aws

用户指南

AWS 支付密码学

AWS 支付密码学: 用户指南

Table of Contents

什么是 AWS Payment Cryptography？

AWS Payment Cryptography 是一项托管 AWS 服务，可根据支付卡行业(PCI)标准提供对支付处理中使用的加密功能和密钥管理的访问，而无需您购买专用的支付 HSM 实例。AWSPayment Cryptography 为执行支付功能的客户（例如收单机构、支付服务商、网络、交换机、处理器和银行）提供了将其支付加密操作移至更靠近云中应用程序的能力，并最大限度地减少包含专用支付 HSM 的辅助数据中心或主机托管设施的依赖。

该服务旨在满足适用的行业规则，包括 PCI PIN、PCI P2PE 和 PCI DSS，并且该服务利用经过 PCI PTS HSM V3 和 FIPS 140-2 Level 3 认证的硬件。它旨在支持低延迟、高水平的正常运行时间和故障恢复能力。 AWSPayment Cryptography 具有完全的弹性，消除了本地 HSM 的许多操作要求，例如需要配置硬件、安全地管理密钥材料以及在安全的设施中维护紧急备份。AWSPayment Cryptography 还为您提供了以电子方式与合作伙伴共享密钥的选项，无需共享纸质明文组件。

您可以使用 AWS Payment Cryptography 控制面板 API 来创建和管理密钥。

您可以使用 AWS Payment Cryptography 数据面板 API，以便使用加密密钥进行与支付相关的交易处理和相关的加密操作。

AWS Payment Cryptagry Clyptagry 提供了可用于管理密钥的重要功能：

- 创建和管理对称和非对称 AWS Payment Cryptography 密钥，包括 TDES、AES 和 RSA 密钥，并指定其预期用途，例如生成 CVV 或 DUKPT 密钥派生。
- 安全地自动存储您的 AWS Payment Cryptography 密钥，并受硬件安全模块 (HSM) 保护，同时在用例之间强制执行密钥分离。
- 创建、删除、列出和更新别名，这些别名是 “友好名称”，可用于访问或控制对您的 AWS Payment Cryptography 密钥的访问权限。
- 标记您的 AWS Payment Cryptography 密钥以进行识别、分组、自动化、访问控制和成本跟踪。
- 使用密钥加密密钥 (KEK) 在 AWS Payment Cryptography 和 HSM（或第三方）之间导入和导出对称密钥，符合 TR-31（可互操作的安全密钥交换密钥区块规范）。
- 使用非对称密钥对在 AWS Payment Cryptography 和其他系统之间导入和导出对称密钥加密密钥 (KEK)，然后使用电子手段，例如 TR-34（使用非对称技术分发对称密钥的方法）。

您可以在加密操作中使用您的 AWS Payment Cryptography 密钥，例如：

- 使用对称或非对称 AWS Payment Cryptography 密钥对数据进行加密、解密和重新加密。

- 根据 PCI PIN 规则，在加密密钥之间安全地转换敏感数据（例如持卡人密码），而不会暴露明文。
- 生成或验证持卡人数据，例如 CVV、CVV2 或 ARQC。
- 生成并验证持卡人 pin 码。
- 生成或验证 MAC 签名。

概念

了解 AWS 支付密码学中使用的基本术语和概念，以及如何使用它们来帮助您保护数据。

别名

与 AWS 支付密码学密钥关联的用户友好名称。在许多 AWS 支付密码学 API 操作中，别名可以与密钥 ARN 互换使用。别名允许轮换或以其他方式更改密钥，而不会影响您的应用程序代码。别名是最多 256 个字符的字符串。它可以唯一标识账户和区域内关联的 AWS 支付密码密钥。在 AWS 支付密码学中，别名始终以开头。`alias/`

别名的格式如下：

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

例如：

```
alias/sampleAlias2
```

密钥 ARN

密钥 ARN 是 AWS Payment Cryptography 中密钥条目的 Amazon 资源名称（ARN）。它是 AWS 支付密码学密钥的唯一且完全限定的标识符。密钥 ARN 包括 AWS 账户、区域和随机生成的 ID。ARN 与密钥材料无关，也并非源自密钥材料。由于它们是在创建或导入操作期间自动分配的，因此这些值不具备幂等性。多次导入同一个密钥将导致多个密钥 ARN 具有自己的生命周期。

密钥 ARN 的格式如下：

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

以下是示例密钥 ARN：

```
arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h
```

密钥标识符

密钥标识符是对密钥的引用，其中一个（或多个）是 AWS 支付密码学操作的典型输入。有效的密钥标识符可以是 Key Arn 或密钥别名。

AWS 付款密码学密钥

AWS 支付密码学密钥（密钥）用于所有加密功能。密钥可以由您使用 create key 命令直接生成，也可以通过调用密钥导入添加到系统中。可以通过查看属性来确定密钥的来源 KeyOrigin。 AWS Payment Cryptography 还支持加密操作期间使用的派生密钥或中间密钥，例如 DUKPT 使用的密钥。

这些密钥在创建时定义了不可变和可变属性。算法、长度和用法等属性是在创建时定义的，无法更改。其他内容（例如生效日期或到期日期）可以修改。有关AWS 支付密码学密钥属性的完整列表，请参阅 AWS 支付密码学 API 参考。

AWS 支付密码学密钥的密钥类型主要由 ANSI X9 TR 31 定义，这些密钥类型仅限于 PCI PIN v3.1 要求 19 中规定的预期用途。

按照 PCI PIN v3.1 第 18-3 条要求的规定进行存储、与其他账户共享或导出时，属性将使用密钥块绑定到密钥。

密钥在 AWS 支付密码学平台中使用称为密钥 Amazon 资源名称 (ARN) 的唯一值进行识别。

> **Note**
>
> 密钥ARN是在最初创建密钥或将密钥导入 AWS 支付加密服务时生成的。因此，如果使用导入密钥功能多次添加相同的密钥材料，则相同的密钥材料将位于多个密钥 下，但每个密钥都有不同的密钥生命周期。

行业术语

主题

常见密钥类型

AWK

收单方工作密钥 (AWK) 是通常用于在收单方/收单方处理器和网络（例如 Visa 或 Mastercard）之间交换数据的密钥。从历史上看，AWK 利用 3DES 进行加密，将表示为 TR31_P0_PIN_ENCRYPTION_KEY。

BDK

基础派生密钥 (BDK) 是用于派生后续密钥的工作密钥，通常用作 PCI PIN 和 PCI P2PE DUKPT 流程的一部分。它表示为 TR31_B0_BASE_DERIVATION_KEY。

CMK

卡片主密钥 (CMK) 是一个或多个卡专用密钥，通常源自发行人主密钥、PAN 和 PSN，通常是 3DES 密钥。在个性化过程中，这些密钥存储在 EMV 芯片上。CMK 的示例包括 AC、SMI 和 SMC 密钥。

CMK-AC

应用程序密码 (AC) 密钥用作 EMV 交易的一部分，用于生成交易密码，是一种卡片主密钥。

CMK-SMI

安全消息完整性 (SMI) 密钥用作 EMV 的一部分，用于验证使用 MAC 发送到卡的有效负载（例如 pin 码更新脚本）的完整性。它是一种卡片主密钥。

CMK-SMC

安全消息保密性 (SMC) 密钥用作 EMV 的一部分，用于加密发送到卡片的数据，例如密码更新。它是一种卡片主密钥。

CVK

卡片验证密钥 (CVK) 是用于使用定义的算法生成 CVV、CVV2 和类似值以及验证输入的密钥。它表示为 TR31_C0_CARD_VERIFICATION_KEY。

iCvV

iCvV 是一个类似 Cvv2 的值，但在 EMV（Chip）卡上嵌入了 track2 的等效数据。该值是使用服务代码 999 计算得出的，与 CVV1/CVV2 不同，以防止窃取的信息被用来创建不同类型的新支付凭证。例如，如果获得了芯片交易数据，则无法使用这些数据生成磁条 (CVV1) 或用于在线购买 (CVV2)。

它使用一把???钥匙

IMK

发行方主密钥 (IMK) 是用作 EMV 芯片卡个性化一部分的主密钥。通常会有 3 个 IMK，分别用于 AC（密码）、SMI（用于完整性/签名的脚本主密钥）和 SMC（用于保密/加密的脚本主密钥）密钥。

IK

初始密钥 (IK) 是 DUKPT 过程中使用的第一个密钥，源自基本派生密钥 (BDK)。此密钥上不会处理任何交易，但它用于派生未来将用于交易的密钥。创建 IK 的推导方法是在 X9. 24-1:2017 中定义的。使用 TDES BDK 时，X9. 24-1:2009 是适用的标准，IK 被初始密码加密密钥 (IPEK) 所取代。

IPEK

初始 PIN 加密密钥 (IPEK) 是 DUKPT 流程中使用的初始密钥，源自基本派生密钥 (BDK)。此密钥上不会处理任何交易，但它用于派生未来将用于交易的密钥。IPEK 用词不当，因为这个密钥也可以用来派生数据加密和 Mac 密钥。创建 IPEK 的推导方法是在 X9 中定义的。24-1:2009。使用 AES BDK 时，X9. 24-1:2017 是适用的标准，IPEK 被初始密钥 (IK) 所取代。

IWK

发行方工作密钥 (IWK) 是通常用于在发行方/发行方处理器和网络（例如 Visa 或 Mastercard）之间交换数据的一种密钥。从历史上看，IWK 利用 3DES 进行加密，表示为 TR31_P0_PIN_ENCRYPTION_KEY。

KEK

密钥加密密钥 (KEK) 是用于加密其他密钥以进行传输或存储的密钥。根据标准，用于保护其他密钥的密钥通常为 TR31_K0_KE KeyUsage Y_ENCRYPTION_KE Y。TR-31

PEK

PIN 加密密钥 (PEK) 是一种工作密钥，用于加密 PIN 以进行存储或在双方之间传输。IWK 和 AWK 是 pin 加密密钥具体用途的两个示例。这些密钥表示为 TR31_P0_PIN_ENCRYPTION_KEY。

