AWS KMS 암호화 세부 정보

AWS Key Management Service

AWS Key Management Service: AWS KMS 암호화 세부 정보

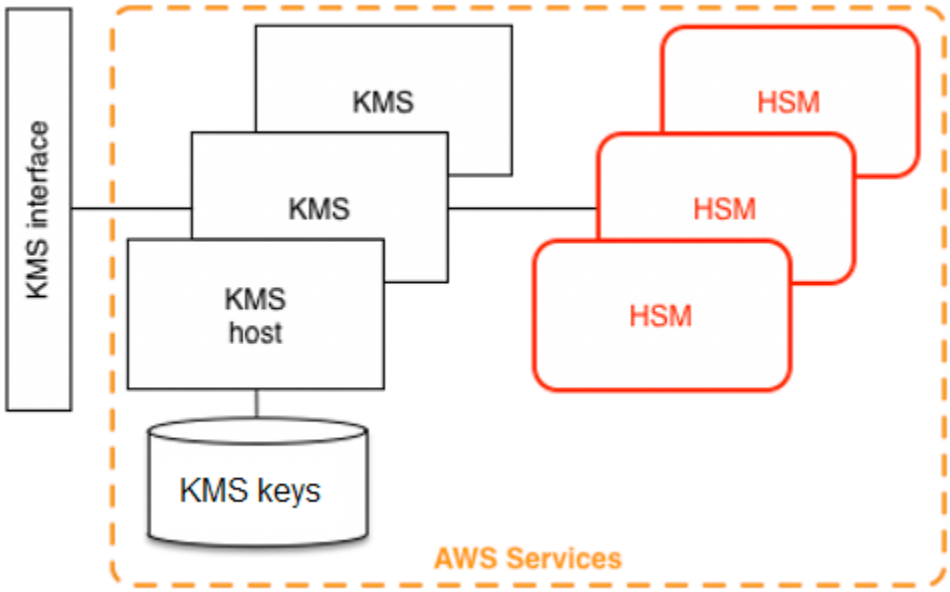
Table of Contents

의 암호화 세부 정보 소개 AWS KMS

AWS Key Management Service (AWS KMS)는 암호화 키를 생성 및 관리할 수 있는 웹 인터페이스를 제공하며 데이터를 보호하기 위한 암호화 서비스 공급자 역할을 합니다. AWS KMS 는 AWS 서비스와 통합된 기존 키 관리 서비스를 제공하여 중앙 집중식 관리 및 감사를 AWS통해 전체 고객의 키를 일관되게 볼 수 있습니다. 이 백서에서는 서비스에서 제공하는 기능을 평가하는 AWS KMS 데 도움이 되는 의 암호화 작업에 대한 자세한 설명을 제공합니다.

AWS KMS 에는 FIPS 140-3 검증 하드웨어 보안 모듈(HSM)[1]의 분산 플릿의 암호화 작업을 요청하는 AWS Management Console명령줄 인터페이스 및 RESTful API 작업을 통한 웹 인터페이스가 포함되어 있습니다.HSMs AWS KMS HSM은의 보안 및 확장성 요구 사항을 충족하는 전용 암호화 기능을 제공하도록 설계된 멀티칩 독립 실행형 하드웨어 암호화 어플라이언스입니다 AWS KMS. AWS KMS keys와 같이 사용자가 관리하는 키로 HSM 기반 암호화 계층 구조를 설정할 수 있습니다. 이러한 키는 HSM에서만 사용할 수 있으며 암호화 요청을 처리하는 데 필요한 시간 동안 메모리에서만 사용할 수 있습니다. KMS 키를 여러 개 생성할 수 있으며 각 키는 키 ID로 표시됩니다. 각 고객이 관리하는 AWS IAM 역할 및 계정에서만 고객 KMS 키를 생성, 삭제 또는 사용하여 데이터를 암호화, 복호화, 서명 또는 확인할 수 있습니다. 키에 연결된 정책을 생성하여 KMS 키를 관리하거나 사용하도록 허용되는 사용자에 대한 액세스 제어를 정의할 수 있습니다. 이러한 정책을 사용하면 각 API 작업을 위한 키의 애플리케이션별 용도를 정의할 수 있습니다.

또한 대부분의 AWS 서비스는 KMS 키를 사용하여 저장 데이터 암호화를 지원합니다. 이 기능을 통해 고객은 KMS 키에 액세스하는 방법과 시기를 제어하여 AWS 서비스가 암호화된 데이터에 액세스하는 방법과 시기를 제어할 수 있습니다.



AWS KMS 는 웹 경계 AWS KMS 호스트와 HSMs. 이러한 계층형 호스트의 그룹화는 AWS KMS 스택을 형성합니다. 에 대한 모든 요청은 TLS(전송 계층 보안 프로토콜)를 통해 이루어져야 하며 AWS KMS host. AWS KMS hosts에서 종료되어야 AWS KMS 합니다.는 완벽한 전달 secrecy. AWS KMS authenticates를 제공하는 ciphersuite가 있는 TLS만 허용하고 다른 모든 AWS API 작업에 사용할 수 있는 (IAM)의 AWS Identity and Access Management 동일한 자격 증명 및 정책 메커니즘을 사용하여 요청을 승인합니다.

기본 개념

몇 가지 기본 용어와 개념을 학습하면 최대한 활용할 수 있습니다 AWS Key Management Service.

AWS KMS key

> **Note**
>
> AWS KMS 는 고객 마스터 키(CMK)라는 용어를 AWS KMS key 및 KMS 키로 대체합니다. 단, 개념은 바뀌지 않았습니다. 변경 중단을 방지하기 위해 AWS KMS 는이 용어의 몇 가지 변형을 유지합니다.

키 계층의 최상위를 나타내는 논리적 키입니다. KMS 키에는 고유 키 식별자 또는 키 ID가 포함된 Amazon 리소스 이름(ARN)이 지정됩니다. AWS KMS keys 에는 세 가지 키 유형이 있습니다.

- 고객 관리 키 - 고객은 고객 관리형 키의 수명 주기 및 키 정책을 생성하고 제어할 수 있습니다. 이러한 키에 대해 수행되는 모든 요청은 CloudTrail 이벤트로 로깅됩니다.
- AWS 관리형 키 - 고객의 리소스 AWS 관리형 키인의 수명 주기 및 키 정책을 AWS 생성하고 제어합니다 AWS 계정. 고객은 AWS 관리형 키에 대한 액세스 정책과 CloudTrail 이벤트를 볼 수 있지만 이러한 키의 어떤 측면도 관리할 수 없습니다. 이러한 키에 대해 수행되는 모든 요청은 CloudTrail 이벤트로 로깅됩니다.
- AWS 소유 키 - 이러한 키는에서 생성되어 다양한 AWS 서비스에서 내부 암호화 작업에 독점적 AWS 으로 사용됩니다. 고객은 CloudTrail의 키 정책 또는 AWS 소유 키 사용량을 볼 수 없습니다.

별칭

KMS 키와 연결된, 사용자에게 친숙한 이름입니다. 별칭은 많은 AWS KMS API 작업에서 키 ID와 상호 교환하여 사용할 수 있습니다.

권한

키에 대한 권한을 정의하는, KMS 키에 연결된 정책입니다. 기본 정책은 정의한 모든 보안 주체를 허용하고가 키를 참조하는 IAM 정책을 AWS 계정 추가할 수 있도록 허용합니다.

권한 부여

의도한 IAM 보안 주체 또는 사용 기간을 처음부터 알 수 없어 키 또는 IAM 정책에 추가할 수 없는 경우, KMS 키를 사용하도록 위임된 권한입니다. 권한 부여의 한 가지 사용은 AWS 서비스가 KMS 키를 사용하는 방법에 대한 범위 축소 권한을 정의하는 것입니다. 사용자가 직접 서명된 API 호출을 하지 않을 경우 암호화된 데이터에 대해 사용자 대신 비동기 작업을 수행하기 위해 서비스가 사용자의 키를 사용해야 할 수 있습니다.

데이터 키

KMS 키로 보호되는 HSMs에서 생성된 암호화 키입니다.는 승인된 엔터티가 KMS 키로 보호되는 데이터 키를 가져올 수 있도록 AWS KMS 허용합니다. 이 키는 일반 텍스트(암호화되지 않음) 데이터 키로 반환될 수도 있고 암호화된 데이터 키로 반환될 수도 있습니다. 데이터 키는 대칭 또는 비대칭일 수 있습니다(퍼블릭 및 프라이빗 부분이 모두 반환됨).

암호화 텍스트

혼동을 제거하기 위해 고객 암호 텍스트라고 AWS KMS도 하는의 암호화된 출력입니다. 암호화 텍스트에는 복호화 프로세스에서 사용할 KMS 키를 식별하는 추가 정보가 있는 암호화된 데이터가 포함되어 있습니다. 암호화된 데이터 키는 KMS 키를 사용할 때 생성되는 암호화 텍스트의 일반적인 예이지만, 크기가 4KB 미만인 데이터는 KMS 키로 암호화하여 암호화 텍스트를 생성할 수 있습니다.

암호화 컨텍스트

보호된 정보와 연결된 추가 정보의 키 AWS KMS-값 페어 맵입니다.는 인증된 암호화를 AWS KMS 사용하여 데이터 키를 보호합니다. 암호화 컨텍스트는 AWS KMS암호화된 사이퍼텍스트에서 인증된 암호화의 AAD에 통합됩니다. 이 컨텍스트 정보는 선택 사항이며 키(또는 암호화 작업)를 요청할 때 반환되지 않습니다. 하지만 이 컨텍스트 값을 사용할 경우, 복호화 작업을 성공적으로 완료하는 데 필요합니다. 암호화 컨텍스트를 사용하는 목적은 추가적인 인증 정보를 제공하는 것입니다. 이 정보는 정책을 적용하고 AWS CloudTrail 로그에 포함시키는 데 도움이 될 수 있습니다. 예를 들어 {"key name":"satellite uplink key"}라는 키-값 페어를 사용하여 데이터 키의 이름을 지정할 수 있습니다. 이후 키를 사용하면 “키 이름”: “위성 업링크 키”가 포함된 AWS CloudTrail 항목이 생성됩니다. 이 추가 정보는 특정 KMS 키가 사용된 이유를 파악하는 데 유용한 컨텍스트를 제공할 수 있습니다.

퍼블릭 키

비대칭 암호(RSA 또는 타원 곡선)를 사용하는 경우 퍼블릭 키는 퍼블릭-프라이빗 키 페어의 '퍼블릭 구성 요소'입니다. 퍼블릭-프라이빗 키 페어의 소유자에 대한 데이터를 암호화해야 하는 엔터티에 퍼블릭 키를 공유하고 배포할 수 있습니다. 디지털 서명 작업의 경우 서명을 확인하는 데 퍼블릭 키가 사용됩니다.

프라이빗 키

비대칭 암호(RSA 또는 타원 곡선)를 사용하는 경우 프라이빗 키는 퍼블릭-프라이빗 키 페어의 '프라이빗 구성 요소'입니다. 프라이빗 키는 데이터를 복호화하거나 디지털 서명을 생성하는 데 사용됩니다. 대칭 KMS 키와 마찬가지로 프라이빗 키는 HSM에서 암호화됩니다. HSM의 단기 메모리에만 복호화되며 암호화 요청을 처리하는 데 필요한 시간 동안만 복호화됩니다.

AWS KMS 설계 목표

AWS KMS 는 다음 요구 사항을 충족하도록 설계되었습니다.

내구성

암호화 키의 내구성은에서 가장 내구성이 높은 서비스와 같도록 설계되었습니다 AWS. 단일 암호화 키를 사용하여 장기간 누적된 대량의 데이터를 암호화할 수 있습니다.

신뢰성

키 사용은 사용자가 정의하고 관리하는 액세스 제어 정책으로 보호됩니다. 일반 텍스트 KMS 키를 내보내는 메커니즘은 없습니다. 암호화 키의 기밀성은 매우 중요합니다. HSM에서 관리 작업을 수행하려면 쿼럼 기반 액세스 제어에 대한 역할별 액세스 권한을 가진 여러 Amazon 직원이 필요합니다.

