakan dalam layanan.

Pengenal 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 Menggunakan otentikasi kuorum untuk petugas kripto.

Menggunakan otentikasi kuorum untuk petugas kripto

Petugas kripto (CO) pada HSM dapat mengatur konfigurasi autentikasi kuorum untuk operasi berikut pada HSM:

- Membuat pengguna HSM
- Membuat pengguna HSM
- Mengubah kata sandi pengguna HSM lainnya

Setelah HSM dikonfigurasi untuk autentikasi kuorum, CO 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 CO lainnya.
3. Setujui token pada HSM.
4. Melakukan operasi manajemen pengguna HSM.

Jika Anda belum mengatur konfigurasi HSM untuk autentikasi kuorum untuk CO, lakukan itu sekarang. Untuk informasi lebih lanjut, lihat Pengaturan pertama kali.

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 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 lebih lanjut, lihat 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
```

Dapatkan tanda tangan dari CO pemberi persetujuan

Seorang CO yang memiliki token kuorum harus mendapatkan token yang disetujui oleh CO lainnya. Untuk memberikan persetujuan mereka, CO lain menggunakan kunci penandatanganan mereka untuk secara kriptografi menandatangani token. 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 penandatangan) untuk menandatangani SHA-256 digest token .

Example — Dapatkan tanda tangan dari CO pemberi persetujuan

Dalam contoh ini, CO yang memiliki token (officer1) membutuhkan setidaknya dua persetujuan. Contoh perintah berikut menunjukkan bagaimana dua CO dapat menggunakan OpenSSL untuk secara kriptografi menandatangani token.

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:
```

Untuk menyetujui token yang ditandatangani pada HSM

Setelah CO mendapat jumlah minimum persetujuan (tanda tangan) dari CO lain, 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 setujui 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
```

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 Mengelola 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
  LoginFailureCnt         2FA
        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
Users on server 1(10.0.1.4):
Number of users found:8

    User Id             User Type       User Name                      MofnPubKey
  LoginFailureCnt         2FA
         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 untuk petugas kripto

Setelah Anda tetapkan nilai minimum kuorum sehingga petugas kripto (CO) dapat menggunakan autentikasi kuorum, Anda mungkin perlu 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. Sebagai contoh, jika nilai minimum kuorum adalah dua, setidaknya dua CO 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 CO (layanan 3), Anda mungkin perlu mengubah nilai minimum kuorum untuk layanan 4.

Tabel berikut mencantumkan pengenal layanan HSM bersama dengan nama, deskripsi, dan perintah yang disertakan dalam layanan.

Pengenal 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 mengubah nilai minimum kuorum untuk petugas kripto

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 getMVValue untuk mendapatkan nilai minimum kuorum untuk layanan 3. Untuk informasi selengkapnya, lihat contoh berikut.
4. Gunakan perintah getMVValue untuk mendapatkan nilai minimum kuorum untuk layanan 4. Untuk informasi selengkapnya, lihat contoh berikut.
5. Jika nilai minimum kuorum untuk layanan 4 lebih rendah dari nilai untuk layanan 3, gunakan perintah setMVValue 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.
6. Dapatkan token kuorum, berhati-hatilah untuk menentukan layanan 4 sebagai layanan yang dapat Anda gunakan dengan token.
7. Dapatkan persetujuan (tanda tangan) dari CO lainnya.
8. Setujui token pada HSM.

9. Gunakan perintah setMValue untuk mengubah nilai minimum kuorum untuk layanan 3 (operasi manajemen pengguna dilakukan oleh CO).

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_CO] on server 0 : [1]
MValue of service 4[MISC_CO] 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_CO] on server 0 : [2]
MValue of service 4[MISC_CO] on server 1 : [2]
```

Mengelola kunci di AWS CloudHSM

Di AWS CloudHSM, gunakan salah satu dari berikut ini untuk mengelola kunci pada HSM 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.

Topik

Sinkronisasi kunci dan pengaturan daya tahan di AWS CloudHSM

Topik ini menjelaskan pengaturan sinkronisasi kunci di AWS CloudHSM, masalah umum yang dihadapi pelanggan yang bekerja dengan kunci pada sebuah klaster, dan strategi untuk membuat kunci lebih tahan lama.

Topik

- Menyinkronkan kunci di seluruh cluster kloning

Konsep

Tombol Token

Kunci persisten yang Anda buat selama operasi membuat, impor, atau membuka kunci. AWS CloudHSM menyinkronkan kunci token di klaster.

Tombol sesi

Kunci sementara yang ada hanya pada satu modul keamanan perangkat keras (HSM) di klaster. AWS CloudHSM tidak menyinkronkan kunci sesi di klaster.

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 klaster dengan minimal dua HSM.

Sinkronisasi tombol sisi server

Secara berkala klon kunci ke setiap HSM di klaster. Tidak memerlukan manajemen.

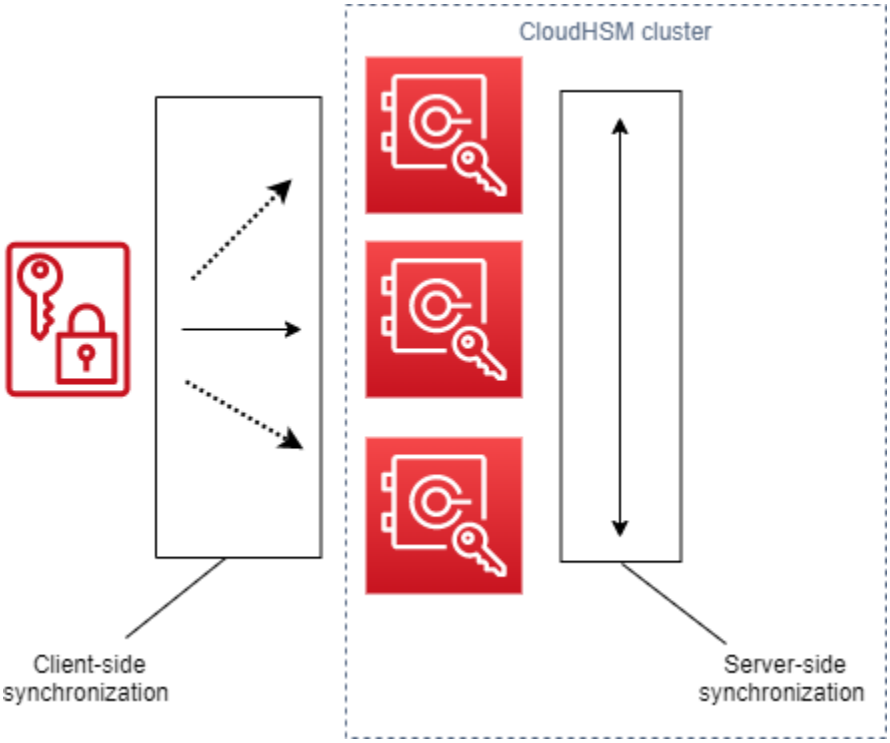
Pengaturan daya tahan kunci klien

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 SDK Klien 3, gunakan pengaturan ini untuk menentukan jumlah HSM yang diperlukan untuk operasi pembuatan kunci agar berhasil.

Memahami sinkronisasi kunci

AWS CloudHSM menggunakan sinkronisasi kunci untuk mengkloning kunci token di semua HSM dalam sebuah klaster. 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 mengatur konfigurasi klaster Anda untuk menggunakan minimal dua HSM. Dengan sinkronisasi sisi klien dan klaster dengan dua HSM, Anda dapat memenuhi tantangan daya tahan kunci dalam lingkungan cloud.

Bekerja dengan pengaturan daya tahan kunci 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.

- Dalam SDK Klien 5, kami memperkenalkan konsep kuorum ketersediaan utama yang mengharuskan Anda untuk menjalankan klaster dengan minimal dua HSM. 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 SDK 5 klien”.
- Di SDK Klien 3, Anda menggunakan pengaturan daya tahan kunci sisi klien untuk menentukan jumlah HSM yang pembuatan kuncinya harus berhasil untuk keseluruhan operasi agar dianggap sukses.

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 HSM di klaster sebelum aplikasi Anda dapat menggunakan kunci. Untuk menggunakan kuorum ketersediaan kunci, klaster Anda harus memiliki minimal dua HSM.

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 HSM dalam sebuah klaster yang kuncinya harus ada sebelum aplikasi Anda dapat menggunakan kunci. Membutuhkan klaster dengan minimal dua HSM.

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.

```
"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
```

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.

```
"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.

```
"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 HSM yang pembuatan kuncinya harus berhasil untuk keseluruhan operasi agar 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 jumlah HSM yang Anda tentukan. Jika Anda menentukan nilai dan sistem tidak dapat mereplikasi kunci ke jumlah HSM tersebut, 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 utama, pertimbangkan untuk menentukan setidaknya dua HSM untuk sinkronisasi sisi klien. Ingat bahwa tidak peduli berapa banyak HSM yang Anda tentukan, beban kerja pada klaster Anda tetap sama. Sinkronisasi sisi klien selalu membuat upaya terbaik untuk mengkloning kunci ke setiap HSM di klaster.

Rekomendasi

- Minimum: Dua HSM per klaster
- Maksimum: Satu lebih sedikit dari jumlah total HSM di klaster

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 simpul `client` dari file, tambahkan `create_object_minimum_nodes` dan tentukan nilai untuk jumlah minimum HSM tempat AWS CloudHSM harus berhasil membuat kunci untuk operasi pembuatan kunci agar 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 HSM yang diperlukan untuk menganggap operasi pembuatan kunci, impor kunci, atau 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

klaster, tetapi untuk kembali sukses, hanya perlu untuk membuat satu kunci tunggal pada satu HSM di klaster.

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 bilangan HSM untuk mengkloning kunci. Jika Anda menentukan parameter baris perintah dan nilai dalam file konfigurasi, AWS CloudHSMmenghormati LEBIH dari dua nilai.

Untuk informasi selengkapnya, lihat topik berikut:

- GenDSA KeyPair
- GeneCC KeyPair
- GenRSA KeyPair
- genSymKey
- importPrivateKey
- importPubKey
- imSymKey
- insertMaskedObject
- unWrapKey

Menyinkronkan kunci di seluruh cluster kloning

Sinkronisasi sisi klien dan sisi server hanya untuk menyinkronkan kunci dalam klaster yang sama. Jika Anda menyalin cadangan klaster ke wilayah lain, Anda dapat menggunakan perintah syncKey dari cloudhsm_mgmt_util (CMU) untuk menyinkronkan kunci di antara klaster. Anda mungkin menggunakan kloning klaster untuk redundansi lintas wilayah atau untuk menyederhanakan proses pemulihan bencana Anda. Untuk informasi selengkapnya, lihat syncKey.

Pembungkus kunci AES di 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

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 ditambahkan 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, makan gunakan IV implisit untuk membuka. Dengan IV implisit, CloudHSM akan menggunakan nilai default untuk membuka.

Tabel berikut menjelaskan nilai-nilai yang diizinkan untuk IV, yang ditetapkan algoritme 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)	Diizinkan

Algoritme Bungkus Kunci AES	IV implisit	IV eksplisit
	Nilai default: `0xA6A6A6A6A6A6A6A6`	Hanya nilai ini yang diterima: `0xA6A6A6A6A6A6A6A6`

Menggunakan bungkus kunci AES di 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 buka kunci didukung dan seperti bungkus kunci yang ditunjukkan pada tabel berikut.
- Di key_mgmt_util (KMU), gunakan perintah wrapKey dan unWrapKey dengan nilai m yang sesuai seperti yang ditunjukkan dalam tabel berikut.

Algoritme Bungkus Kunci AES	Mekanisme PKCS #11	Metode Java	Sub Komando CLI	Argumen Utilitas Manajemen Kunci (KMU)
Bungkus Kunci AES dengan Nol Bantalan	• `CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD` (Mekanisme Ditetapkan Vendor)	`AESWrap/ECB/ZeroPadding`	aes-zero-pad	m = 6
Bungkus Kunci AES dengan Tanpa Bantalan	• `CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD` (Mekanism	`AESWrap/ECB/NoPadding`	aes-no-pad	m = 5

Algoritme Bungkus Kunci AES	Mekanisme PKCS #11	Metode Java	Sub Komando CLI	Argumen Utilitas Manajemen Kunci (KMU)
	e Ditetapkan Vendor)			
Bungkus Kunci AES dengan Bantalan PKCS #5	• `CKM_CLOUDHSM_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
- Atribut kunci tepercaya
- Cara menggunakan kunci tepercaya untuk membungkus kunci data
- Cara membuka kunci data dengan kunci tepercaya

Memahami kunci tepercaya

Kunci tepercaya adalah kunci yang digunakan untuk membungkus kunci lain dan bahwa admin dan petugas kriptografi (CO) secara khusus mengidentifikasi sebagai tepercaya menggunakan atribut. `CKA_TRUSTED` Selain itu, admin dan petugas kriptografi (CO) 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

Atribut berikut memungkinkan Anda menandai 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 (selainCKA_UNWRAP_TEMPLATE) ke kunci yang akan membungkus kunci data untuk menentukan bahwa admin atau petugas kripto (CO) telah melakukan ketekunan yang diperlukan dan mempercayai kunci ini. Hanya admin atau CO yang dapat mengaturCKA_TRUSTED. Pengguna kripto (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 sebagaiCKA_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

Untuk menggunakan kunci tepercaya untuk membungkus kunci data, 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 mengatur `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 menunjukkan 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, lihatkunci generate-asymmetric-pair .

Untuk informasi lebih lanjut tentang pembuatan kunci simetris, lihatkunci menghasilkan-simetris .

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 login perintah untuk masuk sebagai pengguna kripto (CU).
2. Gunakan kunci set-atribut 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` Cara menandai kunci sebagai tepercaya dengan CloudHSM CLI
- Jika menggunakan CMU untuk menyetel `CKA_TRUSTED` atribut kunci, lihatCara menandai kunci sebagai tepercaya dengan CMU.

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 CloudHSMContoh PKCS #11
- AWS CloudHSMContoh JCE

Cara membuka kunci data dengan kunci tepercaya

Untuk membuka kunci data, 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.

Mengelola kunci dengan CloudHSM CLI

Jika menggunakan seri versi SDK terbaru, gunakan CloudHSM CLI untuk mengelola kunci di klaster Anda. AWS CloudHSM Untuk lebih jelasnya, lihat topik di bawah ini.

- Menggunakan kunci tepercaya menjelaskan cara menggunakan atribut library PKCS #11 dan 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.

Menggunakan CloudHSM CLI untuk menghasilkan kunci

Sebelum Anda dapat membuat kunci, Anda harus memulai CloudHSM CLI dan masuk sebagai pengguna kripto (CU). Untuk menghasilkan kunci pada HSM, gunakan perintah yang sesuai dengan jenis kunci yang ingin Anda hasilkan.

Topik

- Hasilkan kunci simetris
- Hasilkan kunci asimetris

- Topik terkait

Hasilkan kunci simetris

Gunakan perintah yang tercantum dalam kunci menghasilkan-simetris untuk menghasilkan kunci simetris. 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
```

Argumen

<LABEL>

Menentukan label yang ditentukan pengguna untuk kunci AES.

Diperlukan: Ya

<KEY-LENGTH-BYTES>

Menentukan panjang kunci dalam byte.

Nilai yang valid:

- 16, 24, dan 32

Diperlukan: Ya

<KEY_ATTRIBUTES>

Menentukan daftar spasi dipisahkan atribut kunci untuk mengatur kunci AES dihasilkan dalam bentuk `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (misalnya,) `token=true`

Untuk daftar atribut AWS CloudHSM kunci yang didukung, lihatAtribut kunci untuk CloudHSM CLI.

Diperlukan: 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 kunci persisten/token. Menggunakan <SESSION>perubahan ini, memastikan kunci yang dihasilkan dengan argumen ini adalah sesi/ephemural

Diperlukan: 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
```

Argumen

<LABEL>

Menentukan label yang ditentukan pengguna untuk kunci rahasia generik.

Diperlukan: Ya

<KEY-LENGTH-BYTES>

Menentukan panjang kunci dalam byte.

Nilai yang valid:

- 1 hingga 800

Diperlukan: 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,) `token=true`

Untuk daftar atribut AWS CloudHSM kunci yang didukung, lihatAtribut kunci untuk CloudHSM CLI.

Diperlukan: 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 kunci persisten/token. Menggunakan <SESSION>perubahan ini, memastikan kunci yang dihasilkan dengan argumen ini adalah sesi/ephemural

Diperlukan: Tidak

Hasilkan kunci asimetris

Gunakan perintah yang tercantum dalam kunci generate-asymmetric-pair untuk menghasilkan pasangan kunci asimetris.

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
```

Argumen

<PUBLIC_LABEL>

Menentukan label yang ditentukan pengguna untuk kunci publik.

Diperlukan: Ya

<PRIVATE_LABEL>

Menentukan label yang ditentukan pengguna untuk kunci pribadi.

Diperlukan: Ya

<MODULUS_SIZE_BITS>

Menentukan panjang modulus dalam bit. Nilai minimum adalah 2048.

Diperlukan: Ya

<PUBLIC_EXPONENT>

Menentukan eksponen publik. Nilai harus angka ganjil yang lebih besar dari atau sama dengan 65537.

Diperlukan: 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,). `token=true`

Untuk daftar atribut AWS CloudHSM kunci yang didukung, lihatAtribut kunci untuk CloudHSM CLI.

Diperlukan: 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 kunci persisten/token. Menggunakan <SESSION>perubahan ini, memastikan kunci yang dihasilkan dengan argumen ini adalah sesi/ephemural

Diperlukan: Tidak

Hasilkan pasangan kunci EC (kriptografi kurva elips)

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
```

Argumen

<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.

Diperlukan: 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.

Diperlukan: Ya

<CURVE>

Menentukan pengenal untuk kurva elips.

Nilai yang valid:

- primer256v1
- secp256r1
- secp224r1
- secp384r1
- secp256k1
- secp521r1

Diperlukan: 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,). `token=true`

Untuk daftar atribut AWS CloudHSM kunci yang didukung, lihatAtribut kunci untuk CloudHSM CLI.

Diperlukan: 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,). `token=true`

Untuk daftar atribut AWS CloudHSM kunci yang didukung, lihatAtribut kunci untuk CloudHSM CLI.

Diperlukan: 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).

Diperlukan: Tidak

Topik terkait

- Atribut kunci untuk CloudHSM CLI
- kunci generate-asymmetric-pair
- kunci menghasilkan-simetris

Menggunakan CloudHSM CLI untuk menghapus kunci

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": "0x0000000000540011",
        "key-info": {
          "key-owners": [
            {
              "username": "my_crypto_user",
```

```
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "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

Menggunakan CloudHSM CLI untuk berbagi dan membatalkan berbagi kunci

Gunakan perintah dalam topik ini untuk berbagi dan membatalkan berbagi kunci di CloudHSM CLI. Pada tahun AWS CloudHSM, pengguna kripto (CU) yang membuat kunci memilikinya. Pemilik dapat menggunakan key unshare perintah key share and untuk berbagi dan membatalkan pembagian kunci dengan CU lain. 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 kripto (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 kripto (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": "0x00000000001c0686",
        "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"
          },
        ],
        "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 kripto (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": "0x00000000001c0686",
        "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"
            }
          ],
          "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` kripto:

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" --verbose
{
```

```
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "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"
            },
          ],
          "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
- pembagian kunci
- kunci unshare
- Menggunakan CloudHSM CLI untuk memfilter kunci

Menggunakan CloudHSM CLI untuk memfilter kunci

Gunakan perintah kunci berikut untuk memanfaatkan mekanisme penyaringan kunci standar untuk CloudHSM CLI.

- key list
- key delete
- key share
- key unshare
- key set-attribute

Untuk memilih dan/atau memfilter kunci dengan CloudHSM CLI, perintah utama menggunakan mekanisme filtrasi standar berdasarkan. Atribut kunci untuk CloudHSM CLI Sebuah 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 penyaringan kunci hanya beroperasi pada kunci yang dimiliki dan dibagikan oleh pengguna yang saat ini masuk, serta semua kunci publik di AWS CloudHSM klaster.

Topik

- Persyaratan
- Penyaringan untuk menemukan satu kunci
- Kesalahan Filtrasi
- Topik terkait

Persyaratan

Untuk memfilter kunci, Anda harus masuk sebagai pengguna kripto (CU).

Penyaringan 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 menyaring ke kunci unik tunggal hanya menggunakan atribut mengidentifikasi tunggal.

```
aws-cloudhsm > key list --filter attr.label="my_unique_key_label" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "alice",
              "key-coverage": "full"
            }
          ],
          "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 satu kunci 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": "0x0000000000540011",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
```

```
              "key-coverage": "full"
            }
          ],
          "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 Penyaringan untuk menemukan satu set kunci

Contoh berikut menunjukkan bagaimana untuk menyaring 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": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "my_crypto_user",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
          ],
          "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": "0x0000000000540011",
        "key-info": {
          "key-owners": [
            {
              "username": "my_crypto_user",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            }
          ],
          "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 tombol pada satu waktu. Untuk operasi ini, CloudHSM CLI akan memberikan kesalahan jika kriteria filtrasi tidak cukup disempurnakan dan beberapa kunci sesuai dengan kriteria. Salah satu contohnya ditunjukkan di bawah ini dengan tombol delete.

Example Kesalahan Filtrasi saat mencocokkan terlalu banyak kunci

```
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

Cara menandai kunci sebagai tepercaya dengan 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).
2. Gunakan key list perintah untuk mengidentifikasi referensi kunci dari kunci yang ingin Anda tandai sebagai tepercaya. Contoh berikut mencantumkan kunci dengan labelkey_to_be_trusted.

```
aws-cloudhsm > key list --filter attr.label=test_aes_trusted
        {
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000200333",
        "attributes": {
          "label": "test_aes_trusted"
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

3. Menggunakan logout perintah, keluar sebagai pengguna kripto (CU).
4. Menggunakan login perintah, masuk sebagai admin.
5. 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"
  }
}
```

Mengelola kunci dengan KMU dan CMU

Jika menggunakan seri versi SDK terbaru, gunakan CloudHSM CLI untuk mengelola kunci di klaster Anda. AWS CloudHSM

Jika menggunakan seri versi SDK sebelumnya, Anda dapat mengelola kunci pada HSM di AWS CloudHSM cluster Anda menggunakan alat baris perintah key_mgmt_util. Sebelum Anda dapat mengelola kunci, Anda harus memulai klien AWS CloudHSM, mulai key_mgmt_util, dan masuk ke HSM. Untuk informasi selengkapnya, lihat Memulai dengan key_mgmt_util.

- Menggunakan kunci tepercaya menjelaskan cara menggunakan atribut pustaka 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.

Menghasilkan kunci

Untuk menghasilkan kunci pada HSM, gunakan perintah yang sesuai dengan jenis kunci yang ingin Anda hasilkan.

Topik

- Hasilkan kunci simetris
- Hasilkan pasangan kunci RSA
- Hasilkan pasangan kunci ECC (eliptic curve cryptography)

Hasilkan kunci simetris

Gunakan genSymKeyperintah untuk menghasilkan AES dan jenis kunci simetris lainnya. 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

Untuk menghasilkan key pair RSA, gunakan perintah KeyPairGenRSA. 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 (eliptic curve cryptography)

Untuk menghasilkan key pair ECC, gunakan perintah KeyPairGenECC. 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
```

Mengimpor kunci

Untuk mengimpor kunci rahasia—yaitu, kunci simetris dan kunci privat asimetris—ke HSM, Anda harus terlebih dahulu membuat kunci pembungkus pada HSM. Anda dapat mengimpor kunci publik secara langsung tanpa kunci pembungkus.

Topik

- Impor kunci rahasia
- Impor kunci publik

Impor kunci rahasia

Selesaikan langkah-langkah berikut untuk mengimpor kunci rahasia. 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 genSymKeyperintah 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

Gunakan perintah importPubKey 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
```

Mengekspor kunci

Untuk mengekspor kunci rahasia—yaitu, kunci simetris dan kunci privat asimetris—dari HSM, 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**
>
> exSymKeyPerintah 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

- Ekspor kunci publik

Ekspor kunci rahasia

Selesaikan langkah-langkah berikut untuk mengekspor kunci rahasia.

Untuk mengekspor kunci rahasia

1. Gunakan genSymKeyperintah 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 exSymKeyperintah. 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

Gunakan perintah exportPubKey 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
```

Menghapus kunci

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
```

Berbagi dan tidak berbagi kunci

Di AWS CloudHSM, CU yang membuat kunci memilikinya. Pemilik mengelola kunci, dapat mengekspor 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 mengekspor 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 genSymKeyatau GenRSA KeyPair. 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 end-to-end enkripsi, dan masuk ke HSM. Untuk berbagi kunci, masuk ke HSM sebagai pengguna kripto (CU) yang memiliki kunci. Hanya pemilik kunci yang dapat berbagi kunci.

Gunakan perintah shareKey untuk berbagi atau batal berbagi kunci, menentukan handel kunci dan ID pengguna atau pengguna. Untuk berbagi atau membatalkan berbagi dengan lebih dari satu pengguna, tentukan daftar ID pengguna dipisahkan koma. 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 CMU

Konten di bagian ini memberikan petunjuk tentang penggunaan CMU untuk menandai kunci sebagai tepercaya.

1. Menggunakan perintah LoginHSM, masuk sebagai petugas kripto (CO).
2. Gunakan setAttribute perintah dengan `OBJ_ATTR_TRUSTED` (value`134`) disetel ke true (1).

```
setAttribute <Key Handle> 134 1
```

Mengelola kloning kloning

Gunakan CloudHSM Management Utility (CMU) untuk menyinkronkan klaster di wilayah yang jauh, jika klaster di wilayah itu awalnya dibuat dari cadangan klaster 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 CMU untuk menyinkronkan klaster. Anda melakukan ini dengan membuat file konfigurasi CMU baru, menentukan modul keamanan perangkat keras (HSM) dari kedua klaster dalam file baru, dan kemudian menggunakan CMU untuk menyambung ke klaster dengan file tersebut.

Untuk menggunakan CMU di klaster kloning

1. Buat salinan file konfigurasi Anda saat ini dan ubah nama salinan ke sesuatu yang lain.

   Misalnya, gunakan lokasi file berikut untuk menemukan dan membuat salinan file konfigurasi Anda saat ini, lalu ubah nama salinan dari `cloudhsm_mgmt_config.cfg` ke `syncConfig.cfg` .

   - Linux: `/opt/cloudhsm/etc/cloudhsm_mgmt_config.cfg`
   - Windows: `C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_config.cfg`
2. Dalam salinan yang berganti nama, tambahkan IP Elastic Network Interface (ENI) HSM tujuan (HSM di wilayah asing yang perlu disinkronkan). Kami sarankan Anda untuk menambahkan HSM tujuan di bawah HSM sumber.

```
{
    ...
    "servers": [
        {
            ...
            "hostname": "<ENI Source IP>",
            ...
        },
        {
```

```
            ...
            "hostname": "<ENI Destination IP>",
            ...
        }
    ]
}
```

Untuk informasi lebih lanjut tentang cara mendapatkan alamat IP, lihat the section called "Dapatkan alamat IP untuk HSM".

3. Inisialisasi CMU dengan file konfigurasi baru:

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/userSync.cfg
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM>cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
   \CloudHSM\data\userSync.cfg
   ```

4. Periksa pesan status yang dikembalikan untuk memastikan bahwa CMU terhubung ke semua HSM yang diinginkan dan menentukan manakah dari IP ENI yang dikembalikan sesuai dengan setiap klaster. Gunakan syncUser dan syncKey untuk menyinkronkan pengguna dan kunci secara manual. Untuk informasi lebih lanjut, lihat. syncUser dan syncKey.

Dapatkan alamat IP untuk HSM

Gunakan bagian ini untuk mendapatkan alamat IP untuk HSM.

Untuk mendapatkan alamat IP untuk HSM (konsol)

1. Buka AWS CloudHSM konsol di https://console.aws.amazon.com/cloudhsm/home.
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, pada tab HSM, pilih salah satu alamat IP yang tercantum di bawah alamat IP ENI.

Untuk mendapatkan alamat IP untuk HSM (CLI)

- Dapatkan alamat IP HSM dengan menggunakan describe-clusters perintah dari CLI. Dalam output dari perintah, alamat IP dari HSM adalah nilai-nilai dari `EniIp`.

```
$  aws cloudhsmv2 describe-clusters


{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
...
```

Topik terkait

- SynCuser
- SyncKey
- Menyalin Cadangan Lintas Wilayah

AWS CloudHSM alat baris perintah

Topik ini menjelaskan alat baris perintah yang tersedia untuk mengelola dan menggunakan AWS CloudHSM.

Topik

Memahami alat baris perintah

Selain antarmuka baris perintah AWS (CLI) yang Anda gunakan untuk mengelola sumber daya AWS Anda, AWS CloudHSM menawarkan alat baris perintah untuk membuat dan mengelola pengguna dan kunci HSM di HSM Anda. Dalam 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 HSM dan cluster

Perintah CloudhSMv2 dalam cmdlet CLI dan HSM2 dalam modul PowerShell AWSPowerShell

- Alat-alat ini mendapatkan, membuat, menghapus, dan menandai AWS CloudHSM cluster dan HSM:
- Untuk menggunakan perintah dalam perintah CloudhSMv2 di CLI, Anda perlu menginstal dan mengkonfigurasi CLI.
- PowerShell Cmdlet HSM2 dalam AWSPowerShell modul tersedia dalam modul Windows dan PowerShell 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 multifaktor 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 HSM 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 SDK sebelumnya

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 .

Alat konfigurasi

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 HSM di klaster 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, lihatManfaat SDK 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 Klien 3 ke SDK Klien 5

Topik

- Alat konfigurasi SDK 5 klien
- Alat konfigurasi SDK 3 klien

Alat konfigurasi SDK 5 klien

Gunakan alat konfigurasi Klien 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.

Sintaks

PKCS #11

```
configure-pkcs11[ .exe ]
             -a <ENI IP address>
             [--hsm-ca-cert <customerCA certificate file path>]
             [--cluster-id <cluster ID>]
             [--endpoint <endpoint>]
             [--region <region>]
             [--server-client-cert-file <client certificate file path>]
             [--server-client-key-file <client key file path>]
             [--log-level <error | warn | info | debug | trace>]
                  Default is <info>
             [--log-rotation <daily | weekly>]
                  Default is <daily>
```

```
              [--log-file <file name with path>]
                   Default is </opt/cloudhsm/run/cloudhsm-pkcs11.log>
                   Default for Windows is <C:\\Program Files\\Amazon\\CloudHSM\
\cloudhsm-pkcs11.log>
              [--log-type <file | term>]
                   Default is <file>
              [-h | --help]
              [-V | --version]
              [--disable-key-availability-check]
              [--enable-key-availability-check]
              [--disable-validate-key-at-init]
              [--enable-validate-key-at-init]
                   This is the default for PKCS #11
```

OpenSSL

```
configure-dyn[ .exe ]
              -a <ENI IP address>
              [--hsm-ca-cert <customerCA certificate file path>]
              [--cluster-id <cluster ID>]
              [--endpoint <endpoint>]
              [--region <region>]
              [--server-client-cert-file <client certificate file path>]
              [--server-client-key-file <client key file path>]
              [--log-level <error | warn | info | debug | trace>]
                   Default is <error>
              [--log-type <file | term>]
                   Default is <term>
              [-h | --help]
              [-V | --version]
              [--disable-key-availability-check]
              [--enable-key-availability-check]
              [--disable-validate-key-at-init]
                   This is the default for OpenSSL
              [--enable-validate-key-at-init]
```

JCE

```
configure-jce[ .exe ]
              -a <ENI IP address>
              [--hsm-ca-cert <customerCA certificate file path>]
              [--cluster-id <cluster ID>]
              [--endpoint <endpoint>]
```

```
            [--region <region>]
            [--server-client-cert-file <client certificate file path>]
            [--server-client-key-file <client key file path>]
            [--log-level <error | warn | info | debug | trace>]
                 Default is <info>
            [--log-rotation <daily | weekly>]
                 Default is <daily>
            [--log-file <file name with path>]
                 Default is </opt/cloudhsm/run/cloudhsm-jce.log>
                 Default for Windows is <C:\\Program Files\\Amazon\\CloudHSM\
\cloudhsm-jce.log>
            [--log-type <file | term>]
                 Default is <file>
            [-h | --help]
            [-V | --version]
            [--disable-key-availability-check]
            [--enable-key-availability-check]
            [--disable-validate-key-at-init]
                 This is the default for JCE
            [--enable-validate-key-at-init]
```

CloudHSM CLI

```
configure-cli[ .exe ]
            -a <ENI IP address>
            [--hsm-ca-cert <customerCA certificate file path>]
            [--cluster-id <cluster ID>]
            [--endpoint <endpoint>]
            [--region <region>]
            [--server-client-cert-file <client certificate file path>]
            [--server-client-key-file <client key file path>]
            [--log-level <error | warn | info | debug | trace>]
                 Default is <info>
            [--log-rotation <daily | weekly>]
                 Default is <daily>
            [--log-file <file name with path>]
                 Default for Linux is </opt/cloudhsm/run/cloudhsm-cli.log>
                 Default for Windows is <C:\\Program Files\\Amazon\\CloudHSM\
\cloudhsm-cli.log>
            [--log-type <file | term>]
                 Default setting is <file>
            [-h | --help]
            [-V | --version]
```

```
            [--disable-key-availability-check]
            [--enable-key-availability-check]
            [--disable-validate-key-at-init]
                 This is the default for CloudHSM CLI
            [--enable-validate-key-at-init]
```

Konfigurasi lanjutan

Untuk daftar konfigurasi lanjutan khusus untuk alat konfigurasi SDK 5 Klien, lihat Konfigurasi lanjutan untuk alat konfigurasi SDK 5 Klien.

> **Important**
>
> Setelah membuat perubahan pada konfigurasi Anda, Anda perlu me-restart aplikasi Anda agar perubahan diterapkan.

Contoh-contoh

Contoh ini menunjukkan cara menggunakan alat konfigurasi untuk Klien SDK 5.

Bootstrap Klien SDK 5

Example

Contoh ini menggunakan parameter `-a` untuk memperbarui data HSM untuk klien SDK 5. Untuk menggunakan parameter `-a`, Anda harus memiliki alamat IP untuk salah satu HSM di klaster 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.

```
"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>
```

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.

```
"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 IP 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.

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

> **Note**
>
> Anda dapat menggunakan `--cluster-id` parameter sebagai pengganti`-a` <HSM_IP_ADDRESSES>. Untuk melihat persyaratan penggunaan`--cluster-id`, lihatAlat konfigurasi SDK 5 klien.

Untuk informasi tentang parameter `-a`, lihat the section called “Parameter-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.

```
"C:\Program Files\Amazon\CloudHSM\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
```

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.

```
"C:\Program Files\Amazon\CloudHSM\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.

```
"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.

```
C:\Program Files\Amazon\CloudHSM\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
```

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.

```
"C:\Program Files\Amazon\CloudHSM\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.

```
"C:\Program Files\Amazon\CloudHSM\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-parameter”.

Perbarui sertifikat klien dan kunci untuk otentikasi timbal balik client-server TLS

Example

Contoh ini menunjukkan cara menggunakan `server-client-cert-file` dan `--server-client-key-file` parameter untuk mengkonfigurasi ulang SSL dengan menentukan kunci kustom dan sertifikat SSL untuk AWS CloudHSM

PKCS #11 library

Untuk menggunakan sertifikat kustom dan kunci untuk autentikasi mutual server-klien TLS dengan Klien SDK 5 di Linux

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

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

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

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

Untuk menggunakan sertifikat kustom dan kunci untuk autentikasi mutual klien-server TLS dengan Klien SDK 5 pada Windows

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

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

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

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

OpenSSL Dynamic Engine

Untuk menggunakan sertifikat kustom dan kunci untuk autentikasi mutual server-klien TLS dengan Klien SDK 5 di Linux

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

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

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

```
$ sudo /opt/cloudhsm/bin/configure-dyn \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

JCE provider

Untuk menggunakan sertifikat kustom dan kunci untuk autentikasi mutual server-klien TLS dengan Klien SDK 5 di Linux

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

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

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

```
$ sudo /opt/cloudhsm/bin/configure-jce \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

Untuk menggunakan sertifikat kustom dan kunci untuk autentikasi mutual klien-server TLS dengan Klien SDK 5 pada Windows

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

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

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

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

CloudHSM CLI

Untuk menggunakan sertifikat kustom dan kunci untuk autentikasi mutual server-klien TLS dengan Klien SDK 5 di Linux

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

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

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

```
$ sudo /opt/cloudhsm/bin/configure-cli \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

Untuk menggunakan sertifikat kustom dan kunci untuk autentikasi mutual klien-server TLS dengan Klien SDK 5 pada Windows

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

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

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

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

Untuk informasi tentang parameter `server-client-cert-file` dan `--server-client-key-file`, lihat the section called “Parameter-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.

```
"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
```

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.

```
"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.

```
"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-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 tersebut, sistem akan 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 tersebut, sistem akan 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 name> --log-file file --
  log-rotation daily --log-level info
  ```

Untuk mengonfigurasi opsi pencatatan terminal

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-type term --log-level info
  ```

JCE provider

Lokasi pencatatan default

- Jika Anda tidak menentukan lokasi untuk file tersebut, sistem akan 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> --
  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 tersebut, sistem akan 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> --
  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, lihatthe section called “Parameter-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.

  ```
  "C:\Program Files\Amazon\CloudHSM\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>
```

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.

```
"C:\Program Files\Amazon\CloudHSM\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.

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

Untuk informasi tentang parameter `--hsm-ca-cert`, lihat the section called “Parameter-parameter”.

Parameter-parameter

- sebuah *<alamat IP ENI>*

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.

Diperlukan: 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 klaster. Anda membuat file ini ketika Anda menginisialisasi klaster. 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 klaster 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 klaster yang terkait dengan ID klaster. 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 dengan. 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`.

Diperlukan: Tidak

-- server-client-cert-file <client certificate file path>

Jalur ke sertifikat klien yang digunakan untuk autentikasi mutual klien-server TLS.

Hanya gunakan opsi ini jika Anda tidak ingin menggunakan kunci default dan sertifikat SSL/TLS yang kami sertakan dengan Klien SDK 5. Anda harus menetapkan pilihan ini dalam kombinasi dengan `--server-client-key-file`.

Diperlukan: Tidak

-- server-client-key-file <client key file path>

Jalur ke kunci klien yang digunakan untuk otentikasi timbal balik client-server TLS.

Hanya gunakan opsi ini jika Anda tidak ingin menggunakan kunci default dan sertifikat SSL/TLS yang kami sertakan dengan Klien SDK 5. Anda harus menetapkan pilihan ini dalam kombinasi dengan `--server-client-cert-file`.

Wajib: Tidak

--log-level ***<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.

Diperlukan: Tidak

--log-rotasi <daily | weekly>

Menentukan frekuensi sistem memutar log. Untuk informasi lebih lanjut, lihat Pencatatan Klien SDK 5.

Diperlukan: Tidak

--log-file <file name with path>

Menentukan tempat sistem akan menulis berkas log. Untuk informasi lebih lanjut, lihat Pencatatan Klien SDK 5.

Diperlukan: 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

-v | --version

Menampilkan versi.

Diperlukan: 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 ???.

Diperlukan: 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 untuk menggunakan kunci sampai kunci tersebut ada pada dua HSM di cluster. Untuk informasi lebih lanjut tentang penggunaan bendera ini untuk mengatur kuorum ketersediaan kunci, lihat ???.

Diaktifkan secara default.

Diperlukan: Tidak

-- disable-validate-key-at -init

Meningkatkan performa dengan menentukan bahwa Anda dapat melewatkan 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.

Diperlukan: 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 menangguhkan mereka.

Wajib: Tidak

Topik terkait

- Operasi API DescribeClusters
- describe-clusters AWS CLI
- Dapatkan-HSM2Cluster cmdlet PowerShell
- Bootstrap Klien SDK 5
- AWS CloudHSM Titik akhir VPC
- Mengelola Pengaturan Daya Tahan Kunci SDK 5 Klien
- Klien SDK 5 Pencatatan

Konfigurasi lanjutan untuk alat konfigurasi Client SDK 5

Alat konfigurasi Client SDK 5 mencakup konfigurasi lanjutan yang bukan merupakan bagian dari fitur umum yang digunakan sebagian besar pelanggan. Konfigurasi lanjutan memberikan kemampuan tambahan.

- Konfigurasi lanjutan untuk PKCS #11
  - Menghubungkan ke beberapa slot dengan PKCS #11
  - Coba lagi perintah untuk PKCS #11

- Konfigurasi lanjutan untuk JCE
  - Menghubungkan ke beberapa klaster dengan penyedia JCE
  - Coba lagi perintah untuk JCE
  - Ekstraksi kunci menggunakan JCE
- Konfigurasi lanjutan untuk OpenSSL
  - Coba lagi perintah untuk OpenSSL
- Konfigurasi lanjutan untuk Antarmuka Baris Perintah (CLI)
  - Menghubungkan ke beberapa cluster dengan CLI

Alat konfigurasi SDK 3 klien

Gunakan alat konfigurasi Klien SDK 3 untuk bootstrap daemon klien dan mengatur konfigurasi Utilitas Manajemen CloudHSM.

Sintaks

```
configure -h | --help
          -a <ENI IP address>
          -m [-i <daemon_id>]
          --ssl --pkey <private key file> --cert <certificate file>
          --cmu <ENI IP address>
```

Contoh-contoh

Contoh ini menunjukkan cara menggunakan alat configure.

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 parameter `-a`, Anda harus memiliki alamat IP untuk salah satu HSM di klaster 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/home.
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, pada tab HSM, pilih salah satu alamat IP yang tercantum di bawah alamat IP ENI.

Untuk mendapatkan alamat IP untuk HSM (CLI)

- Dapatkan alamat IP HSM dengan menggunakan describe-clusters perintah dari CLI. Dalam output dari perintah, alamat IP dari HSM adalah nilai-nilai dari `EniIp`.

```
$  aws cloudhsmv2 describe-clusters


{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
...
```

Untuk memperbarui data HSM

1. 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

```
C:\Program Files\Amazon\CloudHSM\bin\ 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, klien menulis alamat IP ENI dari semua HSM di klaster ke file `cluster.info`.

   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

  ```
  C:\Program Files\Amazon\CloudHSM\bin\ 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

  ```
  C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
  ```

Parameter-parameter

-h | --help

Menampilkan sintaks perintah.

Wajib: Ya

- sebuah ***<alamat IP ENI>***

Menambahkan alamat IP antarmuka jaringan elastis (ENI) HSM tertentu untuk file konfigurasi AWS CloudHSM . Masukkan alamat IP ENI dari salah satu HSM di klaster. Tidak masalah yang mana yang Anda pilih.

Untuk mendapatkan alamat IP ENI dari HSM di cluster Anda, gunakan DescribeClustersoperasi, perintah CLI deskripsi-cluster, atau cmdlet 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 HSM di cluster.

Diperlukan: 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 Klient 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 HSM 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.

Wajib: 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.

Wajib: 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

Topik terkait

- Mengatur key_mgmt_util

Antarmuka Baris Perintah CloudHSM (CLI)

CloudHSM CLI membantu admin mengelola pengguna dan pengguna kripto mengelola kunci di klaster mereka. Ini termasuk alat yang dapat digunakan untuk membuat, menghapus dan daftar pengguna, mengubah kata sandi pengguna, memperbarui otentikasi multifaktor pengguna (MFA). Ini juga mencakup perintah yang menghasilkan, menghapus, mengimpor, dan mengekspor kunci, mendapatkan dan mengatur atribut, menemukan kunci, dan melakukan operasi kriptografi.

Untuk daftar pengguna CloudHSM CLI yang ditentukan, lihat. Mengelola 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. Mengelola kunci dengan CloudHSM CLI

Untuk memulai dengan cepat, lihat Memulai dengan CloudHSM Command Line Interface (CLI). Untuk informasi rinci tentang perintah CloudHSM CLI dan contoh penggunaan perintah, lihat. Referensi untuk perintah CloudHSM CLI

Topik

Platform yang didukung Antarmuka Baris Perintah CloudHSM (CLI)

Dukungan Linux

Platform yang didukung	X86_64 Arsitektur	Arsitektur ARM
Amazon Linux 2	Ya	Ya
Amazon Linux 2023	Ya	Ya
CentOS 7 (7,8+)	Ya	Tidak
Perusahaan Topi Merah Linux 7 (7.8+)	Ya	Tidak
Perusahaan Topi Merah Linux 8 (8.3+)	Ya	Tidak
Perusahaan Topi Merah Linux 9 (9.2+)	Ya	Ya
Ubuntu 20.04 LTS	Ya	Tidak

Platform yang didukung	X86_64 Arsitektur	Arsitektur ARM
Ubuntu 22.04 LTS	Ya	Ya

Catatan: 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

Memulai dengan CloudHSM Command Line Interface (CLI)

CloudHSM Command Line Interface (CLI) memungkinkan Anda untuk mengelola pengguna di cluster Anda. AWS CloudHSM Gunakan topik ini untuk memulai tugas manajemen pengguna HSM dasar, seperti membuat pengguna, mencantumkan pengguna, dan menghubungkan CloudHSM CLI ke cluster.

Instal CloudHSM CLI

Gunakan perintah berikut untuk mengunduh dan menginstal CloudHSM CLI.

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 pada arsitektur ARM64:

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

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

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 pada arsitektur ARM64:

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

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

CentOS 7 (7.8+)

CentOS 7 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
```

RHEL 7 (7.8+)

RHEL 7 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
```

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 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 pada arsitektur ARM64:

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

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

Ubuntu 20.04 LTS

Ubuntu 20.04 LTS pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Focal/cloudhsm-cli_latest_u20.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u20.04_amd64.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 arsitektur ARM64:

```
$ 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 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 msiexec.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 msiexec.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 di cluster Anda.

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

Untuk melakukan bootstrap instans EC2 Windows untuk Klien SDK 5

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

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

Menggunakan CloudHSM CLI

1. Gunakan perintah berikut untuk memulai CloudHSM CLI.

   Linux

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

   Windows

   ```
   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 CU 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 klaster.

```
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"
      }
    ]
  }
}
```

6. Gunakan logout perintah untuk keluar dari AWS CloudHSM cluster.

```
aws-cloudhsm > logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

7. Gunakan quit perintah untuk menghentikan CLI.

```
aws-cloudhsm > quit
```

Mode perintah interaktif dan tunggal

Di CloudHSM CLI, Anda dapat menjalankan perintah 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.

Modus interaktif

Gunakan perintah berikut untuk memulai mode interaktif CloudHSM CLI

Linux

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

Windows

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

Saat menggunakan CLI dalam Mode Interaktif, Anda dapat masuk ke akun pengguna menggunakan login perintah.

Untuk mencantumkan 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 HSM, masukkanuser list.

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

Untuk mengakhiri sesi CloudHSM CLI Anda, jalankan perintah berikut:

```
aws-cloudhsm > quit
```

Mode Perintah Tunggal

Jika Anda menjalankan CloudHSM CLI menggunakan Mode Perintah Tunggal, Anda perlu menyetel 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 kredenial 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 menetapkan 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 lainnya dapat diatur 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 indukkunci.

Atribut yang didukung

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, nilai yang mungkin, default, dan catatan terkait. Sel kosong di kolom Nilai menunjukkan bahwa tidak ada nilai default tertentu yang ditetapkan ke atribut.

Atribut CloudHSM CLI	Nilai	Dimodifikasi dengan kunci set-atribut	Dapat diatur pada pembuatan kunci
`always-sensitive`	Nilainya adalah `True` jika `sensitive` selalu diatur `True` dan tidak pernah berubah.	Tidak	Tidak
`check-value`	Nilai cek kunci. Untuk informasi lebih lanjut, lihat Detail tambahan.	Tidak	Tidak
`class`	Nilai yang mungkin:`secret-key` ,`public-key` , dan`private-key` .	Tidak	Ya
`curve`	Kurva elips digunakan untuk menghasilkan pasangan kunci EC. Nilai yang Valid: `secp224r1` `secp256r1` ,`prime256v1` ,`secp384r1` ,`secp256k1` ,, dan `secp521r1`	Tidak	Dapat diatur dengan RSA, tidak dapat diatur dengan EC

Atribut CloudHSM CLI	Nilai	Dimodifikasi dengan kunci set-atribut	Dapat diatur pada pembuatan kunci
`decrypt`	Default: `False`	Ya	Ya
`derive`	Default: `False`	Ya	Ya
`destroyable`	Default: `True`	Ya	Ya
`ec-point`	Untuk kunci EC, DER-encoding ANSI X9.62 nilai ECpoint “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	Tidak	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	Dimodifikasi dengan kunci set-atribut	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`	Tidak	Tidak
`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 pasangan kunci RSA. Nilai minimum adalah`2048`.	Tidak	Dapat diatur dengan RSA, tidak dapat diatur dengan EC
`never-extractable`	Nilai adalah `True` jika diekstraksi belum pernah diatur ke`False`. Nilainya adalah `False` jika diekstraksi pernah diatur ke`True`.	Tidak	Tidak
`private`	Default: `True`	Tidak	Ya

Atribut CloudHSM CLI	Nilai	Dimodifikasi dengan kunci set-atribut	Dapat diatur pada pembuatan kunci
`public-exponent`	Diperlukan untuk menghasilkan pasangan kunci 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.
`sign`	Default: • Nilainya `True` untuk kunci AES. • Nilainya `False` untuk kunci RSA dan EC.	Ya	Ya
`token`	Default: `False`	Tidak	Ya
`trusted`	Default: `False`	Ya	Tidak
`unwrap`	Default: `False`	Ya	Ya

Atribut CloudHSM CLI	Nilai	Dimodifikasi dengan kunci set-atribut	Dapat diatur pada pembuatan kunci
`unwrap-template`	Nilai harus menggunakan templat atribut yang diterapkan untuk setiap kunci terbuka menggunakan kunci pembungkus ini.	Ya	Tidak
`verify`	Default: • Nilainya `True` untuk kunci AES. • Nilainya `False` untuk kunci RSA dan EC.	Ya	Ya
`wrap`	Default: `False`	Ya	Ya
`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

Detail Tambahan

Periksa nilai

Nilai cek 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 CloudHSMmenggunakan 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

- kunci
- Referensi untuk perintah CloudHSM CLI

Migrasi dari Client SDK 3 CMU dan KMU ke Client SDK 5 CloudHSM CLI

Gunakan topik ini untuk memigrasikan alur kerja yang menggunakan alat baris perintah Client SDK 3, CloudHSM Management Utility (CMU) dan Key Management Utility (KMU), untuk menggunakan alat baris perintah Client SDK 5, CloudHSM CLI.

Pada tahunAWS 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. Topik ini memberikan detail khusus untuk migrasi dari Client SDK 3 ke Client SDK 5 untuk alat baris perintah.

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. Antarmuka Baris Perintah CloudHSM (CLI) Operasi manajemen pengguna dapat ditemukan di bawah subperintah pengguna dankuorum. Operasi manajemen kunci dapat ditemukan di bawah subperintah kunci, dan operasi kriptografi dapat ditemukan di bawah subperintah crypto. Lihat Referensi untuk perintah CloudHSM CLI untuk daftar lengkap perintah.

> **Note**
>
> Jika di Client SDK 3 Anda mengandalkan syncKey dan syncUser fungsionalitas untuk sinkronisasi lintas cluster, terus gunakan CMU. CloudHSM CLI di Client SDK 5 saat ini tidak mendukung fungsi ini.

Untuk petunjuk tentang migrasi ke SDK Klien 5, lihat. Migrasi dari SDK Klien 3 ke SDK Klien 5 Untuk manfaat migrasi, lihatManfaat SDK Klien 5.

Konfigurasi lanjutan untuk 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 CLI

Dengan Client SDK 5, Anda dapat mengonfigurasi AWS CloudHSM CLI untuk mengizinkan koneksi ke beberapa klaster CloudHSM dari satu instance CLI.

Gunakan instruksi dalam topik ini untuk menggunakan AWS CloudHSM Command Line Interface (CLI) menggunakan fungsionalitas multi-cluster untuk terhubung dengan beberapa cluster.

Topik

- Prasyarat multi-cluster
- Konfigurasikan CLI untuk fungsionalitas multi-cluster
- konfigurasi-cli add-cluster
- konfigurasi-cli hapus-cluster
- Menggunakan beberapa cluster

Prasyarat multi-cluster

- 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-cluster, Anda harus sudah mengunduh dan menginstal AWS 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 file`cluster-id`. Anda dapat mengkonfigurasi ulang dengan mengikuti `config-cli add-cluster` seperti yang dijelaskan dalam panduan ini.

Konfigurasikan CLI untuk fungsionalitas multi-cluster

Untuk mengonfigurasi AWS CloudHSM CLI Anda untuk fungsionalitas multi-cluster, ikuti langkah-langkah berikut:

1. Identifikasi cluster yang ingin Anda sambungkan.
2. Tambahkan cluster ini ke konfigurasi CLI Anda menggunakan subperintah AWS CloudHSM `add-cluster` configurre-cli seperti yang dijelaskan di bawah ini.
3. Mulai ulang proses AWS CloudHSM CLI apa pun agar konfigurasi baru diterapkan.

konfigurasi-cli add-cluster

Saat menghubungkan ke beberapa cluster, gunakan `configure-cli add-cluster` perintah untuk menambahkan cluster ke konfigurasi Anda.

Sintaks

```
configure-cli add-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [--region <REGION>]
        [--endpoint <ENDPOINT>]
        [--hsm-ca-cert <HSM CA CERTIFICATE FILE>]
        [--server-client-cert-file <CLIENT CERTIFICATE FILE>]
        [--server-client-key-file <CLIENT KEY FILE>]
        [-h, --help]
```

Contoh-contoh

Tambahkan cluster menggunakan **`cluster-id`** parameter

Example

Gunakan `configure-cli add-cluster` bersama dengan `cluster-id` parameter untuk menambahkan cluster (dengan ID dari`cluster-1234567`) ke konfigurasi Anda.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-cli add-cluster --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\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``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

```
C:\Program Files\Amazon\CloudHSM\> .\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”.

Parameter-parameter

--cluster-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 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 ???.

Diperlukan: Ya

--endpoint *<Endpoint>*

Tentukan titik akhir AWS CloudHSM API yang digunakan untuk melakukan `DescribeClusters` panggilan. Anda harus menetapkan pilihan ini dalam kombinasi dengan `--cluster-id`.

Diperlukan: Tidak

-- hsm-ca-cert <HsmCA Certificate Filepath>

Menentukan filepath ke sertifikat HSM CA.

Diperlukan: Tidak

--region *<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`.

Diperlukan: Tidak

-- server-client-cert-file <Client Certificate Filepath>

Jalur ke sertifikat klien yang digunakan untuk autentikasi mutual klien-server TLS.

Hanya gunakan opsi ini jika Anda tidak ingin menggunakan kunci default dan sertifikat SSL/TLS yang kami sertakan dengan Klien SDK 5. Anda harus menetapkan pilihan ini dalam kombinasi dengan `--server-client-key-file`.

Diperlukan: Tidak

-- server-client-key-file <Client Key Filepath>

Jalur ke kunci klien yang digunakan untuk otentikasi timbal balik client-server TLS.

Hanya gunakan opsi ini jika Anda tidak ingin menggunakan kunci default dan sertifikat SSL/TLS yang kami sertakan dengan Klien SDK 5. Anda harus menetapkan pilihan ini dalam kombinasi dengan `--server-client-cert-file`.

Diperlukan: Tidak

konfigurasi-cli hapus-cluster

Saat menghubungkan ke beberapa cluster dengan CLI, gunakan `configure-cli remove-cluster` perintah untuk menghapus cluster dari konfigurasi Anda.

Sintaks

```
configure-cli remove-cluster [OPTIONS]
```

```
        --cluster-id <CLUSTER ID>
        [-h, --help]
```

Contoh-contoh

Hapus cluster menggunakan **`cluster-id`** parameter

Example

Gunakan `configure-cli remove-cluster` bersama dengan `cluster-id` parameter untuk menghapus cluster (dengan ID dari`cluster-1234567`) dari konfigurasi Anda.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-cli remove-cluster --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\configure-cli.exe remove-cluster --cluster-
id cluster-1234567
```

Untuk informasi tentang parameter `--cluster-id`, lihat the section called “Parameter-parameter”.

Parameter

--cluster-id ***<Cluster ID>***

ID cluster untuk menghapus dari konfigurasi.

Diperlukan: Ya

Menggunakan beberapa cluster

Setelah mengkonfigurasi beberapa cluster dengan AWS CloudHSM CLI, gunakan `cloudhsm-cli` perintah untuk berinteraksi dengan mereka.

Contoh-contoh

Mengatur default **`cluster-id`** saat menggunakan mode interaktif

Example

Gunakan ??? bersama dengan `cluster-id` parameter untuk mengatur cluster default (dengan ID dari`cluster-1234567`) dari konfigurasi Anda.

Linux

```
$ cloudhsm-cli interactive --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\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 dari`cluster-1234567`) untuk mendapatkan ??? dari.

Linux

```
$ cloudhsm-cli cluster hsm-info --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\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 CloudHSM Command Line Interface (CLI).

Untuk menjalankan sebagian besar perintah CloudHSM CLI, Anda harus memulai CloudHSM CLI dan masuk ke HSM. Jika Anda menambahkan atau menghapus HSM, perbarui file konfigurasi untuk CloudHSM CLI. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSM di klaster.

Topik berikut menjelaskan perintah di CloudHSM CLI:

Perintah	Deskripsi	Jenis Pengguna
kluster mengaktifkan	Mengaktifkan klaster CloudHSM dan memberikan konfirmasi bahwa klaster tersebut baru. Ini harus dilakukan sebelum operasi lain dapat dilakukan.	Admin yang tidak aktif
klaster hsm-info	Buat daftar HSM di cluster Anda.	Semua [1], termasuk pengguna yang tidak diautentikasi. Login tidak diperlukan.
tanda kripto ecdsa	Menghasilkan tanda tangan menggunakan kunci pribadi EC dan mekanisme penandatanganan ECDSA.	Pengguna kripto (CU)
tanda kripto rsa-pkcs	Menghasilkan tanda tangan menggunakan kunci pribadi RSA dan mekanisme penandatanganan RSA-PKCS.	CU
tanda kripto rsa-pkcs-pss	Menghasilkan tanda tangan menggunakan kunci pribadi RSA dan mekanisme penandatanganan RSA-PKCS-PSS.	CU

Perintah	Deskripsi	Jenis Pengguna
kripto verifikasi 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 menentuka n apakah keduanya terkait secara kriptografi berdasark an kunci publik ecdsa dan mekanisme penandatanganan yang diberikan.	CU
verifikasi kripto rsa-pkcs	Mengonfirmasi file telah ditandatangani di HSM oleh kunci publik yang diberikan . Verifikasi tanda tangan dihasilkan menggunakan mekanisme penandatanganan RSA-PKCS. Membandin gkan file yang ditandata ngani dengan file sumber dan menentukan apakah keduanya terkait secara kriptografi berdasarkan kunci publik rsa dan mekanisme penandata nganan yang diberikan.	CU

Perintah	Deskripsi	Jenis Pengguna
verifikasi kripto rsa-pkcs-pss	Mengonfirmasi file telah ditandatangani di HSM oleh kunci publik yang diberikan . Verifikasi tanda tangan dihasilkan menggunakan mekanisme penandatanganan RSA-PKCS-PSS. 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
file hasilkan 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 menghasilkan rahasia-generik simetris	Menghasilkan kunci Rahasia Generik simetris di cluster Anda AWS CloudHSM .	CU

Perintah	Deskripsi	Jenis Pengguna
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-atribut	Menetapkan atribut kunci di AWS CloudHSM cluster Anda.	CU dapat menjalankan perintah ini, admin dapat mengatur atribut tepercaya.
pembagian kunci	Berbagi kunci dengan CU lain di AWS CloudHSM cluster Anda.	CU
kunci unshare	Membatalkan kunci dengan CU lain di AWS CloudHSM cluster Anda.	CU
buka kunci aes-gcm	Membuka kunci payload ke dalam cluster menggunakan kunci pembungkus AES dan mekanisme unwrapping AES-GCM.	CU

Perintah	Deskripsi	Jenis Pengguna
buka kunci aes-no-pad	Membuka kunci payload ke dalam cluster menggunakan kunci pembungkus AES dan mekanisme unwrapping AES-NO-PAD.	CU
buka kunci aes-pkcs5-pad	Membuka kunci payload menggunakan kunci pembungkus AES dan mekanisme unwrapping AES-PKCS5-PAD.	CU
buka kunci aes-zero-pad	Membuka kunci payload ke dalam cluster menggunakan kunci pembungkus AES dan mekanisme unwrapping AES-ZERO-PAD.	CU
buka kunci cloudhsm-aes-gcm	Membuka kunci payload ke dalam cluster menggunak an kunci pembungkus AES dan mekanisme unwrapping CLOUDHSM-AES-GCM.	CU
kunci buka rsa-aes	Membuka kunci payload menggunakan kunci pribadi RSA dan mekanisme unwrapping RSA-AES.	CU
kunci buka rsa-oaep	Membuka kunci payload menggunakan kunci pribadi RSA dan mekanisme unwrapping RSA-OAEP.	CU

Perintah	Deskripsi	Jenis Pengguna
buka kunci rsa-pkcs	Membuka kunci payload menggunakan kunci pribadi RSA dan mekanisme unwrapping RSA-PKCS.	CU
bungkus kunci aes-gcm	Membungkus kunci payload menggunakan kunci AES pada HSM dan mekanisme pembungkus AES-GCM.	CU
bungkus kunci aes-no-pad	Membungkus kunci payload menggunakan kunci AES pada HSM dan mekanisme pembungkus AES-NO-PAD.	CU
bungkus kunci aes-pkcs5-pad	Membungkus kunci payload menggunakan kunci AES pada HSM dan mekanisme pembungkus AES-PKCS5-PAD.	CU
bungkus kunci aes-zero-pad	Membungkus kunci payload menggunakan kunci AES pada HSM dan mekanisme pembungkus AES-ZERO-PAD.	CU
bungkus kunci cloudhsm-aes-gcm	Membungkus kunci payload menggunakan kunci AES pada HSM dan mekanisme pembungkus CLOUDHSM-AES-GCM.	CU

Perintah	Deskripsi	Jenis Pengguna
bungkus kunci rsa-aes	Membungkus kunci payload menggunakan kunci publik RSA pada HSM dan mekanisme pembungkus RSA-AES.	CU
bungkus kunci rsa-oaep	Membungkus kunci payload menggunakan kunci publik RSA pada HSM dan mekanisme pembungkus RSA-OAEP.	CU

Perintah	Deskripsi	Jenis Pengguna
key wrap rsa-pkcsPerintah membungkus kunci payload menggunakan kunci publik RSA pada HSM dan mekanisme pembungku s. `RSA-PKCS` `extractable` Atribut kunci payload harus disetel ke `true`. Hanya pemilik kunci, yaitu pengguna kripto (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 keypair RSA menggunakan kunci generate-asymmetric-pair perintah dan `wrap` atribut yang disetel ke. `true` Jenis pengguna Jenis pengguna berikut dapat menjalankan perintah ini. • Pengguna Crypto (CU) Persyaratan • Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.	Membungkus kunci payload menggunakan kunci publik RSA pada HSM dan mekanisme pembungkus RSA-PKCS.	CU

Perintah	Deskripsi	Jenis Pengguna
Login	Masuk ke AWS CloudHSM cluster Anda.	Admin, pengguna kripto (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 klaster CloudHSM Anda.	Semua [1], termasuk pengguna yang tidak diautentikasi. Login tidak diperlukan.
tanda token kuorum list-quorum-values	Daftar nilai kuorum yang ditetapkan dalam klaster CloudHSM Anda.	Semua [1], termasuk pengguna yang tidak diautentikasi. Login tidak diperlukan.
batas waktu daftar tanda token kuorum	Memperoleh periode batas waktu token dalam hitungan detik untuk semua jenis token.	Admin dan pengguna kripto
tanda token kuorum set-quorum-value	Menetapkan nilai kuorum baru untuk layanan resmi kuorum.	Admin
tanda token kuorum set-timeout	Menetapkan periode batas waktu token dalam hitungan detik untuk setiap jenis token.	Admin
perubahan pengguna-mfa	Mengubah strategi otentikasi multi-faktor (MFA) pengguna.	Admin, CU

Perintah	Deskripsi	Jenis Pengguna
perubahan kata sandi pengguna	Mengubah kata sandi pengguna pada HSM. 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
menghapus pengguna	Menghapus pengguna di AWS CloudHSM klaster Anda.	Admin
daftar pengguna	Daftar pengguna di AWS CloudHSM klaster 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.

klaster

clusteradalah kategori induk untuk sekelompok perintah yang, ketika dikombinasikan dengan kategori induk, membuat perintah khusus untuk pengguna. Saat ini, kategori pengguna terdiri dari perintah berikut:

- kluster mengaktifkan
- klaster hsm-info

kluster mengaktifkan

Gunakan cluster activate perintah di CloudHSM CLI untuk mengaktifkan cluster baru. 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

Sintaks

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

klaster hsm-info

Gunakan cluster hsm-info perintah di CloudHSM CLI untuk membuat daftar HSM di cluster Anda. Anda tidak perlu login ke CloudHSM CLI untuk menjalankan perintah ini.

> **Note**
>
> Jika Anda menambahkan atau menghapus HSM, perbarui file konfigurasi yang digunakan AWS CloudHSM klien dan alat baris perintah. Jika tidak, perubahan yang Anda buat mungkin tidak efektif pada semua HSM di klaster.

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 HSM yang ada di AWS CloudHSM cluster Anda.