PVK

PIN 验证密钥 (PVK) 是一种工作密钥，用于生成 PVV 等 PIN 验证值。两种最常见的类型是用于生成 IBM3624 偏移值的TR31_V1_IBM3624_PIN_VERIFICATION_KEY 和用于 Visa/ABA 验证值的 TR31_V2_VISA_PIN_VERIFICATION_KEY。

其他术语

ARQC

授权请求密码 (ARQC) 是由 EMV 标准芯片卡（或等效的非接触式实现）在交易时生成的密码。通常，ARQC 由芯片卡生成，并在交易时转发给发卡机构或其代理进行验证。

DUKPT

每次交易派生唯一密钥 (DUKPT) 是一种密钥管理标准，通常用于定义实物 POS/POI 上一次性加密密钥的使用。从历史上看，DUKPT 利用 3DES 进行加密。DUKPT 的行业标准在 ANSI X9.24-3-2017 中定义。

EMV

EMV（最初是Europay、Mastercard、Visa）是一个与支付利益相关者合作创建可互操作的支付标准和技术的技术机构。一个示例标准是芯片卡/非接触式卡及其与之交互的支付终端，包括使用的加密技术。EMV 密钥派生是指根据一组初始密钥为每张支付卡生成唯一密钥的方法，例如 IMK

HSM

硬件安全模块 (HSM) 是一种物理设备，用于保护加密操作（例如加密、解密和数字签名）以及用于这些操作的底层密钥。

KCV

密钥校验值 (KCV) 是指各种校验和方法，主要用于在无需访问实际密钥材料的情况下比较彼此的密钥。KCV 也被用于完整性验证（尤其是在交换密钥时），尽管此角色现在已作为密钥块格式的一部分包括在内，例如 TR-31。对于 TDES 密钥，KCV 是通过使用要检查的密钥对每个值为零的 8 个字节进行加密，并保留加密结果的 3 个最高阶字节来计算的。对于 AES 密钥，KCV 使用 CMAC 算法计算，其中输入数据为 16 个字节的零，并保留加密结果的 3 个最高位字节。

KDH

密钥分配主机 (KDH) 是在密钥交换过程（例如 TR-34）中发送密钥的设备或系统。从 AWS 支付密码学发送密钥时，它被视为 KDH。

KIF

密钥注入工具 (KIF) 是一种安全设施，用于初始化支付终端，包括向支付终端加载加密密钥。

KRD

密钥接收设备 (KRD) 是在密钥交换过程中（例如 TR-34）中接收密钥的设备。向 AWS 支付密码学发送密钥时，它被视为 KRD。

KSN

密钥序列号 (KSN) 是用作 DUKPT 加密/解密输入的值，用于为每笔交易创建唯一的加密密钥。KSN 通常由一个 BDK 标识符、一个半唯一的终端 ID 以及一个交易计数器组成，该计数器在给定支付终端上处理的每次转换时递增。

PAN

主账号 (PAN) 是信用卡或借记卡等账户的唯一标识符。长度通常为 13-19 位数字。前 6-8 位数字标识网络和发卡行。

PIN 码块

在处理或传输过程中包含 PIN 码以及其他数据元素的数据块。PIN 码块格式标准化了 PIN 码块的内容以及如何处理它以检索 PIN 码。大多数 PIN 块由 PIN 和 PIN 长度组成，通常包含部分或全部 PAN。 AWS 支付密码学支持 ISO 9564-1 格式 0、1、3 和 4。AES 密钥需要格式 4。在验证或转换 PIN 码时，需要指定传入或传出数据的 PIN 码块。

POI

交互点 (POI) 也经常与销售点 (POS) 同义词使用，是持卡人与之交互以出示其支付凭证的硬件设备。POI 的一个示例是商家位置的实物终端。有关经过认证的 PCI PTS POI 终端列表，请访问 PCI 网站。

PSN

PAN 序列号 (PSN) 是一个数值，用于区分使用相同 PAN 发行的多张卡。

公有密钥

使用非对称密码 (RSA) 时，公有密钥是公有-私有密钥对的公有组成部分。加密详细信息介绍公有密钥可以共享并分发给需要为公有-私有密钥对拥有者加密数据的实体。对于数字签名操作，公有密钥用于验证签名。

私有密钥

使用非对称密码 (RSA) 时，私有密钥是公有-私有密钥对的私有组成部分。私有密钥用于解密数据或创建数字签名。与对称 AWS 支付密码学密钥类似，私钥由 HSM 安全创建。这些密钥仅在 HSM 的易失存储器中解密，并且仅在处理加密请求所需的时间内解密。

PVV

PIN 验证值 (PVV) 是通过算法从一系列输入（例如卡号和 PIN 码）中得出的值，它生成的值可用于后续验证。其中一种方案称为 Visa PVV（也称为 ABA 方法），尽管它可用于任何网络上的 PIN 码。

RSA Wrap/Unwrap

RSA wrap 使用非对称密钥来封装对称密钥（例如 TDES 密钥），以便传输到另一个系统。只有具有匹配私钥的系统才能解密有效负载并加载对称密钥。相反，RSA 解包将安全地解密使用 RSA 加密的密钥，然后将该密钥加载到支付密码中。 AWS RSA wrap 是一种交换密钥的低级方法，它不以密钥块格式传输密钥，也不使用发送方的有效载荷签名。应考虑使用其他控制措施来确定天意和关键属性不会发生变化。

TR-34 在内部也使用 RSA，但它是一种单独的格式，不可互操作。

TR-31

TR-31（正式定义为 ANSI X9 TR 31）是由美国国家标准协会 (ANSI) 定义的一种密钥块格式，用于支持在与密钥数据本身相同的数据结构中定义密钥属性。TR-31 密钥块格式定义了一组与密钥关联的按键属性，以便将它们组合在一起。 AWS Payment Cryptography 尽可能使用 TR-31 标准化术语来确保正确的密钥分离和密钥用途。TR-31 已被 ANSI X9.143-2022 所取代。

TR-34

TR-34 是 ANSI X9.24-2 的实现，它描述了一种使用非对称技术（例如 RSA）安全分发对称密钥（例如 3DES 和 AES）的协议。 AWS Payment Cryptography 使用 TR-34 方法来允许安全导入和导出密钥。

相关服务

AWS Key Management Service

AWS 密钥管理服务 (AWS KMS) 是一项托管服务，可让您轻松创建和控制用于保护您数据的加密密钥。AWSKMS 使用硬件安全模块（HSM）保护和验证您的 AWS KMS 密钥。

AWS CloudHSM

AWS CloudHSM 在 AWS 云中为客户提供专用的通用型 HSM 实例。 AWS CloudHSM 可以提供各种加密功能，例如创建密钥、数据签名或加密和解密数据。

有关更多信息

- 要了解 AWS Payment Cryptography 中使用的术语和概念，请参阅 AWS Payment Cryptography 概念。

- 有关 AWS Payment Cryptography 控制面板 API 的信息，请参阅 AWS Payment Cryptography 控制面板 API 参考。
- 有关 AWS Payment Cryptography 数据面板 API 的信息，请参阅 AWS Payment Cryptography 数据面板 API 参考。
- 有关 AWS Payment Cryptography 如何使用密码术并保护 AWS Payment Cryptography 密钥的详细技术信息，请参阅加密详细信息。

的终端节点 AWS Payment Cryptography

要以编程方式连接 AWS Payment Cryptography，您可以使用终端节点，即服务入口点的 URL。 AWS 软件开发工具包和命令行工具会 AWS 区域 根据请求的区域上下文自动使用服务的默认终端节点，因此通常无需显式设置这些值。必要时，您可以为 API 请求指定不同的终端节点。

控制面板端点

区域名称	区域	端点	协议
美国东部（弗吉尼亚州北部）	us-east-1	controlplane.payment-cryptography.us-east-1.amazonaws.com	HTTPS
美国东部（俄亥俄州）	us-east-2	控制飞机。payment-cryptography.us-east-2.amazonaws.com	HTTPS
US West（Oregon）	us-west-2	controlplane.payment-cryptography.us-west-2.amazonaws.com	HTTPS

数据面板端点

区域名称	区域	端点	协议
美国东部（弗吉尼亚州北部）	us-east-1	dataplane.payment-cryptography.us-east-1.amazonaws.com	HTTPS
美国东部（俄亥俄州）	us-east-2	dataplane。payment-cryptography.us-east-2.amazonaws.com	HTTPS

区域名称	区域	端点	协议
US West（Oregon）	`us-west-2`	dataplane.payment-cryptography.us-west-2.amazonaws.com	HTTPS

开始使用 AWS Payment Cryptography

要开始使用 AWS Payment Cryptography，您首先需要创建密钥，然后在各种加密操作中使用它们。下面的教程提供了一个简单的用例，生成用于生成/验证 CVV2 值的密钥。要尝试其他示例并探索 AWS 中的部署模式，请尝试以下 AWS Payment Cryptography Workshop，或探索 Github 上提供的示例项目

本教程将引导您创建单个密钥并使用该密钥执行加密操作。之后，如果您不再需要密钥，则可以将其删除，从而完成密钥的生命周期。

主题

先决条件

在您开始之前，请确保：

- 您有权访问该服务。有关更多信息，请参阅 IAM policy。
- 您已安装 AWS CLI。您也可以使用 AWSSDK 或 AWSAPI 来访问 AWS Payment Cryptography，但本教程中的说明将使用 AWS CLI。

步骤 1：创建密钥

第一步是创建一个密钥。在本教程中，您将创建一个 CVK 双长度 3DES (2KEY TDES) 密钥来生成和验证 CVV/CVV2 值。

```
$ 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}'
```

响应会回显请求参数，包括后续调用的 ARN 以及密钥检查值 (KCV)。

```
{
    "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"
    }
}
```

请注意代表密钥的 `KeyArn`，例如：arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi。您需要在下一步中执行该操作。

步骤 2：使用密钥生成 CVV2 值

在此步骤中，您将使用步骤 1 中的密钥生成给定 PAN 和到期日期的 CVV2。

```
$  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": "CADDA1",
    "CardDataGenerationKeyIdentifier": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/tqv5yij6wtxx64pi",
    "CardDataType": "CARD_VERIFICATION_VALUE_2",
    "CardDataValue": "144"
}
```