짧은 대기 시간과 높은 처리량

AWS KMS 는의 다른 서비스에서 사용하기에 적합한 지연 시간 및 처리량 수준에서 암호화 작업을 제공합니다 AWS.

독립적인 리전

AWS 는 서로 다른 리전에서 데이터 액세스를 제한해야 하는 고객에게 독립적인 리전을 제공합니다. 키 사용은 단일 AWS 리전으로 제한됩니다.

난수의 보안 소스

강력한 암호화를 구현하려면 예측할 수 없는 난수 생성이 중요하므로 AWS KMS 는 고품질의 검증된 난수 소스를 제공합니다.

감사

AWS KMS 는 AWS CloudTrail 로그에서 암호화 키의 사용 및 관리를 기록합니다. AWS CloudTrail 로그를 사용하여 사용자를 대신하여 AWS 서비스의 키 사용을 포함하여 암호화 키 사용을 검사할 수 있습니다.

이러한 목표를 달성하기 위해 AWS KMS 시스템에는 “도메인”을 관리하는 일련의 AWS KMS 연산자 및 서비스 호스트 연산자(총칭하여 “작업자”)가 포함됩니다. 도메인은 리전별로 정의된 AWS KMS 서버, HSMs 및 연산자 집합입니다. 각 AWS KMS 운영자에는 작업을 인증하는 데 사용되는 프라이빗 및 퍼블릭 키 페어가 포함된 하드웨어 토큰이 있습니다. HSM에는 HSM 상태 동기화를 보호하는 암호화 키를 설정하기 위한 추가 프라이빗 및 퍼블릭 키 페어가 있습니다.

이 백서는가 암호화하려는 키 및 기타 데이터를 AWS KMS 보호하는 방법을 보여줍니다. 이 문서에서는 암호화하려는 암호화 키 또는 데이터를 ‘보안 암호’ 또는 ‘보안 암호 구성 요소’라고 합니다.

AWS Key Management Service 파운데이션

이 장의 주제에서는의 암호화 기본 요소와 해당 기본 요소가 사용되는 AWS Key Management Service 위치에 대해 설명합니다. 또한의 기본 요소도 소개합니다 AWS KMS.

주제

암호 프리미티브

AWS KMS 는 구성 가능한 암호화 알고리즘을 사용하므로 시스템이 승인된 알고리즘 또는 모드에서 다른 알고리즘으로 빠르게 마이그레이션할 수 있습니다. 초기 기본 암호화 알고리즘 세트는 보안 속성과 성능을 지원하기 위해 Federal Information Processing Standard(FIPS 승인) 알고리즘에서 선택되었습니다.

엔트로피 및 난수 생성

AWS KMS 키 생성은 AWS KMS HSMs에서 수행됩니다. HSM은 AES-256을 사용한 NIST SP800-90A Deterministic Random Bit Generator(DRBG) CTR_DRBG를 사용하는 하이브리드 난수 생성기를 구현합니다. 384비트의 엔트로피가 포함된 비결정적 임의 비트 생성기로 시드되고 추가 엔트로피가 업데이트되어 암호화 구성 요소를 호출 할 때마다 예측 저항을 제공합니다.

대칭 키 작업(암호화만 해당)

HSM에서 사용되는 모든 대칭 키 암호화 명령은 256비트 키를 사용한 Galois Counter Mode(GCM)의 고급 암호화 표준(AES)을 사용합니다. 복호화하기 위한 유사한 호출에는 역 함수가 사용됩니다.

AES-GCM은 인증된 암호화 체계입니다. 암호화 텍스트를 생성하기 위해 일반 텍스트를 암호화하는 것 외에 암호문과 인증이 필요한 추가 데이터(추가로 인증된 데이터 또는 AAD)를 통해 인증 태그를 계산합니다. 이 인증 태그는 데이터가 취합된 소스로부터 얻어진 것이며 암호화 텍스트와 AAD가 수정되지 않았음을 확인하는 데 유용합니다.

자주, 특히 데이터 키 암호화를 참조할 때 설명에 AAD를 포함하지 AWS 않습니다. 암호화할 구조가 암호화될 일반 텍스트와 보호할 일반 텍스트 AAD로 분할되는 이 같은 경우에는 주변 텍스트에 의해 암시됩니다.

AWS KMS 는에 의존하여 키 구성 요소를 생성하는 AWS KMS key 대신 키 구성 요소를 AWS KMS 로 가져올 수 있는 옵션을 제공합니다. 가져온이 키 구성 요소는 RSAES-OAEP 또는 RSAES-PKCS1-v1_5를 사용하여 암호화하여 AWS KMS HSM으로 전송하는 동안 키를 보호할 수 있습니다. RSA 키 페어는 AWS KMS HSM에 생성됩니다. 가져온 키 구성 요소는 서비스에 저장되기 전에 AWS KMS HSM에서 복호화되고 AES-GCM에서 다시 암호화됩니다.

비대칭 키 작업(암호화, 디지털 서명 및 서명 확인)

AWS KMS 는 암호화 및 디지털 서명 작업 모두에 비대칭 키 작업 사용을 지원합니다. 비대칭 키 작업은 암호화 및 복호화 또는 서명 및 서명 확인에 사용할 수 있는 수학적으로 관련된 퍼블릭 키 및 프라이빗 키 페어를 사용하지만 둘 다 사용할 수는 없습니다. 프라이빗 키는 암호화 AWS KMS 되지 않은 상태로 두지 않습니다. AWS KMS API 작업을 호출 AWS KMS 하여 내에서 퍼블릭 키를 사용하거나 퍼블릭 키를 다운로드하여 외부에서 사용할 수 있습니다 AWS KMS.

AWS KMS 는 세 가지 유형의 비대칭 암호를 지원합니다.

- RSA-OAEP(암호화용) 및 RSA-PSS와 RSA-PKCS-#1-v1_5(서명 및 확인용) - 다양한 보안 요구 사항에 맞게 2,048, 3,072, 4,096의 RSA 키 길이(비트 단위)를 지원합니다.
- 타원 곡선(ECC) - 서명 및 확인에만 사용됩니다. ECC 곡선: NIST P256, P384, P521, SECP 256k1을 지원합니다.
- Post Quantum Cryptography - 양자 컴퓨팅에 내성이 있는 새로운 퍼블릭 키 암호화 알고리즘입니다. ML_DSA_44, ML_DSA_65 및 ML_DSA_87 키 크기를 사용하는 NIST FIPS 204 Module-Lattice Digital Signature Algorithm(ML-DSA)을 지원합니다.

키 유도 함수

키 유도 함수는 초기 보안 아호 또는 키에서 추가 키를 추출하는 데 사용됩니다. AWS KMS 는 키 유도 함수(KDF)를 사용하여 AWS KMS key를 사용한 모든 암호화의 호출별 키를 추출합니다. 모든 KDF 작업에는 HMAC [FIPS197]과 SHA256 [FIPS180]을 사용한 카운터 모드의 KDF가 사용됩니다. 추출된 256비트 키는 고객 데이터 및 키를 암호화하거나 복호화하는 데 AES-GCM과 함께 사용됩니다.

AWS KMS 디지털 서명의 내부 사용

디지털 서명은 AWS KMS 엔터티 간에 명령 및 통신을 인증하는 데에도 사용됩니다. 모든 서비스 엔터티에는 타원 곡선 디지털 서명 알고리즘(ECDSA) 키 페어가 있습니다. 이 키 페어는 Use of Elliptic Curve Cryptography(ECC) Algorithms in Cryptographic Message Syntax(CMS) 및 X9.62-2005: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm(ECDSA)에 정의된 대로 ECDSA를 수행합니다. 엔터티는 SHA384라고 하는 Federal

Information Processing Standards Publications, FIPS PUB 180-4에 정의된 보안 해시 알고리즘을 사용합니다. 모든 키는 secp384r1(NIST-P384) 곡선에 생성됩니다.

봉투 암호화

많은 암호화 시스템에 사용되는 기본 구조는 봉투 암호화입니다. 봉투 암호화는 2개 이상의 암호화 키를 사용하여 메시지를 보호합니다. 일반적으로 키 중 하나는 장기 정적 키 k에서 추출되며, 다른 하나는 메시지를 암호화하기 위해 생성되는 메시지별 키 msgKey입니다. 봉투는 ciphertext = Encrypt(msgKey, message)와 같이 메시지를 암호화하여 구성됩니다. 그런 다음 encKey = Encrypt(k, msgKey)와 같이 메시지 키가 장기 정적 키로 암호화됩니다. 마지막으로, (encKey, ciphertext)라는 두 값이 단일 구조 또는 봉투 암호화 메시지로 패키징됩니다.

k에 대한 액세스 권한이 있는 수신자는 암호화된 키를 먼저 복호화한 다음 메시지를 복호화하여 봉투로 암호화된 메시지를 열 수 있습니다.

AWS KMS 는 이러한 장기 정적 키를 관리하고 데이터의 봉투 암호화 프로세스를 자동화하는 기능을 제공합니다.

AWS Encryption SDK는 AWS KMS 서비스 내에서 제공되는 암호화 기능 외에도 클라이언트 측 봉투 암호화 라이브러리를 제공합니다. 이 라이브러리를 사용하여 데이터 및 해당 데이터를 암호화하는 데 사용되는 암호화 키를 보호할 수 있습니다.

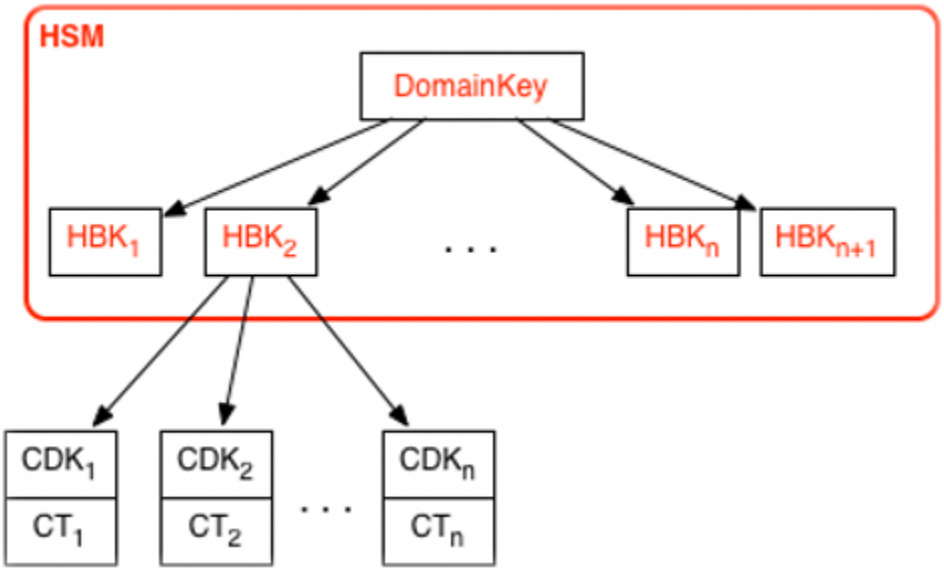
AWS KMS key 계층 구조

키 계층 구조는 최상위 논리적 키인 로 시작합니다 AWS KMS key. KMS 키는 최상위 키 구성 요소의 컨테이너를 나타내며, Amazon 리소스 이름(ARN)을 사용하여 AWS 서비스 네임스페이스에 고유하게 정의됩니다. ARN에는 고유하게 생성된 키 식별자인 키 ID가 포함되어 있습니다. KMS 키는 사용자가 시작한 요청을 기반으로 생성됩니다 AWS KMS. 수신 시는 KMS 키 컨테이너에 배치할 초기 HSM 백업 키(HBK) 생성을 AWS KMS 요청합니다. HBK는 도메인의 HSM에 생성되며 HSM에서 일반 텍스트로 내보낼 수 없도록 설계되었습니다. 대신 HBK는 HSM 관리 도메인 키로 암호화되어 내보내집니다. 이러한 내보낸 HBK를 내보낸 키 토큰(EKT)이라고 합니다.

EKT는 내구성이 높고 대기 시간이 짧은 스토리지로 내보내집니다. 예를 들어 논리적 KMS 키에 대한 ARN 수신한다고 가정합니다. 이는 키 계층 구조 또는 암호화 컨텍스트의 최상위를 나타냅니다. 계정 내에 여러 KMS 키를 생성하고 다른 AWS 명명된 리소스와 마찬가지로 KMS 키에 정책을 설정할 수 있습니다.

