



사용자 가이드

AWS 결제 암호화



AWS 결제 암호화: 사용자 가이드

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 브랜드 디자인은 Amazon 외 제품 또는 서비스와 관련하여 고객에게 혼동을 일으킬 수 있는 방식이나 Amazon 브랜드 이미지를 떨어뜨리는 방식으로 사용할 수 없습니다. 이 소유하지 않은 기타 모든 상표는 과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

Table of Contents

AWS 결제 암호화란 무엇입니까?	1
개념	2
업계 용어	3
공통 키 유형	4
기타 용어	6
관련 서비스	8
자세한 정보	9
엔드포인트	9
컨트롤 플레인 엔드포인트	9
데이터 영역 엔드포인트	10
시작하기	11
필수 조건	11
1단계: 키 생성	11
2단계: 키를 사용하여 CVV2 값 생성	12
3단계: 2단계에서 생성된 값 확인	13
4단계: 네거티브 테스트 수행	14
5단계: (선택 사항) 정리	14
키 관리	16
키 생성	16
2KEY TDES 키 생성	17
핀 암호화 키 생성	18
비대칭 (RSA) 키 생성	19
PIN 검증 값 (PVV) 키 생성	20
키 나열	21
키 활성화 및 비활성화	22
키 사용 시작	22
키 사용 중지	24
키 삭제	25
대기 기간에 대해	26
키 가져오기 및 내보내기	28
키 가져오기	29
키 내보내기	39
별칭 사용	46
별칭 정보	47

애플리케이션에서 별칭 사용	50
관련 API	51
키 가져오기	51
키 페어와 연결된 퍼블릭 키/인증서 가져오기	52
키 태그 지정	53
결제 암호화의 태그에 대한 정보 AWS	53
콘솔에서 키 태그 보기	55
API 작업으로 키 태그 관리	55
태그에 대한 액세스 제어	57
태그를 사용하여 키에 대한 액세스 제어	61
주요 특성 이해	64
대칭 키	65
비대칭 키	67
데이터 작업	68
데이터 암호화, 복호화 및 재암호화	68
Encrypt data	69
데이터 해독	73
카드 데이터 생성 및 확인	76
카드 데이터 생성	76
카드 데이터 확인	77
PIN 데이터 생성, 변환 및 확인	78
PIN 데이터 변환	78
PIN 데이터 생성	80
PIN 데이터 확인	81
인증 요청(ARQC) 암호화 확인	82
트랜잭션 데이터 구축	82
트랜잭션 데이터 패딩	83
예	83
HMAC 생성 및 확인	85
MAC 생성	86
MAC 검증	87
특정 데이터 작업을 위한 키 유형	87
GenerateCardData	88
VerifyCardData	89
GeneratePinData (비자/ABA 제도용)	90
GeneratePinData (용 IBM3624)	91

VerifyPinData (비자/ABA 제도용)	92
VerifyPinData (용IBM3624)	93
데이터 해독	94
데이터 암호화	95
PIN 데이터 변환	96
VerifyAuthRequestCryptogram	97
미사용 키 유형	97
보안	98
데이터 보호	98
키 구성 요소 보호	100
데이터 암호화	100
저장 중 암호화	100
전송 중 암호화	100
인터넷워크 트래픽 개인 정보	101
복원력	101
리전별 격리	102
멀티 테넌트 디자인	102
인프라 보안	103
물리적 호스트의 격리	103
보안 모범 사례	103
규정 준수 검증	105
자격 증명 및 액세스 관리	106
고객	106
자격을 통한 인증	107
AWS 계정 루트 사용자	107
IAM 사용자 및 그룹	107
IAM 역할	108
정책을 사용한 액세스 관리	109
ID 기반 정책	110
리소스 기반 정책	110
액세스 제어 목록(ACLs)	110
기타 정책 타입	111
여러 정책 타입	111
AWS 결제 암호화가 IAM과 작동하는 방식	111
AWS 결제 암호화 ID 기반 정책	112
AWS Payment Cryptography 태그 기반 인증	114

자격 증명 기반 정책 예시	114
정책 모범 사례	115
콘솔 사용	115
사용자가 자신이 권한을 볼 수 있도록 허용	116
AWS 결제 암호화의 모든 측면에 액세스할 수 있습니다.	117
지정된 키를 사용하여 API 호출 가능	117
리소스를 구체적으로 거부할 수 있음	118
문제 해결	119
모니터링(Monitoring)	120
CloudTrail 로그	120
CloudTrail의 AWS 결제 암호화 정보	121
AWS 결제 암호화 로그 파일 항목 이해	122
암호화 세부 정보	126
설계 목표	126
기본	127
암호 프리미티브	128
엔트로피 및 난수 생성	128
대칭 키 작업	128
비대칭 키 작업	129
키 스토리지	129
대칭 키를 사용한 키 가져오기	129
비대칭 키를 사용한 키 가져오기	129
키 내보내기	130
트랜잭션별 파생된 고유 키(DUKPT) 프로토콜	130
키 계층 구조	130
내부 작업	133
HSM 사양 및 수명 주기	133
HSM 디바이스 물리적 보안	133
HSM 초기화	134
HSM 서비스 및 수리	134
HSM 해제	134
HSM 펌웨어 업데이트	135
운영자 액세스	135
키 관리	135
고객 작업	141
키 생성	142

키 가져오기	142
키 내보내기	143
키 삭제	143
키 교체 중	143
할당량	144
사용 설명서 기록	146
.....	cxlvii

AWS 결제 암호화란 무엇입니까?

AWS 결제 암호화는 전용 결제 HSM 인스턴스를 조달할 필요 없이 결제 카드 산업(PCI) 표준에 따라 결제 처리에 사용되는 암호화 기능 및 키 관리에 액세스할 수 있는 관리형 AWS 서비스입니다. AWS 결제 암호화는 취득자, 결제 협력자, 네트워크, 스위치, 프로세서 및 은행 등 결제 기능을 수행하는 고객이 결제 암호화 작업에 클라우드의 애플리케이션을 더 쉽게 이용하고 전용 결제 HSM이 포함된 보조 데이터 센터 또는 콜로케이션 시설에 대한 의존성을 최소화할 수 있는 기능을 제공합니다.

이 서비스는 PCI PIN, PCI P2PE 및 PCI DSS를 포함한 해당 업계 규칙을 준수하도록 설계되었으며, 이 서비스에서 활용하는 하드웨어는 [PCI PTS HSM V3 및 FIPS 140-2 레벨 3 인증](#)을 받았습니다. 짧은 지연 시간과 [높은 수준의 가동 시간 및 복원력](#)을 지원하도록 설계되었습니다. AWS 결제 암호화는 뛰어난 탄력성을 갖추고 있어 하드웨어 프로비저닝, 키 구성 요소의 안전한 관리, 안전한 시설에서의 긴급 백업 유지 등 온프레미스 HSM의 많은 운영 요구 사항을 없애줍니다. AWS 또한 결제 암호화는 파트너와 전자적으로 키를 공유하는 옵션을 제공하기 때문에 종이로 된 일반 텍스트 구성 요소를 공유할 필요가 없습니다.

[AWS 결제 암호화 컨트롤 플레인 API](#)를 사용하여 키를 생성하고 관리할 수 있습니다.

[AWS 결제 암호화 데이터 영역 API](#)를 사용하여 결제 관련 트랜잭션 처리 및 관련 암호화 작업에 암호화 키를 사용할 수 있습니다.

AWS 결제 암호화는 키 관리에 사용할 수 있는 중요한 기능을 제공합니다.

- TDES, AES, RSA 키를 비롯한 대칭 및 비대칭 AWS 결제 암호화 키를 생성 및 관리하고 CVV 생성 또는 DUKPT 키 도출과 같은 용도를 지정할 수 있습니다.
- 사용 사례 간에 키 분리를 적용하면서 하드웨어 보안 모듈(HSM)로 보호되는 AWS 결제 암호화 키를 자동으로 안전하게 저장합니다.
- AWS 결제 암호화 키에 액세스하거나 액세스를 제어하는 데 사용할 수 있는 '친숙한 이름'인 별칭을 만들고, 삭제하고, 나열하고, 업데이트합니다.
- 식별, 그룹화, 자동화, 액세스 제어 및 비용 추적을 위해 AWS 결제 암호화 키에 태그를 지정 합니다.
- TR-31(상호 운용 가능한 보안 키 교환 키 블록 사양)에 따른 키 암호화 키(KEK)를 사용하여 AWS 결제 암호화와 HSM(또는 타사) 간에 대칭 키를 가져오고 내보낼 수 있습니다.
- 비대칭 키 페어를 사용한 다음 TR-34(비대칭 기법을 사용한 대칭 키 배포 방법)와 같은 전자 방식을 사용하여 AWS 결제 암호화와 다른 시스템 간에 대칭 키 암호화 키(KEK)를 가져오고 내보냅니다.

AWS 결제 암호화 키는 다음과 같은 암호화 작업에 사용할 수 있습니다.

- 대칭 또는 비대칭 AWS 결제 암호화 키를 이용해 데이터를 암호화, 해독 및 재암호화합니다.
- PCI PIN 규칙에 따라 일반 텍스트를 노출하지 않고 암호화 키 간에 민감한 데이터(예: 카드 소유자 핀)를 안전하게 변환합니다.
- CVV, CVV2 또는 ARQC와 같은 카드 소유자 데이터를 생성하거나 검증합니다.
- 카드 소유자 핀을 생성하고 검증합니다.
- MAC 서명을 생성하거나 검증합니다.

개념

AWS 결제 암호화에 사용하는 기본 용어 및 개념과 사용자 데이터를 보호하는 방법에 대해 알아보니다.

별칭

AWS 결제 암호화 키와 연결된, 사용자에게 친숙한 이름입니다. 별칭은 다양한 AWS 결제 암호화 API 작업에서 [키 ARN](#)과 상호 교환하여 사용할 수 있습니다. 별칭을 사용하면 애플리케이션 코드에 영향을 주지 않고 키를 교체하거나 변경할 수 있습니다. 별칭 이름은 최대 256자의 문자열입니다. 계정 및 리전 내의 연결된 AWS 결제 암호화 키를 고유하게 식별합니다. AWS 결제 암호화에서 별칭 이름은 항상 `alias/`로 시작합니다.

별칭 이름의 형식은 다음과 같습니다.

```
alias/<alias-name>
```

예:

```
alias/sampleAlias2
```

키 ARN

키 ARN은 AWS 결제 암호화에 있는 키 항목의 Amazon 리소스 이름(ARN)입니다. AWS 결제 암호화 키에 대한 고유한 정규화된 식별자입니다. 키 ARN에는 AWS 계정, 리전, 무작위로 생성된 ID가 포함됩니다. ARN은 키 구성 요소와 관련되거나 파생되지 않습니다. 생성 또는 가져오기 작업 중에 자동으로 할당되기 때문에 이러한 값은 멱등성이 없습니다. 동일한 키를 여러 번 가져오면 고유한 수명 주기를 가진 키 ARN이 여러 개 생성됩니다.

키 ARN의 형식은 다음과 같습니다.

```
arn:<partition>:payment-cryptography:<region>:<account-id>:alias/<alias-name>
```

다음은 키 ARN 샘플입니다.

```
arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaiif1lw2h
```

AWS 결제 암호화 키

AWS 결제 암호화 키는 모든 암호화 기능에 사용됩니다. 키는 `create key` 명령을 사용하여 직접 생성하거나 키 가져오기를 호출하여 시스템에 추가할 수 있습니다. 키의 출처는 `KeyOrigin` 속성을 검토하여 확인할 수 있습니다. AWS 결제 암호화는 DUKPT에서 사용하는 것과 같은 암호화 작업 중에 사용되는 파생형 또는 중간 키도 지원합니다.

이러한 키는 생성 시 정의된 변경 불가 속성과 변경 가능 속성을 모두 가지고 있습니다. 알고리즘, 길이, 사용 등의 속성은 생성 시 정의되며 변경할 수 없습니다. 발효일 또는 만료일과 같은 기타 항목은 수정할 수 있습니다. AWS 결제 암호화 키 속성의 전체 목록은 [AWS 결제 암호화 API 참조](#)를 참고하세요.

AWS 결제 암호화 키에는 주로 [ANSI X9 TR 31](#)에 정의된 키 유형이 있으며, 이러한 키 유형은 PCI PIN v3.1 요구 사항 19에 명시된 용도로만 사용할 수 있습니다.

속성은 PCI PIN v3.1 요구 사항 18-3에 지정된 대로 저장하거나, 다른 계정과 공유하거나, 내보낼 때 키 블록을 사용하여 키에 바인딩됩니다.

키는 AWS 결제 암호화 플랫폼에서 키 Amazon 리소스 이름(ARN)이라는 고유한 값을 사용하여 식별됩니다.

Note

키 ARN는 키를 처음 생성하거나 AWS 결제 암호화 서비스로 가져올 때 생성됩니다. 따라서 키 가져오기 기능을 사용하여 동일한 키 자료를 여러 번 추가하면 동일한 키 구성 요소가 여러 개의 키 아래에 위치하지만 각각의 수명 주기는 다릅니다.

업계 용어

주제

- [공통 키 유형](#)
- [기타 용어](#)

공통 키 유형

AWK

취득자 작업 키(AWK)는 일반적으로 취득자/취득자 프로세서와 네트워크(예: Visa 또는 Mastercard) 간에 데이터를 교환하는 데 사용되는 키입니다. 지금까지 AWK는 암호화에 3DES를 활용하고 이를 TR31_P0_PIN_ENCRYPTION_KEY로 표현했습니다.

BDK

기본 파생 키(BDK)는 후속 키를 도출하는 데 사용되는 작업 키이며 일반적으로 PCI PIN 및 PCI P2PE DUKPT 프로세스의 일부로 사용됩니다. TR31_B0_BASE_DERIVATION_KEY로 표시됩니다.

CMK

카드 마스터 키(CMK)는 일반적으로 [발급자 마스터 키](#), PAN 및 PSN에서 파생된 하나 이상의 카드별 키이며 일반적으로 3DES 키입니다. 이러한 키는 개인화 과정에서 EMV 칩에 저장됩니다. CMK의 예로는 AC, SMI 및 SMC 키가 있습니다.

CMK-AC

애플리케이션 암호(AC) 키는 EMV 트랜잭션의 일부로 트랜잭션 암호를 생성하는 데 사용되며 일종의 [카드 마스터 키](#)입니다.

CMK-SMI

보안 메시징 무결성(SMI) 키는 EMV의 일부로 사용되어 핀 업데이트 스크립트와 같은 MAC을 사용하여 카드로 전송되는 페이로드의 무결성을 확인합니다. 일종의 [카드 마스터 키](#)입니다.

CMK-SMC

보안 메시징 기밀성(SMC) 키는 EMV의 일부로 핀 업데이트와 같이 카드로 전송되는 데이터를 암호화하는 데 사용됩니다. 일종의 [카드 마스터 키](#)입니다.

CVK

카드 검증 키(CVK)는 정의된 알고리즘을 사용하여 CVV, CVV2 및 유사한 값을 생성하고 입력을 검증하는 데 사용되는 키입니다. TR31_C0_CARD_VERIFICATION_KEY로 표시됩니다.

IMK

발급자 마스터 키(IMK)는 EMV 칩 카드 개인화의 일부로 사용되는 마스터 키입니다. 일반적으로 3개의 IMK가 있으며, 각각 AC(암호문), SMI(무결성/서명을 위한 스크립트 마스터 키), SMC(기밀성/암호화를 위한 스크립트 마스터 키) 키에 하나씩 사용됩니다.

IK

초기 키 (IK) 는 DUKPT 프로세스에서 사용되는 첫 번째 키이며 기본 파생 키 (BDK) 에서 파생됩니다. 이 키에 대한 트랜잭션은 처리되지 않지만 트랜잭션에 사용될 미래 키를 도출하는 데 사용됩니다. IK를 생성하기 위한 파생 방법은 X9. 24-1:2017 년에 정의되었습니다. TDES BDK를 사용하는 경우 X9. 24-1:2009 가 해당 표준이며 IK는 초기 핀 암호화 키 (IPEK) 로 대체됩니다.

IPEK

초기 PIN 암호화 키(IPEK)는 DUKPT 프로세스에 사용되는 초기 키이며 기본 파생 키(BDK)에서 파생됩니다. 이 키에 대한 트랜잭션은 처리되지 않지만 트랜잭션에 사용될 미래 키를 도출하는 데 사용됩니다. IPEK는 잘못된 명칭입니다. 이 키는 데이터 암호화 및 Mac 키를 도출하는 데에도 사용될 수 있기 때문입니다. IPEK를 생성하기 위한 파생 방법은 X9. 24-1:2009 에 정의되어 있습니다. AES BDK를 사용하는 경우 X9. 24-1:2017 이 해당 표준이며 IPEK는 초기 키 (IK) 로 대체됩니다.

IWK

발급자 작업 키(IWK)는 일반적으로 발급자/발급자 프로세서와 네트워크(예: Visa 또는 Mastercard) 간에 데이터를 교환하는 데 사용되는 키입니다. 지금까지 IWK는 암호화에 3DES를 활용하고 TR31_P0_PIN_ENCRYPTION_KEY로 표시했습니다.

KEK

키 암호화 키(KEK)는 전송 또는 저장을 위해 다른 키를 암호화하는 데 사용되는 키입니다. 다른 키를 보호하는 데 사용되는 키는 표준에 따라 일반적으로 TR31_K0_KEY_ENCRYPTION_KEY를 KeyUsage 사용합니다. TR-31

PEK

PIN 암호화 키(PEK)는 두 당사자 간의 저장 또는 전송을 위해 PIN을 암호화하는 데 사용되는 일종의 작업 키입니다. IWK와 AWK는 핀 암호화 키를 구체적으로 사용하는 두 가지 예입니다. 이러한 키는 TR31_P0_PIN_ENCRYPTION_KEY로 표시됩니다.

PVK

PIN 검증 키(PVK)는 PVV와 같은 PIN 확인 값을 생성하는 데 사용되는 작업 키 유형입니다. 가장 일반적인 두 가지 종류는 IBM3624 오프셋 값을 생성하는 데 사용되는 TR31_V1_IBM3624_PIN_VERIFICATION_KEY와 Visa/ABA 검증 값에 사용되는 TR31_V2_VISA_PIN_VERIFICATION_KEY입니다.

기타 용어

ARQC

승인 요청 암호문(ARQC)은 EMV 표준 칩 카드(또는 이와 동등한 비접촉식 구현)를 통해 트랜잭션 시점에 생성되는 암호문입니다. 일반적으로 ARQC는 칩 카드로 생성되며 발급자 또는 해당 에이전트에게 전달되어 트랜잭션 시 확인됩니다.

DUKPT

트랜잭션별 파생된 고유 키(DUKPT)는 일반적으로 물리적 POS/POI에서 일회용 암호화 키 사용을 정의하는 데 사용되는 키 관리 표준입니다. 지금까지 DUKPT는 3DES를 암호화에 활용했습니다. DUKPT의 업계 표준은 ANSI X9.24-3-2017에 정의되어 있습니다.

EMV

[EMV](#) (원래는 유로페이, 마스터카드, 비자)는 결제 이해 관계자와 협력하여 상호 운용 가능한 결제 표준 및 기술을 만드는 기술 기관입니다. 한 가지 표준은 사용되는 암호화를 포함하여 칩/비접촉식 카드와 상호 작용하는 결제 단말기에 대한 표준입니다. EMV 키 파생이란 다음과 같은 초기 키 세트를 기반으로 각 결제 카드에 대해 고유한 키를 생성하는 방법을 말합니다. [IMK](#)

HSM

하드웨어 보안 모듈(HSM)은 암호화 작업(예: 암호화, 해독, 디지털 서명)과 이러한 작업에 사용되는 기본 키를 보호하는 물리적 장치입니다.

KCV

키 확인 값(KCV)은 실제 키 자료에 액세스하지 않고도 키를 서로 비교하는 데 주로 사용되는 다양한 체크섬 방법을 말합니다. KCV는 무결성 검증(특히 키 교환 시)에도 사용되었지만, 이제 이 역할은 [TR-31](#)와 같은 키 블록 형식의 일부로 포함되어 있습니다. TDES 키의 경우 KCV는 검사할 키와 함께 각각 값이 0인 8바이트를 암호화하고 암호화된 결과 중 가장 높은 순위의 3바이트를 유지하는 방식으로 계산됩니다. AES 키의 경우 KCV는 입력 데이터가 0의 16바이트이고 암호화된 결과 중 가장 높은 순위의 3바이트를 유지하는 CMAC 알고리즘을 사용하여 계산됩니다.

KDH

키 분포 호스트(KDH)는 [TR-34](#)와 같은 키 교환 프로세스를 통해 키를 전송하는 장치 또는 시스템입니다. AWS 페이먼트 크립토키프로그래피에서 키를 보내는 경우 이를 KDH로 간주합니다.

KIF

키 삽입 기능(KIF)은 암호화 키를 로드하는 것을 포함하여 결제 단말기를 초기화하는 데 사용되는 보안 시설입니다.

KRD

키 수신 장치(KRD)는 [TR-34](#)와 같은 키 교환 프로세스에서 키를 수신하는 장치입니다. AWS 결제 암호화로 키를 보내는 경우 이를 KRD로 간주합니다.

KSN

키 일련 번호(KSN)는 트랜잭션별 고유한 암호화 키를 생성하기 위해 DUKPT 암호화/해독에 입력값으로 사용되는 값입니다. KSN은 일반적으로 BDK 식별자, 존고유 단말기 ID, 특정 결제 단말기에서 전환이 처리될 때마다 증가하는 트랜잭션 카운터로 구성됩니다.

PAN

기본 계좌 번호(PAN)는 신용카드나 직불카드와 같은 계좌의 고유 식별자입니다. 일반적으로 길이는 13-19자리입니다. 처음 6~8자리는 네트워크와 발급 은행을 식별합니다.

PIN 블록

처리 또는 전송 중에 PIN을 포함한 기타 데이터 요소를 포함하는 데이터 블록입니다. PIN 블록 형식은 PIN 블록의 내용과 PIN 블록을 처리하여 PIN을 검색하는 방법을 표준화합니다. 대부분의 PIN 블록은 PIN, PIN 길이로 구성되며 PAN의 일부 또는 전체를 포함하는 경우가 많습니다. AWS 결제 암호화는 ISO 9564-1 형식 0, 1, 3, 4를 지원합니다. AES 키에는 형식 4가 필요합니다. PIN을 확인하거나 변환할 때는 수신 또는 발신 데이터의 PIN 블록을 지정해야 합니다.

POI

Point of Sale(POS)와 동의어로 자주 사용되는 Point of Interaction(POI)은 카드 소유자가 결제 보안 인증 정보를 제시하기 위해 상호 작용하는 하드웨어 디바이스입니다. POI의 예로는 가맹점에 있는 실제 단말기가 있습니다. 인증된 PCI PTS POI 단말기 목록은 [PCI 웹사이트](#)를 참조하세요.

PSN

PAN 시퀀스 번호(PSN)는 동일한 [PAN](#)으로 발급된 여러 카드를 구분하는 데 사용되는 숫자 값입니다.

퍼블릭 키

비대칭 암호(RSA)를 사용하는 경우 퍼블릭 키는 퍼블릭-프라이빗 키 페어의 퍼블릭 구성 요소입니다. 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)를 사용하는 경우 프라이빗 키는 퍼블릭-프라이빗 키 페어의 프라이빗 구성 요소입니다. 프라이빗 키는 데이터를 복호화하거나 디지털 서명을 생성하는 데 사용됩니다. 대칭 AWS

결제 암호화 키와 마찬가지로 개인 키는 HSM에서 안전하게 생성합니다. HSM의 휘발성 메모리에만 해독되며 암호화 요청을 처리하는 데 필요한 시간 동안만 해독됩니다.

PVV

PIN 검증 값(PVV)은 [카드 번호](#) 및 PIN과 같은 일련의 입력에서 알고리즘으로 파생된 값으로, 후속 검증에 사용할 수 있는 값을 생성합니다. 이러한 체계 중 하나는 Visa PVV(ABA 방법이라고도 함)로 알려져 있지만 모든 네트워크의 PIN에 사용됩니다.

RSA 래핑/언랩

RSA 래핑은 비대칭 키를 사용하여 대칭 키 (예: TDES 키) 를 래핑하여 다른 시스템으로 전송합니다. 일치하는 개인 키가 있는 시스템만 페이로드를 해독하고 대칭 키를 로드할 수 있습니다. 반대로 RSA는 래핑을 해제하면 RSA로 암호화된 키를 안전하게 복호화한 다음 이 키를 결제 암호화로 로드합니다. AWS RSA 래핑은 키를 교환하는 저수준 방법이며 키를 키 블록 형식으로 전송하지 않으며 전송 당사자의 페이로드 서명을 활용하지 않습니다. 프로비던스와 키 속성이 변경되지 않았는지 확인하려면 대체 제어를 고려해야 합니다.

TR-34 역시 내부적으로 RSA를 사용하지만 별도의 형식이며 상호 운용이 불가능합니다.

TR-31

TR-31(정식 명칭은 ANSI X9 TR 31)는 키 데이터 자체와 동일한 데이터 구조에서 키 속성을 정의할 수 있도록 미국국립표준협회(ANSI)에서 정의한 키 블록 형식입니다. TR-31 키 블록 형식은 키에 연결되어 함께 유지되는 키 속성 세트를 정의합니다. AWS 결제 암호화는 적절한 키 분리 및 키 용도를 보장하기 위해 가능한 한 TR-31 표준 용어를 사용합니다. TR-31은 [ANSI X9.143-2022](#)로 대체되었습니다.

TR-34

TR-34 버전은 비대칭 기법 (예: RSA) 을 사용하여 대칭 키 (예: 3DES 및 AES) 를 안전하게 배포하는 프로토콜을 설명하는 ANSI X9.24-2의 구현입니다. AWS 결제 암호화는 TR-34 방식을 사용하여 키를 안전하게 가져오고 내보낼 수 있도록 합니다.

관련 서비스

[AWS Key Management Service](#)

AWS키 관리 서비스(AWS KMS)는 데이터 보호에 사용하는 암호화 키를 쉽게 생성하고 제어할 수 있게 해주는 관리형 서비스입니다. AWS KMS는 하드웨어 보안 모듈(HSM)을 사용하여 AWS KMS 키를 보호하고 검증합니다.

AWS CloudHSM

AWS CloudHSM는 고객에게 AWS 클라우드의 전용 범용 HSM 인스턴스를 제공합니다. AWS CloudHSM는 키 생성, 데이터 서명 또는 데이터 암호화 및 해독과 같은 다양한 암호화 기능을 제공할 수 있습니다.

자세한 정보

- AWS 결제 암호화에 사용하는 용어 및 개념에 대한 자세한 내용은 [AWS 결제 암호화 개념](#)을 참조하세요.
- AWS 결제 암호화 컨트롤 플레인 API에 대한 자세한 내용은 [AWS 결제 암호화 컨트롤 플레인 API 참조](#)를 참조하세요.
- AWS 결제 암호화 데이터 영역 API에 대한 자세한 내용은 [AWS 결제 암호화 데이터 영역 API 참조](#)를 참조하세요.
- AWS 결제 암호화에서 어떻게 암호화를 이용하고 AWS 결제 암호화 키를 보호하는지 기술적으로 자세히 알아보려면 [암호화 세부 정보](#)를 참조하세요.

엔드포인트 대상 AWS Payment Cryptography

프로그래밍 방식으로 AWS Payment Cryptography 연결하려면 서비스 진입점의 URL인 엔드포인트를 사용합니다. AWS SDK와 명령줄 도구는 요청의 리전 컨텍스트를 AWS 리전 기반으로 서비스의 기본 엔드포인트를 자동으로 사용하므로 일반적으로 이러한 값을 명시적으로 설정할 필요가 없습니다. 필요한 경우 API 요청에 다른 엔드포인트를 지정할 수 있습니다.

컨트롤 플레인 엔드포인트

지역명	리전	엔드포인트	프로토콜
미국 동부(버지니아 북부)	us-east-1	controlplane.payment-cryptography.us-east-1.amazonaws.com	HTTPS
미국 동부(오하이오)	us-east-2	컨트롤플레인.payment-cryptography.us-east-2.amazonaws.com	HTTPS
미국 서부(오레곤)	us-west-2	controlplane.payment-cryptography.us-west-2.amazonaws.com	HTTPS

데이터 영역 엔드포인트

지역명	리전	엔드포인트	프로토콜
미국 동부(버지니아 북부)	us-east-1	dataplane.payment-cryptography.us-east-1.amazonaws.com	HTTPS
미국 동부(오하이오)	us-east-2	데이터플레인. payment-cryptography.us-east-2.amazonaws.com	HTTPS
미국 서부(오레곤)	us-west-2	dataplane.payment-cryptography.us-west-2.amazonaws.com	HTTPS

AWS 결제 암호화 시작하기

AWS 결제 암호화를 시작하려면 먼저 키를 만든 다음 다양한 암호화 작업에 사용하는 것이 좋습니다. 아래 자습서에서는 CVV2 값을 생성/확인하는 데 사용할 키를 생성하는 간단한 사용 사례를 제공합니다. 다른 예제를 시도하고 AWS 내의 배포 패턴을 살펴보고 싶다면 다음 [AWS 결제 암호화 워크숍](#)을 시도하거나 [Github](#)에서 제공되는 샘플 프로젝트를 살펴보세요.

이 자습서에서는 단일 키를 만들고 키를 사용하여 암호화 작업을 수행하는 방법을 안내합니다. 그런 다음 더 이상 필요하지 않을 경우 키를 삭제하면 키 수명 주기가 완료됩니다.

주제

- [필수 조건](#)
- [1단계: 키 생성](#)
- [2단계: 키를 사용하여 CVV2 값 생성](#)
- [3단계: 2단계에서 생성된 값 확인](#)
- [4단계: 네거티브 테스트 수행](#)
- [5단계: \(선택 사항\) 정리](#)

필수 조건

시작하기 전에 다음을 확인하세요.

- 서비스에 액세스할 권한이 있음. 자세한 내용을 알아보려면 [IAM 정책](#)을 참조하세요.
- [AWS CLI](#)이 설치되어 있음. [AWSSDK](#) 또는 [AWSAPI](#)를 사용하여 AWS 결제 암호화에 액세스할 수도 있지만 이 자습서의 지침에서는 AWS CLI를 사용합니다.

1단계: 키 생성

첫 번째 단계는 키를 만드는 것입니다. 이 자습서에서는 CVV/CVV2 값을 생성하고 확인하기 위한 [CVK](#) 이종 길이 3DES(2KEY TDES) 키를 생성합니다.

```
$ aws payment-cryptography create-key \
  --exportable
  --key-attributes KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,\
```

```
KeyClass=SYMMETRIC_KEY,\
KeyModesOfUse='{Generate=true,Verify=true}'
```

응답은 후속 호출을 위한 ARN과 키 검사 값(KCV)을 포함한 요청 파라미터를 다시 반영합니다.

```
{
  "Key": {
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi",
    "KeyAttributes": {
      "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyAlgorithm": "TDES_2KEY",
      "KeyModesOfUse": {
        "Encrypt": false,
        "Decrypt": false,
        "Wrap": false,
        "Unwrap": false,
        "Generate": true,
        "Sign": false,
        "Verify": true,
        "DeriveKey": false,
        "NoRestrictions": false
      }
    },
    "KeyCheckValue": "CADDA1",
    "KeyCheckValueAlgorithm": "ANSI_X9_24",
    "Enabled": true,
    "Exportable": true,
    "KeyState": "CREATE_COMPLETE",
    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
    "CreateTimestamp": "2023-06-05T06:41:46.648000-07:00",
    "UsageStartTimestamp": "2023-06-05T06:41:46.626000-07:00"
  }
}
```

키를 나타내는 KeyArn에 주의하세요(예: arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi). 다음 단계에서 이 작업을 수행합니다.

2단계: 키를 사용하여 CVV2 값 생성

이 단계에서는 1단계의 키를 사용하여 지정된 [PAN](#) 및 만료일에 대한 CVV2를 생성합니다.

```
$ aws payment-cryptography-data generate-card-validation-data \
  --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
  tqv5yij6wtxx64pi \
  --primary-account-number=171234567890123 \
  --generation-attributes CardVerificationValue2={CardExpiryDate=0123}
```

```
{
  "CardDataGenerationKeyCheckValue": "CADD1",
  "CardDataGenerationKeyIdentifier": "arn:aws:payment-cryptography:us-
  east-2:111122223333:key/tqv5yij6wtxx64pi",
  "CardDataType": "CARD_VERIFICATION_VALUE_2",
  "CardDataValue": "144"
}
```

cardDataValue(이 경우 3자리 숫자 144)를 기록해 두세요. 다음 단계에서 이 작업을 수행합니다.

3단계: 2단계에서 생성된 값 확인

이 예시에서는 1단계에서 생성한 키를 사용하여 2단계의 CVV2를 검증합니다.

다음 명령을 실행하여 CVV2를 검증합니다.

```
$ aws payment-cryptography-data verify-card-validation-data \
  --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
  tqv5yij6wtxx64pi \
  --primary-account-number=171234567890123 \
  --verification-attributes CardVerificationValue2={CardExpiryDate=0123} \
  --validation-data 144
```

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
  tqv5yij6wtxx64pi",
  "KeyCheckValue": "CADD1"
}
```

이 서비스는 CVV2를 검증했음을 나타내는 HTTP 응답 200을 반환합니다.

4단계: 네거티브 테스트 수행

이 단계에서는 CVV2가 올바르지 않고 검증되지 않는 음성 테스트를 생성합니다. 1단계에서 만든 키를 사용하여 잘못된 CVV2를 검증하려고 합니다. 예를 들어 카드 소유자가 결제 시 잘못된 CVV2를 입력한 경우, 이 작업은 필요한 작업입니다.