注意 cardDataValue，在本例中为 3 位数字 144。您需要在下一步中执行该操作。

步骤 3：验证已在步骤 2 中生成的值

在此示例中，您将使用在步骤 1 中创建的密钥验证步骤 2 中的 CVV2。

运行以下命令以验证 CVV2。

```
$  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": "CADDA1"
}
```

服务返回 200 的 HTTP 响应，表示它验证了 CVV2。

第 4 步：进行阴性测试

在此步骤中，您将创建一个阴性测试，其中 CVV2 不正确且无法验证。您尝试使用已在步骤 1 中创建的密钥验证不正确的 CVV2。例如，如果持卡人在结账时输入了错误的 CVV2，则这是预期的操作。

```
$  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.
```

该服务返回 400 的 HTTP 响应，消息为 “信用卡验证数据验证失败”，原因为 INVALID_VALIDATION_DATA。

第 5 步：（可选）清除

现在，您可以删除已在步骤 1 中创建的密钥。为最大限度地减少不可恢复的更改，默认密钥删除期为七天。

```
$ 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": "CADDA1",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "DELETE_PENDING",
        "UsageStartTimestamp": "2022-10-27T08:27:51.753000-07:00"
    }
}
```

请注意输出中的两个字段。默认情况下，deletePendingTimestamp 设置为未来的七天。keyState 设置为 DELETE_PENDING。您可以在预定删除时间之前的任何时间通过调用 restore-key 取消此删除。

管理密钥

要开始使用 AWS 支付加密，您需要创建一个 AWS 支付密码学密钥。

本节中的主题说明了如何创建和管理各种 AWS 支付密码学密钥类型，从创建到删除。其中包含以下主题：创建、编辑和查看密钥、标记密钥、创建密钥别名，以及启用和禁用密钥。

主题

- 生成密钥
- 列表密钥
- 启用和禁用密钥
- 删除密钥
- 导入和导出密钥
- 使用别名
- 获取密钥
- 标记密钥
- 了解 AWS 支付密码学密钥的关键属性

生成密钥

您可以使用 CreateKey API 操作创建 AWS 支付加密密钥。在此过程中，您将指定密钥或结果输出的各种属性，例如密钥算法（例如 TDES_3KEY）、（例如 TR31_P0_PIN_ENCRYPTION_KEY）、允许的操作 KeyUsage （例如加密、签名）以及它是否可导出。创建 AWS 付款加密密钥后，您无法更改这些属性。

示例

- 生成 2KEY TDES 密钥
- 生成 Pin 加密密钥
- 创建非对称 (RSA) 密钥
- 生成 PIN 验证值 (PVV) 密钥

生成 2KEY TDES 密钥

Example

此命令生成 2KEY TDES 密钥，用于生成和验证 CVV/CVV2 值。响应会回显请求参数，包括后续调用的 ARN 以及 KCV（密钥检查值）。

```
$ 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/
hjprdg5o4jtgs5tw",
        "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"
    }
```

```
}
```

生成 Pin 加密密钥

Example 生成 Pin 加密密钥(PEK)

此命令生成 3KEY TDES 密钥，用于加密 PIN 值（称为 Pin 加密密钥）。此密钥可用于保护 PIN 的安全存储或用于解密在验证尝试期间（例如交易期间）提供的 PIN。响应会回显请求参数，包括后续调用的 ARN 以及 KCV（密钥检查值）。

```
$ 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"
    }
}
```

创建非对称 (RSA) 密钥

Example

在此示例中，我们将生成一个新的非对称 RSA 2048 位密钥对。将生成新的私钥以及匹配的公钥。可以使用 ge t PublicCertificate API 检索公钥。

```
$ 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"
    }
}
```

生成 PIN 验证值 (PVV) 密钥

Example

此命令生成 3KEY TDES 密钥，用于生成 PVV 值（称为 Pin 验证值）。您可以使用此密钥生成 PVV 值，该值可以与随后计算的 PVV 进行比较。响应会回显请求参数，包括后续调用的 ARN 以及 KCV（密钥检查值）。

```
$ 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"
    }
}
```

列表密钥

列表密钥显示了该账户和区域的调用者可以访问的密钥列表。

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"
    }
    ]
}
```

启用和禁用密钥

您可以禁用和重新启用 AWS 支付加密密钥。密钥在创建完成后默认处于启用状态。如果禁用某个密钥，则在您重新启用它之前，它不能用于任何加密操作。开始/停止使用命令会立即生效，因此建议您在进行此类更改之前先查看使用情况。您也可以使用可选 `timestamp` 参数，将更改（开始或停止使用）设置为在将来生效。

由于付款加密密钥是临时的且易于撤消，因此禁用 AWS 支付加密密钥是比删除 AWS 付款加密密钥更安全的替代方法，后者具有破坏性和不可逆转性。如果您正在考虑删除 AWS 付款加密密钥，请先将其禁用，并确保将来无需使用该密钥来加密或解密数据。

主题

- 开始密钥使用
- 停止密钥使用

开始密钥使用

必须启用密钥使用才能使用密钥进行加密操作。如果密钥未启用，则可以通过此操作使其可用。字段 `UsageStartTimestamp` 将表示密钥何时变为/将变为活动状态。对于已启用的令牌，这将是过去，如果令牌待激活，则是将来。

Example

在此示例中，要求启用密钥以使用密钥。响应中包含关键信息，并且启用标志已转换为 true。这也将反映在列表密钥响应对象中。

```
$ aws payment-cryptography start-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": true,
      "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",
      "UsageStartTimestamp": "2022-10-27T14:09:59.468000-07:00"
  }
}
```


停止密钥使用

如果您不再打算使用密钥，则可以停止使用密钥以防止进一步的加密操作。此操作不是永久性的，因此您可以通过开始密钥使用来撤消该操作。您也可以将密钥设置为将来禁用。字段 UsageStopTimestamp 将表示密钥何时变为/将变为禁用状态。

Example

在此示例中，要求在将来停止使用密钥。执行后，除非通过开始密钥使用重新启用，否则该密钥不能用于加密操作。响应包含密钥信息，并且启用标志已转换为 false。这也将反映在列表密钥响应对象中。

```
$ 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"
  }
}
```

删除密钥

删除 AWS 支付加密密钥会删除密钥材料和与该密钥关联的所有元数据，除非在 AWS 支付密码学之外有该密钥的副本，否则该密钥的副本不可撤销。删除密钥后，您不能再解密用该密钥加密的数据，这意味着该数据将无法恢复。只有当您确定不再需要使用密钥且其他任何当事方不在使用此密钥时，才能将其删除。如果您不确定，请考虑禁用密钥，而不是将其删除。如果您以后需要再次使用已禁用的密钥，则可以重新启用该密钥，但是除非您可以从其他来源重新导入已删除的 AWS 付款加密密钥，否则无法恢复该密钥。

删除密钥之前，应确保不再需要该密钥。 AWS Payment Cryptography 不存储像 CVV2 这样的加密操作的结果，也无法确定任何永久加密材料是否需要密钥。

AWS Payment Cryptography 永远不会删除属于活跃 AWS 账户的密钥，除非您明确安排将其删除，并且强制等待期已到期。

但是，出于以下一个或多个原因，您可能会选择删除 AWS 付款加密密钥：

- 完成不再需要的密钥的密钥生命周期
- 为了避免与维护未使用的 AWS 支付加密密钥相关的管理开销

> **Note**
>
> 如果您关闭或删除您的 AWS 账户，则无法 AWS 访问您的付款加密密钥。在关闭账户的同时，您无需安排删除 AWS 支付密码密钥。

AWS Payment Cryptography 会在您计划删除 AWS 支付密码密钥以及实际删除支付密码密钥时在 AWS 您的AWS CloudTrail日志中记录一个条目。

关于等待期限

由于删除密钥是不可逆的，因此 AWS 支付密码学要求您将等待期设置为 3 到 180 天之间。默认的等待期限为七天。

但是，实际等待期限可能最多比您计划的时间长 24 小时。要获取删除 AWS 付款加密密钥的实际日期和时间，请使用 GetKey 操作。请务必记下时区。

在等待期间， AWS 付款加密密钥状态和密钥状态为 “待删除”。

> **Note**
>
> 待删除的 AWS 支付加密密钥不能用于任何加密操作。

等待期结束后， AWS Payment Cryptography 会删除 AWS 支付加密密钥、其别名和所有相关的 AWS 支付密码学元数据。

使用等待期来确保你现在或将来都不需要 AWS 支付密码学密钥。如果您在等待期内发现确实需要密钥，可以在等待期限结束前取消密钥删除。等待期限结束后，将无法取消密钥删除， 将删除密钥。

Example

在此示例中，请求删除密钥。除了基本的密钥信息外，还有两个相关的字段是密钥状态已更改为 DELETE_PENDING，以及 deletePendingTimestamp表示当前计划删除密钥的时间。

```
$ aws payment-cryptography delete-key \
                    --key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h
```

```
  {
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
        "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

在此示例中，待处理的删除被取消。成功完成后，密钥将不再按照之前的时间表被删除。响应包含基本的密钥信息；此外，两个相关字段已更改 - KeyState 和 deletePendingTimestamp。KeyState 返回到 CREATE_COMPLETE 的值，同时 DeletePendingTimestamp 被移除。

```
$ aws payment-cryptography restore-key --key-identifier arn:aws:payment-
cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h
```

```
{
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
        "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"
    }
}
```

导入和导出密钥

AWS 支付加密密钥可以从其他解决方案导入，也可以导出到其他解决方案（例如其他 HSM）。使用导入和导出功能与服务提供商交换密钥是一种常见的用例。作为一项云服务，P AWS ayment Cryptography 采用现代化的电子方法进行密钥管理，同时帮助您保持适用的合规性和控制性。长期目标是摆脱纸质密钥组件，转向基于标准的电子手段密钥交换。

密钥加密密钥 (KEK) 交换

AWS 支付密码学鼓励使用公钥加密 (RSA) 进行初始密钥交换，使用既定的 ANSI X9.24 TR-34 规范。这种初始密钥类型的常用名称包括密钥加密密钥 (KEK)、区域主密钥 (ZMK) 和区域控制主密钥 (ZCMK)。如果您的系统或合作伙伴尚无法支持 TR-34，您也可以考虑使用 RSA Wrap /Unwrap。