특정 KMS 키의 계층 구조 내에서 HBK는 KMS 키의 한 버전이라고 할 수 있습니다. KMS 키를를 통해 교체하려는 경우 AWS KMS새 HBK가 생성되고 KMS 키에 대한 활성 HBK로 KMS 키와 연결됩니다.

이전 HBK는 보존되며 이전에 보호된 데이터를 복호화하고 확인하는 데 사용할 수 있습니다. 단, 새 정보를 보호하는 데에는 활성 암호화 키만 사용할 수 있습니다.



AWS KMS 를 통해 KMS 키를 사용하여 정보를 직접 보호하도록 요청하거나 KMS 키로 보호되는 추가 HSM 생성 키를 요청할 수 있습니다. 이러한 키를 고객 데이터 키(CDK)라고 합니다. CDK는 암호화 텍스트(CT), 일반 텍스트 또는 두 가지 형식 모두로 암호화된 상태로 반환될 수 있습니다. KMS 키(고객 제공 데이터 또는 HSM 생성 키)로 암호화된 모든 객체는 직접 호출을 통해 HSM에서만 복호화할 수 있습니다 AWS KMS.

반환된 암호 텍스트 또는 해독된 페이로드는 내에 저장되지 않습니다 AWS KMS. 이 정보는 AWS KMS에 대한 TLS 연결을 통해 사용자에게 반환됩니다. 이는 사용자를 대신하여 AWS 서비스에서 수행한 호출에도 적용됩니다.

다음 표에는 키 계층 구조 및 특정 키 속성이 나와 있습니다.

키	설명	수명 주기
도메인 키	HSM 메모리에만 있는 256비트 AES-GCM 키로, KMS 키의 버전, 즉 HSM 백업 키를 래핑하는 데 사용됩니다.	매일 교체됨[1]
HSM 백업 키	256비트 대칭 키 또는 RSA 또는 타원 곡선 개인 키로, 고객 데이터 및 키를 보호하는 데 사용되며 도메인 키로 암호화됩니다. 하나 이상의 HSM 백업	매년 교체됨[2](선택적 구성)

키	설명	수명 주기
	키가 KMS 키를 구성하며 keyId로 표시됩니다.	
추출된 암호화 키	HSM 메모리에만 있는 256비트 AES-GCM 키로, 고객 데이터 및 키를 암호화하는 데 사용됩니다. 각 암호화에 대해 HBK에서 추출됩니다.	암호화당 한 번 사용되며 복호화할 때 다시 생성됩니다.
고객 데이터 키	HSM에서 일반 텍스트 및 암호화 텍스트 형식으로 내보낸 사용자 정의 대칭 또는 비대칭 키입니다. HSM 백업 키로 암호화되며, TLS 채널을 통해 권한 있는 사용자에게 반환됩니다.	교체 및 애플리케이션에 의한 통제된 사용

[1] AWS KMS 도메인 관리 및 구성 작업을 고려하기 위해 도메인 키 교체를 최대 매주 완화할 수 있습니다.

[2] 사용자를 대신하여 AWS KMS 에서 AWS 관리형 키 생성하고 관리하는 기본값은 매년 자동으로 교체됩니다.

AWS KMS 사용 사례

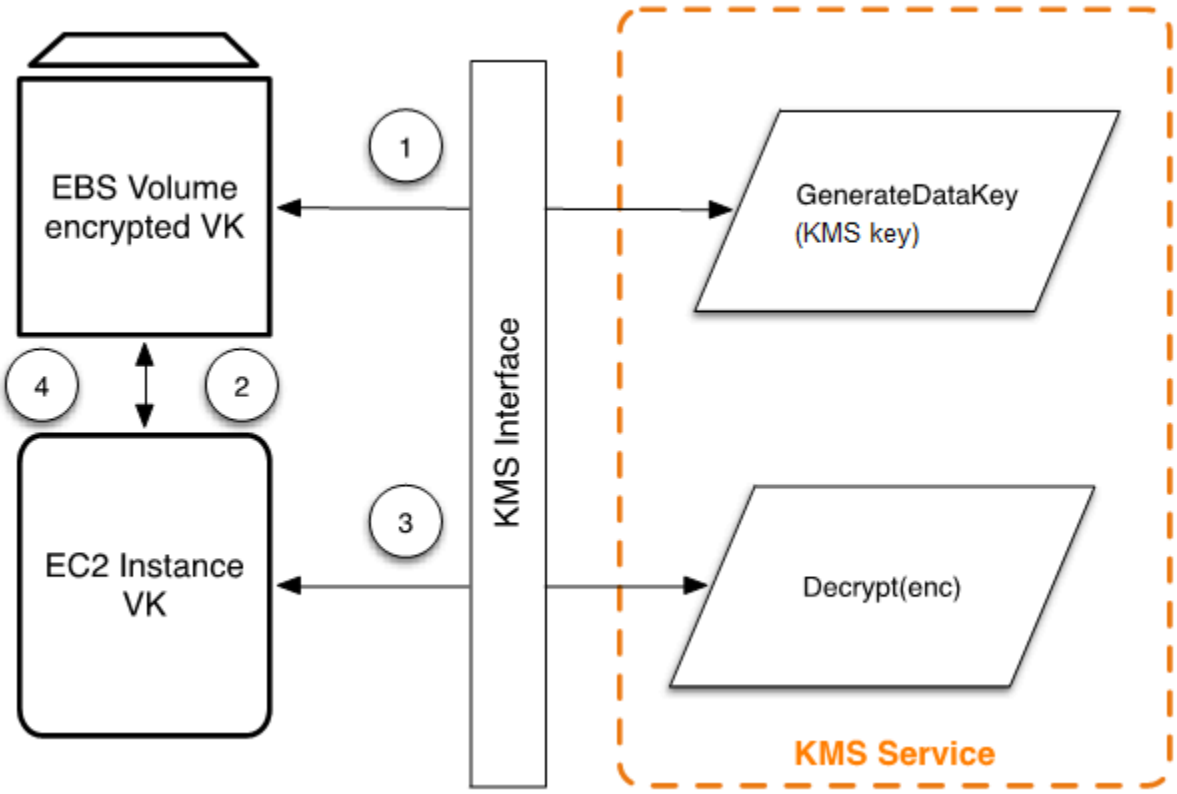
사용 사례는 최대한 활용하는 데 도움이 될 수 있습니다 AWS Key Management Service. 첫 번째는가 Amazon Elastic Block Store(Amazon EBS) 볼륨 AWS KMS keys 에서를 사용하여 서버 측 암호화를 AWS KMS 수행하는 방법을 보여줍니다. 두 번째는 봉투 암호화를 사용하여 콘텐츠를 보호하는 방법을 보여주는 클라이언트 측 애플리케이션입니다 AWS KMS.

주제

- Amazon EBS 볼륨 암호화
- 클라이언트측 암호화

Amazon EBS 볼륨 암호화

Amazon EBS는 볼륨 암호화 기능을 제공합니다. 각 볼륨은 AES-256-XTS를 사용하여 암호화됩니다. 여기에는 2개의 256비트 볼륨 키가 필요하며, 이는 512비트 볼륨 키 하나와 같습니다. 볼륨 키는 계정의 KMS 키로 암호화됩니다. Amazon EBS가 볼륨을 자동으로 암호화하려면 계정의 KMS 키로 볼륨 키(VK)를 생성할 수 있는 액세스 권한이 있어야 합니다. 이 권한을 부여하려면 데이터 키를 생성하고 이러한 볼륨 키를 암호화 및 복호화하도록 KMS 키에 Amazon EBS에 대한 권한을 제공합니다. 이제 Amazon EBS는를 KMS 키 AWS KMS 와 함께 사용하여 AWS KMS 암호화된 볼륨 키를 생성합니다.



다음 워크플로는 Amazon EBS 볼륨에 쓰고 있는 데이터를 암호화합니다.

1. Amazon EBS는 TLS 세션을 AWS KMS 통해 KMS 키로 암호화된 볼륨 키를 가져와 볼륨 메타데이터와 함께 암호화된 키를 저장합니다.
2. Amazon EBS 볼륨이 탑재되면 암호화된 볼륨 키가 검색됩니다.
3. TLS를 AWS KMS 통한 호출은 암호화된 볼륨 키를 복호화하기 위해 이루어집니다.는 KMS 키를 AWS KMS 식별하고 플릿의 HSM에 암호화된 볼륨 키를 복호화하도록 내부 요청을 합니다. AWS KMS 그런 다음는 TLS 세션을 통해 인스턴스가 포함된 Amazon Elastic Compute Cloud(Amazon EC2) 호스트로 볼륨 키를 반환합니다.
4. 이 볼륨 키는 연결된 Amazon EBS 볼륨에서 주고받는 모든 데이터를 암호화하고 복호화하는 데 사용됩니다. Amazon EBS는 메모리의 볼륨 키를 더 이상 사용할 수 없는 경우에 대비하여 나중에 사용할 수 있도록 암호화된 볼륨 키를 유지합니다.

KMS 키로 Amazon EBS 볼륨을 암호화하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 Amazon Elastic Block Store에서 AWS KMS사용 방법 섹션과 Amazon EC2 사용 설명서 및 Amazon EC2 사용 설명서의 Amazon EBS 암호화 섹션을 참조하세요.

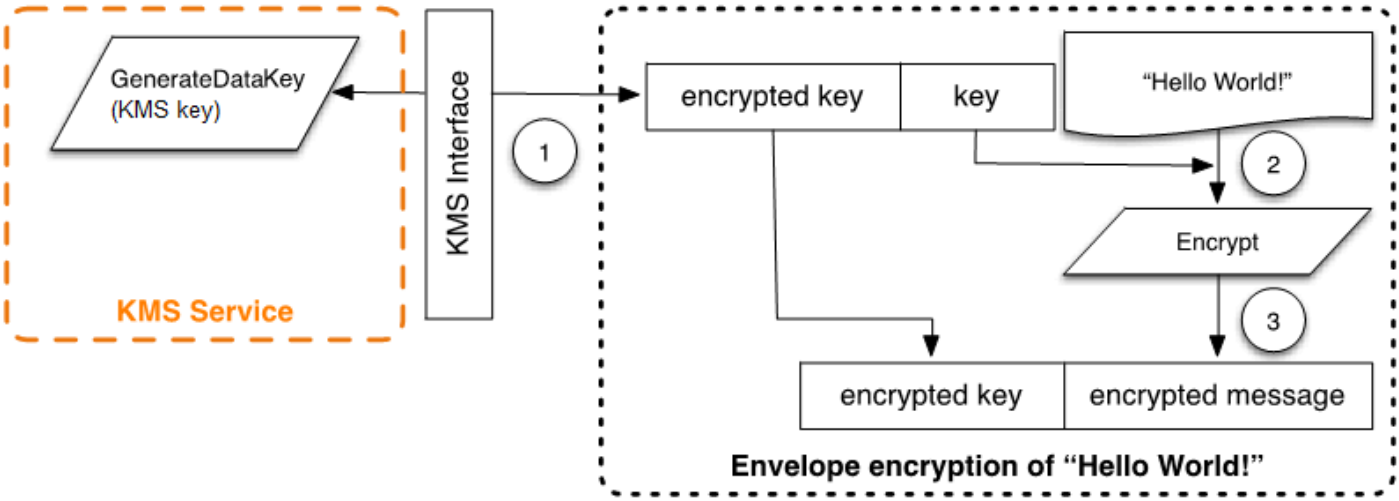
클라이언트측 암호화

AWS Encryption SDK에는 KMS 키를 사용하여 봉투 암호화를 수행하기 위한 API 작업이 포함되어 있습니다. 전체 권장 사항 및 사용 세부 정보는 관련 설명서를 참조하세요. 클라이언트 애플리케이션은를 사용하여 봉투 암호화 AWS Encryption SDK 를 수행할 수 있습니다 AWS KMS.

```
// Instantiate the SDK
final AwsCrypto crypto = new AwsCrypto();
// Set up the KmsMasterKeyProvider backed by the default credentials
final KmsMasterKeyProvider prov = new KmsMasterKeyProvider(keyId);
// Do the encryption
final byte[] ciphertext = crypto.encryptData(prov, message);
```

클라이언트 애플리케이션은 다음 단계를 실행할 수 있습니다.