```
$ aws payment-cryptography-data verify-card-validation-data \
  --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
  tqv5yij6wtxx64pi \
  --primary-account-number=171234567890123 \
  --verification-attributes CardVerificationValue2={CardExpiryDate=0123} \
  --validation-data 999
```

```
Card validation data verification failed.
```

이 서비스는 “카드 검증 데이터 확인에 실패했습니다”라는 메시지와 INVALID_VALIDATION_DATA라는 사유와 함께 400의 HTTP 응답을 반환합니다.

5단계: (선택 사항) 정리

이제 1단계에서 생성한 키를 삭제할 수 있습니다. 복구할 수 없는 변경 사항을 최소화하기 위한 기본 키 삭제 기간은 7일입니다.

```
$ aws payment-cryptography delete-key \
  --key-identifier=arn:aws:payment-cryptography:us-east-2:111122223333:key/
  tqv5yij6wtxx64pi
```

```
{
  "Key": {
    "CreateTimestamp": "2022-10-27T08:27:51.795000-07:00",
    "DeletePendingTimestamp": "2022-11-03T13:37:12.114000-07:00",
    "Enabled": true,
    "Exportable": true,
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
    tqv5yij6wtxx64pi",
    "KeyAttributes": {
      "KeyAlgorithm": "TDES_3KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyModesOfUse": {
```

```
        "Decrypt": true,
        "DeriveKey": false,
        "Encrypt": true,
        "Generate": false,
        "NoRestrictions": false,
        "Sign": false,
        "Unwrap": true,
        "Verify": false,
        "Wrap": true
    },
    "KeyUsage": "TR31_P0_PIN_ENCRYPTION_KEY"
},
"KeyCheckValue": "CADD1",
"KeyCheckValueAlgorithm": "ANSI_X9_24",
"KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
"KeyState": "DELETE_PENDING",
"UsageStartTimestamp": "2022-10-27T08:27:51.753000-07:00"
}
}
```

출력의 두 필드를 기록해 두세요. `deletePendingTimestamp`는 기본적으로 향후 7일로 설정되어 있습니다. `keyState`는 `DELETE_PENDING`로 설정되어 있습니다. [restore-key](#)를 호출하여 예정된 삭제 시간 전에 언제든지 이 삭제를 취소할 수 있습니다.

키 관리

결제 암호화를 시작하려면 AWS 결제 암호화 키를 AWS 생성해야 합니다.

이 섹션의 항목에서는 생성부터 삭제까지 다양한 AWS 결제 암호화 키 유형을 생성하고 관리하는 방법을 설명합니다. 키 생성, 편집 및 보기, 키 태그 지정, 키 별칭 생성, 키 구성 요소 교체 및 비활성화에 대한 주제 등을 다룹니다.

주제

- [키 생성](#)
- [키 나열](#)
- [키 활성화 및 비활성화](#)
- [키 삭제](#)
- [키 가져오기 및 내보내기](#)
- [별칭 사용](#)
- [키 가져오기](#)
- [키 태그 지정](#)
- [AWS 결제 암호화 키의 주요 속성 이해](#)

키 생성

CreateKey API 작업을 사용하여 AWS 결제 암호화 키를 생성할 수 있습니다. 이 과정에서 키 알고리즘 (예: TDES_3KEY), (예: TR31_P0_PIN_ENCRYPTION_KEY), 허용된 작업 KeyUsage (예: 암호화, 서명), 내보내기 가능 여부 등 키의 다양한 속성 또는 결과 출력을 지정합니다. 결제 암호화 키를 생성한 후에는 이러한 속성을 변경할 수 없습니다. AWS

예제

- [2KEY TDES 키 생성](#)
- [핀 암호화 키 생성](#)
- [비대칭 \(RSA\) 키 생성](#)
- [PIN 검증 값 \(PVV\) 키 생성](#)

2KEY TDES 키 생성

Example

이 명령을 사용하여 CVV/CVV2 값을 생성하고 확인하기 위한 2키 TDES 키를 생성합니다. 응답은 KCV(Key Check Value) 뿐만 아니라 후속 호출을 위한 ARN을 포함한 요청 파라미터를 다시 반영합니다.

```
$ aws payment-cryptography create-key --exportable --key-attributes
  KeyAlgorithm=TDES_2KEY,\
  KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY, \
  KeyModesOfUse='{Generate=true,Verify=true}'
```

```
{
  "Key": {
    "CreateTimestamp": "2022-10-26T16:04:11.642000-07:00",
    "Enabled": true,
    "Exportable": true,
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/hjprdg5o4jtg5tw",
    "KeyAttributes": {
      "KeyAlgorithm": "TDES_2KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyModesOfUse": {
        "Decrypt": false,
        "DeriveKey": false,
        "Encrypt": false,
        "Generate": true,
        "NoRestrictions": false,
        "Sign": false,
        "Unwrap": false,
        "Verify": true,
        "Wrap": false
      }
    },
    "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY"
  },
  "KeyCheckValue": "B72F",
  "KeyCheckValueAlgorithm": "ANSI_X9_24",
  "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
  "KeyState": "CREATE_COMPLETE",
```



```

    "UsageStartTimestamp": "2022-10-26T16:04:11.559000-07:00"
  }
}

```

핀 암호화 키 생성

Example PEK(핀 암호화 키) 생성

이 명령을 사용하여 PIN 값(핀 암호화 키라고도 함)을 암호화하기 위한 3KEY TDES 키를 생성합니다. 이 키는 저장 PIN을 보호하거나 확인 시도 중 (예: 트랜잭션 중) 제공된 PIN을 해독하는 데 사용할 수 있습니다. 응답은 KCV(Key Check Value) 뿐만 아니라 후속 호출을 위한 ARN을 포함한 요청 파라미터를 다시 반영합니다.

```

$ aws payment-cryptography create-key --exportable --key-attributes \
    KeyAlgorithm=TDES_3KEY,KeyUsage=TR31_P0_PIN_ENCRYPTION_KEY, \
    KeyClass=SYMMETRIC_KEY,/
KeyModesOfUse=' {Encrypt=true,Decrypt=true,Wrap=true,Unwrap=true}'

```

```

{
  "Key": {
    "CreateTimestamp": "2022-10-27T08:27:51.795000-07:00",
    "Enabled": true,
    "Exportable": true,
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h",
    "KeyAttributes": {
      "KeyAlgorithm": "TDES_3KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyModesOfUse": {
        "Decrypt": true,
        "DeriveKey": false,
        "Encrypt": true,
        "Generate": false,
        "NoRestrictions": false,
        "Sign": false,
        "Unwrap": true,
        "Verify": false,
        "Wrap": true
      }
    }
  }
}

```

```

    },
    "KeyUsage": "TR31_P0_PIN_ENCRYPTION_KEY"
  },
  "KeyCheckValue": "9CA6",
  "KeyCheckValueAlgorithm": "ANSI_X9_24",
  "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
  "KeyState": "CREATE_COMPLETE",
  "UsageStartTimestamp": "2022-10-27T08:27:51.753000-07:00"
}
}

```

비대칭 (RSA) 키 생성

Example

이 예에서는 새로운 비대칭 RSA 2048비트 키 쌍을 생성합니다. 일치하는 퍼블릭 키뿐만 아니라 새 프라이빗 키도 생성됩니다. 공개 키는 API를 사용하여 검색할 수 있습니다. [getPublicCertificate](#)

```

$ aws payment-cryptography create-key --exportable \
--key-attributes
KeyAlgorithm=RSA_2048,KeyUsage=TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION, \
KeyClass=ASYMMETRIC_KEY_PAIR,KeyModesOfUse='{Encrypt=true,
Decrypt=True,Wrap=True,Unwrap=True}'

```

```

{
  "Key": {
    "CreateTimestamp": "2022-11-15T11:15:42.358000-08:00",
    "Enabled": true,
    "Exportable": true,
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
nsq2i3mbg6sn775f",
    "KeyAttributes": {
      "KeyAlgorithm": "RSA_2048",
      "KeyClass": "ASYMMETRIC_KEY_PAIR",
      "KeyModesOfUse": {
        "Decrypt": true,
        "DeriveKey": false,
        "Encrypt": true,
        "Generate": false,
        "NoRestrictions": false,

```

```

        "Sign": false,
        "Unwrap": true,
        "Verify": false,
        "Wrap": true
    },
    "KeyUsage": "TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION"
},
"KeyCheckValue": "40AD487F",
"KeyCheckValueAlgorithm": "CMAC",
"KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
"KeyState": "CREATE_COMPLETE",
"UsageStartTimestamp": "2022-11-15T11:15:42.182000-08:00"
}
}

```

PIN 검증 값 (PVV) 키 생성

Example

이 명령을 사용하여 PVV 값(PIN 검증 값이라고 함)을 생성하기 위한 3KEY TDES 키를 생성합니다. 이 키를 사용하여 이후에 계산된 PVV와 비교할 수 있는 PVV 값을 생성할 수 있습니다. 응답은 KCV(Key Check Value) 뿐만 아니라 후속 호출을 위한 ARN을 포함한 요청 파라미터를 다시 반영합니다.

```

$ aws payment-cryptography create-key --exportable/
--key-attributes KeyAlgorithm=TDES_3KEY,KeyUsage=TR31_V2_VISA_PIN_VERIFICATION_KEY,/
KeyClass=SYMMETRIC_KEY,KeyModesOfUse='{Generate=true,Verify=true}'

```

```

{
  "Key": {
    "CreateTimestamp": "2022-10-27T10:22:59.668000-07:00",
    "Enabled": true,
    "Exportable": true,
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
j4u4cmnzkelhc6yb",
    "KeyAttributes": {
      "KeyAlgorithm": "TDES_3KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyModesOfUse": {
        "Decrypt": false,
        "DeriveKey": false,
        "Encrypt": false,

```

```

        "Generate": true,
        "NoRestrictions": false,
        "Sign": false,
        "Unwrap": false,
        "Verify": true,
        "Wrap": false
    },
    "KeyUsage": "TR31_V2_VISA_PIN_VERIFICATION_KEY"
},
"KeyCheckValue": "5132",
"KeyCheckValueAlgorithm": "ANSI_X9_24",
"KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
"KeyState": "CREATE_COMPLETE",
"UsageStartTimestamp": "2022-10-27T10:22:59.614000-07:00"
}
}

```

키 나열

키 나열은 해당 계정 및 지역의 발신자가 액세스할 수 있는 키 목록을 제공합니다.

Example

```
$ aws payment-cryptography list-keys
```

```

{"Keys": [
  {
    "CreateTimestamp": "2022-10-12T10:58:28.920000-07:00",
    "Enabled": false,
    "Exportable": true,
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/alsuwfxug3pgy6xh",
    "KeyAttributes": {
      "KeyAlgorithm": "TDES_3KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyModesOfUse": {
        "Decrypt": true,
        "DeriveKey": false,
        "Encrypt": true,

```

```

        "Generate": false,
        "NoRestrictions": false,
        "Sign": false,
        "Unwrap": true,
        "Verify": false,
        "Wrap": true
    },
    "KeyUsage": "TR31_P1_PIN_GENERATION_KEY"
},
"KeyCheckValue": "369D",
"KeyCheckValueAlgorithm": "ANSI_X9_24",
"KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
"KeyState": "CREATE_COMPLETE",
"UsageStopTimestamp": "2022-10-27T14:19:42.488000-07:00"
}
]
}

```

키 활성화 및 비활성화

AWS 결제 암호화 키를 비활성화했다가 다시 활성화할 수 있습니다. 키를 생성하면 필터가 기본적으로 활성화됩니다. 키를 비활성화하면 다시 활성화할 때까지 어떠한 [암호화 작업](#)에서도 사용할 수 없습니다. 사용 시작/중지 명령은 즉시 적용되므로 변경하기 전에 사용량을 검토하는 것이 좋습니다. 선택적 timestamp 파라미터를 사용하여 향후에 적용할 변경 사항(사용 시작 또는 중지)을 설정할 수도 있습니다.

일시적이고 쉽게 취소할 수 있으므로 AWS 결제 암호화 키를 비활성화하는 것이 파괴적이고 되돌릴 수 없는 AWS 결제 암호화 키를 삭제하는 것보다 더 안전한 대안입니다. AWS Payment Cryptography 키를 삭제하려는 경우 먼저 해당 키를 비활성화하고 앞으로는 데이터를 암호화하거나 복호화하는 데 이 키를 사용할 필요가 없도록 하십시오.

주제

- [키 사용 시작](#)
- [키 사용 중지](#)

키 사용 시작

암호화 작업에 키를 사용하려면 키 사용을 활성화해야 합니다. 키가 활성화되지 않은 경우 이 작업을 사용하여 키를 사용할 수 있게 만들 수 있습니다. 이 UsageStartTimeStamp 필드는 키가 활성화된

시점 또는 활성화될 시점을 나타냅니다. 이는 활성화된 토큰의 경우 과거이며, 활성화가 보류 중인 경우 미래입니다.

Example

이 예시에서는 키 사용을 위해 키를 활성화하도록 요청합니다. 응답에는 키 정보가 포함되며 활성화 플래그가 true로 전환되었습니다. 이는 리스트 키 응답 개체에도 반영됩니다.

```
$ aws payment-cryptography start-key-usage --key-identifier "arn:aws:payment-cryptography:us-east-2:111122223333:key/alsuwxug3pgy6xh"
```

```
{
  "Key": {
    "CreateTimestamp": "2022-10-12T10:58:28.920000-07:00",
    "Enabled": true,
    "Exportable": true,
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/alsuwxug3pgy6xh",
    "KeyAttributes": {
      "KeyAlgorithm": "TDES_3KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyModesOfUse": {
        "Decrypt": true,
        "DeriveKey": false,
        "Encrypt": true,
        "Generate": false,
        "NoRestrictions": false,
        "Sign": false,
        "Unwrap": true,
        "Verify": false,
        "Wrap": true
      }
    },
    "KeyUsage": "TR31_P1_PIN_GENERATION_KEY"
  },
  "KeyCheckValue": "369D",
  "KeyCheckValueAlgorithm": "ANSI_X9_24",
  "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
  "KeyState": "CREATE_COMPLETE",
  "UsageStartTimestamp": "2022-10-27T14:09:59.468000-07:00"
}
```

키 사용 중지

더 이상 키를 사용하지 않으려는 경우 키 사용을 중지하여 추가 암호화 작업을 방지할 수 있습니다. 이 작업은 영구적이지 않으므로 [키 사용 시작하기](#)를 사용하여 되돌릴 수 있습니다. 향후에 키가 비활성화 되도록 설정할 수도 있습니다. 이 UsageStopTimestamp 필드는 키가 언제 비활성화되었는지 또는 비활성화될 것인지를 나타냅니다.

Example

이 예시에서는 향후에 키 사용을 중지하도록 요청합니다. 실행 후에는 [키 사용 시작](#)을 통해 다시 활성화하지 않는 한 이 키를 암호화 작업에 사용할 수 없습니다. 응답에는 키 정보가 포함되고 활성화 플래그는 false로 전환되었습니다. 이는 리스트 키 응답 개체에도 반영됩니다.

```
$ aws payment-cryptography stop-key-usage --key-identifier "arn:aws:payment-cryptography:us-east-2:111122223333:key/alsuwfxug3pgy6xh"
```

```
{
  "Key": {
    "CreateTimestamp": "2022-10-12T10:58:28.920000-07:00",
    "Enabled": false,
    "Exportable": true,
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/alsuwfxug3pgy6xh",
    "KeyAttributes": {
      "KeyAlgorithm": "TDES_3KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyModesOfUse": {
        "Decrypt": true,
        "DeriveKey": false,
        "Encrypt": true,
        "Generate": false,
        "NoRestrictions": false,
        "Sign": false,
        "Unwrap": true,
        "Verify": false,
        "Wrap": true
      },
      "KeyUsage": "TR31_P1_PIN_GENERATION_KEY"
    },
    "KeyCheckValue": "369D",
    "KeyCheckValueAlgorithm": "ANSI_X9_24",
    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
```

```

    "KeyState": "CREATE_COMPLETE",
    "UsageStopTimestamp": "2022-10-27T14:09:59.468000-07:00"
  }
}

```

키 삭제

AWS 결제 암호화 키를 삭제하면 키와 관련된 키 자료와 모든 메타데이터가 삭제되며 결제 암호화 외부에서 키 사본을 사용할 수 없는 한 되돌릴 수 없습니다. AWS 키가 삭제된 후에는 해당 키로 암호화된 데이터를 더 이상 해독할 수 없습니다. 즉 데이터를 복구할 수 없게 될 수 있습니다. 키를 더 이상 사용할 필요가 없고 다른 당사자가 이 키를 사용하지 않는다고 확신하는 경우에만 키를 삭제해야 합니다. 확실하지 않은 경우에는 삭제하는 대신 키를 비활성화하는 방법을 고려하세요. 비활성화된 키를 나중에 다시 사용해야 하는 경우 다시 활성화할 수 있지만 삭제된 AWS Payment Cryptography 키는 다른 소스에서 다시 가져올 수 없는 한 복구할 수 없습니다.

키를 삭제하기 전에 키가 더 이상 필요하지 않은지 확인해야 합니다. AWS 결제 암호화는 CVV2와 같은 암호화 작업의 결과를 저장하지 않으므로 영구 암호화 자료에 키가 필요한지 여부를 결정할 수 없습니다.

AWS 결제 암호화는 활성 AWS 계정에 속한 키를 삭제하도록 명시적으로 예약하고 필수 대기 기간이 만료되지 않는 한 절대 삭제하지 않습니다.

하지만 다음과 같은 이유 중 하나 이상의 이유로 AWS 결제 암호화 키를 삭제하도록 선택할 수 있습니다.

- 더 이상 필요하지 않은 키의 키 수명 주기를 완료하기 위해
- 사용하지 않는 결제 암호화 키를 유지 관리하는 데 따른 관리 오버헤드를 방지하기 AWS 위험입니다.

Note

결제 암호화 키를 [달거나 삭제하면](#) AWS 결제 암호화 AWS 계정키에 액세스할 수 없게 됩니다. 계정 폐쇄와 별도로 AWS 결제 암호화 키 삭제를 예약할 필요가 없습니다.

AWS 결제 암호화는 사용자가 결제 암호화 키 삭제를 예약한 시점과 AWS 결제 암호화 키가 실제로 삭제될 때 [AWS CloudTrail](#) 로그에 AWS 항목을 기록합니다.

대기 기간에 대해

키를 삭제하면 되돌릴 수 없으므로 AWS 결제 암호화를 사용하려면 3~180일의 대기 기간을 설정해야 합니다. 기본 대기 기간은 7일입니다.

그러나 실제 대기 기간은 예약한 대기 기간보다 최대 24시간 더 길어질 수 있습니다. AWS 결제 암호화 키가 삭제될 실제 날짜와 시간을 확인하려면 작업을 사용하십시오. GetKey 시간대를 기록하세요.

대기 기간 중에는 AWS 결제 암호화 키 상태 및 키 상태가 삭제 보류 중으로 표시됩니다.

Note

삭제 보류 중인 AWS 결제 암호화 키는 [암호화](#) 작업에 사용할 수 없습니다.

대기 기간이 끝나면 AWS 결제 암호화는 결제 암호화 키, 별칭 및 모든 관련 AWS 결제 암호화 메타데이터를 삭제합니다. AWS

대기 기간을 사용하여 현재 또는 미래에 AWS 결제 암호화 키가 필요하지 않은지 확인하세요. 대기 기간 동안 키가 필요한 경우 대기 기간이 종료되기 전에 키 삭제를 취소할 수 있습니다. 대기 기간이 종료된 후에는 키 삭제를 취소할 수 없고 서비스가 키를 삭제합니다.

Example

이 예시에서는 키 삭제를 요청합니다. 기본 키 정보 외에도 두 개의 관련 필드는 키 상태가 DELETE_PENDING으로 변경되었다는 것이며, 이 필드는 키가 현재 삭제될 예정인 시기를 deletePendingTimestamp 나타냅니다.

```
$ aws payment-cryptography delete-key \
    --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaif1lw2h
```

```
{
  "Key": {
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaif1lw2h",
    "KeyAttributes": {
```

```

    "KeyUsage": "TR31_V2_VISA_PIN_VERIFICATION_KEY",
    "KeyClass": "SYMMETRIC_KEY",
    "KeyAlgorithm": "TDES_3KEY",
    "KeyModesOfUse": {
      "Encrypt": false,
      "Decrypt": false,
      "Wrap": false,
      "Unwrap": false,
      "Generate": true,
      "Sign": false,
      "Verify": true,
      "DeriveKey": false,
      "NoRestrictions": false
    }
  },
  "KeyCheckValue": "",
  "KeyCheckValueAlgorithm": "ANSI_X9_24",
  "Enabled": false,
  "Exportable": true,
  "KeyState": "DELETE_PENDING",
  "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
  "CreateTimestamp": "2023-06-05T12:01:29.969000-07:00",
  "UsageStopTimestamp": "2023-06-05T14:31:13.399000-07:00",
  "DeletePendingTimestamp": "2023-06-12T14:58:32.865000-07:00"
}
}
}

```

Example

이 예시에서는 보류 중인 삭제가 취소됩니다. 성공적으로 완료되면 이전 일정에 따라 키가 더 이상 삭제되지 않습니다. 응답에는 기본 키 정보가 포함되며, 추가로 두 개의 관련 필드 (KeyState 및 deletePendingTimestamp)가 변경되었습니다. KeyState는 CREATE_COMPLETE 값으로 반환되고 DeletePendingTimestamp는 제거됩니다.

```
$ aws payment-cryptography restore-key --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qai1lw2h
```

```
{
  "Key": {
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qai1lw2h",

```

```

    "KeyAttributes": {
      "KeyUsage": "TR31_V2_VISA_PIN_VERIFICATION_KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyAlgorithm": "TDES_3KEY",
      "KeyModesOfUse": {
        "Encrypt": false,
        "Decrypt": false,
        "Wrap": false,
        "Unwrap": false,
        "Generate": true,
        "Sign": false,
        "Verify": true,
        "DeriveKey": false,
        "NoRestrictions": false
      }
    },
    "KeyCheckValue": "",
    "KeyCheckValueAlgorithm": "ANSI_X9_24",
    "Enabled": false,
    "Exportable": true,
    "KeyState": "CREATE_COMPLETE",
    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
    "CreateTimestamp": "2023-06-08T12:01:29.969000-07:00",
    "UsageStopTimestamp": "2023-06-08T14:31:13.399000-07:00"
  }
}

```

키 가져오기 및 내보내기

AWS 결제 암호화 키는 다른 솔루션에서 가져오거나 다른 솔루션 (예: 다른 HSM) 으로 내보낼 수 있습니다. 가져오기 및 내보내기 기능을 사용하여 서비스 공급자와 키를 교환하는 것이 일반적인 사용 사례입니다. 클라우드 서비스인 AWS Payment Cryptography는 키 관리에 대한 현대적이고 전자적인 접근 방식을 취하는 동시에 적용 가능한 규정 준수 및 제어를 유지할 수 있도록 지원합니다. 장기 목표는 종이 기반의 주요 구성 요소에서 벗어나 표준 기반의 전자식 키 교환 수단으로 전환하는 것입니다.

KEK(키 암호화 키) 교환

[AWS 결제 암호화는 잘 정립된 ANSI X9.24 TR-34 표준을 사용하여 초기 키 교환에 공개 키 암호화 \(RSA\) 를 사용하도록 권장합니다.](#) 이 초기 키 유형의 일반적인 이름으로는 KEK (키 암호화 키), ZMK (영역 마스터 키) 및 ZCMK (영역 제어 마스터 키) 가 있습니다. [시스템 또는 파트너가 아직 TR-34 지원을 지원하지 않는 경우 RSA 래핑/언랩 활용을 고려해 볼 수도 있습니다.](#)

모든 파트너가 전자 키 교환을 지원할 때까지 종이 키 구성 요소를 계속 처리해야 하는 경우 이를 위해 오프라인 HSM을 유지하는 것을 고려할 수 있습니다.

Note

자체 테스트 키를 가져오려면 [Github](#)의 샘플 프로젝트를 확인해 보세요. 다른 플랫폼에서 키를 가져오거나 내보내는 방법에 대한 지침은 해당 플랫폼의 사용 설명서를 참조하세요.

WK (Working Key) 교환

AWS 결제 암호화는 관련 업계 표준 ([ANSI X9.24 TR 31-2018](#)) 을 사용하여 작업 키를 교환합니다. TR-31 기준으로는 KEK가 이전에 교환된 적이 있다고 가정하겠습니다. 이는 키 자료를 키 유형 및 사용에 항상 암호적으로 바인딩하기 위한 PCI PIN의 요구 사항과 일치합니다. 작업 키에는 취득자 작업 키, 발급자 작업 키, BDK, IPEK 등 다양한 이름이 있습니다.

주제

- [키 가져오기](#)
- [키 내보내기](#)

키 가져오기

Important

예제에는 최신 버전의 AWS CLI V2가 필요할 수 있습니다. 시작하기 전에 [최신 버전으로 업그](#)레이드했는지 확인하십시오.

주제

- [대칭 키 가져오기](#)
- [비대칭\(RSA\) 키 가져오기](#)

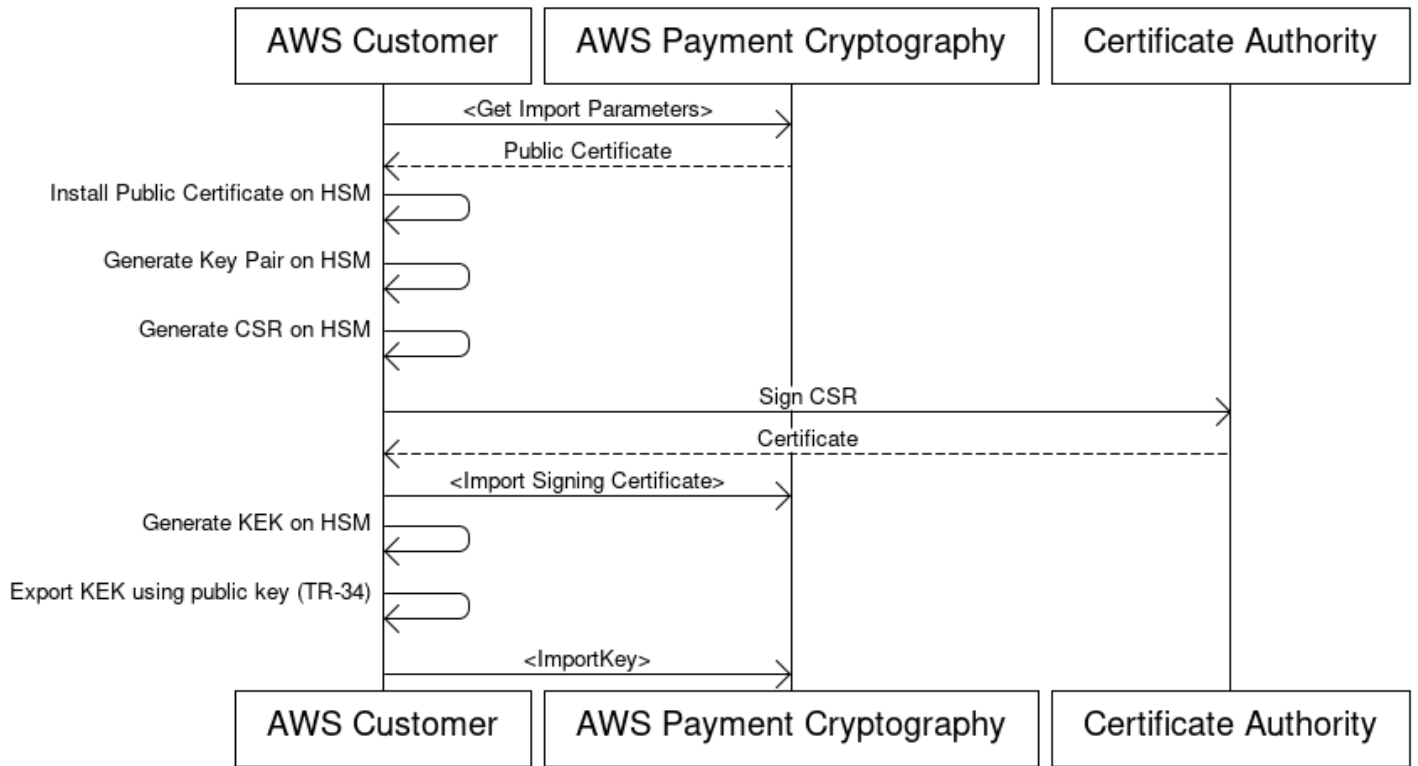
대칭 키 가져오기

주제

- [비대칭 기법을 사용한 키 가져오기\(TR-34\)](#)
- [비대칭 기법 \(RSA 언랩\) 을 사용하여 키 가져오기](#)
- [사전 설정된 키 교환 키\(TR-31\)를 사용하여 대칭 키를 가져옵니다.](#)

비대칭 기법을 사용한 키 가져오기(TR-34)

Key Encryption Key(KEK) Import Process



개요: TR-34 시스템은 RSA 비대칭 암호화를 활용하여 교환을 위한 대칭 키를 암호화하고 데이터 소스(서명)를 보장합니다. 이렇게 하면 래핑된 키의 기밀성(암호화)과 무결성(서명)이 모두 보장됩니다.

자체 키를 가져오려면 [Github](#)의 샘플 프로젝트를 확인하세요. 다른 플랫폼에서 키를 가져오거나 내보내는 방법에 대한 지침은 해당 플랫폼의 사용 설명서를 참조하세요.

1. 가져오기 초기화 명령을 호출하세요.

get-parameters-for-import를 호출하여 가져오기 프로세스를 초기화합니다. 이 API는 키 가져오기를 위한 키페어를 생성하고, 키에 서명하고, 인증서 및 인증서 루트를 반환합니다. 최종적으로 내보낼 키는 이 키를 사용하여 암호화해야 합니다. TR-34 용어로는 이를 KRD 인증서라고 합니다. 이러한 인증서는 유효기간이 짧으며 이 용도로만 사용된다는 점에 유의하세요.

2. 키 소스 시스템에 공개 인증서를 설치하세요.

HSM이 많은 경우 1단계에서 생성한 공개 인증서를 설치/로드/신뢰해야 이를 사용하여 키를 내보낼 수 있습니다.

3. 공개 키를 생성하고 AWS 결제 암호화에 인증서 루트를 제공하십시오.

전송된 페이로드의 무결성을 보장하기 위해 전송 당사자(키 분포 호스트 또는 KDH)가 서명합니다. 전송 당사자는 이러한 목적으로 공개 키를 생성한 다음 결제 암호화에 다시 제공할 수 있는 공개 키 인증서 (X509) 를 생성하기를 원할 것입니다. AWS AWS Private CA 인증서를 생성하는 옵션 중 하나이지만 사용되는 인증 기관에는 제한이 없습니다.

인증서를 얻었으면 `importKey` 명령 및 `KeyMaterialType of`를 사용하여 AWS Payment Cryptography에 루트 인증서를 로드하는 것이 좋습니다. `ROOT_PUBLIC_KEY_CERTIFICATE` `KeyUsageType TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`

4. 소스 시스템에서 키 내보내기

많은 HSM 및 관련 시스템에서는 TR-34 표준을 사용하여 키를 내보내는 기능을 지원합니다. 1단계의 퍼블릭 키는 KRD(암호화) 인증서로 지정하고 3단계의 키는 KDH(서명) 인증서로 지정하는 것이 좋습니다. AWS 결제 암호화로 가져오려면 형식을 TR-34.2012 비 CMS 2패스 형식으로 지정해야 합니다. 이 형식은 TR-34 디에볼트 형식이라고도 합니다.

5. 키 가져오기 호출

마지막 단계에서는 `of`를 사용하여 `ImportKey` API를 호출합니다. `KeyMaterialType TR34_KEY_BLOCK certificate-authority-public-key-identifier`는 3단계에서 가져온 루트 CA의 키ARN, `key-material`는 4단계의 래핑된 키 자료, `signing-key-certificate`는 3단계의 리프 인증서입니다. 또한 1단계에서 가져온 가져오기 토큰을 제공해야 합니다.

6. 가져온 키를 암호화 작업이나 후속 가져오기에 사용합니다.

`KeyUsage` 임포트한 키가 `TR31_K0_KEY_ENCRYPTION_KEY`인 경우, TR-31 키를 사용하여 이후에 키를 가져올 때 이 키를 사용할 수 있습니다. 키 유형이 다른 유형(예: `TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY`)인 경우 키를 암호화 작업에 직접 사용할 수 있습니다.

비대칭 기법 (RSA 언랩) 을 사용하여 키 가져오기

개요: AWS 결제 암호화는 TR-34 상황이 불가능할 때 키 교환을 위한 RSA 래핑/언랩핑을 지원합니다. 이 기법은 TR-34 방식과 마찬가지로 RSA 비대칭 암호화를 활용하여 대칭 키를 암호화하여 교환합니

다. 하지만 TR-34 방식과 달리 이 방법은 전송 당사자가 서명한 페이로드가 없습니다. 또한 이 RSA 랩 기법은 키 블록을 포함하지 않기 때문에 전송 중에 키 메타데이터의 무결성을 유지하지 못합니다.

Note

RSA 랩은 TDES 및 AES-128 키를 가져오거나 내보내는 데 사용할 수 있습니다.

1. 가져오기 초기화 명령을 호출하세요.

키를 `get-parameters-for-import` 호출하여 `KEY_CRYPTOGRAM`의 키 머티리얼 유형을 사용하여 가져오기 프로세스를 초기화합니다. `WrappingKeyAlgorithm` TDES 키를 교환할 때는 `RSA_2048`이 될 수 있습니다. TDES 또는 AES-128 키를 교환할 때는 `RSA_3072` 또는 `RSA_4096`을 사용할 수 있습니다. 이 API는 키 가져오기를 위한 키페어를 생성하고, 인증서 루트를 사용하여 키에 서명하고, 인증서와 인증서 루트를 모두 반환합니다. 최종적으로 내보낼 키는 이 키를 사용하여 암호화해야 합니다. 이러한 인증서는 유효기간이 짧으며 이 용도로만 사용된다는 점에 유의하세요.