如果您需要继续处理纸质密钥组件，直到所有合作伙伴都支持电子密钥交换，则可以考虑为此目的保留离线 HSM。

> **Note**
>
> 如果您想导入自己的测试密钥，请在 Github 上查看示例项目。要了解如何从其他平台导入/导出密钥的说明，请参阅这些平台的用户指南。

工作密钥 (WK) 交换

AWS 支付密码学使用相关的行业规范（ANSI X9.24 TR 31-2018）来交换工作密钥。TR-31 假设先前已交换过 KEK。这符合 PCI PIN 的要求，即始终以加密方式将密钥材料绑定到其密钥类型和用法。工作密钥有不同的名称，包括收单机构工作密钥、发行者工作密钥、BDK、IPEK 等。

主题

导入密钥

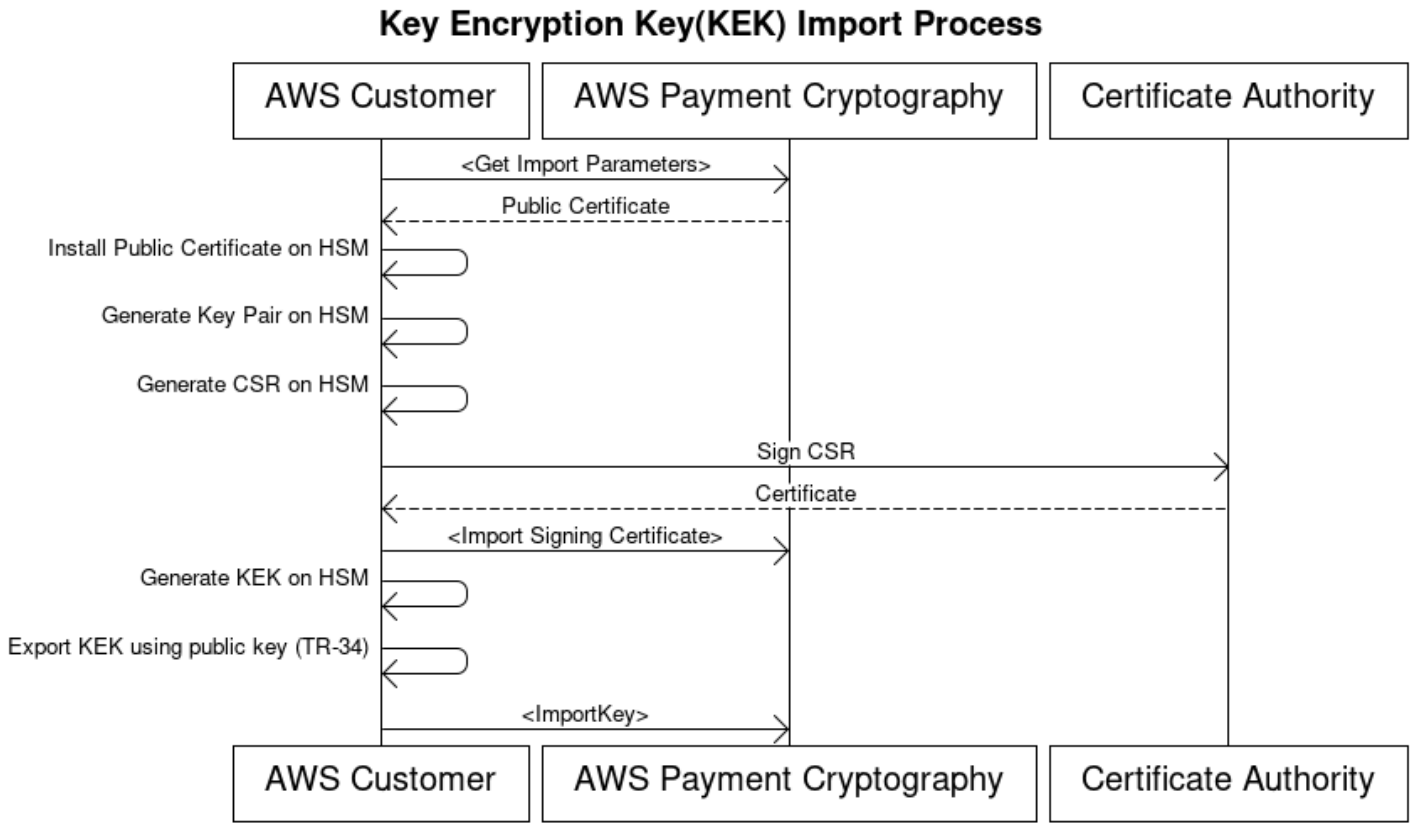
> **Important**
>
> 示例可能需要最新版本的 AWS CLI V2。在开始之前，请确保您已升级到最新版本。

主题

导入对称密钥

主题

使用非对称技术导入密钥 (TR-34)



概述：TR-34 利用 RSA 非对称加密来加密对称密钥以进行交换，并确保数据来源（签名）。这样可以确保封装密钥的机密性（加密）和完整性（签名）。

如果您想导入自己的密钥，请查看 Github 上的示例项目。要了解如何从其他平台导入/导出密钥的说明，请参阅这些平台的用户指南。

1. 调用初始化导入命令

   调用 `get-parameters-for-import` 以初始化导入过程。此 API 将生成用于密钥导入的密钥对，签署密钥并返回证书和证书根。最终，应使用此密钥对要导出的密钥进行加密。在 TR-34 术语中，这被称为 KRD 证书。请注意，这些证书的有效期很短，仅用于此目的。

2. 在密钥源系统上安装公共证书

   对于许多 HSM，您可能需要安装/加载/信任步骤 1 中生成的公共证书，才能使用它导出密钥。

3. 生成公钥并为 AWS 支付密码学提供证书根

   为了确保传输的有效载荷的完整性，它由发送方（称为密钥分发主机或 KDH）签名。发送方需要为此生成一个公钥，然后创建一个可以提供给 AWS 支付密码学的公钥证书 (X509)。 AWS Private CA 是生成证书的一种选择，但对使用的证书颁发机构没有限制。

获得证书后，您需要使用importKey命令和 KeyMaterialType of 和 of 和 of 将根证书加载到 Paym AWS ent Cryptograph ROOT_PUBLIC_KEY_CERTIFICATE y KeyUsageType 中。TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE

4. 从源系统导出密钥

许多 HSM 和相关系统都支持使用 TR-34 标准导出密钥的功能。您需要将步骤 1 中的公钥指定为 KRD（加密）证书，并将步骤 3 中的密钥指定为 KDH（签名）证书。要导入到 P AWS ayment Cryptography，您需要将格式指定为 TR-34.2012 非 CMS 双通格式，也可以称为 TR-34 Diebold 格式。

5. 调用导入密钥

作为最后一步，您将调用 ImportKey API，并设置为 KeyMaterialType 。TR34_KEY_BLOCKcertificate-authority-public-key-identifier 将是步骤 3 中导入的根 CA 的 KeyArn，key-material 将是步骤 4 中的封装密钥材料，signing-key-certificate 是步骤 3 中的叶证书。您还需要提供步骤 1 中的导入令牌。

6. 使用导入的密钥进行加密操作或后续导入

如果导入 KeyUsage 的是 TR31_K0_KEY_ENCRYPTION_KEY，则此密钥可用于后续使用 TR-31 导入密钥。如果密钥类型是任何其他类型（例如 TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY），则该密钥可以直接用于加密操作。

使用非对称技术导入密钥（RSA Unwrap）

概述：当 TR-34 不可行时， AWS 支付密码学支持 RSA 封装/解包进行密钥交换。与 TR-34 类似，该技术利用 RSA 非对称加密来加密对称密钥以进行交换。但是，与 TR-34 不同，此方法没有发送方对有效载荷进行签名。此外，由于不包括密钥块，这种 RSA 封装技术无法在传输过程中保持密钥元数据的完整性。

> **Note**
>
> RSA 封装可用于导入或导出 TDES 和 AES-128 密钥。

1. 调用初始化导入命令

调get-parameters-for-import用使用 KEY_CRYPTOGRAM 的密钥材料类型初始化导入过程。 WrappingKeyAlgorithm 交换 TDES 密钥时可以是 RSA_2048。交换 TDES 或 AES-128 密钥时可以使用 RSA_3072 或 RSA_4096。此 API 将生成用于密钥导入的密钥对，使用证书根对密钥

进行签名，然后返回证书和证书根。最终，应使用此密钥对要导出的密钥进行加密。请注意，这些证书的有效期很短，仅用于此目的。

```
$ 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. 在密钥源系统上安装公共证书

   对于许多 HSM，您可能需要安装/加载/信任步骤 1 中生成的公共证书（和/或其根证书），才能使用它导出密钥。

3. 从源系统导出密钥

   许多 HSM 和相关系统都支持使用 RSA 封装导出密钥的功能。您需要将步骤 1 中的公钥指定为（加密）证书（WrappingKey证书）。如果你需要信任链，它包含在步骤 #1 的响应字段 WrappingKeyCertificateChain 中。从 HSM 导出密钥时，你需要将格式指定为 RSA，填充模式 = PKCS #1 v2.2 OAEP（使用 SHA 256 或 SHA 512）。

4. 调用导入密钥

   作为最后一步，您将调用 ImportKey API，并设置为 KeyMaterialType 。`KeyMaterial`您将需要步骤 1 中的导入令牌和步骤 3 中的`key-material`（封装密钥材料）。由于 RSA wrap 不使用密钥块，因此您需要提供密钥参数（例如密钥用法）。