1. KMS 키를 사용하여 새 데이터 키에 대한 요청을 보냅니다. 암호화된 데이터 키와 일반 텍스트 버전의 데이터 키가 반환됩니다.
2. 내에서 AWS Encryption SDK일반 텍스트 데이터 키는 메시지를 암호화하는 데 사용됩니다. 그런 다음 일반 텍스트 데이터 키가 메모리에서 삭제됩니다.
3. 암호화된 데이터 키와 암호화된 메시지가 단일 암호화 텍스트 바이트 배열로 결합됩니다.

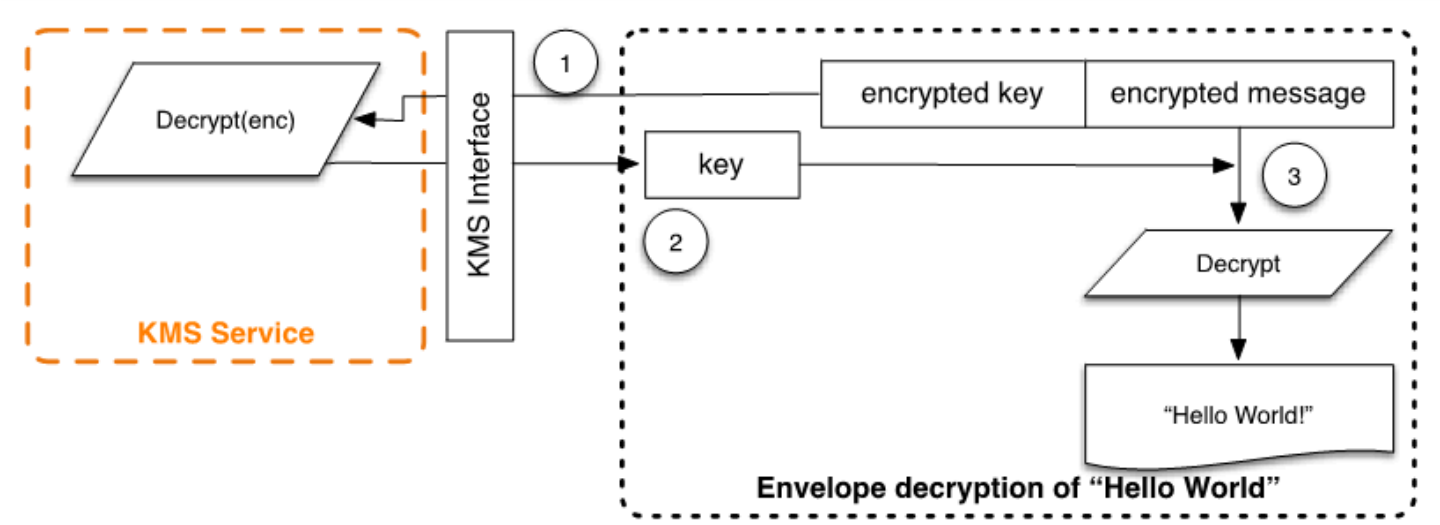


복호화 기능을 사용하여 봉투로 암호화된 메시지를 복호화하면 원래의 암호화된 메시지를 얻을 수 있습니다.

```
final AwsCrypto crypto = new AwsCrypto();
```

```
final KmsMasterKeyProvider prov = new KmsMasterKeyProvider(keyId);
// Decrypt the data
final CryptoResult<byte[], KmsMasterKey> res = crypto.decryptData(prov, ciphertext);
// We need to check the KMS key to ensure that the
// assumed key was used
if (!res.getMasterKeyIds().get(0).equals(keyId)) {
     throw new IllegalStateException("Wrong key id!");
}
byte[] plaintext = res.getResult();
```

1. 는 봉투로 암호화된 메시지를 AWS Encryption SDK 구문 분석하여 암호화된 데이터 키를 얻고에 데이터 키를 복호화 AWS KMS 하도록 요청합니다.
2. 는 일반 텍스트 데이터 키를 AWS Encryption SDK 수신합니다 AWS KMS.
3. 그런 다음 데이터 키를 사용하여 메시지를 복호화하고 최초의 일반 텍스트를 반환합니다.

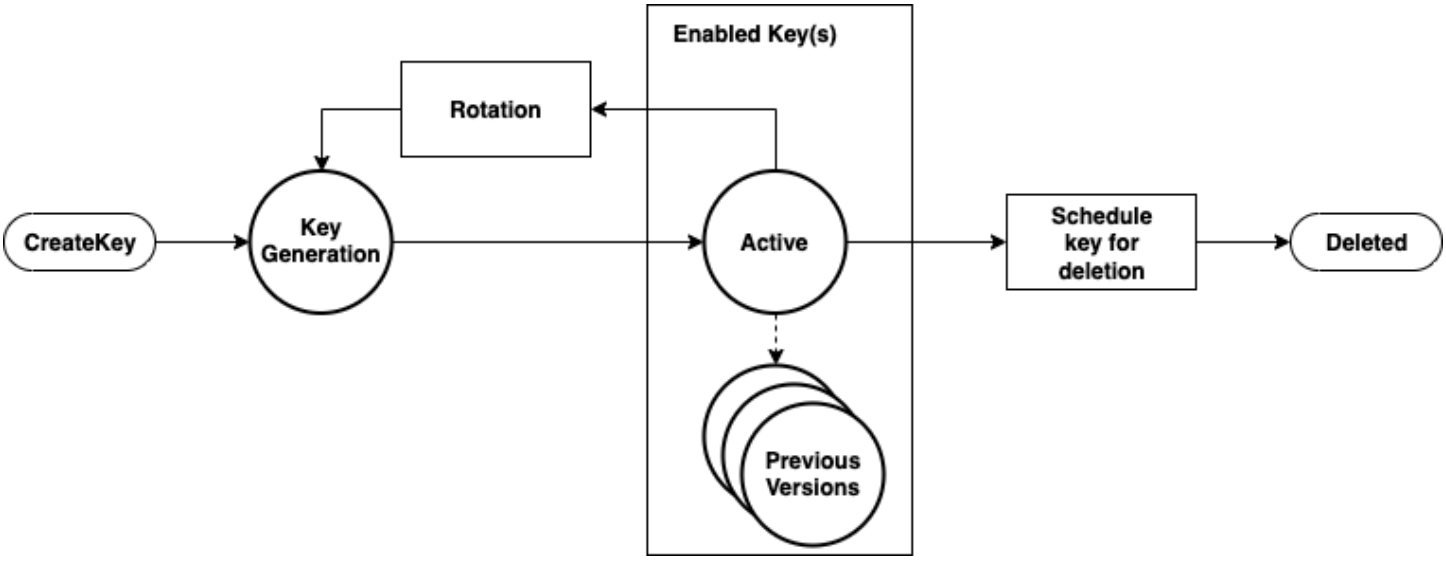


작업 AWS KMS keys

는 하나 이상의 HSM(하드웨어 보안 모듈) 지원 키(HBKs)를 참조할 수 있는 논리적 키를 AWS KMS key 나타냅니다. 이 주제에서는 KMS 키를 생성하고 키 구성 요소를 가져오는 방법과 KMS 키를 활성화, 비활성화, 교체 및 삭제하는 방법에 관해 설명합니다.

> **Note**
>
> AWS KMS 는 고객 마스터 키(CMK)라는 용어를 AWS KMS key 및 KMS 키로 대체합니다. 단, 개념은 바뀌지 않았습니다. 주요 변경 사항을 방지하기 위해 AWS KMS 는이 용어의 몇 가지 변형을 유지합니다.

이 장에서는 다음 이미지에서와 같이 KMS 키 생성부터 삭제까지의 KMS 키 수명 주기에 관해 설명합니다.



주제

- CreateKey 호출
- 키 구성 요소 가져오기
- 키 활성화 및 비활성화
- 키 삭제
- 키 구성 요소 교체

CreateKey 호출

CreateKey API 호출의 결과로 AWS KMS key 가 생성됩니다.

다음은 몇 가지 CreateKey 요청 구문입니다.

```
{
  "Description": "string",
  "KeySpec": "string",
  "KeyUsage": "string",
  "Origin": "string";
  "Policy": "string"
}
```

요청은 JSON 형식으로 다음 데이터를 받습니다.

설명

(선택 사항) 키에 대한 설명입니다. 키가 태스크에 적합한지 여부를 결정하는 데 도움이 되는 설명을 선택하는 것이 좋습니다.

KeySpec

생성할 KMS 키 유형을 지정합니다. 기본값인 SYMMETRIC_DEFAULT를 사용하면 대칭 암호화 KMS 키가 생성됩니다. 이 파라미터는 대칭 암호화 키의 경우 선택 사항이며 다른 모든 키 사양에 필요합니다.

KeyUsage

키 사용을 지정합니다. 유효한 값은 ENCRYPT_DECRYPT, SIGN_VERIFY또는 GENERATE_VERIFY_MAC입니다. 기본값은 ENCRYPT_DECRYPT입니다. 이 파라미터는 대칭 암호화 키의 경우 선택 사항이며 다른 모든 키 사양에 필요합니다.

Origin

(선택 사항) KMS 키용 키 구성 요소의 소스를 지정합니다. 기본값은 이며AWS_KMS, 이는가 KMS 키의 키 구성 요소를 AWS KMS 생성하고 관리함을 나타냅니다. 다른 유효한 값에는 가져온 키 구성 요소에 대한 키 구성 요소 없이 생성된 KMS 키를 EXTERNAL나타내고 사용자가 제어하는 AWS CloudHSM 클러스터가 지원하는 사용자 지정 키 스토어에 KMS 키를 AWS_CLOUDHSM 생성하는가 포함됩니다.

정책

(선택 사항) 키에 연결할 정책입니다. 정책을 생략하면 AWS KMS 권한이 있는 루트 계정 및 IAM 주체가 키를 관리하도록 허용하는 기본 정책(다음)을 사용하여 키가 생성됩니다.

정책에 대한 자세한 내용은 AWS KMS의 키 정책 및 AWS Key Management Service 개발자 가이드의 기본 키 정책을 참조하세요.

`CreateKey` 요청이 키 ARN이 포함된 응답을 반환합니다.

```
arn:<partition>:kms:<region>:<account-id>:key/<key-id>
```

`Origin`이 `AWS_KMS`인 경우 ARN이 생성된 후 인증된 세션을 통해 AWS KMS HSM을 요청하여 HSM(하드웨어 보안 모듈) 백업 키(HBK)를 프로비저닝합니다. HBK는 KMS 키의 키 ID와 연결된 256비트 키입니다. HSM에서만 생성할 수 있으며 일반 텍스트로 HSM 경계 외부로 내보내지 않도록 설계되었습니다. HBK는 현재 도메인 키 DK_0를 사용하여 암호화됩니다. 이러한 암호화된 HBK를 EKT(암호화된 키 토큰)라고 합니다. HSM은 다양한 키 래핑 방법을 사용하도록 구성할 수 있지만 현재 구현에서는 인증된 암호화 체계인 Galois Counter Mode(GCM)의 AES-256을 사용합니다. 이 인증된 암호화 모드를 사용하면 일반 텍스트로 내보낸 일부 키 토큰 메타데이터를 보호할 수 있습니다.

이는 다음과 같이 표시됩니다.

$$EKT = Encrypt(DK_0, HBK)$$

KMS 키와 후속 HBK에는 KMS 키에 설정된 권한 부여 정책과 연결된 HBK에 대한 암호화 보호라는 두 가지 기본 보호 형식이 제공됩니다. 나머지 섹션에서는에 있는 관리 함수의 암호화 보호 및 보안에 대해 설명합니다 AWS KMS.

ARN 외에, 키의 별칭을 만들어 사용자에게 익숙한 이름을 생성하고 KMS 키와 연결할 수 있습니다. 별칭이 KMS 키와 연결되면 암호화 작업에서 KMS 키를 식별하기 위해 별칭을 사용할 수 있습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 별칭 사용을 참조하세요.