```
aws-cloudhsm > cluster hsm-info
{
  "error_code": 0,
  "data": {
    "hsms": [
      {
        "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.
- H ardware-version-major: Versi perangkat keras utama.
- H ardware-version-minor: Versi perangkat keras minor.
- F irmware-version-major: Versi firmware utama.
- F irmware-version-minor: Versi firmware minor.
- F irmware-build-number: Nomor build firmware.
- Firmware-ID: ID firmware, yang mencakup versi mayor dan minor bersama dengan build.

Topik terkait

- kluster mengaktifkan

kripto

cryptoadalah 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 kripto
  - tanda kripto ecdsa
  - tanda kripto rsa-pkcs

  - tanda kripto rsa-pkcs-pss
- verifikasi kripto
  - kripto verifikasi ecdsa
  - verifikasi kripto rsa-pkcs
  - verifikasi kripto rsa-pkcs-pss

tanda kripto

crypto signadalah kategori induk untuk sekelompok perintah yang, bila digabungkan dengan kategori induk, menggunakan kunci pribadi yang dipilih di AWS CloudHSM klaster Anda untuk menghasilkan tanda tangan. crypto signmemiliki subperintah berikut:

- tanda kripto ecdsa
- tanda kripto rsa-pkcs
- tanda kripto rsa-pkcs-pss

Untuk menggunakannyacrypto 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

tanda kripto ecdsa

crypto sign ecdsaPerintah 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 kunci generate-asymmetric-pair ec perintah dengan `sign` atribut yang disetel ke`true`.

> **Note**
>
> Tanda tangan dapat diverifikasi AWS CloudHSM dengan verifikasi kripto subperintah.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

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
  -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": "0x0000000000007808dd",
    "signature": "4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2wOS3Q5bzk
+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": "0x0000000000007808dd",
    "signature": "4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2wOS3Q5bzk
+Hc5irV5iTkuxQbropPttVFZ8V6FgR2fz+sPegwCw=="
  }
}
```

Argumen

<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 yang valid:

- sha1
- sha224
- sha256
- sha384
- sha512

Diperlukan: Ya

<KEY_FILTER>

Referensi kunci (misalnya kunci-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.

Diperlukan: Ya

Topik terkait

- tanda kripto
- verifikasi kripto

tanda kripto rsa-pkcs

crypto sign rsa-pkcsPerintah 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 kunci generate-asymmetric-pair rsa perintah dengan `sign` atribut yang disetel `ketrue`.

> **Note**
>
> Tanda tangan dapat diverifikasi AWS CloudHSM dengan verifikasi kripto subperintah.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

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
  -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": "0x00000000007008db",
    "signature": "XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+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": "0x00000000007008db",
    "signature": "XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ=="
  }
}
```

Argumen

<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 yang valid:

- sha1
- sha224
- sha256
- sha384
- sha512

Diperlukan: 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.

Diperlukan: Ya

Topik terkait

- tanda kripto
- verifikasi kripto

tanda kripto rsa-pkcs-pss

crypto sign rsa-pkcs-pssPerintah menghasilkan tanda tangan menggunakan kunci pribadi RSA dan mekanisme `RSA-PKCS-PSS` penandatanganan.

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 kunci generate-asymmetric-pair rsa perintah dengan `sign` atribut yang disetel `ketrue`.

> **Note**
>
> Tanda tangan dapat diverifikasi AWS CloudHSM dengan verifikasi kripto subperintah.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

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:
      --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
      --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 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": "0x00000000007008db",
    "signature": "H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg=="
  }
}
```

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": "0x00000000007008db",
    "signature": "H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg=="
  }
}
```


Argumen

<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 yang valid:

- sha1
- sha224
- sha256
- sha384
- sha512

Diperlukan: 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.

Diperlukan: Ya

<MGF>

Menentukan fungsi pembuatan topeng.

> **Note**
>
> Fungsi hash fungsi pembuatan topeng harus sesuai dengan fungsi hash mekanisme penandatanganan.

Nilai yang valid:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Diperlukan: Ya

<SALT_LENGTH>

Menentukan panjang garam.

Diperlukan: Ya

Topik terkait

- tanda kripto
- verifikasi kripto

Topik terkait

- verifikasi kripto

verifikasi kripto

crypto verifyadalah kategori induk untuk sekelompok perintah yang, bila dikombinasikan dengan kategori induk, mengonfirmasi apakah file telah ditandatangani oleh kunci yang diberikan. crypto verifymemiliki subperintah berikut:

- kripto verifikasi ecdsa
- verifikasi kripto rsa-pkcs
- verifikasi kripto rsa-pkcs-pss

crypto verifyPerintah 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 operasi tanda kripto.

kripto verifikasi ecdsa

crypto verify ecdsaPerintah ini digunakan 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.

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 kunci impor pem perintah dengan `verify` atribut yang disetel `ketrue`.

> **Note**
>
> Anda dapat menghasilkan tanda tangan di CloudHSM CLI tanda kripto dengan subperintah.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

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
  -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/KqoQvh4ueMAxQQVp7FQguZ2wOS3Q5bzk+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
 +ogk7M7S3iTqFg3SndJfd91dZFr5Qo6YixJl8JwcvqqVgsVuO6o+VKvTRjz0/V05kf3JJbBLr87Q
+wLWcMAJfA==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

Argumen

<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 yang valid:

- sha1
- sha224
- sha256
- sha384
- sha512

Diperlukan: 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.

Diperlukan: 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)

Topik terkait

- tanda kripto
- verifikasi kripto

verifikasi kripto rsa-pkcs

crypto verify rsa-pkcsPerintah ini digunakan untuk 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. tanda kripto

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

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
  -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
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+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
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

Argumen

<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 yang valid:

- sha1
- sha224
- sha256
- sha384
- sha512

Diperlukan: 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.

Diperlukan: 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)

Topik terkait

- tanda kripto
- verifikasi kripto

verifikasi kripto rsa-pkcs-pss

crypto sign rsa-pkcs-pssPerintah ini digunakan untuk menyelesaikan operasi berikut.

- Konfirmasikan file telah ditandatangani di HSM dengan kunci publik yang diberikan.

- Verifikasi tanda tangan dihasilkan menggunakan mekanisme penandatanganan RSA-PKCS-PSS.
- 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. tanda kripto

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

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
      --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 penandatanganan RSA-PKCS-PSS 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-pss --key-filter attr.label=rsa-public
 --hash-function sha256 --data YWJjMTIz --salt-length 10 --mgf mgf1-sha256
 --signature H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg==
{
  "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 RSAPKCSPSS 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/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

Argumen

<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 yang valid:

- sha1
- sha224
- sha256
- sha384
- sha512

Diperlukan: 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.

Diperlukan: Ya

<MFG>

Menentukan fungsi pembuatan topeng.

> **Note**
>
> Fungsi hash fungsi pembuatan topeng harus sesuai dengan fungsi hash mekanisme penandatanganan.

Nilai yang valid:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Diperlukan: 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)

Topik terkait

- tanda kripto
- verifikasi kripto

kunci

keyadalah kategori induk untuk sekelompok perintah yang, bila dikombinasikan dengan kategori induk, membuat perintah khusus untuk kunci. Saat ini, kategori ini terdiri dari perintah berikut:

- hapus kunci
- file hasilkan kunci
- kunci generate-asymmetric-pair
  - kunci generate-asymmetric-pair rsa
  - kunci generate-asymmetric-pair ec
- kunci menghasilkan-simetris
  - kunci menghasilkan aes simetris

    - kunci generik-rahasia generik-simetris
- kunci impor pem
- daftar kunci
- replikasi kunci
- kunci set-atribut
- pembagian kunci
- kunci unshare
- buka kunci
- bungkus kunci

hapus kunci

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 (CU)

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"
  }
}
```

Argumen

<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

Diperlukan: Ya

Topik terkait

- daftar kunci
- file hasilkan kunci

- kunci unshare
- Atribut kunci untuk CloudHSM CLI
- Menggunakan CloudHSM CLI untuk memfilter kunci

file hasilkan kunci

key generate-filePerintah mengekspor kunci asimetris dari 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 pembongkaran SSL/TLS 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 (CU)

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"
  }
}
```

Argumen

<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

Diperlukan: Tidak

<ENCODING>

Menentukan format encoding untuk file kunci

Diperlukan: Ya

<PATH>

Menentukan path file di mana file kunci akan ditulis

Diperlukan: Ya

Topik terkait

- Atribut kunci untuk CloudHSM CLI
- Menggunakan CloudHSM CLI untuk memfilter kunci
- kunci generate-asymmetric-pair
- kunci menghasilkan-simetris

kunci generate-asymmetric-pair

key generate-asymmetric-pairadalah 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

kunci generate-asymmetric-pair ec

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 (CU)

Persyaratan

Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaks

```
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]
      --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
  -h, --help
          Print help
```

Contoh-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": "0x000000000012000b",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "ec",
        "label": "ec-public-key-example",
        "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":
 "0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f38a
        "curve": "secp224r1"
      }
    },
"private_key": {
      "key-reference": "0x000000000012000c",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "ec",
        "label": "ec-private-key-example",
        "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": 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 token=true encrypt=true \
    --private-attributes token=true decrypt=true
 {
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x0000000000002806eb",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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": [],
        "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"
      }
    }
  }
}
```

Argumen

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah dikonfigurasi.

<CURVE>

Menentukan pengenal untuk kurva elips.

- primer256v1
- secp256r1
- secp224r1
- secp384r1
- secp256k1
- secp521r1

Diperlukan: 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,`token=true`)

Untuk daftar atribut kunci yang didukung, lihatAtribut kunci untuk CloudHSM CLI.

Diperlukan: 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.

Diperlukan: 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,`token=true`)

Untuk daftar atribut kunci yang didukung, lihatAtribut kunci untuk CloudHSM CLI.

Diperlukan: 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.

Diperlukan: 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).

Diperlukan: Tidak

Topik terkait

- Atribut kunci untuk CloudHSM CLI
- Menggunakan CloudHSM CLI untuk memfilter kunci

kunci generate-asymmetric-pair rsa

Gunakan key generate-asymmetric-pair rsa perintah menghasilkan key pair RSA asimetris di cluster Anda AWS CloudHSM .

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

Persyaratan

Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaks

```
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
  -h, --help
          Print help
```

Contoh-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": "0x0000000000160010",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-public-key-example",
        "id": "",
```

```
        "check-value": "0x498e1f",
        "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": 512,
        "public-exponent": "0x010001",
        "modulus":
 "0xdfca0669dc8288ed3bad99509bd21c7e6192661407021b3f4cdf4a593d939dd24f4d641af8e4e73b04c847731c6
e89a065e7d1a46ced96b46b909db2ab6be871ee700fd0a448b6e975bb64cae77c49008749212463e37a577baa57ce3e
bcebb7d20bd6df1948ae336ae23b52d73b7f3b6acc2543edb6358e08d326d280ce489571f4d34e316a2ea1904d513ca
        "modulus-size-bits": 2048
      }
    },
"private_key": {
      "key-reference": "0x0000000000160011",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-private-key-example",
```

```
        "id": "",
        "check-value": "0x498e1f",
        "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": 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 token=true encrypt=true \
--private-attributes token=true decrypt=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": "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": "0x0000000000280cc7",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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
      }
    }
  }
}
```

Argumen

<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.

Diperlukan: 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,) `token=true`

Untuk daftar atribut kunci yang didukung, lihatAtribut kunci untuk CloudHSM CLI.

Diperlukan: 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.

Diperlukan: Ya

<PUBLIC_EXPONENT>

Menentukan eksponen publik. Nilai harus angka ganjil yang lebih besar dari atau sama dengan 65537.

Diperlukan: 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,) `token=true`

Untuk daftar atribut kunci yang didukung, lihatAtribut kunci untuk CloudHSM CLI.

Diperlukan: 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.

Diperlukan: 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).

Diperlukan: Tidak

Topik terkait

- Atribut kunci untuk CloudHSM CLI
- Menggunakan CloudHSM CLI untuk memfilter kunci

kunci menghasilkan-simetris

key generate-symmetricadalah 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-simetris aes
- kunci menghasilkan-simetris generik-rahasia

kunci menghasilkan aes simetris

key generate-symmetric aesPerintah menghasilkan kunci AES simetris di cluster Anda AWS CloudHSM .

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

Persyaratan

Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaks

```
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
  -h, --help
          Print help
```

Contoh-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": "0x00000000002e06bf",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "aes",
        "label": "example-aes",
        "id": "",
        "check-value": "0x9b94bd",
        "class": "secret-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": 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": "0x00000000002e06bf",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "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
      }
    }
  }
}
```

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,`token=true`).

Untuk daftar atribut kunci yang didukung, lihatAtribut kunci untuk CloudHSM CLI.

Diperlukan: Tidak

<KEY-LENGTH-BYTES>

Menentukan panjang kunci dalam byte.

Nilai yang valid:

- 16, 24, dan 32

Diperlukan: 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.

Diperlukan: 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).

Diperlukan: Tidak

Topik terkait

- Atribut kunci untuk CloudHSM CLI
- Menggunakan CloudHSM CLI untuk memfilter kunci

kunci menghasilkan rahasia-generik simetris

key generate-asymmetric-pairPerintah menghasilkan kunci Rahasia Generik simetris di klaster AWS CloudHSM Anda.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

Persyaratan

Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaks

```
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
  -h, --help
          Print help
```

Contoh-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": "0x00000000002e08fd",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "generic-secret",
        "label": "example-generic-secret",
        "id": "",
        "class": "secret-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": 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 token=true encrypt=true
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000002e08fd",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "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 dari atribut kunci untuk mengatur kunci AES yang dihasilkan dalam bentuk `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (misalnya,`token=true`).

Untuk daftar atribut kunci yang didukung, lihatAtribut kunci untuk CloudHSM CLI.

Diperlukan: Tidak

<KEY-LENGTH-BYTES>

Menentukan panjang kunci dalam byte.

Nilai yang valid:

- 1 hingga 800

Diperlukan: Ya

<LABEL>

Menentukan label yang ditetapkan pengguna untuk kunci rahasia generik. Ukuran maksimum yang diijinkan `label` adalah 127 karakter untuk Client SDK 5.11 dan setelahnya. Klien SDK 5.10 dan sebelumnya memiliki batas 126 karakter.

Diperlukan: 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).

Diperlukan: Tidak

Topik terkait

- Atribut kunci untuk CloudHSM CLI
- Menggunakan CloudHSM CLI untuk memfilter kunci

kunci impor pem

key import pemPerintah dalam AWS CloudHSM mengimpor kunci format PEM ke HSM. Anda dapat menggunakannya untuk mengimpor kunci privat yang dihasilkan di luar HSM.

> **Note**
>
> Gunakan file hasilkan kunci 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 (CU)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaks

```
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

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": "0x00000000001e08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
```

```
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "example-imported-key",
        "id": "0x",
        "check-value": "0x99fe93",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "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":
 "0x8e9c172c37aa22ed1ce25f7c3a7c936dadc532201400128b044ebb4b96#··3e4930ab910df5a2896eaeb8853cfe
        "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": "0x00000000001e08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "example-imported-key-with-attributes",
        "id": "0x",
        "check-value": "0x99fe93",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "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":
 "0x8e9c172c37aa22ed1ce25f7c3a7c936dadc532201400128b044ebb4b96#··3e4930ab910df5a2896eaeb8853cfe
        "modulus-size-bits": 2048
      }
    },
    "message": "Successfully imported key"
  }
}
```

Argumen

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah dikonfigurasi.

<PATH>

Menentukan path file di mana file kunci berada.

Diperlukan: Ya

<LABEL>

Menentukan label yang ditetapkan pengguna untuk kunci yang diimpor. Ukuran maksimum yang diijinkan `label` adalah 126 karakter.

Diperlukan: Ya

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus.

Kemungkinan nilai:

- ec-publik
- rsa-publik

Diperlukan: Ya

<IMPORT_KEY_ATTRIBUTES>

Menentukan daftar spasi dipisahkan atribut kunci untuk mengatur untuk kunci diimpor dalam bentuk `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (misalnya,`token=true`). Untuk daftar atribut kunci yang didukung, lihatAtribut kunci untuk CloudHSM CLI.

Diperlukan: Tidak

Topik terkait

- tanda kripto
- verifikasi kripto

daftar kunci

key listPerintah menemukan semua kunci untuk pengguna saat ini yang ada di AWS CloudHSM cluster Anda. Outputnya mencakup kunci yang dimiliki dan dibagikan pengguna, serta semua kunci publik di klaster CloudHSM.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

Sintaks

```
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
 -h, --help
        Print help
```

Contoh-contoh

Contoh berikut menunjukkan berbagai cara Anda menjalankan key list perintah.

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 yang 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": "0x00000000000003d5",
        "attributes": {
          "label": "test_label_1"
        }
      },
      {
        "key-reference": "0x0000000000000626",
        "attributes": {
          "label": "test_label_2"
        }
      },.
      ...8 keys later...
    ],
    "total_key_count": 56,
```

```
    "returned_key_count": 10,
    "next_token": "10"
  }
}
```

Example Contoh: Temukan semua kunci - bertele-tele

Outputnya mencakup kunci yang dimiliki dan dibagikan pengguna, serta semua kunci publik int HSM.

> **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": "0x000000000012000c",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "session"
        },
        "attributes": {
          "key-type": "ec",
          "label": "ec-test-private-key",
          "id": "",
          "check-value": "0x2a737d",
          "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": false,
          "trusted": false,
          "unwrap": false,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 122,
          "ec-point":
"0x0442d53274a6c0ec1a23c165dcb9ccdd72c64e98ae1a9594bb5284e752c746280667e11f1e983493c1c605e0a80
          "curve": "secp224r1"
        }
      },
      {
        "key-reference": "0x000000000012000d",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "session"
        },
        "attributes": {
          "key-type": "ec",
          "label": "ec-test-public-key",
          "id": "",
          "check-value": "0x2a737d",
          "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":
 "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": "0x0000000000000030",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
```

```
            }
          ],
          "shared-users": [],
          "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": "0x0000000000000042",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "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": "0x0000000000000081",
```

```
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "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": "0x0000000000000089",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
```

```
            }
          ],
          "shared-users": [],
          "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. Menggunakan CloudHSM CLI untuk memfilter kunci

Argumen

<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

Diperlukan: 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.

Diperlukan: Tidak

<STARTING_TOKEN>

Token untuk menentukan di mana harus memulai paginating. Ini adalah token berikutnya dari respons yang sebelumnya terpotong.

Diperlukan: 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.

Diperlukan: Tidak

Topik terkait

- hapus kunci
- file hasilkan kunci

- kunci unshare
- Atribut kunci untuk CloudHSM CLI
- Menggunakan CloudHSM CLI untuk memfilter kunci

replikasi kunci

key replicatePerintah mereplikasi kunci dari AWS CloudHSM cluster sumber ke AWS CloudHSM cluster tujuan.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

> **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 CU pada 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=crypto-user
```

```
$ export DESTINATION_CLOUDHSM_PIN=username:password
```

- Dalam mode interaktif, pengguna harus secara eksplisit masuk ke cluster sumber dan tujuan.

Sintaks

```
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-contoh

Example Contoh: Kunci replikasi

Perintah ini mereplikasi kunci dari cluster sumber dengan ke klaster tujuan kloning.

```
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": "0x0000000000300006",
      "key-info": {
        "key-owners": [
          {
            "username": "crypto-user-1",
            "key-coverage": "full"
```

```
          }
        ],
        "shared-users": [],
        "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"
  }
}
```

Argumen

<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

Diperlukan: Ya

<SOURCE_CLUSTER_ID>

ID cluster sumber.

Diperlukan: Ya

<DESTINATION_CLUSTER_ID>

ID cluster tujuan.

Diperlukan: Ya

Topik terkait

- Menghubungkan ke beberapa cluster dengan CLI

kunci set-atribut

Gunakan key set-attribute perintah untuk mengatur atribut kunci di AWS CloudHSM cluster Anda. 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 (CU) 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.

Sintaks

```
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
  -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 labelmy_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": "0x00000000006400ec",
```

```
        "key-info": {
          "key-owners": [
            {
              "username": "bob",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "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
  }
}
```

Argumen

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah dikonfigurasi.

<KEY_ATTRIBUTE>

Menentukan nama atribut kunci.

Diperlukan: 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

Diperlukan: Tidak

<KEY_ATTRIBUTE_VALUE>

Menentukan nilai atribut kunci.

Diperlukan: Ya

<KEY_REFERENCE>

Representasi heksadesimal atau desimal dari kunci. (seperti pegangan kunci).

Diperlukan: Tidak

Topik terkait

- Menggunakan CloudHSM CLI untuk memfilter kunci
- Atribut kunci untuk CloudHSM CLI

pembagian kunci

key sharePerintah membagikan kunci dengan CU lain di AWS CloudHSM cluster Anda.

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 (CU)

Persyaratan

Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaks

```
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
```

```
  -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 CU`alice`.

Example

1. Jalankan key share perintah untuk berbagi kunci dengan`alice`.

```
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": "0x00000000001c0686",
        "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"
            }
          ],
          "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`

Argumen

<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, lihatAtribut kunci untuk CloudHSM CLI.

Diperlukan: 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.

Diperlukan: 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 Memahami pengguna HSM.

Diperlukan: Ya

Topik terkait

- Menggunakan CloudHSM CLI untuk memfilter kunci
- Atribut kunci untuk CloudHSM CLI

kunci unshare

key unsharePerintah membatalkan pembagian kunci dengan CU lain di AWS CloudHSM cluster Anda.

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 pembagian kunci. Selain itu, pengguna ini tidak dapat mengubah atribut kunci.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

Persyaratan

Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaks

```
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

  -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 CU`alice`.

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": "0x00000000001c0686",
      "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"
          }
        ],
        "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. 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": "0x00000000001c0686",
        "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"
            },
          ],
          "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
  }
}
```

Argumen

<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, lihatAtribut kunci untuk CloudHSM CLI.

Diperlukan: 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.

Diperlukan: 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 Memahami pengguna HSM.

Diperlukan: Ya

Topik terkait

- Menggunakan CloudHSM CLI untuk memfilter kunci
- Atribut kunci untuk CloudHSM CLI

buka kunci

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 bungkus kunci 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.

- buka kunci aes-gcm
- buka kunci aes-no-pad
- buka kunci aes-pkcs5-pad
- buka kunci aes-zero-pad
- buka kunci cloudhsm-aes-gcm
- kunci buka rsa-aes
- kunci buka rsa-oaep
- buka kunci rsa-pkcs

buka kunci aes-gcm

key unwrap aes-gcmPerintah 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 (CU)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaks

```
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
      --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
  -h, --help
          Print help
```

Contoh-contoh

Contoh-contoh ini menunjukkan cara menggunakan key unwrap aes-gcm perintah menggunakan kunci AES dengan nilai `unwrap` atribut yang disetel `ketrue`.

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 xvslgrtg8kHzrvekny97tLSIeokpPwV8
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e4",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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": "0x00000000001808e4",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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
      }
    }
  }
}
```

Argumen

<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.

Diperlukan: 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.

Diperlukan: Tidak

<AAD>

Nilai Aes GCM Additional Authenticated Data (AAD), dalam hex.

Diperlukan: Tidak

<TAG_LENGTH_BITS>

Panjang tag Aes GCM dalam bit.

Diperlukan: Ya

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus [nilai yang mungkin:`aes`,`des3`,`ec-private`,`generic-secret`,`rsa-private`].

Diperlukan: Ya

<LABEL>

Label untuk kunci yang tidak dibungkus.

Diperlukan: Ya

<SESSION>

Membuat kunci sesi yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Diperlukan: Tidak

<IV>

Nilai awal yang digunakan untuk membungkus kunci, dalam hex.

Diperlukan: Tidak

Topik terkait

- bungkus kunci
- buka kunci

buka kunci aes-no-pad

key unwrap aes-no-padPerintah membuka kunci payload ke 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 (CU)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaks

```
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
     --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
 -h, --help
          Print help
```

Contoh-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 eXK3PMAOnKM9y3YX6brbhtMoC060EOH9
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08ec",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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": "0x00000000001c08ec",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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
      }
    }
  }
}
```

Argumen

<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.

Diperlukan: 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.

Diperlukan: Tidak

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus [nilai yang mungkin:`aes`,`des3`,`ec-private`,`generic-secret`,`rsa-private`].

Diperlukan: Ya

<LABEL>

Label untuk kunci yang tidak dibungkus.

Diperlukan: Ya

<SESSION>

Membuat kunci sesi yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Diperlukan: Tidak

Topik terkait

- bungkus kunci
- buka kunci

buka kunci aes-pkcs5-pad

key unwrap aes-pkcs5-padPerintah 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 (CU)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaks

```
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
      --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
  -h, --help
          Print help
```

Contoh-contoh

Contoh-contoh ini menunjukkan cara menggunakan key unwrap aes-pkcs5-pad perintah menggunakan kunci AES dengan nilai unwrap atribut yang disetel ketrue.

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 MbuYNresfOKyGNnxKWen88nSfX+uUE/0qmGofSisicY=
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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": "0x00000000001c08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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
      }
    }
  }
}
```

Argumen

<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.

Diperlukan: 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.

Diperlukan: Tidak

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus [nilai yang mungkin:`aes`,`des3`,`ec-private`,`generic-secret`,`rsa-private`].

Diperlukan: Ya

<LABEL>

Label untuk kunci yang tidak dibungkus.

Diperlukan: Ya

<SESSION>

Membuat kunci sesi yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Diperlukan: Tidak

Topik terkait

- bungkus kunci
- buka kunci

buka kunci aes-zero-pad

key unwrap aes-zero-padPerintah membuka kunci payload ke 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 (CU)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaks

```
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
      --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
  -h, --help
          Print help
```

Contoh-contoh

Contoh-contoh ini menunjukkan cara menggunakan key unwrap aes-zero-pad perintah menggunakan kunci AES dengan nilai `unwrap` atribut yang disetel `ketrue`.

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 L1wVlL/YeBNVAw6Mpk3owFJZXBzDLONt
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08e7",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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": "0x00000000001c08e7",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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
      }
    }
  }
}
```

Argumen

<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.

Diperlukan: 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.

Diperlukan: Tidak

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus [nilai yang mungkin:`aes`,`des3`,`ec-private`,`generic-secret`,`rsa-private`].

Diperlukan: Ya

<LABEL>

Label untuk kunci yang tidak dibungkus.

Diperlukan: Ya

<SESSION>

Membuat kunci sesi yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Diperlukan: Tidak

Topik terkait

- bungkus kunci
- buka kunci

buka kunci cloudhsm-aes-gcm

key unwrap cloudhsm-aes-gcmPerintah membuka kunci payload ke 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 (CU)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaks

```
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
      --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
  -h, --help
          Print help
```

Contoh-contoh

Contoh-contoh ini menunjukkan cara menggunakan key unwrap cloudhsm-aes-gcm perintah menggunakan kunci AES dengan nilai `unwrap` atribut yang disetel `ketrue`.

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
 6Rn8nkjEriDYlnP3P8nPkYQ8hplOEJ899zsrF+aTB0i/fIlZ
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001408e8",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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": "0x00000000001408e8",
      "key-info": {
        "key-owners": [
          {
```

```
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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
      }
    }
  }
}
```

Argumen

<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.

Diperlukan: 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.

Diperlukan: Tidak

<AAD>

Nilai Aes GCM Additional Authenticated Data (AAD), dalam hex.

Diperlukan: Tidak

<TAG_LENGTH_BITS>

Panjang tag Aes GCM dalam bit.

Diperlukan: Ya

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus [nilai yang mungkin:`aes`,`des3`,`ec-private`,`generic-secret`,`rsa-private`].

Diperlukan: Ya

<LABEL>

Label untuk kunci yang tidak dibungkus.

Diperlukan: Ya

<SESSION>

Membuat kunci sesi yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Diperlukan: Tidak

Topik terkait

- bungkus kunci
- buka kunci

kunci buka rsa-aes

key unwrap rsa-aesPerintah 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 menggunakankey 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`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

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
      --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
  -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 `ketrue`.

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
+NcerbAJO8DJ1lXPygZcI21vIFSZJuWMEiWpe1R9D/5WSYgxLVKex30xCFqebtEzxbKuv4DOmU4meSofqREYvtb3EoIKwjy
+RL5WGXKe4nAboAkC5GO7veI5yHL1SaKlssSJtTL/CFpbSLsAFuYbv/NUCWwMY5mwyVTCSlw+HlgKK
+5TH1MzBaSi8fpfyepLT8sHy2Q/VRl6ifb49p6m0KQFbRVvz/OWUd6l4d97BdgtaEz6ueg==
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e2",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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": "0x00000000001808e2",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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
      }
    }
  }
}
```

Argumen

<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.

Diperlukan: 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.

Diperlukan: Tidak

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus [nilai yang mungkin:`aes`,`des3`,`ec-private`,`generic-secret`,`rsa-private`].

Diperlukan: Ya

<HASH_FUNCTION>

Menentukan fungsi hash.

Nilai yang valid:

- sha1
- sha224
- sha256
- sha384
- sha512

Diperlukan: Ya

<MGF>

Menentukan fungsi pembuatan topeng.

> **Note**
>
> Fungsi hash fungsi pembuatan topeng harus sesuai dengan fungsi hash mekanisme penandatanganan.

Nilai yang valid:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256

- mgf1-sha384
- mgf1-sha512

Diperlukan: Ya

<LABEL>

Label untuk kunci yang tidak dibungkus.

Diperlukan: Ya

<SESSION>

Membuat kunci sesi yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Diperlukan: Tidak

Topik terkait

- bungkus kunci
- buka kunci

kunci buka rsa-oaep

key unwrap rsa-oaepPerintah 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 (CU)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaks

```
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
      --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
  -h, --help
          Print help
```

Contoh-contoh

Contoh-contoh ini menunjukkan bagaimana menggunakan key unwrap rsa-oaep perintah menggunakan kunci pribadi RSA dengan nilai `unwrap` atribut yang disetel `ketrue`.

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
 OjJe4msobPLz9TuSAdULEu17T5rMDWtSlLyBSkLbaZnYzzpdrhsbGLbwZJCtB/jGkDNdB4qyTAOQwEpggGf6v
+Yx6JcesNeKKNU8XZal/YBoHC8noTGUSDI2qr+u2tDc84NPv6d+F2KOONXsSxMhmxzzNG/
gzTVIJhOuy/B1yHjGP4mOXoDZf5+7f5M1CjxBmz4Vva/wrWHGCSG0yOaWblEvOiHAIt3UBdyKmU+/
My4xjfJv7WGGu3DFUUIZ06TihRtKQhUYU1M9u6NPf9riJJfHsk6QCuSZ9yWThDT9as6i7e3htnyDhIhGWaoK8JU855cN/
YNKAUqkNpC4FPL3iw==
{
  "data": {
    "key": {
      "key-reference": "0x00000000001808e9",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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": "0x00000000001808e9",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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
      }
    }
  }
}
```

Argumen

<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.

Diperlukan: 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.

Diperlukan: Tidak

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus [nilai yang mungkin:`aes`,`des3`,`ec-private`,`generic-secret`,`rsa-private`].

Diperlukan: Ya

<HASH_FUNCTION>

Menentukan fungsi hash.

Nilai yang valid:

- sha1
- sha224
- sha256
- sha384
- sha512

Diperlukan: Ya

<MGF>

Menentukan fungsi pembuatan topeng.

> **Note**
>
> Fungsi hash fungsi pembuatan topeng harus sesuai dengan fungsi hash mekanisme penandatanganan.

Nilai yang valid:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

Diperlukan: Ya

<LABEL>

Label untuk kunci yang tidak dibungkus.

Diperlukan: Ya

<SESSION>

Membuat kunci sesi yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Diperlukan: Tidak

Topik terkait

- bungkus kunci
- buka kunci

buka kunci rsa-pkcs

key unwrap rsa-pkcsPerintah 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 (CU)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaks

```
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
      --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
  -h, --help
          Print help
```

Contoh-contoh

Contoh-contoh ini menunjukkan cara menggunakan key unwrap rsa-oaep perintah menggunakan kunci AES dengan nilai `unwrap` atribut yang disetel `ketrue`.

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+5HvU7sIBcXVb24QA0l65nbNAD+1bK+e18BpSfnaI3P+r8Dp+pLu1ofoUy/
vtzRjZoCiDofcz4EqCFnGl4GdcJ1/3W/5WRvMatCa2d7cx02swaeZcjKsermPXYRO1lGlfq6NskwMeeTkV8R7Rx9artFrs1
c3XdFJ2+0Bo94c6og/
yfPcpOOobJlITCoXhtMRepSdO4OggYq/6nUDuHCtJ86pPGnNahyr7+sAaSI3a5ECQLUjwaIARUCyoRh7EFK3qPXcg==
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08ef",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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": "0x00000000001c08ef",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "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
      }
    }
  }
}
```

Argumen

<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.

Diperlukan: 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.

Diperlukan: Tidak

<KEY_TYPE_CLASS>

Jenis kunci dan kelas kunci yang dibungkus [nilai yang mungkin:`aes`,`des3`,`ec-private`,`generic-secret`,`rsa-private`].

Diperlukan: Ya

<LABEL>

Label untuk kunci yang tidak dibungkus.

Diperlukan: Ya

<SESSION>

Membuat kunci sesi yang hanya ada di sesi saat ini. Kunci tidak dapat dipulihkan setelah sesi berakhir.

Diperlukan: Tidak

Topik terkait

- bungkus kunci
- buka kunci

bungkus kunci

key wrapPerintah di CloudHSM CLI mengekspor salinan terenkripsi dari kunci pribadi simetris atau asimetris dari HSM ke file. Ketika Anda menjalankankey wrap, Anda menentukan dua hal: Kunci untuk mengekspor dan file output. Kunci untuk mengekspor 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 mengekspor kunci yang sama beberapa kali. Untuk mengimpor

kunci terenkripsi kembali ke HSM, gunakan. buka kunci Hanya pemilik kunci, yaitu pengguna kripto (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:

- bungkus kunci aes-gcm
- bungkus kunci aes-no-pad
- bungkus kunci aes-pkcs5-pad
- bungkus kunci aes-zero-pad
- bungkus kunci cloudhsm-aes-gcm
- bungkus kunci rsa-aes
- bungkus kunci rsa-oaep
- bungkus kunci rsa-pkcs

bungkus kunci aes-gcm

key wrap aes-gcmPerintah membungkus kunci payload menggunakan kunci AES pada HSM dan mekanisme pembungkus. `AES-GCM` `extractable`Atribut kunci payload harus disetel ke`true`.

Hanya pemilik kunci, yaitu pengguna kripto (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 kunci menghasilkan aes simetris perintah dan `wrap` atribut yang disetel ke. `true`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

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
      --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": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "iv": "0xf90613bb8e337ec0339aad21",
    "wrapped_key_data": "xvslgrtg8kHzrvekny97tLSIeokpPwV8"
```

```
  }
}
```

Argumen

<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.

Diperlukan: Ya

<PATH>

Path ke file biner tempat data kunci yang dibungkus akan disimpan.

Diperlukan: Ya

<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.

Diperlukan: Ya

<AAD>

Nilai AES GCM Additional Authenticated Data (AAD), dalam hex.

Diperlukan: Tidak

<TAG_LENGTH_BITS>

Panjang tag AES GCM dalam bit.

Diperlukan: Ya

Topik terkait

- bungkus kunci
- buka kunci

bungkus kunci aes-no-pad

key wrap aes-no-padPerintah membungkus kunci payload menggunakan kunci AES pada HSM dan mekanisme pembungkus. `AES-NO-PAD` `extractable`Atribut kunci payload harus disetel ke`true`.

Hanya pemilik kunci, yaitu pengguna kripto (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 kunci menghasilkan aes simetris perintah dan `wrap` atribut yang disetel ke. `true`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

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
  -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": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "wrapped_key_data": "eXK3PMAOnKM9y3YX6brbhtMoC060EOH9"
  }
}
```

Argumen

<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.

Diperlukan: Ya

<PATH>

Path ke file biner tempat data kunci yang dibungkus akan disimpan.

Diperlukan: Ya

<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.

Diperlukan: Ya

Topik terkait

- bungkus kunci
- buka kunci

bungkus kunci aes-pkcs5-pad

key wrap aes-pkcs5-padPerintah membungkus kunci payload menggunakan kunci AES pada HSM dan mekanisme pembungkus. `AES-PKCS5-PAD` `extractable`Atribut kunci payload harus disetel ke`true`.

Hanya pemilik kunci, yaitu pengguna kripto (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 kunci menghasilkan aes simetris perintah dan `wrap` atribut yang disetel ke. `true`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

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
  -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": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
```

```
    "wrapped_key_data": "MbuYNresfOKyGNnxKWen88nSfX+uUE/0qmGofSisicY="
  }
}
```

Argumen

<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.

Diperlukan: Ya

<PATH>

Path ke file biner tempat data kunci yang dibungkus akan disimpan.

Diperlukan: Ya

<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.

Diperlukan: Ya

Topik terkait

- bungkus kunci
- buka kunci

bungkus kunci aes-zero-pad

key wrap aes-zero-padPerintah membungkus kunci payload menggunakan kunci AES pada HSM dan mekanisme pembungkus. `AES-ZERO-PAD` `extractable`Atribut kunci payload harus disetel ke`true`.

Hanya pemilik kunci, yaitu pengguna kripto (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 kunci menghasilkan aes simetris perintah dengan `wrap` atribut yang disetel ke. `true`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

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
  -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": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "wrapped_key_data": "L1wVlL/YeBNVAw6Mpk3owFJZXBzDLONt"
  }
}
```

Argumen

<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.

Diperlukan: Ya

<PATH>

Path ke file biner tempat data kunci yang dibungkus akan disimpan.

Diperlukan: Ya

<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.

Diperlukan: Ya

Topik terkait

- bungkus kunci
- buka kunci

bungkus kunci cloudhsm-aes-gcm

key wrap cloudhsm-aes-gcmPerintah membungkus kunci payload menggunakan kunci AES pada HSM dan mekanisme pembungkus. `CLOUDHSM-AES-GCM` `extractable`Atribut kunci payload harus disetel ke`true`.

Hanya pemilik kunci, yaitu pengguna kripto (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 kunci menghasilkan aes simetris perintah dan `wrap` atribut yang disetel ke. `true`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

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
      --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": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "wrapped_key_data": "6Rn8nkjEriDYlnP3P8nPkYQ8hplOEJ899zsrF+aTB0i/fIlZ"
```

```
  }
}
```

Argumen

<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.

Diperlukan: Ya

<PATH>

Path ke file biner tempat data kunci yang dibungkus akan disimpan.

Diperlukan: Ya

<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.

Diperlukan: Ya

<AAD>

Nilai AES GCM Additional Authenticated Data (AAD), dalam hex.

Diperlukan: Tidak

<TAG_LENGTH_BITS>

Panjang tag AES GCM dalam bit.

Diperlukan: Ya

Topik terkait

- bungkus kunci
- buka kunci

bungkus kunci rsa-aes

key wrap rsa-aesPerintah membungkus kunci payload menggunakan kunci publik RSA pada HSM dan mekanisme pembungkus RSA-AES. `extractable`Atribut kunci payload harus disetel ke`true`.

Hanya pemilik kunci, yaitu pengguna kripto (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 keypair RSA menggunakan kunci generate-asymmetric-pair perintah dan `wrap` atribut yang disetel ke. `true`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

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
         --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": "0x00000000001c08f1",
    "wrapping-key-reference": "0x00000000007008da",
    "wrapped-key-data": "HrSE1DEyLjIeyGdPa9R+ebiqB5TIJGyamPker31ZebPwRA
+NcerbAJO8DJ1lXPygZcI21vIFSZJuWMEiWpe1R9D/5WSYgxLVKex30xCFqebtEzxbKuv4DOmU4meSofqREYvtb3EoIKwjy
+RL5WGXKe4nAboAkC5GO7veI5yHL1SaKlssSJtTL/CFpbSLsAFuYbv/NUCWwMY5mwyVTCSlw+HlgKK
+5TH1MzBaSi8fpfyepLT8sHy2Q/VRl6ifb49p6m0KQFbRVvz/OWUd6l4d97BdgtaEz6ueg=="
  }
}
```

Argumen

<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.

Diperlukan: Ya

<PATH>

Path ke file biner tempat data kunci yang dibungkus akan disimpan.

Diperlukan: Ya

<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.

Diperlukan: 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

Diperlukan: Ya

Topik terkait

- bungkus kunci
- buka kunci

bungkus kunci rsa-oaep

key wrap rsa-oaepPerintah membungkus kunci payload menggunakan kunci publik RSA pada HSM dan mekanisme pembungkus. `RSA-OAEP` `extractable`Atribut kunci payload harus disetel ke`true`.

Hanya pemilik kunci, yaitu pengguna kripto (CU) yang membuat kunci, yang dapat membungkus kunci tersebut. Pengguna yang berbagi kunci dapat menggunakan kunci dalam operasi kriptografi.

Untuk menggunakan key wrap rsa-oaep perintah, Anda harus terlebih dahulu memiliki kunci RSA di AWS CloudHSM cluster Anda. Anda dapat menghasilkan keypair RSA menggunakan kunci generate-asymmetric-pair perintah dan `wrap` atribut yang disetel ke. `true`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

Persyaratan

- Untuk menjalankan perintah ini, Anda harus masuk sebagai CU.

Sintaksis

```
aws-cloudhsm > help key wrap rsa-oaep
Usage: key wrap rsa-oaep [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
      --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": "0x00000000001c08f1",
    "wrapping-key-reference": "0x00000000007008da",
    "wrapped-key-data": "OjJe4msobPLz9TuSAdULEu17T5rMDWtSlLyBSkLbaZnYzzpdrhsbGLbwZJCtB/
jGkDNdB4qyTAOQwEpggGf6v+Yx6JcesNeKKNU8XZal/YBoHC8noTGUSDI2qr+u2tDc84NPv6d
+F2KOONXsSxMhmxzzNG/gzTVIJhOuy/B1yHjGP4mOXoDZf5+7f5M1CjxBmz4Vva/
wrWHGCSG0yOaWblEvOiHAIt3UBdyKmU+/
My4xjfJv7WGGu3DFUUIZ06TihRtKQhUYU1M9u6NPf9riJJfHsk6QCuSZ9yWThDT9as6i7e3htnyDhIhGWaoK8JU855cN/
YNKAUqkNpC4FPL3iw=="
  }
}
```

Argumen

<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.

Diperlukan: Ya

<PATH>

Path ke file biner tempat data kunci yang dibungkus akan disimpan.

Diperlukan: Ya

<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.

Diperlukan: 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

Diperlukan: Ya

Topik terkait

- bungkus kunci
- buka kunci

bungkus kunci rsa-pkcs

key wrap rsa-pkcsPerintah membungkus kunci payload menggunakan kunci publik RSA pada HSM dan mekanisme pembungkus. `RSA-PKCS` `extractable`Atribut kunci payload harus disetel ke`true`.

Hanya pemilik kunci, yaitu pengguna kripto (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 keypair RSA menggunakan kunci generate-asymmetric-pair perintah dan `wrap` atribut yang disetel ke. `true`

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna Crypto (CU)

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
  -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": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000007008da",
    "wrapped_key_data": "am0Nc7+YE8FWs+5HvU7sIBcXVb24QA0l65nbNAD+1bK+e18BpSfnaI3P+r8Dp
+pLu1ofoUy/
vtzRjZoCiDofcz4EqCFnGl4GdcJ1/3W/5WRvMatCa2d7cx02swaeZcjKsermPXYRO1lGlfq6NskwMeeTkV8R7Rx9artFrs1
c3XdFJ2+0Bo94c6og/
yfPcpOOobJlITCoXhtMRepSdO4OggYq/6nUDuHCtJ86pPGnNahyr7+sAaSI3a5ECQLUjwaIARUCyoRh7EFK3qPXcg=="
  }
```

Argumen

<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.

Diperlukan: Ya

<PATH>

Path ke file biner tempat data kunci yang dibungkus akan disimpan.

Diperlukan: Ya

<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.

Diperlukan: Ya

Topik terkait

- bungkus kunci
- buka kunci

login

Anda dapat menggunakan login perintah di CloudHSM CLI untuk masuk dan keluar dari setiap HSM dalam sebuah cluster.

> **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. Hal ini karena klien CloudHSM menyeimbangkan beban di berbagai HSM. 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 kripto (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 log Anda ke semua HSM dalam sebuah cluster dengan kredensi dari pengguna admin bernama. `admin1`

```
aws-cloudhsm >login --username admin1 --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

Argumen

<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.

Diperlukan: Ya

<ROLE>

Menentukan peran yang ditetapkan untuk pengguna ini. Parameter ini diperlukan. Nilai yang valid adalahadmin,crypto-user.

Untuk mendapatkan peran pengguna, gunakan user list perintah. Untuk informasi rinci tentang jenis pengguna di HSM, lihat Memahami pengguna HSM.

<PASSWORD>

Menentukan kata sandi dari pengguna yang login ke HSM.

Topik terkait

- Memulai dengan CloudHSM CLI
- Aktifkan Cluster

Login mfa-token-sign

Gunakan login mfa-token-sign perintah di AWS CloudHSM CloudHSM CLI login menggunakan otentikasi multifaktor. Untuk menggunakan perintah ini, Anda harus terlebih dahulu mengatur MFA untuk CloudHSM CLI.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah ini.

- Admin
- Pengguna kripto (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"
  }
}
```

Argumen

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Wajib: Jika beberapa cluster telah dikonfigurasi.

<TOKEN>

Filepath tempat file token yang tidak ditandatangani akan ditulis.

Diperlukan: Ya

Topik terkait

- Memulai dengan CloudHSM CLI
- Aktifkan Cluster
- Menggunakan CloudHSM CLI untuk mengelola MFA

logout

Anda dapat menggunakan logout perintah di CloudHSM CLI untuk keluar dari setiap HSM dalam sebuah cluster.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Admin
- Pengguna kripto (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 membuat Anda keluar dari semua HSM dalam sebuah cluster.

```
aws-cloudhsm > logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

Topik terkait

- Memulai dengan CloudHSM CLI

- Aktifkan Cluster

pengguna

useradalah kategori induk untuk sekelompok perintah yang, ketika dikombinasikan dengan kategori induk, membuat perintah khusus untuk pengguna. Saat ini, kategori pengguna terdiri dari perintah berikut:

- perubahan pengguna-mfa
- perubahan-kata sandi pengguna
- membuat pengguna
- menghapus pengguna
- daftar pengguna

perubahan pengguna-mfa

Saat ini, kategori ini terdiri dari sub-perintah berikut:

- perubahan pengguna-tanda token mfa

perubahan pengguna-tanda token mfa

Gunakan user change-mfa perintah di CloudHSM CLI untuk memperbarui pengaturan Otentikasi Multifaktor (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

Sintaks

Saat ini, hanya ada satu strategi multifaktor 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"
  }
}
```

Argumen

<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 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

perubahan kata sandi pengguna

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 kripto (CU)

Sintaks

> **Note**
>
> Untuk mengaktifkan otentikasi multifaktor (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 kripto (CU).

```
aws-cloudhsm > user change-password --username bob --role crypto-user
Enter password:
```

```
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "bob",
    "role": "crypto-user"
  }
}
```

Argumen

<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.

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

perubahan-kuorum pengguna

user change-quorumadalah kategori induk untuk sekelompok perintah yang, ketika dikombinasikan dengan kategori induk, membuat perintah khusus untuk mengubah kuorum untuk pengguna.

user change-quorumdigunakan untuk mendaftarkan otentikasi kuorum pengguna menggunakan strategi kuorum tertentu. Pada SDK 5.8.0, hanya ada strategi kuorum tunggal yang tersedia untuk pengguna seperti yang ditunjukkan di bawah ini.

Saat ini, kategori ini terdiri dari kategori dan sub-perintah berikut:

- tanda token
  - mendaftar

tanda token perubahan-kuorum pengguna

user change-quorum token-signadalah kategori induk untuk perintah yang, ketika dikombinasikan dengan kategori induk ini, membuat perintah khusus untuk operasi kuorum tanda token.

Saat ini, kategori ini terdiri dari perintah berikut:

- mendaftar

daftar tanda token ubah-kuorum pengguna

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"
      }
    ]
  }
}
```

Argumen

<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

pengguna membuat

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 ini menunjukkan cara menggunakan user create untuk membuat pengguna baru di HSM Anda.

Example : Buat pengguna kripto

Contoh ini membuat akun di AWS CloudHSM klaster Anda dengan peran pengguna kripto.

```
aws-cloudhsm > user create --username alice --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

Argumen

<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.

Diperlukan: Ya

<ROLE>

Menentukan peran yang ditetapkan untuk pengguna ini. Parameter ini diperlukan. Nilai yang valid adalahadmin,crypto-user.

Untuk mendapatkan peran pengguna, gunakan user list perintah. Untuk informasi rinci tentang jenis pengguna di HSM, lihat Memahami pengguna HSM.

<PASSWORD>

Menentukan kata sandi dari pengguna yang login ke HSM.

Diperlukan: Ya

<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.

Topik terkait

- daftar pengguna
- menghapus pengguna
- perubahan kata sandi pengguna

menghapus pengguna

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.

Sintaks

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"
  }
}
```

Argumen

<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.

Diperlukan: Ya

<ROLE>

Menentukan peran yang ditetapkan untuk pengguna ini. Parameter ini diperlukan. Nilai yang valid adalahadmin,crypto-user.

Untuk mendapatkan peran pengguna, gunakan user list perintah. Untuk informasi rinci tentang jenis pengguna di HSM, lihat Memahami pengguna HSM.

Diperlukan: Ya

<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.

Diperlukan: Ya

Topik terkait

- daftar pengguna
- pengguna membuat
- perubahan kata sandi pengguna

daftar pengguna

user listPerintah di CloudHSM CLI mencantumkan akun pengguna yang ada di klaster CloudHSM Anda. Anda tidak perlu login ke CloudHSM CLI untuk menjalankan perintah ini.

> **Note**
>
> Jika Anda menambahkan atau menghapus HSM, perbarui file konfigurasi yang digunakan AWS CloudHSM klien dan alat baris perintah. Jika tidak, perubahan yang Anda buat mungkin tidak efektif pada semua HSM di klaster.

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

kuorum

quorumadalah kategori induk untuk sekelompok perintah yang, ketika dikombinasikan denganquorum, 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.

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
pengguna	Admin	• pengguna membuat • menghapus pengguna • perubahan kata sandi pengguna • perubahan pengguna-mfa
kuorum	Admin	• tanda token kuorum set-quorum-value

Topik terkait

- Menggunakan otentikasi kuorum untuk admin: penyiapan pertama kali
- Menggunakan CloudHSM CLI untuk mengelola autentikasi kuorum (M dari N kontrol akses)

tanda token kuorum

quorum token-signadalah kategori untuk sekelompok perintah yang, bila dikombinasikan denganquorum token-sign, membuat perintah khusus untuk otentikasi kuorum atau M operasi N.

Saat ini, kategori ini terdiri dari perintah berikut:

- hapus
- menghasilkan
- daftar
- list-quorum-values

- daftar-batas waktu
- set-quorum-value
- set-timeout

kuorum tanda token hapus

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"
}
```

Argumen

<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

tanda token kuorum menghasilkan

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.

> **Note**
>
> Hanya admin yang dapat menghasilkan token layanan.

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.

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
pengguna	Admin	• pengguna membuat • menghapus pengguna • perubahan kata sandi pengguna • perubahan pengguna-mfa
kuorum	Admin	• tanda token kuorum set-quorum-value

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Admin
- Pengguna kripto (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
          - registration:
            Registration service is used for registering a public key for quorum
 authentication

      --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"
  }
}
```

Argumen

<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.
- registrasi: Menghasilkan token yang tidak ditandatangani untuk digunakan dalam mendaftarkan kunci publik untuk otorisasi 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 tanda token kuorum

Gunakan quorum token-sign list perintah di CloudHSM CLI untuk mencantumkan semua token kuorum tanda token yang ada di klaster Anda. AWS CloudHSM

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Admin

- Pengguna kripto (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

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": 2
        "token-timeout-seconds": 588
        "cluster-coverage": "full"
```

```
      }
    ]
  }
}
```

Topik terkait

- tanda token kuorum menghasilkan

tanda token kuorum list-quorum-values

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

```
aws-cloudhsm > quorum token-sign list-quorum-values
```

```
{
  "error_code": 0,
  "data": {
    "user": 1,
    "quorum": 1
  }
}
```

Topik terkait

- Nama dan jenis layanan yang mendukung otentikasi kuorum

batas waktu daftar tanda token kuorum

Gunakan quorum token-sign list-timeouts perintah di CloudHSM CLI untuk mendapatkan periode batas waktu token dalam hitungan detik untuk semua jenis token.

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-timeouts
List timeout durations in seconds for token validity

Usage: quorum token-sign list-timeouts

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

```
aws-cloudhsm > quorum token-sign list-timeouts
{
```

```
  "error_code": 0,
  "data": {
    "generated": 600,
    "approved": 600
  }
}
```

Outputnya meliputi yang berikut:

- dihasilkan: Periode batas waktu dalam hitungan detik agar token yang dihasilkan disetujui.
- disetujui: Periode batas waktu dalam hitungan detik untuk token yang disetujui untuk digunakan untuk menjalankan operasi resmi kuorum.

Topik terkait

- tanda token kuorum set-timeout

tanda token kuorum set-quorum-value

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

      --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:

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 1,
```

```
    "quorum": 2
  }
}
```

Argumen

<CLUSTER_ID>

ID cluster untuk menjalankan operasi ini pada.

Diperlukan: 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.
- 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

tanda token kuorum set-timeout

Gunakan quorum token-sign set-timeout perintah di CloudHSM CLI untuk mengatur periode batas waktu token dalam hitungan detik untuk setiap jenis token.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Admin

Sintaksis

```
aws-cloudhsm > help quorum token-sign set-timeout
Set timeout duration in seconds for token validity

Usage: quorum token-sign set-timeout <--generated <GENERATED> |--approved <APPROVED>>

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
      --generated <GENERATED>     Timeout period in seconds for a generated (non-
approved) token to be approved
      --approved <APPROVED>      Timeout period in seconds for an approved token to be
 used to execute a quorum operation
  -h, --help    Print help (see a summary with '-h')
```

Contoh

Contoh berikut menunjukkan cara menggunakan quorum token-sign set-timeout perintah untuk mengatur periode batas waktu token.

```
aws-cloudhsm > quorum token-sign set-timeout --generated 900
{
  "error_code": 0,
  "data": "Set token timeout successful"
}
```

Topik terkait

- batas waktu daftar tanda token kuorum

Utilitas Manajemen CloudHSM (CMU)

Alat baris perintah cloudhsm_mgmt_util membantu petugas kripto (CO) mengelola pengguna di HSM. Ini termasuk alat yang membuat, menghapus, dan mendaftar pengguna serta mengubah kata sandi pengguna.

KMU dan CMU adalah bagian dari Client SDK 3 suite.

cloudhsm_mgmt_util juga menyertakan perintah yang mengizinkan pengguna kripto (CU) berbagi kunci serta mendapatkan dan mengatur atribut kunci. Perintah ini melengkapi perintah manajemen kunci dalam alat manajemen kunci primer, key_mgmt_util.

Untuk memulai dengan cepat, lihat Mengelola kloning kloning. Untuk informasi rinci tentang perintah cloudhsm_mgmt_util dan contoh penggunaan perintah, lihat Referensi Perintah cloudhsm_mgmt_util.

Topik

- Platform yang didukung untuk utilitas AWS CloudHSM manajemen
- Memulai dengan Utilitas Manajemen CloudHSM (CloudHSM)
- Instal dan konfigurasikan AWS CloudHSM klien (Linux)
- Instal dan atur konfigurasiAWS CloudHSM klien (Windows)
- Referensi Perintah cloudhsm_mgmt_util

Platform yang didukung untuk utilitas AWS CloudHSM manajemen

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 Utilitas Manajemen CloudHSM (CloudHSM)

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 parameter `--cmu` dan alamat IP dari salah satu HSM di klaster 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

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ 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

```
C:\Program Files\Amazon\CloudHSM> .\cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
\CloudHSM\data\cloudhsm_mgmt_util.cfg
```

Output harus serupa dengan berikut ini tergantung pada berapa banyak HSM yang 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-cloudhsm>` saat cloudhsm_mgmt_util berjalan.

3. 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 kripto (CO) default. Anda menyetel kata sandi pengguna ini saat Anda mengaktifkan klaster. Anda dapat menggunakan parameter `-hpswd` untuk menyembunyikan kata sandi Anda.

   ```
   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 semua HSM di klaster.

   ```
   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)
```

4. 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: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
```

5. 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 mendaftar 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              CO              admin                                NO
               0               NO
         2              AU              app_user                             NO
               0               NO
         3              CU              example_user                         NO
               0               NO
Users on server 1(10.0.3.11):
Number of users found:3
```

```
    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                   NO
               0              NO
         2              AU              app_user                                NO
               0              NO
         3              CU              example_user                            NO
               0              NO
Users on server 2(10.0.1.12):
Number of users found:3

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                   NO
               0              NO
         2              AU              app_user                                NO
               0              NO
         3              CU              example_user                            NO
               0              NO
```

8. Gunakan perintah logoutHSM untuk keluar dari HSM.

```
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 konfigurasikan AWS CloudHSM klien (Linux)

Untuk berinteraksi dengan HSM di AWS CloudHSM cluster Anda, Anda memerlukan perangkat lunak AWS CloudHSM klien untuk Linux. 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 atur konfigurasiAWS CloudHSM klien (Windows).