```
$ cat import-key-cryptogram.json
              {
                  "KeyMaterial": {
                      "KeyCryptogram": {
                          "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"
                        },
                        "WrappedKeyCryptogram": "18874746731....",
                        "WrappingSpec": "RSA_OAEP_SHA_256"
                    }
                }
            }
```

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

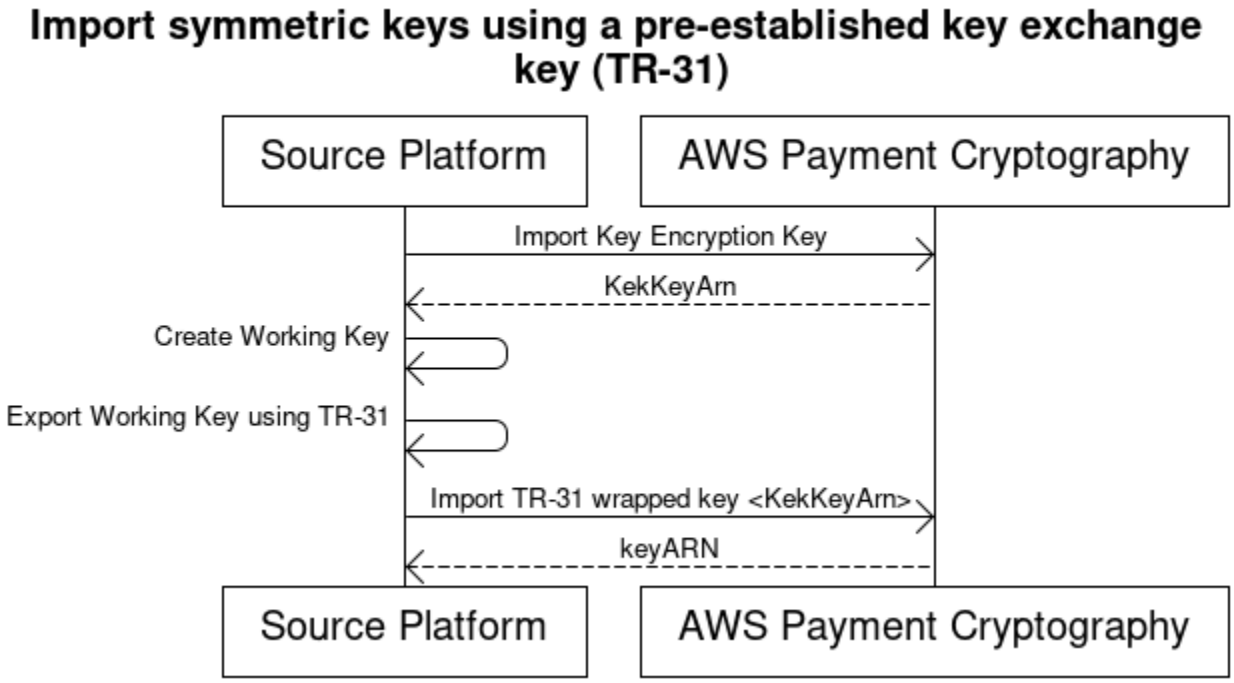
```
{
      "Key": {
          "KeyOrigin": "EXTERNAL",
          "Exportable": true,
          "KeyCheckValue": "DA1ACF",
          "UsageStartTimestamp": 1697643478.92,
          "Enabled": true,
          "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
          "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. 使用导入的密钥进行加密操作或后续导入

   如果导入 KeyUsage 的是 TR31_K0_KEY_ENCRYPTION_KEY，则此密钥可用于后续使用 TR-31 导入密钥。如果密钥类型是任何其他类型（例如 TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY），则该密钥可以直接用于加密操作。

使用预先建立的密钥交换密钥导入对称密钥 (TR-31)



当合作伙伴交换多个密钥（或支持密钥轮换）时，通常首先使用诸如纸质密钥组件之类的技术交换初始密钥加密密钥 (KEK)，或者在 AWS 支付密码学中使用 TR-34 交换初始密钥加密密钥 (KEK)。

建立 KEK 后，您可以使用此密钥传输后续密钥（包括其他 KEK）。 AWS Payment Cryptography 使用 ANSI TR-31 支持这种密钥交换，HSM 供应商广泛使用并广泛支持。

1. 导入密钥加密密钥 (KEK)

   假设您已经导入了 KEK 并且有 KeyArn（或 keyAlias）可供使用。

2. 在源平台上创建密钥

   如果密钥尚不存在，请在源平台上创建密钥。相反，您可以在 AWS Payment Cryptography 上创建密钥并改用 `export` 命令。

3. 从源平台导出密钥

   导出时，请确保将导出格式指定为 TR-31。源平台还将要求您提供要导出的密钥和要使用的密钥加密密钥。

4. 导入 AWS 支付密码学

   调用 importKey 命令时， WrappingKeyIdentifier 应为密钥加密密钥的 KeyArn（或别名），并且 WrappedKeyBlock 是源平台的输出。

   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/
kwapwa6qaifllw2h",
            "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"
    }
}
```

导入非对称 (RSA) 密钥

导入 RSA 公钥

AWS 支付密码学支持以 X.509 证书的形式导入 RSA 公钥。要导入证书，您需要先导入其根证书。导入时，所有证书应均未过期。证书应采用 PEM 格式，并应采用 base64 编码。

1. 导入根证书到 AWS 支付密码学

   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":"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURKVENDQWcyZ0F3SUJBZ0lCWkR
```

```
  {
    "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. 将公钥证书导入 AWS 支付密码学

现在，您可以导入公钥。有两种导入公钥的选项。如果密钥的目的是验证签名（例如，使用 TR-34 导入时），则可以使用 TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE。当加密数据要与其他系统一起使用时，可以使用 TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION。

Example

```
$ aws payment-cryptography import-key \
    --key-material='{"TrustedCertificatePublicKey":
{"CertificateAuthorityPublicKeyIdentifier":"arn:aws:payment-cryptography:us-
east-2:111122223333:key/zabouwe3574jysdl", \
    "KeyAttributes":
{"KeyAlgorithm":"RSA_2048","KeyClass":"PUBLIC_KEY","KeyModesOfUse":
{"Verify":true},"KeyUsage":"TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"},\
    "PublicKeyCertificate":"LS0tLS1CRUdJTiB..."}}'
```

```
{
  "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"
  }
}
```

导出密钥

主题

- 仅导出对称密钥。
- 导出非对称 (RSA) 密钥

仅导出对称密钥。

> **Important**
>
> 示例可能需要最新版本的 AWS CLI V2。在开始之前，请确保您已升级到最新版本。

主题

- 使用非对称技术导出密钥(TR-34)
- 使用非对称技术（RSA Wrap）导出密钥
- 使用预先建立的密钥交换密钥导出对称密钥 (TR-31)
- 导出 DUKPT 初始密钥 (IPEK/IK)

使用非对称技术导出密钥(TR-34)

概述：TR-34 利用 RSA 非对称加密来加密对称密钥以进行交换，并确保数据来源（签名）。这样可以确保封装密钥的机密性（加密）和完整性（签名）。导出时，P AWS ayment Cryptography 成为密钥分发主机 (KDH)，目标系统成为密钥接收设备 (KRD)。

1. 调用初始化导出命令

   调用 `get-parameters-for-export` 以初始化导出过程。此 API 将生成用于密钥导出的密钥对，签署密钥并返回证书和证书根。最终，此命令生成的私钥用于对导出有效载荷进行签名。在 TR-34 术语中，这被称为 KDH 签名证书。请注意，这些证书的有效期很短，仅用于此目的。参数 ParametersValidUntilTimestamp指定它们的持续时间。

   注意：所有证书均以 base64 编码格式返回

   Example

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

```
{
               "SigningKeyCertificate":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUV2RENDQXFTZ0F3SUJBZ0lRZFAzSzNHNEFKT0I4WTNpTmUvYl
               "SigningKeyCertificateChain":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUY0VENDQThtZ0F3SUJBZ0lSQUtlN2piaHFKZjJPd3FGUWI5c3
               "SigningKeyAlgorithm": "RSA_2048",
               "ExportToken": "export-token-au7pvkbsq4mbup6i",
               "ParametersValidUntilTimestamp": "2023-06-13T15:40:24.036000-07:00"
}
```

2. 将 AWS 支付密码证书导入接收系统

   必要时，将步骤 1 中提供的证书链导入您的接收系统。

3. 生成 key pair，创建公共证书并将证书根提供给 Payment Cryptogr AWS aphy

为确保传输的有效载荷的机密性，发送方（称为密钥分配主机或 KDH）对其进行加密。接收方（通常是您的 HSM 或合作伙伴的 HSM）需要为此生成一个公钥，然后创建一个可以提供给支付密码学的公钥证书 (x.509)。 AWS AWS Private CA 是生成证书的一种选择，但对使用的证书颁发机构没有限制。

获得证书后，您需要使用`ImportKey`命令和 KeyMaterialType of 和 of 和 of 将根证书加载到 Paym AWS ent Cryptograph `ROOT_PUBLIC_KEY_CERTIFICATE` y KeyUsageType 中。`TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`

此证书 KeyUsageType 的证书是 TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE，因为它是根密钥，用于对叶证书进行签名。用于导入/导出的树叶证书不会导入到 AWS 支付密码学中，而是以内联方式传递。

> **Note**
>
> 如果之前导入了根证书，则可以跳过此步骤。

4. 调用导出密钥

作为最后一步，您将使用调用 ExportKey API `TR34_KEY_BLOCK`。 KeyMaterialType `certificate-authority-public-key-identifier` 将是步骤 3 中导入的根 CA 的 KeyArn，`WrappingKeyCertificate` 将是步骤 3 中的叶证书，`export-key-identifier` 是要导出的 KeyArn（或别名）。您还需要提供步骤 1 中的导出令牌。

使用非对称技术（RSA Wrap）导出密钥

概述：当交易方不提供 TR-34 选项时， AWS 支付密码学支持 RSA 封装/解包进行密钥交换。与 TR-34 类似，该技术利用 RSA 非对称加密来加密对称密钥以进行交换。但是，与 TR-34 不同，此方法没有发送方对有效载荷进行签名。此外，这种 RSA 封装技术不包括用于在传输过程中维护密钥元数据完整性的密钥块。

> **Note**
>
> RSA 封装可用于导出 TDES 和 AES-128 密钥。

1. 在接收系统上生成 RSA 密钥和证书

   创建（或识别）用于接收封装密钥的 RSA 密钥。 AWS 支付密码学需要采用 X.509 证书格式的密钥。证书应由已导入（或可以导入）到 AWS 支付密码学的根证书签名。

2. 在 AWS 支付密码学上安装根公共证书