여러 수준의 권한 부여가 KMS 키 사용을 둘러싸고 있습니다.는 암호화된 콘텐츠와 KMS 키 간에 별도의 권한 부여 정책을 AWS KMS 활성화합니다. 예를 들어 AWS KMS 봉투 암호화 Amazon Simple Storage Service(Amazon S3) 객체는 Amazon S3 버킷의 정책을 상속합니다. 그러나 필요한 암호화 키에 대한 액세스는 KMS 키의 액세스 정책에 따라 결정됩니다. KMS 키 권한 부여에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 AWS KMS에 대한 인증 및 액세스 제어를 참조하세요.

키 구성 요소 가져오기

AWS KMS 는 HBK에 사용되는 암호화 구성 요소를 가져오는 메커니즘을 제공합니다. CreateKey 호출에 설명된 대로, Origin을 EXTERNAL로 설정하여 CreateKey 명령을 사용할 경우 기본 HBK가 없는 논리적 KMS 키가 생성됩니다. ImportKeyMaterial API 호출을 사용하여 암호화 구성 요소를 가져와야 합니다. 이 기능을 사용하여 암호화 구성 요소의 키 생성 및 내구성을 제어할 수 있습니다. 이 기능을 사용할 경우 사용자 환경에서 이러한 키의 처리 및 내구성에 상당한 주의를 기울이는 것이 좋습니다. 키 구성 요소 가져오기에 대한 자세한 내용 및 권장 사항은 AWS Key Management Service 개발자 가이드에서 키 구성 요소 가져오기를 참조하세요.

ImportKeyMaterial 호출

ImportKeyMaterial 요청은 HBK에 필요한 암호화 구성 요소를 가져옵니다. 암호화 구성 요소는 256비트 대칭 키여야 합니다. 이 키는 WrappingAlgorithm에 지정된 알고리즘을 사용하여 최근 GetParametersForImport 요청에서 반환된 퍼블릭 키로 암호화되어야 합니다.

ImportKeyMaterial 요청은 다음 인수를 받습니다.

```
{
  "EncryptedKeyMaterial": blob,
  "ExpirationModel": "string",
  "ImportToken": blob,
  "KeyId": "string",
  "ValidTo": number
}
```

EncryptedKeyMaterial

GetParametersForImport 요청에 지정된 래핑 알고리즘을 사용하여 요청에 반환된 공개 키로 암호화된 가져온 키 구성 요소입니다.

ExpirationModel

키 구성 요소의 만료 여부를 지정합니다. 이 값이 KEY_MATERIAL_EXPIRES이면, ValidTo 파라미터에 만료 날짜가 포함되어야 합니다. 이 값이 KEY_MATERIAL_DOES_NOT_EXPIRE이면, ValidTo 파라미터를 포함해서는 안 됩니다. 유효 값은 "KEY_MATERIAL_EXPIRES" 및 "KEY_MATERIAL_DOES_NOT_EXPIRE"입니다.

ImportToken

퍼블릭 키를 제공한 동일한 GetParametersForImport 요청이 반환한 가져오기 토큰입니다.

KeyId

가져온 키 구성 요소와 연결될 KMS 키입니다. KMS 키의 `Origin`은 `EXTERNAL`이어야 합니다.

동일한 가져온 키 구성 요소를 삭제하고 지정된 KMS 키로 다시 가져올 수 있지만, KMS 키를 다른 키 구성 요소로 가져오거나 연결할 수 없습니다.

ValidTo

(선택 사항) 가져온 키 구성 요소가 만료되는 시간입니다. 키 구성 요소가 만료되면, AWS KMS 가 키 구성 요소를 삭제하고 KMS 키를 사용할 수 없게 됩니다. 이 파라미터는 `ExpirationModel` 값이 `KEY_MATERIAL_EXPIRES`인 경우 필수입니다. 그렇지 않은 경우 이 값은 잘못되었습니다.

요청이 성공하면 KMS 키가 제공된 경우 지정된 만료 날짜 AWS KMS 까지 KMS 키를 사용할 수 있습니다. 가져온 키 구성 요소가 만료되면 AWS KMS 스토리지 계층에서 EKT가 삭제됩니다.

키 활성화 및 비활성화

KMS 키를 비활성화하면 암호화 작업에 KMS 키가 사용되지 않습니다. KMS 키와 연결된 모든 HBK의 사용 기능이 중지됩니다. 키를 활성화하면 HBK 및 KMS 키 사용이 복원됩니다. 사용 설정과 사용 중지는 KMS 키의 ID 또는 키 ARN만 받는 단순 요청입니다.

키 삭제

권한 있는 사용자는 `ScheduleKeyDeletion` API를 사용하여 KMS 키 및 연결된 모든 HBK의 삭제를 예약할 수 있습니다. 이는 본질적으로 파괴적인 작업이며에서 키를 삭제할 때 주의해야 합니다 AWS KMS.는 KMS 키를 삭제할 때 최소 7일의 대기 시간을 AWS KMS 적용합니다. 이 대기 기간 동안 키 상태가 삭제 보류 중으로 바뀌고 키가 비활성화됩니다. 암호화 작업에 이 키를 사용하는 모든 호출이 실패합니다. ScheduleKeyDeletion은 다음 인수를 받습니다.

```
{
  "KeyId": "string",
  "PendingWindowInDays": number
}
```

KeyId

삭제할 KMS 키의 고유 식별자입니다. 이 값을 지정하려면 KMS 키의 고유 키 ID 또는 키 ARN을 사용합니다.

PendingWindowInDays

(선택 사항) 대기 기간(일수)입니다. 이 값은 선택 사항입니다. 범위는 7~30일이며 기본값은 30일입니다. 대기 기간이 끝나면가 KMS 키와 연결된 모든 HBK를 AWS KMS 삭제합니다. HBKs

키 구성 요소 교체

권한 있는 사용자는 고객 관리형 KMS 키를 매년 자동으로 교체할 수 있습니다. AWS 관리형 키 는 항상 매년 교체됩니다.

KMS 키가 교체되면 새로운 HBK가 생성되고, 모든 새로운 암호화 요청에 대한 키 구성 요소의 현재 버전으로 표시됩니다. 모든 이전 버전의 HBK는 이 HBK 버전을 사용하여 암호화된 사이퍼텍스트를 해독하는 데 영구적으로 사용할 수 있습니다. AWS KMS 는 KMS 키로 암호화된 사이퍼텍스트를 저장하지 않으므로 이전 HBK로 암호화된 사이퍼텍스트를 교체하려면 HBK가 복호화해야 합니다. `ReEncrypt` API를 사용하여 KMS 키용 새 HBK로 또는 일반 텍스트를 노출하지 않고 다른 KMS 키로 사이퍼텍스트를 다시 암호화할 수 있습니다.

키 교체 활성화 및 비활성화에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서에서 AWS KMS 키 교체를 참조하세요.

고객 데이터 작업

KMS 키를 설정한 후에는 암호화 작업을 수행하는 데 사용할 수 있습니다. 데이터가 KMS 키로 암호화될 때마다 그 결과 객체로 고객 암호화 텍스트가 생성됩니다. 암호화 텍스트에는 인증된 암호화 체계에 의해 보호되는 암호화되지 않은 헤더(또는 일반 텍스트) 부분과 암호화된 부분의 두 섹션이 있습니다. 일반 텍스트 부분에는 HBK 식별자(HBKID)가 포함됩니다. 사이퍼텍스트 값의이 두 개의 변경 불가능한 필드는가 향후 객체를 복호화할 AWS KMS 수 있도록 하는 데 도움이 됩니다.

주제

- 데이터 키 생성
- 암호화
- Decrypt
- 암호화된 객체 다시 암호화

데이터 키 생성

권한 있는 사용자는 GenerateDataKey API와 관련 API를 사용하여 특정 유형의 데이터 키 또는 임의의 길이의 임의의 키를 요청합니다. 이 주제에서는 이 API 작업의 단순화된 보기를 제공합니다. 자세한 내용은 AWS Key Management Service API 참조의 GenerateDataKey API를 참조하세요.

- GenerateDataKey
- GenerateDataKeyWithoutPlaintext
- GenerateDataKeyPair
- GenerateDataKeyPairWithoutPlaintext

다음은 `GenerateDataKey` 요청 구문입니다.

```
{
 “EncryptionContext”: {“string” : “string”},
 “GrantTokens”: [“string”],
 “KeyId”: “string”,
 “NumberOfBytes”: “number”
}
```

요청은 JSON 형식으로 다음 데이터를 받습니다.

KeyId

데이터 키를 암호화하는 데 사용되는 키의 식별자입니다. 이 값은 대칭 암호화 KMS 키를 식별해야 합니다.

이 파라미터는 필수 사항입니다.

NumberOfBytes

생성할 바이트 수를 나타내는 정수입니다. 이 파라미터는 필수 사항입니다.

호출자는 `KeySpec` 또는 `NumberOfBytes` 중 하나를 제공해야 하며 둘 다 지정할 수는 없습니다.

EncryptionContext

(선택 사항) 암호화 및 복호화 프로세스 중에 인증할 추가 데이터가 포함된 이름-값 페어입니다.

GrantTokens

(선택 사항) 키를 생성하거나 사용할 권한을 제공하는 권한 부여를 나타내는 권한 부여 토큰 목록입니다. 권한 부여 및 권한 부여 토큰에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드에서 AWS KMS의 인증 및 액세스 제어를 참조하세요.

명령을 인증한 후 AWS KMS는 KMS 키와 연결된 현재 활성 EKT를 획득합니다. 제공된 요청 및 암호화 컨텍스트와 함께 EKT를 AWS KMS 호스트와 도메인의 HSM 간의 보호된 세션을 통해 HSM에 전달합니다.

HSM은 다음 작업을 수행합니다.

1. 요청된 보안 암호 구성 요소를 생성하고 휘발성 메모리에 보관합니다.
2. 활성 HBK = Decrypt(DK_i, EKT)를 가져오기 위해 요청에 정의된 KMS 키의 키 ID와 일치하는 EKT를 복호화합니다.
3. 임의의 임시 N을 생성합니다.
4. HBK 및 N에서 256비트 AES-GCM 파생 암호화 키 K를 생성합니다.
5. 보안 암호 구성 요소 ciphertext = Encrypt(K, context, secret)를 암호화합니다.

`GenerateDataKey`는 AWS KMS 호스트와 HSM 간의 보안 채널을 통해 일반 텍스트 보안 구성 요소와 사이퍼텍스트를 반환합니다. AWS KMS 그런 다음는 TLS 세션을 통해 이를 전송합니다. AWS KMS는 일반 텍스트 또는 사이퍼텍스트를 유지하지 않습니다. KMS 키를 사용하기 위한 암호화 텍스트, 암호화 컨텍스트 및 권한 부여를 소유한 경우가 아니면, 기본 보안 암호가 반환되지 않습니다.

응답 구문은 다음과 같습니다.

```
{
 "CiphertextBlob": "blob",
 "KeyId": "string",
 "Plaintext": "blob"
}
```

데이터 키의 관리는 애플리케이션 개발자로서 사용자가 맡습니다. AWS KMS 데이터 키(데이터 키 페어는 아님)를 사용한 클라이언트 측 암호화 모범 사례를 위해를 사용할 수 있습니다AWS Encryption SDK.

데이터 키 교체 빈도는 제약이 없습니다. 또한 `ReEncrypt` API 작업을 통해 데이터 키를 다른 KMS 키 또는 교체된 KMS 키로 다시 암호화할 수 있습니다. 자세한 내용은 AWS Key Management Service API 참조의 ReEncrypt를 참조하세요.

암호화

의 기본 함수 AWS KMS 는 KMS 키로 객체를 암호화하는 것입니다. 설계상, AWS KMS 는 HSM에서 대기 시간이 짧은 암호화 작업을 제공합니다. 따라서 암호화 함수에 대한 직접 호출로 암호화할 수 있는 일반 텍스트의 양은 4KB로 제한됩니다. 는 더 큰 메시지를 암호화하는 데 사용할 AWS Encryption SDK 수 있습니다. 명령을 인증한 AWS KMS후는 KMS 키와 관련된 현재 활성 EKT를 획득합니다. EKT를 일반 텍스트 및 암호화 컨텍스트와 함께 해당 리전에서 사용 가능한 모든 HSM에 전달합니다. 이는 AWS KMS 호스트와 도메인의 HSM 간에 인증된 세션을 통해 전송됩니다.