```
$ aws payment-cryptography get-parameters-for-import --key-material-type
KEY_CRYPTOGRAM --wrapping-key-algorithm RSA_4096
```

```
{
  "ImportToken": "import-token-bwxli6ocftypneu5",
  "ParametersValidUntilTimestamp": 1698245002.065,
  "WrappingKeyCertificateChain": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0....",
  "WrappingKeyCertificate": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0....",
  "WrappingKeyAlgorithm": "RSA_4096"
}
```

2. 키 소스 시스템에 공개 인증서를 설치하세요.

많은 HSM의 경우 1단계에서 생성한 공개 인증서 (및/또는 해당 루트) 를 설치/로드/신뢰해야 이를 사용하여 키를 내보낼 수 있습니다.

3. 소스 시스템에서 키 내보내기

많은 HSM 및 관련 시스템에서는 RSA 랩을 사용하여 키를 내보내는 기능을 지원합니다. 1단계의 공개 키를 (암호화) `cert ()` 로 지정하는 것이 좋습니다. `WrappingKeyCertificate` 신뢰 체인이 필요한 경우 이 정보는 #1 단계의 응답 `WrappingKeyCertificateChain` 필드에 포함되어 있습니다. HSM에서

키를 내보낼 때는 형식을 RSA, 패딩 모드 = PKCS #1 v2.2 OAEP (SHA 256 또는 SHA 512 사용) 로 지정하는 것이 좋습니다.

4. 키 가져오기 호출

마지막 단계에서는 f를 사용하여 ImportKey API를 호출합니다. KeyMaterialType KeyMaterial 1 단계의 가져오기 토큰과 3단계의 key-material (래핑된 키 자료) 가 필요합니다. RSA 랩은 키 블록을 사용하지 않으므로 주요 매개 변수 (예: 키 사용) 를 제공해야 합니다.

```
$ cat import-key-ciphertext.json
{
  "KeyMaterial": {
    "KeyCiphertext": {
      "Exportable": true,
      "ImportToken": "import-token-bwxli6ocftypneu5",
      "KeyAttributes": {
        "KeyAlgorithm": "AES_128",
        "KeyClass": "SYMMETRIC_KEY",
        "KeyModesOfUse": {
          "Decrypt": true,
          "DeriveKey": false,
          "Encrypt": true,
          "Generate": false,
          "NoRestrictions": false,
          "Sign": false,
          "Unwrap": true,
          "Verify": false,
          "Wrap": true
        },
        "KeyUsage": "TR31_K0_KEY_ENCRYPTION_KEY"
      },
      "WrappedKeyCiphertext": "18874746731....",
      "WrappingSpec": "RSA_OAEP_SHA_256"
    }
  }
}
```

```
$ aws payment-cryptography import-key --cli-input-json file://import-key-ciphertext.json
```

```
{
```



```

    "Key": {
      "KeyOrigin": "EXTERNAL",
      "Exportable": true,
      "KeyCheckValue": "DA1ACF",
      "UsageStartTimestamp": 1697643478.92,
      "Enabled": true,
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaiifllw2h",
      "CreateTimestamp": 1697643478.92,
      "KeyState": "CREATE_COMPLETE",
      "KeyAttributes": {
        "KeyAlgorithm": "AES_128",
        "KeyModesOfUse": {
          "Encrypt": true,
          "Unwrap": true,
          "Verify": false,
          "DeriveKey": false,
          "Decrypt": true,
          "NoRestrictions": false,
          "Sign": false,
          "Wrap": true,
          "Generate": false
        },
        "KeyUsage": "TR31_K0_KEY_ENCRYPTION_KEY",
        "KeyClass": "SYMMETRIC_KEY"
      },
      "KeyCheckValueAlgorithm": "CMAC"
    }
  }
}

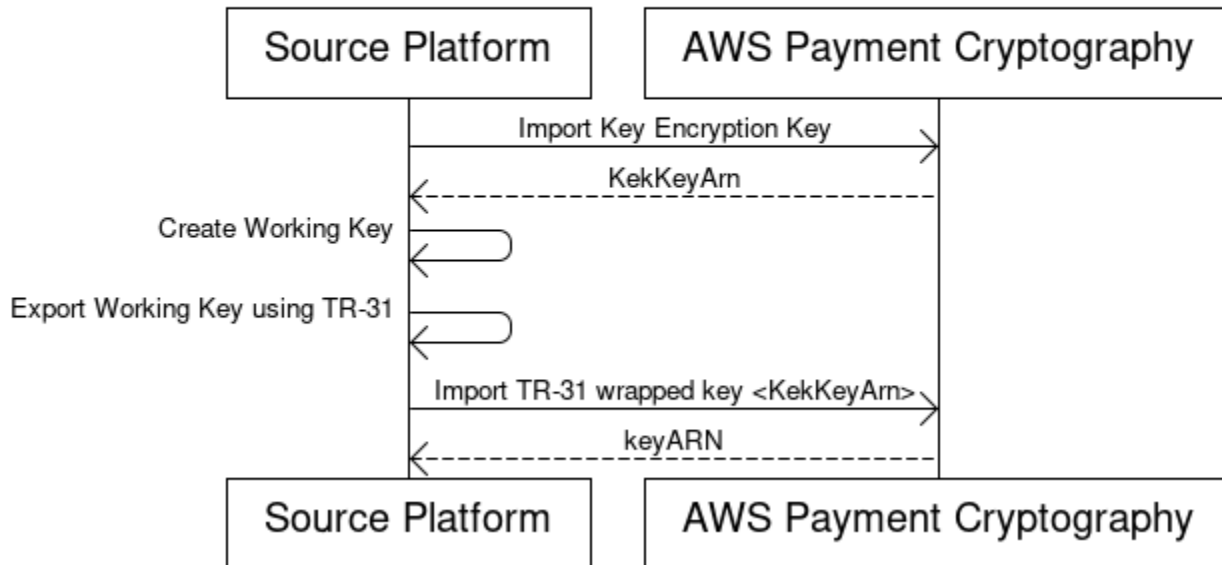
```

5. 가져온 키를 암호화 작업이나 후속 가져오기에 사용합니다.

가져온 KeyUsage 키가 TR31_K0_KEY_ENCRYPTION_KEY인 경우 TR-31 키를 사용하여 이후에 키를 가져올 때 이 키를 사용할 수 있습니다. 키 유형이 다른 유형(예: TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY)인 경우 키를 암호화 작업에 직접 사용할 수 있습니다.

사전 설정된 키 교환 키(TR-31)를 사용하여 대칭 키를 가져옵니다.

Import symmetric keys using a pre-established key exchange key (TR-31)



파트너가 여러 키를 교환하는 경우 (또는 키 교체를 지원하기 위해), 일반적으로 먼저 종이 키 구성 요소와 같은 기술을 사용하여 초기 키 암호화 키 (KEK) 를 교환하거나 TR-34 기반의 AWS 결제 암호화를 사용하여 교환합니다.

KEK가 설정되면 이 키를 사용하여 후속 키 (다른 KEK 포함) 를 전송할 수 있습니다. AWS 결제 암호화는 HSM 공급업체에서 널리 사용되고 널리 지원하는 ANSI TR-31 기반의 이러한 종류의 키 교환을 지원합니다.

1. KEK(키 암호화 키) 가져오기

KEK를 이미 가져왔고 keyARN(또는 keyAlias)을 사용할 수 있는 것으로 가정합니다.

2. 소스 플랫폼에서 키 생성

키가 아직 없는 경우 소스 플랫폼에서 키를 생성하세요. 반대로 AWS Payment Cryptography에서 키를 생성하고 대신 `export` 명령을 사용할 수 있습니다.

3. 소스 플랫폼에서 키 내보내기

내보낼 때는 내보내기 형식을 TR-31 형식으로 지정해야 합니다. 또한 소스 플랫폼은 내보낼 키와 사용할 키 암호화 키를 요청합니다.

4. 결제 암호화로 가져오기 AWS

ImportKey 명령을 호출할 때는 키 암호화 키의 KeyARN (또는 별칭) WrappingKeyIdentifier 이어야 하며 WrappedKeyBlock 소스 플랫폼의 출력이어야 합니다.

Example

```
$ aws payment-cryptography import-key \
    --key-material="Tr31KeyBlock={WrappingKeyIdentifier="arn:aws:payment-
cryptography:us-east-2:111122223333:key/ov6icy4ryas4zcza"},\
    WrappedKeyBlock="D0112B0AX00E00002E0A3D58252CB67564853373D1EBCC1E23B2ADE7B15E967CC27B85D599"
```

```
{
  "Key": {
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaiifllw2h",
    "KeyAttributes": {
      "KeyUsage": "TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyAlgorithm": "AES_128",
      "KeyModesOfUse": {
        "Encrypt": true,
        "Decrypt": true,
        "Wrap": true,
        "Unwrap": true,
        "Generate": false,
        "Sign": false,
        "Verify": false,
        "DeriveKey": false,
        "NoRestrictions": false
      }
    }
  },
  "KeyCheckValue": "0A3674",
  "KeyCheckValueAlgorithm": "CMAC",
  "Enabled": true,
  "Exportable": true,
  "KeyState": "CREATE_COMPLETE",
  "KeyOrigin": "EXTERNAL",
  "CreateTimestamp": "2023-06-02T07:38:14.913000-07:00",
  "UsageStartTimestamp": "2023-06-02T07:38:14.857000-07:00"
```

```
}
}
```

비대칭(RSA) 키 가져오기

RSA 퍼블릭 키 가져오기

AWS 결제 암호화는 X.509 인증서 형태의 공개 RSA 키 가져오기를 지원합니다. 인증서를 가져오려면 먼저 루트 인증서를 가져와야 합니다. 모든 인증서는 가져오는 시점에 만료되지 않아야 합니다. 인증서는 PEM 형식이어야 하며 base64로 인코딩되어야 합니다.

1. 루트 인증서를 결제 암호화로 가져오기 AWS

Example

```
$ aws payment-cryptography import-key \
  --key-material='{"RootCertificatePublicKey":{"KeyAttributes":
{"KeyAlgorithm":"RSA_2048", \
  "KeyClass":"PUBLIC_KEY", "KeyModesOfUse":{"Verify":
true},"KeyUsage":"TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"}, \
  "PublicKeyCertificate":"LS0tLS1CRudJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURKVENDQWcyZ0F3SUJBZ01CWKR
```

```
{
  "Key": {
    "CreateTimestamp": "2023-08-08T18:52:01.023000+00:00",
    "Enabled": true,
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
zabouwe3574jysdl",
    "KeyAttributes": {
      "KeyAlgorithm": "RSA_2048",
      "KeyClass": "PUBLIC_KEY",
      "KeyModesOfUse": {
        "Decrypt": false,
        "DeriveKey": false,
        "Encrypt": false,
        "Generate": false,
        "NoRestrictions": false,
        "Sign": false,
        "Unwrap": false,
```

```

        "Verify": true,
        "Wrap": false
    },
    "KeyUsage": "TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"
},
"KeyOrigin": "EXTERNAL",
"KeyState": "CREATE_COMPLETE",
"UsageStartTimestamp": "2023-08-08T18:52:01.023000+00:00"
}
}

```

2. 공개 키 인증서를 AWS 결제 암호화로 가져오기

이제 퍼블릭 키를 가져올 수 있습니다. 퍼블릭 키를 가져오는 데는 두 가지 옵션이 있습니다. TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE는 키의 용도가 서명을 확인하는 것인 경우(예: TR-34 사용하여 가져오는 경우) 사용할 수 있습니다. TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION는 다른 시스템에서 사용할 데이터를 암호화할 때 사용할 수 있습니다.

Example

```

$ aws payment-cryptography import-key \
  --key-material='{"TrustedCertificatePublicKey":
{"CertificateAuthorityPublicKeyIdentifier":"arn:aws:payment-cryptography:us-
east-2:111122223333:key/zabouwe3574jysd1", \
  "KeyAttributes":
{"KeyAlgorithm":"RSA_2048","KeyClass":"PUBLIC_KEY","KeyModesOfUse":
{"Verify":true},"KeyUsage":"TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"},\
  "PublicKeyCertificate":"LS0tLS1CRUdJTjB..."}}'

```

```

{
  "Key": {
    "CreateTimestamp": "2023-08-08T18:55:46.815000+00:00",
    "Enabled": true,
    "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/4kd6xud22e64wcbk",
    "KeyAttributes": {
      "KeyAlgorithm": "RSA_4096",
      "KeyClass": "PUBLIC_KEY",
      "KeyModesOfUse": {

```

```

    "Decrypt": false,
    "DeriveKey": false,
    "Encrypt": false,
    "Generate": false,
    "NoRestrictions": false,
    "Sign": false,
    "Unwrap": false,
    "Verify": true,
    "Wrap": false
  },
  "KeyUsage": "TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"
},
"KeyOrigin": "EXTERNAL",
"KeyState": "CREATE_COMPLETE",
"UsageStartTimestamp": "2023-08-08T18:55:46.815000+00:00"
}
}

```

키 내보내기

주제

- [대칭 키 내보내기](#)
- [비대칭 \(RSA\) 키 내보내기](#)

대칭 키 내보내기

Important

예제에는 최신 버전의 AWS CLI V2가 필요할 수 있습니다. 시작하기 전에 [최신 버전으로 업그레이드](#)했는지 확인하십시오.

주제

- [비대칭 기법을 사용하여 키 내보내기\(TR-34\)](#)
- [비대칭 기법 \(RSA Wrap\) 을 사용하여 키를 내보냅니다.](#)
- [사전 설정된 키 교환 키\(TR-31\)를 사용하여 대칭 키를 내보냅니다.](#)
- [DUKPT 초기 키 내보내기 \(IPEK/IK\)](#)

비대칭 기법을 사용하여 키 내보내기(TR-34)

개요: TR-34 시스템은 RSA 비대칭 암호화를 활용하여 교환을 위한 대칭 키를 암호화하고 데이터 소스(서명)를 보장합니다. 이렇게 하면 래핑된 키의 기밀성(암호화)과 무결성(서명)이 모두 보장됩니다. 내보낼 때 AWS 결제 암호화는 키 배포 호스트 (KDH) 가 되고 대상 시스템은 키 수신 장치 (KRD) 가 됩니다.

1. 내보내기 초기화 명령 호출

get-parameters-for-export를 호출하여 내보내기 프로세스를 초기화합니다. 이 API는 키 내보내기를 위한 키페어를 생성하고, 키에 서명하고, 인증서 및 인증서 루트를 반환합니다. 최종적으로 이 명령으로 생성된 프라이빗 키는 내보내기 페이로드에 서명하는 데 사용되었습니다. TR-34 용어로는 이를 KDH 서명 인증서라고 합니다. 이러한 인증서는 유효기간이 짧으며 이 용도로만 사용된다는 점에 유의하세요. 매개변수는 기간을 지정합니다 ParametersValidUntilTimestamp.

참고: 모든 인증서는 base64로 인코딩된 형식으로 반환됩니다.

Example

```
$ aws payment-cryptography get-parameters-for-export \
  --signing-key-algorithm RSA_2048 --key-material-type
  TR34_KEY_BLOCK
```

```
{
  "SigningKeyCertificate":
  "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUV2RENDQXFTZ0F3SUJBZ0lRZFAzSzNHNEFKT0I4WTNpTmUvY1
  "SigningKeyCertificateChain":
  "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUY0VENDQTh0Z0F3SUJBZ0lRSQUt1N2piaHFKZjJpPd3FGUWI5c3
  "SigningKeyAlgorithm": "RSA_2048",
  "ExportToken": "export-token-au7pvkbsq4mbup6i",
  "ParametersValidUntilTimestamp": "2023-06-13T15:40:24.036000-07:00"
}
```

2. AWS 결제 암호화 인증서를 수신 시스템으로 가져오기

필요에 따라 1단계에서 제공한 인증서 체인을 수신 시스템으로 가져옵니다.

3. 키 페어를 생성하고, 공개 인증서를 생성하고, AWS 결제 암호화에 인증서 루트를 제공합니다.

전송된 페이로드의 기밀성을 보장하기 위해 전송 당사자(키 배포 호스트 또는 KDH)가 페이로드를 암호화합니다. 수신 당사자(일반적으로 HSM 또는 파트너의 HSM)는 이러한 목적으로 공개 키

를 생성한 다음 결제 암호화에 다시 제공할 수 있는 공개 키 인증서 (x.509) 를 생성하려고 합니다. AWS AWS Private CA 인증서를 생성하는 옵션 중 하나이지만 사용되는 인증 기관에는 제한이 없습니다.

인증서를 얻었으면 `ImportKey` 명령 및 `KeyMaterialType of`를 사용하여 AWS Payment Cryptography에 루트 인증서를 로드하는 것이 좋습니다. `ROOT_PUBLIC_KEY_CERTIFICATE` `KeyUsageType TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`

이 인증서는 `KeyUsageType` 루트 키이고 리프 인증서에 서명하는 데 사용되므로 `TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`입니다. 가져오기/내보내기를 위한 리프 인증서는 결제 암호화로 가져오지 않고 AWS 인라인으로 전달됩니다.

Note

이전에 루트 인증서를 가져온 경우 이 단계를 건너뛸 수 있습니다.

4. 키 내보내기 호출

마지막 단계에서는 `of`를 사용하여 `ExportKey` API를 호출합니다. `KeyMaterialType TR34_KEY_BLOCK certificate-authority-public-key-identifier`는 3단계에서 가져온 루트 CA의 `keyARN`, `WrappingKeyCertificate`는 3단계에서 가져온 리프 인증서, `export-key-identifier`는 내보낼 `keyARN`(또는 별칭)입니다. 또한 1단계의 내보내기 토큰도 제공해야 합니다.

비대칭 기법 (RSA Wrap) 을 사용하여 키를 내보냅니다.

개요: AWS 결제 암호화는 거래 상대방이 TR-34 옵션을 사용할 수 없는 경우 키 교환을 위한 RSA 래핑/언랩을 지원합니다. 이 기법은 TR-34 방식과 마찬가지로 RSA 비대칭 암호화를 활용하여 대칭 키를 암호화하여 교환합니다. 하지만 TR-34 방식과 달리 이 방법은 전송 당사자가 서명한 페이로드가 없습니다. 또한 이 RSA 래핑 기법에는 전송 중에 키 메타데이터의 무결성을 유지하는 데 사용되는 키 블록이 포함되지 않습니다.

Note

RSA 래핑은 TDES 및 AES-128 키를 내보내는 데 사용할 수 있습니다.

1. 수신 시스템에서 RSA 키 및 인증서 생성

래핑된 키를 수신하는 데 사용할 RSA 키를 생성 (또는 식별) 하십시오. AWS 결제 암호화에는 X.509 인증서 형식의 키가 필요합니다. 인증서는 결제 암호화로 가져온 (또는 가져올 수 있는) 루트 인증서로 서명해야 합니다. AWS

2. AWS 결제 암호화에 루트 공개 인증서를 설치하세요.

```
$ aws payment-cryptography import-key --key-material='{"RootCertificatePublicKey":
{"KeyAttributes":{"KeyAlgorithm":"RSA_4096","KeyClass":"PUBLIC_KEY","KeyModesOfUse":
{"Verify":
true},"KeyUsage":"TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"},"PublicKeyCertificate":"LS
```

```
{
  "Key": {
    "CreateTimestamp": "2023-09-14T10:50:32.365000-07:00",
    "Enabled": true,
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
nsq2i3mbg6sn775f",
    "KeyAttributes": {
      "KeyAlgorithm": "RSA_4096",
      "KeyClass": "PUBLIC_KEY",
      "KeyModesOfUse": {
        "Decrypt": false,
        "DeriveKey": false,
        "Encrypt": false,
        "Generate": false,
        "NoRestrictions": false,
        "Sign": false,
        "Unwrap": false,
        "Verify": true,
        "Wrap": false
      },
      "KeyUsage": "TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"
    },
    "KeyOrigin": "EXTERNAL",
    "KeyState": "CREATE_COMPLETE",
    "UsageStartTimestamp": "2023-09-14T10:50:32.365000-07:00"
  }
}
```

3. 통화 내보내기 키

다음으로 리프 인증서를 사용하여 키를 내보내도록 AWS 결제 암호화에 지시해야 합니다. 이전에 가져온 루트 인증서의 ARN, 내보내기에 사용할 리프 인증서 및 내보내기에 사용할 대칭 키를 지정합니다. 출력은 대칭 키의 16진수로 인코딩된 바이너리로 래핑된 (암호화된) 버전입니다.

```
$ cat export-key.json
```

```
{
  "ExportKeyIdentifier": "arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi",
  "KeyMaterial": {
    "KeyCryptogram": {
      "CertificateAuthorityPublicKeyIdentifier": "arn:aws:payment-cryptography:us-east-2:111122223333:key/zabouwe3574jysd1",
      "WrappingKeyCertificate": "LS0tLS1CRUdJTiBD...",
      "WrappingSpec": "RSA_OAEP_SHA_256"
    }
  }
}
```

```
$ aws payment-cryptography export-key --cli-input-json file://export-key.json
```

```
{
  "WrappedKey": {
    "KeyMaterial":
    "18874746731E9E1C4562E4116D1C2477063FCB08454D757D81854AEAEE0A52B1F9D303FA29C02DC82AE7785353",
    "WrappedKeyMaterialFormat": "KEY_CRYPTOGRAM"
  }
}
```

4. 수신 시스템으로 키 가져오기

많은 HSM 및 관련 시스템에서는 RSA 언래핑을 사용하여 키를 가져오는 기능 (AWS 결제 암호화 포함) 을 지원합니다. 이렇게 하려면 1단계의 공개 키를 (암호화) 인증서로 지정합니다. 형식은 RSA, 패딩 모드 = PKCS #1 v2.2 OAEP (SHA 256 포함) 로 지정해야 합니다. 정확한 용어는 HSM 에 따라 다를 수 있습니다.

Note

AWS 결제 암호화는 래핑된 키를 HexBinary로 출력합니다. 시스템에 base64와 같은 다른 바이너리 표현이 필요한 경우 가져오기 전에 형식을 변환해야 할 수 있습니다.

사전 설정된 키 교환 키(TR-31)를 사용하여 대칭 키를 내보냅니다.

파트너가 여러 키를 교환하는 경우 (또는 키 교체를 지원하기 위해), 일반적으로 먼저 종이 키 구성 요소와 같은 기술을 사용하여 초기 키 암호화 키 (KEK) 를 교환하거나 TR-34 기반의 AWS 결제 암호화를 사용하여 교환합니다. KEK가 설정되면 이 키를 사용하여 후속 키 (다른 KEK 포함) 를 전송할 수 있습니다. AWS 결제 암호화는 HSM 공급업체에서 널리 사용되고 널리 지원하는 ANSI TR-31 기반의 이러한 종류의 키 교환을 지원합니다.

1. 키 암호화 키(KEK) 교환

이미 KEK를 교환했고 keyARN(또는 keyAlias)을 사용할 수 있다고 가정합니다.

2. 결제 암호화에 대한 키 생성 AWS

키가 없으면 키를 새로 만듭니다. 반대로, 다른 시스템에서 키를 생성하고 대신 [가져오기](#) 명령을 사용할 수 있습니다.

3. AWS 결제 암호화에서 키 내보내기

내보낼 때 형식은 TR-31이 됩니다. API를 호출할 때 내보낼 키와 사용할 래핑 키를 지정합니다.

```
$ aws payment-cryptography export-key --key-material='{ "Tr31KeyBlock":
  {"WrappingKeyIdentifier": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza"} }' --export-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/5rplquuwozodpwp
```

```
{
  "WrappedKey": {
    "KeyCheckValue": "73C263",
    "KeyCheckValueAlgorithm": "ANSI_X9_24",
    "KeyMaterial":
      "D0144K0AB00E0000A24D3ACF3005F30A6E31D533E07F2E1B17A2A003B338B1E79E5B3AD4FBF7850FACF9A37844",
    "WrappedKeyMaterialFormat": "TR31_KEY_BLOCK"
  }
}
```

4. 시스템으로 가져오기

귀하 또는 파트너는 시스템의 가져오기 키 구현을 사용하여 키를 가져옵니다.

DUKPT 초기 키 내보내기 (IPEK/IK)

[DUKPT](#)를 사용할 경우 여러 터미널에 대해 단일 기본 파생 키 (BDK) 가 생성될 수 있습니다. 그러나 터미널은 원래 BDK에 액세스할 수 없지만 각 터미널에는 IPEK 또는 IK (초기 키) 라는 고유한 초기 터미널 키가 삽입됩니다. 각 IPEK는 BDK에서 파생된 키이며 터미널별로 고유하도록 설계되었지만 원래 BDK에서 파생되었습니다. 이 계산의 파생 데이터를 KSN (키 일련 번호) 이라고 합니다. X9.24에 따라 TDES의 경우 10바이트 KSN은 일반적으로 키 세트 ID의 경우 24비트, 터미널 ID의 경우 19비트, 트랜잭션 카운터의 경우 21비트로 구성됩니다. AES의 경우 12바이트 KSN은 일반적으로 BDK ID의 경우 32비트, 파생 식별자 (ID) 32비트, 트랜잭션 카운터의 경우 32비트로 구성됩니다.

AWS 결제 암호화는 이러한 초기 키를 생성하고 내보내는 메커니즘을 제공합니다. 생성된 키는 TR-31, TR-34 및 RSA 래핑 방법을 사용하여 내보낼 수 있습니다. IPEK 키는 유지되지 않으며 결제 암호화에 대한 후속 작업에 사용할 수 없습니다. AWS

AWS 결제 암호화는 KSN의 처음 두 부분 간의 분할을 강제하지 않습니다. 파생 식별자를 BDK와 함께 저장하려는 경우 이러한 목적으로 태그 기능을 사용할 수 있습니다. AWS

Note

KSN의 카운터 부분 (AES DUKPT의 경우 32비트) 은 IPEK/IK 유도에는 사용되지 않습니다. 따라서 12345678901234560001과 1234567890123456999999를 입력하면 동일한 IPEK가 출력됩니다.

```
$ aws payment-cryptography export-key --key-material='{ "Tr31KeyBlock":
  {"WrappingKeyIdentifier": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza"} }' --export-key-identifier arn:aws:payment-
cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi --export-attributes
'ExportDukptInitialKey={KeySerialNumber=12345678901234560001}'
```

```
{
  "WrappedKey": {
    "KeyCheckValue": "73C263",
    "KeyCheckValueAlgorithm": "ANSI_X9_24",
    "KeyMaterial":
      "B0096B1TX00S000038A8A06588B9011F0D5EEF1CCAECFA6962647A89195B7A98BDA65DDE7C57FEA507559AF2A5D60"
```

```

    "WrappedKeyMaterialFormat": "TR31_KEY_BLOCK"
  }
}

```

비대칭 (RSA) 키 내보내기

인증서 형식으로 퍼블릭 키를 내보내려면 `get-public-key-certificate`을 호출하세요. 이 API는 인증서와 base64 형식으로 인코딩된 루트 인증서를 내보냅니다.

참고: 이 API는 동일하지 않습니다. 이후 호출하면 기본 키가 동일하더라도 다른 인증서가 생성될 수 있습니다.

Example

```

$ aws payment-cryptography get-public-key-certificate \
    -key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/5dza7xqd6soanjtb

```

```

{
  "KeyCertificate": "LS0tLS1CRUdJTi...",
  "KeyCertificateChain": "LS0tLS1CRUdJT..."
}

```

별칭 사용

별칭은 결제 암호화 키의 친숙한 이름입니다. AWS 예를 들어 별칭을 사용하면 키를 `arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaif1lw2h` 대신 `alias/test-key`로 지칭할 수 있습니다.

대부분의 키 관리(컨트롤 플레인) 작업과 [암호화\(데이터플레인\) 작업](#)에서 별칭을 사용하여 키를 식별할 수 있습니다.

또한 정책을 편집하거나 허가를 관리하지 않고도 별칭을 기반으로 AWS 결제 암호화 키에 대한 액세스를 허용하거나 거부할 수 있습니다. 이 기능은 [ABAC\(속성 기반 액세스 제어\)](#)에 대한 서비스 지원의 일부입니다.

별칭의 강력한 기능은 언제든지 별칭과 연결된 키를 변경할 수 있는 기능에서 비롯됩니다. 별칭을 사용하면 코드를 더 쉽게 작성하고 유지 관리할 수 있습니다. 예를 들어 특정 AWS 결제 암호화 키를 참조할

때 별칭을 사용하고 결제 암호화 키를 변경하려는 경우를 가정해 보겠습니다. AWS 이 경우 별칭을 다른 키와 연결하기만 하면 됩니다. 코드나 애플리케이션 구성을 변경할 필요가 없습니다.

또한 별칭을 사용하면 다른 AWS 리전에서 동일한 코드를 더 쉽게 재사용할 수 있습니다. 여러 지역에서 이름이 같은 별칭을 생성하고 각 별칭을 해당 지역의 AWS 결제 암호화 키와 연결합니다. 코드가 각 지역에서 실행될 때 별칭은 해당 지역의 관련 AWS 결제 암호화 키를 가리킵니다.

API를 사용하여 AWS 결제 암호화 키의 별칭을 생성할 수 있습니다. `CreateAlias`

AWS 결제 암호화 API는 각 계정 및 지역의 별칭을 완벽하게 제어할 수 있도록 합니다. API에는 별칭 생성 (`CreateAlias`), 별칭 이름 및 연결된 KeyARN (목록 별칭) 보기, 별칭과 관련된 AWS 결제 암호화 키 변경 (업데이트 별칭), 별칭 삭제 (별칭 삭제) 작업이 포함됩니다.

주제

- [별칭 정보](#)
- [애플리케이션에서 별칭 사용](#)
- [관련 API](#)

별칭 정보

결제 암호화에서 별칭이 작동하는 방식을 알아보세요. AWS

별칭은 독립적인 리소스입니다. AWS

별칭은 AWS 결제 암호화 키의 속성이 아닙니다. 별칭에 대해 수행하는 작업은 연결된 키에 영향을 주지 않습니다. 결제 암호화 키의 별칭을 생성한 다음 다른 AWS 결제 암호화 키와 연결되도록 별칭을 업데이트할 수 있습니다. AWS 연결된 결제 암호화 키에 영향을 주지 않고 별칭을 삭제할 수도 있습니다. AWS AWS Payment Cryptography 키를 삭제하면 해당 키와 연결된 모든 별칭이 할당되지 않습니다.

IAM 정책에서 별칭을 리소스로 지정하는 경우 정책은 관련 결제 암호화 키가 아닌 별칭을 참조합니다. AWS

각 별칭은 친숙한 이름으로 합니다.

별칭을 만들 때는 접두사 `alias/`를 앞에 붙인 별칭 이름을 지정합니다. 예를 들면 `alias/test_1234`과 같습니다.

각 별칭은 한 번에 하나의 AWS 결제 암호화 키와 연결됩니다.

별칭과 해당 AWS 결제 암호화 키는 동일한 계정 및 지역에 있어야 합니다.

AWS 결제 암호화 키는 둘 이상의 별칭과 동시에 연결할 수 있지만 각 별칭은 단일 키에만 매핑할 수 있습니다.

예를 들어 이 `list-aliases` 출력은 `alias/sampleAlias1` 별칭이 `KeyArn` 속성으로 표시되는 정확히 하나의 대상 AWS Payment Cryptography 키와 연결되어 있음을 보여줍니다.

```
$ aws payment-cryptography list-aliases
```

```
{
  "Aliases": [
    {
      "AliasName": "alias/sampleAlias1",
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaif1lw2h"
    }
  ]
}
```

여러 별칭을 동일한 결제 암호화 키에 연결할 수 있습니다. AWS

예를 들어, `alias/sampleAlias1`; 및 `alias/sampleAlias2` 별칭을 동일한 키로 연결할 수 있습니다.

```
$ aws payment-cryptography list-aliases
```

```
{
  "Aliases": [
    {
      "AliasName": "alias/sampleAlias1",
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaif1lw2h"
    },
    {
      "AliasName": "alias/sampleAlias2",
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaif1lw2h"
    }
  ]
}
```

```

    }
  ]
}

```

별칭은 해당 계정 및 리전에 대해 고유해야 합니다.

예를 들어, 각 계정 및 리전에서 오직 한 개의 `alias/sampleAlias1` 별칭만 가질 수 있습니다. 별칭은 대소문자를 구분하지만 오류가 발생하기 쉬우므로 대소문자만 다른 별칭은 사용하지 않는 것이 좋습니다. 별칭 이름은 변경할 수 없습니다. 그러나 별칭을 삭제하고 원하는 이름으로 새 별칭을 생성할 수 있습니다.

다른 리전에서 같은 이름으로 별칭을 만들 수 있습니다.

예를 들어, 미국 동부(버지니아 북부)에 `alias/sampleAlias2` 별칭이 있고 미국 서부(오레곤)에 `alias/sampleAlias2` 별칭이 있을 수 있습니다. 각 별칭은 해당 지역의 AWS 결제 암호화 키와 연결됩니다. 코드가 `alias/finance-key`와 같은 별칭 이름을 참조하는 경우 여러 리전에서 실행할 수 있습니다. 각 리전에서는 다른 별칭/`sampleAlias2`을 사용합니다. 자세한 내용은 [애플리케이션에서 별칭 사용](#) 단원을 참조하세요.

별칭과 연결된 AWS 결제 암호화 키를 변경할 수 있습니다.

`UpdateAlias` 작업을 사용하여 별칭을 다른 AWS 결제 암호화 키와 연결할 수 있습니다. 예를 들어 별칭이 `arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaif1lw2h` AWS 결제 암호화 키와 연결된 경우 키에 연결되도록 `alias/sampleAlias2` 별칭을 업데이트할 수 있습니다. `arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi`

Warning

AWS 결제 암호화는 기존 키와 새 키가 모두 키 사용과 같은 동일한 속성을 갖는지 확인하지 않습니다. 다른 키 유형으로 업데이트하면 애플리케이션에 문제가 발생할 수 있습니다.

일부 키에는 별칭이 없습니다.

별칭은 선택적 기능이며 이러한 방식으로 환경을 운영하도록 선택하지 않는 한 모든 키에 별칭이 있는 것은 아닙니다. `create-alias` 명령을 사용하여 키를 별칭과 연결할 수 있습니다. 또한 `update-alias` 작업을 사용하여 별칭과 연결된 AWS Payment Cryptography 키를 변경하고 `update-alias` 작업을 사용하여 별칭을 삭제할 수 있습니다. 따라서 일부 AWS 결제 암호화 키에는 별칭이 여러 개 있을 수 있고 일부는 별칭이 없을 수 있습니다.

키를 별칭에 매핑하기

`create-alias` 명령을 사용하여 키(ARN으로 표시됨)를 하나 이상의 별칭에 매핑할 수 있습니다. 이 명령은 멍등성이 없습니다. 별칭을 업데이트하려면 `update-alias` 명령을 사용하세요.