```
$ 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. 呼叫导出密钥

接下来，你要指示 Payment AWS Cryptography 使用你的叶子证书导出密钥。您将为先前导入的根证书指定 ARN、用于导出的叶证书和要导出的对称密钥。输出将是对称密钥的十六进制编码二进制包装（加密）版本。

```
$ 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/zabouwe3574jysdl",
              "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. 将密钥导入接收系统

许多 HSM 和相关系统都支持使用 RSA unwrap（包括 AWS 支付加密）导入密钥。为此，请将步骤 1 中的公钥指定为（加密）证书。格式应指定为 RSA，填充模式 = PKCS #1 v2.2 OAEP（使用 SHA 256）。确切的术语可能因 HSM 而异。

> **Note**
>
> AWS 支付密码学在 HexBinary 中输出封装后的密钥。如果您的系统需要不同的二进制表示形式（如 base64），则可能需要在导入之前转换格式。

使用预先建立的密钥交换密钥导出对称密钥 (TR-31)

当合作伙伴交换多个密钥（或支持密钥轮换）时，通常首先使用诸如纸质密钥组件之类的技术交换初始密钥加密密钥 (KEK)，或者在 AWS 支付密码学中使用 TR-34 交换初始密钥加密密钥 (KEK)。建立 KEK 后，您可以使用此密钥传输后续密钥（包括其他 KEK）。 AWS Payment Cryptography 使用 ANSI TR-31 支持这种密钥交换，HSM 供应商广泛使用并广泛支持。

1. 交换密钥加密密钥 (KEK)

   假设您已经交换了 KEK 并且有 KeyArn（或 KeyAlias）可供使用。

2. 在 AWS 支付密码学上创建密钥

   如果该密钥尚不存在，请创建它。相反，您可以在其他系统上创建密钥并改用 import 命令。

3. 从 AWS 支付密码学中导出密钥

   导出时，格式将为 TR-31。在调用 API 时，您将指定要导出的密钥和要使用的封装密钥。

```
$ 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/5rplquuwozodpwsp
```

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

4. 导入到您的系统

   您或您的合作伙伴将使用系统上的导入密钥实现来导入密钥。

导出 DUKPT 初始密钥 (IPEK/IK)

使用 DUKPT 时，可以为一组终端生成单个基本派生密钥 (BDK)。但是，终端永远无法访问原始的 BDK，但每个终端都注入了一个唯一的初始终端密钥，称为 IPEK 或初始密钥 (IK)。每个 IPEK 都是源自 BDK 的密钥，旨在每个终端都是唯一的，但源自原始 BDK。此计算的派生数据称为密钥序列号 (KSN)。根据 X9.24，对于 TDES，10 字节的 KSN 通常由密钥集 ID 的 24 位、终端 ID 的 19 位和事务计数器的 21 位组成。对于 AES，12 字节的 KSN 通常由 BDK ID 的 32 位、派生标识符 (ID) 的 32 位和事务计数器的 32 位组成。

AWS 支付密码学提供了一种生成和导出这些初始密钥的机制。生成这些密钥后，可以使用 TR-31、TR-34 和 RSA 封装方法导出。IPEK 密钥不会永久保存，也不能用于支付密码学的后续操作 AWS

AWS 支付密码学不强制在KSN的前两部分之间进行分割。如果您希望将派生标识符与 BDK 一起存储，则可以使用 AWS 标签功能来实现此目的。

> **Note**
>
> KSN 的计数器部分（AES DUKPT 为 32 位）不用于 IPEK/IK 推导。因此，输入 12345678901234560001 和 12345678901234560999 将输出相同的 IPEK。

```
$ 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"
    }
}
```

导出非对称 (RSA) 密钥

调用 `get-public-key-certificate` 以导出证书形式的公钥。此 API 将导出证书及其以 base64 格式编码的根证书。

注意：此 API 不是幂等的，即使底层密钥相同，后续调用也可能会生成不同的证书。

Example

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

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

使用别名

别名是 AWS 支付密码学密钥的友好名称。例如，别名允许您将密钥引用为 `alias/test-key`，而不是 `arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h`。

在大多数密钥管理（控制面板）操作和加密（数据面板）操作中，您可以使用别名来标识密钥。

您还可以根据其别名允许和拒绝对 AWS 支付密码密钥的访问，而无需编辑政策或管理授权。此功能是基于属性的访问权限控制 (ABAC) 的支持的一部分。

别名的大部分功能来自于您随时更改与别名关联的密钥的能力。别名可以使您的代码更易于编写和维护。例如，假设您使用别名来指代特定的 AWS 支付加密密钥，并且想要更改 AWS 付款加密密钥。在这种情况下，只需将别名与其他密钥关联即可。您无需更改代码或应用程序配置。

别名还您更容易在不同 AWS 区域中重用相同代码。在多个区域中创建同名别名，并将每个别名与其所在地区的 AWS 支付加密密钥相关联。当代码在每个区域运行时，别名是指该区域中关联的 AWS 支付加密密钥。

您可以使用 `CreateAlias` API 为 AWS 支付加密密钥创建别名。

AWS 支付密码学 API 可以完全控制每个账户和地区的别名。API 包括创建别名 (CreateAlias)、查看别名和链接的 KeyArn（列表别名）、更改与别名关联的 AWS 支付加密密钥（更新别名）以及删除别名（删除别名）的操作。

主题

关于别名

了解别名在 AWS 支付密码学中的工作原理。

别名是一种独立的 AWS 资源

别名不是 AWS 支付密码学密钥的属性。您对别名执行的操作不会影响其关联的密钥。您可以为 AWS 支付加密密钥创建别名，然后更新别名，使其与不同的 AWS 支付加密密钥相关联。您甚至可以删除别名，而不会对关联的 AWS 支付加密密钥产生任何影响。如果您删除 AWS Payment Cryptography 密钥，则会取消分配与该密钥关联的所有别名。

如果您在 IAM 策略中将别名指定为资源，则该策略指的是别名，而不是关联的 AWS 支付加密密钥。

每个别名都有友好名称

在创建别名时，您指定前缀为 `alias/` 的别名。例如 `alias/test_1234`

每个别名一次都与一个 AWS 支付密码密钥相关联

别名及其 AWS 支付密码密钥必须位于同一个账户和地区中。

一个 AWS 支付密码学密钥可以同时与多个别名关联，但每个别名只能映射到一个密钥

例如，此 `list-aliases`输出显示`alias/sampleAlias1` 别名仅与一个目标 AWS Payment Cryptography 密钥相关联，该密钥由 `KeyArn` 属性表示。

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

```
  {
    "Aliases": [
        {
            "AliasName": "alias/sampleAlias1",
            "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
        }
    ]
}
```

多个别名可以与同一个 AWS 支付密码密钥相关联

例如，您可以将 alias/sampleAlias1; 和 alias/sampleAlias2 别名与同一个密钥关联。

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

```
{
        "Aliases": [
            {
                "AliasName": "alias/sampleAlias1",
                "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
            },
            {
                "AliasName": "alias/sampleAlias2",
                "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
            }
        ]
    }
```

别名在给定的账户和区域中必须是唯一的

例如，您在每个账户和区域中只能有一个 alias/sampleAlias1 别名。别名区分大小写，但我们建议不要使用仅大小写不同的别名，因为这样很容易出错。您不能更改别名名称。但是，您可以删除别名并使用所需名称创建新别名。

您可以在不同的区域中创建具有相同名称的别名。

例如，您可以在美国东部（弗吉尼亚州北部）拥有 alias/sampleAlias2 别名，在美国西部（俄勒冈州）拥有 alias/sampleAlias2 别名。每个别名都将与其所在地区的 AWS 支付加密密钥相

关联。如果您的代码引用 alias/finance-key 之类的别名名称，您可以在多个区域中运行它。在每个区域中，它使用不同的别名 /sampleAlias2。有关更多信息，请参阅 在应用程序中使用别名。

您可以更改与别名关联的 AWS 支付加密密钥

您可以使用该UpdateAlias操作将别名与不同的 AWS 支付加密密钥相关联。例如，如果alias/sampleAlias2别名与arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h AWS 支付加密密钥相关联，则可以对其进行更新，使其与arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi密钥关联。

> **Warning**
>
> AWS Payment Cryptography 无法验证新旧密钥是否具有所有相同的属性，例如密钥用法。使用不同的密钥类型进行更新可能会导致应用程序出现问题。

有些密钥没有别名

别名是一项可选功能，除非您选择以这种方式操作环境，否则并非所有密钥都有别名。可以使用 create-alias 命令将密钥与别名相关联。此外，您还可以使用 UpdateAlias 操作来更改与别名关联的 AWS Payment Cryptography 密钥，并使用 delete-alias 操作来删除别名。因此，某些 AWS 支付密码学密钥可能有多个别名，而有些可能没有别名。

将密钥映射到别名

您可以使用 create-alias 命令将密钥（由 ARN 表示）映射到一个或多个别名。此命令不是幂等的，要更新别名，请使用 update-alias 命令。

```
$ aws payment-cryptography create-alias --alias-name alias/sampleAlias1 \
             --key-arn arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h
```

```
{
    "Alias": {
        "AliasName": "alias/alias/sampleAlias1",
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
    }
}
```

在应用程序中使用别名

您可以使用别名来表示应用程序代码中的 AWS 支付加密密钥。 AWS 支付密码学数据操作以及其他操作（如列表密钥）中的key-identifier参数接受别名或别名 ARN。

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

使用别名 ARN 时，请记住，映射到 AWS 支付加密密钥的别名是在拥有 AWS 支付密码密钥的账户中定义的，每个区域可能有所不同。

别名的最强大用途之一是在多个 AWS 区域中运行的应用程序中。

您可以在每个区域创建不同版本的应用程序，也可以使用字典、配置或切换语句为每个区域选择正确的 AWS 支付密码密钥。但在每个区域中创建具有相同别名名称的别名可能要容易得多。请记住，别名名称区分大小写。

相关 API

标签

> 标签是密钥和值对，它们充当元数据，用于组织您的 AWS 支付加密密钥。它们可用于灵活识别密钥，或将一个或多个密钥组合在一起。

获取密钥

P AWS ayment Cryptography 密钥代表加密材料的单个单元，只能用于此服务的加密操作。 GetKeys API 将 KeyIdentifier 作为输入并返回密钥的不可变和可变属性，但不包含任何加密材料。

Example