HSM은 다음을 실행합니다.

1. EKT를 복호화하여 HBK = Decrypt(DK_i, EKT)를 가져옵니다.
2. 임의의 임시 N을 생성합니다.
3. HBK 및 N에서 256비트 AES-GCM 파생 암호화 키 K를 파생합니다.
4. 일반 텍스트 ciphertext = Encrypt(K, context, plaintext)를 암호화합니다.

사이퍼텍스트 값은 사용자에게 반환되며, 일반 텍스트 데이터나 사이퍼텍스트는 AWS 인프라 어디에도 보관되지 않습니다. KMS 키를 사용하기 위한 암호화 텍스트, 암호화 컨텍스트 및 권한 부여를 소유한 경우가 아니면, 기본 일반 텍스트가 반환되지 않습니다.

Decrypt

사이퍼텍스트 값을 복호화 AWS KMS 하기 위한 호출은 암호화된 값 사이퍼텍스트 및 암호화 컨텍스트를 수락합니다.는 AWS 서명 버전 4 서명된 요청을 사용하여 호출을 AWS KMS 인증하고 사이퍼텍스트에서 래핑 키의 HBKID를 추출합니다. HBKID는 암호화 텍스트, 키 ID 및 키 ID에 대한 정책을 복호화하는 데 필요한 EKT를 가져오는 데 사용됩니다. 이 요청에는 키 정책, 존재할 수 있는 권한 부여, 그리고 키 ID를 참조하는 연결된 IAM 정책에 따라 권한이 부여됩니다. `Decrypt` 함수는 암호화 함수와 유사합니다.

다음은 `Decrypt` 요청 구문입니다.

```
{
 "CiphertextBlob": "blob",
 "EncryptionContext": { "string" : "string" }
 "GrantTokens": ["string"]
}
```

요청 파라미터는 다음과 같습니다.

CiphertextBlob

메타데이터가 포함된 암호화 텍스트입니다.

EncryptionContext

(선택 사항) 암호화 컨텍스트입니다. `Encrypt` 함수에 이 값을 지정한 경우 여기에서도 지정해야 하며, 그렇지 않으면 복호화 작업이 실패합니다. 자세한 내용은 AWS Key Management Service 개발자 가이드에서 암호화 컨텍스트를 참조하세요.

GrantTokens

(선택 사항) 복호화 작업을 수행할 권한을 제공하는 권한 부여를 나타내는 권한 부여 토큰 목록입니다.

암호화 텍스트 및 EKT는 복호화를 위해 인증된 세션을 통해 암호화 컨텍스트와 함께 HSM에 전송됩니다.

HSM은 다음을 실행합니다.

1. EKT를 복호화하여 HBK = Decrypt(DK_i, EKT)를 가져옵니다.

2. 암호화 텍스트 구조에서 임의의 N을 추출합니다.
3. HBK 및 N에서 256비트 AES-GCM 파생 암호화 키 K를 다시 생성합니다.
4. 암호화 텍스트를 복호화하여 plaintext = Decrypt(K, context, ciphertext)를 가져옵니다.

결과 키 ID와 일반 텍스트는 보안 세션을 통해 AWS KMS 호스트로 반환된 다음 TLS 연결을 통해 호출하는 고객 애플리케이션으로 다시 반환됩니다.

응답 구문은 다음과 같습니다.

```
{
    "KeyId": "string",
    "Plaintext": blob
}
```

호출 애플리케이션이 일반 텍스트의 신뢰성을 확인하려는 경우 반환되는 키 ID가 올바른 ID인지 확인해야 합니다.

암호화된 객체 다시 암호화

단일 KMS 키로 암호화된 기존 고객 암호화 텍스트는 reencrypt 명령을 통해 다른 KMS 키로 다시 암호화할 수 있습니다. reencrypt 명령은 클라이언트 측에서 키의 일반 텍스트를 노출하지 않으면서 새 KMS 키로 서버 측의 데이터를 암호화합니다. 데이터 암호화가 해제된 후 다시 암호화됩니다.

요청 구문은 다음과 같습니다.

```
{
 "CiphertextBlob": "blob",
 "DestinationEncryptionContext": { "string" : "string" },
 "DestinationKeyId": "string",
 "GrantTokens": ["string"],
       "SourceKeyId": "string",
 "SourceEncryptionContext": { "string" : "string"}
}
```

요청은 JSON 형식으로 다음 데이터를 받습니다.

CiphertextBlob

다시 암호화할 데이터의 암호화 텍스트입니다.

DestinationEncryptionContext

(선택 사항) 데이터를 다시 암호화할 때 사용할 암호화 컨텍스트입니다.

DestinationKeyId

데이터를 다시 암호화하는 데 사용되는 키의 식별자입니다.

GrantTokens

(선택 사항) 복호화 작업을 수행할 권한을 제공하는 권한 부여를 나타내는 권한 부여 토큰 목록입니다.

SourceKeyId

(선택 사항) 데이터를 복호화하는 데 사용되는 키의 키 식별자입니다.

SourceEncryptionContext

(선택 사항) `CiphertextBlob` 파라미터에 지정된 데이터를 암호화하고 복호화하는 데 사용되는 암호화 컨텍스트입니다.

이 프로세스는 앞서 설명한 복호화 및 암호화 작업을 결합합니다. 즉, 고객 암호화 텍스트가 고객 암호화 텍스트에서 참조하는 초기 HBK를 사용하여 의도한 KMS 키를 사용한 현재 HBK로 복호화됩니다. 이 명령에 사용된 두 KMS 키가 동일한 경우 이 명령은 고객 암호화 텍스트를 이전 버전의 HBK에서 최신 버전의 HBK로 업데이트합니다.

응답 구문은 다음과 같습니다.

```
{
    "CiphertextBlob": blob,
    "DestinationEncryptionAlgorithm": "string",
    "KeyId": "string",
    "SourceEncryptionAlgorithm": "string",
    "SourceKeyId": "string"
}
```

호출 애플리케이션이 기본 일반 텍스트의 신뢰성을 확인하려는 경우 반환되는 SourceKeyId가 올바른 ID인지 확인해야 합니다.

AWS KMS 내부 작업

AWS KMS 내부는 전 세계에 분산된 키 관리 서비스의 HSMs을 확장하고 보호하는 데 필요합니다.

주제

도메인 및 도메인 상태

내에서 신뢰할 수 있는 내부 AWS KMS 엔터티의 공동 컬렉션을 도메인이라고 AWS 리전 합니다. 도메인에는 신뢰할 수 있는 엔터티 세트, 규칙 세트 및 도메인 키라는 보안 키 세트가 포함됩니다. 도메인 키는 도메인의 멤버인 HSM 간에 공유됩니다. 도메인 상태는 다음 필드로 구성됩니다.

이름(Name)

이 도메인을 식별하는 도메인 이름입니다.

멤버(Members)

퍼블릭 서명 키 및 퍼블릭 계약 키를 포함하여 도메인의 멤버인 HSM의 목록입니다.

운영자(Operators)

이 서비스의 연산자를 나타내는 엔터티, 퍼블릭 서명 키 및 역할(AWS KMS 운용자 또는 서비스 호스트)의 목록입니다.

규칙(Rules)

HSM에서 명령을 실행하기 위해 충족해야 하는 각 명령의 쿼럼 규칙 목록입니다.

도메인 키

도메인 내에서 현재 사용 중인 도메인 키(대칭 키)의 목록입니다.

전체 도메인 상태는 HSM에서만 사용할 수 있습니다. 도메인 상태는 HSM 도메인 멤버 간에 내보낸 도메인 토큰으로서 동기화됩니다.

도메인 키

도메인의 모든 HSM은 $\{DK_r\}$라는 일련의 도메인 키를 공유합니다. 이러한 키는 도메인 상태 내보내기 루틴을 통해 공유됩니다. 내보낸 도메인 상태는 도메인의 구성원인 HSM으로 가져올 수 있습니다.

도메인 키 세트 $\{DK_r\}$에는 항상 활성 도메인 키 하나와 비활성화된 여러 도메인 키가 포함됩니다. 도메인 키는 키 관리 권장 사항 - 1부를 AWS 준수하도록 매일 교체됩니다. 도메인 키 교체 중에 나가는 도메인 키로 암호화된 모든 기존 KMS 키는 새 활성 도메인 키로 다시 암호화됩니다. 활성 도메인 키는 새 EKT를 암호화하는 데 사용됩니다. 만료된 도메인 키는 최근에 교체된 도메인 키 수와 동일한 일수 동안 이전에 암호화된 EKT를 복호화하는 데에만 사용할 수 있습니다.

내보낸 도메인 토큰

도메인 참가자 간에 상태를 정기적으로 동기화해야 합니다. 이 작업은 도메인이 변경될 때마다 도메인 상태를 내보냄으로써 수행됩니다. 도메인 상태는 내보낸 도메인 토큰으로서 내보내집니다.

이름(Name)

이 도메인을 식별하는 도메인 이름입니다.

멤버(Members)

서명 및 계약 퍼블릭 키를 포함하여 도메인의 멤버인 HSM의 목록입니다.

연산자

이 서비스의 운영자를 나타내는 엔터티, 퍼블릭 서명 키 및 역할의 목록입니다.

규칙(Rules)

HSM 도메인 멤버에서 명령을 실행하기 위해 충족해야 하는 각 명령의 쿼럼 규칙 목록입니다.

암호화된 도메인 키(Encrypted domain keys)

봉투 암호화된 도메인 키입니다. 도메인 키는 위에 나열된 각 멤버의 서명 멤버에 의해 암호화되고 퍼블릭 계약 키로 봉투 처리됩니다.

서명(Signature)

HSM에서 생성한 도메인 상태에 대한 서명으로, 도메인 상태를 내보낸 도메인의 멤버여야 합니다.

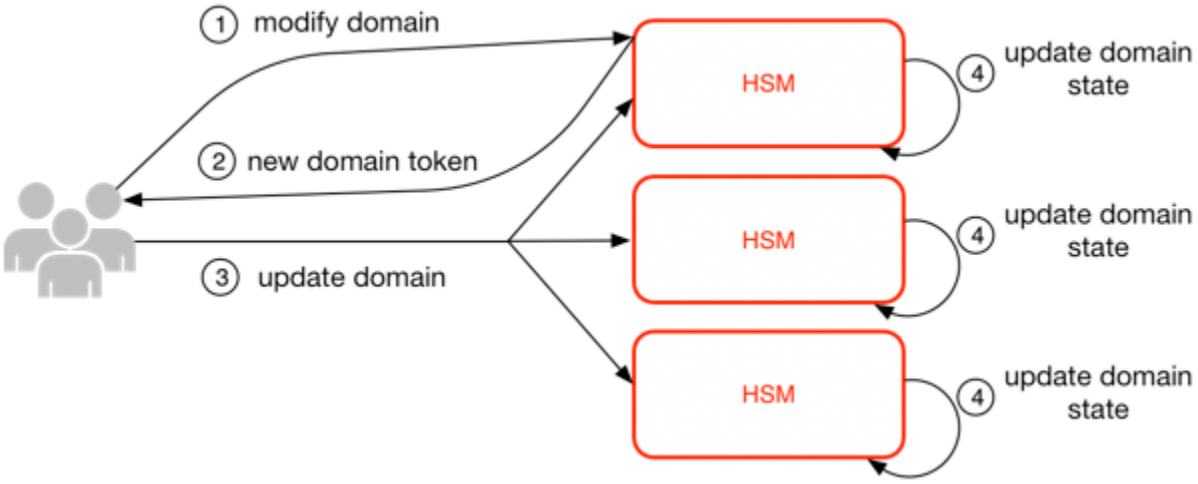
내보낸 도메인 토큰은 도메인 내에서 작동하는 엔터티에 대한 기본 트러스트 소스를 형성합니다.

도메인 상태 관리

도메인 상태는 쿼럼 인증 명령을 통해 관리됩니다. 이러한 변경 사항에는 도메인의 신뢰할 수 있는 참가자 목록 수정, HSM 명령 실행에 대한 쿼럼 규칙 수정, 도메인 키 교체 등이 포함됩니다. 다음 이미지에 표시된 것과 같이 이러한 명령은 인증된 세션 작업과 달리 명령별로 인증됩니다.