Tugas

- Instal AWS CloudHSM klien dan alat baris perintah
- Edit konfigurasi klien

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
```

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 config -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 perintah describe-clusters CLI. Dalam output perintah, alamat IP HSM adalah nilai bidang `EniIp`. Jika Anda memiliki lebih dari satu HSM, pilih alamat IP untuk salah satu HSM; 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 Mengaktifkan klaster.

Instal dan atur konfigurasiAWS CloudHSM klien (Windows)

Untuk bekerja dengan HSM di klaster AWS CloudHSM Anda pada Windows, Anda memerlukan perangkat lunak klien AWS CloudHSM untuk Windows. 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 penginstalAWSCloudHSMClient -latest.msi.
3. Jika menginstal Client SDK 3 di cloudhsm_mgmt_util, selesaikan langkah-langkah berikut untuk memastikan semua node di klaster disinkronkan.

   a. Jalankan configure -a ***<IP of one of the HSMs>***.

   b. Mulai ulang layanan klien.

   c. Jalankan config -m.
4. Pergi ke lokasi unduhan Anda dan jalankan pemasang (AWSCloudHSMClient-latest.msi) dengan hak istimewa 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:

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe -a <HSM IP address>
   ```
8. Kunjungi Mengaktifkan klaster.

Catatan:

- Jika Anda memperbarui klien, file konfigurasi yang ada dari instalasi sebelumnya tidak ditimpa.
- Penginstal klien AWS CloudHSM untuk Windows secara otomatis mendaftarkan API Kriptografi: Next Generation (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 konfigurasikan AWS CloudHSM klien (Linux).

Referensi Perintah cloudhsm_mgmt_util

Alat baris perintah cloudhsm_mgmt_util membantu petugas kripto (CO) mengelola pengguna di HSM. Ini juga menyertakan perintah yang mengizinkan pengguna kripto (CU) berbagi kunci serta mendapatkan dan mengatur atribut kunci. Perintah ini melengkapi perintah manajemen kunci primer di alat baris perintah key_mgmt_util.

Untuk memulai dengan cepat, lihat Mengelola kloning kloning.

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 HSM, 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 Referensi Atribut Kunci.

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 pada HSM. Setiap pengguna dapat mengubah kata sandi mereka sendiri. CO dapat mengubah kata sandi siapa pun.	CO
createUser	Membuat pengguna dari semua jenis pada HSM.	CO
deleteUser	Menghapus pengguna dari semua jenis dari HSM.	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 kunci AWS CloudHSM dan menuliskannya ke file atau stdout (output standar).	CU
getCert	Mendapat sertifikat dari HSM tertentu dan menyimpannya	Semua.

Perintah	Deskripsi	Jenis Pengguna
	dalam format sertifikat yang diinginkan.	
getHSMInfo	Mendapat informasi tentang perangkat keras tempat HSM berjalan.	Semua. Login tidak diperlukan.
getKeyInfo	Mendapatkan pemilik, pengguna bersama, dan status autentikasi kuorum kunci.	Semua. Login tidak diperlukan.
info	Mendapat informasi tentang HSM, termasuk alamat IP, nama host, port, dan pengguna saat ini.	Semua. Login tidak diperlukan.
listUsers	Mendapat pengguna di masing-masing HSM, jenis pengguna dan ID mereka, dan atribut lainnya.	Semua. Login tidak diperlukan.
loginHSM dan logoutHSM	Login dan log out dari HSM.	Semua.
berhenti	Keluar dari cloudhsm_mgmt_util.	Semua. Login tidak diperlukan.
peladen	Memasuki dan keluar modus server pada HSM.	Semua.
registerQuorumPubKey	Mengaitkan pengguna HSM dengan pasangan kunci RSA-2048 asimetris.	CO

Perintah	Deskripsi	Jenis Pengguna
setAttribute	Mengubah nilai-nilai label, mengenkripsi, mendekripsi, membungkus, dan membuka atribut kunci yang ada.	CU
shareKey	Berbagi kunci yang ada dengan pengguna lain.	CU
syncKey	Menyinkronkan kunci di seluruh klaster AWS CloudHSM yang diklon.	CU, CO
syncUser	Menyinkronkan pengguna di seluruh klaster AWS CloudHSM yang diklon.	CO

changePswd

Perintah changePswd di cloudhsm_mgmt_util mengubah kata sandi pengguna yang ada di HSM di klaster.

Setiap pengguna dapat mengubah kata sandi mereka sendiri. Selain itu, Petugas kripto (CO dan PCO) dapat mengubah kata sandi pengguna CO atau kripto lain (CU). Anda tidak perlu memasukkan kata sandi saat ini untuk melakukan perubahan.

> **Note**
>
> Anda tidak dapat mengubah kata sandi pengguna yang saat ini login ke klien AWS CloudHSM 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 akan digunakan.

Jika Anda menambahkan atau menghapus HSM, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSM di klaster.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Petugas krito (CO)
- Pengguna kripto (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 pengguna saat ini atau pengguna lain di HSM Anda.

Example : Ubah kata sandi Anda

Setiap pengguna di HSM dapat menggunakan changePswd untuk mengubah kata sandi mereka sendiri. Sebelum Anda mengubah kata sandi, gunakan Info untuk mendapatkan informasi tentang masing-masing HSM di klaster, termasuk nama pengguna dan jenis pengguna login pengguna.

Output berikut menunjukkan bahwa Bob saat ini login sebagai pengguna kripto (CU).

```
        aws-cloudhsm> info server 0

Id      Name                    Hostname         Port   State           Partition
  LoginState
0       10.1.9.193              10.1.9.193      2225    Connected       hsm-jqici4covtv
  Logged in as 'bob(CU)'

aws-cloudhsm> info server 1
```

```
Id      Name                    Hostname         Port   State           Partition
  LoginState
1       10.1.10.7               10.1.10.7       2225    Connected       hsm-ogi3sywxbqx
  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 sandi CO lain, atau CU pada HSM. 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
 LoginState
0       10.1.9.193       10.1.9.193        2225   Connected      hsm-jqici4covtv
 Logged in as 'alice(CO)'


aws-cloudhsm>info server 1
```

```
Id      Name             Hostname         Port   State          Partition
 LoginState
0       10.1.10.7        10.1.10.7        2225   Connected      hsm-ogi3sywxbqx
 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              0
             NO
         2              AU              jane                      NO              0
             NO
         3              CU              bob                       NO              0
             NO
         4              CU              alice                     NO              0
             NO
         5              CO              john                      NO              0
             NO
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              0
             NO
         2              AU              jane                      NO              0
             NO
         3              CU              bob                       NO              0
             NO
         4              CO              alice                     NO              0
             NO
         5              CO              john                      NO              0
             NO
```

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 Menggunakan CMU untuk mengelola 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 Memahami pengguna HSM.

Wajib: Ya

<user name>

Menentukan nama yang mudah diigat oleh pengguna. Parameter ini tidak peka huruf besar kecil. Anda tidak dapat menggunakan changePswd untuk mengubah nama pengguna.

Wajib: Ya

<kata sandi | -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 Menggunakan CMU untuk mengelola 2FA.

Wajib: Tidak

Topik terkait

- info
- DaftarPengguna
- CreateUser
- DeleteUser

createUser

Perintah createUser di cloudhsm_mgmt_util membuat pengguna di HSM. Hanya petugas kripto (CoS dan PreCoS) yang dapat menjalankan perintah ini. Ketika perintah berhasil, pengguna dibuat di semua HSM di klaster.

Untuk memecahkan masalah createUser

Jika konfigurasi HSM Anda tidak akurat, pengguna mungkin tidak dibuat pada semua HSM. Untuk menambahkan pengguna ke setiap HSM tempatnya hilang, gunakan perintah syncUser atau createUser hanya pada HSM yang kehilangan pengguna tersebut. 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 menambahkan atau menghapus HSM, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSM di klaster.

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 ini menunjukkan cara menggunakan createUser untuk membuat pengguna baru di HSM Anda.

Example : Buat petugas crypto

Contoh ini membuat seorang perwira kripto (CO) pada HSM dalam sebuah klaster. Perintah pertama menggunakan loginHSM untuk masuk ke HSM sebagai pegawai kripto.

```
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 kripto baru di HSM.

Pesan peringatan menjelaskan bahwa perintah membuat pengguna pada semua HSM di klaster. Tapi, jika perintah gagal pada HSM mana pun, pengguna tidak akan ada pada HSM tersebut. Untuk melanjutkan, ketik `y`.

Output menunjukkan bahwa pengguna baru dibuat pada ketiga HSM di klaster.

```
aws-cloudhsm> createUser CO 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(CO) on 3 nodes
```

Saat perintah selesai, `alice` memiliki izin yang sama pada HSM sebagai pengguna CO `admin`, termasuk mengubah kata sandi dari setiap pengguna di HSM.

Perintah terakhir menggunakan perintah listUsers untuk memverifikasi bahwa `alice` ada ketiga HSM di klaster. 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              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
  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
  LoginFailureCnt         2FA
        1               PRECO           admin                             YES
          0               NO
        2               AU              app_user                           NO
          0               NO
        3               CO              alice                              NO
          0               NO
```

Example : Buat pengguna kripto

Contoh ini membuat pengguna kripto (CU), bob, pada HSM. Pengguna kripto dapat membuat dan mengelola kunci, tetapi mereka tidak dapat mengelola pengguna.

Setelah Anda mengetik y untuk menanggapi pesan hati-hati, output menunjukkan bahwa bob dibuat pada ketiga HSM di klaster. 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 Menggunakan CMU untuk mengelola 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 Memahami pengguna HSM.

Nilai yang valid:

- CO: Petugas kripto dapat mengelola pengguna, tetapi mereka tidak dapat mengelola kunci.
- CU: Pengguna kripto dapat membuat mengelola kunci dan menggunakan kunci dalam operasi kriptografi.

PRECO diubah menjadi 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) pada HSM.

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 Menggunakan CMU untuk mengelola 2FA.

Wajib: Tidak

Topik terkait

- DaftarPengguna
- DeleteUser
- sinkUser
- PerubahanSWD

deleteUser

Perintah deleteUser di cloudhsm_mgmt_util menghapus pengguna dari modul keamanan perangkat keras (HSM). Hanya petugas kripto (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 kripto (CU) yang memiliki kunci.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- CO

Sintaks

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 seorang petugas kripto (CO) dari HSM dalam sebuah klaster. Perintah pertama menggunakan listUsers untuk mencantumkan semua pengguna di HSM.

Output menunjukkan bahwa pengguna `3`, `alice`, adalah CO pada HSM.

```
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                                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
  LoginFailureCnt         2FA
         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              PCO             admin                                YES
           0               NO
         2              AU              app_user                              NO
           0               NO
         3              CO              alice                                 NO
           0               NO
```

Perintah kedua menggunakan perintah deleteUser untuk menghapus `alice` dari HSM.

Output menunjukkan bahwa perintah berhasil pada ketiga HSM di klaster.

```
aws-cloudhsm> deleteUser CO alice
Deleting user alice(CO) 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 perintah listUsers untuk memverifikasi bahwa `alice` dihapus dari ketiga HSM di klaster.

```
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              PCO             admin                                YES
           0               NO
         2              AU              app_user                              NO
           0               NO
Users on server 1(10.0.0.2):
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
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
```

Argumen

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 kripto (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 Memahami pengguna 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

Topik terkait

- ListUsers
- CreateUser
- SynCuser
- ChangepSWD

findAllKeys

Perintah findAllKeys di cloudhsm_mgmt_util mendapatkan kunci yang dimiliki atau dibagikan oleh pengguna kripto (CU) tertentu. Hal ini juga mengembalikan hash dari data pengguna pada masing-masing HSM. Anda dapat menggunakan hash untuk menentukan sekilas apakah pengguna, kepemilikan kunci, dan berbagi kunci data yang sama pada semua HSM di klaster. Dalam output, kunci yang dimiliki oleh pengguna dianotasi oleh `(o)` dan kunci bersama dianotasi oleh `(s)`.

findAllKeys mengembalikan kunci publik hanya ketika CU ditentukan memiliki kunci, meskipun semua CU pada HSM dapat menggunakan kunci publik. Perilaku ini berbeda dari findKey di key_mgmt_util, yang mengembalikan kunci publik untuk semua pengguna CU.

Hanya petugas kripto (CO dan PCO) dan pengguna peralatan (AU) yang dapat menjalankan perintah ini. Pengguna kripto (CU) dapat menjalankan perintah berikut:

- listUsers untuk menemukan semua pengguna
- findKey di key_mgmt_util untuk menemukan kunci yang dapat mereka gunakan
- getKeyInfodi key_mgmt_util untuk menemukan pemilik dan pengguna bersama kunci tertentu yang mereka miliki atau bagikan

Sebelum Anda menjalankan perintah CMU, Anda harus memulai CMU dan masuk ke HSM. Pastikan bahwa Anda masuk dengan jenis pengguna yang dapat menjalankan perintah yang ingin Anda gunakan.

Jika Anda menambahkan atau menghapus HSM, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSM di klaster.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Petugas kripto (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 ini menunjukkan cara menggunakan `findAllKeys` untuk menemukan semua kunci untuk pengguna dan mendapatkan hash dari informasi pengguna kunci pada masing-masing HSM.

Example : Menemukan kunci untuk CU

Contoh ini menggunakan findAllKeys untuk menemukan kunci dalam HSM yang dimiliki dan dibagikan oleh pengguna 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 kripto.

```
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 menggunakan findAllKeys untuk memverifikasi bahwa semua HSM di klaster berisi pengguna, kepemilikan kunci, dan nilai berbagi kunci yang sama. 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 pengguna 6, tetapi nilai hash akan sama untuk setiap pengguna yang memiliki atau berbagi salah satu kunci pada HSM. 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 HSM di klaster. 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 dari pengguna pada HSM. Untuk menemukan ID pengguna semua pengguna, gunakan listUsers.

Semua ID pengguna valid, tetapi `findAllKeys` mengembalikan kunci hanya untuk pengguna kripto (CU).

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
- getKeyInfodi key_mgmt_util

getAttribute

Perintah getAttribute di cloudhsm_mgmt_util mendapat satu nilai atribut untuk kunci dari semua HSM di klaster dan menuliskannya ke stdout (output standar) atau ke file. Hanya pengguna kripto (CU) yang bisa 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 Referensi Atribut Kunci.

Sebelum Anda menjalankan perintah CMU, Anda harus memulai CMU dan masuk ke HSM. Pastikan bahwa Anda masuk dengan jenis pengguna yang dapat menjalankan perintah yang ingin Anda gunakan.

Jika Anda menambahkan atau menghapus HSM, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSM di klaster.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Pengguna kripto (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 mendapat nilai atribut yang dapay diekstrak untuk kunci dalam HSM. Anda dapat menggunakan perintah seperti ini untuk menentukan apakah Anda dapat mengekspor kunci dari HSM.

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 Referensi Atribut Kunci.

```
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 menggunakan getAttribute untuk mendapatkan nilai dari atribut yang dapat diekstrak untuk kunci dengan handel kunci `262170` di HSM. 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 nilai adalah 0 (0x0), nilai itu tidak dapat diekspor dari HSM. 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 getKeyInfodalam 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 Referensi Atribut Kunci.

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 Kunci

getCert

Dengan getCert di cloudhsm_mgmt_util, Anda dapat mengambil sertifikat HSM tertentu dalam sebuah klaster. Ketika Anda menjalankan perintah, Anda menetapkan jenis sertifikat untuk diambil. Untuk melakukannya, Anda menggunakan salah satu bilangan bulat 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 CMU, Anda harus memulai CMU dan masuk ke HSM. Pastikan bahwa Anda masuk dengan jenis pengguna yang dapat menjalankan perintah yang ingin Anda gunakan.

Jika Anda menambahkan atau menghapus HSM, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSM di klaster.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Semua pengguna.

Prasyarat

Sebelum memulai, Anda harus memasukkan mode server pada target HSM. Untuk informasi lebih lanjut, lihat server.

Sintaksis

Untuk menggunakan perintah getCert sekali dalam modus server:

```
server> getCert <file-name> <certificate-type>
```

Contoh

Pertama, masukkan mode server. Perintah ini memasuki modus server pada HSM dengan nomor server `0`.

```
aws-cloudhsm> server 0

Server is in 'E2' mode...
```

Kemudian, gunakan perintah getCert. Dalam contoh ini, kami menggunakan `/tmp/P0.crt` sebagai nama file tempat sertifikat akan disimpan dan `4` (Sertifikat Root Pelanggan) sebagai jenis sertifikat yang diinginkan:

```
server0> getCert /tmp/P0.crt 4
getCert Success
```

Pendapat

```
getCert <file-name> <certificate-type>
```

<file-name>

Menentukan nama file tempat sertifikat disimpan.

Wajib: Ya

<certificate-type>

Bilangan bulat yang menentukan jenis sertifikat untuk diambil. Bilangan bulat dan jenis sertifikatnya yang sesuai adalah sebagai berikut:

- 1 – Sertifikat Akar Produsen
- 2 — Sertifikat Perangkat Keras Produsen
- 4 — Sertifikat Root Pelanggan
- 8— Sertifikat Klaster (ditandatangani oleh Sertifikat Root Pelanggan)

- 16 – Sertifikat klaster (dirantai ke Sertifikat Root Produsen)

Wajib: Ya

Topik terkait

- peladen

getHSMInfo

Perintah getHSMInfo di cloudhsm_mgmt_util mendapat informasi tentang perangkat keras tempat setiap HSM berjalan, termasuk model, nomor seri, kondisi 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 bahwa Anda masuk dengan jenis pengguna yang dapat menjalankan perintah yang ingin Anda gunakan.

Jika Anda menambahkan atau menghapus HSM, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSM di klaster.

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 menggunakan getHSMInfo untuk mendapatkan informasi tentang HSM di klaster.

```
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          :xxxxxxxxxxxxxx

...
```

Topik terkait

- info

getKeyInfo

Perintah getKeyInfo di key_mgmt_util mengembalikan ID pengguna HSM dari pengguna yang dapat menggunakan kunci, termasuk pemilik dan pengguna kripto (CU) dengan siapa kunci dibagi. 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 ID pengguna HSM dari pengguna di HSM Anda, gunakan listUsers. Untuk menemukan kunci untuk pengguna tertentu, gunakan findKey `-u` di key_mgmt_util. Petugas kripto (AS) dapat menggunakan findAllKeysdi 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 bahwa Anda masuk dengan jenis pengguna yang dapat menjalankan perintah yang ingin Anda gunakan.

Jika Anda menambahkan atau menghapus HSM, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSM di klaster.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna kripto (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 key pair simetris

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 privat (262177), perintah ini mengembalikan pemilik kunci (3) dan pengguna kripto (CU) 4, dengan siapa kunci 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 menggunakan genSymKeyuntuk membuat kunci AES 256-bit yang dibagi 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

- getKeyInfodi key_mgmt_util

- findKey di key_mgmt_util
- findAllKeysdi cloudhsm_mgmt_util
- listUsers
- shareKey

info

Perintah info di cloudhsm_mgmt_util mendapat informasi tentang masing-masing HSM di klaster, termasuk nama host, port, alamat IP serta nama dan jenis pengguna yang login ke cloudhsm_mgmt_util pada HSM.

Sebelum Anda menjalankan perintah CMU, Anda harus memulai CMU dan masuk ke HSM. Pastikan bahwa Anda masuk dengan jenis pengguna yang dapat menjalankan perintah yang ingin Anda gunakan.

Jika Anda menambahkan atau menghapus HSM, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSM di klaster.

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 klaster. Perintah menggunakan 0 untuk merujuk ke HSM pertama di klaster. 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. HSM mendapat nomor urut yang mewakili urutan di mana mereka ditambahkan ke klaster, dimulai dengan 0. Untuk menemukan ID server dari HSM, gunakan getHSMInfo.

Wajib: Ya

Topik terkait

- getHSMInfo
- loginHSM dan logoutHSM

listAttributes

Perintah listAttributes di cloudhsm_mgmt_util berisi daftar atribut kunci AWS CloudHSM dan konstanta yang mewakilinya. Anda menggunakan konstanta ini untuk mengidentifikasi atribut di getAttribute dan perintah setAttribute.

Untuk membantu menafsirkan atribut kunci, lihat Referensi Atribut Kunci.

Sebelum Anda menjalankan perintah key_mgmt_util, Anda harus memulai key_mgmt_util dan Masuk ke HSM sebagai pengguna kripto (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 Referensi Atribut Kunci. 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.

Diperlukan: Ya

Topik terkait

- getAttribute
- setAttribute
- Referensi Atribut Kunci

listUsers

Parameter listUsers perintah di cloudhsm_mgmt_util mendapat pengguna di masing-masing HSM, bersama dengan jenis pengguna mereka 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 bahwa Anda masuk dengan jenis pengguna yang dapat menjalankan perintah yang ingin Anda gunakan.

Jika Anda menambahkan atau menghapus HSM, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSM di klaster.

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 berisi daftar pengguna pada masing-masing HSM di klaster dan menampilkan atributnya. 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              PCO             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              CO              officer1                 YES              0
            NO
        6              CO              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              PCO             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              CO             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 autentikasi kuorum.
- LoginFailureCnt: Menunjukkan frekuensi pengguna tidak berhasil login.
- 2FA: Menunjukkan bahwa pengguna telah mengaktifkan autentikasi multi-faktor.

Topik terkait

- listUsers di key_mgmt_util
- CreateUser
- deleteUser
- changePswd

loginHSM dan logoutHSM

Anda dapat menggunakan perintah loginHSM dan logoutHSM di cloudhsm_mgmt_util untuk masuk dan keluar dari setiap HSM dalam sebuah klaster. 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 menambahkan atau menghapus HSM, perbarui file konfigurasi yang perintah klien AWS CloudHSM dan alat baris perintah. Jika tidak, perubahan yang Anda buat mungkin tidak efektif pada semua HSM di klaster.

Jika Anda memiliki lebih dari satu HSM di klaster Anda, Anda mungkin diizinkan untuk upaya login salah tambahan sebelum akun Anda terkunci. Hal ini karena klien CloudHSM menyeimbangkan beban di berbagai HSM. 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 prakripto (PRECO)
- Petugas kripto (CO)
- Pengguna kripto (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 untuk masuk dan keluar dari semua HSM dalam sebuah klaster.

Example : Masuk ke HSM di klaster

Perintah ini memasukkan Anda ke semua HSM dalam sebuah klaster dengan kredensials dari pengguna CO bernama `admin` dan kata sandi `co12345`. Output menunjukkan bahwa perintah berhasil dan bahwa Anda telah terhubung ke HSM (yang, dalam hal ini, `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 HSM.

```
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 mengeluarkan Anda dari HSM yang Anda masuki saat ini (yang, dalam kasus ini, adalah `server 0` dan `server 1`). Output menunjukkan bahwa perintah berhasil dan bahwa Anda telah terputus dari HSM.

```
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 Menggunakan CMU untuk mengelola 2FA

```
loginHSM <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<user type>

Menentukan jenis pengguna yang masuk ke HSM. Untuk informasi lebih lanjut, lihat Jenis Pengguna di atas.

Wajib: Ya

<user name>

Menentukan nama pengguna dari pengguna yang masuk ke HSM.

Wajib: Ya

<password | -hpswd >

Menentukan kata sandi dari pengguna yang login ke HSM. 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, lihatMenggunakan CMU untuk mengelola 2FA.

Wajib: Tidak

Topik terkait

- Memulai dengan cloudhsm_mgmt_util

- Aktifkan Cluster

registerQuorumPubKunci

Parameter registerQuorumPubKey di cloudhsm_mgmt_util 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**
>
> Di dokumentasi AWS CloudHSM, autentikasi kuorum terkadang disebut sebagai M dari N (MofN), yang artinya minimalM pemberi persetujuan dari jumlah total N pemberi persetujuan.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Petugas krito (CO)

Sintaks

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

Contoh ini menunjukkan cara menggunakan registerQuorumPubKey untuk mendaftarkan petugas kripto (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 “Argumen”.

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                             NO
          0               NO
        2               AU              app_user                          NO
          0               NO
        3               CO              quorum_officer                    YES
          0               NO
```

Argumen

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 Memahami pengguna HSM.

Nilai yang valid:

- CO: Petugas kripto 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.

Diperlukan: Ya

<signed-registration-token>

Menentukan jalur ke file yang berisi mekanisme SHA256_PKCS ditandatangani hash dari token pendaftaran. 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

server

Biasanya, ketika Anda mengeluarkan perintah di cloudhsm_mgmt_util, perintah memengaruhi semua HSM di klaster 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. Anda dapat menggunakan perintah server di cloudhsm_mgmt_util untuk masuk mode server dan berinteraksi langsung dengan instans HSM tertentu.

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 bahwa HSM yang 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.

Diperlukan: Ya

Tidak ada argumen untuk perintah `exit`.

Topik terkait

- syncKey
- createUser
- deleteUser

setAttribute

Perintah setAttribute di cloudhsm_mgmt_util mengubah nilai label, mengenkripsi, mendekripsi, membungkus, dan membuka atribut kunci di HSM. 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 bahwa Anda masuk dengan jenis pengguna yang dapat menjalankan perintah yang ingin Anda gunakan.

Jika Anda menambahkan atau menghapus HSM, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSM di klaster.

Jenis pengguna

Pengguna berikut dapat menjalankan perintah berikut.

- Pengguna kripto (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 genSymKeydi 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 Referensi Atribut Kunci.

```
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
```

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 benar (1) pada kedua HSM di klaster.

```
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 kedua HSM di klaster.

```
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 salah (0) pada kedua HSM di klaster.

```
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 id>
```

<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 Referensi Atribut Kunci.

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 Kunci

berhenti

Perintah quit di cloudhsm_mgmt_util 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

shareKey

Perintah shareKey di cloudhsm_mgmt_util membagikan dan batal membagikan kunci yang Anda miliki dengan pengguna kripto 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, mengekspor, 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 bahwa Anda masuk dengan jenis pengguna yang dapat menjalankan perintah yang ingin Anda gunakan.

Jika Anda menambahkan atau menghapus HSM, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSM di klaster.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Pengguna kripto (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 kripto lainnya.

Example : Bagikan Kunci

Contoh ini menggunakan shareKey untuk berbagi kunci privat ECC yang dimiliki pengguna saat ini dengan pengguna kripto lain di HSM. Kunci publik tersedia untuk semua pengguna HSM, sehingga Anda tidak dapat membagi atau batal membagikannya.

Perintah pertama menggunakan getKeyInfountuk mendapatkan informasi pengguna untuk kunci262177, kunci privat ECC pada HSM.

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 menggunakan shareKey untuk berbagi kunci 262177 dengan pengguna 4, pengguna kripto lain di HSM. Argumen terakhir menggunakan nilai 1 untuk menunjukkan operasi berbagi.

Output menunjukkan bahwa operasi berhasil pada kedua HSM di klaster.

```
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 : Membatalkan Berbagi Kunci

Contoh ini batal membagikan kunci simetris, yaitu menghapus pengguna kripto 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 kedua HSM. 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 kripto (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

syncKey

Anda dapat menggunakan syncKey di cloudhsm_mgmt_util untuk secara manual menyinkronkan kunci di seluruh instans HSM dalam sebuah klaster atau di seluruh klaster 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 klaster arbitrer: salah satu klaster harus telah dibuat dari cadangan yang lain. Selain itu, kedua klaster harus konsisten dengan kredensial CO dan CU agar operasi menjadi sukses. Untuk informasi lebih lanjut, lihat Pengguna HSM.

Untuk menggunakannyasyncKey, 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 kripto (CU)

> **Note**
>
> CO dapat menggunakan syncKey pada setiap tombol, sementara CU hanya dapat menggunakan perintah ini pada tombol yang mereka miliki. Untuk informasi selengkapnya, lihat the section called “Memahami pengguna HSM”.

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 pengenal untuk pengguna yang bernama. Kemudian, gunakan findAllKeysperintah untuk menemukan semua kunci milik pengguna tertentu.

Anda juga perlu mengetahui `server IDs` yang ditugaskan ke HSM sumber dan tujuan, yang ditampilkan dalam output jejak yang dikembalikan oleh cloudhsm_mgmt_util setelah inisiasi. Kunci ini ditugaskan dalam urutan yang sama yang dimunculkan HSM dalam file konfigurasi.

Ikuti instruksi di Menggunakan CMU di Klaster Kloning dan menginisialisasi cloudhsm_mgmt_util dengan file config baru. Kemudian, masukkan mode server pada HSM sumber dengan mengeluarkan perintah server.

Sintaks

> **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
```

Argumen

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 findAllKeyssaat masuk ke server HSM.

Diperlukan: Ya

<destination hsm>

Menentukan jumlah server yang akan Anda sinkronkan dengan kunci.

Wajib: Ya

Topik terkait

- ListUsers
- findAllKeys
- jelaskan-cluster di CLI
- server

syncUser

Anda dapat menggunakan syncUser perintah di cloudhsm_mgmt_util untuk menyinkronkan pengguna kripto (CU) atau petugas kripto (CO) secara manual di seluruh instance HSM dalam klaster atau di seluruh kloning kloning. AWS CloudHSM tidak secara otomatis menyinkronkan pengguna. Umumnya, Anda mengelola pengguna dalam modus global sehingga semua HSM dalam sebuah klaster 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 klaster 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 menambahkan atau menghapus HSM, perbarui file konfigurasi untuk CMU. Jika tidak, perubahan yang Anda buat mungkin tidak efektif untuk semua HSM di klaster.

Jenis pengguna

Jenis pengguna berikut dapat menjalankan perintah ini.

- Petugas krito (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 mendaftar semua pengguna di HSM dalam sebuah klaster.

Anda juga perlu mengetahui `server ID` yang ditugaskan ke HSM sumber dan tujuan, yang ditampilkan dalam output jejak yang dikembalikan oleh cloudhsm_mgmt_util setelah inisiasi. Pengguna ini ditugaskan dalam urutan yang sama yang dimunculkan HSM dalam file konfigurasi.

Jika Anda menyinkronkan HSM di klaster kloning, ikuti petunjuk di Menggunakan CMU di Kloning Klaster dan inisialisasi cloudhsm_mgmt_util dengan file config baru.

Ketika Anda siap untuk menjalankan syncUser, masuk ke mode server pada HSM sumber dengan mengeluarkan perintah server.

Sintaks

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
```

Argumen

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 CLI
- server

Utilitas Manajemen Kunci (KMU)

Gunakan alat baris perintah utilitas manajemen kunci (KMU) untuk mengelola kunci pada modul keamanan perangkat keras (HSM) di klaster . 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 key_mgmt_util. Untuk informasi detail mengenai perintah, lihat key_mgmt_util referensi perintah. Untuk membantu menafsirkan atribut kunci, lihat Referensi Atribut Kunci.

Untuk menggunakan key_mgmt_util, jika Anda menggunakan Linux, hubungkan ke instans klien Anda dan kemudian lihat Instal dan konfigurasikan AWS CloudHSM klien (Linux). Jika Anda menggunakan Windows, lihat Instal dan konfigurasikan AWS CloudHSM klien (Windows).

Topik

- Memulai dengan key_mgmt_util
- Instal dan konfigurasikan AWS CloudHSM klien (Linux)
- Instal dan konfigurasikan AWS CloudHSM klien (Windows)
- key_mgmt_util referensi perintah

Memulai dengan key_mgmt_util

AWS CloudHSM termasuk dua alat baris perintah dengan perangkat lunak klien AWS CloudHSM. 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 key_mgmt_util
- Penggunaan dasar 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 key_mgmt_util referensi perintah.

Mengatur key_mgmt_util

Selesaikan pengaturan berikut sebelum Anda menggunakan key_mgmt_util.

Mulai AWS CloudHSM klien

Sebelum Anda menggunakan key_mgmt_util, Anda harus memulai klien AWS CloudHSM. Klien adalah daemon yang menetapkan komunikasi terenkripsi ujung ke ujung dengan HSM di klaster Anda. Alat key_mgmt_util menggunakan koneksi klien untuk berkomunikasi dengan HSM di klaster Anda. Tanpa itu, key_mgmt_util tidak bekerja.

Untuk memulai klien AWS CloudHSM

Gunakan perintah berikut untuk memulai klien AWS CloudHSM.

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
```

Mulai key_mgmt_util

Setelah Anda memulai klien AWS CloudHSM, 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

```
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.

Penggunaan dasar key_mgmt_util

Lihat topik berikut untuk penggunaan dasar alat key_mgmt_util.

Topik

- Masuk ke HSM
- Keluar dari HSM
- Hentikan key_mgmt_util

Masuk ke HSM

Gunakan perintah loginHSM untuk masuk ke HSM. Perintah berikut masuk sebagai pengguna kripto (CU) bernama `example_user`. Output menunjukkan login sukses untuk ketiga HSM di klaster.

```
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 HSM

Gunakan perintah logoutHSM untuk log out dari HSM.

```
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 key_mgmt_util

Gunakan perintah exit untuk menghentikan key_mgmt_util.

```
Command:  exit
```

Instal dan konfigurasikan AWS CloudHSM klien (Linux)

Untuk berinteraksi dengan HSM di AWS CloudHSM cluster Anda, Anda memerlukan perangkat lunak AWS CloudHSM klien untuk Linux. 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 (Windows).

Tugas

- Instal AWS CloudHSM klien dan alat baris perintah
- Edit konfigurasi klien

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
```

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 perintah describe-clusters CLI. Dalam output perintah, alamat IP HSM adalah nilai bidang `EniIp`. Jika Anda memiliki lebih dari satu HSM, pilih alamat IP untuk salah satu HSM; 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 Mengaktifkan klaster.

Instal dan konfigurasikan AWS CloudHSM klien (Windows)

Untuk bekerja dengan HSM di klaster AWS CloudHSM Anda pada Windows, Anda memerlukan perangkat lunak klien AWS CloudHSM untuk Windows. 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 yang terbaru (AWSCloudHSMClient-latest.msi) dari halaman unduhan.
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:

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

7. Kunjungi Mengaktifkan klaster.

Catatan:

- Jika Anda memperbarui klien, file konfigurasi yang ada dari instalasi sebelumnya tidak ditimpa.
- Penginstal klien AWS CloudHSM untuk Windows secara otomatis mendaftarkan API Kriptografi: Next Generation (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 konfigurasikan AWS CloudHSM klien (Linux).

key_mgmt_util referensi perintah

Parameter alat baris perintah key_mgmt_util membantu Anda untuk mengelola kunci dalam HSM di klaster 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 key_mgmt_util. Untuk membantu menafsirkan atribut kunci, lihat Referensi Atribut Kunci. Untuk informasi tentang alat baris perintah cloudhsm_mgmt_util, yang mencakup perintah untuk mengelola HSM dan pengguna di klaster Anda, lihat Utilitas Manajemen CloudHSM (CMU).

Sebelum Anda menjalankan perintah key_mgmt_util, Anda harus memulai key_mgmt_util dan masuk ke HSM sebagai pengguna kripto (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
aesWrapUnwrap	Mengenkripsi dan mendekripsi isi kunci dalam sebuah file.
DeleteKey	Menghapus kunci dari HSM.
Kesalahan2String	Mendapatkan kesalahan yang sesuai dengan kode kesalahan heksadesimalkey_mgmt_util.
keluar	Keluar dari key_mgmt_util.
exportPrivateKey	Mengekspor salinan kunci pribadi dari HSM ke file pada disk.

Perintah	Deskripsi
exportPubKey	Ekspor salinan kunci publik dari HSM ke file.
exSymKey	Ekspor salinan plaintext dari kunci simetris dari HSM ke file.
extractMaskedObject	Ekstrak kunci dari HSM sebagai file objek tertutup.
FindKey	Cari kunci berdasarkan nilai atribut kunci.
findSingleKey	Memverifikasi bahwa kunci ada pada semua HSM di klaster.
GendsA KeyPair	Menghasilkan pasangan kunci Algoritme Tanda Tangan Digital (DSA) di HSM Anda.
GeneCC KeyPair	Menghasilkan sebuah pasangan kunci Kriptografi Kurva Elipsis (ECC) di HSM Anda.
GenRSA KeyPair	Menghasilkan sebuah pasangan kunci asimetri RSA dalam HSM anda.
genSymKey	Menghasilkan kunci simetris di HSM Anda
getAttribute	Mendapat nilai atribut untuk kunci AWS CloudHSM dan menuliskannya ke file.
getCaviumPrivKunci	Membuat kunci privat versi format PEM palsu dan mengekspornya ke file.
GetCert	Mengambil sertifikat partisi HSM dan menyimpannya ke file.
getKeyInfo	Mendapat ID pengguna HSM dari pengguna yang dapat menggunakan kunci. Jika kuncinya dikendalikan kuorum, kunci mendapat jumlah pengguna dalam kuorum.

Perintah	Deskripsi
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.
insertMaskedObject	Menyisipkan objek tertutup dari file pada disk ke dalam HSM yang terkandung dalam klaster terkait dengan klaster asal objek. Klaster terkait adalah setiap klaster yang dihasilkan dari cadangan dari klaster asal.
???	Menentukan apakah file yang diberikan berisi kunci privat nyata atau kunci PEM palsu.
ListaTtributes	Mendaftar atribut dari kunci AWS CloudHSM dan konstanta yang mewakilinya.
DaftarPengguna	Mendapatkan pengguna di HSM, jenis pengguna dan ID mereka, dan atribut lainnya.
LoginHSM dan LogouthSM	Masuk dan keluar dari HSM dalam klaster.
SetAttribute	Mengonversi kunci sesi untuk kunci persisten.
tanda	Hasilkan tanda tangan untuk file menggunakan kunci privat yang dipilih.
unWrapKey	Mengimpor kunci yang dibungkus (terenkripsi) dari file ke HSM.
memverifikasi	Memverifikasi apakah kunci yang diberikan digunakan untuk menandatangani file yang diberikan.

Perintah	Deskripsi
WrapKey	Mengekspor salinan kunci terenkripsi dari HSM ke file.

aesWrapUnwrap

Perintah aesWrapUnwrap 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 Bungkus Kunci AES. 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 kripto (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 : Membungkus 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. Sebagai contoh, jika Anda mencoba untuk menggunakan imSymKeyperintah untuk mengimpor kunci terenkripsi, perintah ini mengembalikan kesalahan karena kunci terenkripsi tidak memiliki format yang diperlukan untuk kunci plaintext dari jenis tersebut.

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 handel kunci dari kunci AES pada HSM. Parameter ini diperlukan. Untuk menemukan handel kunci, gunakan perintah findKey.

Untuk membuat kunci pembungkus, gunakan genSymKeyuntuk 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

deleteKey

Perintah deleteKey di key_mgmt_util menghapus kunci dari HSM. 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 kripto (CU).

Sintaksis

```
deleteKey -h

deleteKey -k
```

Contoh

Contoh ini menunjukkan cara menggunakan deleteKey untuk menghapus kunci dari HSM Anda.

Example : Menghapus 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 : Menghapus 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.

Diperlukan: Ya

-k

Menentukan handel kunci dari kunci untuk dihapus. Untuk menemukan kunci handel dari kunci di HSM, gunakan findKey.

Diperlukan: Ya

Topik terkait

- findKey

Error2String

Perintah Error2String perintah di key_mgmt_util 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 kripto (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 : Mendapatkan 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 kripto (CU) yang bisa masuk ke key_mgmt_util.

```
        Command:  Error2String -r 0xdb

        Error Code db maps to HSM Error: Invalid User Type.
```

Example : Menemukan 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.

Diperlukan: Ya

-r

Menentukan kode kesalahan heksadesimal. Indikator heksadesimal `0x` diperlukan.

Diperlukan: Ya

keluar

Perintah exit di key_mgmt_util mengeluarkan 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.

Sintaksis

```
exit
```

Parameter

Tidak ada parameter untuk perintah ini.

Topik terkait

- Mulai key_mgmt_util

exportPrivateKey

exportPrivateKeyPerintah di key_mgmt_util mengekspor kunci privat asimetris dari HSM ke file. HSM tidak mengizinkan ekspor kunci langsung dalam cleartext. Perintah membungkus kunci pribadi menggunakan kunci pembungkus AES yang Anda tentukan, mendekripsi byte yang dibungkus, dan menyalin kunci privat cleartext ke file.

exportPrivateKeyPerintah 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 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` atribut1. 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 kripto (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 Privat

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

exportPubKey

Perintah exportPubKey di key_mgmt_util mengekspor kunci publik di 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 kripto (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 handel `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.

Diperlukan: Ya

`-k`

Menentukan handel kunci dari kunci publik yang akan diekspor.

Diperlukan: Ya

`-out`

Menentukan nama file tempat kunci publik yang diekspor akan ditulis.

Diperlukan: Ya

Topik terkait

- importPubKey
- Hasilkan Kunci

exSymKey

Perintah exSymKey dalam alat key_mgmt_util mengekspor salinan plaintext dari kunci simetris dari HSM dan menyimpannya dalam file pada disk. Untuk mengekspor salinan kunci yang dienkripsi (dibungkus), gunakan wrapKey. Untuk mengimpor kunci plaintext, seperti yang diekspor `exSymKey`, 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 privat, gunakan exportPrivateKey.

Sebelum Anda menjalankan perintah key_mgmt_util, Anda harus memulai key_mgmt_util dan masuk ke HSM sebagai pengguna kripto (CU).

Sintaksis

```
exSymKey -h
```

```
exSymKey -k <key-to-export>
         -w <wrapping-key>
         -out <key-file>
         [-m 4]
         [-wk <unwrapping-key-file> ]
```

Contoh

Contoh ini menunjukkan cara menggunakan exSymKey untuk mengekspor kunci simetris yang Anda miliki dari HSM Anda.

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 sesi saja

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 menggunakan 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 mengekspor 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 mengekspor 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 Pembuka 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 (wordcount) alat, yang mengonfirmasi bahwa file 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 perintah imSymKey untuk mengimpor kunci AES dari file `aes256-forImport.key` 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.

Diperlukan: 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 perintah getKeyInfo.

Diperlukan: 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.

Diperlukan: 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.

Diperlukan: Ya

- m

Menentukan mekanisme pembungkus. Satu-satunya nilai yang valid adalah `4`, yang mewakili mekanisme `NIST_AES_WRAP`.

Diperlukan: 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.

Diperlukan: Tidak

Default: Gunakan kunci pembungkus pada HSM untuk membuka.

Topik terkait

- genSymKey
- imSymKey
- wrapKey

extractMaskedObject

Perintah extractMaskedObject di key_mgmt_util mengekstraksi kunci dari HSM dan menyimpannya ke sebuah file sebagai objek tertutup. 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 klaster 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 Tersembunyi

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.

Diperlukan: Ya

`-o`

Menentukan handel kunci untuk mengekstraksi sebagai objek tertutup.

Diperlukan: Ya

`-out`

Menentukan nama file yang objek tertutup akan disimpan.

Diperlukan: Ya

Topik terkait

- insertMaskedObject
- syncKey
- Menyalin Cadangan Lintas Wilayah
- MembuatAWS CloudHSMKlaster dari Cadangan Sebelumnya

findKey

Gunakan findKey di key_mgmt_util untuk mencari kunci dengan nilai-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 kripto (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 ini menunjukkan cara menggunakan findKey untuk menemukan dan mengidentifikasi kunci di HSM Anda.

Example : Cari Semua Kunci

Perintah ini menemukan semua kunci untuk pengguna saat ini di HSM. Output termasuk kunci yang dimiliki pengguna dan saham, dan semua kunci publik di HSM.

Untuk mendapatkan atribut kunci dengan handel kunci tertentu, gunakan getAttribute. Untuk menentukan apakah pengguna saat ini memiliki atau berbagi kunci tertentu, gunakan getKeyInfoatau findAllKeysdi 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 Berdasarkan 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 ID pengguna HSM yang dipisahkan koma, seperti `-u 3` atau `-u 4,7`. Untuk menemukan ID pengguna semua pengguna, gunakan listUsers.

Bila Anda menentukan satu ID pengguna, findKey mengembalikan kunci untuk pengguna tersebut. Bila Anda menentukan beberapa ID pengguna, findKey mengembalikan kunci yang semua pengguna tertentu dapat menggunakannya.

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 kripto (CO) dapat menggunakan findAllKeysdi 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 mencocokkan 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 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 pemeriksaan 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
- findAllKeysdi cloudhsm_mgmt_util
- Referensi Atribut Kunci

findSingleKey

Perintah findSingleKey di alat key_mgmt_util memverifikasi bahwa kunci ada pada semua HSM di klaster.

Sebelum Anda menjalankan perintah key_mgmt_util, Anda harus memulai key_mgmt_util dan masuk ke HSM sebagai pengguna kripto (CU).

Sintaksis

```
findSingleKey -h

findSingleKey -k <key-handle>
```

Contoh

Example

Perintah ini memverifikasi kunci 252136 ada pada ketiga HSM di klaster.

```
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.

Diperlukan: Ya

-k

Menentukan handel kunci dari satu kunci di HSM. Parameter ini diperlukan.

Untuk menemukan handel kunci, gunakan perintah findKey.

Diperlukan: Ya

Topik terkait

- findKey
- getKeyInfo
- getAttribute

GenDSA KeyPair

Perintah genDSAKeyPair dalam alat key_mgmt_util menghasilkan pasangan kunci Algoritma Tanda Tangan Digital (DSA) di 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 kripto (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 12dan 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 getKeyInfopada 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 HSM tempat kunci disinkronkan sebelum nilai parameter `-timeout` kedaluwarsa. Jika kunci tidak disinkronkan ke jumlah tertentu server dalam waktu yang ditentukan, kunci tidak dibuat.

AWS CloudHSM secara otomatis menyinkronkan setiap kunci untuk setiap HSM di klaster. Untuk mempercepat proses Anda, tetapkan nilai `min_srv` menjadi kurang dari jumlah HSM di klaster dan menetapkan nilai waktu habis rendah. 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 diesktrak. 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 yang akan disinkronkan dengan jumlah HSM yang 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 pengguna kripto (CU) HSM lainnya untuk menggunakan kunci privat dalam operasi kriptografi. Kunci publik dapat digunakan oleh pengguna tanpa berbagi.

Ketik daftar ID pengguna HSM yang dipisahkan koma, seperti `-u 5,6`. Jangan sertakan ID pengguna HSM dari pengguna saat ini. Untuk menemukan ID pengguna HSM dari pengguna di HSM Anda, 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 klaster berjalan belum dirusak.

Default: Tidak ada pemeriksaan pengesahan.

Wajib: Tidak

Topik terkait

- GenRSA KeyPair
- genSymKey
- GeneCC KeyPair

GeneCC KeyPair

Perintah `genECCKeyPair` di alat key_mgmt_util menghasilkan sebuah pasangan kunci Kriptografi Kurva Elipsis (ECC) di HSM Anda. Ketika menjalankan perintah `genECCKeyPair`, Anda harus menentukan pengenal 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 handel kunci yang HSM tetapkan ke kunci ECC 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 kripto (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 HSM Anda.

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 HSM mode FIPS, perintah gagal. Pesan melaporkan bahwa server di klaster tidak tersedia, tapi ini tidak selalu menunjukkan masalah dengan HSM di klaster.

```
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 pengenal untuk kurva elips. Masukkan pengenal.

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 HSM tempat kunci disinkronkan sebelum nilai parameter `-timeout` kedaluwarsa. Jika kunci tidak disinkronkan ke jumlah tertentu server dalam waktu yang ditentukan, kunci tidak dibuat.

AWS CloudHSM secara otomatis menyinkronkan setiap kunci untuk setiap HSM di klaster. Untuk mempercepat proses Anda, tetapkan nilai `min_srv` menjadi kurang dari jumlah HSM di klaster dan menetapkan nilai waktu habis rendah. 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 diesktrak. 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 yang akan disinkronkan dengan jumlah HSM yang 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 pengguna kripto (CU) HSM lainnya untuk menggunakan kunci privat dalam operasi kriptografi. Kunci publik dapat digunakan oleh pengguna tanpa berbagi.

Ketik daftar ID pengguna HSM yang dipisahkan koma, seperti `-u 5,6`. Jangan sertakan ID pengguna HSM dari pengguna saat ini. Untuk menemukan ID pengguna HSM dari pengguna di HSM Anda, 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 klaster berjalan belum dirusak.

Default: Tidak ada pemeriksaan pengesahan.

Wajib: Tidak

Topik terkait

- genSymKey
- GenRSA KeyPair
- GenDSA KeyPair

GenRSA KeyPair

Perintah genRSAKeyPair di alat key_mgmt_util menghasilkan sebuah pasangan kunci 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 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 kripto (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 ini menunjukkan cara menggunakan genRSAKeyPair untuk membuat pasangan kunci asimetris di HSM Anda.

Example : Buat dan periksa key pair RSA

Perintah ini menciptakan key pair RSA dengan modulus 2048-bit dan eksponen 65537. Output menunjukkan bahwa handel kunci publik adalah 2100177 dan handel 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 Referensi Atribut Kunci.

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
1c0fe6e3b61aa83c205280119452868f66541bbbffacbbe787b8284fc81deaeef2b8ec0ba25a077d
6983c77a1de7b17cbe8e15b203868704c6452c2810344a7f2736012424cf0703cf15a37183a1d2d0
97240829f8f90b063dd3a41171402b162578d581980976653935431da0c1260bfe756d85dca63857
d9f27a541676cb9c7def0ef6a2a89c9b9304bcac16fdf8183c0a555421f9ad5dfeb534cf26b65873
970cdf1a07484f1c128b53e10209cc6f7ac308669112968c81a5de408e7f644fe58b1a9ae1286fec
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 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 HSM tempat kunci disinkronkan sebelum nilai parameter `-timeout` kedaluwarsa. Jika kunci tidak disinkronkan ke jumlah tertentu server dalam waktu yang ditentukan, kunci tidak dibuat.

AWS CloudHSM secara otomatis menyinkronkan setiap kunci untuk setiap HSM di klaster. Untuk mempercepat proses Anda, tetapkan nilai `min_srv` menjadi kurang dari jumlah HSM di klaster dan menetapkan nilai waktu habis rendah. 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 diesktrak. 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 yang akan disinkronkan dengan jumlah HSM yang 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 pengguna kripto (CU) HSM lainnya untuk menggunakan kunci privat dalam operasi kriptografi. Kunci publik dapat digunakan oleh pengguna tanpa berbagi.

Ketik daftar ID pengguna HSM yang dipisahkan koma, seperti `-u 5,6`. Jangan sertakan ID pengguna HSM dari pengguna saat ini. Untuk menemukan ID pengguna HSM dari pengguna di HSM Anda, 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 klaster berjalan belum dirusak.

Default: Tidak ada pemeriksaan pengesahan.

Wajib: Tidak

Topik terkait

- genSymKey
- GenDSA KeyPair
- GeneCC KeyPair

genSymKey

Perintah genSymKey di alat key_mgmt_util menghasilkan kunci simetris di 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 kripto (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 ini menunjukkan cara menggunakan genSymKey untuk membuat kunci simetris di HSM Anda.

> **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 handel 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 untuk disinkronkan ke semua HSM di klaster. 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 ke jumlah minimum tertentu HSM sebelum batas waktu berakhir, kunci 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 handel 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 Referensi Atribut Kunci.

```
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. Kunci RC4 tidak valid pada HSM mode FIPS

- 21: Tiga DES (3DES). Dilarang setelah 2023 untuk kepatuhan FIPS sesuai panduan NIST. 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 klaster berjalan belum dirusak.

Default: Tidak ada pemeriksaan pengesahan.

Wajib: Tidak

-id

Menentukan pengenal yang ditetapkan pengguna untuk kunci. Ketik string yang unik dalam klaster. Default-nya adalah string kosong.

Default: Tidak ada nilai ID.

Wajib: Tidak

-min_srv

Menentukan jumlah minimum HSM tempat kunci disinkronkan sebelum nilai parameter `-timeout` kedaluwarsa. Jika kunci tidak disinkronkan ke jumlah tertentu server dalam waktu yang ditentukan, kunci tidak dibuat.

AWS CloudHSM secara otomatis menyinkronkan setiap kunci untuk setiap HSM di klaster. Untuk mempercepat proses Anda, tetapkan nilai `min_srv` menjadi kurang dari jumlah HSM di klaster dan menetapkan nilai waktu habis rendah. 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 yang akan disinkronkan dengan jumlah HSM yang 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 kripto HSM (CU) lainnya untuk menggunakan kunci ini dalam operasi kriptografi.

Ketik daftar ID pengguna HSM yang dipisahkan koma, seperti `-u 5,6`. Jangan sertakan ID pengguna HSM dari pengguna saat ini. Untuk menemukan ID pengguna HSM dari pengguna di HSM Anda, 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
- GenRSA KeyPair
- GenDSA KeyPair
- GeneCC KeyPair

- setAttribute

getAttribute

Perintah getAttribute di key_mgmt_util menulis satu atau semua nilai atribut untuk kunci AWS CloudHSM ke file. 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 mendapat satu nilai atribut kunci dari semua HSM dalam sebuah klaster, dan menuliskannya 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 Referensi Atribut Kunci.

Sebelum Anda menjalankan perintah key_mgmt_util, Anda harus memulai key_mgmt_util dan Masuk ke HSM sebagai pengguna kripto (CU).

Sintaksis

```
getAttribute -h

getAttribute -o <key handle>
             -a <attribute constant>
             -out <file>
```

Contoh

Contoh ini menunjukkan cara menggunakan getAttribute untuk mendapatkan atribut kunci di HSM Anda.

Example : Dapatkan jenis kunci

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 Referensi Atribut Kunci.

```
Command: listAttributes

Description
===========
The following are all of the possible attribute values for getAttributes.

      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 menulisnya 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 Referensi Atribut Kunci.

```
$  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.

Diperlukan: 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.

Diperlukan: 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 Referensi Atribut Kunci.

Diperlukan: 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.

Diperlukan: Ya

Topik terkait

- getAttribute di cloudhsm_mgmt_util
- listAttributes
- setAttribute
- findKey
- Referensi Atribut Kunci

getCaviumPrivKunci

Perintah getCaviumPrivKey di key_mgmt_util mengekspor kunci privat dari HSM dalam format PEM palsu. File PEM palsu, yang tidak mengandung materi kunci privat yang sebenarnya tetapi mereferensikan kunci privat di HSM, kemudian dapat digunakan untuk membangun pembongkaran SSL/TLS dari server web Anda ke AWS CloudHSM. Untuk informasi lebih lanjut, lihat Pembongkaran SSL/TLS di Linux.

Sebelum Anda menjalankan perintah key_mgmt_util, Anda harus memulai key_mgmt_util dan masuk ke HSM sebagai pengguna kripto (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
- pembkaran SSL/TLS pada Linux

getCert

Perintah getCert di key_mgmt_util mengambil sertifikat partisi 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 kripto (CU).

Sintaksis

```
getCert -h

getCert -f <file-name>
        -t <certificate-type>
```

Contoh

Contoh ini menunjukkan cara menggunakangetCertuntuk mengambil sebuah klaster pelanggan root sertifikat dan menyimpannya sebagai file.

Example : Mengambil 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.

Diperlukan: Ya

-f

Menentukan nama file tempat sertifikat disimpan.

Diperlukan: 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)

Diperlukan: Ya

Topik terkait

- Verifikasi Identitas HSM
- getCert (di cloudhsm_mgmt_util)

getKeyInfo

Perintah getKeyInfo di key_mgmt_util mengembalikan ID pengguna HSM dari pengguna yang dapat menggunakan kunci, termasuk pemilik dan pengguna kripto (CU) dengan siapa kunci dibagi. 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 ID pengguna HSM dari pengguna di HSM Anda, 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 kripto (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 privat (20), ini mengembalikan pemilik kunci (3) dan pengguna kripto (CU) 4 dan 5, dengan siapa kuncinya 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 Autentikasi 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 genRSAKeyPair untuk membuat pasangan kunci RSA yang dibagi 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.

Diperlukan: 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.

Diperlukan: Ya

Topik terkait

- getKeyInfo di cloudhsm_mgmt_util
- listUsers
- findKey
- findAllKeys di cloudhsm_mgmt_util

help

Perintah help di key_mgmt_util 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
createPublicKey        Creates an RSA public key
importPubKey           Imports RSA/DSA/EC Public key
exportPubKey           Exports RSA/DSA/EC Public key
importPrivateKey       Imports RSA/DSA/EC private key
exportPrivateKey       Exports RSA/DSA/EC private key
imSymKey               Imports a Symmetric key
exSymKey               Exports a Symmetric key
wrapKey                Wraps a key from from HSM using the specified handle
unWrapKey              UnWraps a key into HSM using the specified handle

     Key Management Commands
```

```
    deleteKey               Delete Key
    setAttribute            Sets an attribute of an object
    getKeyInfo              Get Key Info about shared users/sessions
    findKey                 Find Key
    findSingleKey           Find single Key
    getAttribute            Reads an attribute from an object

         Certificate Setup Commands
    getCert                 Gets Partition Certificates stored on HSM

         Key Transfer Commands
    insertMaskedObject      Inserts a masked object
    extractMaskedObject     Extracts a masked object

         Management Crypto Commands
    sign                    Generates a signature
    verify                  Verifies a signature
    aesWrapUnwrap           Does NIST AES Wrap/Unwrap

         Helper Commands
    Error2String            Converts Error codes to Strings
                            save key handle in fake PEM format
    getCaviumPrivKey        Saves an RSA private key handle
                            in fake PEM format
    IsValidKeyHandlefile    Checks if private key file has
                            an HSM key handle or a real key
    listAttributes          List all attributes for getAttributes
    listECCCurveIds         List HSM supported ECC CurveIds
```

Parameter

Tidak ada parameter untuk perintah ini.

Topik terkait

- loginHSM dan logoutHSM

importPrivateKey

importPrivateKeyPerintah di key_mgmt_util mengimpor kunci pribadi asimetris dari file ke 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 kripto (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 HSM tempat kunci yang diimpor disinkronkan sebelum nilai parameter `-timeout` kedaluwarsa. Jika kunci tidak disinkronkan ke jumlah tertentu server dalam waktu yang ditentukan, kunci tidak dibuat.

AWS CloudHSM secara otomatis menyinkronkan setiap kunci untuk setiap HSM di klaster. Untuk mempercepat proses Anda, tetapkan nilai `min_srv` menjadi kurang dari jumlah HSM di klaster dan menetapkan nilai waktu habis rendah. Namun, perhatikan bahwa beberapa permintaan mungkin tidak menghasilkan kunci.

Default: 1

Wajib: Tidak

`-timout`

Menentukan jumlah detik untuk menunggu kunci untuk disinkronkan di HSM ketika parameter `min-serv` 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 pengguna kripto HSM (CU) lainnya untuk menggunakan kunci yang diimpor dalam operasi kriptografi.

Masukkan daftar ID pengguna HSM yang dipisahkan koma, seperti `-u 5,6`. Jangan sertakan ID pengguna HSM dari pengguna saat ini. Untuk menemukan ID pengguna HSM CU pada 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

importPubKey

Perintah importPubKey di key_mgmt_util mengimpor kunci publik format PEM ke dalam 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 kripto (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.

Diperlukan: Ya

`-l`

Menentukan label kunci privat yang ditetapkan pengguna.

Diperlukan: Ya

`-f`

Menentukan nama file kunci untuk diimpor.

Diperlukan: Ya

`-sess`

Menunjuk kunci yang diimpor sebagai kunci sesi.

Default: Kunci yang diimpor disimpan sebagai kunci persisten (token) di klaster.

Diperlukan: Tidak

`-id`

Menentukan ID kunci yang akan diimpor.

Default: Tidak ada nilai ID.

Diperlukan: Tidak

`-min_srv`

Menentukan jumlah minimum HSM tujuan kunci diimpor disinkronkan sebelum nilai parameter `-timeout` kedaluwarsa. Jika kunci tidak disinkronkan ke jumlah tertentu server dalam waktu yang ditentukan, kunci tidak dibuat.

AWS CloudHSM secara otomatis menyinkronkan setiap kunci untuk setiap HSM di klaster. Untuk mempercepat proses Anda, tetapkan nilai `min_srv` menjadi kurang dari jumlah HSM di klaster dan menetapkan nilai waktu habis rendah. Namun, perhatikan bahwa beberapa permintaan mungkin tidak menghasilkan kunci.

Default: 1

Diperlukan: Tidak

`-timeout`

Menentukan jumlah detik untuk menunggu kunci untuk disinkronkan di HSM ketika parameter `min-serv` disertakan. Jika tidak ada nomor yang ditentukan, polling berlanjut selamanya.

Default: Tidak ada batas

Diperlukan: Tidak

Topik terkait

- exportPubKey
- Hasilkan Kunci

imSymKey

Perintah imSymKey dalam alat key_mgmt_util mengimpor salinan teks terang kunci simetris dari file ke 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 dituliskan perintah exSymKey, ke file.

Selama proses impor, imSymKey menggunakan kunci AES yang Anda pilih (kunci pembungkus) untuk membungkus (mengenkripsi) 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 unWrapKeyperintah.

Selain itu, perintah imSymKey mengimpor hanya kunci simetris. Untuk mengimpor kunci publik, gunakan importPubKey. Untuk mengimpor kunci pribadi, gunakan importPrivateKeyatau 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 kripto (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 ini menunjukkan cara menggunakan imSymKey untuk mengimpor kunci simetris ke dalam HSM Anda.

Example : Impor AES kunci simetris

Contoh ini menggunakan imSymKey untuk mengimpor kunci simetris AES ke dalam HSM.

Perintah pertama menggunakan OpenSSL untuk menghasilkan kunci simetris AES 256-bit acak. Menyimpan kunci dalam file `aes256.key`.