```
$ aws payment-cryptography create-alias --alias-name alias/sampleAlias1 \
    --key-arn arn:aws:payment-cryptography:us-east-2:111122223333:key/
    kwapwa6qaif1lw2h
```

```
{
  "Alias": {
    "AliasName": "alias/alias/sampleAlias1",
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
    kwapwa6qaif1lw2h"
  }
}
```

애플리케이션에서 별칭 사용

별칭을 사용하여 애플리케이션 코드의 AWS 결제 암호화 키를 나타낼 수 있습니다. AWS 결제 암호화 [데이터 작업과 목록 키와 같은 기타 작업의 `key-identifier`](#) 매개변수는 별칭 이름 또는 별칭 ARN 을 허용합니다.

```
$ aws payment-cryptography-data generate-card-validation-data --key-identifier alias/
    BIN_123456_CVK --primary-account-number=171234567890123 --generation-attributes
    CardVerificationValue2={CardExpiryDate=0123}
```

별칭 ARN을 사용하는 경우 결제 암호화 키에 매핑되는 별칭은 결제 암호화 키를 AWS 소유한 계 정에서 정의되며 지역마다 다를 수 있다는 점을 기억하십시오.

별칭의 가장 강력한 용도 중 하나는 여러 AWS 리전에서 실행되는 애플리케이션에서 사용하는 것입 니다.

지역별로 다른 버전의 애플리케이션을 생성하거나 사전, 구성 또는 스위치 명령문을 사용하여 각 지역 에 적합한 AWS 결제 암호화 키를 선택할 수 있습니다. 그러나 각 리전에 동일한 별칭 이름으로 별칭을 만드는 것이 더 쉬울 수 있습니다. 별칭 이름은 대/소문자를 구분합니다.

관련 API

태그

태그는 AWS 결제 암호화 키를 구성하기 위한 메타데이터 역할을 하는 키와 값 쌍입니다. 키를 유연하게 식별하거나 하나 이상의 키를 그룹화하는 데 사용할 수 있습니다.

키 가져오기

AWS 결제 암호화 키는 암호화 자료의 단일 단위를 나타내며 이 서비스의 암호화 작업에만 사용할 수 있습니다. GetKeys API는 a를 KeyIdentifier 입력으로 받아 키의 변경 불가 및 변경 가능 속성을 반환하지만 암호화 자료는 포함하지 않습니다.

Example

```
$ aws payment-cryptography get-key --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaif1lw2h
```

```
{
  "Key": {
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaif1lw2h",
    "KeyAttributes": {
      "KeyUsage": "TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY",
      "KeyClass": "SYMMETRIC_KEY",
      "KeyAlgorithm": "AES_128",
      "KeyModesOfUse": {
        "Encrypt": true,
        "Decrypt": true,
        "Wrap": true,
        "Unwrap": true,
        "Generate": false,
        "Sign": false,
        "Verify": false,
        "DeriveKey": false,
        "NoRestrictions": false
      }
    }
  },
}
```

```

    "KeyCheckValue": "0A3674",
    "KeyCheckValueAlgorithm": "CMAC",
    "Enabled": true,
    "Exportable": true,
    "KeyState": "CREATE_COMPLETE",
    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
    "CreateTimestamp": "2023-06-02T07:38:14.913000-07:00",
    "UsageStartTimestamp": "2023-06-02T07:38:14.857000-07:00"
  }
}

```

키 페어와 연결된 퍼블릭 키/인증서 가져오기

퍼블릭 키/인증서 가져오기는 KeyArn로 표시된 퍼블릭 키를 반환합니다. 이는 AWS 결제 암호화에서 생성된 키 쌍의 공개 키 부분이거나 이전에 가져온 공개 키일 수 있습니다. 가장 일반적인 사용 사례는 데이터를 암호화하는 외부 서비스에 퍼블릭 키를 제공하는 것입니다. 그런 다음 해당 데이터를 결제 암호화를 활용하는 애플리케이션으로 전달하고 AWS 결제 암호화 내에서 보호되는 개인 키를 사용하여 데이터를 해독할 수 있습니다. AWS

이 서비스는 퍼블릭 키를 공개 인증서로 반환합니다. API 결과에는 CA와 퍼블릭 키 인증서가 포함됩니다. 두 데이터 요소 모두 base64로 인코딩됩니다.

Note

반환된 공개 인증서는 단기적인 것이며 멱등성을 갖지 않습니다. 퍼블릭 키 자체가 변경되지 않았더라도 각 API 직접 호출마다 다른 인증서를 받을 수 있습니다.

Example

```

$ aws payment-cryptography get-public-key-certificate --key-identifier
arn:aws:payment-cryptography:us-east-2:111122223333:key/nsq2i3mbg6sn775f

```

```

{

```

```

"KeyCertificate":
"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUV2VENDQXFXZ0F3SUJBZ01SQ0o10Wd2VkpDd3d1Y1dMNldYZEpYY
"KeyCertificateChain":
"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUY0VENDQThZ0F3SUJBZ01SQ0t1N2piaHFKZjJPd3FGUWI5c3VuO
}

```

키 태그 지정

AWS 결제 암호화에서는 키를 [생성할 때 AWS 결제 암호화 키에 태그를 추가하고, 삭제 대기 중인 키가 아닌 한 기존 키에](#) 태그를 지정하거나 태그를 해제할 수 있습니다. 태그는 선택 사항이지만 매우 유용할 수 있습니다.

모범 사례, 태깅 전략, 태그 형식 및 구문 등 태그에 대한 일반 정보는 의 리소스 [AWS 태깅을](#) 참조하십시오. Amazon Web Services 일반 참조

주제

- [결제 암호화의 태그에 대한 정보 AWS](#)
- [콘솔에서 키 태그 보기](#)
- [API 작업으로 키 태그 관리](#)
- [태그에 대한 액세스 제어](#)
- [태그를 사용하여 키에 대한 액세스 제어](#)

결제 암호화의 태그에 대한 정보 AWS

태그는 AWS 리소스에 할당하거나 AWS 할당할 수 있는 선택적 메타데이터 레이블입니다. 각 태그는 태그 키 및 태그 값으로 구성되며, 둘 다 대소 문자를 구분하는 문자열입니다. 태그 값은 빈(null) 문자열일 수도 있습니다. 리소스의 각 태그는 다른 태그 키를 가져야 하지만 여러 AWS 리소스에 동일한 태그를 추가할 수 있습니다. 각 리소스에는 최대 50개의 사용자 생성 태그가 포함될 수 있습니다.

태그 키 또는 태그 값에 기밀 또는 민감한 정보를 포함하지 마세요. 청구를 AWS 서비스포함하여 많은 사람들이 태그에 액세스할 수 있습니다.

AWS 결제 암호화에서는 키를 [생성할 때 키에 태그를 추가하고, 삭제 대기 중인 키가 아닌 한 기존 키에](#) 태그를 지정하거나 태그를 해제할 수 있습니다. 별칭에는 태그를 지정할 수 없습니다. 태그는 선택 사항이지만 매우 유용할 수 있습니다.

예를 들어 Alpha 프로젝트에 사용하는 모든 AWS 결제 암호화 키와 Amazon S3 버킷에 "Project"="Alpha" 태그를 추가할 수 있습니다. 또 다른 예는 특정 은행 식별 번호 (BIN) 와 관련된 모든 키에 "BIN"="20130622" 태그를 추가하는 것입니다.

```
[
  {
    "Key": "Project",
    "Value": "Alpha"
  },
  {
    "Key": "BIN",
    "Value": "20130622"
  }
]
```

형식 및 구문을 비롯한 태그에 대한 일반 정보는 의 [AWS 리소스 태깅](#)을 참조하십시오. Amazon Web Services 일반 참조

태그는 다음을 지원합니다.

- AWS 리소스를 식별하고 구성하세요. 많은 AWS 서비스가 태그 지정을 지원하므로 서로 다른 서비스의 리소스에 동일한 태그를 할당하여 리소스가 관련되어 있음을 나타낼 수 있습니다. 예를 들어 AWS 결제 암호화 키와 Amazon Elastic Block Store (Amazon Elastic Store) 볼륨 또는 암호에 동일한 태그를 할당할 수 있습니다. AWS Secrets Manager 태그를 사용하여 자동화를 위해 키를 식별할 수도 있습니다.
- 비용을 추적하세요. AWS 리소스에 태그를 추가하면 사용량과 비용이 태그별로 집계된 비용 할당 보고서가 AWS 생성됩니다. 이 기능을 사용하여 프로젝트, 애플리케이션 또는 비용 센터의 AWS 결제 암호화 비용을 추적할 수 있습니다.

비용 할당 태그 사용에 대한 자세한 내용은 AWS Billing 사용자 설명서의 [비용 할당 태그 사용](#)을 참조하세요. 태그 키 및 태그 값에 대한 규칙에 대한 자세한 내용은 AWS Billing 사용자 설명서의 [사용자 정의 태그 제한](#)을 참조하세요.

- AWS 리소스에 대한 액세스를 제어하세요. 태그를 기반으로 키에 대한 액세스를 허용하거나 거부하는 것은 속성 기반 액세스 제어 (ABAC) 에 대한 AWS 결제 암호화 지원의 일부입니다. 태그에 기반으로 AWS Payment Cryptography에 대한 액세스를 제어하는 방법에 대한 자세한 내용은 [AWS Payment Cryptography 태그 기반 인증](#)을 참조하세요. 태그를 사용하여 AWS 리소스에 대한 액세스

를 제어하는 방법에 대한 자세한 내용은 [IAM 사용 설명서의 리소스 태그를 사용한 AWS 리소스 액세스 제어를](#) 참조하십시오.

AWS 결제 암호화는 TagResource UntagResource, 또는 ListTagsForResource 작업을 사용할 때 AWS CloudTrail 로그에 항목을 기록합니다.

콘솔에서 키 태그 보기

콘솔에서 태그를 보려면 키가 포함된 IAM 정책의 키에 대한 태그 지정 권한이 필요합니다. 콘솔에서 키를 볼 수 있는 권한과 함께 이러한 권한이 필요합니다.

API 작업으로 키 태그 관리

[AWS Payment Cryptography API](#)를 이용해 현재 관리 중인 키에 대한 태그를 추가, 삭제, 나열할 수 있습니다. 이 예제들은 [AWS Command Line Interface \(AWS CLI\)](#)를 사용하지만, 사용자는 어떤 지원되는 프로그래밍 언어라도 사용할 수 있습니다. 태그를 지정할 AWS 관리형 키수 없습니다.

키의 태그를 추가, 편집, 보기 및 삭제하려면 필요한 사용 권한이 있어야 합니다. 자세한 내용은 [태그에 대한 액세스 제어](#) 단원을 참조하세요.

주제

- [CreateKey: 새 키에 태그 추가](#)
- [TagResource: 키에 태그 추가 또는 변경](#)
- [ListResourceTags: 키의 태그 가져오기](#)
- [UntagResource: 키에서 태그 삭제](#)

CreateKey: 새 키에 태그 추가

키를 생성할 때 태그를 추가할 수 있습니다. 태그를 지정하려면 [CreateKey](#) 작업의 Tags 파라미터를 사용합니다.

키를 생성할 때 태그를 추가하려면 IAM 정책에서 호출자가 payment-cryptography:TagResource 권한을 가지고 있어야 합니다. 최소한 권한에 계정 및 리전의 모든 키가 포함되어야 합니다. 자세한 내용은 [태그에 대한 액세스 제어](#) 섹션을 참조하세요.

CreateKey의 값은 Tags 파라미터 값은 대소문자를 구분하는 태그 키 및 태그 값 페어 모음입니다. 키의 각 태그에는 다른 태그 이름이 있어야 합니다. 태그 값은 null이거나 빈 문자열일 수 있습니다.

예를 들어 다음 AWS CLI 명령은 Project:Alpha 태그가 있는 대칭 암호화 키를 만듭니다. 두 개 이상의 키-값 페어를 지정할 때는 공백을 사용하여 각 페어를 구분합니다.

```
$ aws payment-cryptography create-key --exportable --key-attributes
  KeyAlgorithm=TDES_2KEY, \
    KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY, \
    KeyModesOfUse='{Generate=true,Verify=true}' \
    --tags '[{"Key":"Project","Value":"Alpha"}, {"Key":"BIN","Value":"123456"}]'
```

이 명령이 성공하면 새 키에 대한 정보가 있는 Key 객체를 반환합니다. 그러나 Key에는 태그가 포함되지 않습니다. 태그를 가져오려면 [ListResourceTags](#) 작업을 사용하십시오.

TagResource: 키에 태그 추가 또는 변경

이 [TagResource](#) 작업은 키에 하나 이상의 태그를 추가합니다. 이 작업을 사용하여 다른 AWS 계정의 태그를 추가 또는 편집할 수 없습니다.

태그를 추가하려면 새 태그 키와 태그 값을 지정합니다. 태그를 편집하려면 기존 태그 키와 새 태그 값을 지정합니다. 키의 각 태그에는 다른 태그 키가 있어야 합니다. 태그 값은 null이거나 빈 문자열일 수 있습니다.

예를 들어 다음 명령은 예제 키에 **UseCase** 및 **BIN** 태그를 추가합니다.

```
$ aws payment-cryptography tag-resource --resource-arn arn:aws:payment-
  cryptography:us-east-2:111122223333:key/kwapwa6qaif1lw2h --tags
  '[{"Key":"UseCase","Value":"Acquiring"}, {"Key":"BIN","Value":"123456"}]'
```

이 명령이 제대로 실행되면 메타데이터를 반환하지 않습니다. 키의 태그를 보려면 [ListResourceTags](#) 작업을 사용하십시오.

또한 TagResource를 이용해 기존 태그의 태그 값을 변경할 수도 있습니다. 태그 값을 바꾸려면 동일한 태그 키를 다른 값으로 지정하세요. 수정 명령에 나열되지 않은 태그는 변경되거나 제거되지 않습니다.

예를 들어 이 명령은 Project 태그 값을 Alpha에서 Noe로 바꿉니다.

이 명령은 내용 없이 http/200을 반환합니다. 변경 내용을 보려면 ListTagsForResource를 사용하세요.

```
$ aws payment-cryptography tag-resource --resource-arn arn:aws:payment-cryptography:us-
  east-2:111122223333:key/kwapwa6qaif1lw2h \
    --tags '[{"Key":"Project","Value":"Noe"}]'
```

ListResourceTags: 키의 태그 가져오기

[ListResourceTags](#) 작업은 키의 태그를 가져옵니다. ResourceArn(keyArn 또는 keyAlias) 파라미터가 필요합니다. 이 작업을 사용하여 다른 AWS 계정의 키에 있는 태그를 볼 수 없습니다.

예를 들어 다음 명령은 예제 키에 대한 태그를 가져옵니다.

```
$ aws payment-cryptography list-tags-for-resource --resource-arn arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaif1lw2h

{
  "Tags": [
    {
      "Key": "BIN",
      "Value": "20151120"
    },
    {
      "Key": "Project",
      "Value": "Production"
    }
  ]
}
```

UntagResource: 키에서 태그 삭제

이 [UntagResource](#) 작업은 키에서 태그를 삭제합니다. 삭제할 태그를 식별하려면 태그 키를 지정합니다. 이 작업을 사용하여 다른 AWS 계정의 키에서 태그를 삭제할 수 없습니다.

성공하면 UntagResource 작업은 어떠한 출력도 반환하지 않습니다. 또한 지정된 태그 키가 키에서 발견되지 않으면 예외를 발생시키거나 응답을 반환하지 않습니다. 작업이 제대로 수행되었는지 확인하려면 [ListResourceTags](#) 작업을 사용하십시오.

예를 들어 이 명령은 키에서 **Purpose** 태그와 해당 값을 삭제합니다.

```
$ aws payment-cryptography untag-resource \
  --resource-arn arn:aws:payment-cryptography:us-east-2:111122223333:key/
  kwapwa6qaif1lw2h --tag-keys Project
```

태그에 대한 액세스 제어

API를 사용하여 태그를 추가하고, 보고, 삭제하려면 보안 주체에게 IAM 정책의 태그 지정 권한이 필요합니다.

태그의 AWS 글로벌 조건 키를 사용하여 이러한 권한을 제한할 수도 있습니다. AWS 결제 암호화에서 이러한 조건은 및 와 같은 [TagResource](#) 태깅 작업에 대한 액세스를 제어할 수 있습니다.

[UntagResource](#)

예제 정책과 자세한 내용은 IAM 사용 설명서의 [태그 키를 기반으로 액세스 제어](#) 섹션을 참조하십시오.

태그를 만들고 관리할 수 있는 권한은 다음과 같습니다.

결제 암호화: TagResource

보안 주체가 태그를 추가하거나 편집할 수 있습니다. 키를 생성하는 동안 태그를 추가하려면 보안 주체에 특정 키로 제한되지 않는 IAM 정책에 대한 권한이 있어야 합니다.

결제 암호화: ListTagsForResource

보안 주체가 키의 태그를 볼 수 있도록 허용합니다.

결제 암호화: UntagResource

보안 주체가 키에서 태그를 삭제할 수 있도록 허용합니다.

정책에서 태그 지정 권한

키 정책 또는 IAM 정책에서 태그 지정 권한을 제공할 수 있습니다. 예를 들어 다음 예제 키 정책은 키에 대한 태그 지정 권한을 사용자에게 제공합니다. 예를 들어 관리자 또는 개발자 역할로 가정할 수 있는 모든 사용자에게 태그를 볼 수 있는 권한을 제공합니다.

```
{
  "Version": "2012-10-17",
  "Id": "example-key-policy",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::111122223333:root"},
      "Action": "payment-cryptography:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow all tagging permissions",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:user/LeadAdmin",
        "arn:aws:iam::111122223333:user/SupportLead"
      ]}
    }
  ]
}
```

```

    ]},
    "Action": [
      "payment-cryptography:TagResource",
      "payment-cryptography:ListTagsForResource",
      "payment-cryptography:UntagResource"
    ],
    "Resource": "*"
  },
  {
    "Sid": "Allow roles to view tags",
    "Effect": "Allow",
    "Principal": {"AWS": [
      "arn:aws:iam::111122223333:role/Administrator",
      "arn:aws:iam::111122223333:role/Developer"
    ]},
    "Action": "payment-cryptography:ListResourceTags",
    "Resource": "*"
  }
]
}

```

보안 주체에게 여러 키에 대한 태그 지정 권한을 부여하려면 IAM 정책을 사용할 수 있습니다. 이 정책이 유효하려면 각 키의 키 정책으로 인해 계정이 IAM 정책을 사용하여 키에 대한 액세스를 제어할 수 있어야 합니다.

예를 들어, 다음 IAM 정책은 보안 주체가 키를 생성할 수 있도록 허용합니다. 또한 지정된 계정의 모든 키에 태그를 만들고 관리할 수 있습니다. 이 조합을 사용하면 보안 주체는 키를 생성하는 동안 [CreateKey](#) 작업의 tags 매개 변수를 사용하여 키에 태그를 추가할 수 있습니다.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyCreateKeys",
      "Effect": "Allow",
      "Action": "payment-cryptography:CreateKey",
      "Resource": "*"
    },
    {
      "Sid": "IAMPolicyTags",
      "Effect": "Allow",
      "Action": [
        "payment-cryptography:TagResource",

```

```

    "payment-cryptography:UntagResource",
    "payment-cryptography:ListTagsForResource"
  ],
  "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*"
}
]
}

```

태그 지정 권한 제한

정책 조건을 사용하여 태그 지정 권한을 제한할 수 있습니다. 다음 정책 조건을 `payment-cryptography:TagResource` 및 `payment-cryptography:UntagResource` 권한에 적용할 수 있습니다. 예를 들어, `aws:RequestTag/tag-key` 조건을 사용하여 보안 주체가 특정 태그만 추가하거나 보안 주체가 특정 태그 키를 사용하여 태그를 추가하지 못하도록 할 수 있습니다.

- [aws: RequestTag](#)
- [aws: ResourceTag/태그 키](#) (IAM 정책만 해당)
- [AWS: TagKeys](#)

태그를 사용하여 키에 대한 액세스를 제어할 때 가장 좋은 방법은 `aws:RequestTag/tag-key` 또는 `aws:TagKeys` 조건 키를 사용하여 허용되는 태그 (또는 태그 키)를 결정하는 것입니다.

예를 들어 다음 IAM 정책은 이전 것과 비슷합니다. 그러나 이 정책은 보안 주체가 Project 태그 키가 있는 태그에 대해서만 태그(TagResource)를 생성하고 태그 UntagResource를 삭제할 수 있도록 허용합니다.

UntagResource 요청에는 여러 태그가 포함될 수 있으므로 TagResource [aws: TagKeys](#) 조건으로 ForAllValues or ForAnyValue set 연산자를 지정해야 합니다. ForAnyValue 연산자를 사용하면 요청의 태그 키 중 적어도 하나가 정책의 태그 키 중 하나와 일치해야 합니다. ForAllValues 연산자를 사용하면 요청의 모든 태그 키가 정책의 태그 키 중 하나와 일치해야 합니다. 또한 ForAllValues 연산자는 요청에 태그가 없는 true 경우 반환하지만 TagResource 태그가 지정되지 않으면 UntagResource 실패합니다. 집합 연산자에 대한 자세한 내용은 IAM 사용 설명서의 [여러 키 및 값 사용](#)을 참조하십시오.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyCreateKey",

```

```

    "Effect": "Allow",
    "Action": "payment-cryptography:CreateKey",
    "Resource": "*"
  },
  {
    "Sid": "IAMPolicyViewAllTags",
    "Effect": "Allow",
    "Action": "payment-cryptography:ListResourceTags",
    "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*"
  },
  {
    "Sid": "IAMPolicyManageTags",
    "Effect": "Allow",
    "Action": [
      "payment-cryptography:TagResource",
      "payment-cryptography:UntagResource"
    ],
    "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*",
    "Condition": {
      "ForAllValues:StringEquals": {"aws:TagKeys": "Project"}
    }
  }
]
}

```

태그를 사용하여 키에 대한 액세스 제어

키의 태그를 기반으로 AWS 결제 암호화에 대한 액세스를 제어할 수 있습니다. 예를 들어 보안 주체가 특정 태그가 있는 키만 활성화 및 비활성화할 수 있도록 허용하는 IAM 정책을 작성할 수 있습니다. 또는 IAM 정책을 사용하여 키에 특정 태그가 없으면 보안 주체가 암호화 작업에서 키를 사용하지 못하도록 할 수 있습니다.

이 기능은 속성 기반 액세스 제어 (ABAC) 를 위한 AWS 결제 암호화 지원의 일부입니다. [태그를 사용하여 AWS 리소스에 대한 액세스를 제어하는 방법에 대한 자세한 내용은 ABAC의 용도를 참조하십시오.](#) [AWS](#) 및 IAM 사용 설명서의 [리소스 태그를 사용하여 AWS 리소스에 대한 액세스를 제어합니다.](#)

Note

AWS 결제 암호화는 [aws:ResourceTag/tag-key](#) 글로벌 조건 컨텍스트 키를 지원하며, 이를 통해 키의 태그를 기반으로 키에 대한 액세스를 제어할 수 있습니다. 여러 키가 동일한 태그를 가질 수 있으므로

이 기능을 사용하면 선택한 키 집합에 사용 권한을 적용할 수 있습니다. 태그를 변경하여 집합의 키를 쉽게 변경할 수도 있습니다.

AWS 결제 암호화에서는 `aws:ResourceTag/tag-key` 조건 키가 IAM 정책에서만 지원됩니다. 키 한 개에만 적용되는 키 정책이나 또는 작업과 같이 특정 키를 사용하지 않는 작업에서는 지원되지 않습니다. [ListKeysListAliases](#)

태그를 사용하여 액세스를 제어하면 사용 권한을 단순하고 확장 가능하며 유연하게 관리할 수 있습니다. 그러나 제대로 설계되고 관리되지 않으면 실수로 키에 대한 액세스를 허용하거나 거부할 수 있습니다. 태그를 사용하여 액세스를 제어하는 경우 다음 방법을 고려하세요.

- [최소 권한 액세스](#)를 강화하는 최고의 방식은 태그를 사용하는 것입니다. IAM 보안 주체에 사용하거나 관리해야 하는 키에만 필요한 권한만 부여합니다. 예를 들어 태그를 사용하여 프로젝트에 사용되는 키에 레이블을 지정합니다. 그런 다음 프로젝트 팀에 프로젝트 태그와 함께 키만 사용할 수 있는 권한을 부여합니다.
- 보안 주체에게 태그를 추가, 편집 및 삭제할 수 있는 `payment-cryptography:TagResource` 및 `payment-cryptography:UntagResource` 권한을 부여할 때는 주의해야 합니다. 태그를 사용하여 키에 대한 액세스를 제어하는 경우 태그를 변경하면 보안 주체에게 사용 권한이 없는 키를 사용할 수 있는 권한이 부여될 수도 있습니다. 또한 다른 보안 주체가 작업을 수행하는 데 필요한 키에 대한 액세스를 거부할 수도 있습니다. 키 정책을 변경하거나 권한 부여를 생성할 권한이 없는 키 관리자는 태그를 관리할 권한이 있는 경우 키에 대한 액세스를 제어할 수 있습니다.

가능하면 정책 조건(예: `aws:RequestTag/tag-key` 또는 `aws:TagKeys`)을 사용하여 [보안 주체의 태그 지정 권한](#)을 특정 키의 특정 태그 또는 태그 패턴으로 제한합니다.

- 현재 태그 지정 및 태그 해제 권한이 AWS 계정 있는 보안 주체를 검토하고 필요한 경우 이를 조정하십시오. IAM 정책은 모든 키에 대한 태그 및 태그 해제 권한을 허용할 수 있습니다. 예를 들어 관리자 관리형 정책은 보안 주체가 모든 키에 대해 태그를 지정하고 태그를 해제하고 나열할 수 있도록 허용합니다.
- 태그에 따른 정책을 설정하기 전에 내 키의 태그를 검토하십시오. AWS 계정포함하려는 태그에만 정책을 적용해야 합니다. [CloudTrail 로그](#) 및 CloudWatch 알람을 사용하여 키 액세스에 영향을 미칠 수 있는 태그 변경 사항을 알리세요.
- 태그 기반 정책 조건은 패턴 일치기를 사용하며 태그의 특정 인스턴스에 연결되어 있지 않습니다. 태그 기반 조건 키를 사용하는 정책은 패턴과 일치하는 모든 새 태그와 기존 태그에 영향을 줍니다. 정책 조건과 일치하는 태그를 삭제했다가 다시 만들면 이전 태그와 마찬가지로 조건이 새 태그에 적용됩니다.

예를 들어 다음과 같은 IAM 정책을 살펴보십시오. 이를 통해 보안 주체는 사용자 계정의 키 중 미국 동부 (버지니아 북부) 리전이고 "Project"="Alpha" 태그가 있는 키에 대해서만 [암호 해독](#) 작업을 호출할 수 있습니다. 이 정책은 예제 Alpha 프로젝트의 역할에 연결할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyWithResourceTag",
      "Effect": "Allow",
      "Action": [
        "payment-cryptography:DecryptData"
      ],
      "Resource": "arn:aws::us-east-1:111122223333:key/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": "Alpha"
        }
      }
    }
  ]
}
```

다음 예제 IAM 정책은 보안 주체가 특정 암호화 작업에 대해 계정의 모든 키를 사용하도록 허용합니다. 그러나 보안 주체가 "Type"="Reserved" 태그가 있거나 "Type" 태그가 없는 키에서 이러한 암호화 작업을 사용하는 것은 허용되지 않습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMAllowCryptographicOperations",
      "Effect": "Allow",
      "Action": [
        "payment-cryptography:EncryptData",
        "payment-cryptography:DecryptData",
        "payment-cryptography:ReEncrypt*"
      ],
      "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*"
    },
    {
      "Sid": "IAMDenyOnTag",
```

```

    "Effect": "Deny",
    "Action": [
      "payment-cryptography:EncryptData",
      "payment-cryptography:DecryptData",
      "payment-cryptography:ReEncrypt*"
    ],
    "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/Type": "Reserved"
      }
    }
  },
  {
    "Sid": "IAMDenyNoTag",
    "Effect": "Deny",
    "Action": [
      "payment-cryptography:EncryptData",
      "payment-cryptography:DecryptData",
      "payment-cryptography:ReEncrypt*"
    ],
    "Resource": "arn:aws:kms:*:111122223333:key/*",
    "Condition": {
      "Null": {
        "aws:ResourceTag/Type": "true"
      }
    }
  }
]
}

```

AWS 결제 암호화 키의 주요 속성 이해

적절한 키 관리의 원칙은 키의 범위가 적절하게 지정되고 허용된 작업에만 사용할 수 있어야 한다는 것입니다. 따라서 특정 키는 특정 키 사용 모드에서만 생성할 수 있습니다. 이는 가능한 경우 [TR-31](#) 정의에 따라 사용 가능한 사용 모드에 맞게 조정됩니다.

AWS 결제 암호화를 통해 잘못된 키를 생성하는 것을 방지할 수 있지만 편의를 위해 유효한 조합이 여기에 제공됩니다.

대칭 키

- TR31_B0_BASE_DERIVATION_KEY
 - 허용된 키 알고리즘: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
 - 허용되는 키 사용 모드 조합: { DeriveKey = true}, { NoRestrictions = true}
- TR31_C0_CARD_VERIFICATION_KEY
 - 허용된 키 알고리즘: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
 - 허용되는 주요 사용 모드 조합: {생성=true}, {검증= true}, {생성=true}, {생성=true, 검증= true}, { NoRestrictions = true}
- TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY
 - 허용된 키 알고리즘: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
 - 허용되는 주요 사용 모드 조합: {암호화 = true, 암호 해독 = true, Wrap = true, 줄 바꿈 = true, 줄 바꿈 = true}, {암호화 = true, 줄 바꿈 = true}, {암호 해독 = true}, {암호 해독 = true}, {= true} NoRestrictions
- TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS
 - 허용된 키 알고리즘: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
 - 허용되는 키 사용 모드 조합: {= true}, {= true} DeriveKey NoRestrictions
- TR31_E1_EMV_MKEY_CONFIDENTIALITY
 - 허용된 키 알고리즘: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
 - 허용되는 키 사용 모드 조합: { DeriveKey = true}, { NoRestrictions = true}
- TR31_E2_EMV_MKEY_INTEGRITY
 - 허용된 키 알고리즘: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
 - 허용되는 키 사용 모드 조합: { DeriveKey = true}, { NoRestrictions = true}
- TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS
 - 허용된 키 알고리즘: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
 - 허용되는 키 사용 모드 조합: { DeriveKey = true}, { NoRestrictions = true}
- TR31_E5_EMV_MKEY_CARD_PERSONALIZATION
 - 허용된 키 알고리즘: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
 - 허용되는 키 사용 모드 조합: { DeriveKey = true}, { NoRestrictions = true}
- TR31_E6_EMV_MKEY_OTHER

- 허용되는 키 사용 모드 조합: { DeriveKey = true}, { NoRestrictions = true}
- TR31_K0_KEY_ENCRYPTION_KEY
 - 허용된 키 알고리즘: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
 - 허용되는 주요 사용 모드 조합: {암호화 = true, 암호 해독 = true, 줄 바꿈 = true, 줄 바꿈 = true, 줄 바꿈 = true}, {암호화 = true, 줄 바꿈 = true}, {암호 해독 = true}, {암호 해독 = true}, {= true} NoRestrictions
- TR31_M3_ISO_9797_3_MAC_KEY
 - 허용된 키 알고리즘: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
 - 허용되는 주요 사용 모드 조합: {생성=true}, {검증= true}, {생성=true}, {생성=true}, {= true} NoRestrictions
- TR31_M6_ISO_9797_5_CMAC_KEY
 - 허용된 키 알고리즘: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
 - 허용되는 주요 사용 모드 조합: {생성=true}, {검증= true}, {생성=true}, {생성=true, 검증= true}, {= true} NoRestrictions
- TR31_M7_HMAC_KEY
 - 허용된 키 알고리즘: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
 - 허용되는 주요 사용 모드 조합: {생성=true}, {검증= true}, {생성=true}, {생성=true, 검증= true}, {= true} NoRestrictions
- TR31_P0_PIN_ENCRYPTION_KEY
 - 허용된 키 알고리즘: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
 - 허용되는 주요 사용 모드 조합: {암호화 = true, 암호 해독 = true, Wrap = true, 줄 바꿈 = true, 줄 바꿈 = true}, {암호화 = true, 줄 바꿈 = true}, {암호 해독 = true}, {암호 해독 = true}, {= true} NoRestrictions
- TR31_V1_IBM3624_PIN_VERIFICATION_KEY
 - 허용된 키 알고리즘: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
 - 허용되는 주요 사용 모드 조합: {생성=true}, {검증= true}, {생성=true}, {생성=true}, {= true} NoRestrictions
- TR31_V2_VISA_PIN_VERIFICATION_KEY
 - 허용된 키 알고리즘: TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
 - 허용되는 주요 사용 모드 조합: {생성=true}, {검증= true}, {생성=true}, {생성=true, 검증= true}, {= true} NoRestrictions

비대칭 키

- TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION
 - 허용된 키 알고리즘: RSA_2048 ,RSA_3072 ,RSA_4096
 - 허용되는 주요 사용 모드 조합: { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } , { Encrypt = true, Wrap = true } , { Decrypt = true, Unwrap = true }
 - 참고: 데이터를 암호화하거나 키를 래핑하기 위한 공개 키를 가져올 때 사용할 수 있는 옵션은 {Encrypt = true, Wrap = true} 뿐입니다.
- TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE
 - 허용된 키 알고리즘: RSA_2048 ,RSA_3072 ,RSA_4096
 - 허용되는 키 사용 모드 조합: {Sign = true}, {Verify = true}
 - 참고: 루트 인증서, 중간 인증서 또는 TR-34 서명 인증서 등 서명용 키를 가져올 때 유효한 옵션은 {Verify = true} 뿐입니다.

데이터 작업

AWS 결제 암호화 키를 설정한 후에는 이를 사용하여 암호화 작업을 수행할 수 있습니다. 다양한 작업을 통해 암호화, 해싱, CVV2 생성과 같은 도메인별 알고리즘에 이르기까지 다양한 유형의 활동을 수행합니다.

암호화된 데이터는 매칭된 암호 해독 키(암호화 유형에 따라 대칭 키 또는 프라이빗 키)가 없으면 해독할 수 없습니다. 마찬가지로 해싱 알고리즘과 도메인별 알고리즘도 대칭 키나 퍼블릭 키가 없으면 검증할 수 없습니다.

특정 작업에 유효한 키 유형에 대한 자세한 내용은 [암호화 작업에 유효한 키](#)를 참조하세요.

Note

비프로덕션 환경에서는 테스트 데이터를 사용하는 것이 좋습니다. 비프로덕션 환경에서 프로덕션 키와 데이터(PAN, BDK ID 등)를 사용하면 PCI DSS 및 PCI P2PE와 같은 규정 준수 범위에 영향을 미칠 수 있습니다.

주제

- [데이터 암호화, 복호화 및 재암호화](#)
- [카드 데이터 생성 및 확인](#)
- [PIN 데이터 생성, 변환 및 확인](#)
- [인증 요청\(ARQC\) 암호화 확인](#)
- [HMAC 생성 및 확인](#)
- [암호화 작업을 위한 유효한 키](#)

데이터 암호화, 복호화 및 재암호화

암호화 및 해독 방법은 TDES, AES 및 RSA를 비롯한 다양한 대칭 및 비대칭 기술을 사용하여 데이터를 암호화하거나 해독하는 데 사용될 수 있습니다. [이러한 방법은 DUKPT 및 EMV 기술을 사용하여 파생된 키도 지원합니다.](#) 기본 데이터를 노출하지 않고 새 키로 데이터를 보호하려는 사용 사례의 경우에도 ReEncrypt 명령을 사용할 수 있습니다.

Note

암호화/복호화 함수를 사용하는 경우 모든 입력은 HexBinary로 간주됩니다. 예를 들어 값 1은 31 (16진수) 로 입력되고 소문자 t는 74 (16진수) 로 입력됩니다. 모든 출력값도 hexBinary로 표시됩니다.

[사용 가능한 모든 옵션에 대한 자세한 내용은 암호화, 복호화 및 재암호화에 대한 API 가이드를 참조하십시오.](#)

주제

- [Encrypt data](#)
- [데이터 해독](#)

Encrypt data

[Encrypt Data API는 대칭 및 비대칭 데이터 암호화 키와 DUKPT 및 EMV 파생 키를 사용하여 데이터를 암호화하는 데 사용됩니다.](#) TDES, RSA, AES를 비롯한 다양한 알고리즘과 변형이 지원됩니다.

기본 입력은 데이터를 암호화하는 데 사용되는 암호화 키, 암호화할 HexBinary 형식의 일반 텍스트 데이터, TDES와 같은 블록 암호의 초기화 벡터 및 모드와 같은 암호화 속성입니다. 일반 텍스트 데이터는 크기가 8바이트의 배수, 인 경우 16바이트 TDES, 키 길이의 배수 단위여야 합니다. AES RSA 입력 데이터가 이러한 요구 사항을 충족하지 않는 경우 대칭 키 입력 (TDES, AES, DUKPT, EMV) 을 패딩해야 합니다. 다음 표에는 각 키 유형에 대한 최대 일반 텍스트 길이와 RSA 키에 대해 정의한 패딩 유형이 나와 있습니다. EncryptionAttributes

패딩 유형	RSA_2048	RSA_3072	RSA_4096
OAEP_SHA1	428	684	940
OAEP_SHA256	380	636	892
OAEP_SHA512	252	508	764
PKCS1	488	744	1000
None	488	744	1000

기본 출력값에는 hexBinary 형식의 사이퍼텍스트로 암호화된 데이터와 암호화 키의 체크섬 값이 포함됩니다. 사용 가능한 모든 옵션에 대한 자세한 내용은 [암호화용 API 가이드](#)를 참조하십시오.

예제

- [AES 대칭 키를 사용하여 데이터를 암호화합니다.](#)
- [DUKPT 키를 사용한 데이터 암호화](#)
- [EMV에서 파생된 대칭 키를 사용하여 데이터를 암호화합니다.](#)
- [RSA 키를 사용한 데이터 암호화](#)

AES 대칭 키를 사용하여 데이터를 암호화합니다.

Note

모든 예에서는 관련 키가 이미 존재한다고 가정합니다. 키는 작업을 사용하여 생성하거나 [CreateKey](#) 작업을 사용하여 가져올 수 있습니다. [ImportKey](#)

Example

이 예제에서는 [CreateKey](#) 오퍼레이션을 사용하여 생성하거나 오퍼레이션을 사용하여 가져온 대칭 키를 사용하여 일반 텍스트 데이터를 암호화합니다. [ImportKey](#) 이 작업을 수행하려면 키가 로 설정되고 로 KeyModesOfUse Encrypt 설정되어야 합니다. KeyUsage TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY 자세한 옵션은 [암호화 작업을 위한 키](#)를 참조하십시오.