초기화되고 정상 작동 상태인 HSM에는 자체 생성된 비대칭 자격 증명 키 세트, 서명 키 페어 및 키 설정 키 페어가 포함됩니다. 수동 프로세스를 통해 AWS KMS 운영자는 리전의 첫 번째 HSM에서 생성할 초기 도메인을 설정할 수 있습니다. 이 초기 도메인은 이 주제에서 이전에 정의한 전체 도메인 상태로 구성됩니다. 조인 명령을 통해 도메인의 정의된 각 HSM 멤버에 설치됩니다.

HSM이 초기 도메인에 조인한 후에는 해당 도메인에 정의된 규칙에 바인딩됩니다. 이러한 규칙은 고객 암호화 키를 사용하는 명령이나 호스트 또는 도메인 상태를 변경하는 명령을 제어합니다. 암호화 키를 사용하는 인증된 세션 API 작업이 이전에 정의되었습니다.



위의 이미지는 도메인 상태가 수정되는 방법을 보여줍니다. 프로세스는 다음 네 단계로 구성됩니다.

1. 도메인을 수정하는 쿼럼 기반 명령이 HSM으로 전송됩니다.
2. 새 도메인 상태가 생성되고 새로 내보낸 도메인 토큰으로 내보내집니다. HSM의 상태는 수정되지 않습니다. 즉, 변경 사항이 HSM에 적용되지 않습니다.
3. 두 번째 명령은 새로 내보낸 도메인 토큰의 각 HSM으로 전송되어 도메인 상태를 새 도메인 토큰으로 업데이트합니다.
4. 내보낸 새 도메인 토큰에 나열된 HSM은 명령과 도메인 토큰을 인증할 수 있습니다. 또한 도메인 키의 압축을 풀고 도메인의 모든 HSM에서 도메인 상태를 업데이트할 수도 있습니다.

HSM은 서로 직접 통신하지 않습니다. 대신 운영자 쿼럼이 도메인 상태에 대한 변경을 요청하여 새로 내보낸 도메인 토큰이 생성됩니다. 도메인의 서비스 호스트 멤버는 도메인의 모든 HSM에 새 도메인 상태를 배포하는 데 사용됩니다.

도메인의 탈퇴 및 조인은 HSM 관리 함수를 통해 수행됩니다. 도메인 상태의 수정은 도메인 관리 함수를 통해 수행됩니다.

도메인 탈퇴

HSM이 도메인에서 탈퇴하게 되어 해당 도메인의 나머지 모든 부분과 키가 메모리에서 삭제됩니다.

도메인 조인

HSM이 새 도메인에 조인되거나 현재 도메인 상태가 새 도메인 상태로 업데이트됩니다. 이 메시지를 인증하기 위한 초기 규칙 세트의 소스로 기존 도메인이 사용됩니다.

도메인 생성

HSM에 새 도메인을 만듭니다. 도메인의 멤버 HSM에 배포할 수 있는 첫 번째 도메인 토큰을 반환합니다.

운영자 수정

도메인의 인증된 운영자 및 해당 역할의 목록에서 운영자를 추가하거나 제거합니다.

멤버 수정

도메인의 인증된 HSM 목록에서 HSM을 추가하거나 제거합니다.

규칙 수정

HSM에서 명령을 실행하는 데 필요한 쿼럼 규칙 세트를 수정합니다.

도메인 키 교체

새 도메인 키를 만들어 활성 도메인 키로 표시합니다. 이렇게 하면 기존 활성 키가 비활성화된 키로 이동하고 도메인 상태에서 가장 오래된 비활성화된 키가 제거됩니다.

내부 통신 보안

서비스 호스트 또는 AWS KMS 연산자와 HSMs 간의 명령은에 설명된 두 가지 메커니즘, 인증된 세션즉 쿼럼 서명 요청 방법과 HSM 서비스 호스트 프로토콜을 사용하는 인증된 세션을 통해 보호됩니다.

쿼럼 서명 명령은 단일 운영자가 HSM이 제공하는 중요한 보안 보호 기능을 수정할 수 없도록 설계되었습니다. 인증된 세션을 통해 실행되는 명령을 사용하면 인증된 서비스 운영자만 KMS 키와 관련된 작업을 수행하도록 할 수 있습니다. 모든 고객 기반 보안 암호 정보는 AWS 인프라 전체에서 보호됩니다.

키 설정

내부 통신을 보호하기 위해는 서로 다른 두 가지 키 설정 방법을 AWS KMS 사용합니다. 첫 번째는 이산 로그 암호화를 사용한 페어 방식 키 설정 체계에 대한 권장 사항(개정 2)에 C(1, 2, ECC DH)로 정의되어 있습니다. 이 체계에는 정적 서명 키가 있는 초기자가 있습니다. 이 초기자는 정적 ECDH 계약 키가 있는 수신자를 위한 임시 타원 곡선 디Diffie-Hellman(ECDH) 키를 생성하고 서명합니다. 이 방법은 ECDH를 사용하는 하나의 임시 키와 2개의 정적 키를 사용합니다. 이는 레이블 C(1, 2, ECC DH)의 파생입니다. 이 방법을 원 패스 ECDH라고도 합니다.

두 번째 키 설정 방법은 C(2, 2, ECC, DH)입니다. 이 방식에서 양 당사자는 정적 서명 키를 가지며 임시 ECDH 키를 생성, 서명 및 교환합니다. 이 방법은 각각 ECDH를 사용하는 2개의 정적 키와 2개의 임시 키를 사용합니다. 이는 레이블 C(2, 2, ECC, DH)의 파생입니다. 이 방법은 ECDH 임시 또는 ECDHE라고도 합니다. 모든 ECDH 키는 secp384r1(NIST-P384) 곡선에 생성됩니다.

HSM 보안 경계

의 내부 보안 경계는 HSM AWS KMS 입니다. HSM은 독점적 인터페이스를 제공하며 작동 상태일 때 다른 활성 물리적 인터페이스는 없습니다. 작동 HSM은 초기화 중에 도메인에서 역할을 설정하는 데 필요한 암호화 키를 사용하여 프로비저닝됩니다. HSM의 민감한 암호화 구성 요소는 휘발성 메모리에만 저장되며, HSM이 의도하거나 의도하지 않은 종료 또는 재설정을 비롯한 작동 상태를 벗어날 때 지워집니다.

HSM API 작업은 개별 명령에 의해 또는 서비스 호스트가 설정한 상호 인증된 기밀 세션을 통해 인증됩니다.

쿼럼 서명 명령

Quorum-signed 명령은 운영자가 HSM에 대해 실행합니다. 이 섹션에서는 쿼럼 기반 명령을 만들고 서명하고 인증하는 방법에 대해 설명합니다. 이 규칙은 매우 간단합니다. 예를 들어 Foo 명령의 경우 Bar 역할의 두 멤버가 인증되어야 합니다. 쿼럼 기반 명령의 생성 및 검증은 세 단계로 이루어집니다. 첫 번째 단계는 초기 명령을 생성하는 것이고, 두 번째 단계는 서명할 추가 운영자에게 제출하는 것이며, 세 번째 단계는 검증하고 실행하는 것입니다.

이 개념을 도입하기 위해 운영자의 퍼블릭 키 및 역할 세트가 $\{QOS_s\}$이고, 쿼럼 규칙 세트가 QR = $\{Command_i, Rule_{\{i, t\}}\}$(여기서 각 Rule은 역할의 세트)이며, 최소 개수가 N $\{Role_t, N_t\}$이라고 가정합니다. 명령이 이 쿼럼 규칙을 충족하려면 해당 명령에 대해 나열된 규칙 중 하나를 충족하도록 $\{QOS_s\}$가에 나열된 작업 세트로 명령 데이터 집합에 서명해야 합니다. 앞서 언급했듯이 쿼럼 규칙 및 운영자 세트는 도메인 상태 및 내보낸 도메인 토큰에 저장됩니다.

실제로 초기 Signer는 Sig_1 = Sign(dO_{p1}, Command) 명령에 서명합니다. 두 번째 운영자는 Sig_2 = Sign(dO_{p2}, Command) 명령에 서명합니다. 이중으로 서명된 메시지는 실행을 위해 HSM으로 전송됩니다. HSM은 다음을 수행합니다.

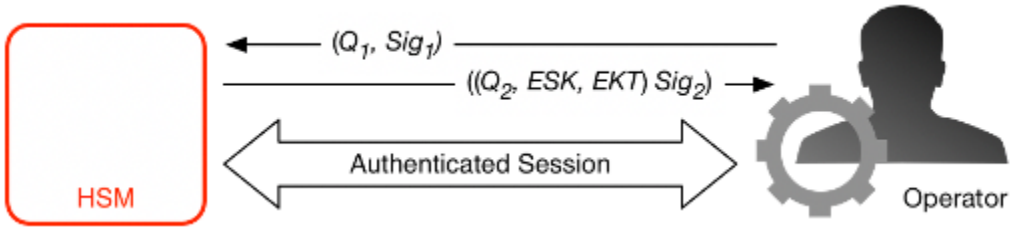
1. 각 서명에 대해 도메인 상태에서 Signer의 퍼블릭 키를 추출하고 명령의 서명을 검증합니다.
2. Signer 세트가 명령에 대한 규칙을 충족하는지 확인합니다.

인증된 세션

키 작업은 외부 대상 AWS KMS 호스트와 HSMs. 이러한 명령은 암호화 키의 생성 및 사용, 그리고 보안 난수 생성과 관련이 있습니다. 이들 명령은 서비스 호스트와 HSM 간의 세션 인증 채널을 통해 실행됩니다. 이러한 세션에는 신뢰성 외에 기밀성도 요구됩니다. 이러한 세션을 통해 실행되는 명령에서는 일반 텍스트 데이터 키 및 복호화된 메시지가 반환됩니다. 중간자 공격을 통해 이러한 세션을 악용할 수 없도록 하기 위해 세션이 인증됩니다.

이 프로토콜은 HSM과 서비스 호스트 간에 상호 인증된 ECDHE 키 계약을 수행합니다. 교환은 서비스 호스트에 의해 시작되고 HSM에 의해 완료됩니다. 또한 HSM은 협상된 키로 암호화된 세션 키(SK)와 이 세션 키가 포함된 내보낸 키 토큰을 반환합니다. 내보낸 키 토큰에는 유효 기간이 포함되며, 이 기간이 지나면 서비스 호스트가 세션 키를 다시 협상해야 합니다.

서비스 호스트는 도메인의 멤버이며 자격 증명 서명 키 페어($dHOS_i$, $QHOS_i$)와 HSM의 자격 증명 퍼퍼블릭 키의 올바른 사본을 가지고 있습니다. 자격 증명 서명 키 세트를 사용하여 도메인의 서비스 호스트와 HSM 간에 사용할 수 있는 세션 키를 안전하게 협상합니다. 내보낸 키 토큰에는 유효 기간이 있으며 그 기간이 지나면 새 키를 협상해야 합니다.



이 프로세스는 그 자체와 도메인의 HSM 멤버 간의 중요한 통신 흐름을 보내고 받기 위해 세션 키가 필요하다는 것을 서비스 호스트가 인식하면서 시작됩니다.

1. 서비스 호스트가 ECDH 임시 키 페어 (d_1, Q_1)를 생성하고 자격 증명 키 Sig1 = Sign(dOS,Q_1)를 사용하여 서명합니다.
2. HSM이 현재 도메인 토큰을 사용하여 수신된 퍼블릭 키의 서명을 확인하고 ECDH 임시 키 페어 (d_2, Q_2)를 생성합니다. 그런 다음 이산 로그 암호화를 사용한 페어 방식 키 설정 체계에 대한 권장 사항(개정)에 따라 ECDH 키 교환을 완료하여 협상된 256비트 AES-GCM 키를 구성합니다. HSM이 새로운 256비트 AES-GCM 세션 키를 생성합니다. 협상된 키로 세션 키를 암호화하여 암호화된 세션 키(ESK)를 만듭니다. 또한 내보낸 키 토큰 EKT으로서 도메인 키를 사용하여 세션 키를 암호화합니다. 마지막으로 자격 증명 키 페어 Sig_2 = Sign(dHSK, (Q_2, ESK, EKT))를 사용하여 반환 값에 서명합니다.
3. 서비스 호스트가 현재 도메인 토큰을 사용하여 수신된 키의 서명을 확인합니다. 그런 다음 서비스 호스트는 이산 로그 암호화를 사용한 페어 방식 키 설정 체계에 대한 권장 사항(개정)에 따라 ECDH 키 교환을 완료합니다. 다음으로, ESK를 복호화하여 세션 키 SK를 가져옵니다.