```
$  openssl rand -out aes256-forImport.key 32
```

Perintah kedua menggunakan imSymKey untuk mengimpor kunci AES dari file `aes256.key` ke dalam HSM. 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 handel 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 : Memindahkan kunci simetris antar klaster

Contoh ini menunjukkan cara menggunakan exSymKeydanimSymKey memindahkan kunci AES teks jelas antara klaster. Anda mungkin menggunakan proses seperti ini untuk membuat pembungkus AES yang ada di HSM kedua klaster. Setelah kunci pembungkus bersama sudah terpasang, Anda dapat menggunakan wrapKey dan unWrapKeymemindahkan kunci terenkripsi antara klaster.

Pengguna CU yang melakukan operasi ini harus memiliki izin untuk masuk ke HSM di kedua klaster.

Perintah pertama menggunakan exSymKeyuntuk mengekspor kunci 14, kunci AES 32-bit, dari klaster 1 `keaes.key` file. Perintah menggunakan kunci 6, kunci AES pada HSM di klaster 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 klaster 2 dan menjalankan perintah imSymKey untuk mengimpor kunci dalam file `aes.key` ke dalam HSM di klaster 2. Perintah ini menggunakan kunci 252152, kunci AES pada HSM di klaster 2, sebagai kunci pembungkus.

Karena kunci pembungkus yang exSymKeydanimSymKey gunakan membungkus dan segera membuka kunci target, kunci pembungkus pada kelompok 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 HSM klaster kedua, Anda sekarang dapat berbagi kunci yang dienkripsi antara klaster tanpa pernah mengekspos kunci teks terang. Misalnya, Anda dapat menggunakan perintah `wrapKey` dengan kunci pembungkus 14 untuk mengekspor 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 pengenal 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 handel 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 klaster berjalan belum dirusak.

Default: Tidak ada pemeriksaan pengesahan.

Diperlukan: Tidak

-f

Menentukan file yang berisi kunci untuk diimpor.

File harus berisi salinan teks terang dari kunci AES atau Triple DES dengan panjang yang ditentukan. Kunci RC4 dan DES tidak valid pada HSM mode FIPS.

- AES: 16, 24 atau 32 byte
- Triple DES (3DES): 24 byte

Diperlukan: Ya

-h

Menampilkan bantuan untuk perintah.

Diperlukan: Ya

-id

Menentukan pengenal yang ditetapkan pengguna untuk kunci. Ketik string yang unik dalam klaster. Default-nya adalah string kosong.

Default: Tidak ada nilai ID.

Diperlukan: 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.

Diperlukan: Ya

-min_srv

Menentukan jumlah minimum HSM tempat kunci disinkronkan sebelum nilai parameter `-timeout` kedaluwarsa. Jika kunci tidak disinkronkan ke jumlah tertentu server dalam waktu yang ditentukan, kunci tidak dibuat.

AWS CloudHSM secara otomatis menyinkronkan setiap kunci untuk setiap HSM di klaster. Untuk mempercepat proses Anda, tetapkan nilai `min_srv` menjadi kurang dari jumlah HSM di klaster dan menetapkan nilai waktu habis rendah. Namun, perhatikan bahwa beberapa permintaan mungkin tidak menghasilkan kunci.

Default: 1

Diperlukan: 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.

Diperlukan: Tidak

-timeout

Menentukan berapa lama (dalam detik) perintah menunggu kunci yang akan disinkronkan dengan jumlah HSM yang 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.

Diperlukan: Tidak

-t

Menentukan jenis kunci simetris. Masukkan konstanta yang mewakili jenis kunci. Misalnya, untuk membuat kunci AES, masukkan `-t 31`.

Nilai yang valid:

- 21: Triple DES (3DES).
- 31: AES

Diperlukan: Ya

-u

Berbagi kunci yang Anda impor dengan pengguna tertentu. Parameter ini memberikan izin kepada pengguna kripto HSM (CU) lainnya untuk menggunakan kunci ini dalam operasi kriptografi.

Ketik satu ID atau daftar ID pengguna HSM yang dipisahkan koma, 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.

Diperlukan: 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, gunakangetAttribute 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.

Diperlukan: 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.

Diperlukan: Tidak

Topik terkait

- genSymKey
- exSymKey
- wrapKey
- unWrapKey
- exportPrivateKey
- exportPubKey

insertMaskedObject

Perintah insertMaskedObject di key_mgmt_util menyisipkan objek tertutup dari file ke HSM yang ditunjuk. 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 klaster 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 kripto (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 : Sisipkan Objek Tersembunyi

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.

Diperlukan: Ya

`-f`

Menentukan nama file dari objek tertutup untuk disisipkan.

Diperlukan: 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

Diperlukan: 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: Tidak ada batas

Diperlukan: Tidak

Topik terkait

- extractMaskedObject
- syncKey
- Menyalin Cadangan Lintas Wilayah
- MembuatAWS CloudHSMKlaster dari Cadangan Sebelumnya

IsValidKeyHandlefile

IsValidKeyHandlefilePerintah di key_mgmt_util digunakan 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 tersebut dapat digunakan untuk membangun pembongkaran SSL/TLS dari server web Anda ke AWS CloudHSM. Untuk informasi lebih lanjut, lihat Pembongkaran SSL/TLS di Linux.

> **Note**
>
> IsValidKeyHandlefilehanya berfungsi untuk kunci RSA.

Sebelum Anda menjalankan perintah key_mgmt_util, Anda harus memulai key_mgmt_util dan Masuk ke HSM sebagai pengguna kripto (CU).

Sintaks

```
IsValidKeyHandlefile -h
```

```
IsValidKeyHandlefile -f <rsa-private-key-file>
```

Contoh-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.

Diperlukan: Ya

`-f`

Menentukan file kunci pribadi RSA yang akan diperiksa untuk materi kunci yang valid.

Diperlukan: Ya

Topik terkait

- getCaviumPrivKunci
- SSL/TLS Offload di Linux

listAttributes

Perintah listAttributes di key_mgmt_util mendaftar atribut kunci AWS CloudHSM dan konstanta yang mewakilinya. Anda menggunakan konstanta ini untuk mengidentifikasi atribut di getAttribute dan perintah setAttribute. Untuk membantu menafsirkan atribut kunci, lihat Referensi Atribut Kunci.

Sebelum Anda menjalankan perintah key_mgmt_util, Anda harus memulai key_mgmt_util dan Masuk ke HSM sebagai pengguna kripto (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 Referensi Atribut Kunci.

Untuk mewakili semua atribut dalam getAttribute di key_mgmt_util, gunakan 512.

```
Command: listAttributes

Following are the possible attribute values for getAttributes:

      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 Kunci

listUsers

Perintah listUsers di key_mgmt_util mendapat pengguna di HSM, bersama dengan jenis penggunanya dan atribut lainnya.

Dalam key_mgmt_util, listUsers mengembalikan output yang mewakili semua HSM di klaster, 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 memiliki izin untuk dijalankan oleh pengguna kripto (CU). Perintah manajemen pengguna yang tersisa adalah bagian dari cloudhsm_mgmt_util. Perintah itu dijalankan oleh petugas kripto (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 kripto (CU).

Sintaksis

```
listUsers

listUsers -h
```

Contoh

Perintah ini berisi daftar pengguna HSM di klaster dan atribut mereka. 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 pasangan kunci untuk menandatangani token autentikasi 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

loginHSM dan logoutHSM

Perintah loginHSM dan logoutHSM di key_mgmt_util mengizinkan Anda untuk masuk dan keluar dari HSM dalam sebuah klaster. Setelah masuk ke HSM, Anda dapat menggunakan key_mgmt_util untuk melakukan berbagai operasi manajemen kunci, termasuk pembuatan kunci publik dan privat, sinkronisasi, dan pembungkusan.

Sebelum Anda menjalankan perintah key_mgmt_util, Anda harus memulai key_mgmt_util. Untuk mengelola kunci dengan key_mgmt_util, Anda harus masuk ke HSM sebagai pengguna kripto (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. Hal ini karena klien CloudHSM menyeimbangkan beban di berbagai HSM. 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 untuk masuk dan keluar dari HSM dalam sebuah klaster dengan perintah `loginHSM` dan `logoutHSM`

Example : Masuk ke HSM

Perintah ini memasukkan Anda ke HSM sebagai pengguna kripto (CU) dengan nama pengguna `example_user` dan kata sandi `aws`. Output menunjukkan bahwa Anda telah login ke semua HSM di klaster.

```
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 HSM.

```
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 HSM

Perintah ini mengeluarkan Anda dari HSM. Output menunjukkan bahwa Anda telah keluar dari semua HSM di klaster.

```
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

setAttribute

Perintah setAttribute di key_mgmt_util mengonversi kunci yang berlaku hanya dalam 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 kripto (CU).

Sintaksis

```
setAttribute -h

setAttribute -o <object handle>
             -a 1
```

Contoh

Contoh ini menunjukkan cara mengonversi kunci sesi menjadi kunci persisten.

Perintah pertama menggunakan parameter `-sess` dari genSymKey untuk membuat kunci AES 192-bit yang hanya berlaku dalam sesi saat ini. Output menunjukkan bahwa handel 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 Referensi Atribut Kunci.

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.

Diperlukan: 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.

Diperlukan: 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.

Diperlukan: Ya

Topik terkait

- setAttribute di cloudhsm_mgmt_util
- getAttribute
- listAttributes
- Referensi Atribut Kunci

tanda tangan

Perintah sign di key_mgmt_util menggunakan kunci privat yang dipilih untuk menghasilkan tanda tangan untuk sebuah 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 kripto (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 handel `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.

Diperlukan: Ya

`-k`

Handel kunci privat yang akan digunakan untuk penandatanganan.

Diperlukan: 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

Mekanisme Penandatanganan	Integer yang Sesuai
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

Diperlukan: Ya

`-out`

Nama file tempat file yang ditandatangani akan disimpan.

Diperlukan: Ya

Topik terkait

- memverifikasi
- importPrivateKey
- genRSAKeyPair

- genECCKeyPair
- genSymKey
- Hasilkan Kunci

unWrapKey

Perintah unWrapKey di alat key_mgmt_util mengimpor kunci simetris atau privat dibungkus (dienkripsi) 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 bekerja 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 handel 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 kripto (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 ini menunjukkan cara menggunakan unWrapKey untuk mengimpor kunci yang dibungkus dari file ke HSM. 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 handel 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 yan diperlukan: label kunci

(unwrapped3DES), kelas kunci (4), dan jenis kunci (21). Output menunjukkan bahwa kunci dalam file itu dibuka dan diimpor, dan bahwa handel 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 genSymKeyuntuk menghasilkan kunci AES (tipe 31) atau GenRSA KeyPair 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 HSM tempat kunci disinkronkan sebelum nilai parameter `-timeout` kedaluwarsa. Jika kunci tidak disinkronkan ke jumlah tertentu server dalam waktu yang ditentukan, kunci tidak dibuat.

AWS CloudHSM secara otomatis menyinkronkan setiap kunci untuk setiap HSM di klaster. Untuk mempercepat proses Anda, tetapkan nilai `min_srv` menjadi kurang dari jumlah HSM di klaster dan menetapkan nilai waktu habis rendah. Namun, perhatikan bahwa beberapa permintaan mungkin tidak menghasilkan kunci.

Default: 1

Wajib: Tidak

-timeout

Menentukan berapa lama (dalam detik) perintah menunggu kunci yang akan disinkronkan dengan jumlah HSM yang 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.

Diperlukan: Tidak

-nex

Membuat kunci tidak dapat diekstrak. Kunci yang dihasilkan tidak dapat diekspor dari HSM.

Default: Kunci dapat diekstrak.

Diperlukan: 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

Diperlukan: 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 = modulusLengthIn Bytes- (2* Bytes) -2. hashLengthIn
Saat menggunakan mekanisme pembungkus RSA_PKCS, ukuran kunci maksimum yang dapat Anda bungkus ditentukan oleh modulus kunci RSA sebagai berikut: Ukuran kunci maksimum = (Bytes -11). modulusLengthIn

-t

Algoritma hash	Nilai
SHA1	2
SHA256	3
SHA384	4
SHA512	5
SHA224 (berlaku untuk mekanisme RSA_AES dan RSA_0AEP)	6

Diperlukan: 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.

  Diperlukan: Tidak

[1] Dilarang setelah 2023 untuk kepatuhan FIPS sesuai panduan NIST. Lihat Kepatuhan FIPS 140: Penutupan Mekanisme 2024 untuk detail.

Topik terkait

- WrapKey
- exSymKey
- imSymKey

verifikasi

Perintah verify di key_mgmt_util menegaskan apakah file telah ditandatangani oleh kunci yang diberikan atau belum. 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 operasi sign.

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 kripto (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 `hardwarCert.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.

Diperlukan: Ya

-s

Nama file yang akan ditandatangani.

Memerlukan: Ya

-k

Handel kunci publik yang diduga digunakan untuk menandatangani file.

Diperlukan: 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

Diperlukan: Ya

Topik terkait

- tanda
- getCert
- Hasilkan Kunci

wrapKey

Perintah wrapKey di key_mgmt_util mengekspor salinan kunci simetris atau privat terenkripsi dari 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 kripto (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 exSymKeyatau exportPrivateKeysesuai kebutuhan. aesWrapUnwrapPerintah 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 kripto (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 Referensi Atribut Kunci.

Anda hanya dapat mengekspor kunci yang Anda miliki. Untuk menemukan pemilik kunci, gunakan getKeyInfoperintah.

Diperlukan: 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 genSymKeyuntuk menghasilkan kunci AES (tipe 31) atau GenRSA KeyPair 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

Diperlukan: Ya

> **Note**
>
> Saat menggunakan mekanisme RSA_0AEP pembungkus, ukuran kunci maksimum yang dapat Anda bungkus ditentukan oleh modulus kunci RSA dan panjang hash yang ditentukan sebagai berikut: Ukuran kunci maksimum = (modulusLengthInBytes-2* Bytes-2). hashLengthIn
> Saat menggunakan mekanisme pembungkus RSA_PKCS, ukuran kunci maksimum yang dapat Anda bungkus ditentukan oleh modulus kunci RSA sebagai berikut: Ukuran kunci maksimum = (Bytes -11). 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_0AEP)	6

Diperlukan: 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.

Diperlukan: 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.

Diperlukan: Tidak

[1] Dilarang setelah 2023 untuk kepatuhan FIPS sesuai panduan NIST. Lihat Kepatuhan FIPS 140: Penutupan Mekanisme 2024 untuk detail.

Topik terkait

- exSymKey
- imSymKey
- unWrapKey

Referensi Atribut Kunci

Perintah key_mgmt_util menggunakan konstanta untuk mewakili atribut kunci dalam 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 bisa diekspor dari HSM.

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 HSM. 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 ECPoint ANSI X9.62 “Q” dalam 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_EXTRACTABLE	356	0: Salah. 1: Benar. Kunci tidak dapat diekspor dari HSM.
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 templat 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 pemeriksaan 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.

AWS CloudHSM SDK Klien

Gunakan SDK Klien untuk melepaskan operasi kriptografi dari platform atau aplikasi berbasis bahasa ke modul keamanan perangkat keras (HSM).

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, lihatPlatform yang didukung SDK Klien 5.

Untuk informasi tentang penggunaan SDK Klien 3, lihatSDK Klien Sebelumnya (SDK Klien 3).

the section called “Pustaka PKCS #11”

PKCS #11 adalah standar untuk melakukan operasi kriptografi pada modul keamanan perangkat keras (HSM). 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 JCE”

Penyedia AWS CloudHSM JCE sesuai dengan Java Cryptographic Architecture (JCA). Penyedia memungkinkan Anda untuk melakukan operasi kriptografi pada HSM.

the section called “Penyedia KSP dan CNG”

AWS CloudHSM Klien untuk Windows termasuk penyedia CNG dan KSP. Saat ini, hanya Client SDK 3 yang mendukung penyedia CNG dan KSP.

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 pastikan platform yang Anda inginkan muncul di bagian dasar untuk SDK, kemudian 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 Platform yang didukung SDK Klien 3

Klien SDK 5 tidak memerlukan daemon klien.

Dukungan Linux untuk Client SDK 5

Platform yang didukung	X86_64 Arsitektur	Arsitektur ARM
Amazon Linux 2	Ya	Ya
Amazon Linux 2023	Ya	Ya
CentOS 7 (7,8+)	Ya	Tidak
Perusahaan Topi Merah Linux 7 (7.8+)	Ya	Tidak
Perusahaan Topi Merah Linux 8 (8.3+)	Ya	Tidak
Perusahaan Topi Merah Linux 9 (9.2+)	Ya	Ya
Ubuntu 20.04 LTS	Ya	Tidak
Ubuntu 22.04 LTS	Ya	Ya

Catatan: SDK 5.4.2 adalah rilis terakhir yang memberikan dukungan platform CentOS 8. Untuk informasi lebih lanjut, lihat situs web CentOS.

Dukungan Windows untuk Client SDK 5

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

Dukungan tanpa server untuk Klien SDK 5

- AWS Lambda
- Docker/ECS

Dukungan komponen

Pustaka PKCS #11

Pustaka PKCS #11 adalah komponen lintas-platform yang cocok dengan dukungan dasar Klien SDK 5 Linux dan Windows. Untuk informasi lebih lanjut, lihat the section called “Dukungan Linux untuk Client SDK 5” dan the section called “Dukungan Windows untuk Client SDK 5”.

OpenSSL Dynamic Engine

OpenSSL Dynamic Engine adalah komponen Linux saja yang membutuhkan OpenSSL 1.0.2, 1.1.1, atau 3.x.

Penyedia JCE

Penyedia JCE adalah Java SDK yang kompatibel dengan OpenJDK 8, OpenJDK 11, OpenJDK 17, dan OpenJDK 21 pada semua platform yang didukung.

Manfaat SDK Klien 5

Dibandingkan dengan 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.

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.

Migrasi dari SDK Klien 3 ke SDK Klien 5

Untuk petunjuk mendetail tentang migrasi dari SDK Klien 3 ke SDK Klien 5, lihat petunjuk migrasi untuk setiap SDK Klien individual:

- Migrasi pustaka PKCS #11 Anda dari Client SDK 3 ke Client SDK 5
- Migrasikan OpenSSL Dynamic Engine Anda dari Client SDK 3 ke Client SDK 5
- Migrasikan penyedia JCE Anda dari Client SDK 3 ke Client SDK 5

- Migrasi dari Client SDK 3 CMU dan KMU ke Client SDK 5 CloudHSM CLI

Untuk fungsionalitas atau kasus penggunaan yang tidak didukung oleh CloudHSM CLI, silakan hubungi dukungan.

> **Note**
>
> Pustaka Klien SDK 5 PKCS #11 sekarang didukung pada platform Windows. Ini dapat menangani sebagian besar kasus penggunaan yang penyedia CNG dan KSP dapat dan harus dianggap sebagai pengganti. KSP hanya tersedia di Client SDK 3 saat ini.

Pustaka PKCS #11

PKCS #11 adalah standar untuk melakukan operasi kriptografi pada modul keamanan perangkat keras (HSM). AWS CloudHSMmenawarkan 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 informasi tentang penggunaan SDK Klien 3, lihatSDK Klien Sebelumnya (SDK Klien 3).

Topik

- Instal Client SDK 5 untuk pustaka PKCS #11
- Pustaka PKCS #11
- Jenis kunci yang didukung
- Mekanisme yang didukung
- Operasi API yang didukung
- Atribut kunci yang didukung
- Contoh kode untuk pustaka PKCS #11
- Migrasi pustaka PKCS #11 Anda dari Client SDK 3 ke Client SDK 5
- Konfigurasi lanjutan lanjutan lanjutan #11

Instal Client SDK 5 untuk pustaka PKCS #11

Topik ini memberikan petunjuk untuk menginstal pustaka PKCS #11 versi terbaru dari seri versi Client SDK 5. Untuk informasi selengkapnya tentang pustaka SDK Klien atau PKCS #11, lihat Menggunakan pustaka SDK Klien dan PKCS #11.

Penginstalan

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 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 arsitektur ARM64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-pkcs11-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el7.aarch64.rpm
```

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
```

CentOS 7 (7.8+)

Instal pustaka PKCS #11 untuk CentOS 7.8+ 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
```

RHEL 7 (7.8+)

Instal pustaka PKCS #11 untuk RHEL 7 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
```

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
```

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
```

Ubuntu 20.04 LTS

Instal pustaka PKCS #11 untuk Ubuntu 20.04 LTS pada arsitektur X86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Focal/cloudhsm-pkcs11_latest_u20.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u20.04_amd64.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 2016

Instal pustaka PKCS #11 untuk Windows Server 2016 pada arsitektur X86_64:

1. Unduh pustaka PKCS #11 untuk SDK Klien 5.
2. Jalankan penginstal pustaka PKCS #11 (AWSCloudHSMPKCS11-latest.msi) dengan hak administratif Windows.

Windows Server 2019

Instal pustaka PKCS #11 untuk Windows Server 2019 pada arsitektur X86_64:

1. Unduh pustaka PKCS #11 untuk SDK Klien 5.
2. Jalankan penginstal pustaka PKCS #11 (AWSCloudHSMPKCS11-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, lihatBootstrap 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:\ProgramFiles\Amazon\CloudHSM
```

Skrip konfigurasi Windows dan berkas log:

```
C:\ProgramData\Amazon\CloudHSM
```

Pustaka PKCS #11

Bila Anda menggunakan pustaka PKCS #11, aplikasi Anda berjalan sebagai pengguna kripto (CU) khusus di HSM Anda. Aplikasi Anda dapat melihat dan mengelola hanya kunci yang CU miliki dan bagikan. Anda dapat menggunakan CU yang ada di HSM atau buat CU baru untuk aplikasi Anda. Untuk informasi tentang mengelola CU, 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!
```

Jenis kunci yang didukung

Pustaka PKCS #11 mendukung jenis kunci berikut.

Tipe Kunci	Deskripsi
AES	Hasilkan kunci AES 128, 192, dan 256-bit.
Triple 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.
RSA	Hasilkan 2048-bit sampai 4096-bit kunci RSA, dengan penambahan 256 bit.

[1] Dilarang setelah 2023 untuk kepatuhan FIPS sesuai panduan NIST. Lihat Kepatuhan FIPS 140: Penutupan Mekanisme 2024 untuk detail.

Mekanisme yang didukung

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 olehAWS CloudHSM.

Pustaka PKCS #11 mendukung algoritme berikut ini:

- Enkripsi dan dekripsi— AES-CBC, AES-CTR, AES-ECB, AES-GCM, DES3-CBC, DES3-ECB, RSA-OAEP, dan RSA-PKCS
- Tanda tangani dan verifikasi- RSA, HMAC, dan ECDSA; dengan dan tanpa hashing
- Hash/digest— SHA1, SHA224, SHA256, SHA384, dan SHA512
- Bungkus kunci— Bungkus Kunci AES,[1] AES-GCM, RSA-AES, dan RSA-OAEP

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_GEN`Mekanisme 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_HMAC[2]
- CKM_SHA224_HMAC[2]
- CKM_SHA256_HMAC[2]
- CKM_SHA384_HMAC[2]
- CKM_SHA512_HMAC[2]

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_GCM[3]

Turunkan fungsi kunci

Pustaka AWS CloudHSM perangkat lunak untuk pustaka PKCS #11 memungkinkan Anda menggunakan mekanisme berikut untuk fungsi Derive.

- CKM_SP800_108_COUNTER_KDF

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

Mekanisme	Ukuran data maksimum dalam byte
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, lihatUkuran 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/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] Dilarang setelah 2023 untuk kepatuhan FIPS sesuai panduan NIST. Lihat Kepatuhan FIPS 140: Penutupan Mekanisme 2024 untuk detail.

Operasi API yang didukung

Pustaka PKCS #11 mendukung operasi API PKCS #11 berikut.

- `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 yang didukung

Objek kunci bisa berupa kunci publik, pribadi, atau rahasia. Tindakan diizinkan pada objek kunci ditentukan melalui atribut. Atribut didefinisikan ketika objek kunci dibuat. Ketika Anda menggunakan pustaka PKCS #11, kami menetapkan nilai default sebagaimana 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.

Menafsirkan tabel atribut pustaka PKCS #11

Tabel pustaka PKCS #11 berisi daftar atribut yang berbeda menurut jenis kunci. Hal ini menunjukkan apakah atribut tertentu didukung untuk jenis kunci tertentu ketika 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

Atribut	Tipe Kunci				Nilai Default
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	Benar
CKA_DESTROYABLE	✓	✓	✓	✓	Benar
CKA_SIGN	✓	✖	✓	✖	Salah
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	✓	✖	✓	✖	

Atribut	Tipe Kunci				Nilai Default
CKA_SENSITIVE	✓[1]	✖	✓[1]	✖	Benar
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	✖	✖	✖	✖	

Atribut	Tipe Kunci				Nilai Default
CKA_PRIVATE_EXPONENT	✖	✖	✖	✖	
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	✓	✓	✓	

Atribut	Tipe Kunci			Nilai Default
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
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	✖	✖	✖	

Atribut	Tipe Kunci			Nilai Default
CKA_WRAP	✓	✓	✖	Salah
CKA_WRAP_TEMPLATE	✓	✓	✖	
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	✖	✖	✖	

Atribut	Tipe Kunci			Nilai Default
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
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]	

Atribut	Tipe Kunci			Nilai Default
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

Atribut	Tipe Kunci							Nilai Default
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓	Salah
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	Benar
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓	Benar
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓	Salah
CKA_SIGN_RECOVER	✖	✖	✖	✖	✖	✖	✖	Salah
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓	Salah
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	✖	✖	
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖	Salah
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖	
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✖	Salah
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓	Salah
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖	Salah

Atribut	Tipe Kunci							Nilai Default
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖	
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓	Benar
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	✖	✖	✓	✖	✖	✖	✖	

Atribut	Tipe Kunci							Nilai Default
CKA_EXPONENT_2	✖	✖	✓	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✓[2]	✖	✖	✖	✖	
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	

Atribut	Tipe Kunci					Nilai Default
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	✓	✓	
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

Atribut	Tipe Kunci					Nilai Default
CKA_SIGN_RECOVER	✖	✖	✖	✖	✖	Salah
CKA_VERIFY	✖	✖	✓	✓	✓	Salah
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	
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	✖	✖	✖	✖	✖	

Atribut	Tipe Kunci					Nilai Default
CKA_PRIME_1	✖	✖	✖	✖	✖	
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	✖	✖	✖	✖	✖	

Atribut	Tipe Kunci					Nilai Default
CKA_CHECK_VALUE		R	R	R	R	R	

DeriveKey

Atribut	Tipe Kunci				Nilai Default
		AES	DES3	Rahasia Generik	
CKA_CLASS		✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE		✓[2]	✓[2]	✓[2]	
CKA_LABEL		✓	✓	✓	
CKA_ID		✓	✓	✓	
CKA_LOCAL		R	R	R	Benar
CKA_TOKEN		✓	✓	✓	Salah
CKA_PRIVATE		✓[1]	✓[1]	✓[1]	Benar
CKA_ENCRYPT		✓	✓	✖	Salah
CKA_DECRYPT		✓	✓	✖	Salah
CKA_DERIVE		✓	✓	✓	Salah

Atribut	Tipe Kunci			Nilai Default
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	Benar
CKA_DESTROYABLE	✓[1]	✓[1]	✓[1]	Benar
CKA_SIGN	✓	✓	✓	Salah
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	Salah
CKA_VERIFY_RECOVER	✖	✖	✖	
CKA_WRAP	✓	✓	✖	Salah
CKA_UNWRAP	✓	✓	✖	Salah
CKA_SENSITIVE	R	R	R	Benar
CKA_EXTRACTABLE	✓	✓	✓	Benar
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	

Atribut	Tipe Kunci			Nilai Default
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
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	✖	✖	✖	

Atribut	Tipe Kunci			Nilai Default
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
CKA_CLASS	✓	✓	✓	✓	✓	✓	✓
CKA_KEY_TYPE	✓	✓	✓	✓	✓	✓	✓
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓
CKA_ID	✓	✓	✓	✓	✓	✓	✓
CKA_LOCAL	✓	✓	✓	✓	✓	✓	✓
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]
CKA_ENCRYPT	✖	✖	✖	✓	✓	✓	✖

Atribut	Tipe Kunci						
CKA_DECRYPT	✖	✖	✓	✖	✓	✓	✖
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	✓
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓
CKA_SIGN_RECOVER	✖	✖	✓	✖	✖	✖	✖
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓
CKA_VERIFY_RECOVER	✖	✖	✖	✓	✖	✖	✖
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✓
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖

Atribut	Tipe Kunci						
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓
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	✖	✖	✖	✖

Atribut	Tipe Kunci						
CKA_EXPONENT_2	✖	✖	S	✖	✖	✖	✖
CKA_PRIVATE_EXPONENT	✖	✖	S	✖	✖	✖	✖
CKA_PUBLIC_EXPONENT	✖	✖	✓	✓	✖	✖	✖
CKA_EC_PARAMS	✓	✓	✖	✖	✖	✖	✖
CKA_EC_POINT	✖	✓	✖	✖	✖	✖	✖
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

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

Menentukan dalam templat suatu atribut yang tidak didukung oleh hasil khusus kunci dalam 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

Contoh kode GitHub menunjukkan cara menyelesaikan tugas dasar menggunakan pustaka PKCS #11.

Prasyarat

Sebelum menjalankan sampel, lakukan langkah-langkah berikut untuk mengatur lingkungan Anda:

- Instal dan konfigurasikan 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

Migrasi pustaka PKCS #11 Anda dari Client SDK 3 ke Client SDK 5

Gunakan topik ini untuk memigrasikan pustaka PKCS #11 Anda dari SDK Klien 3 ke SDK Klien 5. Untuk manfaat migrasi, lihatManfaat SDK Klien 5.

Pada tahunAWS 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, lihatMigrasi dari SDK 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_PKCS5_PAD`
`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 CloudHSMPKCS #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-pkcs11
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-pkcs11
   ```

   RHEL 7

   ```
   $ sudo yum remove cloudhsm-pkcs11
   ```

   RHEL 8

   ```
   $ sudo yum remove cloudhsm-pkcs11
   ```

2. 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
```

> **Note**
>
> Konfigurasi khusus perlu diaktifkan lagi.

3. Instal pustaka PKCS #11 Client SDK dengan mengikuti langkah-langkahnya. Instal Client SDK 5 untuk pustaka PKCS #11
4. 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
5. 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

Konfigurasi lanjutan lanjutan lanjutan #11

PenyediaAWS 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
- Konfigurasi coba lagi PKCS #11

Menghubungkan ke beberapa slot dengan PKCS #11

Satu slot di pustaka Client SDK 5 PKCS #11 mewakili satu koneksi ke cluster di. AWS CloudHSM Dengan Client SDK 5, Anda dapat mengonfigurasi pustaka PKCS11 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
- Konfigurasikan Perpustakaan PKCS #11 untuk fungsionalitas multi-slot
- konfigurasi-pkcs11 add-cluster
- konfigurasi-pkcs11 hapus-cluster

Prasyarat multi-slot

- 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???.

Konfigurasikan Perpustakaan PKCS #11 untuk fungsionalitas multi-slot

Untuk mengonfigurasi pustaka PKCS #11 Anda untuk fungsionalitas multi-slot, 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.

konfigurasi-pkcs11 add-cluster

Saat menghubungkan ke beberapa slot dengan PKCS #11, gunakan configure-pkcs11 add-cluster perintah untuk menambahkan cluster ke konfigurasi Anda.

Sintaks

```
configure-pkcs11 add-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [--region <REGION>]
        [--endpoint <ENDPOINT>]
        [--hsm-ca-cert <HSM CA CERTIFICATE FILE>]
        [--server-client-cert-file <CLIENT CERTIFICATE FILE>]
        [--server-client-key-file <CLIENT KEY FILE>]
        [-h, --help]
```

Contoh-contoh

Tambahkan cluster menggunakan **`cluster-id`** parameter

Example

Gunakan configure-pkcs11 add-cluster bersama dengan `cluster-id` parameter untuk menambahkan cluster (dengan ID dari`cluster-1234567`) ke konfigurasi Anda.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 add-cluster --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\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

```
C:\Program Files\Amazon\CloudHSM\> .\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”.

Parameter-parameter

--cluster-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 instans EC2 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 ???.

Diperlukan: Ya

--endpoint ***<Endpoint>***

Tentukan titik akhir AWS CloudHSM API yang digunakan untuk melakukan `DescribeClusters` panggilan. Anda harus menetapkan pilihan ini dalam kombinasi dengan `--cluster-id`.

Diperlukan: Tidak

-- hsm-ca-cert <HsmCA Certificate Filepath>

Menentukan filepath ke sertifikat HSM CA.

Diperlukan: Tidak

--region ***<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`.

Diperlukan: Tidak

-- server-client-cert-file <Client Certificate Filepath>

Jalur ke sertifikat klien yang digunakan untuk autentikasi mutual klien-server TLS.

Hanya gunakan opsi ini jika Anda tidak ingin menggunakan kunci default dan sertifikat SSL/TLS yang kami sertakan dengan Klien SDK 5. Anda harus menetapkan pilihan ini dalam kombinasi dengan `--server-client-key-file`.

Diperlukan: Tidak

-- server-client-key-file <Client Key Filepath>

Jalur ke kunci klien yang digunakan untuk otentikasi timbal balik client-server TLS.

Hanya gunakan opsi ini jika Anda tidak ingin menggunakan kunci default dan sertifikat SSL/TLS yang kami sertakan dengan Klien SDK 5. Anda harus menetapkan pilihan ini dalam kombinasi dengan `--server-client-cert-file`.

Diperlukan: Tidak

konfigurasi-pkcs11 hapus-cluster

Saat menghubungkan ke beberapa slot dengan PKCS #11, gunakan configure-pkcs11 remove-cluster perintah untuk menghapus cluster dari slot PKCS #11 yang tersedia.

Sintaks

```
configure-pkcs11 remove-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [-h, --help]
```

Contoh-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

```
C:\Program Files\Amazon\CloudHSM\> .\configure-pkcs11.exe remove-cluster --cluster-
id cluster-1234567
```

Untuk informasi tentang parameter `--cluster-id`, lihat the section called “Parameter-parameter”.

Parameter

--cluster-id *<Cluster ID>*

ID cluster untuk menghapus dari konfigurasi

Diperlukan: Ya

Coba lagi perintah untuk PKCS #11

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, SDK klien 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, SDK klien 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:

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-pkcs11.exe --default-retry-
mode off
```

OpenSSL Dynamic Engine

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. Pemuatan SSL/TLS di Linux Untuk contoh penggunaan AWS CloudHSM dengan OpenSSL, lihat blog keamanan AWS ini. Untuk informasi tentang dukungan platform untuk SDK, lihatthe section called “Platform yang didukung”. Untuk pemecahan masalah, lihat. Masalah yang diketahui untuk OpenSSL Dynamic Engine

Untuk informasi tentang penggunaan SDK Klien 3, lihatSDK Klien Sebelumnya (SDK Klien 3).

Untuk informasi lebih lanjut, lihat topik di bawah ini.

Topik

- Memasang OpenSSL Dynamic Engine
- Jenis kunci OpenSSL Dynamic Engine
- Mekanisme Mesin Dinamis OpenSSL

- Migrasikan OpenSSL Dynamic Engine Anda dari Client SDK 3 ke Client SDK 5
- Konfigurasi lanjutan untuk OpenSSL

Memasang OpenSSL Dynamic Engine

> **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 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
   ```

   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
```

CentOS 7 (7.8+)

Instal OpenSSL Dynamic Engine untuk CentOS 7 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
```

RHEL 7 (7.8+)

Instal OpenSSL Dynamic Engine untuk RHEL 7 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
```

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
```

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
```

Ubuntu 20.04 LTS

Instal OpenSSL Dynamic Engine untuk Ubuntu 20.04 LTS pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Focal/cloudhsm-dyn_latest_u20.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u20.04_amd64.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 kredensil pengguna kripto (CU). Untuk informasi tentang membuat CU, lihatMenggunakan CMU untuk mengelola pengguna.

```
$ 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`.

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 ]
```

Jenis kunci OpenSSL Dynamic Engine

AWS CloudHSM OpenSSL Dynamic Engine mendukung jenis kunci berikut.

Tipe Kunci	Deskripsi
EC	Tanda tangan/verifikasi ECDSA untuk tipe kunci P-256, P-384, dan secp256k1. Untuk menghasilkan kunci EC yang dapat dioperasi kan dengan mesin OpenSSL, lihat. file hasilkan kunci
RSA	Pembuatan kunci RSA untuk kunci 2048, 3072, dan 4096-bit. Tanda/verifikasi RSA. Verifikasi diturunkan ke perangkat lunak OpenSSL.

Mekanisme Mesin Dinamis OpenSSL

Pelajari cara menggunakan mekanisme AWS CloudHSM OpenSSL Dynamic Engine.

Tanda tangani dan verifikasi fungsi

AWS CloudHSM OpenSSL Dynamic Engine 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.

Jenis Tanda Tangan RSA

- SHA1denganRSA
- SHA224denganRSA
- SHA256denganRSA
- SHA384denganRSA

- SHA512denganRSA

Jenis Tanda Tangan ECDSA

- SHA1denganECDSA
- SHA224denganECDSA
- SHA256denganECDSA
- SHA384denganECDSA
- SHA512denganECDSA

Migrasikan OpenSSL Dynamic Engine Anda dari Client SDK 3 ke Client SDK 5

Gunakan topik ini untuk memigrasikan OpenSSL Dynamic Engine Anda dari Client SDK 3 ke Client SDK 5. Untuk manfaat migrasi, lihatManfaat SDK Klien 5.

Dalam 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, lihatMigrasi dari SDK 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-dyn
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-dyn
   ```

   RHEL 7

   ```
   $ sudo yum remove cloudhsm-dyn
   ```

   RHEL 8

   ```
   $ sudo yum remove cloudhsm-dyn
   ```

2. 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
```

> **Note**
>
> Konfigurasi khusus perlu diaktifkan lagi.

3. Instal Client SDK OpenSSL Dynamic Engine dengan mengikuti langkah-langkahnya. Memasang OpenSSL Dynamic Engine
4. Client SDK 5 memperkenalkan format file konfigurasi baru dan alat bootstrap baris perintah. Untuk mem-bootstrap Client SDK 5 OpenSSL Dynamic Engine Anda, ikuti petunjuk yang tercantum dalam panduan pengguna di bawah. Bootstrap Klien SDK
5. 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

Konfigurasi lanjutan untuk OpenSSL

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

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, SDK klien 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, SDK klien 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 perintah coba lagi ke off mode:

```
$ sudo /opt/cloudhsm/bin/configure-dyn --default-retry-mode off
```

Windows

Untuk mengatur perintah coba lagi off untuk Client SDK 5 pada Windows

- Anda dapat menggunakan perintah berikut untuk mengatur perintah coba lagi ke off mode:

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-dyn.exe --default-retry-mode
 off
```

Penyedia JCE

Penyedia JCE AWS CloudHSM adalah implementasi penyedia dibangun dari kerangka kerja 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 informasi tentang penggunaan SDK Klien 3, lihatSDK Klien Sebelumnya (SDK Klien 3).

Topik

Instal dan gunakan penyedia AWS CloudHSM JCE untuk Client SDK 5

Penyedia JCE kompatibel dengan OpenJDK 8, OpenJDK 11, OpenJDK 17, dan OpenJDK 21. Anda dapat mengunduh keduanya dari situs web OpenJDK.

> **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

Instal penyedia JCE

1. Gunakan perintah berikut untuk mengunduh dan menginstal penyedia JCE.

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 arsitektur ARM64:

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

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

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 arsitektur ARM64:

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

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

CentOS 7 (7.8+)

Instal penyedia JCE untuk CentOS 7 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
```

RHEL 7 (7.8+)

Instal penyedia JCE untuk RHEL 7 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
```

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
```

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
```

Ubuntu 20.04 LTS

Instal penyedia JCE untuk Ubuntu 20.04 LTS pada arsitektur x86_64:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Focal/
cloudhsm-jce_latest_u20.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u20.04_amd64.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 2016

Instal penyedia JCE 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/AWSCloudHSMJCE-latest.msi -Outfile C:\AWSCloudHSMJCE-latest.msi
```

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

Windows Server 2019

Instal penyedia JCE 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/AWSCloudHSMJCE-latest.msi -Outfile C:\AWSCloudHSMJCE-latest.msi
```

```
PS C:\> Start-Process msiexec.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`

Memberikan kredensi kepada penyedia JCE

Sebelum aplikasi Java Anda dapat menggunakan HSM, HSM harus terlebih dahulu mengotentikasi aplikasi. HSM mengautentikasi menggunakan login eksplisit atau metode login implisit.

Login eksplisit - Metode ini memungkinkan Anda memberikan AWS CloudHSM kredensil 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 kredensil 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.

  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 lingkungan

Dasar-dasar manajemen utama di penyedia JCE

Dasar-dasar manajemen kunci dalam penyedia JCE melibatkan mengimpor kunci, mengekspor 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.

Jenis kunci yang didukung

Pustaka perangkat lunak AWS CloudHSM untuk Java memungkinkan Anda untuk menghasilkan jenis kunci berikut.

Tipe Kunci	Deskripsi
AES	Hasilkan kunci AES 128, 192, dan 256-bit.
Triple DES (3DES, DeSede)	Hasilkan Kunci Triple DES 192-bit [1] untuk perubahan yang akan datang. Lihat catatan kaki
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.

[1] Dilarang setelah 2023 untuk kepatuhan FIPS sesuai panduan NIST. Lihat Kepatuhan FIPS 140: Penutupan Mekanisme 2024 untuk rincian selengkapnya.

Mekanisme yang didukung

Untuk informasi tentang antarmuka dan kelas mesin Java Cryptography Architecture (JCA) yang didukung oleh AWS CloudHSM, lihat topik berikut.

Topik

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 perangkat lunak AWS CloudHSM untuk Java mendukung algoritme, mode, dan kombinasi bantalan berikut.

Algoritme	Mode	Bantalan	Catatan
AES	CBC	`AES/CBC/NoPadding`	Menerapkan `Cipher.EN`

Algoritme	Mode	Bantalan	Catatan
		AES/CBC/PKCS5Padding	CRYPT_MODE dan Cipher.DECRYPT_MODE. Menerapkan Cipher.UNWRAP_MODE for AES/CBC NoPadding
AES	ECB	AES/ECB/PKCS5Padding AES/ECB/NoPadding	Menerapkan Cipher.ENCRYPT_MODE dan Cipher.DECRYPT_MODE.
AES	CTR	AES/CTR/NoPadding	Menerapkan Cipher.ENCRYPT_MODE dan Cipher.DECRYPT_MODE.

Algoritme	Mode	Bantalan	Catatan
AES	GCM	`AES/GCM/NoPadding`	Menerapkan `Cipher.WRAP_MODE`, `Cipher.UNWRAP_MODE`, `Cipher.ENCRYPT_MODE`, dan `Cipher.DECRYPT_MODE`. Ketika melakukan enkripsi AES-GCM, HSM mengabaikan vektor inisialisasi (IV) dalam permintaan dan menggunakan IV yang dihasilkannya. Saat operasi selesai, Anda harus memanggil `Cipher.getIV()` untuk mendapatkan IV.

AESWrap	ECB	`AESWrap/ECB/NoPadding`
		
		`AESWrap/ECB/PKCS5Padding`
		
		`AESWrap/ECB/ZeroPadding`
			Menerapkan `Cipher.WRAP_MODE` dan `Cipher.UNWRAP_MODE`.

Algoritme	Mode	Bantalan	Catatan

DESede (Tiga DES)	CBC	`DESede/CBC/PKCS5Padding`
		
		`DESede/CBC/NoPadding`
			Menerapkan `Cipher.ENCRYPT_MODE` dan `Cipher.DECRYPT_MODE`. Lihat catatan 1 di bawah untuk perubahan yang akan datang.

DESede (Tiga DES)	ECB	`DESede/ECB/NoPadding`
		
		`DESede/ECB/PKCS5Padding`
			Menerapkan `Cipher.ENCRYPT_MODE` dan `Cipher.DECRYPT_MODE`. Lihat catatan 1 di bawah untuk perubahan yang akan datang.

Algoritme	Mode	Bantalan	Catatan

RSA	ECB	`RSA/ECB/PKCS1Padding` lihat catatan 1
		
		`RSA/ECB/OAEPPadding`
		
		`RSA/ECB/OAEPWithSHA-1ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-224ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-256ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-384ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-512ANDMGF1Padding`
			Menerapkan `Cipher.WRAP_MODE`, `Cipher.UNWRAP_MODE`, `Cipher.ENCRYPT_MODE`, dan `Cipher.DECRYPT_MODE`.

Algoritme	Mode	Bantalan	Catatan
RSA	ECB	`RSA/ECB/NoPadding`	Menerapkan `Cipher.ENCRYPT_MODE` dan `Cipher.DECRYPT_MODE` .

RSAAESWrap	ECB	`RSAAESWrap/ECB/OAEPPadding`
		
		`RSAAESWrap/ECB/OAEPWithSHA-1ANDMGF1Padding`
		
		`RSAAESWrap/ECB/OAEPWithSHA-224ANDMGF1Padding`
		
		`RSAAESWrap/ECB/OAEPWithSHA-256ANDMGF1Padding`
		
		`RSAAESWrap/ECB/OAEPWithSHA-384ANDMGF1Padding`
		
		`RSAAESWrap/ECB/OAEPWithSHA-512ANDMGF1Padding`
			Menerapkan `Cipher.WRAP_MODE` dan `Cipher.UNWRAP_MODE` .

Tanda tangani dan verifikasi fungsi

Pustaka perangkat lunak AWS CloudHSM 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 perangkat lunak AWS CloudHSM untuk Java mendukung digest 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 perangkat lunak AWS CloudHSM untuk Java mendukung algoritme 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)

CMAC (kode otentikasi pesan berbasis Cipher) membuat kode otentikasi pesan (MAC) menggunakan sandi blok dan kunci rahasia. Mereka berbeda dari HMAC karena mereka menggunakan metode kunci simetris blok untuk MAC daripada metode hashing.

Pustaka AWS CloudHSM perangkat lunak untuk Java mendukung algoritma CMAC berikut.

- `AESCMAC`

Konversi kunci ke spesifikasi utama menggunakan pabrik-pabrik utama

Anda dapat menggunakan pabrik-pabrik utama untuk mengonversi kunci ke spesifikasi utama. AWS CloudHSMmemiliki 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 menurunkan kunci simetris ke dalam. AWS CloudHSM Berikut ini adalah spesifikasi yang didukung untuk KeyFactory:

- `generateSecret`Metode untuk SecretKeyFactory KeySpeckelas berikut didukung:
  - KeyAttributesMapdapat digunakan untuk mengimpor byte kunci dengan atribut tambahan sebagai CloudHSM Key. Contohnya dapat ditemukan di sini di sini.
  - SecretKeySpecdapat digunakan untuk mengimpor spesifikasi kunci simetris sebagai Kunci CloudHSM.
  - AesCmacKdfParameterSpecdapat 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 asimetris ke dalamAWS CloudHSM. Untuk informasi lebih lanjut, lihat sumber daya berikut:

- `generatePublic`Metode KeyFactory For, KeySpeckelas 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
  - RSA PublicKeySpec untuk Kunci Publik RSA
  - EC PublicKeySpec untuk Kunci Publik EC
- `generatePrivate`Metode KeyFactory For, KeySpeckelas berikut didukung:

- KeyAttributesMap CloudHSM untuk RSA dan EC, termasuk: KeyTypes
  - KeyAttributesMap CloudHSM untuk RSA dan EC publik. KeyTypes Contoh dapat ditemukan di sini
  - PKCS8 EncodedKeySpec untuk EC dan RSA Private Key
  - RSA PrivateCrtKeySpec untuk Kunci Pribadi RSA
  - EC PrivateKeySpec untuk Kunci Pribadi EC

`translateKey`Metode KeyFactory For, dibutuhkan setiap Key yang mengimplementasikan Key Interface.

Anotasi mekanisme

[1] Dilarang setelah 2023 untuk kepatuhan FIPS sesuai panduan NIST. Lihat Kepatuhan FIPS 140: Penutupan Mekanisme 2024 untuk rincian selengkapnya.

Atribut kunci Java yang didukung

Topik ini menjelaskan cara menggunakan ekstensi berpemilik untuk penyedia JCE untuk menyetel 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, lihatthe 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 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

Atribut	Nilai Default			Catatan
	Kunci simetris	Kunci Publik dalam Pasangan Kunci	Kunci Privat dalam Pasangan Kunci	
				mendapatkan kunci lainnya.
ENCRYPT	TRUE	TRUE		Benar menunjukkan Anda dapat menggunakan kunci untuk mengenkripsi penyangga apa pun.
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, DeSede, rahasia generik, 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 direplikasi di semua HSM di klaster dan termasuk dalam cadangan. TOKEN = FALSE menyiratkan kunci sementara yang secara otomatis terhapus 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 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	get(keyAttribute)
Isi nilai satu atribut kunci	Nilai sebelumnya terkait dengan atribut kunci, atau nol jika tidak ada pemetaan untuk atribut kunci	put(keyAttribute, value)
Isi nilai untuk beberapa atribut kunci	N/A	putAll () keyAttributesMap
Hapus pasangan nilai kunci dari peta atribut	Nilai sebelumnya terkait dengan atribut kunci, atau nol jika tidak ada pemetaan untuk atribut kunci	remove(keyAttribute)

> **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

Prasyarat

Sebelum menjalankan sampel, Anda harus mengatur lingkungan Anda:

- Instal dan konfigurasikan 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] Dilarang setelah 2023 untuk kepatuhan FIPS sesuai panduan NIST. Lihat Kepatuhan FIPS 140: Penutupan Mekanisme 2024 untuk rincian selengkapnya.

AWS CloudHSMJCE 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 terbaruAWS CloudHSM, lihat Rilis terbaru bagian di halaman Unduhan.

Anda dapat mengimpor Javadocs ke lingkungan pengembangan terintegrasi (IDE) atau melihatnya di browser web.

Menggunakan kelas AWS CloudHSM KeyStore Java

AWS CloudHSM `KeyStore`Kelas ini menyediakan toko kunci PKCS12 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 `KeyStore`Kelas mengimplementasikan `KeyStore` Service Provider Interface (SPI) dari Java Cryptography Extension (JCE). Untuk informasi selengkapnya tentang penggunaan`KeyStore`, lihat Kelas KeyStore.

> **Note**
>
> Karena sertifikat adalah informasi publik, dan untuk memaksimalkan kapasitas penyimpanan untuk kunci kriptografi, AWS CloudHSM tidak mendukung penyimpanan sertifikat pada HSM.

Memilih toko kunci yang sesuai

Penyedia AWS CloudHSM Java Cryptographic Extension (JCE) menawarkan AWS CloudHSM tujuan khusus. KeyStore AWS CloudHSM `KeyStore`Kelas 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 AWS CloudHSM KeyStore

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 file PKCS12 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);
```

Menggunakan AWS CloudHSM KeyStore

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 file PKCS12 dan objek yang ada 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. Penyimpanan kunci memeriksa objek yang disimpan secara lokal dalam file PKCS12 dan bendaobjek yang ada di HSM.

- `deleteEntry`

  Menghapus entri sertifikat dari file PKCS12 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 file PKCS12 dan HSM untuk alias.

- `setCertificateEntry`

  Menetapkan sertifikat yang diberikan untuk alias yang diberikan. Jika alias yang diberikan sudah digunakan untuk me ngidentifikasi 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 untuk pengaliran output yang diberikan sebagai file PKCS12 dan mengamankannya dengan kata sandi yang diberikan. Selain itu, semua kunci dimuat tetap ada (yang ditetapkan menggunakan panggilan`setKey`).

Migrasikan penyedia JCE Anda dari Client SDK 3 ke Client SDK 5

Gunakan topik ini untuk memigrasikan penyedia JCE Anda dari Client SDK 3 ke Client SDK 5. Untuk manfaat migrasi, lihatManfaat SDK Klien 5.

Dalam 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 API yang 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, lihatMigrasi dari SDK Klien 3 ke SDK Klien 5.

Topik

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 `CaviumProvider` dan memiliki nama Provider. `Cavium`	Di Client SDK 5, kelas Provider dipanggil `CloudHsmProvider` dan memiliki nama `CloudHSM` Provider.	Contoh cara menginisialisasi `CloudHsmProvider` 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 `LoginManager` kelas untuk login eksplisit. [1]	Di Client SDK 5, `CloudHSM` penyedia mengimplementasikan `AuthProvider` untuk login eksplisit. `AuthProvider` adalah kelas	Untuk contoh tentang cara menggunakan login eksplisit dengan Client SDK 5, lihat LoginRunner 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, `Cavium[Key-type]AlgorithmParameterSpec` digunakan untuk menentukan parameter pembuatan kunci. Untuk cuplikan kode, lihat catatan kaki. 1	Di Client SDK 5, `KeyAttributesMap` digunakan untuk menentukan atribut pembuatan kunci. Untuk cuplikan kode, lihat catatan kaki. 2	Untuk contoh tentang cara menggunakan `KeyAttributesMap` untuk menghasilkan kunci simetris, lihat SymmetricKeys sampel di repositori sampel AWS CloudHSM Github.
Generasi pasangan kunci	Di Client SDK 3, `Cavium[Key-type]Algorithmparameterspec` digunakan untuk menentukan parameter pembuatan key pair. Untuk cuplikan kode, lihat catatan kaki. 3	Di Client SDK 5, `KeyPairAttributesMap` digunakan untuk menentukan parameter ini. Untuk cuplikan kode, lihat catatan kaki. 4	Untuk contoh tentang cara menggunakan `KeyAttributesMap` untuk menghasilkan kunci asimetris, lihat AsymmetricKeys sampel di repositori 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 menggunak an label kunci atau pegangan kunci	Di Client SDK 5, aplikasi dapat menggunakan tombol Menggunakan kelas	

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.	AWS CloudHSM KeyStore Java untuk menemukan kunci berdasarkan label. Untuk menemukan kunci dengan pegangan, gunakan AWS CloudHSM `KeyStoreWithAttributes` dengan AWS CloudHSM `KeyRefereneSpec` .	

Apa yang telah berubah	Apa itu di Client SDK 3	Apa itu di Client SDK 5	Contoh
Menemukan beberapa entri	Saat mencari kunci menggunak angetEntry,getKey, atau getCertificate 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 danKeyStoreWithAttributes , skenario yang sama ini akan menghasilkan pengecualian yang dilemparkan. Untuk memperbaiki masalah ini, disarankan untuk mengatur label unik untuk kunci menggunakan kunci set-atribut perintah di CloudHSM CLI. Atau gunakan KeyStoreWithAttributes#getKeys untuk mengembalikan semua kunci yang cocok 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. `Util.findAllKeys()`	Client SDK 5 membuat pencarian kunci lebih sederhana dan lebih efisien dengan menggunakan `KeyStoreWithAttributes` 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 `KeyStoreWithAttributes#getKeys` dengan kosong. `KeyAttributesMap`	Contoh yang menggunakan `KeyStoreWithAttributes` kelas untuk menemukan kunci tersedia di repositori sampel AWS CloudHSM Github 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 `Util.deleteKey()` untuk menghapus kunci.	`KeyObjek` di Client SDK 5 mengimplementasikan `Destroyable` antarmuka yang memungkinkan kunci dihapus menggunakan `destroy()` metode antarmuka ini.	Kode contoh yang menunjukkan fungsionalitas tombol hapus dapat ditemukan di repositori sampel CloudHSM Github. 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 enkripsi/dekripsi/bungkus Cipher.

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-jce
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-jce
   ```

   RHEL 7

   ```
   $ sudo yum remove cloudhsm-jce
   ```

   RHEL 8

   ```
   $ sudo yum remove cloudhsm-jce
   ```

2. 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
```

> **Note**
>
> Konfigurasi khusus perlu diaktifkan lagi.

3. Instal penyedia Client SDK JCE dengan mengikuti langkah-langkahnya. Instal dan gunakan penyedia AWS CloudHSM JCE untuk Client SDK 5
4. 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
5. 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

Konfigurasi lanjutan untuk JCE

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 klaster dengan penyedia JCE

Konfigurasi ini memungkinkan satu instance klien untuk berkomunikasi ke beberapa cluster. Dibandingkan dengan memiliki satu instans yang hanya berkomunikasi dengan satu klaster, ini bisa menjadi fitur penghematan biaya untuk beberapa kasus penggunaan. `CloudHsmProvider`Kelas adalahAWS CloudHSM implementasi dari kelas Provider Java Security. Setiap instance dari kelas ini mewakili koneksi ke seluruhAWS CloudHSM klaster Anda. Anda instantiate kelas ini dan menambahkannya ke daftar penyedia Java Security sehingga Anda dapat berinteraksi dengannya menggunakan kelas JCE standar.

Contoh berikut instantiates kelas ini dan menambahkannya ke daftar penyedia Java Security:

```
if (Security.getProvider(CloudHsmProvider.PROVIDER_NAME) == null) {
    Security.addProvider(new CloudHsmProvider());
}
```

`CloudHsmProvider`konfigurasi

`CloudHsmProvider`dapat dikonfigurasi dengan dua cara:

1. Konfigurasikan dengan file (konfigurasi default)
2. Konfigurasikan menggunakan kode

Konfigurasikan dengan file (Konfigurasi default)

Ketika Anda instantiate`CloudHsmProvider` menggunakan konstruktor default, secara default akan mencari file konfigurasi di`/opt/cloudhsm/etc/cloudhsm-jce.cfg` jalan di Linux. File konfigurasi ini dapat dikonfigurasi menggunakan file`configure-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 menggunakan kode

Pada Client SDK versi 5.8.0, Anda juga dapat mengkonfigurasi`CloudHsmProvider` menggunakan kode Java. Cara untuk melakukannya adalah dengan menggunakan`CloudHsmProviderConfig` objek. Anda dapat membangun objek ini menggunakan`CloudHsmProviderConfigBuilder`.

`CloudHsmProvider`memiliki konstruktor lain yang mengambil`CloudHsmProviderConfig` objek, sebagai 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`CloudHsmCluster1` adak. 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 pengenal unik tidak ditentukan dengan`withClusterUniqueIdentifier` metode ini, nama penyedia yang dibuat secara acak dibuat untuk Anda. Untuk mendapatkan pengenal yang dibuat secara acak ini, aplikasi dapat memanggil`provider.getName()` untuk mendapatkan pengenal.

Menghubungkan ke Klaster

Seperti yang dinyatakan di atas, masing-masing`CloudHsmProvider` mewakili koneksi ke Cluster CloudHSM 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 klaster 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 mengkonfigurasi 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 contoh berikut untukNoPadding operasi AES/GCM/:

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider1);
```

Dan dalam aplikasi yang sama untuk melakukan “AES” Generasi kunci pada cluster kedua menggunakan nama penyedia, Anda juga dapat menggunakan contoh berikut:

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider2.getName());
```

Coba lagi perintah untuk JCE

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, SDK klien 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, SDK klien 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:

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-jce.exe --default-retry-mode
 off
```

Ekstraksi kunci menggunakan JCE

Java Cryptography Extension (JCE) menggunakan arsitektur yang memungkinkan implementasi kriptografi yang berbeda untuk dicolokkan. AWS CloudHSMmengirimkan salah satu penyedia JCE seperti yang membongkar operasi kriptografi ke HSM. Agar sebagian besar penyedia JCE lainnya dapat bekerja dengan kunci yang disimpan di AWS CloudHSM, mereka harus mengekstrak byte kunci dari HSM Anda dalam teks yang jelas ke dalam memori mesin Anda untuk penggunaannya. HSM 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 di clear.

> **Important**
>
> JCE menurunkan operasi keAWS CloudHSM setiap kali penyedia AWS CloudHSM ditentukan atau objekAWS CloudHSM 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.

PenyediaAWS CloudHSM JCE memungkinkan ekstraksi kunci publik untuk bekerja dengan penyedia JCE eksternal secara default. Metode berikut selalu diizinkan:

Kelas	Metode	Format (GetEncoded)
EcPublicKey	getEncoded ()	X.509
	GetW ()	T/A
RSAPublicKey	getEncoded ()	X.509
	getPublicExponent()	T/A

Kelas	Metode	Format (GetEncoded)
CloudHsmRsaPrivateCrtKey	getPublicExponent()	T/A

PenyediaAWS CloudHSM JCE tidak mengizinkan ekstraksi byte kunci yang jelas untuk kunci pribadi atau rahasia secara default. Jika kasus penggunaan Anda memerlukannya, Anda dapat mengaktifkan ekstraksi byte kunci yang jelas untuk kunci pribadi atau rahasia dengan ketentuan 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 tepercaya untuk mengontrol pembukaan kunci.

Memungkinkan penyediaAWS CloudHSM JCE untuk mengekstrak rahasia kunci pribadi dariAWS CloudHSM

> **Important**
>
> Perubahan konfigurasi ini memungkinkan ekstraksi semua byte`EXTRACTABLE` kunci di clear dari klaster 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

```
C:\Program Files\Amazon\CloudHSM\> .\configure-jce.exe --enable-clear-key-extraction-in-software
```

2. Setelah Anda mengaktifkan ekstraksi kunci yang jelas, metode berikut diaktifkan untuk mengekstraksi kunci pribadi ke dalam memori.

Kelas	Metode	Format (GetEncoded)
Key	getEncoded ()	RAW
ECPrivateKey	getEncoded ()	PKCS #8
	getS ()	T/A
RSAPrivateCrtKey	getEncoded ()	X.509
	getPrivateExponent()	T/A
	GetPrimep ()	T/A
	getPrimeq ()	T/A
	getPrimeExponentP ()	T/A
	getPrimeExponentQ ()	T/A
	getCrtCoefficient()	T/A

Jika Anda ingin mengembalikan perilaku default dan tidak mengizinkan JCE untuk mengekspor kunci di clear, jalankan perintah berikut:

Linux

```
$ /opt/cloudhsm/bin/configure-jce --disable-clear-key-extraction-in-software
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\configure-jce.exe --disable-clear-key-
extraction-in-software
```

API Kriptografi: Next Generation (CNG) dan penyedia penyimpanan kunci (KSP) untuk Microsoft Windows

Klien AWS CloudHSM untuk Windows termasuk penyedia CNG dan KSP. Saat ini, hanya Client SDK 3 yang mendukung penyedia CNG dan KSP.

Penyedia penyimpanan kunci (KSP) 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 lebih lanjut, 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, pelaksanaan AWS CloudHSM CNG menyediakan fungsionalitas berikut:

- Primitif Kriptografi - memungkinkan Anda untuk melakukan operasi kriptografi mendasar.
- Impor dan Ekspor Kunci - memungkinkan Anda untuk mengimpor dan mengekspor kunci asimetris.
- API Perlindungan Data (CNG DPAPI) - memungkinkan Anda untuk dengan mudah mengenkripsi dan mendekripsi data.
- Penyimpanan Kunci dan Pengambilan - memungkinkan Anda untuk menyimpan dan mengisolasi kunci privat dari pasangan kunci asimetris dengan aman.

Topik

- Memverifikasi Penyedia KSP dan CNG untuk Windows
- AWS CloudHSMPrasyarat Windows
- Kaitkan AWS CloudHSM kunci dengan sertifikat

- Contoh kode untuk penyedia CNG

Memverifikasi Penyedia KSP dan CNG untuk Windows

Penyedia KSP dan CNG diinstal ketika Anda menginstal klien AWS CloudHSM Windows. Anda dapat menginstal klien dengan mengikuti langkah-langkah di Instal klien (Windows).

Konfigurasikan 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 menggunakan penyedia KSP dan CNG dan setelah Anda menambahkan atau menghapus HSM di klaster Anda. Dengan cara ini, AWS CloudHSM mampu menyinkronkan data dan menjaga konsistensi di semua HSM di klaster.

Langkah 1: Hentikan AWS CloudHSM klien

Sebelum Anda memperbarui file konfigurasi yang digunakan penyedia, hentikan klien AWS CloudHSM. 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 di mana Anda memulai klien AWS CloudHSM.

Langkah 2: Perbarui file AWS CloudHSM konfigurasi

Langkah ini menggunakan parameter -a Konfigurasi alat untuk menambahkan alamat IP antarmuka jaringan antarmuka jaringan elastis (ENI) dari salah satu HSM di klaster ke file konfigurasi.