```
$ aws payment-cryptography-data encrypt-data --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi --plain-text 31323334313233343132333431323334 --encryption-attributes 'Symmetric={Mode=CBC}'
```

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi",
  "KeyCheckValue": "71D7AE",
  "CipherText": "33612AB9D6929C3A828EB6030082B2BD"
}
```

DUKPT 키를 사용한 데이터 암호화

Example

이 예시에서는 [DUKPT 키를 사용하여 일반 텍스트 데이터를 암호화합니다](#). AWS 결제 암호화 지원 및 DUKPT 키. TDES AES 이 작업을 수행하려면 키가 `DeriveKey` 설정되고 `KeyUsage` 로 `KeyModesOfUse` 설정되어 있어야 합니다. `TR31_B0_BASE_DERIVATION_KEY` 자세한 옵션은 [암호화 작업을 위한 키](#)를 참조하세요.

```
$ aws payment-cryptography-data encrypt-data --key-identifier
arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
--plain-text 31323334313233343132333431323334 --encryption-attributes
'Dukpt={KeySerialNumber=FFFF9876543210E00001}'
```

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
  "KeyCheckValue": "71D7AE",
  "CipherText": "33612AB9D6929C3A828EB6030082B2BD"
}
```

EMV에서 파생된 대칭 키를 사용하여 데이터를 암호화합니다.

Example

이 예제에서는 이미 생성된 EMV에서 파생된 대칭 키를 사용하여 일반 텍스트 데이터를 암호화합니다. 다음과 같은 명령을 사용하여 EMV 카드로 데이터를 보낼 수 있습니다. 이 작업을 수행하려면 키가 `or`로 `KeyModesOfUse Derive` 설정되고 `KeyUsage` 설정되어야 합니다. `TR31_E1_EMV_MKEY_CONFIDENTIALITY` `TR31_E6_EMV_MKEY_OTHER` 자세한 내용은 [암호화 작업을 위한 키](#)를 참조하십시오.

```
$ aws payment-cryptography-data encrypt-data --key-identifier
arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
```

```
--plain-text 33612AB9D6929C3A828EB6030082B2BD --encryption-attributes
'Emv={MajorKeyDerivationMode=EMV_OPTION_A,PanSequenceNumber=27,PrimaryAccountNumber=1000000000
InitializationVector=1500000000000999,Mode=CBC}'
```

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
  tqv5yij6wtxx64pi",
  "KeyCheckValue": "71D7AE",
  "CipherText": "33612AB9D6929C3A828EB6030082B2BD"
}
```

RSA 키를 사용한 데이터 암호화

Example

이 예시에서는 작업을 통해 가져온 [RSA 공개 키](#) 사용하여 일반 텍스트 데이터를 암호화합니다. [ImportKey](#) 이 작업을 수행하려면 키가 로 설정되고 로 KeyModesOfUse Encrypt 설정되어야 합니다. KeyUsage TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION 자세한 옵션은 [암호화 작업을 위한 키](#)를 참조하세요.

PKCS #7 또는 현재 지원되지 않는 기타 패딩 체계의 경우 서비스를 호출하기 전에 적용하고 패딩 표시 기 'Asymmetric={}'을 생략하여 패딩 없음을 선택하세요.

```
$ aws payment-cryptography-data encrypt-data --key-identifier
arn:aws:payment-cryptography:us-east-2:111122223333:key/thfezpmsalcfwmsg
--plain-text 31323334313233343132333431323334 --encryption-attributes
'Asymmetric={PaddingType=0AEP_SHA256}'
```

```
{
  "CipherText":
  "12DF6A2F64CC566D124900D68E8AFEAA794CA819876E258564D525001D00AC93047A83FB13 \
  E73F06329A100704FA484A15A49F06A7A2E55A241D276491AA91F6D2D8590C60CDE57A642BC64A897F4832A3930
  \
  0FAEC7981102CA0F7370BFBF757F271EF0BB2516007AB111060A9633D1736A9158042D30C5AE11F8C5473EC70F067
  \
  72590DEA1638E2B41FAE6FB1662258596072B13F8E2F62F5D9FAF92C12BB70F42F2ECDCF56AADF0E311D4118FE3591
  \
  FB672998CCE9D00FFFE05D2CD154E3120C5443C8CF9131C7A6A6C05F5723B8F5C07A4003A5A6173E1B425E2B5E42AD
  \
```

```

7A2966734309387C9938B029AFB20828ACFC6D00CD1539234A4A8D9B94CDD4F23A",
"KeyArn": "arn:aws:payment-cryptography:us-east-1:529027455495:key/5dza7xqd6soanjtb",
"KeyCheckValue": "FF9DE9CE"
}

```

데이터 해독

[Decrypt Data API는 대칭 및 비대칭 데이터 암호화 키와 DUKPT 및 EMV 파생 키를 사용하여 데이터를 복호화하는 데 사용됩니다.](#) TDES, RSA, AES를 비롯한 다양한 알고리즘과 변형이 지원됩니다.

기본 입력값은 데이터를 해독하는 데 사용되는 암호 해독 키, 해독할 hexBinary 형식의 사이퍼텍스트 데이터, 초기화 벡터, 블록 암호 모드 등과 같은 해독 특성입니다. 기본 출력값에는 hexBinary 형식의 일반 텍스트로 해독된 데이터와 암호 해독 키의 체크섬 값이 포함됩니다. [사용 가능한 모든 옵션에 대한 자세한 내용은 복호화를 위한 API 가이드를 참조하십시오.](#)

예제

- [AES 대칭 키를 사용한 데이터 해독](#)
- [DUKPT 키를 사용한 데이터 해독](#)
- [EMV에서 파생된 대칭 키를 사용하여 데이터를 복호화합니다.](#)
- [RSA 키를 사용한 데이터 해독](#)

AES 대칭 키를 사용한 데이터 해독

Example

이 예시에서는 대칭 키를 사용하여 암호문 데이터를 해독합니다. 이 예제는 AES 키를 보여 주지만 지원되기도 합니다. TDES_2KEY TDES_3KEY 이 작업을 수행하려면 키가 로 KeyModesOfUse Decrypt 설정되고 로 KeyUsage 설정되어야 TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY 합니다. 자세한 옵션은 [암호화 작업을 위한 키](#)를 참조하세요.

```

$ aws payment-cryptography-data decrypt-data --key-identifier arn:aws:payment-
cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi --cipher-text
33612AB9D6929C3A828EB6030082B2BD --decryption-attributes 'Symmetric={Mode=CBC}'

```



```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
  tqv5yij6wtxx64pi",
  "KeyCheckValue": "71D7AE",
  "PlainText": "31323334313233343132333431323334"
}
```

DUKPT 키를 사용한 데이터 해독

Note

P2PE 거래에 DUKPT와 함께 암호 해독 데이터를 사용하면 PCI DSS 범위를 결정할 때 고려해야 하는 신용 카드 PAN 및 기타 카드 소유자 데이터가 애플리케이션에 반환될 수 있습니다.

Example

이 예시에서는 Operation을 사용하여 생성했거나 Operation을 사용하여 가져온 [DUKPT 키를 사용하여 암호문](#) 데이터를 해독합니다 [CreateKey](#). [ImportKey](#) 이 작업을 수행하려면 키가 로 설정되고 로 설정되어야 합니다. KeyModesOfUse DeriveKey KeyUsage TR31_B0_BASE_DERIVATION_KEY 자세한 옵션은 [암호화 작업을 위한 키](#)를 참조하세요. TDES 알고리즘에 DUKPT를 사용하는 경우 사이퍼텍스트 데이터 길이는 16바이트의 배수여야 합니다. AES 알고리즘의 경우 사이퍼텍스트 데이터 길이는 32바이트의 배수여야 합니다.

```
$ aws payment-cryptography-data decrypt-data --key-identifier
arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
--cipher-text 33612AB9D6929C3A828EB6030082B2BD --decryption-attributes
'Dukpt={KeySerialNumber=FFFF9876543210E00001}'
```

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
  tqv5yij6wtxx64pi",
  "KeyCheckValue": "71D7AE",
  "PlainText": "31323334313233343132333431323334"
}
```

```
}

```

EMV에서 파생된 대칭 키를 사용하여 데이터를 복호화합니다.

Example

이 예제에서는 작업을 사용하여 생성되었거나 작업을 사용하여 가져온 EMV 파생 대칭 키를 사용하여 암호문 데이터를 해독합니다. [CreateKeyImportKey](#) 이 작업을 수행하려면 키가 OR로 설정되고 설정되어야 합니다. KeyModesOfUse Derive KeyUsage TR31_E1_EMV_MKEY_CONFIDENTIALITY TR31_E6_EMV_MKEY_OTHER 자세한 내용은 [암호화 작업을 위한 키](#) 참조하십시오.

```
$ aws payment-cryptography-data decrypt-data --key-identifier
arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
--cipher-text 33612AB9D6929C3A828EB6030082B2BD --decryption-attributes
'Emv={MajorKeyDerivationMode=EMV_OPTION_A, PanSequenceNumber=27, PrimaryAccountNumber=1000000000
InitializationVector=15000000000000999, Mode=CBC}'

```

```
{
"KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi",
"KeyCheckValue": "71D7AE",
"PlainText": "31323334313233343132333431323334"
}

```

RSA 키를 사용한 데이터 해독

Example

이 예제에서는 작업을 통해 생성된 RSA [키 쌍](#)을 사용하여 암호문 데이터를 해독합니다. [CreateKey](#) 이 작업을 수행하려면 키를 활성화로 설정하고 로 KeyModesOfUse 설정해야 합니다. Decrypt KeyUsage TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION 자세한 옵션은 [암호화 작업을 위한 키](#)를 참조하세요.

PKCS #7 또는 현재 지원되지 않는 다른 패딩 체계의 경우, 패딩 표시기 'Asymmetric={}'을 생략하여 패딩 없음을 선택하고 서비스를 호출한 후 패딩을 제거하세요.

```
$ aws payment-cryptography-data decrypt-data \

```

```
--key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/5dza7xqd6soanjtb --cipher-text
8F4C1CAFE7A5DEF9A40BEDE7F2A264635C... \
--decryption-attributes 'Asymmetric={PaddingType=0AEP_SHA256}'
```

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-1:529027455495:key/5dza7xqd6soanjtb",
  "KeyCheckValue": "FF9DE9CE",
  "PlainText": "31323334313233343132333431323334"
}
```

카드 데이터 생성 및 확인

카드 데이터를 생성하고 CVV, CVV2, CVC, DCVV와 같은 카드 데이터에서 파생된 데이터가 카드 데이터에 통합되는지 확인합니다.

주제

- [카드 데이터 생성](#)
- [카드 데이터 확인](#)

카드 데이터 생성

Generate Card Data API는 CVV, CVV2 또는 동적 CVV2와 같은 알고리즘을 사용하여 카드 데이터를 생성하는 데 사용됩니다. 이 명령에 사용할 수 있는 키를 보려면 [암호화 작업에 유효한 키](#) 섹션을 참조하세요.

Example

이 예제에서는 [PAN](#) 및 카드 만료일을 입력하여 주어진 PAN에 대한 CVV/CVV2를 생성합니다. 카드 만료일은 MMYYY 또는 YYMM 형식일 수 있지만 검증이 제대로 작동하려면 이후의 모든 사용과 일치해야 합니다. 여기서는 카드 확인 키가 [생성](#)되었다고 가정합니다.

```
$ aws payment-cryptography-data generate-card-validation-data --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pig --primary-account-number=171234567890123 --generation-attributes CardVerificationValue2={CardExpiryDate=0123}
```

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
  tqv5yij6wtxx64pi",
  "KeyCheckValue": "CADD1",
  "ValidationData": "801"
}
```

카드 데이터 확인

Verify Card Data는 DISCOVER_DYNAMIC_CARD_VERIFICATION_CODE와 같은 암호화 원리에 의존하는 결제 알고리즘을 사용하여 생성된 데이터를 확인하는 데 사용됩니다.

입력 값은 일반적으로 인바운드 트랜잭션의 일부로 발급자 또는 지원 플랫폼 파트너에게 제공됩니다. ARQC 암호화(EMV 칩 카드에 사용)를 확인하려면 [ARQC 확인](#)을 참조하세요.

값이 확인되면 API는 http/200을 반환합니다. 값이 확인되지 않은 경우 http/400을 반환합니다.

Example

이 예시에서는 주어진 PAN에 대해 CVV/CVV2를 검증해 보겠습니다. CVV2는 일반적으로 카드 소유자 또는 사용자가 거래 시간 동안 검증을 위해 제공합니다. 입력값을 검증하기 위해 런타임 시 [검증에 사용할 키\(CVK\)](#), [PAN](#), 카드 만료일 및 CVV2 입력 등의 값이 제공됩니다. 카드 만료 형식은 초기 값 생성에 사용된 형식과 일치해야 합니다.

```
$ aws payment-cryptography-data verify-card-validation-data --key-identifier
arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
--primary-account-number=171234567890123 --verification-attributes
CardVerificationValue2={CardExpiryDate=0123} --validation-data 801
```

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
  tqv5yij6wtxx64pi",
  "KeyCheckValue": "CADD1"
}
```

PIN 데이터 생성, 변환 및 확인

PIN 데이터 함수를 사용하면 무작위 핀, 핀 확인 값(PVV)을 생성하고 PVV 또는 PIN 오프셋에 대해 인바운드 암호화 핀을 검증할 수 있습니다.

핀 변환을 사용하면 PCI PIN 요구 사항 1에 지정된 일반 텍스트로 핀을 노출하지 않고도 한 작업 키에서 다른 작업 키로 핀을 변환할 수 있습니다.

Note

PIN 생성 및 검증은 일반적으로 발급자 기능이고 PIN 변환은 일반적인 취득자 기능이므로 권한이 가장 적은 액세스를 고려하고 시스템 사용 사례에 적합한 정책을 설정하는 것이 좋습니다.

주제

- [PIN 데이터 변환](#)
- [PIN 데이터 생성](#)
- [PIN 데이터 확인](#)

PIN 데이터 변환

PIN 데이터 변환 함수는 암호화된 데이터가 HSM을 벗어나지 않고 한 키 세트에서 다른 키 세트로 암호화된 PIN 데이터를 변환하는 데 사용됩니다. 작업 키는 변경해야 하지만 처리 시스템에서 데이터를 해독할 필요가 없거나 해독이 허용되지 않는 P2PE 암호화에 사용됩니다. 기본 입력값은 암호화된 데이터, 데이터를 암호화하는 데 사용되는 암호화 키, 입력 값을 생성하는 데 사용되는 파라미터입니다. 다른 입력 세트는 출력값을 암호화하는 데 사용되는 키와 해당 출력값을 생성하는 데 사용되는 파라미터와 같은 요청된 출력 파라미터입니다. 기본 출력값은 새로 암호화된 데이터 세트와 이를 생성하는 데 사용되는 파라미터입니다.

Note

AES 키 유형은 ISO 형식 [4핀 블록](#)만 지원합니다.

주제

- [PEK에서 DUKPT로의 PIN](#)

- [DUKPT에서 AWK로의 PIN](#)

PEK에서 DUKPT로의 PIN

Example

이 예제에서는 ISO 0 PIN 블록을 사용하는 PEK TDES 암호화의 PIN을 [DUKPT](#) 알고리즘을 사용하는 AES ISO 4 PIN 블록으로 변환해 보겠습니다. 일반적으로 이는 결제 단말기가 ISO 4에서 핀을 암호화 한 다음 다운스트림 처리를 위해 TDES로 다시 변환하는 역순으로 수행될 수 있습니다.

```
$ aws payment-cryptography-data translate-pin-data --encrypted-pin-block
"AC17DC148BDA645E" --incoming-translation-
attributes=IsoFormat0='{PrimaryAccountNumber=171234567890123}' --incoming-
key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt --outgoing-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/4pmyquwjs3yj4vwe --outgoing-translation-attributes
IsoFormat4="{PrimaryAccountNumber=171234567890123}" --outgoing-dukpt-attributes
KeySerialNumber="FFFF9876543210E00008"
```

```
{
  "PinBlock": "1F4209C670E49F83E75CC72E81B787D9",
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt --outgoing-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/4pmyquwjs3yj4vwe",
  "KeyCheckValue": "7CC9E2"
}
```

DUKPT에서 AWK로의 PIN

Example

이 예시에서는 AES [DUKPT](#)로 암호화된 PIN을 [AWK](#)에서 암호화된 핀으로 변환해 보겠습니다. 기능적으로는 이전 예와 반대입니다.

```
$ aws payment-cryptography-data translate-pin-data --encrypted-pin-
block "1F4209C670E49F83E75CC72E81B787D9" --outgoing-translation-
attributes=IsoFormat0='{PrimaryAccountNumber=171234567890123}' --outgoing-
key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
```

```
ivi5ksfsuplneuyt --incoming-key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/4pmyquwjs3yj4vve --incoming-translation-attributes IsoFormat4="{PrimaryAccountNumber=171234567890123}" --incoming-dukpt-attributes KeySerialNumber="FFFF9876543210E00008"
```

```
{
  "PinBlock": "AC17DC148BDA645E",
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/ivi5ksfsuplneuyt",
  "KeyCheckValue": "FE23D3"
}
```

PIN 데이터 생성

PIN 데이터 생성 함수는 트랜잭션 또는 승인 기간 동안 사용자의 핀 입력을 검증하는 데 사용되는 [PVV](#) 및 핀 블록 오프셋과 같은 PIN 관련 값을 생성하는 데 사용됩니다. 또한 이 API는 다양한 알고리즘을 사용하여 임의의 새로운 핀을 생성할 수 있습니다.

Example

이 예시에서는 Visa 핀 체계를 사용하여 새로운 (무작위) 핀을 생성합니다. 여기서 출력은 암호화됩니다 PIN block (. PinData PinBlock) 와 a PVV (PinData.offset). 키 입력값은 [PAN](#), [Pin Verification Key](#), [Pin Encryption Key](#), PIN block format입니다.

```
$ aws payment-cryptography-data generate-pin-data --generation-key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/37y2tsl45p5zjbh2 --encryption-key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/ivi5ksfsuplneuyt --primary-account-number 171234567890123 --pin-block-format ISO_FORMAT_0 --generation-attributes VisaPin={PinVerificationKeyIndex=1}
```

```
{
  "GenerationKeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/37y2tsl45p5zjbh2",
  "GenerationKeyCheckValue": "7F2363",
  "EncryptionKeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/ivi5ksfsuplneuyt",
  "EncryptionKeyCheckValue": "7CC9E2",
}
```

```

    "EncryptedPinBlock": "AC17DC148BDA645E",
    "PinData": {
      "VerificationValue": "5507"
    }
  }
}

```

PIN 데이터 확인

PIN 데이터 확인 함수는 PIN이 올바른지 확인하는 데 사용됩니다. 여기에는 일반적으로 이전에 저장된 핀 값을 카드 소유자가 POI에 입력한 값과 비교하는 작업이 포함됩니다. 이러한 함수는 각 소스의 기본 값을 노출하지 않고 두 값을 비교합니다.

암호화된 PIN 검증

Example

이 예시에서는 지정된 PAN의 PIN을 검증해 보겠습니다. PIN은 일반적으로 카드 소유자 또는 사용자가 트랜잭션 시간 동안 검증을 위해 제공하며, PIN은 파일에 있는 값(암호화된 값으로 제공됨)과 비교됩니다. 이 입력값을 검증하기 위해 입력 핀을 암호화하는 데 사용되는 키(보통 IWK), [PAN](#) 및 검증할 값 (PVV 또는 PIN offset)과 같은 값도 런타임에 제공됩니다.

AWS 결제 암호화로 핀을 검증할 수 있는 경우 http/200이 반환됩니다. 핀이 검증되지 않은 경우 http/400을 반환합니다.

```

$ aws payment-cryptography-data verify-pin-data --verification-key-identifier
  arn:aws:payment-cryptography:us-east-2:111122223333:key/37y2tsl45p5zjbh2 --encryption-
  key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/ivi5ksfsuplneuyt
  --primary-account-number 171234567890123 --pin-block-format ISO_FORMAT_0 --
  verification-attributes VisaPin="{PinVerificationKeyIndex=1,VerificationValue=5507}" --
  encrypted-pin-block AC17DC148BDA645E

```

```

{
  "VerificationKeyArn": "arn:aws:payment-cryptography:us-
  east-2:111122223333:key/37y2tsl45p5zjbh2",
  "VerificationKeyCheckValue": "7F2363",
  "EncryptionKeyArn": "arn:aws:payment-cryptography:us-
  east-2:111122223333:key/ivi5ksfsuplneuyt",
  "EncryptionKeyCheckValue": "7CC9E2",
}

```


}

인증 요청(ARQC) 암호화 확인

인증 요청 암호화 API는 [ARQC](#)를 확인하는 데 사용됩니다. ARQC 생성은 AWS 결제 암호화의 범위를 벗어나며 일반적으로 거래 승인 기간 동안 EMV 칩 카드 (또는 모바일 지갑과 같은 디지털 카드) 에서 수행됩니다. ARQC는 각 트랜잭션마다 고유하며 카드의 유효성을 암호화하여 보여주고 트랜잭션 데이터가 현재(예상) 트랜잭션과 정확히 일치하는지 확인하는 데 사용됩니다.

AWS 결제 암호화는 [EMV 4.4 Book 2](#) 및 Visa 및 Mastercard에서 사용하는 기타 체계에 정의된 값을 포함하여 ARQC를 검증하고 선택적 ARQC 값을 생성하는 다양한 옵션을 제공합니다. [사용 가능한 모든 옵션의 전체 목록은 API 가이드의 섹션을 참조하십시오. VerifyCardValidationData](#)

ARQC 암호화에는 일반적으로 다음과 같은 입력이 필요합니다 (구현에 따라 다를 수 있음).

- [PAN](#) - 필드에 지정됩니다. PrimaryAccountNumber
- [PAN 시퀀스 번호 \(PSN\)](#) - 필드에 PanSequenceNumber 지정됨
- CSK (공통 세션 키) 와 같은 키 도출 방법 - 에 지정됨 SessionKeyDerivationAttributes
- 마스터 키 파생 모드 (예: EMV 옵션 A) - 에 지정됨 MajorKeyDerivationMode
- 거래 데이터 - 금액 및 날짜와 같은 다양한 거래, 터미널 및 카드 데이터의 문자열 - 필드에 지정됨 TransactionData
- [발급자 마스터 키](#) - 개별 거래를 보호하는 데 사용되는 암호 (AC) 키를 도출하는 데 사용되며 필드에 지정된 마스터 키 KeyIdentifier

주제

- [트랜잭션 데이터 구축](#)
- [트랜잭션 데이터 패딩](#)
- [예](#)

트랜잭션 데이터 구축

트랜잭션 데이터 필드의 정확한 내용 (및 순서) 은 구현 및 네트워크 체계에 따라 다르지만 최소 권장 필드 (및 연결 순서) 는 [EMV 4.4 Book 2](#) 섹션 8.1.1 - 데이터 선택에 정의되어 있습니다. 처음 세 필드가 금액(17.00), 기타 금액(0.00), 구매 국가인 경우 트랜잭션 데이터는 다음과 같이 시작됩니다.

- 000000001700 - 금액 - 12자리(소수점 두 자리 생략)
- 000000000000 - 기타 금액 - 12자리(소수점 두 자리 생략)
- 0124 - 4자리 국가 코드
- 출력값 (부분) 트랜잭션 데이터 - 00000000170000000000000000124

트랜잭션 데이터 패딩

서비스로 전송하기 전에 트랜잭션 데이터를 패딩해야 합니다. 대부분의 체계에서는 ISO 9797 메서드 2 패딩을 사용하며 16진수 문자열은 필드가 암호화 블록 크기의 배수가 될 때까지 16진수 80 다음에 00이 추가됩니다. TDES의 경우 8바이트 또는 16자, AES의 경우 16바이트 또는 32자입니다. 대안(메서드 1)은 흔하지는 않지만 00만 패딩 문자로 사용합니다.

ISO 9797 메서드 1 패딩

패딩 해제:

00000000170000000000000008400080008000084016051700000000093800000B03011203(74자 또는 37바이트)

패딩:

00000000170000000000000008400080008000084016051700000000093800000B03011203000000(80자 또는 40바이트)

ISO 9797 메서드 2 패딩

패딩 해제:

00000000170000000000000008400080008000084016051700000000093800000B1F220103000000(80자 또는 40바이트)

패딩:

00000000170000000000000008400080008000084016051700000000093800000B1F220103000000800000(84자 또는 44바이트)

예

Visa CVN10

Example

이 예제에서는 Visa CVN10을 사용하여 생성된 ARQC를 검증합니다.


```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk",
  "KeyCheckValue": "08D7B4"
}
```

HMAC 생성 및 확인

메시지 인증 코드(MAC)는 일반적으로 메시지의 무결성(수정 여부)을 인증하는 데 사용됩니다. HMAC(해시 기반 메시지 인증 코드), CBC-MAC 및 CMAC(암호 기반 메시지 인증 코드)와 같은 암호화 해시는 암호화를 활용하여 MAC 발신자에 대한 추가 보안을 제공합니다. HMAC는 해시 함수를 기반으로 하는 반면 CMAC는 블록 암호를 기반으로 합니다.

이 서비스의 모든 HMAC 알고리즘은 암호화 해시 함수와 공유 비밀 키를 결합합니다. 키의 키 구성 요소와 같은 메시지와 비밀 키를 가져와서 고유한 태그 또는 mac를 반환합니다. 메시지의 한 문자라도 변경되거나 비밀 키가 변경되면 결과 태그는 완전히 달라집니다. 비밀 키를 요구함으로써 암호화 HMAC는 신뢰성도 제공합니다. 비밀 키가 없다면 동일한 mac를 생성할 수 없습니다. 암호화 HMAC는 디지털 서명처럼 작동하지만 서명과 확인 모두에 단일 키를 사용하기 때문에 대칭 서명이라고도 합니다.

AWS 결제 암호화는 다음과 같은 여러 유형의 MAC을 지원합니다.

ISO9797 알고리즘 1

ISO9797_ALGORITHM1의 KeyUsage로 표시

ISO9797 알고리즘 3(Retail MAC)

ISO9797_ALGORITHM3의 KeyUsage로 표시

ISO9797 알고리즘 5(CMAC)

TR31_M6_ISO_9797_5_CMAC_KEY의 KeyUsage로 표시

HMAC

TR31_M7_HMAC_KEY의 KeyUsage로 표시(HMAC_SHA224, HMAC_SHA256, HMAC_SHA384, HMAC_SHA512 포함)

주제

- [MAC 생성](#)
- [MAC 검증](#)

MAC 생성

MAC API 생성은 알려진 데이터 값을 사용하여 송신 당사자와 수신 당사자 간의 데이터 검증을 위한 MAC(메시지 인증 코드)를 생성함으로써 카드 마그네틱 스트라이프의 트랙 데이터와 같은 카드 관련 데이터를 인증하는 데 사용됩니다. MAC 생성에 사용되는 데이터에는 메시지 데이터, 비밀 MAC 암호화 키 및 전송을 위한 고유한 MAC 값을 생성하는 MAC 알고리즘이 포함됩니다. MAC의 수신 당사자는 동일한 MAC 메시지 데이터, MAC 암호화 키 및 알고리즘을 사용하여 비교 및 데이터 인증을 위해 다른 MAC 값을 다시 만듭니다. 메시지의 한 문자라도 변경되거나 인증에 사용된 MAC 키가 동일하지 않은 경우 결과 MAC 값은 달라집니다. API는 이 작업을 위한 DUPKT MAC, HMAC 및 EMV MAC 암호화 키를 지원합니다.

message-data의 입력 값은 hexBinary 데이터여야 합니다.

이 예제에서는 HMAC 알고리즘 HMAC_SHA256과 HMAC 암호화 키를 사용하여 카드 데이터 인증을 위한 HMAC(해시 기반 메시지 인증 코드)를 생성합니다. 키에는 KeyUse가 TR31_M7_HMAC_KEY로 설정되고 KeyModeSofUse는 Generate로 설정되어 있어야 합니다. MAC 키는 [CreateKey](#)를 호출하여 AWS 결제 암호화를 통해 생성하거나 [ImportKey](#)를 호출하여 가져올 수 있습니다.