EKT의 유효 기간 동안, 서비스 호스트는 협상된 세션 키 SK를 사용하여 봉투 암호화 명령을 HSM으로 보낼 수 있습니다. 이 인증된 세션을 통해 서비스 호스트가 시작한 모든 명령에는 EKT가 포함됩니다. HSM은 동일한 협상된 세션 키 SK를 사용하여 응답합니다.

다중 리전 키의 복제 프로세스

AWS KMS 는 리전 간 복제 메커니즘을 사용하여 KMS 키의 키 구성 요소를 HSM에서 다른 HSM AWS 리전 으로 복사합니다 AWS 리전. 이 메커니즘이 작동하려면 복제하는 KMS 키가 다중 리전 키여야 합니다. 한 리전에서 다른 리전으로 KMS 키를 복제할 때 리전의 HSM은 격리된 네트워크에 있으므로 직접 통신할 수 없습니다. 대신 리전 간 복제 중에 교환된 메시지가 프록시 서비스를 통해 전달됩니다.

리전 간 복제 중에 AWS KMS HSM에서 생성된 모든 메시지는 복제 서명 키를 사용하여 암호화 방식으로 서명됩니다. RSK(복제 서명 키)는 NIST P-384 곡선의 ECDSA 키입니다. 모든 리전은 하나 이상의 RSK를 소유하며, 각 RSK의 퍼블릭 구성 요소는 동일한 AWS 파티션의 다른 모든 리전과 공유됩니다.

리전 A에서 리전 B로 주요 구성 요소를 복사하는 리전 간 복제 프로세스는 다음과 같이 작동합니다.

1. 리전 B의 HSM은 NIST P-384 곡선의 임시 ECDH 키인 RAKB(Replication Agreement Key B)를 생성합니다. RAKB의 퍼블릭 구성 요소는 프록시 서비스를 통해 리전 A의 HSM으로 전송됩니다.
2. 리전 A의 HSM은 RAKB의 퍼블릭 구성 요소를 수신한 다음 NIST P-384 곡선의 다른 임시 ECDH 키인 RAKA(Replication Agreement Key A)를 생성합니다. HSM은 RAKA 및 RAKB의 퍼블릭 구성 요소에 ECDH 키 설정 체계를 실행하고 출력에서 대칭 키 RWK(Replication Wrapping Key)를 파생합니다. RWK는 복제되는 다중 리전 KMS 키의 키 구성 요소를 암호화하는 데 사용됩니다.
3. RAKA의 퍼블릭 구성 요소와 RWK로 암호화된 키 구성 요소는 프록시 서비스를 통해 리전 B의 HSM으로 전송됩니다.
4. 리전 B의 HSM은 RAKA의 퍼블릭 구성 요소와 RWK를 사용하여 암호화된 키 구성 요소를 수신합니다. HSM은 RAKB와 RAKA의 퍼블릭 구성 요소에 대해 ECDH 키 설정 체계를 실행하여 RWK를 파생합니다.
5. 리전 B의 HSM은 RWK를 사용하여 리전 A의 키 구성 요소를 복호화합니다.

내구성 보호

이 서비스에 의해 생성된 키의 추가 서비스 내구성은 오프라인 HSM, 내보낸 도메인 토큰의 다중 비휘발성 스토리지, 암호화된 KMS 키의 이중화 스토리지를 사용하여 제공됩니다. 오프라인 HSM은 기존 도메인의 멤버입니다. 온라인 상태가 아닌 채로 일반 도메인 작업에 참여하는 것을 제외하고, 오프라인 HSM은 도메인 상태에 기존 HSM 멤버와 동일하게 표시됩니다.

내구성 설계는 리전의 모든 KMS 키가 기본 스토리지 시스템에 저장된 온라인 HSMs 또는 KMS 키 세트가 광범위하게 손실되는 AWS 것을 방지하기 위한 것입니다. AWS KMS keys 가져온 키 구성 요소는 다른 KMS 키에 제공된 내구성 보호에 포함되지 않습니다. 리전 전체에 장애가 발생하는 경우 AWS KMS가져온 키 구성 요소를 KMS 키로 다시 가져와야 할 수 있습니다.

오프라인 HSM과 해당 HSM에 액세스하기 위한 자격 증명은 독립적인 여러 지리적 위치에 있는 모니터링되는 안전실 내의 금고에 저장됩니다. 각 금고에는 이러한 자료를 얻 AWS기 위해 두 개의 독립 팀에서 AWS 보안 책임자와 AWS KMS 운영자가 한 명 이상 필요합니다. 이러한 재료의 사용은 연 AWS KMS 산자 쿼럼이 있어야 하는 내부 정책에 의해 관리됩니다.

레퍼런스

이 문서에 인용된 약어, 키, 기여자 및 소스에 대한 정보는 다음 참고 자료를 참조하세요.

주제

약어

다음 목록에는 이 문서에서 참조한 약어가 나와 있습니다.

AES

고급 암호화 표준

CDK

고객 데이터 키

DK

도메인 키

ECDH

Elliptic Curve Diffie-Hellman

ECDHE

Elliptic Curve Diffie-Hellman Ephemeral

ECDSA

Elliptic Curve Digital Signature Algorithm

EKT

내보낸 키 토큰

ESK

암호화된 세션 키

GCM

Galois Counter Mode

HBK

HSM 백업 키

HBKID

HSM 백업 키 식별자

HSM

하드웨어 보안 모듈

RSA

Rivest Shamir and Adleman(암호학)

secp384r1

효율적인 암호화 프라임 384비트 임의 곡선 1의 표준

SHA256

다이제스트 길이 256비트의 보안 해시 알고리리즘

키

다음은 이 문서에서 참조하는 키를 정의하는 목록입니다.

HBK

HSM 백업 키: HSM 백업 키는 특정 사용 키가 파생되는 256비트 루트 키입니다.

DK

도메인 키: 도메인 키는 256비트 AES-GCM 키입니다. 도메인의 모든 멤버 간에 공유되며 HSM 백업 키 자료 및 HSM 서비스 호스트 세션 키를 보호하는 데 사용됩니다.

DKEK

도메인 키 암호화 키: 도메인 키 암호화 키는 호스트에서 생성된 AES-256-GCM 키이며 HSM 호스트에서 도메인 상태를 동기화하는 현재 도메인 키 세트를 암호화하는 데 사용됩니다.

(dHAK,QHAK)

HSM 계약 키 페어: 시작된 모든 HSM에는 secp384r1(NIST-P384) 곡선에 로컬로 생성된 타원 곡선 Diffie-Hellman 계약 키 페어가 있습니다.

(dE, QE)

임시 계약 키 페어: HSM 및 서비스 호스트는 임시 계약 키를 생성합니다. 이 키는 secp384r1(NIST-P384) 곡선에 있는 타원 곡선 Diffie-Hellman 키입니다. 이 키는 도메인 토큰에서 도메인 키 암호화 키를 전송하기 위해 호스트 간 암호화 키를 설정하는 경우와 민감한 통신을 보호하기 위해 HSM 서비스 호스트 세션 키를 설정하는 경우의 두 가지 사용 사례에서 생성됩니다.

(dHSK,QHSK)

HSM 서명 키 페어: 시작된 모든 HSM에는 secp384r1(NIST-P384) 곡선에 로컬로 생성된 타원 곡선 디지털 서명 키 페어가 있습니다.

(dOS,QOS)

연산자 서명 키 페어: 서비스 호스트 연산자와 AWS KMS 연산자 모두에 다른 도메인 참가자에게 자신을 인증하는 데 사용되는 자격 증명 서명 키가 있습니다.

K

데이터 암호화 키: SHA256 기반 HMAC를 사용하는 카운터 모드에서 NIST SP800-108 KDF를 사용하는 HBK에서 파생된 256비트 AES-GCM 키입니다.

SK

세션 키: 서비스 호스트 운영자와 HSM 간에 교환된 인증된 타원 곡선 Diffie-Hellman 키의 결과로 생성된 세션 키입니다. 교환의 목적은 서비스 호스트와 도메인 멤버 간의 통신을 보호하는 것입니다.

기여자

다음 개인과 조직이 이 문서에 기여했습니다.

- Ken Beer, 총지배인 - KMS, AWS Cryptography
- Matthew Campagna, 보안 담당 수석 엔지니어, AWS 암호화

참고문헌

AWS Key Management Service HSMs에 대한 자세한 내용은 NIST 컴퓨터 보안 리소스 센터 암호화 모듈 검증 프로그램 검색 페이지로 이동하여 AWS Key Management Service HSM을 검색합니다.

Amazon Web Services, 일반 참조(버전 1.0), “API AWS 요청 서명”, http://docs.aws.amazon.com/general/latest/gr/signing_aws_api_requests.html.

Amazon Web Services, “What is the” AWS Encryption SDK, http://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/introduction.html.

Federal Information Processing Standards Publications, FIPS PUB 180-4. 보안 해시 표준, 2012년 8월 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf에서 확인할 수 있습니다.

Federal Information Processing Standards Publication 197, Announcing the Advanced Encryption Standard (AES), 2001년 11월. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf에서 확인할 수 있습니다.

Federal Information Processing Standards Publication 198-1, The Keyed-Hash Message Authentication Code (HMAC), 2008년 7월. http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf에서 확인할 수 있습니다.

NIST Special Publication 800-52 Revision 2, Guidelines for the Selection, Configuration, and Use of Transport Layer Security(TLS) Implementations, 2019년 8월. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r2.pdf.

PKCS#1 v2.2: RSA Cryptography Standard (RFC 8017), Internet Engineering Task Force(IETF), 2016년 11월. https://tools.ietf.org/html/rfc8017.

Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, NIST Special Publication 800-38D, 2007년 11월. http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf에서 확인할 수 있습니다.

Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices, NIST Special Publication 800-38E, 2010년 1월. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf에서 확인할 수 있습니다.

Recommendation for Key Derivation Using Pseudorandom Functions, NIST Special Publication 800-108, 2009년 10월, https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-108.pdf에서 확인할 수 있습니다.

Recommendation for Key Management - Part 1: General (Revision 5), NIST Special Publication 800-57A, 2020년 5월, https://doi.org/10.6028/NIST.SP.800-57pt1r5에서 확인할 수 있습니다.

Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (Revised), NIST Special Publication 800-56A Revision 3, 2018년 4월. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf에서 확인할 수 있습니다.

Recommendation for Random Number Generation Using Deterministic Random Bit Generators, NIST Special Publication 800-90A Revision 1, 2015년 1월, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf에서 확인할 수 있습니다.

SEC 2: Recommended Elliptic Curve Domain Parameters, Standards for Efficient Cryptography Group, Version 2.0, 2010년 1월 27일.

Use of Elliptic Curve Cryptography(ECC) Algorithms in Cryptographic Message Syntax(CMS), Brown, D., Turner, S., Internet Engineering Task Force, 2010년 7월, http://tools.ietf.org/html/rfc5753/.

X9.62-2005: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm(ECDSA), American National Standards Institute, 2005년.

AWS KMS 암호화 세부 정보에 대한 문서 기록

다음 표에서는 AWS Key Management Service 암호화 세부 정보에 대한 설명서에서 변경된 중요 사항에 대해 설명합니다. 또한 사용자로부터 받은 의견을 수렴하기 위해 설명서가 자주 업데이트됩니다.

변경 사항	설명	날짜
업데이트 내용	작업 구현 AWS KMS `ReplicateKey` 에 대한 세부 정보가 추가되었습니다.	2021년 10월 28일
문서 변경	고객 마스터 키(CMK) 라는 용어가 AWS KMS key과 KMS 키로 바뀌었습니다.	2021년 8월 30일
최초 릴리스	이 가이드는 KMS 암호화 세부 정보 기술 백서를 기준으로 작성되었습니다.	2020년 12월 30일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.