```
C:\Program Files\Amazon\CloudHSM configure.exe -a <HSM ENI IP>
```

Untuk mendapatkan alamat IP ENI dari HSM di klaster Anda, arahkan ke konsol AWS CloudHSM, pilih klaster, dan pilih klaster yang diinginkan. Anda juga dapat menggunakan DescribeClustersoperasi, 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 klien AWS CloudHSM. Saat klien AWS CloudHSM dimulai, klien menggunakan alamat IP ENI dalam file konfigurasi untuk membuat kueri klaster. Kemudian, klien menambahkan alamat IP ENI dari semua HSM di klaster untuk file informasi klaster.

- 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.

```
C:\Program Files\Amazon\CloudHSM>ksp_config.exe -enum
```

```
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.

```
C:\Program Files\Amazon\CloudHSM>cng_config.exe -register
```

Jika penyedia KSP hilang, jalankan perintah berikut.

```
C:\Program Files\Amazon\CloudHSM>ksp_config.exe -register
```

AWS CloudHSMPrasyarat Windows

Sebelum Anda dapat memulai klien AWS CloudHSM Windows dan menggunakan penyedia KSP dan CNG, Anda harus mengatur kredensial login untuk HSM pada sistem Anda. Anda dapat mengatur kredensial melalui Windows Credential Manager atau variabel lingkungan sistem. Kami sarankan Anda menggunakan Windows Credential Manager untuk menyimpan kredensial. Pilihan ini tersedia dengan klien AWS CloudHSM versi 2.0.4 dan yang lebih baru. Menggunakan variabel lingkungan lebih mudah untuk mengatur, tapi kurang aman daripada menggunakan Windows Credential Manager.

Pengelola Kredensional 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 pilihCredential 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 kripto (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 kripto (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 CU yang ada, gunakan listUsers.

Misalnya:

```
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 pengenal 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, seperti certutil, untuk menampilkan informasi sertifikat yang diperlukan. Output contoh 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: Mengaitkan kunci privat AWS CloudHSM dengan sertifikat

Untuk mengaitkan kunci dengan sertifikat, pertama-tama pastikan untuk memulai daemon klien AWS CloudHSM. 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 klien AWS CloudHSM masih berjalan. Kemudian, gunakan kata kerja certutil, -repairstore, untuk memperbarui nomor seri sertifikat. Sampel berikut menunjukkan perintah dan output. Lihat dokumentasi Microsoft untuk informasi tentangkata kerja -repairstore.

```
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 ProviderPrivate key is NOT exportableEncryption test
 passedCertUtil: -repairstore command completed successfully.
```

Setelah memperbarui nomor seri sertifikat, Anda dapat menggunakan sertifikat ini dan kunci privat AWS CloudHSM yang sesuai dengan alat penandatanganan pihak ketiga pada Windows.

Contoh kode untuk penyedia CNG

⚠ ** 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 AWS CloudHSMPrasyarat Windows.

```
// 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>

#ifndef 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;
}
```

SDK Klien Sebelumnya (SDK Klien 3)

AWS CloudHSMmencakup dua versi SDK Klien utama:

- SDK Klien 5: Ini adalah SDK Klien terbaru dan default kami. Untuk informasi tentang manfaat dan keuntungan yang diberikannya, lihatManfaat SDK 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 Klien 3 ke SDK Klien 5

Dokumentasi SDK 3 klien tercantum dalam topik ini.

Untuk mengunduh, lihatUnduh.

Periksa versi SDK klien Anda

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
```

Perbandingan 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 CNG dan KSP		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.

Topik

Platform yang didukung SDK Klien 3

Klien 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 SDKAWS CloudHSM Klien. Biasanya dukungan platform untuk komponen dalam SDK cocok dengan dukungan dasar, tetapi tidak selalu. Untuk menentukan dukungan platform untuk komponen tertentu, pertama pastikan platform yang Anda inginkan muncul

di bagian dasar untuk SDK, kemudian 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.

Daftar Isi

Dukungan Linux

- Amazon Linux
- Amazon Linux 2
- CentOS 6.10+ [2]
- CentOS 7.3+
- CentOS
- Red Hat Enterprise Linux (RHEL) 6.10+ [2]
- Red Hat Enterprise Linux (RHEL) 7.3+
- Red Hat Enterprise 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 Klien 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

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

Dukungan komponen

Pustaka PKCS #11

Pustaka PKCS #11 adalah satu-satunya komponen Linux yang cocok dengan dukungan dasar Linux. Untuk informasi lebih lanjut, lihat the section called “Dukungan Linux”.

Penyedia JCE

Penyedia JCE adalah satu-satunya komponen Linux yang cocok dengan dukungan dasar Linux. Untuk informasi lebih lanjut, lihat the section called “Dukungan Linux”.

- Membutuhkan OpenJDK 1.8

OpenSSL Dynamic Engine

OpenSSL Dynamic Engine adalah satu-satunya komponen Linux yang tidak cocok dengan dukungan dasar 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.

Penyedia CNG dan KSP

Penyedia CNG dan KSP adalah satu-satunya komponen Windows yang cocok dengan dukungan dasar Windows. Untuk informasi selengkapnya, lihat Dukungan Windows.

Memutakhirkan SDK Klien 3 di Linux

Dengan AWS CloudHSM Klien SDK 3.1 dan yang lebih tinggi, versi daemon klien dan komponen apa pun yang Anda instal harus cocok untuk peningkatan. 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 klien AWS CloudHSM 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 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
```

Ubuntu 20.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 22.04 LTS

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

Pustaka PKCS #11 untuk SDK Klien 3

PKCS #11 adalah standar untuk melakukan operasi kriptografi pada modul keamanan perangkat keras (HSM).

Untuk informasi tentang bootstrap, lihat. Menghubungkan ke cluster

Topik

- Menginstal Client SDK 3 untuk pustaka PKCS #11
- Mengautentikasi ke pustaka PKCS #11 (SDK Klien 3)
- Jenis kunci yang didukung (SDK Klien 3)
- Mekanisme yang didukung (SDK Klien 3)
- Operasi API yang didukung (SDK Klien 3)
- Atribut kunci yang didukung (SDK Klien 3)
- Contoh kode untuk pustaka PKCS #11 (SDK Klien 3)

Menginstal Client SDK 3 untuk pustaka PKCS #11

Prasyarat untuk Klien SDK 3

Pustaka PKCS #11 membutuhkan klien AWS CloudHSM.

Jika Anda belum menginstal dan mengatur konfigurasi klien AWS CloudHSM, 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 cloudhsm-client start
```

CentOS 7

```
$ sudo systemctl cloudhsm-client start
```

CentOS 8

```
$ sudo systemctl cloudhsm-client start
```

RHEL 7

```
$ sudo systemctl cloudhsm-client start
```

RHEL 8

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 20.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

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 (SDK Klien 3)

Bila Anda menggunakan pustaka PKCS #11, aplikasi Anda berjalan sebagai pengguna kripto (CU) khusus di HSM Anda. Aplikasi Anda dapat melihat dan mengelola hanya kunci yang CU miliki dan bagikan. Anda dapat menggunakan CU yang ada di HSM Anda atau membuat CU baru. Untuk informasi tentang mengelola CU, 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!
```

Jenis kunci yang didukung (SDK Klien 3)

Pustaka PKCS #11 mendukung jenis kunci berikut.

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 DES3 192-bit. Lihat catatan 1 di bawah untuk perubahan yang akan datang.
GENERIC_SECRET	Hasilkan 1 sampai 64 byte rahasia generik.

- [1] Dilarang setelah 2023 untuk kepatuhan FIPS sesuai panduan NIST. Lihat Kepatuhan FIPS 140: Penutupan Mekanisme 2024 untuk detail.

Mekanisme yang didukung (SDK Klien 3)

Pustaka PKCS #11 mendukung algoritme berikut ini:

- Enkripsi dan dekripsi— AES-CBC, AES-CTR, AES-ECB, AES-GCM, DES3-CBC, DES3-ECB, RSA-OAEP, dan RSA-PKCS
- Tanda tangani dan verifikasi- RSA, HMAC, dan ECDSA; dengan dan tanpa hashing
- Hash/digest— SHA1, SHA224, SHA256, SHA384, dan SHA512
- Bungkus kunci— Bungkus Kunci AES,[4] AES-GCM, RSA-AES, dan RSA-OAEP
- Derivasi kunci— ECDH, [5] SP800-108 CTR KDF

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. Kami tidak mendukung semua fungsi yang mungkin tercantum dalam spesifikasi PKCS #11. Tanda ✖ menunjukkan bahwa AWS CloudHSM belum mendukung mekanisme fungsi tertentu, meskipun standar PKCS #11 memungkinkannya. 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	Tandatangani & Verifikasi	SR & VR	Intisari	Enkripsi & Dekripsi	Turunkan Kunci	Bungkus & UnWrap
CKM_RSA_PKCS_KEY_PAIR_GEN	✓						
CKM_RSA_X9_31_KEY_PAIR_GEN	✓[2]						
CKM_RSA_X_509		✓			✓		
CKM_RSA_PKCS lihat catatan 8		✓[1]	✖		✓[1]		✓[1]
CKM_RSA_PKCS_OAEP					✓[1]		✓[6]
CKM_SHA1_RSA_PKCS		✓[3.2]					

Mekanisme	Fungsi						
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]					
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]					

Mekanisme	Fungsi						
CKM_SHA512_RSA_PKCS_PSS		✓[3.2]					
CKM_EC_KEY_PAIR_GEN	✓						
CKM_ECDSA		✓[1]					
CKM_ECDSA_SHA1		✓[3.2]					
CKM_ECDSA_SHA224		✓[3.2]					
CKM_ECDSA_SHA256		✓[3.2]					
CKM_ECDSA_SHA384		✓[3.2]					
CKM_ECDSA_SHA512		✓[3.2]					
CKM_ECDH1_DERIVE						✓[5]	
CKM_SP800_108_COUNTER_KDF						✓	
CKM_GENERIC_SECRET_KEY_GEN	✓						

Mekanisme	Fungsi						
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]	✓						
CKM_DES3_CBC lihat catatan [8]					✓[3.3]		✖
CKM_DES3_CBC_PAD lihat catatan [8]					✓		✖
CKM_DES3_ECB lihat catatan [8]					✓		✖
CKM_AES_GCM					✓[3.3, 4]		✓[7.1]

Mekanisme	Fungsi						
CKM_CLOUDHSM_AES_GCM					✓[7.1]		✓[7.1]
CKM_SHA_1				✓[3.1]			
CKM_SHA_1_HMAC		✓[3.3]					
CKM_SHA224				✓[3.1]			
CKM_SHA224_HMAC		✓[3.3]					
CKM_SHA256				✓[3.1]			
CKM_SHA256_HMAC		✓[3.3]					
CKM_SHA384				✓[3.1]			
CKM_SHA384_HMAC		✓[3.3]					
CKM_SHA512				✓[3.1]			
CKM_SHA512_HMAC		✓[3.3]					
CKM_RSA_AES_KEY_WRAP							✓

Mekanisme	Fungsi						
CKM_AES_KEY_WRAP							✓
CKM_AES_KEY_WRAP_PAD							✓
CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD							✓[7.1]
CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_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

Mekanisme	Ukuran Data Maksimum
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 CKM_ECDH1_DERIVEtidak 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/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

  Untuk informasi tambahan mengenai pembungkus kunci AES, lihat Pembungkus Kunci AES.
- [8] Dilarang setelah 2023 untuk kepatuhan FIPS sesuai panduan NIST. Lihat Kepatuhan FIPS 140: Penutupan Mekanisme 2024 untuk detail.

Operasi API yang didukung (SDK Klien 3)

Pustaka PKCS #11 mendukung operasi API PKCS #11 berikut.

- 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 yang didukung (SDK 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. Ketika Anda menggunakan pustaka PKCS #11, kami menetapkan nilai default sebagaimana 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 melewatkan templat atribut selama pembuatan objek, lihat sampel Hasilkan kunci melalui pustaka PKCS #11.

Menafsirkan tabel atribut pustaka PKCS #11

Tabel pustaka PKCS #11 berisi daftar atribut yang berbeda menurut jenis kunci. Hal ini menunjukkan apakah atribut tertentu didukung untuk jenis kunci tertentu ketika 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

Atribut	Tipe Kunci				Nilai Default
CKA_ENCRYPT	✖	✓	✖	✓	Salah
CKA_DECRYPT	✓	✖	✓	✖	Salah
CKA_DERIVE	✓	✓	✓	✓	Salah
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	Benar
CKA_DESTROYABLE	✓	✓	✓	✓	Benar
CKA_SIGN	✓	✖	✓	✖	Salah
CKA_SIGN_RECOVER	✖	✖	✓[3]	✖	
CKA_VERIFY	✖	✓	✖	✓	Salah
CKA_VERIFY_RECOVER	✖	✖	✖	✓[4]	
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	R	✖	R	✖	
CKA_EXTRACTABLE	✓	✖	✓	✖	Benar
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	
CKA_MODULUS	✖	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✓[2]	
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	✖	✖	✖	✓[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	✓	✓	✓	
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

Atribut	Tipe Kunci			Nilai Default
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	Benar
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	✓	✓	✓	Benar
CKA_ALWAYS_SENSITIVE	✖	✖	✖	

Atribut	Tipe Kunci			Nilai Default
CKA_EXTRACTABLE	✓	✓	✓	Benar
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	

Atribut	Tipe Kunci			Nilai Default
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

Atribut	Tipe Kunci							Nilai Default
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
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓	Benar
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓	Salah
CKA_SIGN_RECOVER	✖	✖	✓[3]	✖	✖	✖	✖	Salah
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓	Salah
CKA_VERIFY_RECOVER	✖	✖	✖	✓[4]	✖	✖	✖	
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	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	✖	✖	✓	✖	✖	✖	✖	

Atribut	Tipe Kunci							Nilai Default
CKA_PRIME_2	✖	✖	✓	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✓	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✓	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✓	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✓[2]	✖	✖	✖	✖	
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	✓	✓	✓	✓	✓	
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

Atribut	Tipe Kunci					Nilai Default
CKA_DESTROYABLE	✓	✓	✓	✓	✓	Benar
CKA_SIGN	✓	✓	✓	✓	✓	Salah
CKA_SIGN_RECOVER	✖	✓[3]	✖	✖	✖	Salah
CKA_VERIFY	✖	✖	✓	✓	✓	Salah
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	
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	✖	✖	✖	✖	✖	

Atribut	Tipe Kunci					Nilai Default
CKA_MODULUS_BITS	✖	✖	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	✖	✖	
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	✖	✖	✖	✖	✖	

Atribut	Tipe Kunci					Nilai Default
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]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	Benar
CKA_TOKEN	✓	✓	✓	Salah
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	Benar
CKA_ENCRYPT	✓	✓	✖	Salah
CKA_DECRYPT	✓	✓	✖	Salah

Atribut	Tipe Kunci			Nilai Default
CKA_DERIVE	✓	✓	✓	Salah
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	Benar
CKA_DESTROYABLE	✓[1]	✓[1]	✓[1]	Benar
CKA_SIGN	✓	✓	✓	Salah
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	Salah
CKA_VERIFY_RECOVER	✖	✖	✖	
CKA_WRAP	✓	✓	✖	Salah
CKA_UNWRAP	✓	✓	✖	Salah
CKA_SENSITIVE	✓	✓	✓	Benar
CKA_EXTRACTABLE	✓	✓	✓	Benar
CKA_NEVER_EXTRACTABLE	R	R	R	

Atribut	Tipe Kunci			Nilai Default
CKA_ALWAYS_SENSITIVE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	

Atribut	Tipe Kunci			Nilai Default
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
CKA_CLASS	✓	✓	✓	✓	✓	✓	✓
CKA_KEY_TYPE	✓	✓	✓	✓	✓	✓	✓
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓
CKA_ID	✓	✓	✓	✓	✓	✓	✓
CKA_LOCAL	✓	✓	✓	✓	✓	✓	✓
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓

Atribut	Tipe Kunci						
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]
CKA_ENCRYPT	✖	✖	✖	✓	✓	✓	✖
CKA_DECRYPT	✖	✖	✓	✖	✓	✓	✖
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	✓
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓
CKA_SIGN_RECOVER	✖	✖	✓	✖	✖	✖	✖
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓
CKA_VERIFY_RECOVER	✖	✖	✖	✓	✖	✖	✖
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✓

Atribut	Tipe Kunci						
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓
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	✖	✖	✖	✖

Atribut	Tipe Kunci						
CKA_COEFFICIENT	✖	✖	S	✖	✖	✖	✖
CKA_EXPONENT_1	✖	✖	S	✖	✖	✖	✖
CKA_EXPONENT_2	✖	✖	S	✖	✖	✖	✖
CKA_PRIVATE_EXPONENT	✖	✖	S	✖	✖	✖	✖
CKA_PUBLIC_EXPONENT	✖	✖	✓	✓	✖	✖	✖
CKA_EC_PARAMS	✓	✓	✖	✖	✖	✖	✖
CKA_EC_POINT	✖	✓	✖	✖	✖	✖	✖
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_RECOVER`Atribut 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_RECOVER`Atribut 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

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

Menentukan dalam templat suatu atribut yang tidak didukung oleh hasil khusus kunci dalam 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.

Kode Kesalahan	Deskripsi
`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 (SDK Klien 3)

Contoh kode GitHub menunjukkan kepada Anda cara menyelesaikan tugas dasar menggunakan pustaka PKCS #11.

Contoh prasyarat kode

Sebelum menjalankan sampel, lakukan langkah-langkah berikut untuk mengatur lingkungan Anda:

- Instal dan konfigurasikan 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

Menginstal Client SDK 3 untuk OpenSSL Dynamic Engine

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.

Topik

- Prasyarat untuk OpenSSL Dynamic Engine dengan Client SDK 3
- Instal OpenSSL Dynamic Engine untuk SDK Klien 3
- Gunakan OpenSSL Dynamic Engine untuk SDK Klien 3

Prasyarat untuk OpenSSL Dynamic Engine dengan Client SDK 3

Untuk informasi tentang dukungan platform, lihatPlatform yang didukung SDK Klien 3.

Sebelum Anda dapat menggunakan mesin dinamis AWS CloudHSM untuk OpenSSL, Anda memerlukan klien AWS CloudHSM.

Klien adalah daemon yang membangun komunikasi end-to-end terenkripsi dengan HSM di cluster Anda, dan mesin OpenSSL berkomunikasi secara lokal dengan klien. Untuk menginstal dan mengkonfigurasi AWS CloudHSM klien, lihatInstal klien (Linux). Kemudian gunakan perintah berikut untuk memulainya.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo systemctl cloudhsm-client start
```

CentOS 6

```
$ sudo systemctl start cloudhsm-client
```

CentOS 7

```
$ sudo systemctl cloudhsm-client start
```

RHEL 6

```
$ sudo systemctl start cloudhsm-client
```

RHEL 7

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Instal OpenSSL Dynamic Engine untuk SDK Klien 3

Langkah-langkah berikut menjelaskan cara menginstal dan mengatur konfigurasi mesinAWS CloudHSM dinamis untuk OpenSSL. Untuk informasi selengkapnya, lihatMemutakhirkan 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`.

2. Gunakan perintah berikut untuk mengatur variabel lingkungan bernama `n3fips_password` yang berisi kredensial dari pengguna kripto (CU).

```
$ export n3fips_password=<HSM user name>:<password>
```

Gunakan OpenSSL Dynamic Engine untuk SDK Klien 3

Untuk menggunakan mesin dinamis AWS CloudHSM untuk OpenSSL dari aplikasi terintegrasi OpenSSL, memastikan 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 dinamis AWS CloudHSM untuk OpenSSL dari baris perintah OpenSSL, gunakan opsi `-engine` untuk menentukan mesin dinamis OpenSSL bernama `cloudhsm`. Misalnya:

```
$ openssl s_server -cert server.crt -key server.key -engine cloudhsm
```

Klien SDK 3 untuk penyedia JCE

Penyedia JCE AWS CloudHSM adalah implementasi penyedia dibangun dari kerangka kerja 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 dan gunakan penyedia AWS CloudHSM JCE untuk Client SDK 3

Sebelum Anda dapat menggunakan penyedia JCE, Anda memerlukan klien AWS CloudHSM.

Klien adalah daemon yang membangun komunikasi end-to-end terenkripsi dengan HSM di cluster Anda. Penyedia JCE berkomunikasi secara lokal dengan klien. Jika Anda belum menginstal dan mengatur konfigurasi paket klien AWS CloudHSM, 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 cloudhsm-client start
```

CentOS 7

```
$ sudo systemctl cloudhsm-client start
```

CentOS 8

```
$ sudo systemctl cloudhsm-client start
```

RHEL 7

```
$ sudo systemctl cloudhsm-client start
```

RHEL 8

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 20.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Topik

Menginstal 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 Memutakhirkan 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`

Memvalidasi 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
   ```

2. Gunakan perintah berikut untuk mengatur variabel lingkungan yang diperlukan. Ganti *<nama pengguna HSM>* dan *<kata sandi>* dengan kredensial pengguna kripto (CU).

   ```
   $ export LD_LIBRARY_PATH=/opt/cloudhsm/lib
   ```

   ```
   $ export HSM_PARTITION=PARTITION_1
   ```

   ```
   $ export HSM_USER=<HSM user name>
   ```

   ```
   $ export HSM_PASSWORD=<password>
   ```

3. 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)
```

Memberikan kredensi kepada penyedia JCE

HSM perlu mengautentikasi aplikasi Java Anda sebelum aplikasi dapat menggunakannya. Setiap aplikasi bisa menggunakan satu sesi. HSM mengautentikasi sesi dengan menggunakan baik metode login eksplisit atau 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 untuk dipahami ketika kredensial tidak valid atau HSM mengalami masalah dalam sesi autentikasi.

Dasar-dasar manajemen utama di penyedia JCE

Dasar-dasar manajemen kunci dalam penyedia JCE melibatkan mengimpor kunci, mengekspor 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 Klien SDK 3

Untuk informasi tentang antarmuka dan kelas mesin Java Cryptography Architecture (JCA) yang didukung oleh AWS CloudHSM, lihat topik berikut.

Topik

Kunci yang didukung

Pustaka perangkat lunak AWS CloudHSM 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 perangkat lunak AWS CloudHSM untuk Java mendukung algoritme, mode, dan kombinasi bantalan berikut.

Algoritme	Mode	Bantalan	Catatan
AES	CBC	`AES/CBC/NoPadding` `AES/CBC/PKCS5Padding`	Menerapkan `Cipher.ENCRYPT_MODE` dan `Cipher.DECRYPT_MODE` .
AES	ECB	`AES/ECB/NoPadding` `AES/ECB/PKCS5Padding`	Menerapkan `Cipher.ENCRYPT_MODE` dan `Cipher.DECRYPT_MODE` . Gunakan AES Transformasi.
AES	CTR	`AES/CTR/NoPadding`	Menerapkan `Cipher.ENCRYPT_MODE` dan `Cipher.DECRYPT_MODE` .
AES	GCM	`AES/GCM/NoPadding`	Menerapkan `Cipher.ENCRYPT_MODE` dan `Cipher.DECRYPT_MODE` , `Cipher.WRAP_MODE` , dan `Cipher.UNWRAP_MODE` .

Algoritme	Mode	Bantalan	Catatan
			Ketika melakukan enkripsi AES-GCM, HSM mengabaikan vektor inisialisasi (IV) dalam permintaan dan menggunakan IV yang dihasilkannya. Saat operasi selesai, Anda harus memanggil `Cipher.getIV()` untuk mendapatkan IV.

AESWrap	ECB	`AESWrap/ECB/ZeroPadding`
		
		`AESWrap/ECB/NoPadding`
		
		`AESWrap/ECB/PKCS5Padding`
			Menerapkan `Cipher.WRAP_MODE` dan `Cipher.UNWRAP_MODE`. Gunakan AES Transformasi.

Algoritme	Mode	Bantalan	Catatan

DESede (Tiga DES)	CBC	`DESede/CBC/NoPadding`
		
		`DESede/CBC/PKCS5Padding`
			Menerapkan `Cipher.ENCRYPT_MODE` dan `Cipher.DECRYPT_MODE` .
			
			Rutinitas pembuatan kunci menerima ukuran 168 atau 192 bit. Namun, secara internal, semua kunci DESede 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 `Cipher.ENCRYPT_MODE` dan `Cipher.DECRYPT_MODE` .
			
			Rutinitas pembuatan kunci menerima ukuran 168 atau 192 bit. Namun, secara internal, semua kunci DESede adalah 192 bit.
			
			Lihat catatan 1 di bawah untuk perubahan yang akan datang.

RSA	ECB	`RSA/ECB/NoPadding`
		
		`RSA/ECB/PKCS1Padding`
			Menerapkan `Cipher.ENCRYPT_MODE` dan `Cipher.DECRYPT_MODE` .
			
			Lihat catatan 1 di bawah untuk perubahan yang akan datang.

Algoritme	Mode	Bantalan	Catatan

RSA	ECB	`RSA/ECB/OAEPPadding`
		
		`RSA/ECB/OAEPWithSHA-1ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-224ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-256ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-384ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-512ANDMGF1Padding`
			Menerapkan `Cipher.ENCRYPT_MODE`, `Cipher.DECRYPT_MODE`, `Cipher.WRAP_MODE`, dan `Cipher.UNWRAP_MODE`.
			
			`OAEPPadding` adalah OAEP dengan jenis bantalan `SHA-1`.

RSAAESWrap	ECB	`OAEPPADDING`	Menerapkan `Cipher.WRAP_Mode` dan `Cipher.UNWRAP_MODE`.

Intisari yang didukung

Pustaka perangkat lunak AWS CloudHSM untuk Java mendukung digest 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

Pustaka perangkat lunak AWS CloudHSM untuk Java mendukung algoritme HMAC berikut.

- `HmacSHA1`
- `HmacSHA224`
- `HmacSHA256`
- `HmacSHA384`
- `HmacSHA512`

Mekanisme tanda/verifikasi yang didukung

Pustaka perangkat lunak AWS CloudHSM 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] Dilarang setelah 2023 untuk kepatuhan FIPS sesuai panduan NIST. Lihat Kepatuhan FIPS 140: Penutupan Mekanisme 2024 untuk rincian selengkapnya.

Atribut kunci Java yang didukung untuk Client SDK 3

Topik ini menjelaskan cara menggunakan ekstensi proprietary untuk pustaka Java versi 3.1 untuk mengatur atribut kunci. 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

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 direplikasi di semua HSM di klaster dan termasuk 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 menggunak an kunci untuk membungkus kunci lain.
CKA_SIGN	TRUE		TRUE	Benar menunjukkan Anda dapat menggunak an 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 menggunak an 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.

Menetapkan 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	get(keyAttribute)
Isi nilai satu atribut kunci	Nilai sebelumnya terkait dengan atribut kunci, atau nol jika tidak ada pemetaan untuk atribut kunci	put(keyAttribute, value)
Isi nilai untuk beberapa atribut kunci	T/A	PutAll () keyAttributesMap
Hapus pasangan nilai kunci dari peta atribut	Nilai sebelumnya terkait dengan atribut kunci, atau nol jika tidak ada pemetaan untuk atribut kunci	remove(keyAttribute)

> **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 pasangan kunci

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. Misalnya:

```
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 lebih lanjut 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 kripto 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	`CaviumKeyGenAlgorithmParameterSpec`	`CaviumKeyGenAlgorithmParameterSpec(CloudHsmKeyAttributesMap keyAttributesMap)`
DES	`CaviumDESKeyGenParameterSpec`	`CaviumDESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttributesMap keyAttributesMap)`
RSA	`CaviumRSAKeyGenParameterSpec`	`CaviumRSAKeyGenParameterSpec(int`

Tipe Kunci	Kelas Spesifikasi Parameter	Contoh Konstruktor
		`keysize, BigInteger publicExponent, CloudHsmKeyPairAttributesMap keyPairAttributesMap)`
Rahasia	`CaviumGenericSecretKeyGenParameterSpec`	`CaviumGenericSecretKeyGenParameterSpec(int size, CloudHsmKeyAttributesMap key AttributesMap)`
AES	`CaviumAESKeyGenParameterSpec`	`CaviumAESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttributesMap key AttributesMap)`
EC	`CaviumECGenParameterSpec`	`CaviumECGenParameterSpec(String stdName, CloudHsmKeyPairAttributesMap keyPairAttributesMap)`

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

Prasyarat

Sebelum menjalankan sampel, Anda harus mengatur lingkungan Anda:

- Instal dan konfigurasi Penyedia Java Cryptographic Extension (JCE) dan paket klienAWS 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] Dilarang setelah 2023 untuk kepatuhan FIPS sesuai panduan NIST. Lihat Kepatuhan FIPS 140: Penutupan Mekanisme 2024 untuk rincian selengkapnya.

MenggunakanAWS CloudHSMKeyStore Java untuk Klien SDK 3

Kelas AWS CloudHSM `KeyStore` menyediakan penyimpanan kunci PKCS12 tujuan khusus yang mengizinkan akses ke AWS CloudHSM 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 pada HSM.

Kelas AWS CloudHSM `KeyStore` mengimplementasikan Service Provider Interface (SPI) `KeyStore` dari Java Cryptography Extension (JCE). Untuk informasi lebih lanjut tentang penggunaan `KeyStore`, lihatKeyStore Kelas.

Memilih toko kunci yang tepat

Penyedia Java Cryptographic Extension (JCE) AWS CloudHSM dilengkapi dengan penyimpanan kunci hanya baca, default pass-through, yang meneruskan semua transaksi ke HSM. Penyimpanan kunci default ini berbeda dari AWS CloudHSM Keystore tujuan khusus. Dalam kebanyakan situasi, Anda akan mendapatkan performa waktu aktif yang lebih baik dan throughput dengan menggunakan default. Anda sebaiknya hanya menggunakan KeyStore AWS CloudHSM untuk aplikasi di mana Anda memerlukan dukungan untuk sertifikat dan operasi berbasis sertifikat selain pembongkaran operasi kunci pada 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 KeyStore tujuan khusus sebagai berikut:

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

Menginisialisasi KeyStore AWS CloudHSM

Masuk ke KeyStore AWS CloudHSM dengan cara yang sama seperti Anda login ke penyedia JCE. Anda dapat menggunakan variabel lingkungan atau file properti sistem, dan Anda harus login sebelum Anda 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 file PKCS12 lokal yang menyimpan data penyimpanan kunci. Saat Anda membuat Keystore AWS CloudHSM, Anda mengatur kata sandi dan memberikannya ketika menggunakan metode load, set, dan get.

Instantiate objek KeyStore CloudHSM baru sebagai berikut:

```
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);
```

MenggunakanAWS CloudHSMKeystore

Sebuah objek CloudHSM KeyStore umumnya digunakan melalui aplikasi pihak ketiga seperti jarsigner atau keytool. Anda juga dapat mengakses objek secara langsung dengan kode.

KeyStore AWS CloudHSM sesuai dengan spesifikasi KeyStore Kelas JCE 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 file PKCS12 dan objek yang ada di HSM.

  Kode sampel:

  ```
  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. Penyimpanan kunci memeriksa objek yang disimpan secara lokal dalam file PKCS12 dan bendaobjek yang ada di HSM.

- `DeleteEntry`

Menghapus entri sertifikat dari file PKCS12 lokal. Menghapus data kunci yang disimpan dalam HSM tidak didukung menggunakan Keystore AWS CloudHSM. 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. Karena `getKey` langsung membuat kueri ke HSM, ini dapat digunakan untuk setiap kunci pada HSM, terlepas dari apakah ini 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 file PKCS12 dan HSM untuk alias.

- `SetCertificateEntry`

  Menetapkan sertifikat yang diberikan untuk alias yang diberikan. Jika alias yang diberikan sudah digunakan untuk me ngidentifikasi kunci atau sertifikat,`KeyStoreException`dikeluarkan. Anda dapat menggunakan kode JCE untuk mendapatkan objek kunci dan kemudian menggunakan metode `SetKeyEntry` KeyStore untuk mengaitkan sertifikat dengan kunci.

- `SetKeyEntry` dengan `byte[]` kunci

  API ini saat ini tidak didukung dengan SDK Klien 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 untuk pengaliran output yang diberikan sebagai file PKCS12 dan mengamankannya dengan kata sandi yang diberikan. Selain itu, semua kunci dimuat tetap ada (yang ditetapkan menggunakan panggilan`setKey`).

Mengintegrasikan aplikasi pihak ketiga denganAWS CloudHSM

Beberapa kasus penggunaan untuk AWS CloudHSM melibatkan pengintegrasian aplikasi perangkat lunak pihak ketiga dengan HSM di klaster AWS CloudHSM. Dengan mengintegrasikan perangkat lunak pihak ketiga dengan 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 pembongkaran SSL/TLS di AWS CloudHSM
- Konfigurasikan Windows Server sebagai otoritas sertifikat (CA) dengan AWS CloudHSM
- Enkripsi data transparan (TDE) database Oracle dengan AWS CloudHSM
- Menggunakan Microsoft SignTool bersamaAWS CloudHSMuntuk menandatangani file
- Java Keytool dan Jarsigner
- Integrasi vendor pihak ketiga lainnya

Tingkatkan keamanan server web Anda dengan pembongkaran SSL/TLS 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. Web server menggunakan pasangan kunci publik-privat dan sertifikat kunci publik SSL/TLS 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 di HSM.

Topik berikut memberikan ikhtisar tentang bagaimana SSL/TLS offload dengan AWS CloudHSM karya dan tutorial untuk menyiapkan SSL/TLS offload dengan pada platform berikut. AWS CloudHSM

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

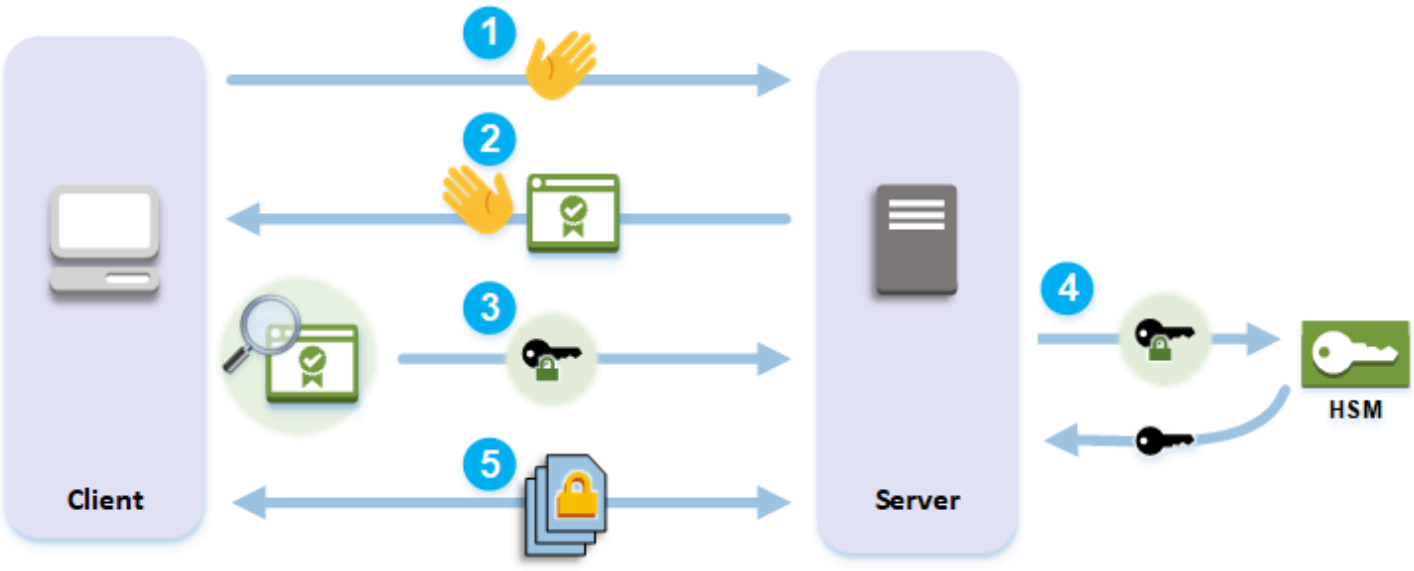
Topik

- Cara Pembongkaran SSL/TLS denganAWS CloudHSMbekerja
- Pemuatan SSL/TLS di Linux
- Menggunakan pembongkaran SSL/TLS di Windows
- Tambahkan load balancer dengan Elastic Load Balancing (opsional)

Cara Pembongkaran SSL/TLS denganAWS CloudHSMbekerja

Untuk membuat sambungan HTTPS, server web Anda melakukan proses jabat tangan dengan klien. Sebagai bagian dari proses ini, server membongkar beberapa pengolahan kriptografi ke HSM, 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 server SSL/TLS ditandatangani oleh sertifikat root yang klien percaya.
   b. Ekstraksi kunci publik dari sertifikat.
   c. Menghasilkan rahasia premaster dan mengenkripsi dengan kunci publik server.
   d. Mengirim rahasia premaster terenkripsi ke server.
4. Untuk mendekripsi rahasia premaster klien, server mengirimkannya ke HSM. HSM menggunakan kunci privat di HSM untuk mendekripsi rahasia premaster dan kemudian mengirimkan rahasia premaster ke server. Secara independen, klien dan server masing-masing menggunakan rahasia premaster dan beberapa informasi dari pesan halo untuk menghitung rahasia master.
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 mengatur konfigurasi pembongkaran SSL/TLS dengan AWS CloudHSM, lihat salah satu topik berikut:

- Pemuatan SSL/TLS di Linux
- Menggunakan pembongkaran SSL/TLS di Windows

Pemuatan SSL/TLS di Linux

DenganAWS CloudHSM, Anda dapat melakukan offload SSL/TLS di Linux dengan NGINX, Apache, dan Tomcat. Untuk informasi lebih lanjut, lihat topik di bawah.

Topik

- Menggunakan NGINX atau Apache dengan OpenSSL untuk pembongkaran SSL/TLS di Linux
- Menggunakan Tomcat dengan JSSE untuk pembongkaran SSL/TLS di Linux

Menggunakan NGINX atau Apache dengan OpenSSL untuk pembongkaran SSL/TLS di Linux

Topik ini memberikan step-by-step instruksi untuk menyiapkan pembongkaran SSL/TLS denganAWS CloudHSM pada server web Linux.

Topik

Gambaran Umum

Di Linux, perangkat lunak server web NGINX dan Apache HTTP Server terintegrasi dengan OpenSSL untuk mendukung HTTPS. Parameter mesin dinamis AWS CloudHSM untuk OpenSSL menyediakan antarmuka yang memungkinkan perangkat lunak server web menggunakan HSM di klaster Anda untuk pembongkaran kriptografi dan penyimpanan kunci. Mesin OpenSSL adalah jembatan yang menghubungkan server web ke klaster 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 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: Mengatur prasyarat.

Langkah 1: Mengatur prasyarat

Platform yang berbeda memerlukan prasyarat yang berbeda. Gunakan bagian prasyarat di bawah ini yang sesuai dengan platform Anda.

Topik

- Prasyarat untuk Klien SDK 3

Prasyarat untuk Client SDK 5

Untuk menyiapkan server web pembongkaran SSL/TLS dengan Klien SDK 5, Anda memerlukan yang berikut:

- Klaster AWS CloudHSM 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 kripto (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 atur 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 klaster Anda, instal server web NGINX atau Apache:

   Amazon Linux

   - NGINX

     ```
     $ sudo yum install nginx
     ```

   - Apache

     ```
     $ sudo yum install httpd24 mod24_ssl
     ```

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 perpustakaan AWS CloudHSM OpenSSL Dynamic Engine ke lokasi yang diharapkan versi 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
  ```

CentOS 7

- Untuk informasi tentang cara mengunduh versi terbaru NGINX di CentOS 7, lihat situs web NGINX.

  Versi terbaru NGINX yang tersedia untuk CentOS 7 menggunakan versi OpenSSL yang lebih baru dari versi sistem OpenSSL. Setelah menginstal NGINX, Anda perlu membuat tautan simbolis dari perpustakaan AWS CloudHSM OpenSSL Dynamic Engine ke lokasi yang diharapkan versi 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
  ```

Red Hat 7

- Untuk informasi tentang cara mengunduh versi terbaru NGINX di Red Hat 7, lihat situs web NGINX.

  Versi terbaru dari NGINX tersedia untuk Red Hat 7 menggunakan versi OpenSSL yang lebih baru dari versi sistem OpenSSL. Setelah menginstal NGINX, Anda perlu membuat tautan simbolis dari perpustakaan AWS CloudHSM OpenSSL Dynamic Engine ke lokasi yang diharapkan versi 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
  ```

CentOS 8

- 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
  ```

Ubuntu 18.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

Ubuntu 20.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

Ubuntu 22.04

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

3. Gunakan CloudHSM CLI untuk membuat 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.

Setelah Anda menyelesaikan langkah ini, buka Langkah 2: Menghasilkan atau mengimpor kunci pribadi dan sertifikat SSL/TLS.

Catatan

- Untuk menggunakan Security-Enhanced Linux (SELinux) dan server web, Anda harus mengizinkan koneksi TCP keluar pada port 2223, yaitu port yang Klien SDK 5 gunakan 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 klaster aktif dengan satu instans klien HSM dan 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.
- 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.

Prasyarat untuk Klien SDK 3

Untuk mengatur pembongkaran SSL/TLS server web dengan Client SDK 3, Anda memerlukan yang berikut ini:

- Klaster AWS CloudHSM aktif dengan setidaknya satu HSM.
- Instans Amazon EC2 menjalankan sistem operasi Linux dengan perangkat lunak berikut diinstal:
  - Klien AWS CloudHSM dan alat baris perintah.
  - Aplikasi server web NGINX atau Apache.
  - Mesin dinamis AWS CloudHSM untuk OpenSSL.
- Pengguna kripto (CU) harus memiliki dan mengelola kunci privat server web pada HSM.

Untuk mengatur sebuah instans server web Linux dan membuat CU pada HSM

1. Selesaikan langkah-langkah dalam Mulai. Anda kemudian akan memiliki klaster aktif dengan satu HSM dan instans klien Amazon EC2. Instans EC2 Anda akan dikonfigurasi dengan alat baris perintah. Gunakan instans klien ini sebagai server web Anda.

2. Hubungkan ke instans klien Anda. 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.
3. Pada instans EC2 Linux yang memiliki akses ke klaster Anda, instal server web NGINX atau Apache:

   Amazon Linux

   - NGINX

     ```
     $ sudo yum install nginx
     ```

   - Apache

     ```
     $ sudo yum install httpd24 mod24_ssl
     ```

   Amazon Linux 2

   - NGINX versi 1.19 adalah versi terbaru dari NGINX yang kompatibel dengan mesin Client SDK 3 di Amazon Linux 2.

     Untuk informasi lebih lanjut dan untuk mengunduh NGINX versi 1.19, lihat situs web NGINX.

   - Apache

     ```
     $ sudo yum install httpd mod_ssl
     ```

   CentOS 7

   - NGINX versi 1.19 adalah versi terbaru dari NGINX yang kompatibel dengan mesin Client SDK 3 pada CentOS 7.

     Untuk informasi lebih lanjut dan untuk mengunduh NGINX versi 1.19, lihat situs web NGINX.

   - Apache

     ```
     $ sudo yum install httpd mod_ssl
     ```

Red Hat 7

- NGINX versi 1.19 adalah versi terbaru dari NGINX yang kompatibel dengan mesin Client SDK 3 pada Red Hat 7.

  Untuk informasi lebih lanjut dan untuk mengunduh NGINX versi 1.19, lihat situs web NGINX.

- Apache

```
$ sudo yum install httpd mod_ssl
```

Ubuntu 16.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

Ubuntu 18.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

4. (Opsional) Tambahkan lebih banyak HSM ke klaster Anda. Untuk informasi lebih lanjut, lihat Menambahkan HSM.
5. Gunakan cloudhsm_mgmt_util untuk membuat CU. Untuk informasi lebih lanjut, lihat Mengelola pengguna HSM. 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: Menghasilkan atau mengimpor kunci pribadi dan sertifikat SSL/TLS.

Langkah 2: Menghasilkan atau mengimpor kunci pribadi dan sertifikat SSL/TLS

Untuk mengaktifkan HTTPS, aplikasi server web Anda (NGINX atau Apache) memerlukan kunci privat dan sertifikat SSL/TLS yang sesuai. Untuk menggunakan pembongkaran SSL/TLS server web dengan AWS CloudHSM, Anda harus menyimpan kunci privat di HSM klaster AWS CloudHSM Anda. Anda dapat mencapai ini dengan salah satu cara berikut:

- Jika Anda belum memiliki kunci pribadi dan sertifikat yang sesuai, buat kunci pribadi di HSM. Anda menggunakan kunci privat untuk membuat permintaan penandatanganan sertifikat (CSR), yang Anda gunakan untuk membuat sertifikat SSL/TLS.
- Jika Anda sudah memiliki kunci pribadi dan sertifikat yang sesuai, impor kunci privat ke dalam HSM.

Terlepas dari metode mana yang Anda pilih sebelumnya, Anda 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 (ini bukan kunci pribadi yang sebenarnya). Server web Anda menggunakan file kunci pribadi PEM palsu untuk mengidentifikasi kunci pribadi pada HSM selama pembongkaran SSL/TLS.

Lakukan salah satu dari berikut:

- Membuat kunci pribadi dan sertifikat
- Mengimpor kunci privat dan sertifikat yang ada

Membuat kunci pribadi dan sertifikat

Menghasilkan kunci pribadi

Bagian ini menunjukkan cara membuat keypair menggunakan Key Management Utility (KMU) dari Client SDK 3. Setelah Anda memiliki pasangan kunci yang dihasilkan di dalam HSM, Anda dapat mengekspornya sebagai file PEM palsu, dan menghasilkan sertifikat yang sesuai.

Kunci pribadi yang dihasilkan dengan Key Management Utility (KMU) dapat digunakan dengan Client SDK 3 dan Client SDK 5.

Instal dan konfigurasikan Key Management Utility (KMU)

1. Hubungkan ke instans klien Anda.
2. Instal dan Konfigurasi SDK Klien 3.
3. Jalankan perintah berikut untuk memulai klien AWS CloudHSM.

   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
   ```

Ubuntu 20.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 22.04 LTS

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

4. Jalankan perintah berikut untuk memulai alat baris perintah key_mgmt_util.

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

5. Jalankan perintah berikut untuk masuk ke HSM. Ganti *<nama pengguna>* dan *<kata sandi>* dengan nama pengguna dan kata sandi pengguna kriptografi (CU).

```
Command: loginHSM -u CU -s <user name> -p <password>>
```

Menghasilkan Kunci Pribadi

Tergantung pada kasus penggunaan Anda, Anda dapat menghasilkan RSA atau pasangan kunci EC. Lakukan salah satu dari berikut:

- Untuk menghasilkan kunci pribadi RSA pada HSM

  Gunakan `genRSAKeyPair` perintah untuk menghasilkan pasangan kunci RSA. *Contoh ini menghasilkan pasangan kunci RSA dengan modulus 2048, eksponen publik 65537, dan label tls_rsa_keypair.*

  ```
  Command: genRSAKeyPair -m 2048 -e 65537 -l tls_rsa_keypair
  ```

  Jika perintah berhasil, Anda akan melihat output berikut yang menunjukkan bahwa Anda telah berhasil menghasilkan pasangan kunci RSA.

  ```
  Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

                  Cfm3GenerateKeyPair:    public key handle: 7    private key handle: 8

                  Cluster Status:
                  Node id 1 status: 0x00000000 : HSM Return: SUCCESS
  ```

- Untuk menghasilkan kunci pribadi EC pada HSM

  Gunakan `genECCKeyPair` perintah untuk menghasilkan pasangan kunci EC. Contoh ini menghasilkan pasangan kunci EC dengan ID kurva 2 (sesuai dengan `NID_X9_62_prime256v1` kurva) dan label *`tls_ec_keypair`*.

  ```
  Command: genECCKeyPair -i 2 -l tls_ec_keypair
  ```

  Jika perintah berhasil, Anda akan melihat output berikut yang menunjukkan bahwa Anda telah berhasil membuat pasangan kunci EC.

  ```
  Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

                  Cfm3GenerateKeyPair:    public key handle: 7    private key handle: 8

                  Cluster Status:
                  Node id 1 status: 0x00000000 : HSM Return: SUCCESS
  ```

Mengekspor file kunci pribadi PEM palsu

Setelah Anda memiliki kunci pribadi pada HSM, Anda harus mengekspor file kunci pribadi PEM palsu. File ini tidak berisi data kunci aktual, 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.

> **Note**
>
> File PEM palsu yang dihasilkan dengan Key Management Utility (KMU) dapat digunakan dengan Client SDK 3 dan Client SDK 5.

Identifikasi pegangan kunci yang sesuai dengan kunci yang ingin Anda ekspor sebagai PEM palsu, lalu jalankan perintah berikut untuk mengekspor kunci pribadi dalam format PEM palsu dan menyimpannya ke file. Ganti nilai berikut dengan nilai Anda sendiri:

- <private_key_handle>- Menangani kunci pribadi yang dihasilkan. Pegangan ini dihasilkan oleh salah satu perintah generasi kunci pada langkah sebelumnya. Dalam contoh sebelumnya, handel kunci privat adalah 8.
- <web_server_fake_PEM.key>- Nama file yang akan ditulis kunci PEM palsu Anda.

```
Command: getCaviumPrivKey -k <private_key_handle> -out <web_server_fake_PEM.key>
```

Keluar

Jalankan perintah berikut untuk menghentikan key_mgmt_util.

```
Command: exit
```

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.

Membuat sertifikat yang ditandatangani sendiri

Setelah Anda membuat kunci pribadi PEM palsu, Anda dapat menggunakan file ini untuk membuat 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 sertifikat SSL/TLS yang ditandatangani 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 konfigurasikan OpenSSL Dynamic Engine

1. Hubungkan ke instans klien Anda.
2. Untuk menginstal dan mengkonfigurasi, lakukan salah satu dari berikut ini:
   - the section called “Memasang OpenSSL Dynamic Engine”
   - the section called “OpenSSL Dynamic Engine”

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 privat PEM Anda yang palsu. Ganti *<web_server.csr>* dengan nama file yang berisi CSR Anda.

   Perintah `req` bersifat interaktif. Tanggapi setiap bidang. Informasi bidang disalin ke sertifikat SSL/TLS Anda.

   ```
   $ openssl req -engine cloudhsm -new -key <web_server_fake_PEM.key> -
   out <web_server.csr>
   ```
3. Membuat 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 privat 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: Konfigurasikan server web.

Mengimpor kunci privat dan sertifikat yang ada

Anda mungkin sudah memiliki kunci privat dan sertifikat SSL/TLS yang sesuai yang Anda gunakan untuk HTTPS di server web Anda. Jika demikian, Anda dapat mengimpor kunci itu ke HSM dengan mengikuti langkah-langkah dalam bagian ini.

> **Note**
>
> Beberapa catatan tentang impor kunci privat dan kompatibilitas Klien SDK:
>
> - Mengimpor kunci privat yang ada memerlukan Klien SDK 3.
> - Anda dapat menggunakan kunci privat dari Klien SDK 3 dengan Klien SDK 5.
> - Mesin Dinamis OpenSSL untuk Klien SDK 3 tidak mendukung platform Linux terbaru, tetapi implementasi Mesin Dinamis OpenSSL untuk Klient SDK 5 mendukung. Anda dapat mengimpor kunci pribadi yang ada menggunakan Key Management Utility (KMU) yang disediakan dengan Client SDK 3, kemudian menggunakan kunci pribadi tersebut dan implementasi OpenSSL Dynamic Engine dengan Client SDK 5 untuk mendukung pembongkaran SSL/TLS pada platform Linux terbaru.

Untuk mengimpor kunci privat yang ada ke HSM dengan klien SDK 3

1. Hubungkan ke instans klien Amazon EC2 Anda. Jika perlu, salin kunci privat dan sertifikat yang ada ke instans.
2. Instal dan Konfigurasikan SDK Klien 3
3. Jalankan perintah berikut untuk memulai klien AWS CloudHSM.

   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
```

Ubuntu 20.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 22.04 LTS

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

4. Jalankan perintah berikut untuk memulai alat baris perintah key_mgmt_util.

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

5. Jalankan perintah berikut untuk masuk ke HSM. Ganti *<nama pengguna>* dan *<kata sandi>* dengan nama pengguna dan kata sandi pengguna kriptografi (CU).

```
Command: loginHSM -u CU -s <user name> -p <password>
```

6. Jalankan perintah berikut untuk mengimpor kunci privat Anda ke HSM.

   a. Jalankan perintah berikut untuk membuat kunci pembungkus simetris yang berlaku untuk sesi saat ini saja. Perintah dan output ditampilkan.

```
Command: genSymKey -t 31 -s 16 -sess -l wrapping_key_for_import

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
```

   b. Jalankan perintah berikut untuk mengimpor kunci privat yang ada ke HSM. Perintah dan output ditampilkan. Ganti nilai berikut dengan nilai Anda sendiri:

   - *<web_server_existing.key>* — Nama file yang berisi kunci privat Anda.
   - *<web_server_imported_key>* — Label untuk kunci privat yang diimpor.
   - *<wrapping_key_handle>* — Membungkus handel kunci yang dihasilkan oleh perintah sebelumnya. Pada contoh sebelumnya, handel kunci pembungkus adalah 6.

```
Command: importPrivateKey -f <web_server_existing.key> -
l <web_server_imported_key> -w <wrapping_key_handle>

BER encoded key length is 1219
Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS
Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS
Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS
Private Key Unwrapped.  Key Handle: 8
Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

7. Jalankan perintah berikut untuk mengekspor kunci privat dalam format PEM palsu dan menyimpannya ke file. Ganti nilai berikut dengan nilai Anda sendiri:

   - *<private_key_handle>* — Menangani kunci privat yang diimpor. Handel ini dihasilkan oleh perintah kedua di langkah sebelumnya. Dalam contoh sebelumnya, handel kunci privat adalah 8.

- *<web_server_fake_PEM.key>* — Nama file yang berisi kunci privat PEM palsu yang diekspor.

```
Command: getCaviumPrivKey -k <private_key_handle> -out <web_server_fake_PEM.key>
```

8. Gunakan perintah untuk menghentikan key_mgmt_util.

```
Command: exit
```

Setelah Anda menyelesaikan langkah ini, buka Langkah 3: Konfigurasikan server web.

Langkah 3: Konfigurasikan 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 pembongkaran SSL/TLS dengan AWS CloudHSM.

Menyelesaikan langkah-langkah dari salah satu bagian berikut.

Topik

- Konfigurasikan server web NGINX
- Mengkonfigurasi server web Apache

Konfigurasikan 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 privat PEM Anda yang palsu.

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/nginx/private/server.key
```

5. Jalankan perintah berikut untuk mengubah kepemilikan file sehingga pengguna bernamanginx 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 klaster dapat mendukung maksimum 1000 proses pekerja NGINX di semua server web NGINX.

Amazon Linux

Gunakan editor teks untuk mengedit file `/etc/nginx/nginx.conf`. Ini memerlukan izin root Linux. Di bagian atas file, tambahkan baris berikut:

- Jika menggunakan Client SDK 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- Jika menggunakan Client SDK 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Kemudian tambahkan yang berikut 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 2

Gunakan editor teks untuk mengedit file `/etc/nginx/nginx.conf`. Ini memerlukan izin root Linux. Di bagian atas file, tambahkan baris berikut:

- Jika menggunakan Client SDK 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- Jika menggunakan Client SDK 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Kemudian tambahkan yang berikut 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 {
    }
}
```

CentOS 7

Gunakan editor teks untuk mengedit file `/etc/nginx/nginx.conf`. Ini memerlukan izin root Linux. Di bagian atas file, tambahkan baris berikut:

- Jika menggunakan Client SDK 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- Jika menggunakan Client SDK 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Kemudian tambahkan yang berikut 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 {
    }
}
```

CentOS 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 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 7

Gunakan editor teks untuk mengedit file `/etc/nginx/nginx.conf`. Ini memerlukan izin root Linux. Di bagian atas file, tambahkan baris berikut:

- Jika menggunakan Client SDK 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- Jika menggunakan Client SDK 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Kemudian tambahkan yang berikut 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 {
    }
}
```

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 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 16.04 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 n3fips_password;
```

Kemudian tambahkan yang berikut 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 {
        }
    }
```

Ubuntu 18.04 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 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 20.04 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 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

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

Simpan file tersebut.

8. Cadangkan file konfigurasi `systemd`, dan kemudian atur jalur `EnvironmentFile`.

   Amazon Linux

   Tidak ada tindakan diperlukan.

   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
      ```

   CentOS 7

   Tidak ada tindakan diperlukan.

   CentOS 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 7

Tidak ada tindakan diperlukan.

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
```

Ubuntu 16.04

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 18.04

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 20.04 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 22.04 LTS

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

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. Konfigurasikan lingkungan NGINX.

> **Note**
>
> SDK Klien 5 memperkenalkan `CLOUDHSM_PIN` variabel lingkungan untuk menyimpan kredensial CU.

Amazon Linux

Buka file `/etc/sysconfig/nginx` di editor teks. Ini memerlukan izin root Linux. Tambahkan kredensi Cryptography User (CU):

- Jika menggunakan Client SDK 3

```
n3fips_password=<CU user name>:<password>
```

- Jika menggunakan Client SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU.

Simpan file tersebut.

Amazon Linux 2

Buka file `/etc/sysconfig/nginx` di editor teks. Ini memerlukan izin root Linux. Tambahkan kredensi Cryptography User (CU):

- Jika menggunakan Client SDK 3

```
n3fips_password=<CU user name>:<password>
```

- Jika menggunakan Client SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU.

Simpan file tersebut.

CentOS 7

Buka file `/etc/sysconfig/nginx` di editor teks. Ini memerlukan izin root Linux. Tambahkan kredensi Cryptography User (CU):

- Jika menggunakan Client SDK 3

```
n3fips_password=<CU user name>:<password>
```

- Jika menggunakan Client SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU.

Simpan file tersebut.

CentOS 8

Buka file `/etc/sysconfig/nginx` di editor teks. Ini memerlukan izin root Linux. Tambahkan kredensi Cryptography User (CU):

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU.

Simpan file tersebut.

Red Hat 7

Buka file `/etc/sysconfig/nginx` di editor teks. Ini memerlukan izin root Linux. Tambahkan kredensi Cryptography User (CU):

- Jika menggunakan Client SDK 3

```
n3fips_password=<CU user name>:<password>
```

- Jika menggunakan Client SDK 5

```
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 Cryptography User (CU):

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU.

Simpan file tersebut.

Ubuntu 16.04 LTS

Buka file `/etc/sysconfig/nginx` di editor teks. Ini memerlukan izin root Linux. Tambahkan kredensi Cryptography User (CU):

```
n3fips_password=<CU user name>:<password>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU.

Simpan file tersebut.

Ubuntu 18.04 LTS

Buka file `/etc/sysconfig/nginx` di editor teks. Ini memerlukan izin root Linux. Tambahkan kredensi Cryptography User (CU):

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU.

Simpan file tersebut.

Ubuntu 20.04 LTS

Buka file `/etc/sysconfig/nginx` di editor teks. Ini memerlukan izin root Linux. Tambahkan kredensi Cryptography User (CU):

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU.

Simpan file tersebut.

Ubuntu 22.04 LTS

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

11. Mulai server web NGINX.

Amazon Linux

Buka file `/etc/sysconfig/nginx` di editor teks. Ini memerlukan izin root Linux. Tambahkan kredensi Cryptography User (CU):

```
$ sudo service nginx start
```

Amazon Linux 2

Hentikan proses NGINX yang berjalan

```
$ sudo systemctl stop nginx
```

Muat ulang `systemd` konfigurasi untuk mengambil perubahan terbaru

```
$ sudo systemctl daemon-reload
```

Mulai proses NGINX

```
$ sudo systemctl start nginx
```

CentOS 7

Hentikan proses NGINX yang berjalan

```
$ sudo systemctl stop nginx
```

Muat ulang `systemd` konfigurasi untuk mengambil perubahan terbaru

```
$ sudo systemctl daemon-reload
```

Mulai proses NGINX

```
$ sudo systemctl start nginx
```

CentOS 8

Hentikan proses NGINX yang 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 7

Hentikan proses NGINX yang 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 8

Hentikan proses NGINX yang 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 16.04 LTS

Hentikan proses NGINX yang 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 18.04 LTS

Hentikan proses NGINX yang 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 20.04 LTS

Hentikan proses NGINX yang 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

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

12. (Opsional) Konfigurasikan platform Anda untuk memulai NGINX saat mulai.

Amazon Linux

```
$ sudo chkconfig nginx on
```

Amazon Linux 2

```
$ sudo systemctl enable nginx
```

CentOS 7

Tidak ada tindakan diperlukan.

CentOS 8

```
$ sudo systemctl enable nginx
```

Red Hat 7

Tidak ada tindakan diperlukan.

Red Hat 8

```
$ sudo systemctl enable nginx
```

Ubuntu 16.04 LTS

```
$ sudo systemctl enable nginx
```

Ubuntu 18.04 LTS

```
$ sudo systemctl enable nginx
```

Ubuntu 20.04 LTS

```
$ sudo systemctl enable nginx
```

Ubuntu 22.04 LTS

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

Setelah memperbarui konfigurasi server web, buka Langkah 4: Aktifkan lalu lintas HTTPS dan verifikasi sertifikat.

Mengkonfigurasi 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

   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 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
```

CentOS 7

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
```

CentOS 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 7

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
```

Ubuntu 16.04 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 18.04 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 20.04 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 22.04 LTS

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

3. Salin sertifikat server web Anda ke lokasi yang diperlukan untuk platform Anda.

Amazon Linux

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Ganti *<web_server.crt>* dengan nama sertifikat server web 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.