Example

```
$ aws payment-cryptography-data generate-mac \
  --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
  qnobl5lghrzunce6 \
  --message-data
  "3b313038383439303031303733393431353d32343038323236303030373030303f33" \
  --generation-attributes Algorithm=HMAC_SHA256
```

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
  qnobl5lghrzunce6",
  "KeyCheckValue": "2976E7",
  "Mac": "ED87F26E961C6D0DDB78DA5038AA2BDDEA0DCE03E5B5E96BDDD494F4A7AA470C"
}
```

MAC 검증

MAC API가 카드 관련 데이터 인증을 위한 MAC(메시지 인증 코드)를 확인하는 데 사용되는지 검증합니다. 인증을 위한 MAC 값을 재생성하려면 MAC 생성 시 사용한 것과 동일한 암호화 키를 사용해야 합니다. MAC 암호화 키는 [CreateKey](#)를 호출하여 AWS 결제 암호화를 통해 생성하거나 [ImportKey](#)를 호출하여 가져올 수 있습니다. API는 이 작업을 위한 DUPKT MAC, HMAC 및 EMV MAC 암호화 키를 지원합니다.

값이 확인되면 응답 파라미터 `MacDataVerificationSuccessful`는 `Http/200`를 반환하고, 그렇지 않으면 `Mac verification failed`를 나타내는 메시지와 함께 `Http/400`를 반환합니다.

이 예제에서는 HMAC 알고리즘 HMAC_SHA256과 HMAC 암호화 키를 사용하여 카드 데이터 인증을 위한 HMAC(해시 기반 메시지 인증 코드)를 검증합니다. 키에는 `KeyUse`가 `TR31_M7_HMAC_KEY`로 설정되고 `KeyModeSofUse`는 `Verify`로 설정되어 있어야 합니다.

Example

```
$ aws payment-cryptography-data verify-mac \
  --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
qno151ghrzunce6 \
  --message-data
"3b343038383439303031303733393431353d32343038323236303030373030303f33" \
  --verification-attributes='Algorithm=HMAC_SHA256' \
  --mac ED87F26E961C6D0DDB78DA5038AA2BDDEA0DCE03E5B5E96BDDD494F4A7AA470C
```

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
qno151ghrzunce6",
  "KeyCheckValue": "2976E7",
}
```

암호화 작업을 위한 유효한 키

특정 키는 특정 작업에만 사용할 수 있습니다. 또한 일부 작업에서는 키의 키 사용 모드가 제한될 수 있습니다. 허용되는 조합은 다음 표를 참조하세요.

Note

특정 조합이 허용되더라도 CVV 코드 (generate)를 생성한 후 이를 확인할 수 없는 (verify)와 같이 사용할 수 없는 상황을 만들 수 있습니다.

주제

- [GenerateCardData](#)
- [VerifyCardData](#)
- [GeneratePinData \(비자/ABA 제도용\)](#)
- [GeneratePinData \(용IBM3624\)](#)
- [VerifyPinData \(비자/ABA 제도용\)](#)
- [VerifyPinData \(용IBM3624\)](#)
- [데이터 해독](#)
- [데이터 암호화](#)
- [PIN 데이터 변환](#)
- [VerifyAuthRequestCryptogram](#)
- [미사용 키 유형](#)

GenerateCardData

API 엔드포인트	암호화 연산 또는 알고리즘	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
GenerateCardData	<ul style="list-style-type: none"> • AMEX_CARD_SECURITY_CODE_VERIFICATION_1 • AMEX_CARD_SECURITY_CODE_VERIFICATION_2 	TR31_C0_CARD_VERIFICATION_KEY	<ul style="list-style-type: none"> • TDES_2KEY • TDES_3KEY 	{ Generate = true }, { Generate = true, Verify = true }

API 엔드포인트	암호화 연산 또는 알고리즘	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
GenerateCardData	<ul style="list-style-type: none"> CARD_VERIFICATION_VALUE_1 CARD_VERIFICATION_VALUE_2 	TR31_C0_CARD_VERIFICATION_KEY	<ul style="list-style-type: none"> TDES_2KEY 	{ Generate = true }, { Generate = true, Verify = true }
GenerateCardData	<ul style="list-style-type: none"> CARDHOLDER_AUTHENTICATION_VERIFICATION_VALUE 	TR31_E6_EMV_MKEY_OTHER	<ul style="list-style-type: none"> TDES_2KEY 	{ DeriveKey = 참 }
GenerateCardData	<ul style="list-style-type: none"> DYNAMIC_CARD_VERIFICATION_CODE 	TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS	<ul style="list-style-type: none"> TDES_2KEY 	{ DeriveKey = 트루 }
GenerateCardData	<ul style="list-style-type: none"> DYNAMIC_CARD_VERIFICATION_VALUE 	TR31_E6_EMV_MKEY_OTHER	<ul style="list-style-type: none"> TDES_2KEY 	{ DeriveKey = 트루 }

VerifyCardData

암호화 연산 또는 알고리즘	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
<ul style="list-style-type: none"> AMEX_CARD_SECURITY_CODE_VERIFICATION_1 	TR31_C0_CARD_VERIFICATION_KEY	<ul style="list-style-type: none"> TDES_2KEY TDES_3KEY 	{ Generate = true }, { Generate = true, Verify = true }

암호화 연산 또는 알고리즘	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
<ul style="list-style-type: none"> AMEX_CARD_SECURITY_CODE_VERIFICATION_2 			
<ul style="list-style-type: none"> CARD_VERIFICATION_VALUE_1 CARD_VERIFICATION_VALUE_2 	TR31_C0_CARD_VERIFICATION_KEY	<ul style="list-style-type: none"> TDES_2KEY 	{ Generate = true }, { Generate = true, Verify = true }
<ul style="list-style-type: none"> CARDHOLDER_AUTHENTICATION_VERIFICATION_VALUE 	TR31_E6_EMV_MKEY_OTHER	<ul style="list-style-type: none"> TDES_2KEY 	{ DeriveKey = 트루 }
<ul style="list-style-type: none"> DYNAMIC_CARD_VERIFICATION_CODE 	TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS	<ul style="list-style-type: none"> TDES_2KEY 	{ DeriveKey = 트루 }
<ul style="list-style-type: none"> DYNAMIC_CARD_VERIFICATION_VALUE 	TR31_E6_EMV_MKEY_OTHER	<ul style="list-style-type: none"> TDES_2KEY 	{ DeriveKey = 트루 }

GeneratePinData (비자/ABA 제도용)

VISA_PIN or VISA_PIN_VERIFICATION_VALUE

키 유형	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
PIN 암호화 키	TR31_P0_P IN_ENCRYPT TION_KEY	<ul style="list-style-type: none"> TDES_2KEY TDES_3KEY 	<ul style="list-style-type: none"> { Encrypt = true, Wrap = true } { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } NoRestrictions {= true}
PIN 생성 키	TR31_V2_V ISA_PIN_VERIFICATI ON_KEY	<ul style="list-style-type: none"> TDES_3KEY 	<ul style="list-style-type: none"> { Generate = true } { Generate = true, Verify = true }

GeneratePinData (용 IBM3624)

IBM3624_PIN_OFFSET, IBM3624_NATURAL_PIN, IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET)

키 유형	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
PIN 암호화 키	TR31_P0_P IN_ENCRYPT TION_KEY	<ul style="list-style-type: none"> TDES_2KEY TDES_3KEY 	<p>IBM3624_NATURAL_PIN, IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET의 경우</p> <ul style="list-style-type: none"> { Encrypt = true, Wrap = true } { Encrypt = true, Decrypt = true,

키 유형	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
			Wrap = true, Unwrap = true } <ul style="list-style-type: none"> { NoRestrictions = 참 } IBM 3624_PIN_OFFSET의 경우 <ul style="list-style-type: none"> { Encrypt = true, Unwrap = true } { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } { NoRestrictions = 트루 }
PIN 생성 키	TR31_V1_IBM3624_PIN_VERIFICATION_KEY	<ul style="list-style-type: none"> TDES_3KEY 	<ul style="list-style-type: none"> { Generate = true } { Generate = true, Verify = true }

VerifyPinData (비자/ABA 제도용)

VISA_PIN

키 유형	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
PIN 암호화 키	TR31_P0_PIN_ENCRYPTION_KEY	<ul style="list-style-type: none"> TDES_2KEY TDES_3KEY 	<ul style="list-style-type: none"> { Decrypt = true, Unwrap = true } { Encrypt = true, Decrypt = true,

키 유형	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
			Wrap = true, Unwrap = true } <ul style="list-style-type: none"> • NoRestrictions {= true}
PIN 생성 키	TR31_V2_VISA_PIN_VERIFICATION_KEY	<ul style="list-style-type: none"> • TDES_3KEY 	<ul style="list-style-type: none"> • { Verify = true } • { Generate = true, Verify = true }

VerifyPinData (용 IBM3624)

IBM3624_PIN_OFFSET, IBM3624_NATURAL_PIN, IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET)

키 유형	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
PIN 암호화 키	TR31_P0_PIN_ENCRYPTION_KEY	<ul style="list-style-type: none"> • TDES_2KEY • TDES_3KEY 	IBM3624_NATURAL_PIN, IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET의 경우 <ul style="list-style-type: none"> • { Decrypt = true, Unwrap = true } • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } • { NoRestrictions = 참 }

키 유형	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
PIN 확인 키	TR31_V1_I BM3624_PI N_VERIFIC ATION_KEY	<ul style="list-style-type: none"> TDES_3KEY 	<ul style="list-style-type: none"> { Verify = true } { Generate = true, Verify = true }

데이터 해독

키 유형	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
DUKPT	TR31_B0_B ASE_DERIV ATION_KEY	<ul style="list-style-type: none"> TDES_2KEY AES_128 AES_192 AES_256 	<ul style="list-style-type: none"> { DeriveKey = 트루 } { NoRestrictions = 트루 }
EMV	TR31_E1_E MV_MKEY_C ONFIDENTIALITY TR31_E6_E MV_MKEY_OTHER	<ul style="list-style-type: none"> TDES_2KEY 	<ul style="list-style-type: none"> { DeriveKey = 트루 }
RSA	TR31_D1_A SYMMETRIC _KEY_FOR_ DATA_ENCRYPTION	<ul style="list-style-type: none"> RSA_2048 RSA_3072 RSA_4096 	<ul style="list-style-type: none"> { Decrypt = true, Unwrap=true } { Encrypt=true, Wrap=true, Decrypt = true, Unwrap=true }
대칭 키	TR31_D0_S YMMETRIC_ DATA_ENCR YPTION_KEY	<ul style="list-style-type: none"> TDES_2KEY TDES_3KEY AES_128 	<ul style="list-style-type: none"> { Decrypt = true, Unwrap=true } { Encrypt=true, Wrap=true, Decrypt

키 유형	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
		<ul style="list-style-type: none"> AES_192 AES_256 	<ul style="list-style-type: none"> = true, Unwrap=true} { NoRestrictions = true}

데이터 암호화

키 유형	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
DUKPT	TR31_B0_B ASE_DERIVATION_KEY	<ul style="list-style-type: none"> TDES_2KEY AES_128 AES_192 AES_256 	<ul style="list-style-type: none"> { DeriveKey = true} { NoRestrictions = true}
EMV	TR31_E1_E MV_MKEY_CONFIDENTIALITY TR31_E6_E MV_MKEY_OTHER	<ul style="list-style-type: none"> TDES_2KEY 	<ul style="list-style-type: none"> { DeriveKey = true}
RSA	TR31_D1_A SYMMETRIC_KEY_FOR_DATA_ENCRYPTION	<ul style="list-style-type: none"> RSA_2048 RSA_3072 RSA_4096 	<ul style="list-style-type: none"> { Encrypt = true, Wrap=true} {Encrypt=true, Wrap=true, Decrypt = true, Unwrap=true}
대칭 키	TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY	<ul style="list-style-type: none"> TDES_2KEY TDES_3KEY AES_128 AES_192 	<ul style="list-style-type: none"> {Encrypt = true, Wrap=true} {Encrypt=true, Wrap=true, Decrypt

키 유형	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
		<ul style="list-style-type: none"> AES_256 	= true, Unwrap=true} <ul style="list-style-type: none"> { NoRestrictions = true }

PIN 데이터 변환

Direction	키 유형	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
인바운드 데이터 소스	DUKPT	TR31_B0_B ASE_DERIVATION_KEY	<ul style="list-style-type: none"> TDES_2KEY AES_128 AES_192 AES_256 	<ul style="list-style-type: none"> { DeriveKey = true } { NoRestrictions = true }
인바운드 데이터 소스	비 DUKPT(PEK, AWK, IWK 등)	TR31_P0_PIN_ENCRYPTION_KEY	<ul style="list-style-type: none"> TDES_2KEY TDES_3KEY AES_128 AES_192 AES_256 	<ul style="list-style-type: none"> { Decrypt = true, Unwrap = true } { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } { NoRestrictions = true }
아웃바운드 데이터 대상	DUKPT	TR31_B0_B ASE_DERIVATION_KEY	<ul style="list-style-type: none"> TDES_2KEY AES_128 AES_192 AES_256 	<ul style="list-style-type: none"> { DeriveKey = true } { NoRestrictions = true }

Direction	키 유형	허용된 키 사용	허용된 키 알고리즘	허용된 키 사용 모드 조합
아웃바운드 데이터 대상	비 DUKPT(PEK, IWK, AWK 등)	TR31_P0_P IN_ENCRYPTION_KEY	<ul style="list-style-type: none"> • TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256 	<ul style="list-style-type: none"> • { Encrypt = true, Wrap = true } • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } • { NoRestrictions = 트루 }

VerifyAuthRequestCryptogram

허용된 키 사용	EMV 옵션	허용된 키 알고리즘	허용된 키 사용 모드 조합
<ul style="list-style-type: none"> • 옵션 A • 옵션 B 	TR31_E0_E MV_MKEY_A PP_CRYPTOGRAMS	<ul style="list-style-type: none"> • TDES_2KEY 	<ul style="list-style-type: none"> • { DeriveKey = 트루 }

미사용 키 유형

다음 키 유형은 현재 AWS 결제 암호화에서 사용되지 않습니다.

- TR31_K1_KEY_BLOCK_PROTECTION_KEY
- TR31_P1_PIN_GENERATION_KEY
- TR31_K3_ASYMMETRIC_KEY_FOR_KEY_AGREEMENT

AWS 결제 암호화의 보안

클라우드 AWS 보안이 최우선 과제입니다. AWS 고객은 가장 보안에 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 기업과 기업 간의 AWS 공동 책임입니다. [공동 책임 모델](#)은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 -AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호하는 역할을 합니다. AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. AWS 결제 암호화에 적용되는 규정 준수 프로그램에 대해 자세히 알아보려면 규정 준수 [프로그램별 AWS 범위 내 서비스 규정 준수 프로그램](#)이 참조하십시오.
- 클라우드에서의 보안 — 사용하는 AWS 서비스에 따라 책임이 결정됩니다. 또한 귀하는 귀사의 데이터의 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 항목은 AWS 결제 암호화를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 보안 및 규정 준수 목표를 충족하도록 AWS 결제 암호화를 구성하는 방법을 보여줍니다. 또한 AWS 결제 암호화 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

주제

- [AWS 결제 암호화에서의 데이터 보호](#)
- [결제 암호화의 탄력성 AWS](#)
- [의 인프라 보안 AWS Payment Cryptography](#)
- [결제 암호화에 대한 보안 모범 사례 AWS](#)

AWS 결제 암호화에서의 데이터 보호

AWS [공동 책임 모델](#) AWS 결제 암호화의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 은 (는) 모두를 실행하는 글로벌 인프라를 보호하는 역할을 합니다 AWS 클라우드. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 [데이터 프라이버시 FAQ](#)를 참조하십시오. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 [AWS 공동 책임 모델 및 GDPR](#) 블로그 게시물을 참조하십시오.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 AWS IAM Identity Center OR AWS Identity and Access Management (IAM) 을 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 멀티 팩터 인증 설정(MFA)을 사용하세요.
- SSL/TLS를 사용하여 리소스와 통신할 수 있습니다. AWS TLS 1.2는 필수이며 TLS 1.3를 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다. AWS CloudTrail
- 포함된 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용하십시오 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 AWS 통해 액세스할 때 FIPS 140-2로 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용하십시오. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 [연방 정보 처리 표준\(FIPS\) 140-2](#)를 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 콘솔, API 또는 SDK를 AWS 서비스 사용하여 AWS 결제 암호화 또는 기타 작업을 하는 경우가 포함됩니다. AWS CLI AWS 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

AWS Payment Cryptography는 Payment Cryptography 키를 저장하고 보호하여 높은 가용성을 제공하는 동시에 강력하고 유연한 액세스 제어를 제공합니다.

주제

- [키 구성 요소 보호](#)
- [데이터 암호화](#)
- [저장 중 암호화](#)
- [전송 중 암호화](#)
- [인터넷워크 트래픽 개인 정보](#)

키 구성 요소 보호

기본적으로 AWS Payment Cryptography는 서비스에서 관리하는 결제 키의 암호화 키 자료를 보호합니다. 또한, AWS Payment Cryptography는 서비스 외부에서 생성된 주요 자료를 가져오는 옵션을 제공합니다. 결제 키 및 키 구성 요소에 대한 기술 세부 정보는 AWS Payment Cryptography 세부 정보를 참조하세요.

데이터 암호화

AWS Payment Cryptography의 데이터는 AWS Payment Cryptography 키, 해당 키가 나타내는 암호화 키 구성 요소, 사용 특성으로 구성됩니다. 이 키 구성 요소는 AWS Payment Cryptography 하드웨어 보안 모듈(HSM) 내에서만 일반 텍스트로 존재하며 사용 중일 때만 존재합니다. 그렇지 않으면 키 구성 요소와 특성이 암호화되어 내구성이 뛰어난 영구 스토리지에 저장됩니다.

AWS Payment Cryptography가 결제 키에 대해 생성하거나 로드하는 키 구성 요소는 암호화되지 않은 상태로 두지 않습니다. AWS Payment Cryptography API 작업을 통해 암호화된 상태로 내보낼 수 있습니다.

저장 중 암호화

AWS Payment Cryptography는 PCI PTS HSM 등재 HSM에서 결제 키용 키 구성 요소를 생성합니다. 사용하지 않을 경우 키 구성 요소는 HSM 키에 의해 암호화되어 내구성이 뛰어난 영구 스토리지에 기록됩니다. Payment Cryptography 키의 키 구성 요소와 키 구성 요소를 보호하는 암호화 키는 HSM을 일반 텍스트 형식으로 남기지 않습니다.

Payment Cryptography 키에 대한 키 구성 요소의 암호화 및 관리는 전적으로 서비스에서 처리됩니다.

자세한 내용은 AWS 키 관리 서비스 암호화 세부 정보를 참조하세요.

전송 중 암호화

AWS Payment Cryptography가 결제 키용으로 생성하거나 로드하는 키 구성 요소는 AWS Payment Cryptography API 작업을 통해 일반 텍스트로 내보내거나 전송하지 않습니다. AWS Payment Cryptography는 키 식별자를 사용하여 API 작업의 키를 나타냅니다.

하지만 일부 AWS Payment Cryptography API 작업은 이전에 공유한 키 교환 키 또는 비대칭 키 교환 키로 암호화된 키를 내보냅니다. 또한 고객은 API 작업을 사용하여 결제 키에 대한 암호화된 키 구성 요소를 가져올 수 있습니다.

모든 AWS Payment Cryptography API 직접 호출은 서명되어야 하며 전송 계층 보안(TLS)을 사용하여 전송되어야 합니다. AWS Payment Cryptography에는 PCI에서 '강력한 암호화'로 정의한 TLS 버전 및 암호 제품군이 필요합니다. 모든 서비스 엔드포인트는 TLS 1.0~1.3 및 하이브리드 포스트 쿼텀 TLS를 지원합니다.

자세한 내용은 AWS 키 관리 서비스 암호화 세부 정보를 참조하세요.

인터넷워크 트래픽 개인 정보

AWS Payment Cryptography는 AWS Management Console과 결제 키를 생성 및 관리하고 이를 암호화 작업에 사용할 수 있는 일련의 API 작업을 지원합니다.

AWS 결제 암호화는 비공개 네트워크에서 AWS로의 두 가지 네트워크 연결 옵션을 지원합니다.

- 인터넷을 통한 IPSec VPN 연결.
- AWS Direct Connect는 표준 이더넷 광섬유 케이블을 통해 내부 네트워크를 AWS Direct Connect 위치에 연결합니다.

모든 Payment Cryptography API 직접 호출은 서명되어야 하며 전송 계층 보안(TLS)을 사용하여 전송되어야 합니다. 호출에는 또한 완벽한 순방향 보안을 지원하는 최신 암호 제품군도 필요합니다. 결제 키에 대한 키 구성 요소를 저장하는 하드웨어 보안 모듈(HSM)에 대한 트래픽은 AWS 내부 네트워크를 통해 알려진 AWS Payment Cryptography API 호스트에서만 허용됩니다.

퍼블릭 인터넷을 통해 트래픽을 전송하지 않고 가상 사설 클라우드 (VPC) 에서 AWS Payment Cryptography에 직접 연결하려면 AWS에서 제공하는 VPC 엔드포인트를 사용하십시오. PrivateLink 자세한 내용은 VPC 엔드포인트를 통한 AWS Payment Cryptography 연결을 참조하세요.

AWS Payment Cryptography에서는 전송 계층 보안(TLS) 네트워크 암호화 프로토콜에 대한 하이브리드 포스트 쿼텀 키 교환 옵션을 지원합니다. AWS Payment Cryptography API 엔드포인트에 연결할 때 TLS와 함께 이 옵션을 사용할 수 있습니다.

결제 암호화의 탄력성 AWS

AWS 글로벌 인프라는 지역 및 가용 AWS 영역을 중심으로 구축됩니다. 리전은 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며, 이러한 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크를 통해 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 지역 및 가용 영역에 대한 자세한 내용은 [AWS 글로벌 인프라](#)를 참조하십시오.

리전별 격리

AWS 결제 암호화는 여러 지역에서 사용할 수 있는 지역 서비스입니다.

지역별로 격리된 AWS Payment Cryptography 설계는 한 AWS 리전의 가용성 문제가 다른 지역의 AWS Payment Cryptography 운영에 영향을 미치지 않도록 보장합니다. AWS Payment Cryptography는 모든 소프트웨어 업데이트 및 조정 작업이 원활하고 눈에 띄지 않게 수행되므로 계획된 가동 중지 시간이 전혀 발생하지 않도록 설계되었습니다.

AWS Payment Cryptography 서비스 수준에 관한 계약(SLA)에는 모든 Payment Cryptography API에 대한 99.99%의 서비스 보장이 포함됩니다. 이 보장을 이행하기 위해 AWS Payment Cryptography는 API 요청을 실행하는 데 필요한 모든 데이터 및 권한 부여 정보가 요청을 수신하는 모든 리전 호스트에서 사용할 수 있도록 합니다.

이 AWS Payment Cryptography 인프라는 각 리전에서 최소 3개의 가용 영역(AZ)에 복제됩니다. 여러 호스트 장애가 AWS Payment Cryptography 성능에 영향을 미치지 않도록 하기 위해 한 리전의 모든 AZ에서 고객 트래픽을 처리하도록 AWS Payment Cryptography가 설계되었습니다.

결제 키의 속성 또는 권한에 대한 변경 사항은 리전의 모든 호스트에서 후속 요청을 올바르게 처리할 수 있도록 리전의 모든 호스트에 복제됩니다. 결제 키를 사용한 암호화 작업에 대한 요청은 결제 키로 작업을 수행할 수 있는 AWS Payment Cryptography 하드웨어 보안 모듈(HSM)의 플릿에 전달됩니다.

멀티 테넌트 디자인

AWS Payment Cryptography의 다중 테넌트 설계를 통해 가용성 SLA를 충족하고 높은 요청률을 유지하면서 키 및 데이터의 기밀성을 보호할 수 있습니다.

암호화 작업에 지정한 결제 키가 항상 사용되는 키인지 확인하기 위해 여러 무결성 보장 메커니즘이 배포됩니다.

Payment Cryptography 키의 일반 텍스트 키 구성 요소는 광범위하게 보호됩니다. 키 구성 요소는 생성되는 즉시 HSM에서 암호화되며 암호화된 키 구성 요소는 즉시 안전한 스토리지로 옮겨집니다. 암호화된 키는 사용 시 HSM 내에서 검색 및 해독됩니다. 일반 텍스트 키는 암호화 작업을 완료하는 데 필요한 시간 동안만 HSM 메모리에 남아 있습니다. 일반 텍스트 키 구성 요소는 HSM을 절대 떠나지 않으며 영구 스토리지에 절대 기록되지 않습니다.

AWS Payment Cryptography에서 키를 보호하기 위해 사용하는 메커니즘에 대한 자세한 내용은 AWS Payment Cryptography 세부 정보를 참조하세요.

의 인프라 보안 AWS Payment Cryptography

관리형 서비스로서 [Amazon Web Services: 보안 프로세스 개요](#) 백서에 설명된 [AWS 글로벌 네트워크 보안 절차에 따라](#) 보호됩니다. AWS Payment Cryptography

AWS 게시된 API 호출을 사용하여 네트워크를 AWS Payment Cryptography 통해 액세스합니다. 클라이언트가 전송 계층 보안(TLS) 1.2 이상을 지원해야 합니다. 클라이언트는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)과 같은 PFS(전달 완전 보안, Perfect Forward Secrecy)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 보안 액세스 키를 사용하여 서명해야 합니다. 또는 [AWS Security Token Service\(AWS STS\)](#)를 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

물리적 호스트의 격리

AWS Payment Cryptography가 사용하는 물리적 인프라의 보안은 Amazon Web Services: 보안 프로세스 개요의 물리적 및 환경적 보안 섹션에 설명된 제어의 적용을 받습니다. 이전 섹션에 나열된 규정 준수 보고서와 타사 감사 결과에 대한 자세한 내용을 확인할 수 있습니다.

AWS 결제 암호화는 전용 commercial-off-the-shelf PCI PTS HSM 등재 하드웨어 보안 모듈 (HSM) 에서 지원됩니다. AWS Payment Cryptography 키에 대한 키 구성 요소는 HSM의 휘발성 메모리에만 저장되며 Payment Cryptography 키가 사용되는 동안에만 저장됩니다. HSM은 모든 물리적 액세스에 대해 이중 제어를 적용하는 Amazon 데이터 센터 내의 액세스 제어 랙에 있습니다. AWS Payment Cryptography HSM의 작동에 대한 자세한 내용은 AWS Payment Cryptography 세부 정보를 참조하세요.

결제 암호화에 대한 보안 모범 사례 AWS

AWS 결제 암호화는 IAM 정책, 키 정책 및 [IAM 정책을](#) 구체화하기 위한 광범위한 정책 조건 키 세트, 키 블록과 관련된 내장된 PCI PIN 규칙 적용 등 암호화 키의 보호를 강화하고 암호화 키가 의도한 용도에 맞게 사용되도록 하기 위해 선택적으로 구현할 수 있는 다양한 보안 기능을 지원합니다.

Important

제공된 일반적인 지침은 완전한 보안 솔루션을 나타내지는 않습니다. 모든 상황에 적합한 것은 아니기 때문에 이들이 규범적인 것은 아닙니다.

- 키 사용 및 사용 방식: AWS 결제 암호화는 ANSI X9 TR 31-2018 상호 운용 가능한 보안 키 교환 키 블록 사양에 설명되어 있고 PCI PIN 보안 요구 사항 18-3에 따라 키 사용 및 사용 모드 제한을 따르고 적용합니다. 이로 인해 단일 키를 여러 용도로 사용하는 것이 제한되고 키 메타데이터(예: 허용된 작업)가 키 자료 자체에 암호적으로 바인딩됩니다. AWS 결제 암호화는 키 암호화 키(TR31_K0_KEY_ENCRYPTION_KEY)를 데이터 복호화에도 사용할 수 없도록 하는 등의 제한을 자동으로 적용합니다. 자세한 내용은 [AWS 결제 암호화 키의 주요 속성 이해](#) 단원을 참조하세요.
- 대칭 키 구성 요소의 공유 제한: 대칭 키 구성 요소(예: 핀 암호화 키 또는 키 암호화 키)를 최대 한 개체와 공유하세요. 민감한 자료를 더 많은 법인 또는 파트너에게 전송해야 하는 경우 추가 키를 생성하세요. AWS 결제 암호화는 절대 대칭 키 자료나 비대칭 개인 키 자료를 절대 노출하지 않습니다.
- 별칭이나 태그를 사용하여 키를 특정 사용 사례 또는 파트너와 연결: 별칭을 사용하여 키와 연결된 사용 사례를 쉽게 나타낼 수 있습니다(예: BIN 12345와 연결된 카드 인증 키를 나타내는 alias/BIN_12345_CVK). 유연성을 높이려면 bin=12345, use_case=acquiring, country=us, partner=foo와 같은 태그를 만들어 보세요. 별칭과 태그를 사용하여 사용 사례 발행과 획득 간의 액세스 제어를 적용하는 등 액세스를 제한할 수도 있습니다.
- 최소 권한 액세스 실행: IAM을 사용하여 개별 사용자가 키를 생성하거나 암호화 작업을 실행하는 것을 금지하는 등 개인이 아닌 시스템에 대한 프로덕션 액세스를 제한할 수 있습니다. IAM을 사용하여 취득자의 핀 생성 또는 검증 기능을 제한하는 등 사용 사례에 적합하지 않을 수 있는 명령과 키 모두에 대한 액세스를 제한할 수도 있습니다. 최소 권한 액세스를 사용하는 또 다른 방법은 민감한 작업(예: 키 가져오기)을 특정 서비스 계정으로 제한하는 것입니다. [AWS 결제 암호화 ID 기반 정책 예제](#)의 예제를 참조하세요.

참고 항목

- [AWS 결제 암호화를 위한 ID 및 액세스 관리](#)
- IAM 사용 설명서의 [IAM 보안 모범 사례](#)

AWS 결제 암호화에 대한 규정 준수 검증

타사 감사자는 여러 AWS 규정 준수 프로그램의 일환으로 AWS 결제 암호화의 보안 및 규정 준수를 평가합니다. 여기에는 SOC, PCI 등이 포함됩니다.

AWS 결제 암호화는 PCI DSS 외에도 여러 PCI 표준에 대해 평가되었습니다. 여기에는 PCI PIN 보안 (PCI PIN) 및 PCI 포인트 투 포인트(P2PE) 암호화가 포함됩니다. 사용 가능한 증명 및 규정 준수 가이드는 AWS Artifact를 참조하세요.

특정 규정 준수 프로그램의 범위 내에 있는 AWS 서비스 목록은 [규정 준수 프로그램 제공 범위 내 AWS 서비스](#)를 참조하세요. 일반적인 정보는 [AWS 규정 준수 프로그램](#)을 참조하세요.

AWS Artifact를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 [AWS Artifact에서 보고서 다운로드](#)를 참조하세요.

AWS 결제 암호화 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다. AWS에서는 규정 준수를 지원할 다음과 같은 리소스를 제공합니다.

- [보안 및 규정 준수 빠른 시작 안내서](#)—이 배포 안내서에서는 아키텍처 고려 사항에 관해 설명하고 AWS에서 보안 및 규정 준수에 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- [AWS 규정 준수 리소스](#)—고객 조직이 속한 산업 및 위치에 적용될 수 있는 워크북 및 가이드 컬렉션입니다.
- AWS Config 개발자 가이드의 [규칙을 사용하여 리소스 평가](#)—AWS Config, 리소스 구성이 내부 사례, 업계 지침, 규정을 얼마나 잘 준수하는지 평가합니다.
- [AWS Security Hub](#)—이 AWS 서비스는 보안 업계 표준 및 모범 사례 규정 준수 여부를 확인하는 데 도움이 되는 AWS 내 보안 상태에 대한 포괄적인 관점을 제공합니다.

AWS 결제 암호화를 위한 ID 및 액세스 관리

AWS Identity and Access Management (IAM)은 관리자가 리소스에 대한 액세스를 안전하게 제어할 수 있는 AWS 서비스 있도록 도와줍니다. AWS IAM 관리자는 AWS 결제 암호화 리소스를 사용할 수 있는 인증 (로그인) 및 권한 부여 (권한 부여) 를 받을 수 있는 사용자를 관리합니다. IAM은 추가 비용 AWS 서비스 없이 사용할 수 있습니다.

주제

- [고객](#)
- [자격 증명을 통한 인증](#)
- [정책을 사용한 액세스 관리](#)
- [AWS 결제 암호화가 IAM과 작동하는 방식](#)
- [AWS 결제 암호화 ID 기반 정책 예제](#)
- [AWS 결제 암호화 ID 및 액세스 문제 해결](#)

고객

결제 암호화에서 AWS 수행하는 작업에 따라 사용 방법 AWS Identity and Access Management (IAM) 이 다릅니다.

서비스 사용자 — AWS 결제 암호화 서비스를 사용하여 업무를 수행하는 경우 관리자가 필요한 자격 증명과 권한을 제공합니다. 더 많은 AWS 결제 암호화 기능을 사용하여 업무를 수행함에 따라 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. AWS Payment Cryptography의 기능에 액세스할 수 없는 경우 [AWS 결제 암호화 ID 및 액세스 문제 해결](#)을 참조하세요.

서비스 관리자 — 회사에서 AWS 결제 암호화 리소스를 담당하는 경우 결제 암호화에 대한 전체 액세스 권한이 있을 수 있습니다. AWS 서비스 사용자가 액세스해야 하는 AWS 결제 암호화 기능 및 리소스를 결정하는 것은 여러분의 몫입니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해합니다. 회사에서 IAM을 AWS 결제 암호화와 함께 사용하는 방법에 대한 자세한 내용은 [AWS 결제 암호화가 IAM과 작동하는 방식](#)을 참조하십시오.

IAM 관리자 — IAM 관리자라면 결제 암호화에 대한 액세스를 관리하기 위한 정책을 작성하는 방법에 대해 자세히 알고 싶을 것입니다. AWS IAM에서 사용할 수 있는 AWS 결제 암호화 ID 기반 정책의 예를 보려면 [AWS 결제 암호화 ID 기반 정책 예제](#)을 참조하십시오.

자격 증명을 통한 인증

인증은 ID 자격 증명을 AWS 사용하여 로그인하는 방법입니다. IAM 사용자로 인증 (로그인 AWS) 하거나 IAM 역할을 맡아 인증 (로그인) 해야 합니다. AWS 계정 루트 사용자

ID 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 ID로 로그인할 수 있습니다. AWS IAM Identity Center (IAM ID 센터) 사용자, 회사의 싱글 사인온 인증, Google 또는 Facebook 자격 증명이 페더레이션 ID의 예입니다. 페더레이션 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 액세스하는 경우 AWS 간접적으로 역할을 맡게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 로그인에 대한 자세한 내용은 AWS 로그인 사용 설명서의 [내 로그인 방법을](#) 참조하십시오. AWS 계정

AWS 프로그래밍 방식으로 액세스하는 경우 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트 (SDK)와 명령줄 인터페이스 (CLI)를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 직접 요청에 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS [API 요청 서명을](#) 참조하십시오.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS 계정의 보안을 강화하기 위해 다단계 인증 (MFA)을 사용할 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 [다중 인증](#) 및 IAM 사용 설명서의 [AWS에서 다중 인증\(MFA\) 사용](#)을 참조하세요.