```
$ aws payment-cryptography get-key --key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h
```

```
{
  "Key": {
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
      "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"
  }
}
```

获取与密钥对关联的公钥/证书

获取公钥/证书会返回 KeyArn 所指示的公钥。这可以是在 P AWS ayment Cryptography 上生成的密钥对的公钥部分，也可以是之前导入的公钥。最常见的用例是向将加密数据的外部服务提供公钥。然后，这些数据可以传递到利用 AWS 支付密码学的应用程序，并且可以使用支付密码学中保护的私钥对数据进行解密。 AWS

该服务返回公钥作为公共证书。API 结果包含 CA 和公钥证书。两个数据元素均采用 base64 编码。

> **Note**
>
> 返回的公共证书是短暂的，而不是幂等的。即使公钥本身未更改，您也可能会在每次 API 调用中收到不同的证书。

Example

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


```
{
  "KeyCertificate":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUV2VENDQXFXZ0F3SUJBZ0lSQUo1OWd2VkpDd3dlYldMNldYZEpYY
  "KeyCertificateChain":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUY0VENDQThtZ0F3SUJBZ0lSQUtlN2piaHFKZjJPd3FGUWI5c3VuO
}
```


标记密钥

在 P AWS ayment Cryptography 中，您可以在创建密钥时为 AWS 支付加密密钥添加标签，也可以标记或取消标记现有密钥，除非这些密钥待删除。标签是可选的，但它们可能非常有用。

有关标签的一般信息，包括最佳做法、标记策略以及标签的格式和语法，请参阅中的标记 AWS 资源。Amazon Web Services 一般参考

主题

- 关于 AWS 支付密码学中的标签
- 在控制台中查看密钥标签
- 使用 API 操作管理密钥标签
- 控制对标签的访问
- 使用标签控制对密钥的访问

关于 AWS 支付密码学中的标签

标签是您可以为 AWS 资源分配（或 AWS 可以分配）的可选元数据标签。每个标签都包含一个标签键和一个标签值，它们都是区分大小写的字符串。标签值可为空 (null) 字符串。资源上的每个标签必须具有不同的标签密钥，但您可以为多个 AWS 资源添加相同的标签。每个资源最多可以有 50 个用户创建的标签。

不要在标签键或标签值中包含机密或敏感信息。许多人都可以访问标签 AWS 服务，包括账单。

在 P AWS ayment Cryptography 中，您可以在创建密钥时为密钥添加标签，也可以标记或取消标记现有密钥，除非这些密钥待删除。无法标记别名。标签是可选的，但它们可能非常有用。

例如，您可以为用于 Alpha 项目的所有 AWS 付款加密密钥和 Amazon S3 存储桶添加`"Project"="Alpha"`标签。另一个例子是为与特定银行识别码 (BIN) 关联的所有密钥添加 `"BIN"="20130622"` 标签。

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

有关标签的一般信息，包括格式和语法，请参阅中的Amazon Web Services 一般参考标记 AWS 资源。

标签可帮助您：

- 识别和整理您的 AWS 资源。许多 AWS 服务都支持标记，因此您可以为来自不同服务的资源分配相同的标签，以表明这些资源是相关的。例如，您可以为 AWS 支付加密密钥和亚马逊弹性区块存储 (Amazon EBS) 卷或密钥分配相同的标签。 AWS Secrets Manager 您还可以使用标签来标识密钥以实现自动化。
- 追踪您的 AWS 成本。向 AWS 资源添加标签时， AWS 会生成一份成本分配报告，其中包含按标签汇总的使用量和成本。您可以使用此功能来跟踪项目、应用程序或成本中心的 AWS 支付加密成本。

有关对成本分配使用标签的更多信息，请参阅 AWS Billing 用户指南中的使用成本分配标签。有关适用于标签键和标签值的规则的信息，请参阅 AWS Billing 用户指南中的用户定义的标签限制。

- 控制对 AWS 资源的访问权限。根据密钥的标签允许和拒绝访问密钥是 AWS 支付密码学对基于属性的访问控制 (ABAC) 的支持的一部分。有关基于 AWS Payment Cryptography 的标签控制对其访问的更多信息，请参阅 基于 AWS Payment Cryptography 标签的授权。有关使用标签控制 AWS 资源访问权限的更多一般信息，请参阅 IAM 用户指南中的使用 AWS 资源标签控制对资源的访问权限。

AWS 当您使用 TagResource、 UntagResource或 ListTagsForResource 操作时，Payment Cryptography 会在您的 AWS CloudTrail 日志中写入一个条目。

在控制台中查看密钥标签

要在控制台中查看标签，您需要在包含该密钥的 IAM policy 中拥有密钥的标记权限。除了在控制台中查看密钥的权限之外，您还需要这些权限。

使用 API 操作管理密钥标签

您可以使用 AWS Payment Cryptography API 为您管理的密钥添加、删除和列出标签。这些示例使用 AWS Command Line Interface (AWS CLI)，但您可以使用任何受支持的编程语言。您无法标记 AWS 托管式密钥。

要添加、编辑、查看和删除密钥的标签，您必须具有所需的权限。有关更多信息，请参阅 控制对标签的访问。

主题

- CreateKey: 为新密钥添加标签
- TagResource: 为密钥添加或更改标签
- ListResource标签：获取密钥的标签
- UntagResource: 从密钥中删除标签

CreateKey: 为新密钥添加标签

您可以在创建密钥时向其添加标签。要指定标签，请使用CreateKey操作的Tags参数。

要在创建密钥时添加标签，调用方必须具有 IAM policy 中的 `payment-cryptography:TagResource` 权限。权限至少必须涵盖账户和区域中的所有密钥。有关更多信息，请参阅 控制对标签的访问。

CreateKey 的 Tags 参数的值是区分大小写的标签键和标签值对的集合。密钥上的每个标签都必须具有不同的标签名称。标签值可为 null 或空字符串。

例如，以下 AWS CLI 命令创建带有Project:Alpha标签的对称加密密钥。指定多个键值对时，请使用空格分隔每个对。

```
$ 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"}]'
```

当此命令成功时，它会返回一个 Key 对象以及有关新密钥的信息。但是，Key 不包括标签。要获取标签，请使用ListResource标签操作。

TagResource: 为密钥添加或更改标签

该TagResource操作向密钥添加一个或多个标签。此操作不能用于添加或编辑不同 AWS 账户中的标签。

要添加标签，请指定新标签键和标签值。要编辑标签，请指定现有标签键和新标签值。密钥上的每个标签都必须具有不同的标签键。标签值可为 null 或空字符串。

例如，以下命令将 **UseCase** 和 **BIN** 标签添加到示例密钥中。

```
$ aws payment-cryptography tag-resource --resource-arn arn:aws:payment-
cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h --tags
 '[{"Key":"UseCase","Value":"Acquiring"},{"Key":"BIN","Value":"123456"}]'
```

此命令成功执行后，不会返回任何输出。要查看密钥上的标签，请使用ListResource标签操作。

您也可以使用 TagResource 来更改现有标签的标签值。要替换标签值，请指定具有不同值的相同标签键。修改命令中未列出的标签不会被更改或删除。

例如，此命令会将 Project 标签的值从 Alpha 更改为 Noe。

该命令将返回 http/200，但不包含任何内容。要查看您的更改，请使用 ListTagsForResource

```
$ aws payment-cryptography tag-resource --resource-arn arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h \
```

```
        --tags '[{"Key":"Project","Value":"Noe"}]'
```

ListResource标签：获取密钥的标签

ListResource标签操作获取密钥的标签。ResourceArn（keyArn 或 keyAlias）参数是必需的。此操作不能用于查看其他 AWS 账户中的密钥上的标签。

例如，以下命令获取示例密钥的标签。

```
$ aws payment-cryptography list-tags-for-resource --resource-arn arn:aws:payment-
cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h

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

UntagResource: 从密钥中删除标签

该UntagResource操作会从密钥中删除标签。要标识要删除的标签，请指定标签键。此操作不能用于从其他 AWS 账户中的密钥中删除标签。

当它成功时，UntagResource 操作不返回任何输出。此外，如果在密钥上未找到指定的标签键，则不会抛出异常或返回响应。要确认该操作是否有效，请使用ListResource标签操作。

例如，此命令将从指定的密钥中删除 **Purpose** 标签及其值。

```
$ aws payment-cryptography untag-resource \
        --resource-arn arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h --tag-keys Project
```

控制对标签的访问

要通过使用 API 来添加、查看和删除标签，主体需要在 IAM policy 中获取标记权限。

您也可以通过对标签使用 AWS 全局条件键来限制这些权限。在 AWS 支付密码学中，这些条件可以控制对标记操作（例如TagResource和）的访问。UntagResource

有关示例策略和更多信息，请参阅 IAM 用户指南中的根据标签键控制访问。

用于创建和管理标签的权限如下所示。

支付密码学：TagResource

允许主体添加或编辑标签。要在创建密钥时添加标签，主体必须在 IAM policy 中具有不限于特定密钥的权限。

支付密码学：ListTagsForResource

允许主体查看密钥上的标签。

支付密码学：UntagResource

允许主体从密钥中删除标签。

标记策略中的权限

您可以在密钥政策或 IAM policy 中提供权限标记。例如，以下示例密钥政策向选定用户授予标记密钥的权限。它为所有可以担任示例管理员或开发人员角色的用户授予查看标签的权限。

```
{
  "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": "*"
    }
  ]
}
```

要授予主体对多个密钥的标记权限，您可以使用 IAM policy。为使此策略生效，每个密钥的密钥政策都必须允许账户使用 IAM policy 来控制对密钥的访问。