CentOS 7

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Ganti *<web_server.crt>* dengan nama sertifikat server web Anda.

CentOS 8

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Ganti *<web_server.crt>* dengan nama sertifikat server web Anda.

Red Hat 7

```
$ 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.

Ubuntu 16.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

Ganti *<web_server.crt>* dengan nama sertifikat server web Anda.

Ubuntu 18.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

Ganti *<web_server.crt>* dengan nama sertifikat server web Anda.

Ubuntu 20.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

Ganti *<web_server.crt>* dengan nama sertifikat server web Anda.

Ubuntu 22.04 LTS

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

4. Salin kunci privat PEM palsu Anda ke lokasi yang diperlukan untuk platform Anda.

Amazon Linux

```
$ 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 privat PEM Anda yang palsu.

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 privat PEM Anda yang palsu.

CentOS 7

```
$ 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 privat PEM Anda yang palsu.

CentOS 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 privat PEM Anda yang palsu.

Red Hat 7

```
$ 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 privat PEM Anda yang palsu.

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 privat PEM Anda yang palsu.

Ubuntu 16.04 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 privat PEM Anda yang palsu.

Ubuntu 18.04 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 privat PEM Anda yang palsu.

Ubuntu 20.04 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 privat PEM Anda yang palsu.

Ubuntu 22.04 LTS

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

5. Ubah kepemilikan file-file ini jika diperlukan oleh platform Anda.

   Amazon Linux

   ```
   $ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
   localhost.key
   ```

   Memberikan izin baca kepada pengguna bernama apache.

   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.

CentOS 7

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

Memberikan izin baca kepada pengguna bernama apache.

CentOS 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 7

```
$ 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.

Ubuntu 16.04 LTS

Tidak ada tindakan diperlukan.

Ubuntu 18.04 LTS

Tidak ada tindakan diperlukan.

Ubuntu 20.04 LTS

Tidak ada tindakan diperlukan.

Ubuntu 22.04 LTS

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

6. Atur konfigurasi arahan Apache untuk platform Anda.

Amazon Linux

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:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

Simpan file tersebut.

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:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

Simpan file tersebut.

CentOS 7

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:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

Simpan file tersebut.

CentOS 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 7

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:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-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.

Ubuntu 16.04 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 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:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

Simpan file tersebut.

Mengaktifkan modul SSL dan konfigurasi situs SSL default:

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 18.04 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 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: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 20.04 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 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: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 22.04 LTS

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

7. Atur konfigurasi file nilai lingkungan untuk platform Anda.

Amazon Linux

Tidak ada tindakan diperlukan. Nilai lingkungan masuk `/etc/sysconfig/httpd`

Amazon Linux 2

Buka file layanan httpd:

```
/lib/systemd/system/httpd.service
```

Di bawah bagian `[Service]`, tambahkan berikut ini:

```
EnvironmentFile=/etc/sysconfig/httpd
```

CentOS 7

Buka file layanan httpd:

```
/lib/systemd/system/httpd.service
```

Di bawah bagian `[Service]`, tambahkan berikut ini:

```
EnvironmentFile=/etc/sysconfig/httpd
```

CentOS 8

Buka file layanan httpd:

```
/lib/systemd/system/httpd.service
```

Di bawah bagian `[Service]`, tambahkan berikut ini:

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 7

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
```

Ubuntu 16.04 LTS

Tidak ada tindakan diperlukan. Nilai lingkungan masuk `/etc/sysconfig/httpd`

Ubuntu 18.04 LTS

Tidak ada tindakan diperlukan. Nilai lingkungan masuk `/etc/sysconfig/httpd`

Ubuntu 20.04 LTS

Tidak ada tindakan diperlukan. Nilai lingkungan masuk `/etc/sysconfig/httpd`

Ubuntu 22.04 LTS

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

8. Dalam file yang menyimpan variabel lingkungan untuk platform Anda, tetapkan variabel lingkungan yang berisi kredensial pengguna kriptografi (CU):

Amazon Linux

Gunakan editor teks untuk mengedit `/etc/sysconfig/httpd`.

- Jika menggunakan Client SDK 3

```
n3fips_password=<CU user name>:<password>
```

- Jika menggunakan Client SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU.

Amazon Linux 2

Gunakan editor teks untuk mengedit `/etc/sysconfig/httpd`.

- Jika menggunakan Client SDK 3

```
n3fips_password=<CU user name>:<password>
```

- Jika menggunakan Client SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU.

CentOS 7

Gunakan editor teks untuk mengedit `/etc/sysconfig/httpd`.

- Jika menggunakan Client SDK 3

```
n3fips_password=<CU user name>:<password>
```

- Jika menggunakan Client SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU.

CentOS 8

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 7

Gunakan editor teks untuk mengedit `/etc/sysconfig/httpd`.

- Jika menggunakan Client SDK 3

```
n3fips_password=<CU user name>:<password>
```

- Jika menggunakan Client SDK 5

```
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`.

```
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.

Ubuntu 16.04 LTS

Gunakan editor teks untuk mengedit `/etc/apache2/envvars`.

```
export n3fips_password=<CU user name>:<password>
```

Ganti *<CU user name>* dan *<password>* dengan kredensial CU.

Ubuntu 18.04 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 20.04 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 22.04 LTS

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

9. Mulai server web Apache.

Amazon Linux

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

Amazon Linux 2

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

CentOS 7

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

CentOS 8

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

Red Hat 7

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

Red Hat 8

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

Ubuntu 16.04 LTS

```
$ sudo service apache2 start
```

Ubuntu 18.04 LTS

```
$ sudo service apache2 start
```

Ubuntu 20.04 LTS

```
$ sudo service apache2 start
```

Ubuntu 22.04 LTS

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

10. (Opsional) Atur konfigurasi platform Anda untuk memulai Apache saat mulai.

Amazon Linux

```
$ sudo chkconfig httpd on
```

Amazon Linux 2

```
$ sudo chkconfig httpd on
```

CentOS 7

```
$ sudo chkconfig httpd on
```

CentOS 8

```
$ systemctl enable httpd
```

Red Hat 7

```
$ sudo chkconfig httpd on
```

Red Hat 8

```
$ systemctl enable httpd
```

Ubuntu 16.04 LTS

```
$ sudo systemctl enable apache2
```

Ubuntu 18.04 LTS

```
$ sudo systemctl enable apache2
```

Ubuntu 20.04 LTS

```
$ sudo systemctl enable apache2
```

Ubuntu 22.04 LTS

Dukungan untuk OpenSSL Dynamic Engine belum tersedia.

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 Anda untuk pembongkaran SSL/TLS dengan AWS CloudHSM, tambahkan instans server web Anda ke grup keamanan yang mengizinkan lalu lintas HTTPS masuk. Hal ini memungkinkan klien, seperti peramban web, untuk membuat koneksi HTTPS dengan server web Anda. Kemudian, buat sambungan HTTPS ke server web Anda dan verifikasi bahwa itu menggunakan sertifikat yang Anda konfigurasikan untuk pembongkaran SSL/TLS dengan 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 menetapkannya 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 konfigurasikan

Setelah menambahkan server web ke grup keamanan, Anda dapat memverifikasi bahwa pembongkaran SSL/TLS menggunakan sertifikat yang ditandatangani sendiri. Anda dapat melakukan ini dengan peramban web atau dengan alat seperti OpenSSL s_client.

Untuk memverifikasi pembongkaran SSL/TLS dengan peramban 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 EC2dalam 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 sertifikat SSL/TLS adalah salah satu yang Anda konfigurasi server web Anda untuk digunakan.

Untuk memverifikasi pembongkaran SSL/TLS dengan OpenSSL s_client

1. Jalankan perintah OpenSSL berikut untuk terhubung ke server web Anda menggunakan HTTPS. Ganti *<nama server>* 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 EC2dalam Panduan Developer Amazon Route 53 atau dalam dokumentasi untuk layanan DNS Anda.

2. Pastikan bahwa sertifikat SSL/TLS adalah salah satu yang Anda konfigurasi server web Anda untuk digunakan.

Anda sekarang memiliki situs web yang diamankan dengan HTTPS. Kunci privat untuk server web disimpan dalam HSM di klaster AWS CloudHSM Anda.

Untuk menambahkan penyeimbang beban, lihatTambahkan load balancer dengan Elastic Load Balancing (opsional).

Menggunakan Tomcat dengan JSSE untuk pembongkaran SSL/TLS di Linux

Topik ini memberikan step-by-step petunjuk untuk menyiapkan pembongkaran SSL/TLS menggunakan Java Secure Socket Extension (JSSE) denganAWS CloudHSM JCE SDK.

Topik

- Gambaran Umum
- Langkah 1: Siapkan prasyarat
- Langkah 2: Menghasilkan atau Mengimpor kunci Privat dan Sertifikat SSL/TLS
- Langkah 3: Mengkonfigurasi server web Tomcat

- Langkah 4: Aktifkan lalu lintas HTTPS dan verifikasi sertifikat

Gambaran Umum

DiAWS CloudHSM, server web Tomcat bekerja di Linux untuk mendukung HTTPS. AWS CloudHSMJCE SDK menyediakan antarmuka yang dapat digunakan dengan JSSE (Java Secure Socket Extension) untuk mengaktifkan penggunaan HSM untuk server web tersebut. AWS CloudHSM JCE adalah jembatan yang menghubungkan JSSE ke klaster AWS CloudHSM Anda. JSSE adalah Java API untuk Lapisan Soket Aman (SSL) dan Keamanan Lapisan Pengangkutan (TLS).

Langkah 1: Siapkan prasyarat

Ikuti prasyarat ini untuk menggunakan server web Tomcat dengan pembongkaran SSL/TLS di LinuxAWS CloudHSM. Prasyarat ini harus dipenuhi untuk mengatur server web SSL/TLS offload 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 kripto (CU) harus memiliki dan mengelola kunci privat server web pada HSM.
- AWS CloudHSMCluster aktif dengan setidaknya dua modul keamanan perangkat keras (HSM) 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 konfigurasikan JCE untuk AWS CloudHSM pada AWS CloudHSM cluster aktif dengan setidaknya dua modul keamanan perangkat keras (HSM). 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 kripto (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. Menggunakan Client SDK 5 untuk berintegrasi dengan Java Keytool dan Jarsigner

Setelah Anda menyelesaikan langkah ini, buka Langkah 2: Menghasilkan atau Mengimpor kunci Privat dan Sertifikat SSL/TLS.

Catatan

- Untuk menggunakan Security-Enhanced Linux (SELinux) dan server web, Anda harus mengizinkan koneksi TCP keluar pada port 2223, yaitu port yang Klien SDK 5 gunakan untuk berkomunikasi dengan HSM.
- Untuk membuat dan mengaktifkan klaster dan memberikan akses instans EC2 ke klaster, 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 klaster Anda. Untuk informasi lebih lanjut, lihat Menambahkan HSM.
- 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: Menghasilkan atau Mengimpor kunci Privat dan Sertifikat SSL/TLS

Untuk mengaktifkan HTTPS, aplikasi server web Anda memerlukan kunci privat dan sertifikat SSL/TLS yang sesuai. Untuk menggunakan pembongkaran SSL/TLS server web dengan AWS CloudHSM, Anda harus menyimpan kunci privat di HSM klaster AWS CloudHSM Anda.

> **Note**
>
> Jika Anda belum memiliki kunci privat dan sertifikat yang sesuai, buat kunci privat. Anda menggunakan kunci privat untuk membuat permintaan penandatanganan sertifikat (CSR), yang Anda gunakan untuk membuat sertifikat SSL/TLS.

Anda membuatAWS CloudHSM KeyStore file lokal yang berisi referensi ke kunci pribadi Anda pada HSM dan sertifikat terkait. Server web Anda menggunakanAWS CloudHSM KeyStore file untuk mengidentifikasi kunci pribadi pada HSM selama offload SSL/TLS.

Topik

- Menghasilkan kunci Privat at at at
- Menghasilkan sebuah sertifikat yang ditandatangani sendiri

Menghasilkan kunci Privat at at at

Bagian ini menunjukkan cara untuk menghasilkan keypair menggunakan KeyTool dari JDK. Setelah Anda memiliki key pair yang dihasilkan di dalam HSM, Anda dapat mengekspornya sebagai KeyStore file, dan menghasilkan sertifikat yang sesuai.

Tergantung pada kasus penggunaan Anda, Anda dapat menghasilkan RSA atau key pair EC. Langkah-langkah berikut menunjukkan bagaimana untuk menghasilkan key pair kunci kunci RSA.

Gunakan**`genkeypair`** perintah KeyTool untuk menghasilkan key pair kunci kunci RSA

1. Setelah mengganti data di <VARIABLES>bawah ini dengan data spesifik Anda, gunakan perintah berikut untuk membuat file keystore bernama`jsse_keystore.keystore`, yang akan memiliki referensi kunci pribadi Anda pada 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 tersebut.
- <KEYSTORE PASSWORD>: Ini adalah kata sandi untuk file keystore lokal Anda.
- <JSSE KEYSTORE NAME>: Nama file Keystore.
- <CERT DOMAIN NAME>: X.500 Nama dibedakan.
- <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, SHA1withRSA, SHA224withRSA, SHA256withRSA, SHA384withRSA, dan SHA512withRSA).

2. Untuk mengonfirmasi perintah berhasil, masukkan perintah berikut dan verifikasi bahwa Anda telah berhasil menghasilkan key pair RSA.

```
$ ls <PATH>/<JSSE KEYSTORE NAME>.keystore
```

Menghasilkan sebuah sertifikat yang ditandatangani sendiri

Setelah Anda menghasilkan kunci privat, 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 ke sana dan gunakan sertifikat SSL/TLS yang ditandatangani yang mereka berikan 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 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 output dari permintaan penandatanganan sertifikat adalah`certreq.csr`.

Tanda tangani sertifikat

- Setelah mengganti <VARIABLES>data di bawah ini, jalankan perintah berikut untuk menandatangani CSR Anda dengan kunci privat Anda pada 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.

Mengimpor sertifikat di Keystore

- Setelah mengganti data di <VARIABLES>bawah ini dengan data spesifik Anda, jalankan perintah berikut untuk mengimpor sertifikat yang ditandatangani sebagai sertifikat tepercaya. Langkah ini akan menyimpan sertifikat dalam 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 mengkonversi file sertifikat 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 ini, buka Langkah 3: Konfigurasi server web.

Langkah 3: Mengkonfigurasi 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 pembongkaran SSL/TLS dengan AWS CloudHSM. Untuk informasi lebih lanjut, lihat Apache Tomcat 9 Configuration Reference.

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 dari Tomcat

1. Hubungkan ke instans klien Anda.
2. Temukan folder instalasi Tomcat.
3. Setelah mengganti di <VARIABLES>bawah ini dengan data spesifik Anda, gunakan perintah berikut untuk menambahkan perpustakaan Java dan Cloudhsm Java jalan di Tomcat classpath, terletak di file Tomcat/bin/catalina.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.

1. Pergi ke folder instalasi Tomcat.
2. Setelah mengganti data 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 tersebut.
- <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 Anda untuk pembongkaran SSL/TLS dengan AWS CloudHSM, tambahkan instans server web Anda ke grup keamanan yang mengizinkan lalu lintas HTTPS masuk. Hal ini memungkinkan klien, seperti peramban web, untuk membuat koneksi HTTPS dengan server web Anda. Kemudian, buat sambungan HTTPS ke server web Anda dan verifikasi bahwa itu menggunakan sertifikat yang Anda konfigurasikan untuk pembongkaran SSL/TLS dengan 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 menetapkannya 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 konfigurasikan

Setelah menambahkan server web ke grup keamanan, Anda dapat memverifikasi bahwa pembongkaran SSL/TLS menggunakan sertifikat yang ditandatangani sendiri. Anda dapat melakukan ini dengan peramban web atau dengan alat seperti OpenSSL s_client.

Untuk memverifikasi pembongkaran SSL/TLS dengan peramban 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 EC2dalam 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 sertifikat SSL/TLS adalah salah satu yang Anda konfigurasi server web Anda untuk digunakan.

Untuk memverifikasi pembongkaran SSL/TLS dengan OpenSSL s_client

1. Jalankan perintah OpenSSL berikut untuk terhubung ke server web Anda menggunakan HTTPS. Ganti *<nama server>* 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 EC2dalam Panduan Developer Amazon Route 53 atau dalam dokumentasi untuk layanan DNS Anda.

2. Pastikan bahwa sertifikat SSL/TLS adalah salah satu yang Anda konfigurasi server web Anda untuk digunakan.

Anda sekarang memiliki situs web yang diamankan dengan HTTPS. Kunci privat untuk server web disimpan dalam HSM di klaster AWS CloudHSM Anda.

Untuk menambahkan penyeimbang beban, lihatTambahkan load balancer dengan Elastic Load Balancing (opsional).

Menggunakan pembongkaran SSL/TLS di Windows

Tutorial ini memberikan step-by-step instruksi untuk menyiapkan pembongkaran SSL/TLS denganAWS CloudHSM pada server web Windows.

Topik

- Gambaran Umum
- Langkah 1: Mengatur prasyarat
- Langkah 2: Membuat permintaan penandatanganan sertifikat (CSR) dan sertifikat
- Langkah 3: Konfigurasi server web
- Langkah 4: Aktifkan lalu lintas HTTPS dan verifikasi sertifikat

Gambaran Umum

Pada Windows, aplikasi server web Layanan Informasi Internet (IIS) untuk Windows Server secara native mendukung HTTPS. Penyedia penyimpanan kunci (KSP) AWS CloudHSM untuk Kriptografi API Microsoft: Next Generation (CNG) menyediakan antarmuka yang memungkinkan IIS untuk

menggunakan HSM di klaster Anda untuk pembongkaran kriptografi dan penyimpanan kunci. KSP AWS CloudHSM adalah jembatan yang menghubungkan IIS ke klaster 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 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: Mengatur prasyarat.

Langkah 1: Mengatur prasyarat

Untuk menyiapkan pebongkaran SSL/TLS server web dengan AWS CloudHSM, Anda memerlukan yang berikut ini:

- Klaster AWS CloudHSM aktif dengan setidaknya satu HSM.
- Instans Amazon EC2 menjalankan sistem operasi Windows dengan perangkat lunak berikut diinstal:
  - Perangkat lunak klien AWS CloudHSM untuk Windows.
  - Layanan Informasi Internet (IIS) untuk Windows Server.
- Pengguna kripto (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. Saat Anda meluncurkan klien Amazon EC2, pilih AMI Windows Server 2016 atau Windows Server 2012. 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 klien AWS CloudHSM untuk Windows diinstal.

2. (Opsional) Tambahkan lebih banyak HSM ke klaster Anda. Untuk informasi lebih lanjut, lihat Menambahkan HSM.
3. Hubungkan ke server Windows Anda. Untuk informasi selengkapnya, lihat Hubungkan ke Instans Anda dalam Panduan Pengguna Amazon EC2 untuk Instans Windows.
4. Gunakan CloudHSM CLI untuk membuat pengguna kripto (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 kredensi HSM, download *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 menginstal IIS pada Windows Server Anda

1. Jika Anda belum melakukannya, hubungkan ke server Windows server Anda. Untuk informasi selengkapnya, lihat Hubungkan ke Instans Anda dalam Panduan Pengguna Amazon EC2 untuk Instans Windows.
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: Membuat permintaan penandatanganan sertifikat (CSR) dan sertifikat.

Langkah 2: Membuat permintaan penandatanganan sertifikat (CSR) dan sertifikat

Untuk mengaktifkan HTTPS, server web Anda memerlukan sertifikat SSL/TLS dan kunci privat yang sesuai. Untuk menggunakan pembongkaran SSL/TLS dengan AWS CloudHSM, Anda menyimpan kunci privat di HSM di klaster 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

- Buat CSR
- Dapatkan sertifikat yang ditandatangani dan impor

Buat CSR

Gunakan AWS CloudHSM KSP pada Windows Server untuk membuat CSR.

Untuk membuat CSR

1. Jika Anda belum melakukannya, hubungkan ke server Windows server Anda. Untuk informasi selengkapnya, lihat Hubungkan ke Instans Anda dalam Panduan Pengguna Amazon EC2 untuk Instans Windows.
2. Gunakan perintah berikut untuk memulai daemonAWS 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 Perintah certreq Windows untuk membuat CSR dari file `IISCertRequest.inf` 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 *`CertRequestIIS.inf`* dengan nama file yang sesuai. Anda dapat mengganti *`CertRequestIIS.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 sertifikat SSL/TLS 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 mengganti *`SelfSignedCA.key`* dengan nama file berisi kunci privat 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 privat Anda (jika berbeda). Anda dapat mengganti *SelfSignedCA.crt* dengan nama file berisi sertifikat yang Anda tanda tangani 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).
  - *CertRequestIIS.csr* — Nama file yang berisi CSR server web Anda
  - *SelfSignedCA.crt* — Nama file yang berisi sertifikat yang Anda tanda tangani sendiri
  - *SelfSignedCA.key* — Nama file yang berisi kunci privat Anda
  - *IISCert.crt* — Nama file yang berisi sertifikat 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) untuk pembongkaran SSL/TLS dengan 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 Hubungkan ke Instans Anda dalam Panduan Pengguna Amazon EC2 untuk Instans Windows.
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 Sertifikat - Komputer Lokal, 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 tandatangani sendiri. Jika Anda membuat sertifikat yang ditandatangani sendiri dengan mengikuti petunjuk di langkah sebelumnya dari tutorial ini, sertifikat yang Anda tandatangani 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 Selanjutnua 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 Hubungkan ke Instans Anda dalam Panduan Pengguna Amazon EC2 untuk Instans Windows.
2. Mulai daemon klien AWS CloudHSM.
3. Salin sertifikat yang ditandatangani oleh server web Anda—sertifikat yang Anda buat di akhir langkah sebelumnya tutorial ini —ke server Windows Anda.
4. Pada Windows Server, gunakan perintah certreq Windows untuk menerima sertifikat yang ditandatangani, seperti dalam 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 Anda untuk pembongkaran SSL/TLS dengan AWS CloudHSM, tambahkan instans server web Anda ke grup keamanan yang mengizinkan lalu lintas HTTPS masuk. Hal ini memungkinkan klien, seperti peramban web, untuk membuat koneksi HTTPS dengan server web Anda. Kemudian, buat sambungan HTTPS ke server web Anda dan verifikasi bahwa itu menggunakan sertifikat yang Anda konfigurasikan untuk pembongkaran SSL/TLS dengan AWS CloudHSM.

Topik

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 menetapkannya 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 konfigurasikan

Setelah menambahkan server web ke grup keamanan, Anda dapat memverifikasi bahwa pembongkaran SSL/TLS menggunakan sertifikat yang ditandatangani sendiri. Anda dapat melakukan ini dengan peramban web atau dengan alat seperti OpenSSL s_client.

Untuk memverifikasi pembongkaran SSL/TLS dengan peramban 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 EC2dalam 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 sertifikat SSL/TLS adalah salah satu yang Anda konfigurasi server web Anda untuk digunakan.

Untuk memverifikasi pembongkaran SSL/TLS dengan OpenSSL s_client

1. Jalankan perintah OpenSSL berikut untuk terhubung ke server web Anda menggunakan HTTPS. Ganti *<nama server>* 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 EC2dalam Panduan Developer Amazon Route 53 atau dalam dokumentasi untuk layanan DNS Anda.

2. Pastikan bahwa sertifikat SSL/TLS adalah salah satu yang Anda konfigurasi server web Anda untuk digunakan.

Anda sekarang memiliki situs web yang diamankan dengan HTTPS. Kunci privat untuk server web disimpan dalam HSM di klaster AWS CloudHSM Anda.

Untuk menambahkan penyeimbang beban, lihatTambahkan load balancer dengan Elastic Load Balancing (opsional).

Tambahkan load balancer dengan Elastic Load Balancing (opsional)

Setelah Anda mengatur pembongkaran SSL/TLS offload satu server web, Anda dapat membuat lebih banyak server web dan penyeimbang beban Elastic Load Balancing yang merutekan 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

- Membuat subnet untuk server web kedua
- Membuat server web kedua

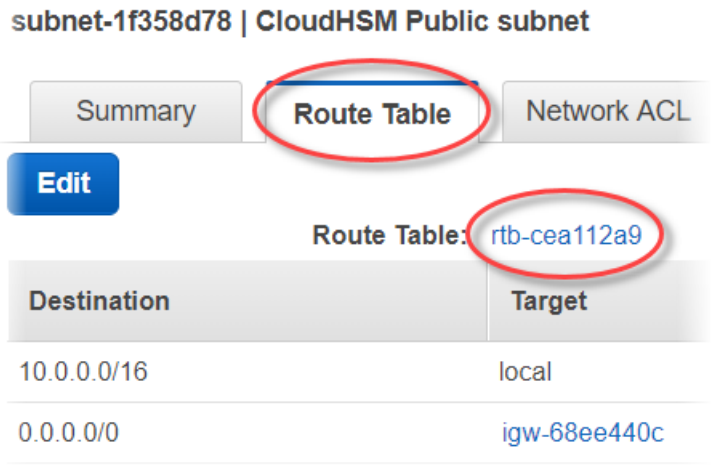
- Membuat penyeimbang beban

Membuat subnet untuk server web kedua

Sebelum Anda dapat membuat server web lain, Anda perlu membuat subnet baru di VPC yang sama yang berisi server web yang ada dan klaster AWS CloudHSM.

Untuk membuat subnet baru

1. Buka bagian Subnet dari konsol Amazon VPC.
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 yang ada dan klaster AWS CloudHSM.
   c. Untuk Availability Zone, pilih Availability Zone yang berbeda dari availability zone yang berisi server web Anda yang sudah ada.
   d. Untuk Blok CIDR IPv4, 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.



6. Centang kotak di samping tabel rute.

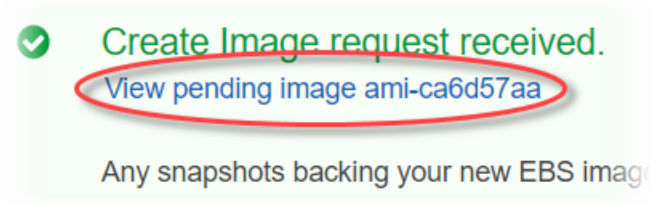
7. Pilih tab Asosiasi Subnet. Lalu, pilih Edit.
8. Centang kotak di samping subnet publik yang telah Anda buat sebelumnya dalam prosedur ini. Lalu, pilih Simpan.

Membuat server web kedua

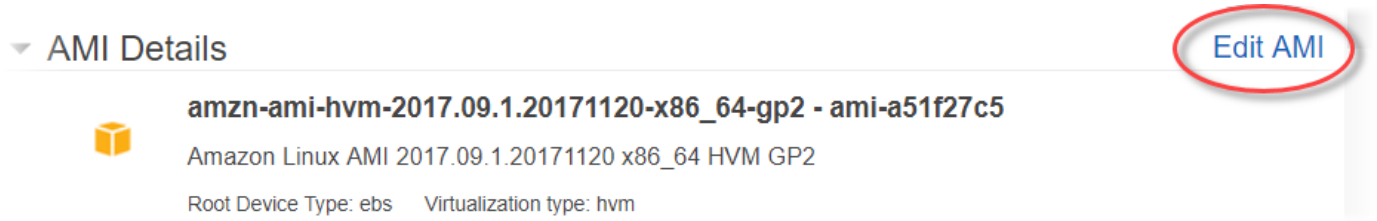
Selesaikan langkah-langkah berikut untuk membuat server web kedua dengan konfigurasi yang sama seperti server web Anda yang ada.

Untuk membuat server web kedua

1. Buka bagian Instans 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 tautan Lihat gambar ami-***<ID AMI> 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.



9. Di panel navigasi kiri, pilih AMI saya. Lalu, hapus teks di kotak pencarian.
10. Di samping gambar server web Anda, pilih Pilih.
11. Pilih Ya, saya ingin melanjutkan dengan AMI ini (***<nama gambar>***- ami-***<ID AMI>***).
12. Pilih Selanjutnya.
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-***<ID klaster>***-sg. AWS CloudHSM membuat grup keamanan ini atas nama Anda saat Anda membuat klaster. Anda harus memilih grup keamanan ini untuk mengizinkan instans server web untuk menyambung ke HSM di klaster.
    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 menetapkannya 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 Pilih pasangan kunci yang sudah ada.
        2. Untuk Pilih pasangan kunci, pilih pasangan kunci yang akan digunakan.
        3. Centang kotak di samping Saya menyatakan bahwa saya memiliki akses ke file kunci privat yang dipilih (***<nama file kunci privat>***.pem), dan bahwa tanpa file ini, saya tidak akan dapat masuk ke instans 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. PilihUnduh 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.

Membuat 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 balancers pada 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, pilihTutup.

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 penyeimbang beban mengirim lalu lintas

HTTPS ke server web Anda dengan pembongkaran SSL/TLS 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 EC2dalam 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 EC2dalam 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 privat server web disimpan dalam HSM di klaster AWS CloudHSM. Situs web Anda memiliki dua server web dan penyeimbang beban untuk membantu meningkatkan efisiensi dan ketersediaan.

Konfigurasikan Windows Server sebagai otoritas sertifikat (CA) dengan AWS CloudHSM

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 mengatur konfigurasi 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. Bila Anda mengatur konfigurasi peran ini, Anda menggunakan penyedia penyimpanan kunci (KSP) AWS CloudHSM untuk membuat dan menyimpan kunci privat CA di klaster AWS CloudHSM. KSP adalah jembatan yang menghubungkan server Windows Anda ke klaster AWS CloudHSM. Pada langkah terakhir, Anda menandatangani permintaan penandatanganan sertifikat (CSR) dengan Windows Server CA.

Untuk informasi selengkapnya, lihat topik berikut:

Topik

Windows Server CA langkah 1: Mengatur prasyarat

Untuk mengatur Windows Server sebagai otoritas sertifikat (CA) dengan AWS CloudHSM, Anda memerlukan yang berikut ini:

- Klaster AWS CloudHSM aktif dengan setidaknya satu HSM.
- Instans Amazon EC2 yang menjalankan sistem operasi Windows Server dengan perangkat lunak klien AWS CloudHSM untuk Windows diinstal. Tutorial ini menggunakan Microsoft Windows Server 2016.
- Pengguna kriptografi (CU) harus memiliki dan mengelola kunci privat CA pada HSM.

Untuk menyiapkan prasyarat untuk menggunakan Windows Server CA dengan AWS CloudHSM

1. Selesaikan langkah-langkah dalam Mulai. 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 klien AWS CloudHSM untuk Windows diinstal.
2. (Opsional) Tambahkan lebih banyak HSM ke klaster Anda. Untuk informasi lebih lanjut, lihat Menambahkan HSM.
3. Hubungkan ke instans klien Anda. Untuk informasi selengkapnya, lihat Hubungkan ke Instans Anda dalam Panduan Pengguna Amazon EC2 untuk Instans Windows.
4. Buat pengguna kripto (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, download *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, pergi ke Buat Windows Server CA.

Windows Server CA langkah 2: Membuat Windows Server CA dengan AWS CloudHSM

Untuk membuat Windows Server CA, Anda menambahkan peran Active Directory Certificate Services (AD CS) ke Windows Server. Bila Anda menambahkan peran ini, Anda menggunakan penyedia penyimpanan kunci (KSP) AWS CloudHSM untuk membuat dan menyimpan kunci privat CA di klaster AWS CloudHSM.

> **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 Hubungkan ke Instans Anda dalam Panduan Pengguna Amazon EC2 untuk Instans Windows.
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 Kredenitas, verifikasi atau ubah kredensial yang ditampilkan. Lalu, pilih Selanjutnya.
13. Untuk Peran Layanan, pilih Otoritas Sertifikasi. Lalu, pilih Selanjutnya.
14. UntukJenis Pengaturan, pilih CA Mandiri. Lalu, pilih Selanjutnya.
15. UntukJenis 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 pilihKonfigurasi.
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.

Windows Server CA langkah 3: Menandatangani permintaan penandatanganan sertifikat (CSR) dengan Windows Server CA denganAWS CloudHSM

Anda dapat menggunakan Windows Server CA dengan 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 menandatanganinya 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 Hubungkan ke Instans Anda dalam Panduan Pengguna Amazon EC2 untuk Instans Windows.
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. Plih tab Detail, dan kemudian pilih Salin ke File.

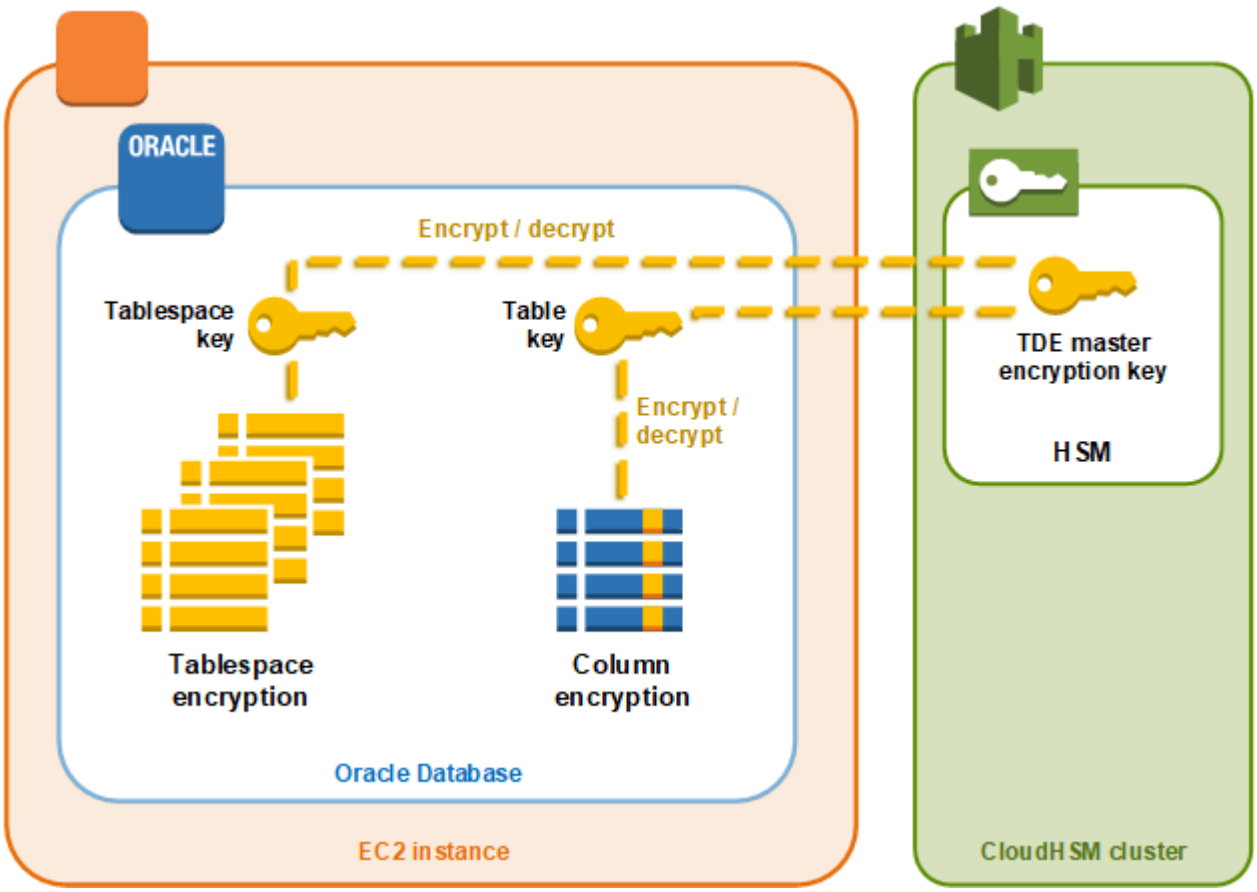
    c. Ikuti instruksi di Wizard Ekspor Sertifikat.

Anda sekarang memiliki Windows Server CA dengan AWS CloudHSM, dan sertifikat valid yang 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 di HSM di cluster AndaAWS CloudHSM.



Dalam solusi ini, Anda menggunakan Oracle Database yang diinstal pada instans Amazon EC2. Oracle Database terintegrasi dengan pustaka perangkat lunak AWS CloudHSM untuk PKCS #11untuk menyimpan kunci utama TDE di HSM di klaster Anda.

> **Important**
>
> - Sebaiknya instal Oracle Database pada instans Amazon EC2.

Selesaikan langkah-langkah berikut untuk menyelesaikan integrasi Oracle TDE dengan AWS CloudHSM.

Untuk mengatur konfigurasi integrasi Oracle TDE dengan AWS CloudHSM

1. Ikuti langkah-langkah di Menyiapkan prasyarat untuk mempersiapkan lingkungan Anda.

2. Ikuti langkah-langkah di Konfigurasi Basis Data Data untuk mengatur konfigurasi Oracle Database dengan klaster AWS CloudHSM Anda.

Oracle TDE denganAWS CloudHSM: Mengatur prasyarat

Untuk mencapai integrasi Oracle TDE dengan AWS CloudHSM, Anda memerlukan yang berikut ini:

- Klaster AWS CloudHSM aktif dengan setidaknya satu HSM.
- Instans Amazon EC2 menjalankan sistem operasi Amazon Linux dengan perangkat lunak berikut diinstal:
  - Klien AWS CloudHSM dan alat baris perintah.
  - Pustaka perangkat lunak AWS CloudHSM untuk PKCS #11.
  - Oracle Database. AWS CloudHSMmendukung integrasi Oracle TDE. SDK Klien 5.6 dan dukungan Oracle TDE 19c Oracle Database 19c. Klien SDK 3 mendukung Oracle TDE untuk Oracle Database versi 11g dan 12c.
- Pengguna kriptografi (CU) harus memiliki dan mengelola kunci enkripsi master TDE pada HSM di klaster 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. Setelah Anda melakukannya, Anda akan memiliki klaster aktif dengan satu HSM. Anda juga akan memiliki instans Amazon EC2 menjalankan sistem operasi Amazon Linux. Klien AWS CloudHSM dan alat baris perintah juga akan diinstal dan dikonfigurasi.
2. (Opsional) Tambahkan lebih banyak HSM ke klaster Anda. Untuk informasi lebih lanjut, lihat Menambahkan HSM.
3. Hubungkan ke instans klien Amazon EC2 Anda dan lakukan hal berikut:
   a. Instal pustaka perangkat lunak AWS CloudHSM untuk PKCS #11.
   b. Instal Oracle Database. Untuk informasi selengkapnya, lihat dokumentasi Oracle Database. SDK Klien 5.6 dan dukungan Oracle TDE 19c Oracle Database 19c. Klien 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 klaster Anda. Untuk informasi selengkapnya tentang cara membuat CU, lihat Cara Mengelola Pengguna HSM dengan CMU dan Mengelola pengguna HSM.

Setelah menyelesaikan langkah ini, Anda dapat Konfigurasi Basis Data Data.

Oracle TDE denganAWS CloudHSM: Konfigurasi Basis Data Data dan buat kunci enkripsi master

Untuk integrasi Oracle TDE dengan klaster AWS CloudHSM Anda, lihat topik berikut:

1. Memperbarui Konfigurasi Oracle Database untuk menggunakan HSM di klaster Anda sebagai modul keamanan eksternal. Untuk informasi tentang modul keamanan eksternal, lihat Pengenalan Enkripsi Data Transparan di Panduan Keamanan Tingkat Lanjut Oracle Database.
2. Hasilkan kunci enkripsi kunci Oracle TDE pada HSM di klaster Anda.

Memperbarui Konfigurasi Oracle Database

Untuk memperbarui konfigurasi Oracle Database untuk menggunakan HSM di klaster 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.

4. Jalankan perintah berikut untuk membuat direktori tempat Oracle Database mengharapkan untuk menemukan file pustaka untuk pustaka perangkat lunak PKCS #11 AWS CloudHSM.

```
sudo mkdir -p /opt/oracle/extapi/64/hsm
```

5. Jalankan perintah berikut untuk menyalin file pustaka perangkat lunak AWS CloudHSM untuk PKCS #11 ke direktori yang Anda buat di 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.

Hasilkan kunci enkripsi kunci Oracle TDE

Untuk menghasilkan kunci utama Oracle TDE pada HSM di klaster Anda, selesaikan langkah-langkah dalam prosedur berikut.

Untuk menghasilkan kunci utama

1. Gunakan perintah berikut untuk menjalankan 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

menghasilkan kunci utama. Variabel ini hanya diperlukan untuk pembuatan kunci utama. Untuk informasi lebih lanjut, lihat “Masalah: Oracle menetapkan atribut PCKS #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 *<nama pengguna CU>* dengan nama pengguna dari pengguna kriptografi (CU). Ganti *<kata sandi>* 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 *<nama pengguna CU>* dengan nama pengguna kriptografi (CU). Ganti *<kata sandi>* 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 dompert 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 dompert secara manual, jalankan perintah berikut.

```
SQL> administer key management set keystore close identified by "<CU user
 name>:<password>";
```

Menggunakan Microsoft SignTool bersamaAWS CloudHSMuntuk 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 privat dimiliki oleh pengirim, tanda tangan dapat dipercaya. AWS CloudHSM menyediakan perangkat keras divalidasi FIPS 140-2 tingkat 3 yang aman bagi Anda untuk mengamankan kunci ini dengan akses penghuni tunggal eksklusif.

Banyak organisasi menggunakan Microsoft SignTool, alat baris perintah yang menandatangani, memverifikasi, dan memberi stempel waktu file untuk menyederhanakan proses penandatanganan kode. Anda dapat menggunakanAWS CloudHSMuntuk menyimpan pasangan kunci Anda dengan

aman sampai dibutuhkan oleh SignTool, sehingga menciptakan alur kerja yang mudah otomatis untuk penandatanganan data.

Topik berikut memberikan ikhtisar tentang cara menggunakan SignTool bersamaAWS CloudHSM:

Topik

Microsoft SignTool denganAWS CloudHSMLangkah 1: Menyiapkan 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.
- Klaster AWS CloudHSM aktif di virtual public cloud (VPC) yang sama dengan instans EC2 Anda. Klaster harus berisi setidaknya satu HSM.
- Pengguna kripto (CU) harus memiliki dan mengelola kunci di klaster AWS CloudHSM.
- File tanpa tanda tangan atau dapat dijalankan.
- Kit Pengembangan Perangkat Lunak (SDK) Microsoft Windows.

Untuk menyiapkan prasyarat untuk menggunakan AWS CloudHSM dengan Windows 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 CA Windows Server Anda sendiri, ikuti langkah 1 dan 2 di Mengatur Konfigurasi Windows Server Sebagai Otoritas Sertifikat dengan AWS CloudHSM. Jika tidak, terus gunakan CA pihak ketiga yang dipercaya secara publik.
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, dan CA Anda untuk Membuat Sertifikat Penandatanganan.

Microsoft SignTool denganAWS CloudHSM 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 Hubungkan ke Instans Anda dalam Panduan Pengguna Amazon EC2 untuk Instans Windows.
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, pasangan kunci baru dibuat di klaster AWS CloudHSM, dan kunci privat 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.

Microsoft SignTool dengan AWS CloudHSM langkah 3: Menandatangani file

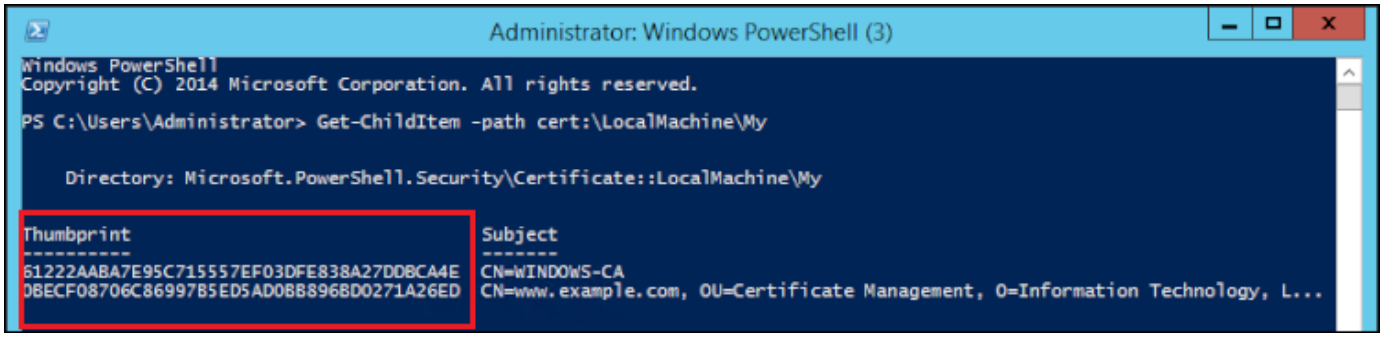
Anda sekarang siap untuk menggunakan SignTool dan sertifikat impor Anda untuk menandatangani file contoh Anda. Untuk melakukannya, Anda perlu mengetahui SHA-1 hash atau cap jempol. Sidik jari digunakan untuk memastikan bahwa SignTool hanya menggunakan sertifikat yang diverifikasi oleh. AWS CloudHSM Dalam 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:

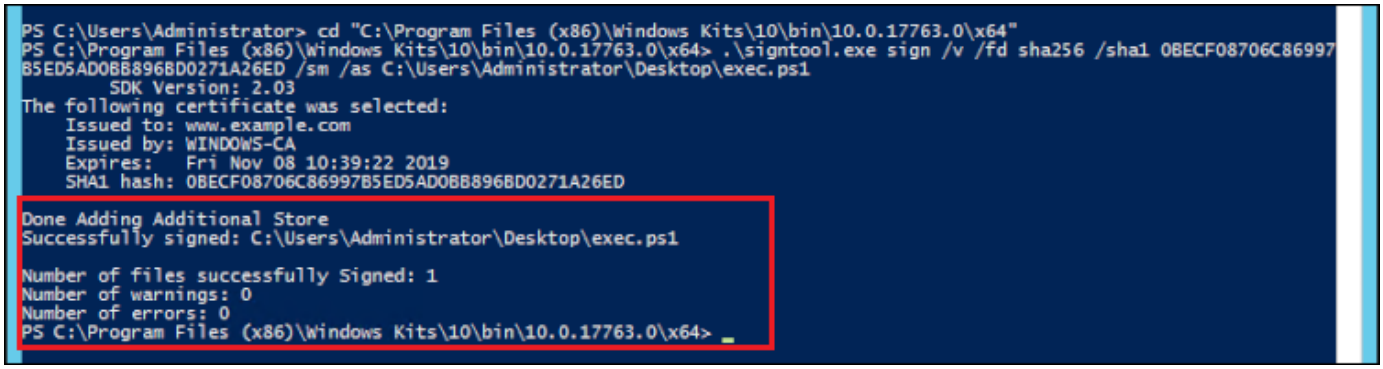
   ```
   Get-ChildItem -path cert:\LocalMachine\My
   ```

   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 Jarsigner

AWS CloudHSMmenawarkan integrasi dengan utilitas Java dan Jarsigner melalui SDK Klien 3 dan SDK Klien 5. Langkah-langkah untuk menggunakan alat ini akan bervariasi tergantung pada versi SDK klien yang saat ini telah Anda unduh:

- Menggunakan Client SDK 5 untuk berintegrasi dengan Java Keytool dan Jarsigner
- Menggunakan Client SDK 3 untuk berintegrasi dengan Java Keytool dan Jarsigner

Menggunakan Client SDK 5 untuk berintegrasi dengan Java Keytool dan Jarsigner

Penyimpanan kunci AWS CloudHSM adalah penyimpanan kunci JCE tujuan khusus yang menggunakan sertifikat yang terkait dengan kunci pada HSM Anda melalui alat pihak ketiga seperti `keytool` dan `jarsigner`. AWS CloudHSM tidak menyimpan sertifikat di HSM, karena sertifikat merupakan data publik, non-rahasia. Penyimpanan kunci AWS CloudHSM menyimpan sertifikat dalam file lokal dan memetakan sertifikat ke kunci yang sesuai pada HSM Anda.

Saat Anda menggunakan penyimpanan kunci AWS CloudHSM untuk menghasilkan kunci baru, tidak ada entri yang dihasilkan dalam file penyimpanan kunci lokal - kunci dibuat pada HSM. Demikian pula, ketika Anda menggunakan penyimpanan kunci AWS CloudHSM untuk mencari kunci, pencarian diteruskan ke HSM. Ketika Anda menyimpan sertifikat di penyimpanan kunci AWS CloudHSM,

penyedia memverifikasi bahwa pasangan kunci dengan alias yang sesuai ada di HSM, dan kemudian mengaitkan sertifikat yang disediakan dengan pasangan kunci yang sesuai.

Topik

Prasyarat

Untuk menggunakan penyimpanan kunci AWS CloudHSM, Anda harus terlebih dahulu menginisialisasi dan mengatur konfigurasi AWS CloudHSM SDK JCE.

Langkah 1: Instal JCE

Untuk menginstal JCE, termasuk prasyarat klien AWS CloudHSM, ikuti langkah-langkah untuk menginstal pustaka 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 CloudHSMJCE menawarkan berbagai opsi login. Untuk menggunakan penyimpanan kunci AWS CloudHSM 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 penyimpanan kunci AWS CloudHSM. Untuk informasi tambahan, lihat artikel tentang Menggunakan penyimpanan kunci AWS CloudHSM.

Langkah 3: Mendaftarkan penyedia JCE

Untuk mendaftarkan 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 menggunakan keytool dengan penyimpanan kunci AWS CloudHSM dapat menyebabkan kesalahan.

Menggunakan AWS CloudHSM key store dengan keytool

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 root kepercayaan melalui penyimpanan kunci AWS CloudHSM.

Saat menggunakan keytool dengan penyimpanan kunci AWS CloudHSM, tentukan argumen berikut untuk setiap perintah keytool:

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 penyimpanan kunci AWS CloudHSM, lihat MenggunakanAWS CloudHSMKeystore. 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, penyimpanan kunci AWS CloudHSM menciptakan file penyimpanan kunci baru.

Buat kunci baru dengan keytool

Anda dapat menggunakan keytool untuk menghasilkan jenis kunci RSA, AES, dan DeSede yang didukung oleh JCE SDK. AWS CloudHSM

> **Important**
>
> Kunci yang dihasilkan melalui keytool dihasilkan dalam perangkat lunak, dan kemudian diimpor ke AWS CloudHSM sebagai kunci yang dapat diekstrak dan persisten.

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 mengekspor 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 dalamthe section called “Menghasilkan 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, “my_keystore.store”, di direktori kerja. Ganti <secret label>dengan label unik.

Linux

```
$ keytool -genseckey -alias <secret label> -keyalg aes \
 -keysize 256 -keystore my_keystore.store \
 -storetype CloudHSM -J-classpath '-J/opt/cloudhsm/java/*' \
```

Windows

```
PS C:\> keytool -genseckey -alias <secret label> -keyalg aes `
 -keysize 256 -keystore my_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, “my_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 my_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 my_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 kunci menggunakan keytool

Penyimpanan kunci AWS CloudHSM tidak mendukung penghapusan kunci. Anda dapat menghapus kunci menggunakan metode penghancuran antarmuka Destroyable.

```
((Destroyable) key).destroy();
```

Menghasilkan CSR menggunakan keytool

Anda menerima fleksibilitas terbesar dalam membuat permintaan penandatanganan sertifikat (CSR) jika menggunakan OpenSSL Dynamic Engine. Perintah berikut menggunakan keytool untuk menghasilkan CSR untuk pasangan kunci dengan alias, `my-key-pair`.

Linux

```
$ keytool -certreq -alias <key pair label> \
 -file my_csr.csr \
 -keystore my_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -certreq -alias <key pair label> `
 -file my_csr.csr `
 -keystore my_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 “Menggunakan keytool untuk mengimpor sertifikat perantara dan root ke dalam penyimpanan AWS CloudHSM kunci ”

Menggunakan keytool untuk mengimpor sertifikat perantara dan root ke dalam penyimpanan AWS CloudHSM kunci

Untuk mengimpor sertifikat CA, Anda harus mengaktifkan verifikasi rantai sertifikat penuh pada sertifikat yang baru diimpor. Perintah berikut menunjukkan sebuah contoh.

Linux

```
$ keytool -import -trustcacerts -alias rootCAcert \
 -file rootCAcert.cert -keystore my_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -import -trustcacerts -alias rootCAcert `
 -file rootCAcert.cert -keystore my_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Jika Anda menghubungkan beberapa instans klien ke klaster AWS CloudHSM, mengimpor sertifikat pada satu penyimpanan kunci instans klien tidak akan secara otomatis membuat sertifikat tersebut tersedia pada instans klien lainnya. Anda harus mengimpor sertifikat pada setiap instans klien.

Menggunakan keytool untuk menghapus sertifikat dari toko AWS CloudHSM kunci

Perintah berikut menunjukkan contoh bagaimana untuk menghapus sertifikat dari penyimpanan kunci keytool Java.

Linux

```
$ keytool -delete -alias mydomain \
 -keystore my_keystore.store \
```

```
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -delete -alias mydomain `
 -keystore my_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Jika Anda menghubungkan beberapa instans klien ke klaster AWS CloudHSM, menghapus sertifikat di satu penyimpanan kunci satu instans klien tidak akan secara otomatis menghapus sertifikat tersebut dari instans klien lainnya. Anda harus menghapus sertifikat pada setiap instans klien.

Mengimpor 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 my_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 my_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.

Mengekspor sertifikat menggunakan keytool

Contoh berikut menghasilkan sertifikat dalam format X.509 biner. Untuk mengekspor sertifikat yang dapat dibaca manusia, tambahkan `-rfc` ke perintah `-exportcert`.

Linux

```
$ keytool -exportcert -alias <key pair label> \
 -file my_exported_certificate.crt \
 -keystore my_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 my_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Menggunakan toko AWS CloudHSM kunci dengan Jarsigner

Jarsigner adalah utilitas baris perintah populer untuk menandatangani file JAR menggunakan kunci yang disimpan dengan aman di HSM. Tutorial lengkap tentang Jarsigner ada di luar lingkup dokumentasi AWS CloudHSM. Bagian ini menjelaskan parameter Jarsigner yang harus Anda gunakan untuk menandatangani dan memverifikasi tanda tangan dengan AWS CloudHSM sebagai root kepercayaan melalui penyimpanan kunci AWS CloudHSM.

Menyiapkan kunci dan sertifikat

Sebelum Anda dapat menandatangani file JAR dengan Jarsigner, pastikan Anda telah mengatur atau menyelesaikan langkah-langkah berikut:

1. Ikuti panduan di prasyarat toko AWS CloudHSM utama.

2. Atur kunci penandatanganan Anda dan sertifikat terkait dan rantai sertifikat yang harus disimpan dalam penyimpanan kunci AWS CloudHSM dari server atau instans klien saat ini. Buat kunci pada AWS CloudHSM lalu impor metadata terkait ke penyimpanan kunci AWS CloudHSM. Jika Anda ingin menggunakan keytool untuk mengatur kunci dan sertifikat, lihat the section called “Buat kunci baru dengan keytool”. Jika Anda menggunakan beberapa instans klien untuk menandatangani JAR Anda, 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 untuk penyimpanan kunci AWS CloudHSM. Lihat cuplikan kode the section called “Tandatangani 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, lihatthe section called “Menggunakan keytool untuk mengimpor sertifikat perantara dan root ke dalam penyimpanan AWS CloudHSM kunci ”.

Tandatangani file JAR menggunakan AWS CloudHSM dan Jarsigner

Gunakan perintah berikut untuk menandatangani file JAR:

Linux;

Untuk OpenJDK 8

```
jarsigner -keystore my_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 my_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 OpenJDK8

```
jarsigner -keystore my_keystore.store `
 -signedjar signthisclass_signed.jar `
 -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.jar <key pair label>
```

Untuk OpenJDK 11, OpenJDK 17, dan OpenJDK 21

```
jarsigner -keystore my_keystore.store `
 -signedjar signthisclass_signed.jar `
 -sigalg 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 OpenJDK8

```
jarsigner -verify \
 -keystore my_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>
```

Untuk OpenJDK 11, OpenJDK 17, dan OpenJDK 21

```
jarsigner -verify \
 -keystore my_keystore.store \
 -sigalg 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 my_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 my_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

1. Kami tidak mendukung kunci EC dengan Keytool dan Jarsigner.

Menggunakan Client SDK 3 untuk berintegrasi dengan Java Keytool dan Jarsigner

Penyimpanan kunci AWS CloudHSM adalah penyimpanan kunci JCE tujuan khusus yang menggunakan sertifikat yang terkait dengan kunci pada HSM Anda melalui alat pihak ketiga seperti `keytool` dan `jarsigner`. AWS CloudHSM tidak menyimpan sertifikat di HSM, karena sertifikat merupakan data publik, non-rahasia. Penyimpanan kunci AWS CloudHSM menyimpan sertifikat dalam file lokal dan memetakan sertifikat ke kunci yang sesuai pada HSM Anda.

Saat Anda menggunakan penyimpanan kunci AWS CloudHSM untuk menghasilkan kunci baru, tidak ada entri yang dihasilkan dalam file penyimpanan kunci lokal - kunci dibuat pada HSM. Demikian pula, ketika Anda menggunakan penyimpanan kunci AWS CloudHSM untuk mencari kunci, pencarian diteruskan ke HSM. Ketika Anda menyimpan sertifikat di penyimpanan kunci AWS CloudHSM, penyedia memverifikasi bahwa pasangan kunci dengan alias yang sesuai ada di HSM, dan kemudian mengaitkan sertifikat yang disediakan dengan pasangan kunci yang sesuai.

Topik

Prasyarat

Untuk menggunakan penyimpanan kunci AWS CloudHSM, Anda harus terlebih dahulu menginisialisasi dan mengatur konfigurasi AWS CloudHSM SDK JCE.

Langkah 1: Instal JCE

Untuk menginstal JCE, termasuk prasyarat klien AWS CloudHSM, ikuti langkah-langkah untuk menginstal pustaka 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 kunci AWS CloudHSM 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 penyimpanan kunci AWS CloudHSM. Untuk informasi tambahan, lihat artikel tentang Menggunakan penyimpanan kunci AWS CloudHSM.

Langkah 3: Mendaftarkan 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 untuk menentukan opsi baris perintah ketika membuat kunci dengan penyimpanan kunci AWS CloudHSM, itu akan menyebabkan kesalahan.

Menggunakan AWS CloudHSM key store dengan keytool

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 root kepercayaan melalui penyimpanan kunci AWS CloudHSM.

Saat menggunakan keytool dengan penyimpanan kunci AWS CloudHSM, tentukan argumen berikut untuk setiap perintah keytool:

```
-storetype CLOUDHSM \
  -J-classpath '-J/opt/cloudhsm/java/*' \
  -J-Djava.library.path=/opt/cloudhsm/lib
```

Jika Anda ingin membuat file penyimpanan kunci baru menggunakan penyimpanan kunci AWS CloudHSM, lihat MenggunakanAWS CloudHSMKeystore. 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, penyimpanan kunci AWS CloudHSM menciptakan file penyimpanan kunci baru.

Buat 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 ke AWS CloudHSM sebagai kunci yang dapat diekstrak dan persisten.

Petunjuk untuk membuat kunci yang tidak dapat diekstrak secara langsung pada HSM, dan kemudian menggunakannya dengan keytool atau Jarsigner, ditampilkan dalam sampel kode di Mendaftarkan Kunci yang Sudah Ada dengan Penyimpanan Kunci 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 ekstrak melalui keytool dan jarsigner, penyedia mengekspor kunci dari AWS CloudHSM dan kemudian menggunakan kunci lokal untuk menandatangani operasi.

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, “my_keystore.store”, di direktori kerja. Ganti <secret label>dengan label unik.

```
keytool -genseckey -alias <secret label> -keyalg aes \
  -keysize 256 -keystore my_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, “my_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 my_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, “my_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 my_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 kunci menggunakan keytool

Penyimpanan kunci AWS CloudHSM tidak mendukung penghapusan kunci. Untuk menghapus kunci, Anda harus menggunakan fungsi `deleteKey` dari alat baris perintah AWS CloudHSM,deleteKey.

Menghasilkan CSR menggunakan keytool

Anda menerima fleksibilitas terbesar dalam membuat permintaan penandatanganan sertifikat (CSR) jika menggunakan OpenSSL Dynamic Engine. Perintah berikut menggunakan keytool untuk menghasilkan CSR untuk pasangan kunci dengan alias, `my-key-pair`.

```
keytool -certreq -alias <key pair label> \
        -file my_csr.csr \
        -keystore my_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 mengimpor data keystore lihat Mengimpor sertifikat Perantara dan root ke Penyimpanan Kunci AWS CloudHSM menggunakan Keytool.