AWS 계정 루트 사용자

계정을 AWS 계정만들 때는 먼저 계정의 모든 AWS 서비스 리소스에 대한 완전한 액세스 권한을 가진 하나의 로그인 ID로 시작합니다. 이 ID를 AWS 계정 루트 사용자라고 하며, 계정을 만들 때 사용한 이메일 주소와 비밀번호로 로그인하여 액세스할 수 있습니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 [루트 사용자 보안 인증이 필요한 태스크](#)를 참조하세요.

IAM 사용자 및 그룹

[IAM 사용자는 단일 사용자](#) 또는 애플리케이션에 대한 특정 권한을 가진 사용자 내의 자격 증명입니다. AWS 계정 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명에 필요한 특정 사용 사례가 있는 경우 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용자 설명서의 [장기 보안 인증이 필요한 사용 사례의 경우 정기적으로 액세스 키 교체](#)를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어 IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수입할 수 있습니다. 사용자는 영구적인 보안 인증 정보를 가지고 있지만, 역할은 임시 보안 인증 정보만 제공합니다. 자세한 정보는 IAM 사용 설명서의 [IAM 사용자\(역할을 대신하여\)를 만들어야 하는 경우](#)를 참조하세요.

IAM 역할

IAM 역할은 특정 권한을 가진 사용자 AWS 계정 내의 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할을 AWS Management Console [전환하여](#) 에서 일시적으로 IAM 역할을 맡을 수 있습니다. AWS CLI 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 사용하여 역할을 수입할 수 있습니다. 역할 사용 방법에 대한 자세한 정보는 IAM 사용자 설명서의 [IAM 역할 사용](#)을 참조하세요.

임시 보안 인증 정보가 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 연동 사용자 액세스 - 연동 자격 증명에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 연동 자격 증명이 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 [서드 파티 자격 증명 공급자의 역할 만들기](#) 부분을 참조하세요. IAM Identity Center를 사용하는 경우 권한 세트를 구성합니다. 인증 후 자격 증명이 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연관짓습니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 [권한 세트](#)를 참조하세요.
- 임시 IAM 사용자 권한: IAM 사용자 또는 역할은 IAM 역할을 수입하여 특정 태스크에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 크로스 계정 액세스: IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스 경우에는 역할을 프록시로 사용하는 대신 정책을 리소스에 직접 연결할 수 있습니다. 크로스 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 [IAM 역할과 리소스 기반 정책의 차이](#)를 참조하세요.
- 서비스 간 액세스 — 일부는 다른 AWS 서비스서비스의 기능을 AWS 서비스 사용합니다. 예컨대, 어떤 서비스에서 호출을 수행하면 일반적으로 해당 서비스는 EC2에서 애플리케이션을 실행하거나 S3에 객체를 저장합니다. 서비스는 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.

- 순방향 액세스 세션 (FAS) — IAM 사용자 또는 역할을 사용하여 작업을 수행하는 경우 보안 AWS 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 전화를 거는 주체의 권한을 다운스트림 AWS 서비스서비스에 AWS 서비스 요청하기 위한 요청과 결합하여 사용합니다. FAS 요청은 다른 서비스 AWS 서비스 또는 리소스와 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 [전달 액세스 세션](#)을 참조하세요.
- 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 맡는 [IAM 역할](#)입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 [AWS 서비스에 대한 권한을 위임할 역할 생성](#)을 참조하세요.
- 서비스 연결 역할 — 서비스 연결 역할은 연결된 서비스 역할의 한 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수입할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.
- Amazon EC2에서 실행되는 애플리케이션 — IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 API 요청을 AWS CLI 하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. AWS 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있게 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 [IAM 역할을 사용하여 EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여](#)를 참조하세요.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 [IAM 사용자 설명서](#)의 IAM 역할(사용자 대신)을 생성하는 경우를 참조하세요.

정책을 사용한 액세스 관리

정책을 생성하고 이를 AWS ID 또는 리소스에 AWS 연결하여 액세스를 제어할 수 있습니다. 정책은 ID 또는 리소스와 연결될 때 AWS 해당 권한을 정의하는 객체입니다. AWS 주도자 (사용자, 루트 사용자 또는 역할 세션) 가 요청할 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 JSON 문서로 AWS 저장됩니다. JSON 정책 문서의 구조와 콘텐트에 대한 자세한 정보는 IAM 사용자 설명서의 [JSON 정책 개요](#)를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는 지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수입할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, iam:GetRole태스크를 허용하는 정책이 있다고 가정합니다. 해당 정책을 사용하는 사용자는 AWS Management Console, AWS CLI, 또는 AWS API에서 역할 정보를 가져올 수 있습니다.

ID 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 [IAM 정책 생성](#) 단원을 참조하세요.

ID 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 내 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립형 정책입니다. AWS 계정관리형 정책에는 AWS 관리형 정책과 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용자 설명서의 [관리형 정책과 인라인 정책 사이의 선택](#)을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 [보안 주체를 지정](#)해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 등이 포함될 수 있습니다. AWS 서비스

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. IAM의 AWS 관리형 정책은 리소스 기반 정책에 사용할 수 없습니다.

액세스 제어 목록(ACLs)

액세스 제어 목록(ACLs)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

ACL을 지원하는 서비스의 예로는 아마존 S3와 아마존 VPC가 있습니다. AWS WAF ACL에 대해 자세히 알아보려면 Simple Storage Service 개발자 안내서의 [ACL\(액세스 제어 목록\) 개요](#)를 참조하세요.

기타 정책 타입

AWS 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 타입에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- **권한 경계:** 권한 경계는 ID 기반 정책에 따라 IAM 엔티티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 엔티티의 자격 증명 기반 정책과 그 권한 경계의 교집합입니다. Principal 필드에서 사용자나 역할을 보안 주체로 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용자 설명서의 [IAM 엔티티에 대한 권한 경계](#)를 참조하세요.
- **서비스 제어 정책 (SCP)** - SCP는 조직 또는 조직 단위 (OU)에 대한 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations AWS Organizations 사업체가 소유한 여러 AWS 계정 개를 그룹화하고 중앙에서 관리하는 서비스입니다. 조직에서 모든 특성을 활성화할 경우, 서비스 제어 정책 (SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 구성원 계정의 엔티티 (각 엔티티 포함)에 대한 권한을 제한합니다. AWS 계정 루트 사용자조직 및 SCP에 대한 자세한 정보는 AWS Organizations 사용 설명서의 [SCP 작동 방식](#)을 참조하십시오.
- **세션 정책:** 세션 정책은 역할 또는 연합된 사용자에게 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할 자격 증명 기반 정책의 교차 및 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용자 설명서의 [세션 정책](#)을 참조하세요.

여러 정책 타입

여러 정책 타입이 요청에 적용되는 경우 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련되어 있을 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 [정책 평가 로직](#)을 참조하십시오.

AWS 결제 암호화가 IAM과 작동하는 방식

IAM을 사용하여 결제 암호화에 대한 액세스를 관리하려면 먼저 AWS 결제 암호화와 함께 사용할 수 있는 IAM 기능을 이해해야 합니다. AWS AWS 결제 암호화 및 기타 AWS 서비스가 IAM과 어떻게 작동하는지 자세히 알아보려면 IAM 사용 설명서의 IAM과 [연동되는AWS 서비스](#)를 참조하십시오.

주제

- [AWS 결제 암호화 ID 기반 정책](#)
- [AWS Payment Cryptography 태그 기반 인증](#)

AWS 결제 암호화 ID 기반 정책

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. AWS 결제 암호화는 특정 작업, 리소스 및 조건 키를 지원합니다. JSON 정책에서 사용하는 모든 요소에 대해 알고 싶다면 IAM 사용 설명서의 [IAM JSON 정책 요소 참조](#)를 참조하세요.

작업

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는 지를 지정할 수 있습니다.

JSON 정책의 Action 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 태스크를 설명합니다. 정책 작업은 일반적으로 관련 AWS API 작업과 이름이 같습니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

AWS 결제 암호화의 정책 조치는 조치 앞에 다음 접두사를 사용합니다. `payment-cryptography:` 예를 들어, 누군가에게 AWS 결제 암호화 `VerifyCardData` API 작업을 실행할 권한을 부여하려면 해당 작업을 해당 사용자의 정책에 포함해야 합니다 `payment-cryptography:VerifyCardData`. 정책 문에는 Action 또는 NotAction 요소가 포함되어야 합니다. AWS 결제 암호화는 이 서비스로 수행할 수 있는 작업을 설명하는 자체 조치 세트를 정의합니다.

명령문 하나에 여러 태스크를 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
  "payment-cryptography:action1",
  "payment-cryptography:action2"
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, List라는 단어(ListKeys 및 ListAliases 등)로 시작하는 모든 태스크를 지정하려면 다음 태스크를 포함합니다.

```
"Action": "payment-cryptography:List*"
```

AWS 결제 암호화 작업 목록을 보려면 IAM 사용 설명서의 [AWS 결제 암호화로 정의된 작업을](#) 참조하십시오.

리소스

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

Resource JSON 정책 요소는 작업이 적용되는 하나 이상의 개체를 지정합니다. 문장에는 Resource 또는 NotResource 요소가 반드시 추가되어야 합니다. 모범 사례에 따라 [리소스 이름 \(ARN\)](#)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이 작업을 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

결제 암호화 키 리소스에는 다음 ARN이 있습니다.

```
arn:${Partition}:payment-cryptography:${Region}:${Account}:key/${keyARN}
```

ARN 형식에 대한 자세한 내용은 [Amazon 리소스 이름 \(ARN\) 및 AWS 서비스 네임스페이스](#)를 참조하십시오.

예를 들어 문에서 arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qai11w2h 인스턴스를 지정하려면 다음 ARN을 사용합니다.

```
"Resource": "arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qai11w2h"
```

특정 계정에 속하는 모든 키를 지정하려면 와일드카드(*)를 사용합니다.

```
"Resource": "arn:aws:payment-cryptography:us-east-2:111122223333:key/*"
```


일부 AWS 결제 암호화 작업 (예: 키 생성 작업) 은 특정 리소스에서 수행할 수 없습니다. 이러한 경우, 와일드카드(*)를 사용해야 합니다.

```
"Resource": "*"
```

단일 명령문에서 여러 리소스를 지정하려면 아래와 같이 쉼표를 사용합니다.

```
"Resource": [
  "resource1",
  "resource2"
```

예제

AWS 결제 암호화 ID 기반 정책의 예를 보려면 을 참조하십시오. [AWS 결제 암호화 ID 기반 정책 예제](#)

AWS Payment Cryptography 태그 기반 인증

AWS 결제 암호화 ID 기반 정책 예제

기본적으로 IAM 사용자 및 역할은 AWS Payment Cryptography 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console AWS CLI, 또는 API를 사용하여 작업을 수행할 수 없습니다. AWS IAM 관리자는 지정된 리소스에서 특정 API 작업을 수행할 수 있는 권한을 사용자와 역할에게 부여하는 IAM 정책을 생성해야 합니다. 그런 다음 관리자는 해당 권한이 필요한 IAM 사용자 또는 그룹에 이러한 정책을 연결해야 합니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 [JSON 탭에서 정책 생성](#)을 참조하세요.

주제

- [정책 모범 사례](#)
- [AWS Payment Cryptography 콘솔 사용](#)
- [사용자가 자신이 권한을 볼 수 있도록 허용](#)
- [AWS 결제 암호화의 모든 측면에 액세스할 수 있습니다.](#)
- [지정된 키를 사용하여 API 호출 가능](#)
- [리소스를 구체적으로 거부할 수 있음](#)

정책 모범 사례

ID 기반 정책에 따라 다른 사람이 계정에서 AWS 결제 암호화 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책으로 시작하여 최소 권한 권한으로 이동 — 사용자와 워크로드에 권한을 부여하려면 여러 일반적인 사용 사례에 권한을 부여하는 AWS 관리형 정책을 사용하세요. 해당 내용은 에서 사용할 수 있습니다. AWS 계정사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 더 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 [AWS 관리형 정책](#) 또는 [AWS 직무에 대한 관리형 정책](#)을 참조하세요.
- 최소 권한 적용: IAM 정책을 사용하여 권한을 설정하는 경우 태스크를 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 [IAM의 정책 및 권한](#)을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한: 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어 SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. 예를 AWS 서비스들어 특정 작업을 통해 서비스 작업을 사용하는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 AWS CloudFormation있습니다. 자세한 정보는 IAM 사용자 설명서의 [IAM JSON 정책 요소: 조건](#)을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 검증하여 안전하고 기능적인 권한 보장: IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 신규 및 기존 정책을 검증합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 정보는 IAM 사용 설명서의 [IAM Access Analyzer 정책 검증](#)을 참조하세요.
- 멀티 팩터 인증 (MFA) 필요 - IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 AWS 계정 MFA를 활성화하십시오. API 작업을 직접 호출할 때 MFA가 필요하다면 정책에 MFA 조건을 추가합니다. 자세한 정보는 IAM 사용 설명서의 [MFA 보호 API 액세스 구성](#)을 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용자 설명서의 [IAM의 보안 모범 사례](#)를 참조하세요.

AWS Payment Cryptography 콘솔 사용

AWS 결제 암호화 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한을 통해 계정의 AWS 결제 암호화 리소스에 대한 세부 정보를 나열하고 볼 수 AWS 있어야 합니다. 최소 필수 권한보다

더 제한적인 보안 인증 기반 정책을 만들면 콘솔이 해당 정책에 연결된 개체(IAM 사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

해당 엔티티가 계속해서 AWS 결제 암호화 콘솔을 사용할 수 있도록 하려면 다음 AWS 관리형 정책도 엔티티에 연결하십시오. 자세한 내용은 IAM 사용 설명서의 [사용자에게 권한 추가](#)를 참조하십시오.

AWS CLI 또는 AWS API만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요는 없습니다. 그 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자가 자신이 권한을 볼 수 있도록 허용

이 예시는 IAM 사용자가 자신의 사용자 자격 증명에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로 이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsWithUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ],
    }
  ],
}
```

```

    "Resource": "*"
  }
]
}

```

AWS 결제 암호화의 모든 측면에 액세스할 수 있습니다.

⚠ Warning

이 예시는 광범위한 권한을 제공하므로 권장되지 않습니다. 대신 권한이 가장 낮은 액세스 모델을 고려해 보세요.

이 예시에서는 AWS 계정의 IAM 사용자에게 모든 결제 암호화 키에 대한 액세스 권한을 부여하고 두 작업을 모두 포함한 모든 AWS 결제 암호화 API를 호출할 수 있는 권한을 부여하려고 합니다.

ControlPlane DataPlane

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "payment-cryptography:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}

```

지정된 키를 사용하여 API 호출 가능

이 예시에서는 AWS 계정의 IAM 사용자에게 AWS 결제 암호화 키 중 하나에 대한 액세스 권한을 부여한 다음, 이 리소스를 두 API `arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaif11w2h` 및 `arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaif11w2h`에서 사용하려고 합니다.

`GenerateCardData` `VerifyCardData` 반대로 IAM 사용자는 `DeleteKey` 또는 `ExportKey`와 같은 다른 작업에서 이 키를 사용할 수 없습니다.

리소스는 key가 앞에 붙은 키 또는 alias가 앞에 붙은 별칭일 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "payment-cryptography:VerifyCardData",
        "payment-cryptography:GenerateCardData"
      ],
      "Resource": [
        "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaiif1lw2h"
      ]
    }
  ]
}
```

리소스를 구체적으로 거부할 수 있음

Warning

와일드카드 액세스 권한 부여의 영향을 신중하게 고려하세요. 대신 최소 권한 모델을 고려해 보세요.

이 예시에서는 AWS 계정의 IAM 사용자가 AWS 결제 암호화 키에 액세스할 수 있도록 허용하되 특정 키 하나에 대한 권한은 거부하려고 합니다. 사용자는 거부 문에 지정된 키를 제외한 모든 키로 VerifyCardData 및 GenerateCardData에 액세스할 수 있습니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "payment-cryptography:VerifyCardData",
        "payment-cryptography:GenerateCardData"
      ]
    }
  ]
}
```

```
    ],
    "Resource": [
      "arn:aws:payment-cryptography:us-east-2:111122223333:key/*"
    ]
  },
  {
    "Effect": "Deny",
    "Action": [
      "payment-cryptography:GenerateCardData"
    ],
    "Resource": [
      "arn:aws:payment-cryptography:us-east-2:111122223333:key/
arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaiif1lw2h"
    ]
  }
]
```

AWS 결제 암호화 ID 및 액세스 문제 해결

AWS 결제 암호화와 관련된 IAM 관련 문제가 확인되면 이 섹션에 주제가 추가될 예정입니다. IAM 주제에 대한 일반적인 문제 해결 내용은 IAM 사용 설명서의 [문제 해결 섹션](#)을 참조하세요.

AWS 결제 암호화 모니터링

모니터링은 AWS 결제 암호화 및 기타 AWS 솔루션의 신뢰성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. AWS는 AWS 결제 암호화를 관찰하고, 문제 발생 시 보고하고, 적절한 경우 자동 조치를 취하는 다음과 같은 모니터링 도구를 제공합니다.

- Amazon CloudWatch는 AWS에서 실행하는 AWS 리소스와 애플리케이션을 실시간으로 모니터링합니다. 지표를 수집 및 추적하고, 사용자 지정 대시보드를 생성할 수 있으며, 지정된 지표가 지정된 임계값에 도달하면 사용자에게 알리거나 조치를 취하도록 경보를 설정할 수 있습니다. 예를 들어 CloudWatch에서 Amazon EC2 인스턴스의 CPU 사용량 또는 기타 지표를 추적하고 필요할 때 자동으로 새 인스턴스를 시작할 수 있습니다. 자세한 내용은 [Amazon CloudWatch 사용 설명서](#)를 참조하세요.
- Amazon CloudWatch Logs로 Amazon EC2 인스턴스, CloudTrail, 기타 소스의 로그 파일을 모니터링, 저장 및 액세스할 수 있습니다. CloudWatch Logs는 로그 파일의 정보를 모니터링하고 특정 임계값에 도달하면 사용자에게 알릴 수 있습니다. 또한 매우 내구력 있는 스토리지에 로그 데이터를 저장할 수 있습니다. 자세한 내용은 [Amazon CloudWatch Logs 사용 설명서](#)를 참조하세요.
- Amazon EventBridge를 사용하면 AWS 서비스를 자동화하고 애플리케이션 가용성 문제나 리소스 변경 같은 시스템 이벤트에 자동으로 대응할 수 있습니다. AWS 서비스의 이벤트는 거의 실시간으로 EventBridge로 전송됩니다. 원하는 이벤트만 표시하도록 간단한 규칙을 작성한 후 규칙과 일치하는 이벤트 발생 시 실행할 자동화 작업을 지정할 수 있습니다. 자세한 내용은 [Amazon EventBridge 사용 설명서](#)를 참조하세요.
- AWS CloudTrail은 직접 수행하거나 AWS 계정을 대신하여 수행한 API 호출 및 관련 이벤트를 캡처하고 지정한 Amazon S3 버킷에 로그 파일을 전송합니다. 어떤 사용자 및 계정이 AWS를 호출했는지, 어떤 소스 IP 주소에 호출이 이루어졌는지, 언제 호출이 발생했는지 확인할 수 있습니다. 자세한 내용은 [AWS CloudTrail 사용 설명서](#)를 참조하세요.

Note

AWS CloudTrail 로그는 CreateKey와 같은 컨트롤 플레인 작업에서는 지원되지만 카드 데이터 생성과 같은 데이터 영역 작업에서는 지원되지 않습니다.

AWS CloudTrail를 사용하여 AWS 결제 암호화 API 직접 호출 로깅

AWS 결제 암호화는 사용자, 역할 또는 AWS 결제 암호화의 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 AWS 결제 암호화에 대한 모든 API

직접 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 AWS 결제 암호화 콘솔의 호출과 AWS 결제 암호화 API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 AWS 결제 암호화 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 AWS 결제 암호화에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 [AWS CloudTrail 사용 설명서](#)를 참조하세요.

Note

Cloudtrail 통합은 현재 컨트롤 플레인 운영에만 지원됩니다.

CloudTrail의 AWS 결제 암호화 정보

CloudTrail은 계정 생성 시 AWS 계정에서 사용되도록 설정됩니다. AWS 결제 암호화에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 내용은 [CloudTrail 이벤트 기록을 사용하여 이벤트 보기](#)를 참조하세요.

또는 AWS 결제 암호화에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 리전에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 리전의 이벤트를 로깅하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 또는 CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- [추적 생성 개요](#)
- [CloudTrail 지원 서비스 및 통합](#)
- [CloudTrail에 대한 Amazon SNS 알림 구성](#)
- [여러 리전에서 CloudTrail 로그 파일 수신](#)
- [여러 계정에서 CloudTrail 로그 파일 수신](#)

CloudTrail은 [CreateKey](#), [ImportKey](#), [DeleteKey](#), [ListKeys](#), [TagResource](#) 및 기타 모든 컨트롤 플레인 작업과 같은 AWS 결제 암호화 작업을 기록합니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 자격 증명 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 AWS Identity and Access Management(IAM) 사용자 자격 증명으로 했는지.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

자세한 내용은 [CloudTrail userIdentity 요소](#)를 참조하세요.

AWS 결제 암호화 로그 파일 항목 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다.

CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함합니다. CloudTrail 로그 파일은 퍼블릭 API 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음은 AWS 결제 암호화 CreateKey 작업을 보여주는 CloudTrail 로그 항목이 나타낸 예제입니다.

```
{
  CloudTrailEvent: {
    tlsDetails= {
      TlsDetails: {
        cipherSuite=TLS_AES_128_GCM_SHA256,
        tlsVersion=TLSv1.3,
        clientProvidedHostHeader=pdx80.controlplane.paymentcryptography.us-
west-2.amazonaws.com
      }
    },
    requestParameters=CreateKeyInput (
      keyAttributes=KeyAttributes(
        KeyUsage=TR31_B0_BASE_DERIVATION_KEY,
        keyClass=SYMMETRIC_KEY,
        keyAlgorithm=AES_128,
        keyModesOfUse=KeyModesOfUse(
          encrypt=false,
          decrypt=false,
          wrap=false
          unwrap=false,
          generate=false,
```

```
        sign=false,
        verify=false,
        deriveKey=true,
        noRestrictions=false)
    ),
    keyCheckValueAlgorithm=null,
    exportable=true,
    enabled=true,
    tags=null),
  eventName=CreateKey,
  userAgent=Coral/Apache-HttpClient5,
  responseElements=CreateKeyOutput(
    key=Key(
      keyArn=arn:aws:payment-cryptography:us-
east-2:111122223333:key/5rplquuwozodpwsp,
      keyAttributes=KeyAttributes(
        KeyUsage=TR31_B0_BASE_DERIVATION_KEY,
        keyClass=SYMMETRIC_KEY,
        keyAlgorithm=AES_128,
        keyModesOfUse=KeyModesOfUse(
          encrypt=false,
          decrypt=false,
          wrap=false,
          unwrap=false,
          generate=false,
          sign=false,
          verify=false,
          deriveKey=true,
          noRestrictions=false)
        ),
      keyCheckValue=FE23D3,
      keyCheckValueAlgorithm=ANSI_X9_24,
      enabled=true,
      exportable=true,
      keyState=CREATE_COMPLETE,
      keyOrigin=AWS_PAYMENT_CRYPTOGRAPHY,
      createTimestamp=Sun May 21 18:58:32 UTC 2023,
      usageStartTimestamp=Sun May 21 18:58:32 UTC 2023,
      usageStopTimestamp=null,
      deletePendingTimestamp=null,
      deleteTimestamp=null)
    ),
    sourceIPAddress=192.158.1.38,
    userIdentity={
```

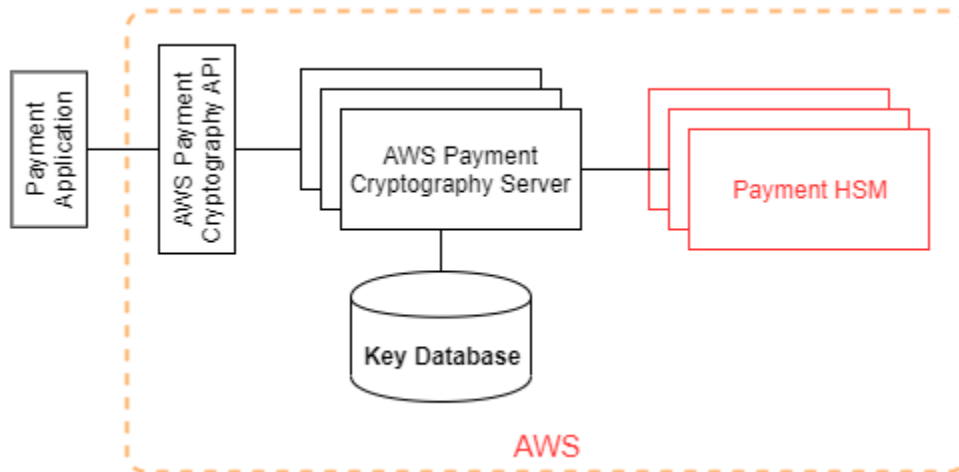
```
    UserIdentity: {
      arn=arn:aws:sts::111122223333:assumed-role/TestAssumeRole-us-west-2-PDX80/
ControlPlane-IntegTest-68211a2a-3e9d-42b7-86ac-c682520e0410,
      invokedBy=null,
      accessKeyId=,
      type=AssumedRole,
      sessionContext={
        SessionContext: {
          sessionIssuer={
            SessionIssuer: {arn=arn:aws:iam::111122223333:role/TestAssumeRole-us-
west-2-PDX80,
              type=Role,
              accountId=111122223333,
              userName=TestAssumeRole-us-west-2-PDX80,
              principalId=}
          },
          attributes={
            SessionContextAttributes: {
              creationDate=Sun May 21 18:58:31 UTC 2023,
              mfaAuthenticated=false
            }
          },
          webIdFederationData=null
        }
      },
      username=null,
      principalId=:ControlPlane-User,
      accountId=111122223333,
      identityProvider=null
    }
  },
  eventTime=Sun May 21 18:58:32 UTC 2023,
  managementEvent=true,
  recipientAccountId=111122223333,
  awsRegion=us-west-2,
  requestID=151cdd67-4321-1234-9999-dce10d45c92e,
  eventVersion=1.08, eventType=AwsApiCall,
  readOnly=false,
  eventID=c69e3101-eac2-1b4d-b942-019919ad2faf,
  eventSource=payment-cryptography.amazonaws.com,
  eventCategory=Management,
  additionalEventData={
}
}
```

```
}
```

암호화 세부 정보

AWS 결제 암호화는 결제 트랜잭션을 위한 암호화 키를 생성하고 관리할 수 있는 웹 인터페이스를 제공합니다. AWS 결제 암호화는 중앙 집중형 관리 및 감사에 사용할 수 있는 표준 키 관리 서비스와 결제 트랜잭션 암호화 및 도구를 제공합니다. 이 설명서에서는 이 서비스가 제공하는 기능을 평가하는 데 도움이 되도록 AWS 결제 암호화에서 사용할 수 있는 암호화 작업에 대한 자세한 설명을 제공합니다.

AWS 결제 암호화에는 분산된 [PCI PTS HSM 검증 하드웨어 보안 모듈](#) 플릿의 암호화 작업을 요청하기 위한 여러 인터페이스(AWS CLI, AWS SDK, AWS Management Console 등을 통한 RESTful API 포함)가 포함되어 있습니다.



AWS 결제 암호화는 웹에 연결된 AWS 결제 암호화 호스트와 HSM의 계층으로 구성된 계층형 서비스입니다. 이러한 계층화된 호스트의 그룹은 AWS 결제 암호화 스택을 구성합니다. AWS 결제 암호화에 대한 모든 요청은 전송 계층 보안 프로토콜(TLS)을 통해 이루어져야 하며 AWS 결제 암호화 호스트에서 종료되어야 합니다. 서비스 호스트는 [PFS\(Perfect Forward Secrecy\)](#)을 제공하는 암호 제품군이 있는 TLS만 허용합니다. 이 서비스는 다른 모든 AWS API 작업에 사용할 수 있는 것과 동일한 IAM의 보안 인증 정보 및 정책 메커니즘을 사용하여 요청을 인증하고 권한을 부여합니다.

AWS 결제 암호화 서버는 비 가상 사실 네트워크를 통해 기본 [HSM](#)에 연결합니다. 서비스 구성 요소와 [HSM](#) 간의 연결은 인증 및 암호화를 위한 상호 TLS(mTLS)로 보호됩니다.

설계 목표

AWS 결제 암호화는 다음 요구 사항을 충족하도록 설계되었습니다.

- 신뢰성 — 키 사용은 사용자가 정의하고 관리하는 액세스 제어 정책으로 보호됩니다. 일반 텍스트 AWS 결제 암호화 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. Amazon 직원은 HSM 기본(또는 마스터) 키 또는 백업에 액세스할 수 없습니다. 기본 키는 AWS 결제 암호화 리전에 속하지 않는 HSM과 동기화할 수 없습니다. 다른 모든 키는 HSM 기본 키로 보호됩니다. 따라서 고객 계정 내에서 운영되는 AWS 결제 암호화 서비스 외부에서는 고객 AWS 결제 암호화 키를 사용할 수 없습니다.

- 짧은 지연 시간 및 높은 처리량 — AWS 결제 암호화는 결제 암호화 키 관리 및 결제 트랜잭션 처리에 적합한 지연 시간 및 처리량 수준에서 암호화 작업을 제공합니다.
- 내구성 — 암호화 키의 내구성은 AWS에서 내구성이 가장 높은 서비스의 내구성과 동일하게 설계되었습니다. 단일 암호 키를 결제 단말기, EMV 칩 카드 또는 수년 동안 사용 중인 기타 보안 암호화 장치(SCD)와 공유할 수 있습니다.
- 독립 리전 — AWS는 여러 리전에서 데이터 액세스를 제한해야 하거나 데이터 레지던시 요건을 준수해야 하는 고객을 위해 독립적인 리전을 제공합니다. 키 사용은 단일 AWS 리전으로 제한됩니다.
- 안전한 난수 소스 — 강력한 암호화를 구현하려면 예측할 수 없는 난수 생성이 중요하므로 AWS 결제 암호화는 고품질의 검증된 난수 소스를 제공합니다. AWS 결제 암호화를 위한 모든 키 생성은 PCI 모드에서 운영되는 PCI PTS HSM 등록 HSM을 사용합니다.
- 감사 — AWS 결제 암호는 암호화 키의 사용 및 관리를 Amazon CloudWatch를 통해 사용할 수 있는 CloudTrail 로그와 서비스 로그에 기록합니다. CloudTrail 로그를 사용하여 키를 공유한 계정의 키 사용을 포함하여 암호화 키의 사용을 검사할 수 있습니다. AWS 결제 암호화는 해당 PCI, 카드 브랜드 및 지역 결제 보안 표준에 대해 제3자 평가자의 감사를 받습니다. 증명 및 공동 책임 가이드는 AWS Artifact에서 제공됩니다.
- 탄력성 — AWS 결제 암호화는 수요에 따라 확장 및 축소됩니다. AWS 결제 암호화는 HSM 용량을 예측하고 예약하는 대신 온디맨드형 결제 암호화를 제공합니다. AWS 결제 암호화는 고객의 최대 수요를 충족하기에 충분한 용량을 제공하기 위해 HSM의 보안 및 규정 준수를 유지할 책임이 있습니다.

기본

이 장의 항목에서는 AWS 결제 암호화의 암호화 기본 요소와 이러한 기본 요소가 사용되는 위치에 대해 설명합니다. 또한 이 서비스의 기본 요소를 소개합니다.

주제

- [암호 프리미티브](#)
- [엔트로피 및 난수 생성](#)
- [대칭 키 작업](#)

- [비대칭 키 작업](#)
- [키 스토리지](#)
- [대칭 키를 사용한 키 가져오기](#)
- [비대칭 키를 사용한 키 가져오기](#)
- [키 내보내기](#)
- [트랜잭션별 파생된 고유 키\(DUKPT\) 프로토콜](#)
- [키 계층 구조](#)

암호 프리미티브

AWS 결제 암호화는 매개 변수가 가능한 표준 암호화 알고리즘을 사용하므로 애플리케이션이 해당 사용 사례에 필요한 알고리즘을 구현할 수 있습니다. 암호화 알고리즘 세트는 PCI, ANSI X9, EMVCo 및 ISO 표준에 따라 정의됩니다. 모든 암호화는 PCI 모드에서 실행되는 PCI PTS HSM 표준 등재 HSM에 의해 수행됩니다.

엔트로피 및 난수 생성

AWS 결제 암호화 키 생성은 결제 암호화 HSM에서 수행됩니다. AWS HSM은 지원되는 모든 키 유형 및 파라미터에 대한 PCI PTS HSM 요구 사항을 충족하는 난수 생성기를 구현합니다.

대칭 키 작업

ANSI X9 TR 31, ANSI X9.24 및 PCI PIN 부록 C에 정의된 대칭 키 알고리즘 및 키 강도가 지원됩니다.

- 해시 함수 - 출력값 크기가 2551보다 큰 SHA2 및 SHA3 제품군의 알고리즘입니다. PCI PTS POI v3 이전 버전 단말기와의 역호환성은 제외됩니다.
- 암호화 및 암호 해독 - 키 크기가 128비트 이상인 AES 또는 키 크기가 112비트 이상인 TDEA(2 키 또는 3 키)입니다.
- 메시지 인증 코드(MAC) CMAC 또는 GMAC(AES 사용), 승인된 해시 함수가 있고 키 크기가 128보다 크거나 같은 HMAC.