Menggunakan keytool untuk mengimpor sertifikat perantara dan root ke dalam penyimpanan AWS CloudHSM kunci

Untuk mengimpor sertifikat CA, Anda harus mengaktifkan verifikasi rantai sertifikat penuh pada sertifikat yang baru diimpor. Perintah berikut menunjukkan sebuah contoh.

```
keytool -import -trustcacerts -alias rootCAcert \
        -file rootCAcert.cert -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

Jika Anda menghubungkan beberapa instans klien ke klaster AWS CloudHSM, mengimpor sertifikat pada satu penyimpanan kunci instans klien tidak akan secara otomatis membuat sertifikat tersebut tersedia pada instans klien lainnya. Anda harus mengimpor sertifikat pada setiap instans klien.

Menggunakan keytool untuk menghapus sertifikat dari toko AWS CloudHSM kunci

Perintah berikut menunjukkan contoh bagaimana untuk menghapus sertifikat dari penyimpanan kunci keytool Java.

```
keytool -delete -alias mydomain -keystore \
        -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

Jika Anda menghubungkan beberapa instans klien ke klaster AWS CloudHSM, menghapus sertifikat di satu penyimpanan kunci satu instans klien tidak akan secara otomatis menghapus sertifikat tersebut dari instans klien lainnya. Anda harus menghapus sertifikat pada setiap instans klien.

Mengimpor 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 my_certificate.crt \
        -keystore my_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 dengan Penyimpanan Kunci 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.

Mengekspor sertifikat menggunakan keytool

Contoh berikut menghasilkan sertifikat dalam format X.509 biner. Untuk mengekspor sertifikat yang dapat dibaca manusia, tambahkan `-rfc` ke perintah `-exportcert`.

```
keytool -exportcert -alias <key pair label> \
        -file my_exported_certificate.crt \
        -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

Menggunakan toko AWS CloudHSM kunci dengan jarsigner

Jarsigner adalah utilitas baris perintah populer untuk menandatangani file JAR menggunakan kunci yang disimpan dengan aman di HSM. Tutorial lengkap tentang Jarsigner ada di luar lingkup dokumentasi AWS CloudHSM. Bagian ini menjelaskan parameter Jarsigner yang harus Anda gunakan untuk menandatangani dan memverifikasi tanda tangan dengan AWS CloudHSM sebagai root kepercayaan melalui penyimpanan kunci AWS CloudHSM.

Menyiapkan kunci dan sertifikat

Sebelum Anda dapat menandatangani file JAR dengan Jarsigner, pastikan Anda telah mengatur atau menyelesaikan langkah-langkah berikut:

1. Ikuti petunjuk diPrasyarat penyimpanan kunci AWS CloudHSM.
2. Atur kunci penandatanganan Anda dan sertifikat terkait dan rantai sertifikat yang harus disimpan dalam penyimpanan kunci AWS CloudHSM dari server atau instans klien saat ini. Buat kunci pada AWS CloudHSM lalu impor metadata terkait ke penyimpanan kunci AWS CloudHSM. Gunakan contoh kode di Mendaftarkan Kunci yang Sudah Ada dengan Penyimpanan Kunci AWS CloudHSM untuk mengimpor metadata ke penyimpanan kunci. Jika Anda ingin menggunakan keytool untuk mengatur kunci dan sertifikat, lihat Buat kunci baru dengan keytool. Jika Anda menggunakan beberapa instans klien untuk menandatangani JAR Anda, 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 untuk penyimpanan kunci

AWS CloudHSM. Lihat cuplikan kode di Menandatangani file JAR menggunakan AWS CloudHSM dan Jarsigner untuk instruksi menggunakan kode Java untuk memverifikasi rantai sertifikat. Jika mau, Anda dapat menggunakan keytool untuk mengimpor sertifikat. Untuk petunjuk tentang cara menggunakan keytool, lihat Menggunakan Keytool untuk mengimpor sertifikat perantara dan root ke penyimpanan kunci AWS CloudHSM.

Menandatangani file JAR menggunakan AWS CloudHSM dan jarsigner

Gunakan perintah berikut untuk menandatangani file JAR:

```
jarsigner -keystore my_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 my_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

Daftar berikut menyediakan daftar masalah yang diketahui saat ini.

- 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 diberikan alias dan diimpor ke AWS CloudHSM HSM sebagai kunci persisten (token) selama proses penambahan kunci.

- Saat menggunakan keytool dengan penyimpanan kunci AWS CloudHSM, jangan tentukan opsi `-providerName`, `-providerclass`, atau `-providerpath` pada baris perintah. Tentukan opsi ini dalam file penyedia keamanan seperti yang dijelaskan dalam Prasyarat penyimpanan kunci.
- Bila menggunakan kunci EC yang tidak dapat diekstrak melalui keytool dan Jarsigner, penyedia SunEC perlu dihapus/dinonaktifkan dari daftar penyedia dalam file java.security. Jika Anda menggunakan kunci EC yang dapat diekstrak melalui keytool dan Jarsigner, penyedia mengekspor bit kunci dari HSM AWS CloudHSM dan menggunakan kunci lokal untuk menandatangani operasi. Kami tidak menyarankan Anda menggunakan kunci yang dapat diekspor dengan keytool atau Jarsigner.

Mendaftarkan kunci yang sudah ada sebelumnya dengan toko AWS CloudHSM kunci

Untuk keamanan maksimum dan fleksibilitas dalam atribut dan pelabelan, kami sarankan Anda membuat kunci tanda tangan menggunakan key_mgmt_util. Anda juga dapat menggunakan aplikasi Java untuk menghasilkan kunci dalam AWS CloudHSM.

Bagian berikut menyediakan contoh kode yang menunjukkan bagaimana cara menghasilkan pasangan kunci baru pada HSM dan mendaftarkannya menggunakan kunci yang ada yang diimpor ke penyimpanan kunci AWS CloudHSM. 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 di AWS CloudHSM dengan penyimpanan kunci lokal tidak mengekspor kunci. Ketika kunci terdaftar, penyimpanan kunci mendaftar 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.Enumeration;

//
// 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\t\tLabel to store the key and certificate under.%n" +
           "\t--list\t\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 alises 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());
        }
    }

}
```

Integrasi vendor pihak ketiga lainnya

Beberapa vendor pihak ketiga mendukung AWS CloudHSM sebagai root kepercayaan. Ini berarti bahwa Anda dapat memanfaatkan solusi perangkat lunak pilihan Anda saat membuat dan menyimpan kunci yang mendasari dalam klaster CloudHSM Anda. Akibatnya, beban kerja Anda di AWS dapat mengandalkan latensi, ketersediaan, keandalan, dan manfaat 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. Alat ini mendukung AWS Key Management Service dan AWS CloudHSM sebagai root kepercayaan untuk perlindungan tambahan.
- Thycotic Secrets Server membantu pelanggan mengelola kredensial sensitif di seluruh akun istimewa. Ini mendukung AWS CloudHSM sebagai root kepercayaan.

- Adaptador KMIP P6R mengizinkan Anda untuk memanfaatkan instans AWS CloudHSM melalui antarmuka KMIP standar.
- PrimeKey EJBCA adalah solusi sumber terbuka yang populer untuk PKI. Hal ini memungkinkan Anda untuk membuat dan menyimpan pasangan kunci dengan aman dengan AWS CloudHSM.
- Box KeySafe menyediakan manajemen kunci enkripsi untuk konten cloud ke banyak organisasi dengan persyaratan keamanan, privasi, dan kepatuhan terhadap peraturan yang ketat. Pelanggan dapat mengamankan KeySafe kunci lebih lanjut secara langsung diAWS Key Management Service atau secara tidak langsung diAWS CloudHSM melalui Penyimpanan KunciAWS KMS Kustom.
- Insyde Software mendukung AWS CloudHSM sebagai root kepercayaan untuk penandatanganan firmware.
- F5 BIG-IP LTM mendukung AWS CloudHSM sebagai root kepercayaan.
- Cloudera Navigator Key HSM memungkinkan Anda untuk menggunakan klaster 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 penandatanganan TLS, SSH, dan kode dengan pembuatan dan perlindungan kunci AWS CloudHSM.

AWS CloudHSM Pemantauan

Selain fitur pencatatan yang dibangun ke SDK Klien, Anda juga dapat menggunakanAWS CloudTrailAmazon CloudWatch Log, dan Amazon CloudWatch memantauAWS 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 diAWSakun, termasuk panggilan yang Anda buat untuk membuat dan menghapus klaster, modul keamanan perangkat keras (HSM), dan tag sumber daya.

CloudWatch Beberapa catatan

Gunakan CloudWatch Log untuk memantau log dari instans HSM, yang mencakup peristiwa untuk membuat dan menghapus pengguna HSM, mengubah kata sandi pengguna, membuat serta menghapus kunci, dan lainnya.

CloudWatch

Gunakan CloudWatch untuk memantau kondisi klaster dalam waktu nyata.

Topik

- Bekerja dengan log SDK klien
- Bekerja denganAWS CloudTraildanAWS CloudHSM
- Bekerja dengan CloudWatch Log Amazon dan Log AWS CloudHSM Audit
- Mendapatkan CloudWatch metrik untukAWS CloudHSM

Bekerja dengan log SDK klien

Anda dapat mengambil log yang dihasilkan oleh SDK Klien. AWS CloudHSM menawarkan implementasi pencatatan dengan SDK Klien 3 dan SDK Klien 5.

Topik

- SDK Klien 5
- SDK Klien 3

SDK Klien 5

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:

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
  ```

Untuk informasi tentang cara mengkonfigurasi logging untuk Client SDK 5, lihat Client SDK 5 Configure tool

SDK Klien 3

Log SDK Klien 3 berisi informasi detail dari daemon klien AWS CloudHSM. Lokasi log tergantung pada sistem operasi instans klien Amazon EC2 tempat Anda menjalankan daemon klien.

Amazon Linux

Di Amazon Linux, log klien AWS CloudHSM 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 Klien AWS CloudHSM dikumpulkan dan disimpan dalam jurnal. Anda dapat menggunakan journalctl untuk melihat dan mengelola log ini. Misalnya, gunakan perintah berikut untuk melihat log Klien AWS CloudHSM.

```
journalctl -f -u cloudhsm-client
```

CentOS 7

Di CentOS 7, log Klien AWS CloudHSM dikumpulkan dan disimpan dalam jurnal. Anda dapat menggunakan journalctl untuk melihat dan mengelola log ini. Misalnya, gunakan perintah berikut untuk melihat log Klien AWS CloudHSM.

```
journalctl -f -u cloudhsm-client
```

CentOS 8

Dalam CentOS 8, log Klien AWS CloudHSM dikumpulkan dan disimpan dalam jurnal. Anda dapat menggunakan journalctl untuk melihat dan mengelola log ini. Misalnya, gunakan perintah berikut untuk melihat log Klien AWS CloudHSM.

```
journalctl -f -u cloudhsm-client
```

RHEL 7

Di Red Hat Enterprise Linux 7, log Klien AWS CloudHSM dikumpulkan dan disimpan dalam jurnal. Anda dapat menggunakan journalctl untuk melihat dan mengelola log ini. Misalnya, gunakan perintah berikut untuk melihat log Klien AWS CloudHSM.

```
journalctl -f -u cloudhsm-client
```

RHEL 8

Di Red Hat Enterprise Linux 8, log Klien AWS CloudHSM dikumpulkan dan disimpan dalam jurnal. Anda dapat menggunakan journalctl untuk melihat dan mengelola log ini. Misalnya, gunakan perintah berikut untuk melihat log Klien AWS CloudHSM.

```
journalctl -f -u cloudhsm-client
```

Ubuntu 16.04

Di Ubuntu 16.04, log Klien AWS CloudHSM dikumpulkan dan disimpan dalam jurnal. Anda dapat menggunakan journalctl untuk melihat dan mengelola log ini. Misalnya, gunakan perintah berikut untuk melihat log Klien AWS CloudHSM.

```
journalctl -f -u cloudhsm-client
```

Ubuntu 18.04

Di Ubuntu 18.04, log Klien AWS CloudHSM dikumpulkan dan disimpan dalam jurnal. Anda dapat menggunakan journalctl untuk melihat dan mengelola log ini. Misalnya, gunakan perintah berikut untuk melihat log Klien AWS CloudHSM.

```
journalctl -f -u cloudhsm-client
```

Windows

- Untuk klien Windows 1.1.2+:

  Log Klien AWS CloudHSM ditulis ke file `cloudhsm.log` di folder file program AWS CloudHSM (`C:\Program Files\Amazon\CloudHSM\`). Setiap nama file log mempunyai akhiran dengan stempel waktu yang menunjukkan kapan klien AWS CloudHSM dimulai.
- Untuk klien Windows 1.1.1 dan yang lebih lama:

Log klien tidak ditulis ke file. Log ditampilkan pada prompt perintah atau di PowerShell jendela tempat Anda memulai AWS CloudHSM klien.

Bekerja denganAWS CloudTraildanAWS CloudHSM

AWS CloudHSM terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau layanan AWS di AWS CloudHSM. CloudTrail merekam semua panggilan API untuk AWS CloudHSM sebagai peristiwa. Panggilan yang direkam mencakup panggilan dari AWS CloudHSM konsol dan panggilan kode ke operasi API AWS CloudHSM ini. Jika membuat jejak, Anda dapat mengaktifkan pengiriman peristiwa CloudTrail berkelanjutan ke bucket Amazon S3, termasuk peristiwa untuk AWS CloudHSM. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru dalam konsol CloudTrail di Riwayat peristiwa. Menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke AWS CloudHSM, alamat IP asal permintaan tersebut dibuat, siapa yang membuat permintaan, kapan permintaan dibuat, dan detail lainnya.

Untuk mempelajari selengkapnya tentang CloudTrail, lihat Panduan Pengguna AWS CloudTrail. Untuk daftar lengkap operasi API AWS CloudHSM, lihat Tindakan dalam Referensi API AWS CloudHSM.

Informasi AWS CloudHSM di CloudTrail

CloudTrail diaktifkan pada akun AWS Anda saat Anda membuat akun tersebut. Ketika aktivitas terjadi di AWS CloudHSM, aktivitas tersebut dicatat dalam peristiwa CloudTrail bersama peristiwa layanan AWS lainnya di Riwayat peristiwa. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di akun AWS Anda. Untuk informasi selengkapnya, lihat Melihat Peristiwa dengan Riwayat Peristiwa CloudTrail.

Untuk catatan berkelanjutan tentang peristiwa di akun AWS Anda, termasuk peristiwa untuk AWS CloudHSM, buat jejak. Jejak memungkinkan CloudTrail untuk mengirim berkas log ke bucket Amazon S3. Secara default, ketika Anda membuat jejak di konsol tersebut, jejak tersebut diterapkan ke semua Wilayah AWS. Log acara jejak dari semua Wilayah di partisi AWS dan mengirimkan berkas log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi layanan AWS lainnya untuk menganalisis lebih lanjut dan bertindak berdasarkan data peristiwa yang dikumpulkan di log CloudTrail. Untuk informasi selengkapnya, lihat yang berikut:

- Ikhtisar untuk Membuat Jejak

- Layanan yang Didukung dan Integrasi CloudTrail
- Mengonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima Berkas Log CloudTrail dari Beberapa Wilayah dan Menerima Berkas Log CloudTrail dari Beberapa Akun

CloudTrail mencatat semua operasi AWS CloudHSM, termasuk operasi hanya-baca, seperti `DescribeClusters` dan `ListTags`, dan operasi manajemen, seperti `InitializeCluster`, `CreatHsm`, dan `DeleteBackup`.

Setiap peristiwa atau entri log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut:

- Bahwa permintaan dibuat dengan kredensial pengguna root atau pengguna AWS Identity and Access Management (IAM).
- Bahwa permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan.
- Apakah permintaan dibuat oleh layanan AWS lain.

Untuk informasi lebih lanjut, lihat Elemen userIdentity CloudTrail.

Memahami AWS CloudHSM entri file log

Jejak adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai berkas log ke bucket Amazon S3 yang telah Anda tentukan. File log CloudTrail berisi satu atau beberapa entri log. Peristuwa mewakili satu permintaan dari sumber apa pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. Berkas log CloudTrail bukanlah pelacakan tumpukan terurut dari panggilan API publik, sehingga tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri log CloudTrail yang menunjukkan tindakan AWS CloudHSM `CreateHsm`.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAJZVM5NEGZSTCITAMM:ExampleSession",
        "arn": "arn:aws:sts::111122223333:assumed-role/AdminRole/ExampleSession",
```

```
        "accountId": "111122223333",
        "accessKeyId": "ASIAIY22AX6VRYNBGJSA",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2017-07-11T03:48:44Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAJZVM5NEGZSTCITAMM",
                "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-6lz2hfmnzbx",
            "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 mengekspor data log ke Amazon S3. Anda dapat bekerja dengan log audit HSM di CloudWatch konsol Amazon atau menggunakan perintah CloudWatch Log di CLI CloudWatch dan Logs SDK.

Topik

Bagaimana HSM audit logging bekerja

Pencatatan audit diaktifkan secara otomatis di semua klaster AWS CloudHSM. Pencatatan tidak dapat dinonaktifkan atau dimatikan, dan tidak ada pengaturan yang dapat mencegah AWS CloudHSM dari mengekspor log ke CloudWatch Logs. 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 berbagai HSM, bahkan yang berada di klaster yang sama, cenderung berbeda. Misalnya, hanya HSM pertama di setiap klaster mencatat inisialisasi HSM. Peristiwa inisialisasi tidak muncul di log HSM yang merupakan kloning dari cadangan. Demikian pula, ketika Anda membuat kunci, HSM yang menghasilkan kunci mencatat peristiwa pembuatan kunci. HSM lain dalam klaster merekam peristiwa ketika mereka menerima kunci melalui sinkronisasi.

AWS CloudHSM mengumpulkan log dan mengirimnya ke CloudWatch Logs di akun Anda. Untuk berkomunikasi dengan layanan CloudWatch Logs atas nama Anda, AWS CloudHSM menggunakan peran yang terhubung dengan layanan. Kebijakan IAM yang terkait dengan peran memungkinkan AWS CloudHSM untuk hanya menjalankan tugas yang diperlukan untuk mengirim log audit ke CloudWatch Logs.

> **Important**
>
> Jika Anda membuat klaster sebelum 20 Januari 2018, dan belum membuat peran terkait layanan terlampir, Anda harus membuatnya secara manual. Hal ini diperlukan agar CloudWatch menerima log audit dari klaster AWS CloudHSM Anda. Untuk informasi lebih lanjut tentang pembuatan peran terkait layanan, lihat Memahami Peran Terkait Layanan, serta Membuat Peran Terkait Layanan dalamPanduan Pengguna IAM.

Melihat log audit HSM 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.

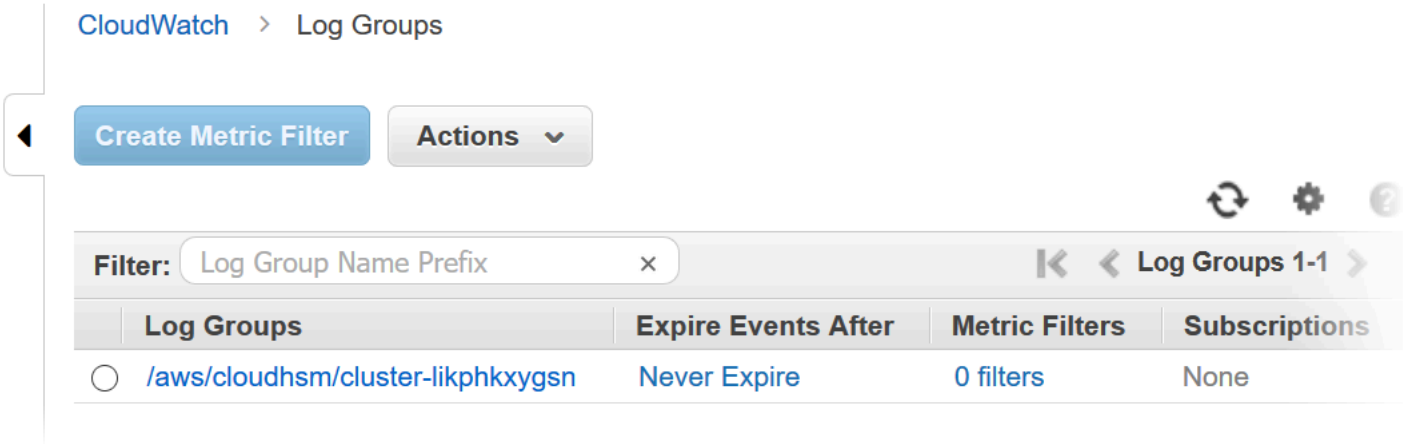
- Nama grup log adalah `/aws/cloudhsm/<cluster ID>`; misalnya `/aws/cloudhsm/cluster-likphkxygsn`. Saat Anda menggunakan nama grup log dalam CLI atau PowerShell perintah, pastikan untuk melampirkannya dalam tanda kutip ganda.
- Nama pengaliran log adalah ID HSM; sebagai contohnya, `hsm-nwbbiqbj4jk`.

  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.

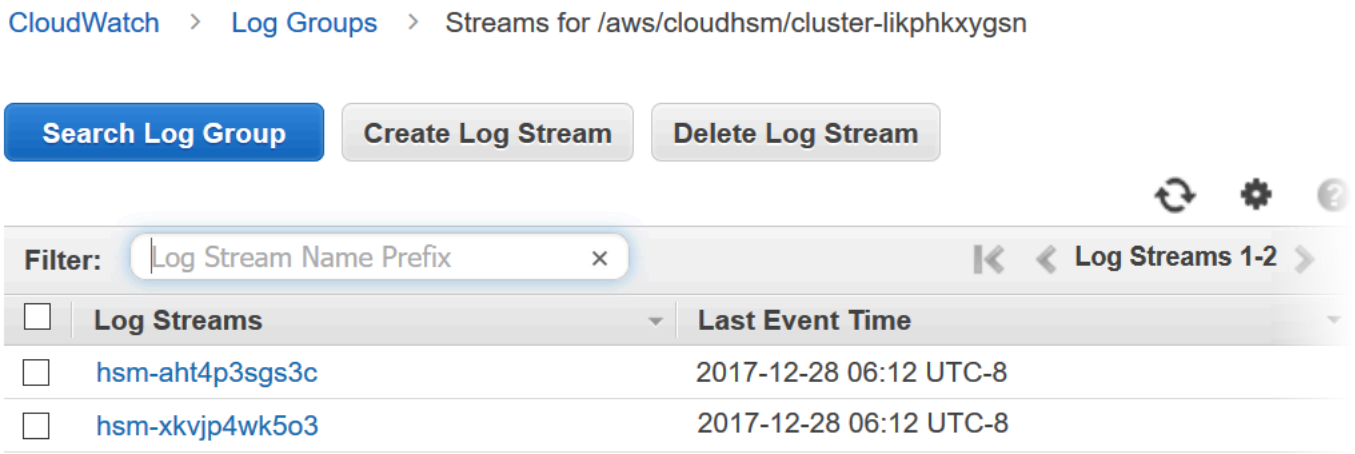
Untuk informasi selengkapnya tentang konsep CloudWatch Log, lihat Konsep di Panduan Pengguna CloudWatch Log Amazon.

Anda dapat melihat log audit untuk HSM dari halaman CloudWatch Log di, perintah CloudWatch Log di CLI AWS Management Console, cmdlet Log, CloudWatch atau PowerShellLog SDK. CloudWatch Untuk petunjuknya, lihat Melihat Data Log di Panduan Pengguna Amazon CloudWatch Logs.

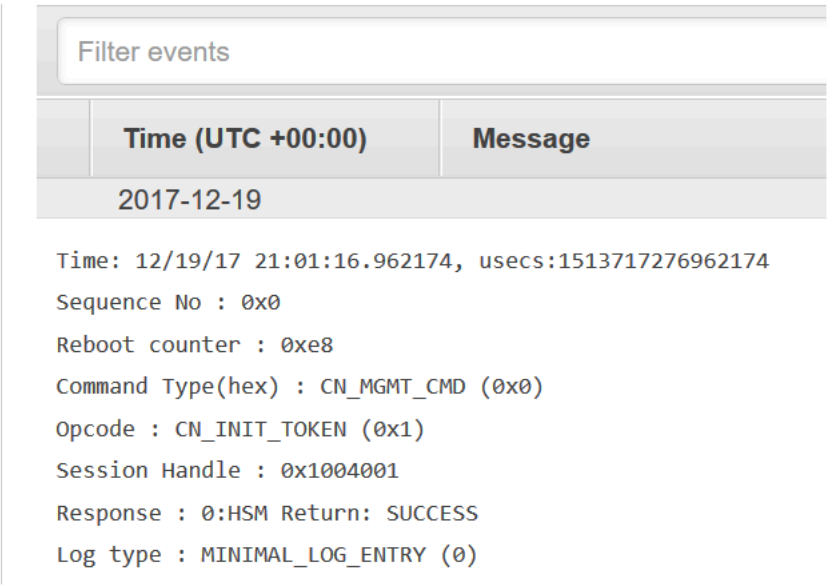
Sebagai contoh, gambar berikut menunjukkan grup log untuk klaster `cluster-likphkxygsn` di AWS Management Console.



Bila Anda memilih nama grup log klaster, Anda dapat melihat pengaliran log untuk masing-masing HSM di klaster. Gambar berikut menunjukkan pengaliran log untuk HSM di klaster `cluster-likphkxygsn`.

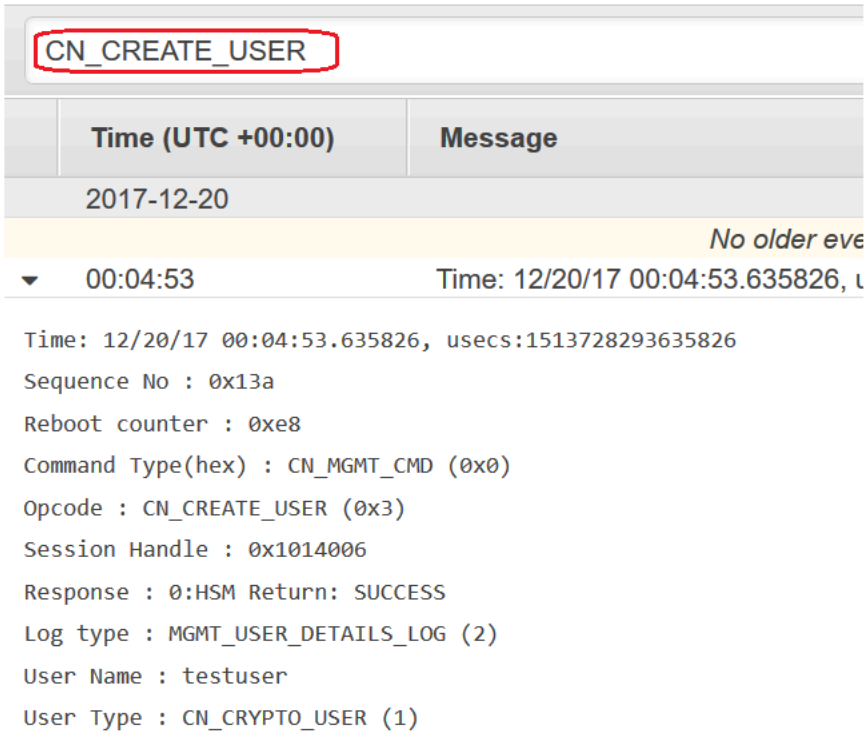


Bila Anda memilih nama pengaliran log HSM, Anda dapat melihat peristiwa di log audit. Sebagai contoh, peristiwa ini, yang memiliki nomor urut 0x0 dan `Opcode` dari `CN_INIT_TOKEN`, biasanya merupakan peristiwa pertama untuk HSM pertama di setiap klaster. Peristiwa ini mencatat inisialisasi HSM di klaster.



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` `0pcode`.

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.



Menafsirkan log audit HSM

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 adalah umum untuk setiap peristiwa AWS CloudHSM 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 nilai `Opcode` di log audit AWS CloudHSM, lihat Referensi log audit HSM.

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 dari log AWS CloudHSM yang mencatat 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 HSM Anda 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 di klaster

Pengaliran log audit untuk HSM pertama di setiap klaster berbeda secara signifikan dari pengaliran log HSM lain di klaster. Log audit untuk HSM pertama di setiap klaster mencatat pembuatan dan inisialisasinya. Log HSM tambahan di klaster, yang dihasilkan dari cadangan, dimulai dengan peristiwa login.

> **Important**
>
> Entri inisialisasi berikut tidak akan muncul di klaster yang diinisialisasi sebelum rilis fitur pencatatan audit CloudHSM (30 Agustus 2018). CloudWatch Untuk informasi lebih lanjut, lihat Riwayat Dokumentasi.

Contoh peristiwa berikut muncul dalam pengaliran log untuk HSM pertama dalam sebuah klaster. 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 klaster. Karena HSM berikutnya di klaster yang sama adalah klon dari yang pertama, HSM tersebut 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 sandipetugas precrypto (PRECO). Perintah ini mengaktifkan klaster.

```
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_CRYPTO_PRE_OFFICER (6)
```

Peristiwa Login

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 masuk ke HSM, HSM lain di klaster juga mencatat peristiwa login untuk pengguna. Anda dapat menemukan peristiwa login yang sesuai di pengaliran log HSM lainnya di klaster 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_CRYPTO_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_CRYPTO_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_CRYPTO_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_CRYPTO_USER (1)
```

Contoh: Buat pengguna

Contoh ini menunjukkan peristiwa log yang dicatat ketika seorang petugas kripto (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_CRYPTO_OFFICER (2)
```

Peristiwa berikutnya dalam pengaliran log (urutan `0x1`) mencatat CO yang membuat pengguna kripto 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_CRYPTO_USER (1)
```

Kemudian, CO membuat petugas kripto (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_CRYPTO_OFFICER (2)
```

Kemudian, CO bernama `admin` masuk dan menghapus petugas kripto (CO) bernama `alice`. HSM mencatat peristiwa `CN_DELETE_USER`. Nama dan jenis pengguna yang dihapis 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_CRYPTO_OFFICER (2)
```

Contoh: Buat key pair

Contoh ini menunjukkan peristiwa yang dicatat dalam log audit HSM ketika Anda membuat dan menghapus pasangan kunci.

Peristiwa berikut mencatat pengguna kripto (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_CRYPTO_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_CRYPTO_USER (1)
```

Contoh: Menghasilkan Kunci

Contoh ini menunjukkan efek pembuatan kunci dalam sebuah klaster dengan beberapa HSM. Kuncinya dihasilkan pada satu HSM, diekstrak dari HSM sebagai objek tertutup, dan dimasukkan ke dalam HSM lain sebagai objek tertutup.

> **Note**
>
> Alat klien mungkin gagal untuk menyinkronkan kunci. Atau, perintah mungkin termasuk parameter min_srv, yang menyinkronkan kunci hanya untuk jumlah HSM yang ditentukan. Dalam kedua kasus itu, layanan AWS CloudHSM menyinkronkan kunci dengan HSM lain dalam klaster. Karena HSM mencatat hanya perintah manajemen sisi klien dalam 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_CRYPTO_USER (1)
```

CU menjalankan exSymKeyperintah 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_CRYPTO_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 di kedua HSM di klaster.

```
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 dicatat ketika pengguna kripto (CU) mengeksspor kunci dari klaster dengan beberapa HSM.

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_CRYPTO_USER (1)
```

CU menjalankan exSymKeykunci7, kunci AES 256-bit. Perintah menggunakan kunci 6, kunci AES 256-bit HSM, 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 exSymKeymenulis kunci yang diekspor ke file, tetapi tidak mengubah kunci pada HSM. Akibatnya, tidak ada peristiwa yang sesuai dalam log HSM lain di klaster.

Contoh: Impor Kunci

Contoh ini menunjukkan peristiwa log audit yang direkam saat Anda mencatat kunci ke HSM di klaster. Dalam instance ini, pengguna kripto (CU) menggunakan imSymKeyperintah untuk mengimpor kunci AES ke HSM. 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 yang baru dihasilkan atau diimpor klien, alat klien secara otomatis mencoba untuk menyinkronkan kunci baru dengan HSM lainnya di klaster. 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 dari HSM lainnya di klaster 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: Berbagi Kunci

Contoh ini menunjukkan peristiwa log audit yang dicatat ketika pengguna kripto (CU) berbagi atau batal berbagi kunci privat ECC dengan pengguna kripto lainnya. CU menggunakan perintah shareKey dan menyediakan handel 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)
```

Referensi log audit HSM

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)`	Penggunaan Login
`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 kripto (CU)
`CN_CREATE_CO`	Buat petugas kripto (CO)
`CN_DELETE_USER`	Hapus pengguna
`CN_CHANGE_PSWD`	Ubah kata sandi pengguna
`CN_SET_M_VALUE`	Set otentikasi kuorum (M of N) for a user action
`CN_APPROVE_TOKEN`	Approve a otentikasi kuorum token for a user action
`CN_DELETE_TOKEN`	Delete one or more token kuorum
`CN_GET_TOKEN`	Request a signing token to initiate a operasi kuorum
Manajemen Kunci: Peristiwa ini termasuk pegangan kunci	
`CN_GENERATE_KEY`	Hasilkan sebuah kunci simetrik
`CN_GENERATE_KEY_PAIR (0x19)`	Generate an asymmetric key pair
`CN_CREATE_OBJECT`	Import a public key (without wrapping)
`CN_MODIFY_OBJECT`	Set a key attribute
`CN_DESTROY_OBJECT (0x11)`	Deletion of a kunci sesi

Kode Operasi (Opcode)	Deskripsi
`CN_TOMBSTONE_OBJECT`	Deletion of a kunci token
`CN_SHARE_OBJECT`	Berbagi atau membatalkan berbagi kunci
`CN_WRAP_KEY`	Export an encrypted copy of a key (wrapKey)
`CN_UNWRAP_KEY`	Import an encrypted copy of a key (unwrapKey)
`CN_DERIVE_KEY`	Derive a symmetric key from an existing key
`CN_NIST_AES_WRAP`	Enkripsi atau dekripsi kunci dengan kunci AES
`CN_INSERT_MASKED_OBJECT_USER`	Insert an encrypted key with attributes from another HSM in the cluster.
`CN_EXTRACT_MASKED_OBJECT_USER`	Wraps/encrypts a key with attributes from the HSM to be sent to another HSM in the cluster.
Back up HSMs	
`CN_BACKUP_START`	Begin the backup process
`CN_BACKUP_END`	Completed the backup process
`CN_RESTORE_START`	Begin restoring from a backup
`CN_RESTORE_END`	Completed the restoration process from a backup
Certificate-Based Authentication	
`CN_CERT_AUTH_STORE_CERT`	Stores the cluster certificate
HSM Instance Commands	
`CN_INIT_TOKEN (0x1)`	Start the HSM initialization process
`CN_INIT_DONE`	The HSM initialization process has finished
`CN_GEN_KEY_ENC_KEY`	Generate a key encryption key (KEK)

Kode Operasi (Opcode)	Deskripsi
`CN_GEN_PSWD_ENC_KEY (0x1d)`	Generate a password encryption key (PEK)
HSM crypto commands	
`CN_FIPS_RAND`	Generate a FIPS-compliant random number

Mendapatkan CloudWatch metrik untukAWS CloudHSM

Gunakan CloudWatch untuk memantauAWS CloudHSMklaster dalam waktu nyata. Metrik dapat dikelompokkan berdasarkan wilayah, ID klaster, atau ID klaster dan ID HSM.

ParameterAWS/CloudHSMnamespace mencakup metrik berikut:

Metrik	Deskripsi
HsmUnhealthy	Instans HSM tidak berkinerja dengan benar.AWS CloudHSMsecara 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	Suhu persimpangan prosesor perangkat keras. Sistem mati jika suhu mencapai 110 derajat Centigrade.
HsmKeysSessionOccupied	Jumlah kunci sesi yang digunakan oleh instance HSM.
HsmKeysTokenOccupied	Jumlah kunci token yang digunakan oleh instance HSM dan klaster.
HsmSslCtxsOccupied	Jumlah end-to-end Saluran terenkripsi yang saat ini dibuat untuk instans HSM. Hingga 2.048 saluran diperbolehkan.
HsmSessionCount	Jumlah koneksi terbuka ke instans HSM. Hingga 2.048 diperbolehkan. Secara default, daemon klien dikonfigurasi untuk membuka dua sesi dengan setiap instans HSM di bawah satu end-to-end Saluran terenkripsi.AWS CloudHSMm

Metrik	Deskripsi
	ungkin juga memiliki hingga 2 koneksi terbuka dengan HSM untuk memantau kesehatan HSM.
HsmUsersAvailable	Jumlah pengguna tambahan yang bisa dibuat. Ini sama dengan jumlah maksimum pengguna (tercantum dalamHsmUsersMax) dikurangi pengguna yang dibuat sampai saat ini.
HsmUsersMax	Jumlah maksimum pengguna yang dapat dibuat pada instans HSM. Saat ini adalah 1.024.
Interface Eth2OctetsInput	Jumlah kumulatif lalu lintas masuk ke HSM sampai saat ini.
Interface Eth2OctetsOutput	Jumlah kumulatif lalu lintas keluar ke HSM sampai saat ini.

AWS CloudHSM Kinerja

Untuk cluster produksi, Anda harus memiliki setidaknya dua instans HSM yang tersebar di zona ketersediaan yang berbeda 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. Tabel di bawah ini menunjukkan perkiraan kinerja untuk algoritma kriptografi umum yang berjalan pada instans EC2. 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.

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

- [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 cluster Anda, Anda akan menerima pesan kesalahan yang menyatakan HSM sibuk atau dibatasi. 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 Services in Scope by Compliance Program .
- 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

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 kredensyal 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 sumber daya. AWS Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi yang lebih lengkap tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-2.

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, CLI, atau AWS SDK. 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 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 CloudHSMcadangan kluster.

Enkripsi dalam bergerak

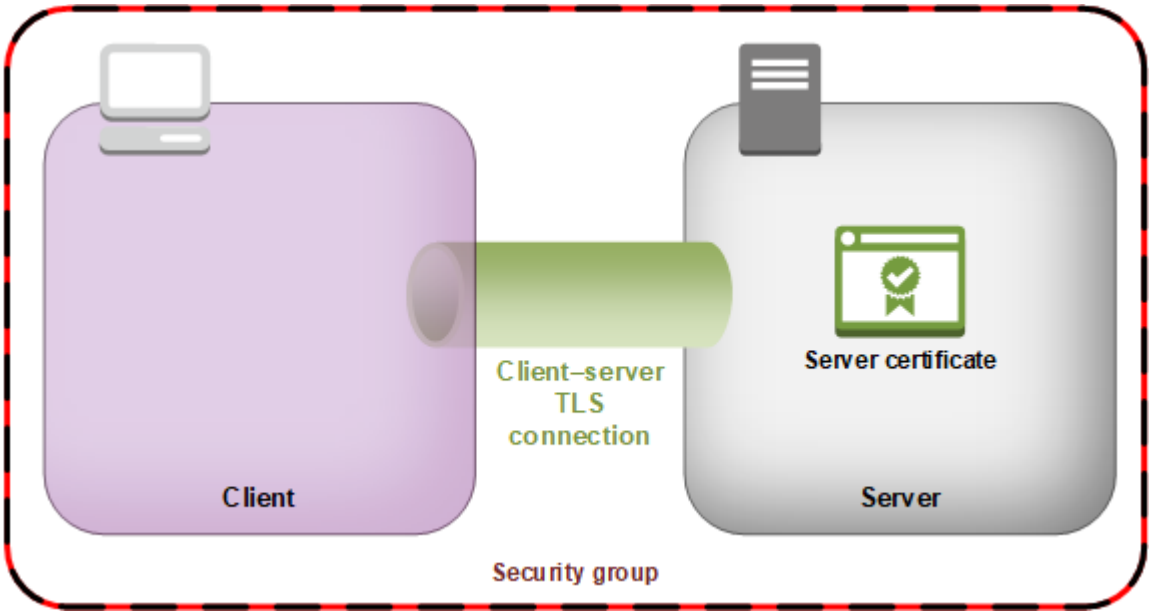
Komunikasi antara AWS CloudHSM klien dan HSM di cluster Anda dienkripsi dari ujung ke ujung. Komunikasi ini dapat didekripsi hanya oleh klien Anda dan HSM Anda. Untuk informasi selengkapnya, lihat nd-to-end Enkripsi E.

AWS CloudHSM end-to-end enkripsi klien

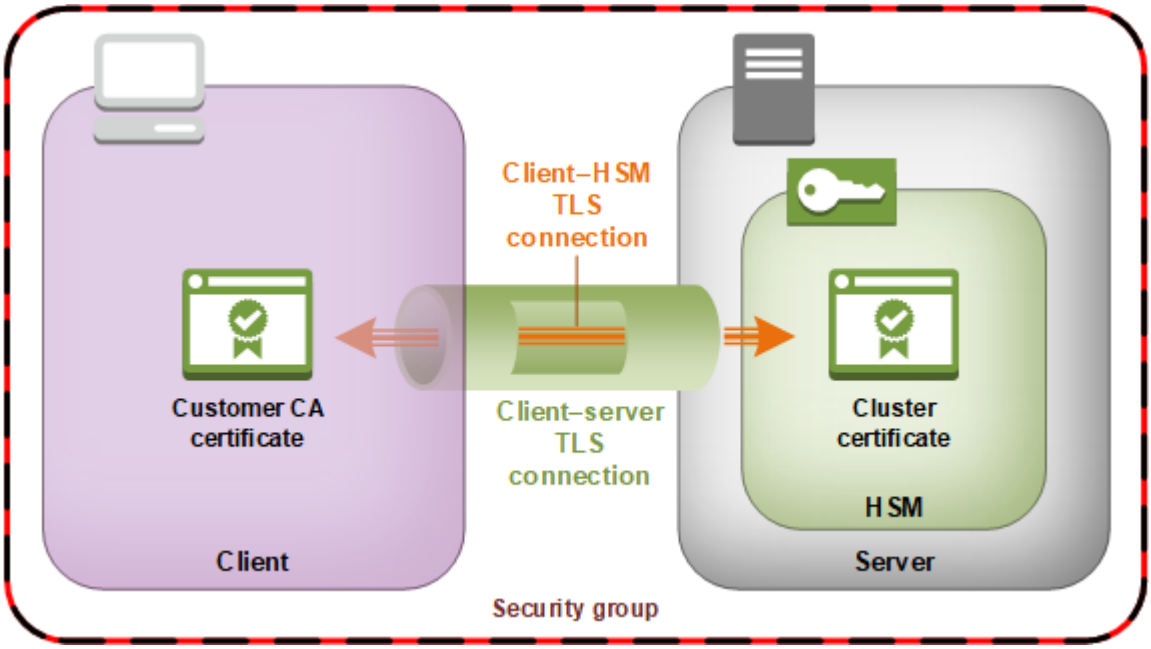
Komunikasi antara instans klien dan HSM di klaster Anda dienkripsi dari ujung ke ujung. Hanya klien Anda dan HSM Anda yang dapat mendekripsi komunikasi.

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 klaster 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 klaster yang Anda ditandatangani dengan sertifikat otoritas (CA) Anda sendiri, dan klien memiliki sertifikat root CA. Sebelum sambungan terenkripsi klien-HSM didirikan, klien memverifikasi sertifikat klaster HSM terhadap sertifikat root. Sambungan dibuat hanya ketika klien berhasil memverifikasi bahwa HSM tepercaya.



Keamanan cadangan klaster

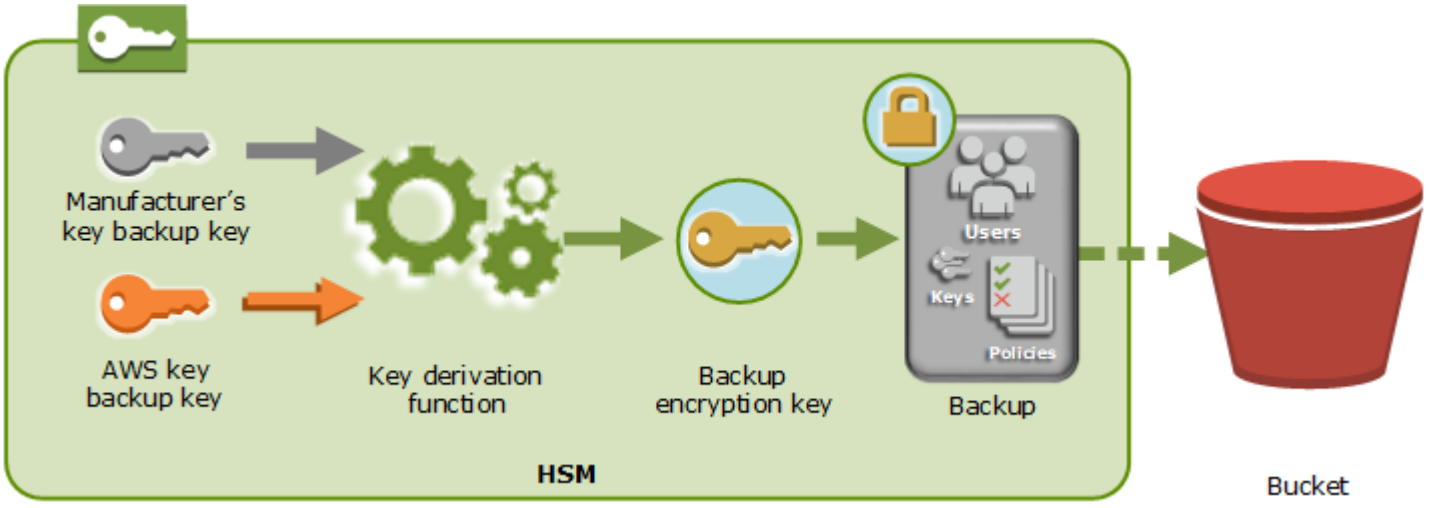
Ketika AWS CloudHSM membuat cadangan dari HSM, HSM mengenkripsi semua datanya sebelum mengirimnya ke. AWS CloudHSM Data tidak pernah meninggalkan HSM dalam bentuk plaintext. Selain itu, cadangan tidak dapat didekripsi oleh AWS karena AWS tidak memiliki akses ke kunci yang digunakan untuk mendekripsi 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 HSM AWS milik yang 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 terenkripsi. 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. Setiap pernyataan terstruktur seperti yang ditunjukkan dalam contoh berikut.

```
{
  "Version": "2012-10-17",
  "Statement":[{
    "Effect": "effect",
    "Action": "action",
    "Resource": "resource-arn",
    "Condition": {
      "condition": {
        "key":"value"
      }
    }
  }]
}
```

- 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 menentukantindakan, 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 persegi 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:

- AWSCloudHSMFullAccess— 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 klaster dan HSM. Karena grup ini tidak memungkinkan pengguna untuk menghapus klaster atau HSM, pengguna tepercaya 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-contoh

- Izin Baca Saja
- 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. Ini tidak memungkinkan pengguna untuk menghapus klaster atau HSM.

```
{
   "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. Ini tidak memungkinkan pengguna untuk menghapus klaster atau HSM. Anda harus menyertakan `iam:CreateServiceLinkedRole` tindakan AWS CloudHSM untuk memungkinkan secara otomatis membuat peran AWSServiceRoleForCloudHSMterkait layanan 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.

```
{
   "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 AWSServiceRoleForCloudHSMterkait layanan di akun Anda. Peran ini memungkinkan AWS CloudHSM untuk mencatat peristiwa. Untuk informasi selengkapnya, lihat Peran terkait layanan untuk AWS CloudHSM.

```
{
   "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. AWSServiceRoleForCloudHSM 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.

```
{
    "Version": "2018-06-12",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
```

```
    ]
}
```

Kebijakan kepercayaan untuk AWSServiceRoleForCloudHSMperan memungkinkan AWS CloudHSM untuk mengambil peran.

```
{
  "Version": "2018-06-12",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudhsm.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Membuat peran terkait layanan (otomatis)

AWS CloudHSM membuat AWSServiceRoleForCloudHSMperan saat Anda membuat klaster 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 AWSServiceRoleForCloudHSMperan, Anda dapat menggunakan konsol, perintah create-cluster, atau operasi CreateClusterAPI untuk membuat cluster. Kemudian gunakan konsol, perintah delete-cluster, atau operasi DeleteClusterAPI untuk menghapusnya. Membuat klaster baru menciptakan peran terkait layanan dan berlaku untuk semua klaster 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 AWSServiceRoleForCloudHSMperan.

Membuat peran terkait layanan (manual)

Anda dapat menggunakan konsol IAM, AWS CLI, atau API untuk membuat AWSServiceRoleForCloudHSMperan. 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 AWSServiceRoleForCloudHSMperan. 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 klaster yang telah diterapkan masih ada. Untuk menghapus peran, Anda harus terlebih dahulu menghapus setiap HSM di klaster Anda dan kemudian menghapus klaster. Setiap klaster di akun Anda harus dihapus. Anda kemudian dapat menggunakan konsol IAM, AWS CLI, atau API untuk menghapus peran. Untuk informasi lebih lanjut tentang penghapusan klaster, lihat Menghapus sebuah cluster AWS CloudHSM. Untuk informasi lebih lanjut, lihat Menghapus Peran Tertaut Layanan dalam Panduan Pengguna IAM.

Kepatuhan

AWS CloudHSM menyediakan kebijakan keamanan yang disetujui FIPS untuk HSM. Selain itu, AWS CloudHSM memenuhi persyaratan kepatuhan PCI-PIN, PCI-3DS, dan SOC2. Mengandalkan HSM yang divalidasi FIPS dapat membantu Anda memenuhi persyaratan kepatuhan perusahaan, kontrak, dan peraturan untuk keamanan data di Cloud. AWS Untuk informasi lebih lanjut, lihat informasi di bawah ini.

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. HSM yang AWS CloudHSM disediakan oleh FIPS 140-2 level 3 bersertifikat (Sertifikat #4218). Untuk informasi lebih lanjut tentang ini, 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. HSM yang disediakan oleh AWS CloudHSM mematuhi PCI DSS.

Kepatuhan PIN PCI

PIN PCI menyediakan persyaratan keamanan dan standar penilaian untuk transmisi, pemrosesan, dan pengelolaan data nomor identifikasi pribadi (PIN), informasi yang digunakan untuk transaksi di ATM dan terminal point-of-sale (POS). AWS CloudHSM telah mematuhi PIN PCI sejak Januari 2023. 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. AWS CloudHSM sesuai dengan PCI-3DS.

SOC2

SOC2 adalah kerangka kerja untuk membantu organisasi layanan mendemonstrasikan kontrol keamanan cloud dan pusat data mereka. AWS CloudHSM telah menerapkan kontrol SOC2 di area kritis untuk mematuhi prinsip layanan tepercaya. Untuk informasi lebih lanjut, lihat halaman FAQ AWS SOC.

AWS CloudHSM FAQ kepatuhan PCI-PIN

PIN PCI menyediakan persyaratan keamanan dan standar penilaian untuk transmisi, pemrosesan, dan pengelolaan data nomor identifikasi pribadi (PIN), informasi yang digunakan untuk transaksi di ATM 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.

FAQ

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 PCI-PIN Attestion of Compliance (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 HSM. 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 menyediakan HSM tujuan umum. 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 dan SOC2. Halaman ini mencantumkan perubahan yang saat ini berlaku.

Kepatuhan FIPS 140: Penutupan Mekanisme 2024

Institut Standar dan Teknologi Nasional (NIST) [1] menyarankan bahwa dukungan untuk enkripsi Triple DES (DeSeDE, 3DES, DES3) dan pembungkus kunci RSA dan buka pembungkus dengan padding PKCS #1 v1.5 tidak diizinkan setelah 31 Desember 2023. Oleh karena itu, dukungan untuk akhir ini pada 1 Januari 2024 dalam contoh yang sesuai dengan Standar Pemrosesan Informasi Federal (FIPS) kami.

Panduan ini berlaku untuk operasi kriptografi berikut:

- Generasi kunci Triple DES
  - `CKM_DES3_KEY_GEN`untuk Perpustakaan PKCS #11
  - `DESede`Keygen 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_PKCS`bungkus, buka bungkus, enkripsi, dan dekripsi untuk PKCS #11 SDK

- `RSA/ECB/PKCS1Padding`bungkus, buka bungkus, enkripsi, dan dekripsi untuk JCE SDK
- `wrapKey`dan `unWrapKey` dengan `-m 12` untuk KMU (catatan `12` adalah nilai untuk mekanisme`RSA_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 Ketersediaan klaster tinggi dan penyeimbangan beban.

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 dengan menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan pengguna utama 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 Membuat virtual private cloud (VPC).

Saat Anda membuat HSM, AWS CloudHSM letakkan elastic network interface (ENI) di subnet Anda sehingga Anda dapat berinteraksi dengan HSM Anda. Untuk informasi selengkapnya, lihat 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 HSM di klaster Anda. Untuk informasi lebih lanjut, lihat Konfigurasikan grup keamanan instans Amazon EC2 Klien.

Otorisasi pengguna

Dengan AWS CloudHSM, operasi yang dilakukan pada HSM memerlukan kredensyal dari pengguna HSM yang diautentikasi. Untuk informasi selengkapnya, lihat the section called “Memahami pengguna HSM”.

AWS CloudHSM dan titik akhir VPC

Anda dapat membuat koneksi pribadi antara VPC Anda dan AWS CloudHSM dengan membuat antarmuka VPC endpoint. Endpoint antarmuka didukung oleh AWS PrivateLink, teknologi yang memungkinkan Anda mengakses AWS CloudHSM API secara pribadi 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 dengan API. 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.

Buat VPC endpoint antarmuka untuk AWS CloudHSM

Anda dapat membuat titik akhir VPC untuk AWS CloudHSM layanan menggunakan konsol VPC Amazon atau (CLI). AWS Command Line Interface 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`) akan diselesaikan ke titik akhir VPC Anda.

Opsi ini mempermudah untuk menggunakan VPC endpoint. AWS SDK dan CLI menggunakan nama host DNS AWS CloudHSM 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. 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 menemukan masalah dengan AWS CloudHSM, topik berikut dapat membantu Anda memecahkannya.

Topik

Masalah yang diketahui

AWS CloudHSM memiliki masalah yang diketahui berikut. Pilih topik untuk mempelajari selengkapnya.

Topik

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 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 HSM atau perangkat lunak lain 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 Bantalam 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 klaster Anda dan kemudian menambahkan HSM lain, yang mendapat alamat IP baru, daemon klien terus mencari HSM Anda di alamat IP asli mereka.
- Solusi: Jika Anda menjalankan beban kerja intermiten, kami sarankan Anda menggunakan `IpAddress` argumen dalam CreateHsmfungsi 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 oleh AWS CloudHSM

- 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 dan SDK secara eksplisit akan gagal jika penyangga data lebih besar dari 64 KB. Anda harus memperbarui klien dan SDK ke versi 1.1.1 atau lebih tinggi untuk mendapatkan manfaat dari perbaikan.

Masalah: Kunci yang diimpor tidak dapat ditetapkan 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 unWrapKeyartikel untuk informasi selengkapnya.

Masalah: Jika Anda memiliki satu HSM di klaster Anda, failover HSM tidak bekerja dengan benar

- Dampak: Jika instans HSM tunggal di klaster 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 klaster produksi. Jika Anda menggunakan konfigurasi ini, Anda tidak akan terpengaruh oleh masalah ini. Untuk klaster 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 HSM di klaster Anda dalam waktu singkat, klien memasuki keadaan galat 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 klaster 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 selengkapnya, lihat dokumentasi IMDSv2.

- 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 kredensyal.
- Solusi: Setel batas hop respons PUT menjadi setidaknya dua. untuk panduan tentang cara melakukannya, lihat Mengonfigurasi opsi metadata instance.

Masalah yang diketahui untuk pustaka PKCS #11

Topik

Masalah: Pembungkus kunci AES dalam versi 3.0.0 pustaka PKCS #11 tidak memvalidasi IV sebelum penggunaan

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 melewatkan NULL IV, atau tentukan IV default dari `0xA6A6A6A6A6A6A6A6`. Untuk informasi selengkapnya, lihat IV Kustom dengan Panjang Tidak Patuh untuk Pembungkus Kunci AES
  - 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

IV yang disediakan pengguna diabaikan secara diam-diam.

> **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 melewatkan 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 SDK untuk menerapkan hashing multibagian dengan benar. 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: Anda tidak dapat melakukan hash lebih dari 16 KB data

Untuk penyangga yang lebih besar, hanya 16 KB pertama akan di-hash dan dikembalikan. Kelebihan data akan diabaikan secara diam-diam.

- 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 dan SDK secara eksplisit akan gagal jika penyangga data lebih besar dari 64 KB. Anda harus memperbarui klien dan SDK ke versi 1.1.1 atau lebih tinggi untuk mendapatkan manfaat dari perbaikan.

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 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.

- 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.
- Pemecahan masalah: Jika Anda menerapkan Pembongkaran SSL/TLS di AWS CloudHSM, pembatasan 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: Kami sedang mengembangkan pilihan untuk melakukan derivasi kunci ECDH sepenuhnya dalam HSM. Implementasi yang diperbarui akan diumumkan pada halaman sejarah versi setelah tersedia.

Masalah: Verifikasi tanda tangan secp256k1 gagal pada platform EL6 seperti CentOS6 dan RHEL 6

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 EL6, inisialisasi gagal.

- Dampak: Verifikasi tanda tangan Secp256k1 akan gagal pada platform EL6. Panggilan verifikasi akan gagal dengan kesalahan `CKR_HOST_MEMORY`.
- Pemecahan masalah: Sebaiknya gunakan Amazon Linux 1 atau platform EL7 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 denga 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

Topik

- Masalah: Penyangga untuk enkripsi AES-GCM tidak dapat melebihi 16.000 byte
- Masalah: Derivasi kunci Eliptic-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 menyebabkan HSM Anda tiba-tiba kehabisan ruang kunci dan terjadi ketika aplikasi Anda menggunakan objek kunci JCE standar untuk operasi kripto, 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 membangun saat aplikasi berjalan dan dapat menyebabkan HSM 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: Kami menambahkan dukungan untuk penyimpanan sertifikat di penyimpanan kunci. Pembaruan akan diumumkan di halaman riwayat versi setelah tersedia.

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 mengharuskan vektor inisialisasi (IV) untuk AES-GCM dihasilkan pada HSM, IV untuk setiap potong data yang dienkripsi AES-GCM 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: Kami sedang mengembangkan pilihan untuk melakukan derivasi kunci ECDH sepenuhnya dalam HSM. Bila tersedia, kami akan mengumumkan implementasi yang diperbarui pada halaman riwayat versi.

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 mengharapkan argumen menjadi jumlah byte kunci, padahal seharusnya jumlah bit kunci.

- Dampak: SDK klien versi 5.4.0 hingga 5.4.2 salah mengharapkan ukuran kunci diberikan ke API yang ditentukan sebagai byte.
- 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 mengharapkan 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
```
```

Peringatan ini tidak berdampak. Kami menyadari masalah ini dan sedang berupaya menyelesaikannya. Tidak diperlukan resolusi atau solusi.

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	`doFinal()` dalam mode enkripsi atau dekripsi `wrap()`dalam mode bungkus `unwrap()`dalam mode buka bungkus
KeyAgreement	`generateSecret()` atau `generateSecret(String)`
KeyPairGenerator	`generateKeyPair()` ,`genKeyPair()` , atau `reset()`
KeyStore	Tidak diperlukan metode

Operasi	Metode penyelesaian
MAC	`doFinal()` atau `reset()`
MessageDigest	`digest()` atau `reset()`
SecretKeyFactory	Tidak diperlukan metode
SecureRandom	Tidak diperlukan metode
Tanda tangan	`sign()`dalam mode tanda `verify()`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

Ini adalah masalah yang diketahui untuk OpenSSL Dynamic Engine

Topik

Masalah: Anda tidak dapat menginstal AWS CloudHSM OpenSSL Dynamic Engine pada RHEL 6 dan CentOS6

- 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 atau Penyedia JCE.
- 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 menandatangi/memverifikasi tanda tangan kriptografi yang 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 yang diketahui untuk instans Amazon EC2 yang menjalankan Amazon Linux 2

Masalah: Amazon Linux 2 versi 2018.07 menggunakan **`ncurses`** paket yang diperbarui (versi 6) yang saat ini tidak kompatibel dengan SDK AWS CloudHSM

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

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

Kegagalan Sinkronisasi Klien SDK 3

Di SDK Klien 3, jika sinkronisasi sisi klien gagal, AWS CloudHSM membuat 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 yaitu 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 kegagalanbatu nisan.

```
2018-12-24T18:28:54Z liquidSecurity ERR: print_node_ts_status:
[create_object_min_nodes]Key: 264617 failed to tombstone on node:1
```

Penyebab: AWS CloudHSM gagal menghapus dan menandai materi kunci yang tidak diinginkan.

Resolusi: HSM di klaster Anda berisi materi kunci yang tidak diinginkan yang tidak diinginkan yang tidak diinginkan yang tidak diinginkan yang 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 SDK Klien.

Untuk membuat kunci token lebih tahan lama,AWS CloudHSM mengagalkan operasi pembuatan kunci yang tidak berhasil pada jumlah minimum HSM yang ditentukan dalam pengaturan sinkronisasi sisi klien. Untuk informasi selengkapnya, lihat Sinkronisasi Kunci dalam AWS CloudHSM.

SDK Klien 3: Verifikasi kinerja HSM dengan alat pkpspeed

Topik ini menjelaskan cara memverifikasi performa HSM dengan SDK Klien 3.

Untuk memverifikasi performa HSM di klaster AWS CloudHSM, Anda dapat menggunakan alat pkpspeed (Linux) atau pkpspeed_blocking (Windows) yang disertakan dengan SDK Klien 3. Alat pkpspeed dijalankan 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 konfigurasikan AWS CloudHSM klien (Linux). Untuk informasi selengkapnya tentang menginstal klien di instans Windows, lihat Instal dan atur konfigurasiAWS CloudHSM klien (Windows).

Setelah Anda menginstal dan mengatur konfigurasi klien AWS CloudHSM, gunakan 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 kripto (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 CloudHSMFAQ untuk detailnya). Ini dapat diverifikasi dengan menggunakan alat CLI seperti yang didokumentasikan dalam Panduan AWS CloudHSM Pengguna dan menjalankan getHSMInfo 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 concurrency.
- 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: Tes ECDSA memverifikasi.
- 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] Dilarang setelah 2023 untuk kepatuhan FIPS sesuai panduan NIST. Lihat Kepatuhan FIPS 140: Penutupan Mekanisme 2024 untuk detail.

Contoh-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/...
```

Klien SDK 5 pengguna berisi nilai-nilai yang tidak konsisten

`user list`Perintah 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 klaster Anda. Ini berarti bahwa pengguna ada dengan properti yang berbeda pada HSM yang berbeda di cluster. Berdasarkan properti mana yang tidak konsisten, langkah-langkah perbaikan yang berbeda dapat diambil.

Tabel berikut mencakup langkah-langkah untuk menyelesaikan inkonsistensi 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, sepenuhnya menyelesaikan inkonsistensi untuk pengguna tersebut sebelum melanjutkan yang berikutnya.

> **Note**
>
> Untuk melakukan langkah-langkah ini Anda idealnya harus login sebagai admin. Jika akun admin Anda tidak konsisten, ikuti langkah-langkah ini masuk dengan admin dan ulangi langkah-langkahnya sampai semua properti konsisten. Setelah akun admin Anda konsisten, Anda dapat melanjutkan untuk menggunakan admin tersebut untuk menyinkronkan pengguna lain di klaster.

Properti yang tidak konsisten	Contoh keluaran daftar pengguna	Implikasi	Metode Pemulihan metode

Pengguna “peran” adalah “tidak konsisten”	<pre>{
	"username":
	"test_user",
	"role":
	"**inconsistent** ",
	
	"locked":
	"false",
	"mfa": [],
	"cluster-
	coverage":
	"full"
	}</pre>
		Pengguna ini adalah CryptoUser pada beberapa HSM, dan Admin di HSM lainnya. Hal ini dapat terjadi jika dua SDK mencoba untuk membuat pengguna yang sama, pada saat yang sama, dengan peran yang berbeda.Anda harus menghapus pengguna ini, dan membuatnya kembali dengan peran yang diinginkan.	1. Login sebagai admin.
			2. Hapus pengguna di semua HSM:
			
			user delete --username **<user's name>** --role admin
			
			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>**

Pengguna “cakupan kluster” adalah “tidak konsisten”	<pre>{
	"username":
	"test_user",
	
	"role": "crypto-u
	ser",
	"locked":
	"false",
	"mfa": [],</pre>
		Pengguna ini ada pada subset HSM di cluster.This dapat terjadi jikauser create sebagian berhasil, atau jikauser delete sebagian berhasil.
		
		Anda harus menyelesaikan
			Jika pengguna seharusnya tidak ada, ikuti langkah-langkah ini:
			
			1. Login sebagai admin.
			2. Jalankan perintah ini:

Properti yang tidak konsisten	Contoh keluaran daftar pengguna	Implikasi	Metode Pemulihan metode

	<pre>"cluster-coverage": "**inconsistent** "
	}</pre>
		operasi sebelumnya, baik membuat atau menghapus pengguna ini dari klaster Anda.	user delete --username**<user's name>** --role admin
			3. Sekarang, jalankan perintah berikut:
			user delete --username**<user's name>** --role crypto-user
			
			Jika pengguna harus ada, ikuti langkah-l angkah ini:
			1. Login sebagai admin.
			2. Jalankan perintah berikut:
			user create --username **<user's name>** --role **<desired role>**

Properti yang tidak konsisten	Contoh keluaran daftar pengguna	Implikasi	Metode Pemulihan metode
Parameter pengguna “terkunci” adalah “tidak konsisten” atau “benar”	{ "username": "test_user", "role": "crypto-user", "locked": **inconsistent**, "mfa": [], "cluster-coverage": "full" }	Pengguna ini terkunci pada subset HSM. Hal ini dapat terjadi jika pengguna menggunakan password yang salah dan hanya terhubung ke subset HSM di cluster. Anda harus mengubah kredensi pengguna agar konsisten di seluruh klaster.	Jika pengguna mengaktifkan MFA, ikuti langkah-langkah berikut: 1. Login sebagai admin. 2. Jalankan perintah berikut untuk menonaktifkan sementara MFA: user change-mfa token-sign --username **<user's name>** --role **<desired role>** --disable 3. Ubah kata sandi pengguna sehingga mereka dapat masuk ke semua HSM: user change-password --username **<user's name>** --role **<desired role>** Jika MFA harus aktif bagi pengguna, ikuti langkah-langkah berikut:

Properti yang tidak konsisten	Contoh keluaran daftar pengguna	Implikasi	Metode Pemulihan metode
			1. Minta pengguna masuk dan mengaktifkan 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 keluaran daftar pengguna	Implikasi	Metode Pemulihan metode

Status MFA “tidak konsisten”	{
	"username":
	"test_user",
	
	"role": "crypto-user",
	"locked":
	"false",
	"mfa": [
	{
	"strategy":
	"token-sign",
	"status":
	"**inconsistent** "
	}
	],
	"cluster-coverage":
	"full"
	}
		Pengguna ini memiliki bendera MFA yang berbeda pada HSM yang berbeda di klaster.
		
		Hal ini dapat terjadi jika operasi MFA hanya selesai pada subset HSM.
		
		Anda harus mengatur ulang kata sandi pengguna, dan memungkinkan mereka untuk mengaktifkan kembali MFA.
			Jika pengguna mengaktifkan MFA, ikuti langkah-langkah berikut:
			
			1. Login sebagai admin.
			2. Jalankan perintah berikut untuk menonaktifkan sementara MFA:
			
			user change-mfa token-sign --username **<user's name>** --role **<desired role>** --disable
			3. Anda juga perlu mengubah kata sandi pengguna sehingga mereka dapat masuk ke semua HSM:
			
			user change-password --username **<user's name>** --role **<desired role>**
			
			Jika MFA harus aktif bagi pengguna, ikuti

Properti yang tidak konsisten	Contoh keluaran daftar pengguna	Implikasi	Metode Pemulihan metode
			langkah-langkah berikut: 1. Minta pengguna masuk dan mengaktifkan 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>

Kesalahan terlihat selama pemeriksaan ketersediaan kunci

Masalah: 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 HSM 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 kunci menghasilkan-simetris atau kunci generate-asymmetric-pair .
- Sebuah daftar kunci 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 Parameter-parameter bagian alat Konfigurasi.
- Jika menggunakan cluster dengan dua HSM, hindari menggunakan operasi yang memicu kesalahan, kecuali selama kode inisialisasi.
- Tingkatkan jumlah HSM di cluster Anda menjadi setidaknya tiga.

Mengekstrak kunci menggunakan JCE

getEncoded,, atau GETS mengembalikan getPrivateExponent null

`getEncoded`,`getPrivateExponent`, dan `getS` akan mengembalikan null karena secara default dinonaktifkan. Untuk mengaktifkannya, lihatEkstraksi kunci menggunakan JCE.

Jika`getEncoded`,`getPrivateExponent`, dan `getS` mengembalikan null setelah diaktifkan, kunci Anda tidak memenuhi prasyarat yang tepat. Untuk informasi lebih lanjut, lihatEkstraksi kunci menggunakan JCE.

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
- Melindungi dan mengekstraksi kunci dari HSM

Pelambatan HSM

Ketika beban kerja Anda melebihi kapasitas HSM klaster Anda, Anda akan menerima pesan kesalahan yang menyatakan HSM sibuk atau dibatasi. Ketika ini terjadi, Anda mungkin melihat pengurangan throughput atau peningkatan tingkat permintaan penolakan dari HSM. Selain itu, HSM dapat mengirimkan galat sibuk berikut.

Untuk Client SDK 5

- Di PKCS11, kesalahan sibuk memetakan ke. `CKR_FUNCTION_FAILED` Kesalahan ini dapat terjadi karena beberapa alasan, tetapi jika pembatasan 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 memetakan `com.amazonaws.cloudhsm.jce.jni.exception.InternalException: Unexpected error with the Provider: The HSM could not queue the request for processing.`
- Kesalahan sibuk SDK lainnya mencetak pesan berikut:. `Received error response code from Server. Response Code: 187`

Untuk Client SDK 3

- Di PKCS11, kesalahan sibuk memetakan kesalahan. `CKR_OPERATION_ACTIVE`
- Di JCE, kesalahan sibuk memetakan `CFM2Exception` dengan status. `0xBB (187)` Aplikasi dapat menggunakan `getStatus()` fungsi `CFM2Exception` untuk memeriksa status apa yang dikembalikan oleh HSM.

- Kesalahan sibuk SDK lainnya akan mencetak pesan berikut: `HSM Error: HSM is already busy generating the keys(or random bytes) for another request.`

Resolusi

Anda dapat menyelesaikan 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 klaster Anda berukuran cukup untuk memenuhi beban puncak.

  > **Note**
  >
  > Untuk Client SDK 5.8.0 dan yang lebih baru, coba lagi perintah diaktifkan secara default. Untuk detail tentang konfigurasi perintah percobaan ulang setiap SDK, lihat. Konfigurasi lanjutan untuk alat konfigurasi Client SDK 5

- Tambahkan lebih banyak HSM ke klaster Anda dengan mengikuti petunjuk di. Menambahkan atau menghapus HSM dalam sebuah cluster AWS CloudHSM

  > **Important**
  >
  > Kami merekomendasikan pengujian muatan klaster Anda untuk menentukan beban puncak yang harus Anda antisipasi, dan kemudian tambahkan satu HSM lagi untuk memastikan ketersediaan tinggi.

Menjaga agar pengguna HSM tetap sinkron di seluruh HSM di klaster

Untuk mengelola pengguna HSM, Anda menggunakan alat baris perintah AWS CloudHSM dikenal sebagai cloudhsm_mgmt_util. Alat ini berkomunikasi hanya dengan HSM yang ada di file konfigurasi alat. Alat ini tidak menyadari HSM lain di klaster yang tidak dalam file konfigurasi.

AWS CloudHSM menyinkronkan kunci pada HSM Anda di semua HSM lain di klaster, tetapi tidak menyinkronkan pengguna atau kebijakan HSM. Bila Anda menggunakan cloudhsm_mgmt_util untuk mengelola pengguna HSM, perubahan pengguna ini mungkin hanya memengaruhi beberapa HSM

klaster — yang ada di file konfigurasi cloudhsm_mgmt_util. Hal ini dapat menyebabkan masalah ketika AWS CloudHSM menyinkronkan kunci di HSM di klaster, karena pengguna yang memiliki bukti kunci mungkin tidak ada pada semua HSM di klaster.

Untuk menghindari masalah ini, edit file konfigurasi cloudhsm_mgmt_util sebelummengelola pengguna. Untuk informasi lebih lanjut, lihat ???.

Kehilangan koneksi ke 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 tab HSM pada halaman detail klaster di konsol AWS CloudHSM.
   - Gunakan AWS Command Line Interface (CLI) untuk mengeluarkan perintah. describe-clusters

   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
```

HilangAWS CloudHSMlog audit di CloudWatch

Jika Anda membuat klaster sebelum 20 Januari 2018, Anda harus secara manual mengatur konfigurasi peran terkait layanan untuk mengaktifkan pengiriman log audit klaster tersebut. Untuk petunjuk tentang cara mengaktifkan peran terkait layanan di klaster HSM, lihat Memahami Peran Terkait Layanan, serta Membuat Peran Terkait Layanan di Panduan Pengguna IAM.

IV khusus dengan panjang yang tidak sesuai untuk bungkus kunci AES

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 dibungkus yang tidak dapat dipulihkan
- Tindakan yang harus Anda ambil jika kode Anda menghasilkan kunci dibungkus yang tidak dapat dipulihkan

Tentukan apakah kode Anda menghasilkan kunci dibungkus 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 `libpkcs11.so` di folder `/opt/cloudhsm/`

Kondisi	Bagaimana saya tahu?
	`lib`. 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, Anda mungkin menggunakan versi 3.0.0 dari pustaka PKCS #11. Untuk memeriksa versi perangkat lunak pada instans aplikasi Anda, gunakan perintah ini: `rpm -qa | grep ^cloudhsm`
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 (IV), seperti `CKM_AES_GCM` dan `CKM_CLOUDHSM_AES_GCM`, tidak terpengaruh oleh masalah ini. Pelajari lebih lanjut tentang fungsi dan mekanisme.

Kondisi	Bagaimana saya tahu?
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: `CK_MECHANISM mech = {CKM_AES_KEY_WRAP, IV_POINTER, IV_LENGTH};` 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 dibungkus yang tidak dapat dipulihkan”.

Tindakan yang harus Anda ambil jika kode Anda menghasilkan kunci dibungkus 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 patuh 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 menyarankan menggunakan IV lain untuk pembungkus kunci AES, dan secara eksplisit akan menonaktifkan IV kustom untuk pembungkus kunci AES dalam versi masa mendatang dari pustaka PKCS #11.

Contoh kode untuk menentukan IV dengan benar muncul di aes_wrapping.c di GitHub.

3. Identifikasi dan pulihkan kunci dibungkus yang ada

   Anda harus mengidentifikasi kunci yang dibungkus menggunakan versi 3.0.0 dari pustaka PKCS #11, dan kemudian menghubungi dukungan untuk bantuan (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 kegagalan pembuatan cluster

Saat Anda membuat klaster, AWS CloudHSM buat peran AWSServiceRoleForCloudHSM terkait layanan, 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 tertaut 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 AWS Support. Untuk informasi selengkapnya tentang peran AWSServiceRoleForCloudHSM terkait layanan, lihat. Peran terkait layanan untuk AWS CloudHSM

Topik

- Tambahkan izin yang hilang
- Buat peran terkait layanan secara manual
- Gunakan pengguna nonfederasi

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 lebih lanjut, lihat Membuat Peran yang Terhubung dengan Layanan di Panduan Pengguna IAM.

Gunakan pengguna nonfederasi

Pengguna federasi, yang kredensialnya berasal dari luar AWS, dapat melakukan banyak tugas dari pengguna nonfederasi. 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-gabugan dengan izin `iam:CreateServiceLinkedRole`, atau berikan kepada pengguna non-gabungan yang ada izin `iam:CreateServiceLinkedRole`. Kemudian minta pengguna itu membuat cluster dari CLI. Tindakan ini membuat peran terkait layanan di akun Anda.

Setelah peran terkait layanan dibuat, jika Anda suka, Anda dapat menghapus klaster yang dibuat pengguna non-gabungan. 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 CLI.

```
$  aws iam get-role --role-name AWSServiceRoleForCloudHSM
{
    "Role": {
        "Description": "Role for CloudHSM service operations",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": "sts:AssumeRole",
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "cloudhsm.amazonaws.com"
                    }
                }
            ]
        },
        "RoleId": "AROAJ4I6WN5QVGG5G7CBY",
        "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 klien

AWS CloudHSM menawarkan alat untuk SDK Klien 3 dan SDK Klien 5 untuk mengumpulkan informasi tentang lingkungan Anda untuk AWS Support untuk memecahkan masalah.

Topik

- Alat dukungan SDK SDK SDK SDK Klien
- Alat dukungan SDK SDK SDK SDK SDK Klien

Alat dukungan SDK SDK SDK SDK Klien

Skrip mengekstraksi informasi berikut:

- File konfigurasi untuk komponen SDK Klien 5
- Berkas log tersedia
- Versi sistem operasi saat ini
- Informasi paket

Menjalankan alat info untuk SDK SDK Klien SDK

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 ]_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.

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.

  ```
  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.

```
C:\Program Files\Amazon\CloudHSM\bin\jce_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`term`, lingkungan tanpa server akan dapat output ke. CloudWatch

Untuk mendapatkan log log klienCloudWatch, lihat Bekerja dengan grup log dan pengaliran log dalam Panduan Pengguna Amazon CloudWatch Logs.

Alat dukungan SDK SDK SDK SDK SDK Klien

Skrip mengekstraksi 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 SDK SDK Klien SDK

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.

Kuota AWS CloudHSM

Kuota, sebelumnya dikenal sebagai batas, adalah nilai yang ditetapkan untuk sumber daya AWS. Kuota berikut berlaku untuk sumber daya AWS CloudHSM per Wilayah AWS dan akun AWS. 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
HSM	6	Ya
HSM per klaster	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
Tombol maksimum per cluster	3.300
Maksimum pengguna per cluster	1,024
Panjang maksimum nama pengguna	31 karakter
Panjang sandi yang diperlukan	7 hingga 32 karakter
Jumlah maksimum koneksi klien bersamaan per cluster [1]	900
Jumlah maksimum sesi PKCS #11 per aplikasi	1,024

[1] Koneksi klien untuk Client SDK 3 adalah daemon klien. Untuk Client SDK 5, koneksi klien adalah aplikasi.

Untuk informasi selengkapnya, lihat Sumber daya sistem.

Sumber daya sistem

Kuota sumber daya sistem adalah kuota pada apa yang klien AWS CloudHSM diperbolehkan untuk menggunakannya saat berjalan.

Deskriptor file adalah mekanisme sistem operasi untuk mengidentifikasi dan mengelola file terbuka secara per proses.

Daemon klien CloudHSM menggunakan deskriptor file untuk mengelola koneksi antara aplikasi dan klien, serta antara klien dan server.

Secara default, konfigurasi klien CloudHSM akan mengalokasikan 3000 deskriptor file. Nilai default ini dirancang untuk menghasilkan sesi optimal dan kapasitas threading antara daemon klien dan HSM Anda.

Dalam keadaan yang jarang terjadi, jika Anda menjalankan klien Anda di lingkungan sumber daya terbatas, mungkin menjadi perlu untuk mengubah nilai default ini.

> **Note**
>
> Dengan mengubah nilai-nilai ini, performa klien CloudHSM Anda mungkin menderita dan/atau aplikasi Anda mungkin menjadi tidak dapat dioperasikan.

1. Edit file `/etc/security/limits.d/cloudhsm.conf`

```
#
# DO NOT EDIT THIS FILE
#
hsmuser soft nofile 3000
hsmuser hard nofile 3000
```

2. Edit nilai numerik, sesuai kebutuhan.

> **Note**
>
> Kuota `soft` harus kurang dari atau sama dengan kuota `hard`.

3. Mulai ulang proses daemon klien CloudHSM Anda.

> **Note**
>
> Opsi konfigurasi ini tidak tersedia pada platform Microsoft Windows.

Unduh untuk AWS CloudHSM Client SDK

Unduh

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 seperti Client SDK 3, dengan pengecualian dukungan untuk penyedia CNG dan KSP. Untuk informasi selengkapnya, lihat Perbandingan komponen SDK klien.

> **Note**
>
> Untuk informasi tentang platform apa yang didukung oleh setiap SDK Klien, lihat Platform yang didukung SDK Klien 5 danPlatform yang didukung SDK Klien 3.

Rilis terbaru

Bagian ini mencakup versi terbaru SDK Klien.

Rilis Klien SDK 5: Versi 5.12.0

Amazon Linux 2

Unduh perangkat lunak versi 5.12.0 untuk Amazon Linux 2 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- Mesin Dinamis OpenSSL (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- Penyedia JCE (SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)

- CloudHSM CLI (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

Unduh perangkat lunak versi 5.12.0 untuk Amazon Linux 2 pada arsitektur ARM64:

- Perpustakaan PKCS #11 (SHA256 checksum c28a1f27e23e6ab1550dab6a353c6c9338a391a84d57f4ac99a1a3a9810c753f)
- Mesin Dinamis OpenSSL (SHA256 checksum 7d2e864c31c13f55443c1b1d04589fbdd4558fe103954de4384691e2c429a872)
- Penyedia JCE (SHA256 checksum e9a35eb87b2f257c47fb083d286deb835da45858b2d89759ca7d5bb4ef747b4b)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 28b6f918912b5c63bf10018824b642a805b309c21947a1d0ebbdc44647e80554)

Amazon Linux 2023

Unduh perangkat lunak versi 5.12.0 untuk Amazon Linux 2023 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 02801365cba449c5238a4e5ad3df1ddf7edd00ade976f47e956e885286503f3f)
- Mesin Dinamis OpenSSL (SHA256 checksum 0abed69a7c6acaafdaabdcc5fab7d56611ffd94f5480cade6f8beace9aeae056)
- Penyedia JCE (SHA256 checksum 3d5d9a903d3a216eca40f92dbb0b4030b7a86ad7ceee8d62241c97a6e1881e25)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum f96671d882b862033bba0b3633448dc6a26e45a25063e29b79a5cd4b7fc4945c)

Unduh perangkat lunak versi 5.12.0 untuk Amazon Linux 2023 pada arsitektur ARM64:

- Perpustakaan PKCS #11 (SHA256 checksum 53d05006b46bda8e9c1dd76e8307a780bfe0a67b10a9a87723c97f94e29f5b8e)
- Mesin Dinamis OpenSSL (SHA256 checksum ec1cca8e01b3303ff9473eeef6b33dc85b6affac7a47387b098905f9f2fc85ba)
- Penyedia JCE (SHA256 checksum c828ae56f46233215b9f35798b5859ebdac962af442acbc457081c3baaa44f11)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)

- CloudHSM CLI (SHA256 checksum ddd5dcd68d01f4fafaf13dc0b4ddcf98e3731ed51bdd51f85535b29353644a9f)

CentOS 7 (7.8+)

Unduh perangkat lunak versi 5.12.0 untuk CentOS 7 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- Mesin Dinamis OpenSSL (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- Penyedia JCE (SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

RHEL 7 (7.8+)

Unduh perangkat lunak versi 5.12.0 untuk RHEL 7 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- Mesin Dinamis OpenSSL (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- Penyedia JCE (SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

RHEL 8 (8.3+)

Unduh perangkat lunak versi 5.12.0 untuk RHEL 8 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 6e51e95122fd0991278888287f0c408808b26fb5f1196c46168477b9090fc478)
- Mesin Dinamis OpenSSL (SHA256 checksum 1f1d52ff7af6c537d8cfeb5973c691a9d90a518accd685ff9b66cd78daf98928)

- Penyedia JCE (SHA256 checksum 156944607de987d6b39bd8a2d21ccd294c01377a9e35f9f15f8b0f4c8bb90033)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 351e802f79dd2d0b5f7d23bb74c146be05e5169b603c9aace24189094a45a35d)

RHEL 9 (9.2+)

Unduh perangkat lunak versi 5.12.0 untuk RHEL 9 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum d1b2f4ac7e6e0c18e788512e7726bc68b571d99a1442ce2f2e80f4b0f9956266)
- Mesin Dinamis OpenSSL (SHA256 checksum cf86a3f17cd6c51969d4ce80c1e3ea6513b995611be7e2e72e5e5233c71d6add)
- Penyedia JCE (SHA256 checksum ae89e256eb89ec6b4fa0f001e7a4e1d8f1c08530423e81aa74d69a17b25d9a99)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum dfe6fe5d890c33b2f5d38f906ade113b06c8c05f3427a327744c454e7302f1a5)

Unduh perangkat lunak versi 5.12.0 untuk RHEL 9 pada arsitektur ARM64:

- Perpustakaan PKCS #11 (SHA256 checksum cad72a6ab2232b4c38b90d7c62147520b975d646773dd90d7be897fa0a537d2d)
- Mesin Dinamis OpenSSL (SHA256 checksum ad751f756530a2317c3c64380ea3a07865b13e1874fab0e61ac530b21487c7fb)
- Penyedia JCE (SHA256 checksum d204e69acfbb90996fb08ae3573607b65630b1124fb379e078c002d55ac07766)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum c0f412cc59bafd235e046cdc1a0c5d330f2d72f7d6434672e9522f86bc945090)

Ubuntu 20.04 LTS

Unduh perangkat lunak versi 5.12.0 untuk Ubuntu 20.04 LTS pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum d37b1f872eb2b1ab34303d5b8b803daa925902b645c57c6e15a28bb6321e0f42)

- Mesin Dinamis OpenSSL (SHA256 checksum cdc6e737652556b57d26d8816b2bc9820128cb3919360660b6f7fe65f9d39e3f)
- Penyedia JCE (SHA256 checksum f567a08344414a4776e1c5a9715657476925ca32695c4c2dd84a4f3fc5dc1615)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum f2ee5ad01c5018fc3670f602228fd71087228cd3923bf5b9bc73e4d7084dac6c)

Ubuntu 22.04 LTS

Unduh perangkat lunak versi 5.12.0 untuk Ubuntu 22.04 LTS pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 0e78928acd7a1662e4b07b15d5c3ccb88714ff89e47b991c8ab6e4c2229ee5aa)
- Mesin Dinamis OpenSSL (SHA256 checksum 4f3168745edc5592234891a7b1d82b179a4947e87c72fade1be3bad58b7ed1a3)
- Penyedia JCE (SHA256 checksum d4c3655cdc2b00d1ab5ceafac94dfbc5c5244ed20e10fdd9db9f4e741e013733)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum d00bbacb6f2e57bd92d832a2bd11cadede972f8e82cc402ec0684b9c6b23123c)

Unduh perangkat lunak versi 5.12.0 untuk Ubuntu 22.04 LTS pada arsitektur ARM64:

- Perpustakaan PKCS #11 (SHA256 checksum 0c1121535c523acb864215338292bab32acee438357878b5fc0b6d268713b86f)
- Mesin Dinamis OpenSSL (SHA256 checksum dc7a219302021570bc8c36674d2bd33165557bb2f9a0af8fdf114f1b85a70d84)
- Penyedia JCE (SHA256 checksum af3834a10081f1e4e7894275c8b9c7b7649b8de3b6f0aeb0781a3358183a9046)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum baa253ac62c2fbcc5712561e0fb0feb25461efc3ce68cf86d4c7bf0af0f14a34)

Windows Server 2016

Unduh perangkat lunak versi 5.12.0 untuk Windows Server 2016 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 11c3255fcc90b47810cfe4b2f71d56a006d295efccdd90f0d3f2dec5d2bab893)
- Penyedia JCE (SHA256 checksum 09001458196590f54352c0c8986f442003bfc2db71bac6392ce512899d386806)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum b446ad1387fe406dcc0a12b6de86fa98e9db4a18f9829b745efb87750c6e31ea)

Windows Server 2019

Unduh perangkat lunak versi 5.12.0 untuk Windows Server 2019 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 11c3255fcc90b47810cfe4b2f71d56a006d295efccdd90f0d3f2dec5d2bab893)
- Penyedia JCE (SHA256 checksum 09001458196590f54352c0c8986f442003bfc2db71bac6392ce512899d386806)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum b446ad1387fe406dcc0a12b6de86fa98e9db4a18f9829b745efb87750c6e31ea)

Client SDK 5.12.0 menambahkan dukungan ARM ke beberapa platform dan peningkatan kinerja untuk semua SDK. Fitur baru telah ditambahkan ke penyedia CloudHSM CLI dan JCE.

Dukungan platform

- Menambahkan dukungan untuk Amazon Linux 2023 pada arsitektur ARM64 untuk semua SDK.
- Menambahkan dukungan untuk Red Hat Enterprise Linux 9 (9.2+) pada arsitektur ARM64 untuk semua SDK.
- Menambahkan dukungan untuk Ubuntu 22.04 LTS pada arsitektur ARM64 untuk semua SDK.

CloudHSM CLI

- Ditambahkan perintah berikut:
  - replikasi kunci
- Menambahkan dukungan untuk menghubungkan ke beberapa cluster. Untuk informasi selengkapnya, lihat Menghubungkan ke beberapa cluster dengan 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 SDK.

Rilis SDK Klien sebelumnya

Bagian ini mencantumkan rilis SDK Klien sebelumnya.

Versi 5.11.0

Amazon Linux 2

Unduh perangkat lunak versi 5.11.0 untuk Amazon Linux 2 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- Mesin Dinamis OpenSSL (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- Penyedia JCE (SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

Unduh perangkat lunak versi 5.11.0 untuk Amazon Linux 2 pada arsitektur ARM64:

- Perpustakaan PKCS #11 (SHA256 checksum 5ac16449ec149c9b5e7776865803245ab17d0f1ad56df80173840c5e8d257b19)
- Mesin Dinamis OpenSSL (SHA256 checksum 28c2eb7f3f60172b0186e5c25f71bb7341537058a71f288673936766048083c1)

- Penyedia JCE (SHA256 checksum 06c9d9d281c12b1d2bd9a7b601d6317e46cedf175706bbfa3e4dcaed6ba05448)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 218982bb17aa751969a7866b0a9ff27e7aa5007a07817627d9cc1f7d60a78160)

Amazon Linux 2023

Unduh perangkat lunak versi 5.11.0 untuk Amazon Linux 2023 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 55310ab333d18bcfabdc4b74115b040386b4508934bdff93e1d054c4c4a6f9ea)
- Mesin Dinamis OpenSSL (SHA256 checksum f3d4934dc872a9b5212a180b9814ca2af3eca01ee228a8725563f1770add0dce)
- Penyedia JCE (SHA256 checksum 757d3abb515aeb08f4b1c83970ee0979399efee00ee78c9a9dbec05f4ed9768d)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 22af8f0501ff9a45a9e0683a408a63771c2c06c66abf5478d310d6d32e013555)

CentOS 7 (7.8+)

Unduh perangkat lunak versi 5.11.0 untuk CentOS 7 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- Mesin Dinamis OpenSSL (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- Penyedia JCE (SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

RHEL 7 (7.8+)

Unduh perangkat lunak versi 5.11.0 untuk RHEL 7 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- Mesin Dinamis OpenSSL (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- Penyedia JCE (SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

RHEL 8 (8.3+)

Unduh perangkat lunak versi 5.11.0 untuk RHEL 8 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum b95b9f588656fb14fd08bb66ce0e0da807b96daa38348dec07a508c9bef7403a)
- Mesin Dinamis OpenSSL (SHA256 checksum 7bb437b91a52e863b2b00ff7f427ce22522026daf757be873ee031ec6ffffd88)
- Penyedia JCE (SHA256 checksum e0db887e05eb535314f4d99f21da12d87d35ebb8baf9726f4ce8f01d9df0ea01)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 8485b5a6d679767ca9b4f611718159a643cf3e85090a8e4d20fe53c3707e25c3)

RHEL 9 (9.2+)

Unduh perangkat lunak versi 5.11.0 untuk RHEL 9 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 87b56a20accf67df53a203b7f115655b2acfaec4516682d4976d9475b10bec8e)
- Mesin Dinamis OpenSSL (SHA256 checksum 83a6b58572e985df937beede4b10e867b0ac6050ace8010dc8d535be365d2747)
- Penyedia JCE (SHA256 checksum ee95213d02d913250478d0793d6dd578e5c54d765e635c7468a49bdff4c2a6f3)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 7e168ed3bef8e9c5110645e9960680e9a57f7b94e16aec71422e3c67ebc58fb5)

Ubuntu 20.04 LTS

Unduh perangkat lunak versi 5.11.0 untuk Ubuntu 20.04 LTS pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum abc3a339d1fe5850db65620804e9a910f8b4f913624ef9b7189f2f0df1825c01)
- Mesin Dinamis OpenSSL (SHA256 checksum 075fc3f9974d552f27ad67fa92c8abff31b756b9add875b8cd4957e6801583a4)
- Penyedia JCE (SHA256 checksum 5de45c519133a0dae8da3ac01809db7974be25c14c15eb773fc5c972c0178c13)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 83e0e4505a063792c19feb3d4cfd032b9089091916168d92b0f51a967a007734)

Ubuntu 22.04 LTS

Unduh perangkat lunak versi 5.11.0 untuk Ubuntu 22.04 LTS pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum b8f20be125c8530b2a7bd945956e9c04296fba5634af408b40be4e03bdbad72a)
- Mesin Dinamis OpenSSL (SHA256 checksum d728c156eb4ee5c67159e57d6b092785800baa5fb61c14d64f460a8b8f53a778)
- Penyedia JCE (SHA256 checksum 44e943b8cd1176ad666e249342687744a280c6222df58b5a9f084c932f628284)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 8ccf5389d459611be813e42d7f9d040090f94f3fe88f9d110bcfb25e9619e4a7)

Windows Server 2016

Unduh perangkat lunak versi 5.11.0 untuk Windows Server 2016 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum aa4bce5be15bbe0978b7205c619bb91c55a8e0f1f4636be311f24878f7709e07)
- Penyedia JCE (SHA256 checksum 004cdb9ecb4a4d72458084997de7f562fb76a4e2f0567009f1dfafa7b2bded47)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 679795db759fda4823232142297a281e21a7d6f32cb5ddd6ac4c479866fa33b7)

Windows Server 2019

Unduh perangkat lunak versi 5.11.0 untuk Windows Server 2019 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum aa4bce5be15bbe0978b7205c619bb91c55a8e0f1f4636be311f24878f7709e07)
- Penyedia JCE (SHA256 checksum 004cdb9ecb4a4d72458084997de7f562fb76a4e2f0567009f1dfafa7b2bded47)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 679795db759fda4823232142297a281e21a7d6f32cb5ddd6ac4c479866fa33b7)

Client SDK 5.11.0 menambahkan fitur baru, meningkatkan stabilitas, dan menyertakan perbaikan bug untuk semua SDK.

Dukungan platform

- Menambahkan dukungan untuk Amazon Linux 2023 dan RHEL 9 (9.2+) untuk semua SDK.
- 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:
  - tanda kripto
  - verifikasi kripto
  - kunci impor pem
  - buka kunci
  - bungkus kunci
- file hasilkan kuncisekarang 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 logging di semua SDK untuk kunci Triple DES yang telah mencapai batas blok enkripsi maksimumnya, per FIPS 140-2.
- Menambahkan masalah yang diketahui untuk OpenSSL Dynamic Engine. Lihat Masalah yang diketahui untuk OpenSSL Dynamic Engine untuk detail.

Versi 5.10.0

Amazon Linux

Unduh perangkat lunak versi 5.10.0 untuk Amazon Linux pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum d63adf3e96c19c2d894b2defcbadd916dbb0398993050b1358bd93a36aa5acab)
- Mesin Dinamis OpenSSL (SHA256 checksum 4daa3e591ffd5f7ce8ef3759c41deaa38867f5e5d21f15927aea83afb1678ac5)
- Penyedia JCE (SHA256 checksum 6c1ac94d3080f1c609d9dafbcb14480911beef3a488c4ed6f2b11b377da9b477)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum c12617fcd7990ba53e96f477979b410e3a5f17842ca7a912861b8b820809b5b5)

Amazon Linux 2

Unduh perangkat lunak versi 5.10.0 untuk Amazon Linux 2 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- Mesin Dinamis OpenSSL (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- Penyedia JCE (SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

Unduh perangkat lunak versi 5.10.0 untuk Amazon Linux 2 pada arsitektur ARM64:

- Perpustakaan PKCS #11 (SHA256 checksum 5d8dfd835f1ed5a7f5a4fcc8ecf81cfa29883aca7e2985de69b5db723ab663db)
- Mesin Dinamis OpenSSL (SHA256 checksum 91fb8efe2646bf0dbd9087554baa09554714e9d56e9bfd5c0dc3023a9f485574)
- Penyedia JCE (SHA256 checksum 99f6e55c37fdf00085a816d46835aeff54470797b3b71f4d28a70dc79c9caf44)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum 4a88ba9b4cf0dd5573f3dd88ab9dc257e4c486069cb529c5d554979ee2dd83af)

CentOS 7 (7.8+)

Unduh perangkat lunak versi 5.10.0 untuk CentOS 7 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- Mesin Dinamis OpenSSL (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- Penyedia JCE (SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)

- CloudHSM CLI (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

RHEL 7 (7.8+)

Unduh perangkat lunak versi 5.10.0 untuk RHEL 7 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- Mesin Dinamis OpenSSL (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- Penyedia JCE (SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

RHEL 8 (8.3+)

Unduh perangkat lunak versi 5.10.0 untuk RHEL 8 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 96afb7042a148ddc7a60ab6235b49e176d0460d1c2957bd76ca3d8406ac1cb03)
- Mesin Dinamis OpenSSL (SHA256 checksum 2caad2bffea8aef73c91ad422d09772ef830fe7f80a7be19020e6a107eadfbe8)
- Penyedia JCE (SHA256 checksum 3543551f08fbe3900821ea2d4ea148b4e86e2334bc94d7ffef6f3b831457cd71)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum 812eccaadfc490f13bcd0b0a835ef58f3a3d4344ad7e0a237de476dd24509525)

Ubuntu 18.04 LTS

Unduh perangkat lunak versi 5.10.0 untuk Ubuntu 18.04 LTS pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum be4c61766b8b46e1f6c14c3dcf90aaab9f38240fcd9c68b4009704276c5f6f4a)
- Mesin Dinamis OpenSSL (SHA256 checksum 64bd8af827b6dc3786e8ad28858cbc4ef6a0fd42164a0945f427eddcf5f02858)

- Penyedia JCE (SHA256 checksum 9fcbdf08e93641468588b608173f26f18781bbc029ed95b2e086da29a968cc00)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum 13808bddddb7eedeb2b8486d23a9976c7fa8d9220149a6b9400626bcaff3b513)

> **Note**
>
> Karena akhir kehidupan baru-baru ini untuk Ubuntu 18.04 LTS, tidak AWS CloudHSM akan lagi dapat mendukung platform ini dengan rilis berikutnya.

Ubuntu 20.04 LTS

Unduh perangkat lunak versi 5.10.0 untuk Ubuntu 20.04 LTS pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 99ae96504580ff85ed4958a582903a847f666bdaafafbe887a5a76db58f24500)
- Mesin Dinamis OpenSSL (SHA256 checksum 13e3f6fe086acf9617b163f66e3941f973daa583fb9322d16c396aa29fc3611d)
- Penyedia JCE (SHA256 checksum 44562cebd9af1aa965840cd9bcb237e518d24c715b3c8bca1405c9c1871835e2)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum ab71b4ec531c5e6d05c91539c7edc1c07e6c748052ebf6200f148cb6812538c5)

Ubuntu 22.04 LTS

Unduh perangkat lunak versi 5.10.0 untuk Ubuntu 22.04 LTS pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum ee331a44fbe4936ec98a3ae55d58e67ed38e8bbff0a4f4ce8b1bd8239b75877b)
- Support untuk OpenSSL Dynamic Engine belum tersedia untuk platform ini.
- Penyedia JCE (SHA256 checksum 9e44d14dd33624f6fe36711633013e47e4a93f4d4635e08900546113ded56e3d)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum 2df361546848cd3f8965b1007dca42a0c959eb10d9e3f4995e8e1c852406751d)

Windows Server 2016

Unduh perangkat lunak versi 5.10.0 untuk Windows Server 2016 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 7aae9bfd99a6dd0f4d376c227c206c01847f83a9efd774d1063d76cc6fdaa89f)
- Penyedia JCE (SHA256 checksum 1c58fd651e51be2ba59051a87aceca0452990b29837b8a7efabcd510ccbf8c1f)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum f745a2236c9eb9f6f128313eddc35795bd5e47fdf67332bedeb2554201b61a24)

Windows Server 2019

Unduh perangkat lunak versi 5.10.0 untuk Windows Server 2019 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 7aae9bfd99a6dd0f4d376c227c206c01847f83a9efd774d1063d76cc6fdaa89f)
- Penyedia JCE (SHA256 checksum 1c58fd651e51be2ba59051a87aceca0452990b29837b8a7efabcd510ccbf8c1f)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum f745a2236c9eb9f6f128313eddc35795bd5e47fdf67332bedeb2554201b61a24)

Client SDK 5.10.0 meningkatkan stabilitas dan menyertakan perbaikan bug untuk semua SDK.

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 bungkusan, menurunkan, dan membuat kunci sesi untuk semua SDK.
- Memperbaiki bug di Penyedia JCE yang mencegah file temp dihapus saat keluar.
- Memperbaiki bug yang menyebabkan kesalahan koneksi dalam kondisi tertentu setelah HSM di cluster diganti.
- Format `getVersion` output JCE yang dimodifikasi untuk menangani nomor versi minor besar dan menyertakan nomor patch.

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

Amazon Linux

Unduh perangkat lunak versi 5.9.0 untuk Amazon Linux pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 4f368be41f006b751ac41b14e1435c27841f60bbde0f032ec02a359fea637dcf)
- Mesin Dinamis OpenSSL (SHA256 checksum 81af0d34683825cd6ff844ccacf9c8f4842a4ba76e3875a89121d09a286b4490)
- Penyedia JCE (SHA256 checksum e8e5bc09d8e0b3cb24f30ab420fe08902a19073012335ac94382ec55fcc45abd)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 17284144b45043204ce012fe8b62b1973f10068950abedbd9c2c6172ed0979c6)

Amazon Linux 2

Unduh perangkat lunak versi 5.9.0 untuk Amazon Linux 2 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)
- Mesin Dinamis OpenSSL (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- Penyedia JCE (SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

Unduh perangkat lunak versi 5.9.0 untuk Amazon Linux 2 pada arsitektur ARM64:

- Perpustakaan PKCS #11 (SHA256 checksum 4337dca5a08c5194b1118fa197bb4a4f7988df4e1b961e6f2e367295ba99d61d)
- Mesin Dinamis OpenSSL (SHA256 checksum 4f08689934e877662a7ce64554fb04eb4b2c213b936018609ff187d100e34a85)
- Penyedia JCE (SHA256 checksum b337b80271a2d308949d5911971fe6ad35df4e34876a481fcac347f1d897fe39)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum a4d466e6b5f74dcd283ba32c9dd87441941d5e5a05936b7c2b4cc7ef85eb1071)

CentOS 7 (7.8+)

Unduh perangkat lunak versi 5.9.0 untuk CentOS 7 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)
- Mesin Dinamis OpenSSL (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- Penyedia JCE (SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

RHEL 7 (7.8+)

Unduh perangkat lunak versi 5.9.0 untuk RHEL 7 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)
- Mesin Dinamis OpenSSL (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- Penyedia JCE (SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

RHEL 8 (8.3+)

Unduh perangkat lunak versi 5.9.0 untuk RHEL 8 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 081887f6ea1d9df9d1e409b2b5bde83e965c42229acbeb1f950c8fe478361edc)
- Mesin Dinamis OpenSSL (SHA256 checksum 6b0500a42fd57c39f076f14e5079f80145b6ebd2c441395761eb04600c07bda5)
- Penyedia JCE (SHA256 checksum 2bc7ac26b259af92a65fbd5a30d5eb2a92ce0e70efe41feb53bf82f168aa90bb)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 79ecbe9b4c5316ccf447d8c59b76b5ac2cc854bd79cd50c1f29197aa8cb080db)

Ubuntu 18.04 LTS

Unduh perangkat lunak versi 5.9.0 untuk Ubuntu 18.04 LTS pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum bc6d2227edd7b5a83fed32741fbacbb1756d5df89ebb3435d96f0609a180db65)
- Mesin Dinamis OpenSSL (SHA256 checksum 2d6a26434fa6faf337f1dfb42de033220fa405a82d4540e279639a03b3ee6e9d)
- Penyedia JCE (SHA256 checksum e12aef122f490e9026452ce31c25625b1accb9a5866b3d470488f10f047f1873)

  - Javadocs untuk AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum f0bcabe594db3e8ff86cc0f65c2a10858d34452eb6b9fc33d7aac05c0f5f4f30)

Ubuntu 20.04 LTS

Unduh perangkat lunak versi 5.9.0 untuk Ubuntu 20.04 LTS pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum 15dde8182f432de9e7d369b05e384e1f2d80dcca85db3b16ecc26cdef1a34bb9)
- Mesin Dinamis OpenSSL (SHA256 checksum c8ba94a999038af87d4905b7c1feb4cc87e20d1776a32ef6f6d11ee000b5a896)
- Penyedia JCE (SHA256 checksum de33cd3e8130a06d9da5207079533aac8276a1319ac435a3737b4f65bd8fb972)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum cfa31535ad9a99a5113496c06fbace38e9593491aca9bb031a18b51075973e68)

Windows Server 2016

Unduh perangkat lunak versi 5.9.0 untuk Windows Server 2016 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum ab5380805b0e17dd89dbbefd3fbda8b54da3c140f82e9f3d021850c31837bbe3)
- Penyedia JCE (SHA256 checksum f0941d7a20193818133de8a742d3b848ea19abaf25f5a71ac65949ce5a37c533)
  - Javadocs untuk AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 131530ffe5caff963d483f440d06dcfb41dc11b0f8d78f1dd07bb07f76aeb6d2)

Windows Server 2019

Unduh perangkat lunak versi 5.9.0 untuk Windows Server 2019 pada arsitektur x86_64:

- Perpustakaan PKCS #11 (SHA256 checksum ab5380805b0e17dd89dbbefd3fbda8b54da3c140f82e9f3d021850c31837bbe3)
- Penyedia JCE (SHA256 checksum f0941d7a20193818133de8a742d3b848ea19abaf25f5a71ac65949ce5a37c533)

- Javadocs untuk AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 131530ffe5caff963d483f440d06dcfb41dc11b0f8d78f1dd07bb07f76aeb6d2)

Client SDK 5.9.0 meningkatkan stabilitas dan menyertakan perbaikan bug untuk semua SDK. Optimalisasi telah dibuat untuk semua SDK untuk menginformasikan aplikasi kegagalan operasi segera ketika HSM ditentukan tidak tersedia. Rilis ini mencakup peningkatan kinerja untuk JCE.

Penyedia JCE

- Kinerja yang ditingkatkan
- Memperbaiki masalah yang diketahui untuk kelelahan kumpulan sesi

Versi 3.4.4

Untuk meningkatkan Klien SDK 3 pada platform Linux, Anda harus menggunakan perintah batch yang meningkatkan daemon klien dan semua pustaka pada saat yang sama. Untuk informasi selengkapnya tentang pemutakhiran, lihat Peningkatan SDK 3 Klien.

Untuk mengunduh perangkat lunak, pilih tab untuk sistem operasi pilihan Anda, lalu pilih tautan ke setiap paket perangkat lunak.

Amazon Linux

Unduh perangkat lunak versi 3.4.4 untuk Amazon Linux:

- AWS CloudHSM Klien (SHA256 checksum 900de424d70f41e661aa636f256a6a79cc43bea6b0fe6eb95c2aaa63e5289505)
- Perpustakaan PKCS #11 (SHA256 checksum a3f93f084d59fee5d7c859292bc02cb7e7f15fb06e971171ebf9b52bbd229c30)
- Mesin Dinamis OpenSSL (SHA256 checksum 8db07b9843d49016b0b6fec46d39881d94e426fcaae1cee2747be14af9313bb0)
- Penyedia JCE (SHA256 checksum 360617c55bf4caa8e6e78ede079ca68cf9ef11473e7918154c22ba908a219843)
- AWS CloudHSM Utilitas Manajemen (SHA256 checksum c9961ffe38921131bd6f3702e10d73588e68b8ab10fbb241723e676f4fa8c4fa)

Amazon Linux 2

Unduh perangkat lunak versi 3.4.4 untuk Amazon Linux 2:

- AWS CloudHSM Klien (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- Perpustakaan PKCS #11 (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- Mesin Dinamis OpenSSL (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- Penyedia JCE (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM Utilitas Manajemen (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

CentOS 6

AWS CloudHSM tidak mendukung CentOS 6 dengan Client SDK Versi 3.4.4.

Gunakan the section called “Versi 3.2.1” untuk CentOS 6 atau pilih platform yang didukung.

CentOS 7 (7.8+)

Unduh perangkat lunak versi 3.4.4 untuk CentOS 7:

- AWS CloudHSM Klien (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- Perpustakaan PKCS #11 (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- Mesin Dinamis OpenSSL (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- Penyedia JCE (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM Utilitas Manajemen (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

CentOS 8

Unduh perangkat lunak versi 3.4.4 untuk CentOS 8:

- AWS CloudHSM Klien (SHA256 checksum 81639c9ec83e501709c4117ba9d98b23dea7838a206ed244c9c6cc0d65130f8c)
- Perpustakaan PKCS #11 (SHA256 checksum 9a15daa87b8616cf03a6bf6b375f53451ef448dbc54bf2c27fbc2be7823fc633)
- Penyedia JCE (SHA256 checksum 2b1c4208992903cf7bcc669c1392c59a64fbfc82e010c626ffa58d0cb8e9126b)
- AWS CloudHSM Utilitas Manajemen (SHA256 checksum 3adbecc802e0854c23aa4b8d80540d1748903c8dba93b6c8042fb7885051c360)

> **Note**
>
> Karena End of Life of CentOS 8 baru-baru ini, kami tidak akan lagi dapat mendukung platform ini dengan rilis berikutnya.

RHEL 6

AWS CloudHSM tidak mendukung RedHat Enterprise Linux 6 dengan Client SDK Versi 3.4.4.

Gunakan the section called “Versi 3.2.1” untuk RedHat Enterprise Linux 6 atau pilih platform yang didukung.

RHEL 7 (7.8+)

Unduh perangkat lunak versi 3.4.4 untuk RedHat Enterprise Linux 7:

- AWS CloudHSM Klien (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- Perpustakaan PKCS #11 (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- Mesin Dinamis OpenSSL (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- Penyedia JCE (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM Utilitas Manajemen (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

RHEL 8 (8.3+)

Unduh perangkat lunak versi 3.4.4 untuk RedHat Enterprise Linux 8:

- AWS CloudHSM Klien (SHA256 checksum 81639c9ec83e501709c4117ba9d98b23dea7838a206ed244c9c6cc0d65130f8c)
- Perpustakaan PKCS #11 (SHA256 checksum 9a15daa87b8616cf03a6bf6b375f53451ef448dbc54bf2c27fbc2be7823fc633)
- Penyedia JCE (SHA256 checksum 2b1c4208992903cf7bcc669c1392c59a64fbfc82e010c626ffa58d0cb8e9126b)
- AWS CloudHSM Utilitas Manajemen (SHA256 checksum 3adbecc802e0854c23aa4b8d80540d1748903c8dba93b6c8042fb7885051c360)

Ubuntu 16.04 LTS

Unduh perangkat lunak versi 3.4.4 untuk Ubuntu 16.04 LTS:

- AWS CloudHSM Klien (SHA256 checksum 317c92c2e0b5d60afab1beb947f053d13ddaacb994cccc2c2b898e997ece29b9)
- Perpustakaan PKCS #11 (SHA256 checksum 91451c420c51488a022569fd32f052a3b988a2883ea4c2ac952acb61a2fea37c)
- Mesin Dinamis OpenSSL (SHA256 checksum 4098771ad0e38df9bf14d50520ca49b9395f819f0387e2bc3b0e61abb5888e66)
- Penyedia JCE (SHA256 checksum e136ff183271c2f9590a9fccb8261a7eb809506686b070e3854df1b8686c6641)
- AWS CloudHSM Utilitas Manajemen (SHA256 checksum cbf24a4032f393a913a9898b1b27036392104e8e05d911cab84049b2bcca2541)

> **Note**
>
> Karena EOL Ubuntu 16.04 yang akan datang, kami berniat untuk menjatuhkan dukungan untuk platform ini dengan rilis berikutnya.

Ubuntu 18.04 LTS

Unduh perangkat lunak versi 3.4.4 untuk Ubuntu 18.04 LTS:

- AWS CloudHSM Klien (SHA256 checksum cf57d5e0e95efbf032aac8887aebd59ac8cc80e97c69e7c39fdad40873374fe8)
- Perpustakaan PKCS #11 (SHA256 checksum 428f8bdad7925db5401112f707942ee8f3ca554f4ab53fa92237996e69144d2f)

- Penyedia JCE (SHA256 checksum 1ff17b8f7688e84f7f0bfc96383564dca598a1cab2f2c52c888d0361682f2b9e)
- AWS CloudHSM Utilitas Manajemen (SHA256 checksum afe253046146ed6177c520b681efc680dac1048c4a95b3d8ad0f305e79bbe93e)

Windows Server

AWS CloudHSM mendukung versi 64-bit Windows Server 2012, Windows Server 2012 R2, Windows Server 2016, dan Windows Server 2019. Perangkat lunak klien AWS CloudHSM 3.4.4 untuk Windows Server mencakup penyedia CNG dan KSP yang diperlukan. Untuk detailnya, lihat Menginstal dan Mengkonfigurasi AWS CloudHSM Klien (Windows). Unduh versi terbaru (3.4.4) perangkat lunak untuk Windows Server:

- AWS CloudHSM untuk Windows Server (SHA256 checksum d51a7db588e9121d8f0b0351606bd986e1c4de6547f2c8235200dc8a5ffbe53e)
- AWS CloudHSM Utilitas Manajemen (SHA256 checksum 0c12d7da9086735cdf189535937a8e036163009c5018dcaf2ee9cdbb6bd4c06f)

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.

Rilis usang

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 CLI, SSL/TLS offload dengan JSSE, dukungan multi-slot untuk PKCS #11, dukungan multi-cluster/multi-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 SDK.

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 NoPadding AES/CBC/.
- 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/, untuk mengenkripsi dan NoPadding 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

  - Mode AES/CBC/Enkripsi NoPadding dan Dekripsi
  - AES/ECB/PKCS5Padding Mode Enkripsi dan Dekripsi
  - Mode NoPadding AES/CTR/Enkripsi dan Dekripsi
  - Mode NoPadding AES/GCM/Bungkus dan Buka Bungkus
  - Desede/ECB/PKCS5Padding Mode Enkripsi dan Dekripsi
  - Mode Desede/CBC/Enkripsi NoPadding dan Dekripsi
  - Mode AESWrap/ECB/Bungkus dan Buka NoPadding Bungkus
  - Mode AESWrap/ECB/PKCS5Padding Bungkus dan Buka Bungkus
  - Mode AESWrap/ECB/Bungkus dan Buka ZeroPadding Bungkus
  - Mode RSA/ECB/PKCS1Padding Bungkus dan Buka Bungkus
  - Mode RSA/ECB/OAEPPadding Bungkus dan Buka Bungkus
  - RSA/ecb/oaepwithsha-1andmgf1padding mode Bungkus dan Buka Bungkus
  - RSA/ECB/OAEPWithSHA-224andmgf1mode Padding Bungkus dan Buka Bungkus
  - RSA/ECB/OAEPWithSHA-256andMGF1Mode Padding Bungkus dan Buka Bungkus
  - RSA/ECB/OAEPWithSHA-384andmgf1mode Padding Bungkus dan Buka Bungkus
  - RSA/ECB/OAEPWithSHA-512andmgf1mode Padding Bungkus dan Buka Bungkus
  - Mode RSAAESWrap/ECB/OAEPPadding Bungkus dan Buka Bungkus
  - RSAAESWrap/ECB/OAEPWithSHA-1andMGF1Mode Padding Bungkus dan Buka Bungkus
  - RSAAESWrap/ECB/OAEPWithSHA-224andmgf1mode Padding Bungkus dan Buka Bungkus
  - RSAAESWrap/ECB/OAEPWithSHA-256andMGF1Mode Padding Bungkus dan Buka Bungkus
  - RSAAESWrap/ECB/OAEPWithSHA-384andMGF1Mode Padding Bungkus dan Buka Bungkus
  - RSAAESWrap/ECB/OAEPWithSHA-512andmgf1mode Padding Bungkus dan Buka Bungkus
- KeyFactory dan SecretKeyFactory
  - 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 dukungan hash SHA1, SHA224, SHA256, SHA384, SHA512
- Tanda/Verifikasi

- RSASSA-PSS
- SHA1withRSA/PSS
- SHA224denganRSA/PSS
- SHA256denganRSA/PSS
- SHA384denganRSA/PSS
- SHA512denganRSA/PSS
- SHA1denganRSAandMGF1
- SHA224denganRSAANDMGF1
- SHA256denganRSAANDMGF1
- SHA384denganRSAandMGF1
- SHA512denganRSAANDMGF1

Versi 5.4.2

Versi 5.4.2 mencakup peningkatan stabilitas dan perbaikan bug untuk semua SDK. 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 dukungan hash SHA1, SHA224, SHA256, SHA384, SHA512.
- 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-1andmgf1padding
  - RSA/ECB/OAEPWithSHA-224andmgf1padding
  - RSA/ECB/OAEPWithSHA-256andmgf1padding
  - RSA/ECB/OAEPWithSHA-384andmgf1padding
  - RSA/ECB/OAEPWithSHA-512andmgf1padding
- Intisari
  - SHA-1
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- Tanda/Verifikasi
  - NoneWithRSA
  - SHA1denganRSA
  - SHA224denganRSA
  - SHA256denganRSA
  - SHA384denganRSA
  - SHA512denganRSA
  - NoneWithECDSA
  - SHA1denganECDSA
  - SHA224denganECDSA
  - SHA256denganECDSA
  - SHA384denganECDSA
  - SHA512denganECDSA

- Integrasi dengan Java KeyStore

Versi 5.3.0

Perpustakaan PKCS #11

- Peningkatan stabilitas dan perbaikan bug.

Mesin Dinamis OpenSSL

- Tambahkan dukungan untuk tanda tangan/verifikasi ECDSA 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 dukungan hash SHA1, SHA224, SHA256, SHA384, SHA512.
- 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-1andmgf1padding
  - RSA/ECB/OAEPWithSHA-224andmgf1padding
  - RSA/ECB/OAEPWithSHA-256andmgf1padding
  - RSA/ECB/OAEPWithSHA-384andmgf1padding
  - RSA/ECB/OAEPWithSHA-512andmgf1padding
- Intisari
  - SHA-1
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- Tanda/Verifikasi
  - NoneWithRSA
  - SHA1denganRSA
  - SHA224denganRSA
  - SHA256denganRSA
  - SHA384denganRSA
  - SHA512denganRSA
  - NoneWithECDSA
  - SHA1denganECDSA
  - SHA224denganECDSA
  - SHA256denganECDSA
  - SHA384denganECDSA
  - SHA512denganECDSA
- Integrasi dengan Java KeyStore

Versi 5.2.1

- 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_DES3_CBC
- CKM_DES3_CBC_PAD
- CKM_DES3_ECB
- 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_SP800_108_COUNTER_KDF
- CKM_GENERIC_SECRET_KEY_GEN
- CKM_SHA_1_HMAC
- CKM_SHA224_HMAC
- CKM_SHA256_HMAC
- CKM_SHA384_HMAC
- CKM_SHA512_HMAC
- CKM_RSA_PKCS_OAEP hanya Membungkus/Membuka
- CKM_RSA_AES_KEY_WRAP
- CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD
- CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD
- 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 jenis kunci dan API OpenSSL:
  - Pembuatan kunci RSA untuk tombol 2048, 3072, dan 4096 bit
  - API OpenSSL:
    - Tanda RSA menggunakan PKCS RSA dengan SHA1/224/256/384/512 & RSA PSS
    - 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.
- Dukungan untuk Pembongkaran SSL/TLS di CentOS 8.3+, Red Hat Enterprise Linux (RHEL) 8.3, dan Ubuntu 18.04 LTS, termasuk NGINX 1.19 (untuk suite cipher terpilih).

  Untuk informasi lebih lanjut, lihat Menggunakan Pembongkaran SSL/TLS di Linux.

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_SHA1_RSA_PKCS
- CKM_SHA224
- CKM_SHA224_RSA_PKCS
- CKM_SHA256
- CKM_SHA256_RSA_PKCS
- CKM_SHA384
- CKM_SHA384_RSA_PKCS
- CKM_SHA512

- CKM_SHA512_RSA_PKCS

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 3.4.3

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 kripto (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 lebih lanjut, 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 lebih lanjut, 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 lebih lanjut, 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

- Perbaikan 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 kustom ketika membungkus kunci menggunakan AES_KEY_WRAP_PAD_PKCS5. 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

- PERUBAHAN: TLS 1.0 dan 1.2 cipher suite dengan SHA1 tidak 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

- PERUBAHAN : Untuk mengatasi masalah dengan kepatuhan Java Cryptography Extension (JCE), bungkus dan buka bungkus AES sekarang menggunakan dengan benar algoritme AESWrap, bukan algoritma AES. Ini berarti `Cipher.WRAP_MODE` dan `Cipher.UNWRAP_MODE` tidak lagi berhasil untuk mekanisme AES/ECB dan 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 klien versi 3.0.0 atau sebelumnya, maka di 3.1.1 Anda harus menggunakan Bantalan AESWrap/ECB/PKCS5Padding untuk membuka kunci Anda 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.

nd-of-life Rilis E

AWS CloudHSM mengumumkan akhir masa pakai untuk rilis yang tidak lagi kompatibel dengan layanan. Untuk menjaga keamanan aplikasi Anda, kami berhak untuk secara aktif menolak koneksi dari end-of-life rilis.

- Saat ini tidak ada versi SDK klien yang end-of-life dirilis.

Riwayat dokumen

Topik ini menjelaskan pembaruan yang signifikan untuk Panduan Pengguna AWS CloudHSM .

Topik

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 5.12.0.	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. AWS 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 Enterpris e 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 sebelumnya

Tabel berikut menjelaskan perubahan penting AWS CloudHSM sebelum 2018.

Perubahan	Deskripsi	Tanggal
Konten baru	Penambahan autentikasi kuorum (M dari N kontrol akses) untuk petugas kripto (CO). Untuk informasi lebih lanjut, lihat Menggunakan CMU Manajemen CMU CMU	9 November 2017

Perubahan	Deskripsi	Tanggal
	untuk mengelola autentikasi kuorum autentikasi kuorum kontrol akses M dari N kontrol kuorum untuk mengelola autentikasi kuorum autentikasi kuorum.	
Perbarui	Penambahan dokumentasi tentang menggunakan alat baris perintah key_mgmt_util. Untuk informasi lebih lanjut, lihat key_mgmt_util referensi perintah.	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.