AWS 결제 암호화는 HSM 기본 키, 데이터 보호 키 및 TLS 세션 키에 AES 256을 사용합니다.

비대칭 키 작업

ANSI X9 TR 31, ANSI X9.24 및 PCI PIN 부록 C에 정의된 비대칭 키 알고리즘 및 키 강도가 지원됩니다.

- 승인된 키 설정 체계 - NIST SP800-56A(ECC/FCC2 기반 키 계약), NIST SP800-56B(IFC 기반 키 계약) 및 NIST SP800-38F(AES 기반 키 암호화/래핑)에 설명되어 있습니다.

[AWS 결제 암호화 호스트는 완벽한 순방향 보안을 제공하는 암호 제품군이 있는 TLS를 사용하는 서비스에만 연결을 허용합니다.](#)

키 스토리지

AWS 결제 암호화 키는 HSM AES 256 기본 키로 보호되며 암호화된 데이터베이스의 ANSI X9 TR 31 키 블록에 저장됩니다. 데이터베이스는 결제 암호화 서버의 인메모리 데이터베이스에 복제됩니다.

AWS

PCI PIN 보안 규범 부록 C에 따르면 AES 256 키는 다음과 같거나 더 강력합니다.

- 3-key TDEA
- RSA 15360비트
- ECC 512비트
- DSA, DH 및 MQV 15360/512

대칭 키를 사용한 키 가져오기

AWS 결제 암호화는 대칭 또는 공개 키를 사용하는 암호 및 키 블록을 가져오고, 가져오기용 보호 키보다 강력하거나 강력한 대칭 키 암호화 키 (KEK) 를 사용하여 가져올 수 있도록 지원합니다.

비대칭 키를 사용한 키 가져오기

AWS 결제 암호화는 가져오기를 위한 보호 키보다 강력하거나 강력한 개인 키 암호화 키 (KEK) 로 보호되는 대칭 또는 공개 키를 사용하는 암호 및 키 블록을 가져올 수 있도록 지원합니다. 암호 해독을 위해 제공되는 퍼블릭 키는 고객이 신뢰하는 기관의 인증서를 통해 신뢰성과 무결성을 보장해야 합니다.

AWS 결제 암호학에서 제공하는 공용 KEK는 PCI PIN 보안 및 PCI P2PE 부록 A를 준수하는 것으로 입증된 인증 기관 (CA) 의 인증 및 무결성 보호 기능을 갖추고 있습니다.

키	설명
리전 기본 키	암호화 처리에 사용되는 가상 HSM 이미지 또는 프로필을 보호합니다. 이 키는 HSM 및 보안 백업에만 존재합니다.
프로필 기본 키	최고 수준 고객 키 보호 키로, 전통적으로 고객 키에 대한 로컬 마스터 키(LMK) 또는 마스터 파일 키(MFK)라고 합니다. 이 키는 HSM 및 보안 백업에만 존재합니다. 프로필은 결제 사용 사례의 보안 표준에서 요구하는 대로 고유한 HSM 구성을 정의합니다.
AWS 결제 암호화 공개 키 암호화 키 (KEK) 키의 신뢰 루트	비대칭 키를 사용한 키 가져오기 및 내보내기를 위해 AWS Payment Cryptography에서 제공하는 공개 키 인증 및 검증을 위한 신뢰할 수 있는 루트 공개 키 및 인증서.

고객 키는 다른 키를 보호하는 데 사용되는 키와 결제 관련 데이터를 보호하는 키별로 그룹화됩니다. 두 가지 유형의 고객 키의 예는 다음과 같습니다.

키	설명
공개 KEK 키에 대한 고객이 제공한 신뢰할 수 있는 루트	비대칭 키를 사용하여 키를 가져오고 내보낼 때 제공하는 퍼블릭 키를 인증하고 검증하기 위한 신뢰 루트로 사용자가 제공하는 퍼블릭 키 및 인증서입니다.
키 암호화 키(KEK)	KEK는 외부 키 저장소와 Payment Cryptography, 비즈니스 파트너, AWS 결제 네트워크 또는 조직 내 다른 애플리케이션 간의 교환을 위해 다른 키를 암호화하는 데만 사용됩니다.
트랜잭션별 파생된 고유 키(DUKPT) 기본 파생 키(BDK)	BDK는 각 결제 단말기의 고유 키를 생성하고 여러 단말기의 트랜잭션을 하나의 인수 은행 또는 취득자의 작업 키로 변환하는 데 사용됩니다. PCI Point-to-Point Encryption(P2PE)에서 요구

키	설명
	하는 모범 사례는 단말기 모델, 키 삽입 또는 초기화 서비스 또는 기타 세분화에 서로 다른 BDK를 사용하여 BDK 손상의 영향을 제한하는 것입니다.
결제 네트워크 영역 제어 마스터 키(ZCMK)	영역 키 또는 영역 마스터 키라고도 하는 ZCMK는 결제 네트워크에서 초기 작업 키를 설정하기 위해 제공됩니다.
DUKPT 트랜잭션 키	DUKPT용으로 구성된 결제 단말기는 단말기 및 트랜잭션을 위한 고유한 키를 생성합니다. 트랜잭션을 수신하는 HSM은 단말기 식별자와 거래 시퀀스 번호로부터 키를 결정할 수 있습니다.
카드 데이터 준비 키	EMV 발급자 마스터 키, EMV 카드 키 및 확인 값, 카드 개인 설정 데이터 파일 보호 키는 카드 개인화 공급자가 사용할 개별 카드에 대한 데이터를 생성하는 데 사용됩니다. 발급 은행 또는 발급자가 트랜잭션 승인의 일환으로 카드 데이터를 인증하는 데에도 이러한 키와 암호 검증 데이터를 사용합니다.
카드 데이터 준비 키	EMV 발급자 마스터 키, EMV 카드 키 및 확인 값, 카드 개인 설정 데이터 파일 보호 키는 카드 개인화 공급자가 사용할 개별 카드에 대한 데이터를 생성하는 데 사용됩니다. 발급 은행 또는 발급자가 트랜잭션 승인의 일환으로 카드 데이터를 인증하는 데에도 이러한 키와 암호 검증 데이터를 사용합니다.

키	설명
결제 네트워크 작업 키	발급자 작업 키 또는 취득자 작업 키라고도 하는 이 키는 결제 네트워크로 전송되거나 결제 네트워크로부터 수신되는 거래를 암호화하는 키입니다. 이러한 키는 네트워크에서 자주 교체되며, 매일 또는 매시간 교체되는 경우가 많습니다. PIN/Debit 트랜잭션을 위한 PIN 암호화 키(PEK)입니다.
개인 식별 번호(PIN) 암호화 키(PEK)	PIN 블록을 만들거나 해독하는 애플리케이션은 PEK를 사용하여 일반 텍스트 PIN의 저장 또는 전송을 방지합니다.

내부 작업

이 주제에서는 전 세계적으로 분산되고 확장 가능한 Payment Cryptography 및 키 관리 서비스를 위해 고객 키와 암호화 작업을 보호하기 위해 서비스에서 구현하는 내부 요구 사항을 설명합니다.

HSM 사양 및 수명 주기

AWS 결제 암호화는 상용 HSM을 사용합니다. HSM은 FIPS 140-2 Level 3 인증을 받았으며 펌웨어 버전과 PCI 보안 표준 위원회가 [승인한 PCI PTS 디바이스 목록](#)에 PCI HSM v3 준수 사항으로 나열된 보안 정책도 사용합니다. PCI PTS HSM 표준에는 결제 보안 및 규정 준수에 중요하지만 FIPS 140에서 다루지 않는 HSM 하드웨어의 제조, 배송, 배포, 관리 및 폐기에 대한 추가 요구 사항이 포함되어 있습니다.

모든 HSM은 PCI 모드에서 작동하며 PCI PTS HSM 보안 정책에 따라 구성됩니다. AWS 결제 암호화 사용 사례를 지원하는 데 필요한 기능만 사용할 수 있습니다. AWS 결제 암호화는 일반 텍스트 PIN의 인쇄, 표시 또는 반환을 제공하지 않습니다.

HSM 디바이스 물리적 보안

배송 전에 제조업체가 AWS 결제 암호화 인증 기관 (CA) 에서 서명한 장치 키가 있는 HSM만 서비스에서 사용할 수 있습니다. AWS 결제 암호화는 HSM 제조업체 및 장치 인증서에 대한 신뢰의 근원인 제조업체 CA의 하위 CA입니다. 제조업체 CA는 ANSI TR 34를 구현하고 PCI PIN 보안 부록 A 및 PCI P2PE 부록 A를 준수함을 입증했습니다. 제조업체는 결제 암호화 CA에서 서명한 디바이스 키가 있는 모든 HSM이 AWS의 지정 수취인에게 배송되는지 확인합니다. AWS

PCI PIN 보안에서 요구하는 대로 제조업체는 HSM 배송과 다른 통신 채널을 통해 일련번호 목록을 제공합니다. 이러한 일련 번호는 AWS 데이터 센터에 HSM을 설치하는 프로세스의 각 단계에서 확인됩니다. 마지막으로, AWS 결제 암호화 사업자는 설치된 HSM 목록을 제조업체 목록과 비교하여 유효성을 확인한 후 결제 암호화 키를 받을 수 있는 HSM 목록에 일련 번호를 추가합니다. AWS

HSM은 항상 안전한 스토리지에 보관되거나 이중 제어 하에 있으며, 여기에는 다음이 포함됩니다.

- 제조업체에서 AWS 랙 조립 시설로 배송.
- 랙 조립 중.
- 랙 조립 시설에서 데이터 센터로 배송.
- 입고 및 데이터 센터 보안 처리실에 설치. HSM 랙은 카드 액세스 제어 잠금 장치, 알람 도어 센서 및 카메라를 통한 이중 제어 기능을 갖추고 있습니다.
- 작동 중.
- 해체 및 폐기 중.

개별 책임과 함께 각 HSM에 대해 완전한 chain-of-custody 정보를 유지 관리하고 모니터링합니다.

HSM 초기화

HSM은 일련 번호, 제조업체에서 설치한 장치 키, 펌웨어 체크섬을 통해 ID 및 무결성을 확인한 후에만 AWS 결제 암호화 플릿의 일부로 초기화됩니다. HSM의 신뢰성과 무결성이 검증된 후에는 PCI 모드 활성화를 포함하여 HSM이 구성됩니다. 그러면 AWS 결제 암호화 영역 기본 키와 프로필 기본 키가 설정되고 서비스에서 HSM을 사용할 수 있습니다.

HSM 서비스 및 수리

HSM에는 디바이스의 암호화 경계를 위반할 필요가 없는 서비스 가능한 구성 요소가 있습니다. 이러한 구성 요소에는 냉각 팬, 전원 공급 장치 및 배터리가 포함됩니다. HSM 또는 HSM 랙 내의 다른 장치에 서비스가 필요한 경우 랙이 열려 있는 전체 기간 동안 이중 제어가 유지됩니다.

HSM 해제

서비스 해제는 HSM의 장애 end-of-life 또는 장애로 인해 발생합니다. HSM은 정상적으로 작동하는 경우 랙에서 분리되기 전에 논리적으로 제로화된 다음 AWS 데이터 센터의 안전한 처리실 내에서 파기됩니다. 이러한 제품은 수리를 위해 제조업체에 반환되거나, 다른 용도로 사용되지 않으며, 또는 파기 전에 안전한 처리실에서 제거되지 않습니다.

HSM 펌웨어 업데이트

업데이트가 보안과 관련이 있거나 고객이 새 버전의 기능을 활용할 수 있다고 판단되는 경우 PCI PTS HSM 및 FIPS 140-2 (또는 FIPS 140-3) 등재 버전과의 조정을 유지하기 위해 필요한 경우 HSM 펌웨어 업데이트가 적용됩니다. AWS 결제 암호화 HSM은 PCI PTS HSM 등재 버전과 일치하는 펌웨어를 실행합니다. off-the-shelf 새 펌웨어 버전은 PCI 또는 FIPS 인증 펌웨어 버전과의 무결성을 검증한 다음 모든 HSM에 롤아웃하기 전에 기능 테스트를 거칩니다.

운영자 액세스

운영자는 정상적인 운영 중에 HSM에서 수집한 정보만으로는 문제를 식별하거나 변경을 계획하기에 충분하지 않은 경우 콘솔 없이 HSM에 액세스하여 문제를 해결할 수 있습니다. 다음 단계가 실행됩니다.

- 문제 해결 활동이 개발 및 승인되고 비콘솔 세션이 예약됩니다.
- HSM이 고객 처리 서비스에서 제거됩니다.
- 이중 제어 하에 기본 키가 삭제됩니다.
- 운영자는 콘솔 이외에서도 HSM에 액세스하여 이중 제어 상태에서 승인된 문제 해결 활동을 수행할 수 있습니다.
 - 비콘솔이 세션이 종료되면 HSM에서 초기 프로비저닝 프로세스를 수행하여 표준 펌웨어와 구성을 반환한 다음 기본 키를 동기화한 후 HSM을 반환하여 고객에게 서비스합니다.
 - 세션 레코드는 변경 내용 추적에 기록됩니다.
 - 세션에서 얻은 정보는 향후 변경 계획을 수립하는 데 사용됩니다.

콘솔이 아닌 모든 액세스 기록을 검토하여 프로세스 규정 준수 여부와 HSM 모니터링, 관리 프로세스 또는 운영자 교육의 잠재적 변경 사항을 검토합니다. non-console-access

키 관리

리전 내 모든 HSM은 리전 기본 키와 동기화됩니다. 리전 기본 키는 하나 이상의 프로필 기본 키를 보호합니다. 프로필 기본 키는 고객 키를 보호합니다.

모든 기본 키는 HSM에서 생성되며 ANSI X9 TR 34 및 PCI PIN 부록 A에 따라 비대칭 기법을 사용하여 대칭 키 분포를 통해 배포됩니다.

주제

- [생성](#)

- [리전 기본 키 동기화](#)
- [리전 기본 키 교체](#)
- [프로필 기본 키 동기화](#)
- [프로필 기본 키 교체](#)
- [보호](#)
- [내구성](#)
- [통신 보안](#)
- [고객 키 관리](#)
- [로깅 및 모니터링](#)

생성

AES 256비트 기본 키는 PCI PTS HSM 난수 생성기를 사용하여 서비스 HSM 플릿용으로 프로비저닝된 HSM 중 하나에서 생성됩니다.

리전 기본 키 동기화

HSM 리전 기본 키는 ANSI X9 TR-34에 따라 정의한 메커니즘을 사용하여 지역 플릿 전반의 서비스에 의해 동기화되며, 여기에는 다음이 포함됩니다.

- KDH(키 분포 호스트)와 KRD(키 수신 디바이스) 키 및 인증서를 사용한 상호 인증으로 퍼블릭 키에 대한 인증 및 무결성을 제공합니다.
- 인증서는 PCI PIN Annex A2의 요구 사항을 충족하는 인증 기관(CA)에서 서명합니다. 단, AES 256 비트 키를 보호하는 데 적합한 비대칭 알고리즘과 키 강도는 예외입니다.
- 배포된 대칭 키의 식별 및 키 보호는 ANSI X9 TR-34 및 PCI PIN 부록 A1과 일치합니다. 단, AES 256비트 키를 보호하는 데 적합한 비대칭 알고리즘 및 키 강도는 예외입니다.

리전 기본 키는 다음과 같은 방법으로 해당 리전에 대해 인증 및 프로비저닝된 HSM에 대해 설정됩니다.

- 기본 키는 해당 리전의 HSM에서 생성됩니다. 해당 HSM은 키 분포 호스트로 지정됩니다.
- 해당 리전에 프로비저닝된 모든 HSM은 HSM의 퍼블릭 키와 재생할 수 없는 인증 정보를 포함하는 KRD 인증 토큰을 생성합니다.
- KDH가 HSM의 ID와 키 수신 권한을 확인한 후 KRD 토큰이 KDH 허용 목록에 추가됩니다.
- KDH는 각 HSM에 대해 인증 가능한 기본 키 토큰을 생성합니다. 토큰에는 생성 대상 HSM에서만 로드할 수 있는 KDH 인증 정보와 암호화된 기본 키가 포함되어 있습니다.

- 각 HSM에는 이를 위해 빌드된 기본 키 토큰이 전송됩니다. HSM의 자체 인증 정보와 KDH 인증 정보를 검증한 후 KRD 프라이빗 키로 기본 키를 해독하여 기본 키에 로드합니다.

단일 HSM을 특정 리전과 다시 동기화해야 하는 경우:

- 펌웨어 및 구성을 통해 다시 검증되고 프로비저닝됩니다.
- 해당 리전을 처음 사용하는 경우:
 - HSM은 KRD 인증 토큰을 생성합니다.
 - KDH는 토큰을 허용 목록에 추가합니다.
 - KDH는 HSM의 기본 키 토큰을 생성합니다.
 - HSM은 기본 키를 로드합니다.
 - HSM은 서비스에서 사용할 수 있습니다.

이를 통해 다음이 보장됩니다.

- 지역 내에서 AWS 결제 암호화 처리가 검증된 HSM만 해당 지역의 마스터 키를 받을 수 있습니다.
- AWS 결제 암호화 HSM의 마스터 키만 플릿 내 HSM에 배포할 수 있습니다.

리전 기본 키 교체

리전 기본 키는 암호화 기간 만료 시, 키 손상이 의심되는 경우 교체되거나 키 보안에 영향을 미치는 것으로 판단되는 서비스 변경 이후 교체됩니다.

초기 프로비저닝과 마찬가지로 새 리전 기본 키가 생성되고 배포됩니다. 저장된 프로필 기본 키를 새 리전 기본 키로 변환해야 합니다.

리전 기본 키 교체는 고객 처리에 영향을 미치지 않습니다.

프로필 기본 키 동기화

프로필 기본 키는 리전 기본 키로 보호됩니다. 이렇게 하면 프로필이 특정 리전으로 제한됩니다.

프로필 기본 키는 다음에 따라 제공됩니다.

- 프로필 기본 키가 리전 기본 키가 동기화된 HSM에서 생성됩니다.
- 프로필 기본 키가 프로필 구성 및 기타 컨텍스트와 함께 저장되고 암호화됩니다.

- 프로필이 리전 기본 키가 있는 리전의 모든 HSM에서 고객 암호화 기능에 사용됩니다.

프로필 기본 키 교체

프로필 기본 키는 암호화 기간 만료 시, 키 손상이 의심되는 경우 또는 키 보안에 영향을 미치는 것으로 판단되는 서비스 변경 이후 교체됩니다.

교체 단계:

- 새 프로필 기본 키가 생성되고 초기 프로비저닝과 마찬가지로 보류 중인 기본 키로 배포됩니다.
- 백그라운드 프로세스는 고객 키 자료를 설정된 프로필 기본 키에서 보류 중인 기본 키로 변환합니다.
- 보류 중인 키를 사용하여 모든 고객 키를 암호화하면 보류 중인 키가 프로필 기본 키로 승격됩니다.
- 백그라운드 프로세스를 통해 만료된 키로 보호되는 고객 키 구성 요소가 삭제됩니다.

프로필 기본 키 교체는 고객 처리에 영향을 주지 않습니다.

보호

키는 키 계층 구조에만 의존하여 보호됩니다. 모든 고객 키의 손실이나 손상을 방지하려면 기본 키를 보호하는 것이 중요합니다.

리전 기본 키는 백업에서 서비스를 위해 인증되고 프로비저닝된 HSM으로만 복원할 수 있습니다. 이러한 키는 특정 HSM에 대한 특정 KDH의 상호 인증 가능하고 암호화된 기본 키 토큰으로만 저장할 수 있습니다.

프로필 마스터 키는 지역별로 암호화된 프로필 구성 및 컨텍스트 정보와 함께 저장됩니다.

고객 키는 키 블록에 저장되며 프로필 마스터 키로 보호됩니다.

모든 키는 HSM 내에서만 존재하거나 암호화 강도가 같거나 더 강력한 다른 키로 보호되어 저장됩니다.

내구성

거래 암호화 및 비즈니스 기능을 위한 고객 키는 일반적으로 중단이 발생할 수 있는 극한 상황에서도 사용할 수 있어야 합니다. AWS 결제 암호화는 가용 영역 및 지역 전반에 걸쳐 다단계 중복 모델을 활용합니다. AWS 결제 암호화 작업에 대해 서비스에서 제공하는 것보다 더 높은 가용성과 내구성을 원하는 고객은 다중 리전 아키텍처를 구현해야 합니다.

HSM 인증 및 기본 키 토큰은 저장되며 HSM을 재설정해야 하는 경우 기본 키를 복원하거나 새 기본 키와 동기화하는 데 사용할 수 있습니다. 토큰은 필요할 때만 이중 제어 상태에서 보관 및 사용됩니다.

통신 보안

외부

AWS 결제 암호화 API 엔드포인트는 요청 인증 및 무결성에 대한 1.2 이상의 TLS 및 서명 버전 4를 포함한 AWS 보안 표준을 충족합니다.

수신 TLS 연결은 네트워크 로드 밸런서에서 종료되고 내부 TLS 연결을 통해 API 핸들러로 전달됩니다.

Internal

서비스 구성 요소 간 그리고 서비스 구성 요소와 다른 AWS 서비스 간의 내부 통신은 강력한 암호화를 사용하는 TLS로 보호됩니다.

HSM은 서비스 구성 요소에서만 연결할 수 있는 비 가상 사설 네트워크에 있습니다. HSM과 서비스 구성 요소 간의 모든 연결은 TLS 1.2 이상의 상호 TLS(mTLS)로 보호됩니다. TLS 및 MTL용 내부 인증서는 AWS 프라이빗 인증 기관을 사용하는 Amazon 인증서 관리자에서 관리합니다. 내부 VPC와 HSM 네트워크를 모니터링하여 예외가 없는 활동과 구성 변경이 있는지 확인합니다.

고객 키 관리

AWS에서는 고객의 신뢰를 최우선으로 생각합니다. AWS 계정으로 서비스에 업로드하거나 생성한 키를 완전히 제어하고 키에 대한 액세스를 구성할 책임이 있습니다.

AWS 결제 암호화는 서비스에서 관리하는 키에 대한 HSM의 물리적 규정 준수 및 키 관리에 대한 전적인 책임을 집니다. 이를 위해서는 HSM 기본 키의 소유권 및 관리와 AWS 결제 암호화 키 데이터베이스 내에 보호된 고객 키를 저장해야 합니다.

고객 키 스페이스 분리

AWS 결제 암호화는 키를 다른 계정과 명시적으로 공유하지 않는 한 키를 소유한 계정으로 보안 주체를 제한하는 것을 포함하여 모든 키 사용에 대해 키 정책을 적용합니다.

백업 및 복구

AWS는 특정 지역의 키와 키 정보를 암호화된 아카이브에 백업합니다. 아카이브를 복원하려면 이중 제어 기능이 필요합니다. AWS

키 블록

모든 키는 ANSI X9 TR-31 형식 키 블록에 저장됩니다.

에서 지원하는 크립토그램 또는 기타 키 블록 형식에서 키를 서비스로 가져올 수 있습니다. ImportKey 마찬가지로 키를 내보낼 수 있는 경우 키 내보내기 프로필에서 지원하는 다른 키 블록 형식 또는 암호문으로 내보낼 수 있습니다.

키 사용

키 사용은 KeyUsage 서비스에서 구성한 것으로 제한됩니다. 요청된 암호화 작업에 대한 부적절한 키 사용, 사용 모드 또는 알고리즘이 있는 요청은 서비스가 실패합니다.

키 교환 관계

PCI PIN 보안 및 PCI P2PE를 사용하려면 해당 키를 공유하는 데 사용되는 KEK를 포함하여 PIN을 암호화하는 키를 공유하는 조직이 해당 키를 다른 조직과 공유하지 않도록 해야 합니다. 대칭 키는 같은 조직 내를 포함하여 두 당사자 간에만 공유하는 것이 가장 좋습니다. 이렇게 하면 키 손상이 의심되어 영향을 받은 키를 교체해야 하는 등의 영향을 최소화할 수 있습니다.

두 명 이상의 당사자 간에 키를 공유해야 하는 비즈니스 케이스라도 당사자 수를 최소한으로 유지해야 합니다.

AWS 결제 암호화는 해당 요구 사항 내에서 키 사용을 추적하고 적용하는 데 사용할 수 있는 키 태그를 제공합니다.

예를 들어, 서비스 공급자와 공유하는 모든 키에 대해 "KIF"="POSStation"으로 설정하여 다양한 키 입력 기능에 대한 KEK 및 BDK를 식별할 수 있습니다. 또 다른 예로 결제 네트워크와 공유된 키에 "Network" = "PayCard" 로 태그를 지정하는 경우를 들 수 있습니다. 태그를 사용하면 액세스 제어를 생성하고 감사 보고서를 생성하여 키 관리 방식을 적용하고 입증할 수 있습니다.

키 삭제

DeleteKey 고객이 설정할 수 있는 기간이 지나면 데이터베이스의 키를 삭제하도록 표시합니다. 이 기간이 지나면 키는 복구할 수 없이 삭제됩니다. 이는 실수 또는 악의적으로 키를 삭제하는 것을 방지하기 위한 안전 메커니즘입니다. 삭제 대상으로 표시된 키는 다음을 제외한 모든 작업에 사용할 수 없습니다. RestoreKey

삭제된 키는 삭제 후 7일 동안 서비스 백업에 남아 있습니다. 이 기간 동안에는 복구할 수 없습니다.

폐쇄된 AWS 계정에 속한 키는 삭제하도록 표시됩니다. 삭제 기간이 되기 전에 계정을 다시 활성화하면 삭제 표시된 모든 키는 복원되지만 비활성화됩니다. 암호화 작업에 사용하려면 사용자가 다시 활성화해야 합니다.

키 공유

AWS 리소스 액세스 관리자(<https://docs.aws.amazon.com/ARG/index.html>)를 사용하여 키를 조직 내부 또는 외부의 다른 계정과 공유할 수 있습니다. 키를 리소스 공유로 그룹화한 다음 계정 또는 계정 내 특정 IAM 사용자 및 역할과 공유할 수 있습니다. 각 리소스 공유에 대한 사용 권한을 지정합니다. 공유 권한은 키 리소스 정책에 의해 제한됩니다. 공유 키는 자체 정책에 의해 제한되는 작업을 허용하지 않습니다. 공유 권한은 언제든지 철회될 수 있습니다.

로깅 및 모니터링

내부 서비스 로그에는 다음이 포함됩니다.

- CloudTrail 서비스에서 이루어진 AWS 서비스 호출 로그
- CloudWatch 두 이벤트의 로그는 HSM의 CloudWatch 로그 또는 이벤트에 직접 기록됩니다.
- HSM 및 서비스 시스템의 로그 파일
- 로그 보관

모든 로그 소스는 키 정보를 포함한 민감한 정보를 모니터링하고 필터링합니다. 로그를 체계적으로 검토하여 민감한 고객 정보가 포함되어 있지 않은지 확인합니다.

직무 수행에 필요한 개인만 로그에 액세스할 수 있습니다.

모든 로그는 AWS 로그 보존 정책에 따라 보관됩니다.

고객 작업

AWS 결제 암호화는 PCI 표준에 따른 HSM 물리적 규정 준수에 대한 전적인 책임을 집니다. 또한 이 서비스는 안전한 키 저장소를 제공하며, 키를 생성하거나 가져올 때 PCI 표준에서 허용하고 사용자가 지정한 용도로만 사용할 수 있도록 합니다. 서비스의 보안 및 규정 준수 기능을 활용하기 위한 주요 속성 및 액세스 권한을 구성하는 것은 사용자의 책임입니다.

주제

- [키 생성](#)
- [키 가져오기](#)
- [키 내보내기](#)
- [키 삭제](#)
- [키 교체 중](#)

키 생성

키를 생성할 때 서비스가 규정을 준수하여 키를 사용하도록 강제하는 데 사용하는 속성을 설정합니다.

- 알고리즘 및 키 길이
- 사용량
- 가용성 및 만료

ABAC(속성 기반 액세스 제어)에 사용되는 태그를 사용하여 특정 파트너나 애플리케이션에서 사용할 수 있도록 키를 제한하는 데 사용되는 태그도 생성 중에 설정해야 합니다. 태그를 삭제하거나 변경할 수 있는 역할을 제한하는 정책을 포함해야 합니다.

키를 생성하기 전에 키를 사용하고 관리할 수 있는 역할을 결정하는 정책을 설정해야 합니다.

Note

CreateKey 명령에 대한 IAM 정책을 사용하여 키 생성에 대한 이중 제어를 시행하고 입증할 수 있습니다.

키 가져오기

키를 가져올 때 키 블록에 암호적으로 바인딩된 정보를 사용하여 서비스에서 키 사용의 규정 준수를 강제하는 속성을 설정합니다. 기본 키 컨텍스트를 설정하는 메커니즘은 소스 HSM으로 생성되고 공유 또는 비대칭 [KEK](#)로 보호되는 키 블록을 사용하는 것입니다. 이는 PCI PIN 요구 사항에 부합하며 소스 애플리케이션의 사용량, 알고리즘 및 키 강도를 보존합니다.

가져올 때 키 블록의 정보 외에도 중요한 주요 속성, 태그 및 액세스 제어 정책을 설정해야 합니다.

암호화를 사용하여 키를 가져오는 경우 소스 애플리케이션의 주요 속성이 전송되지 않습니다. 이 메커니즘을 사용하여 속성을 적절하게 설정해야 합니다.

키 관리자가 전송하는 일반 텍스트 구성 요소를 사용하여 키를 교환한 다음 보안실에서 이중 제어를 구현하는 과정을 통해 키를 로드하는 경우가 많습니다. 이는 AWS 결제 암호화에서 직접 지원되지 않습니다. API는 자체 HSM에서 가져올 수 있는 인증서와 함께 퍼블릭 키를 내보내 서비스에서 가져올 수 있는 키 블록을 내보냅니다. 이를 통해 자체 HSM을 사용하여 일반 텍스트 구성 요소를 로드할 수 있습니다.

키 검사 값(KCV)을 사용하여 가져온 키가 소스 키와 일치하는지 확인해야 합니다.

ImportKey API의 IAM 정책을 사용하여 키 가져오기에 대한 이중 제어를 시행하고 입증할 수 있습니다.

키 내보내기

파트너 또는 온프레미스 애플리케이션과 키를 공유하려면 키를 내보내야 할 수 있습니다. 내보내기에 키 블록을 사용하면 암호화된 키 구성 요소와 함께 기본적인 키 컨텍스트를 유지할 수 있습니다.

키 태그를 사용하여 동일한 태그와 값을 공유하는 키를 KEK로 내보내도록 제한할 수 있습니다.

AWS 결제 암호화는 일반 텍스트 키 구성 요소를 제공하거나 표시하지 않습니다. 이를 위해서는 키 관리자가 PCI PTS HSM 또는 ISO 13491 테스트를 거친 보안 암호화 디바이스(SCD)에 직접 액세스하여 표시하거나 인쇄해야 합니다. SCD에 비대칭 KEK 또는 대칭 KEK를 설정하여 이중 제어 하에 일반 텍스트 키 구성 요소 생성 과정을 수행할 수 있습니다.

대상 HSM에서 가져온 항목이 소스 키와 일치하는지 확인하려면 키 검사 값(KCV)을 사용해야 합니다.

키 삭제

키 삭제 API를 사용하여 구성된 기간이 지난 후 키가 삭제되도록 스케줄을 지정할 수 있습니다. 그 전에는 키를 복구할 수 있습니다. 키가 삭제되면 서비스에서 영구적으로 제거됩니다.

DeleteKey API의 IAM 정책은 키 삭제에 대한 이중 제어를 시행하고 입증하는 데 사용될 수 있습니다.

키 교체 중

키 별칭을 사용하여 새 키를 생성하거나 가져온 다음 새 키를 참조하도록 키 별칭을 수정하여 키 교체의 효과를 구현할 수 있습니다. 이전 키는 관리 방식에 따라 삭제되거나 비활성화될 수 있습니다.

에 대한 할당량 AWS Payment Cryptography

AWS 계정에는 각 AWS 서비스에 대한 기본 할당량(이전에는 제한이라고 함)이 있습니다. 다르게 표시되지 않는 한, 리전별로 각 할당량이 적용됩니다. 일부 할당량에 대한 증가를 요청할 수 있으며 다른 할당량은 늘릴 수 없습니다.

명칭	기본값	조정 가능	설명
에일리어스	지원되는 각 리전: 2,000개	예	현재 리전에서 이 계정에 사용할 수 있는 별칭의 최대 수입입니다.
컨트롤 플레인 요청의 합산 비율	각각 지원되는 리전: 초당 5개	예	현재 리전의 이 계정에서 수행할 수 있는 초당 컨트롤 플레인 요청의 최대 수입입니다. 이 할당량은 모든 컨트롤 플레인 작업의 합산에 적용됩니다.
데이터 영역 요청의 합산 비율(비대칭)	각각 지원되는 리전: 초당 20개	예	현재 리전의 이 계정에서 수행할 수 있는 비대칭 키를 사용하는 데이터 영역 작업에 대한 초당 최대 요청 수입입니다. 이 할당량은 모든 데이터 영역 작업의 합산에 적용됩니다.
데이터 영역 요청의 합산 비율(대칭)	지원되는 각 리전: 초당 500개	예	현재 리전의 이 계정에서 수행할 수 있는 대칭 키를 사용하는 데이터 영역 작업에 대한 초당 최대 요청 수입입니다. 이 할당량은 모든 데이터 영역 작업의 합산에 적용됩니다.

명칭	기본값	조정 가능	설명
키	지원되는 각 리전: 2,000	예	현재 지역에 있는 이 계정의 최대 키 수입니다. 삭제된 키는 제외됩니다.

AWS 결제 암호화 사용 설명서의 문서 기록

다음 표에서는 AWS 결제 암호화에 대한 문서 릴리스를 소개합니다.

변경 사항	설명	날짜
기능 출시	RSA를 사용한 키 가져오기/내보내기 및 DUKPT IPEK/IK 키 내보내기와 관련된 새로운 기능에 대한 정보가 추가되었습니다.	2024년 1월 15일
최초 릴리스	AWS 결제 암호화 사용 설명서의 초기 릴리스	2023년 6월 8일

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