例如，以下 IAM policy 允许主体创建密钥。它还允许他们在指定账户中的所有密钥上创建和管理标签。这种组合允许委托人在创建密钥时使用CreateKey操作的 tags 参数向密钥添加标签。

```
{
  "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/*"
```

```
    }
  ]
}
```

限制标签权限

您可以通过使用策略条件限制标记权限。以下策略条件可应用于 payment-cryptography:TagResource 和 payment-cryptography:UntagResource 权限。例如，您可以使用 aws:RequestTag/*tag-key* 条件来允许主体仅添加特定标签，或阻止主体添加具有特定标签键的标签。

- aws：RequestTag
- a@@ ws:ResourceTag/tag-key（仅限 IAM 策略）
- aws：TagKeys

作为使用标签控制对密钥的访问的最佳实践，请使用 aws:RequestTag/*tag-key* 或 aws:TagKeys 条件键来确定允许哪些标签（或标签键）。

例如，以下 IAM policy 与上一个类似。但是，此策略允许主体为具有 Project 标签键的标签创建标签 (TagResource) 并删除标签 UntagResource。

由于TagResource和UntagResource请求可以包含多个标签，因此您必须使用 aws: TagKeys 条件指定ForAllValues或ForAnyValue设置运算符。ForAnyValue 运算符要求请求中至少有一个标签键与策略中的其中一个标签键匹配。ForAllValues 运算符要求请求中所有的标签键与策略中的其中一个标签键匹配。true如果请求中没有标签，则ForAllValues运算符也会返回，但 TagResource 如果未指定标签，则会 UntagResource失败。有关集合运算符的详细信息，请参阅 IAM 用户指南中的使用多个键和值。

```
{
  "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"}
      }
    }
  ]
}
```

使用标签控制对密钥的访问

您可以根据密钥上的标签控制对 AWS 支付密码的访问权限。例如，您可以编写 IAM policy，以允许主体仅启用和禁用具有特定标签的密钥。或者，您可以使用 IAM policy 防止主体在加密操作中使用密钥，除非密钥具有特定标签。

此功能是基于属性的访问控制 (ABAC) 的 AWS 支付密码学支持的一部分。有关使用标签控制 AWS 资源访问的信息，请参阅 ABAC 有什么用 AWS？ 以及使用 IAM 用户指南中的资源标签控制对资源的访问权限。 AWS

> **Note**

AWS Payment Crypto graph y 支持 aws: ResourceTag /tag- key 全局条件上下文密钥，它允许您根据密钥上的标签控制对密钥的访问权限。由于多个密钥可以具有相同的标签，此功能可使您将权限应用于一组选定的密钥。您还可以通过更改密钥的标签轻松更改集合中的 KMS 密钥。

在 AWS 支付密码学中，`aws:ResourceTag/tag-key`条件密钥仅在 IAM 策略中受支持。密钥策略（仅适用于一个密钥）或不使用特定密钥的操作（例如ListKeys或ListAliases操作）不支持它。

使用标签控制访问提供了一种简单、可扩展且灵活的方式来管理权限。但是，如果设计和管理不当，它可能会无意中允许或拒绝对您的密钥的访问。如果您使用标签来控制访问，请考虑以下做法。

- 使用标签来强化最低权限访问的最佳实践。仅为 IAM 主体授予他们对必须使用或管理的密钥的所需权限。例如，使用标签来标记用于项目的密钥。然后授予项目团队仅使用带有项目标签的密钥的权限。
- 谨慎为主体提供 `payment-cryptography:TagResource` 和 `payment-cryptography:UntagResource` 权限，以允许他们添加、编辑和删除标签。当您使用标签控制对密钥的访问时，更改标签可以授予主体使用他们没有权限使用的密钥的权限。它还可以拒绝对其他主体执行其工作所需的密钥的访问。不具有更改密钥策略或创建授权权限的密钥管理员可以控制对密钥的访问，前提是他们有权管理标签。

  如有可能，请使用策略条件，例如 `aws:RequestTag/`*`tag-key`* 或 `aws:TagKeys`，以将主体的标记权限限制为特定 密钥上的特定标签或标签模式。
- 查看您中当前拥有标记和取消标记权限 AWS 账户 的委托人，并在必要时对其进行调整。IAM policy 可能允许对所有密钥的标记和取消标记权限。例如，管理员托管策略允许主体标记、取消标记和列出所有密钥上的标签。
- 在设置依赖于标签的策略之前，请查看您的密钥上的标签 AWS 账户。请确保您的策略仅适用于您要包含的标签。使用CloudTrail 日志和 CloudWatch 警报提醒您注意可能影响密钥访问权限的标记更改。
- 基于标签的策略条件使用模式匹配；它们不绑定到标签的特定实例。使用基于标签的条件键的策略会影响与模式匹配的所有新标签和现有标签。如果删除并重新创建与策略条件匹配的标签，则该条件将应用于新标签，就像对旧标签一样。

例如，请考虑以下 IAM policy。它允许主体仅对您账户中位于美国东部（弗吉尼亚州北部）地区且带有 `"Project"="Alpha"` 标签的密钥调用 Decrypt 操作。您可以将此策略附加到示例 Alpha 项目中的角色。

```
{
  "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"
```

```
        }
      }
    }
  ]
}
```

以下示例 IAM policy 允许主体使用账户中的密钥执行特定加密操作。但它禁止主体对具有 "Type"="Reserved" 标签或不包含 "Type" 标签的密钥使用这些加密操作。

```
{
  "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"
        }
      }
    }
  ]
}
```

了解 AWS 支付密码学密钥的关键属性

正确管理密钥的原则是，密钥的范围必须适当，并且只能用于允许的操作。因此，某些密钥只能在特定密钥使用模式下创建。只要有可能，这就与 TR-31 定义的可用使用模式保持一致。

尽管 AWS 支付密码学可以防止您创建无效密钥，但为了方便起见，此处提供了有效的密钥组合。

对称密钥

- TR31_B0_BASE_DERIVATION_KEY
  - 允许的密钥算法：TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 允许的按键使用模式组合：{ DeriveKey = true}、{ NoRestrictions = true}
- TR31_C0_CARD_VERIFICATION_KEY
  - 允许的密钥算法：TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 允许的密钥使用模式组合：{生成 = true}、{Verify = true}、{Generate = true、Verify= true}、{= true}、{ NoRestrictions = true}
- TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY
  - 允许的密钥算法：TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 允许的密钥使用模式组合：{Encrypt = true、Decrypt = true、Wrap = true、Unwrap = true}、{encrypt = true、Unwrap = true}、{= true}、{= true} NoRestrictions
- TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS
  - 允许的密钥算法：TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 允许的按键使用模式组合：{ DeriveKey = true}、{ NoRestrictions = true}
- TR31_E1_EMV_MKEY_CONFIDENTIALITY

  - 允许的密钥算法：TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 允许的按键使用模式组合：{ DeriveKey = true}、{ NoRestrictions = true}
- TR31_E2_EMV_MKEY_INTEGRITY
  - 允许的密钥算法：TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 允许的按键使用模式组合：{ DeriveKey = true}、{ NoRestrictions = true}
- TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS
  - 允许的密钥算法：TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 允许的按键使用模式组合：{ DeriveKey = true}、{ NoRestrictions = true}
- TR31_E5_EMV_MKEY_CARD_PERSONALIZATION
  - 允许的密钥算法：TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 允许的按键使用模式组合：{ DeriveKey = true}、{ NoRestrictions = true}
- TR31_E6_EMV_MKEY_OTHER
  - 允许的密钥算法：TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 允许的按键使用模式组合：{ DeriveKey = true}、{ NoRestrictions = true}
- TR31_K0_KEY_ENCRYPTION_KEY
  - 允许的密钥算法：TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 允许的密钥使用模式组合：{Encrypt = true、Decrypt = true、Wrap = true、Unwrap = true}、{encrypt = true、Unwrap = true}、{= true}、{= true} NoRestrictions
- TR31_K1_KEY_BLOCK_PROTECTION_KEY
  - 允许的密钥算法：TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 允许的密钥使用模式组合：{Encrypt = true、Decrypt = true、Wrap = true、Unwrap = true}、{encrypt = true、Unwrap = true}、{= true}、{= true} NoRestrictions
- TR31_M1_ISO_9797_1_MAC_KEY
  - 允许的密钥算法：TDES_2KEY、TDES_3KEY
  - 允许的密钥使用模式组合：{生成 = true}、{Verify = true}、{Generate = true、Verify= true}、{= true}、{ NoRestrictions = true}
- TR31_M3_ISO_9797_3_MAC_KEY
  - 允许的密钥算法：TDES_2KEY、TDES_3KEY
  - 允许的密钥使用模式组合：{生成 = true}、{Verify = true}、{Generate = true、Verify= true}、{= true}、{ NoRestrictions = true}
- TR31_M6_ISO_9797_5_CMAC_KEY

  - 允许的密钥算法：TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 允许的密钥使用模式组合：{生成 = true}、{Verify = true}、{Generate = true、Verify= true}、{= true}、{ NoRestrictions = true}
- TR31_M7_HMAC_KEY
  - 允许的密钥算法：TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 允许的密钥使用模式组合：{生成 = true}、{Verify = true}、{Generate = true、Verify= true}、{= true}、{ NoRestrictions = true}
- TR31_P0_PIN_ENCRYPTION_KEY
  - 允许的密钥算法：TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 允许的密钥使用模式组合：{Encrypt = true、Decrypt = true、Wrap = true、Unwrap = true}、{encrypt = true、Unwrap = true}、{= true}、{= true} NoRestrictions
- TR31_V1_IBM3624_PIN_VERIFICATION_KEY
  - 允许的密钥算法：TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 允许的密钥使用模式组合：{生成 = true}、{Verify = true}、{Generate = true、Verify= true}、{= true}、{ NoRestrictions = true}
- TR31_V2_VISA_PIN_VERIFICATION_KEY
  - 允许的密钥算法：TDES_2KEY ,TDES_3KEY ,AES_128 ,AES_192 ,AES_256
  - 允许的密钥使用模式组合：{生成 = true}、{Verify = true}、{Generate = true、Verify= true}、{= true}、{ NoRestrictions = true}

非对称密钥

- TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION
  - 允许的密钥算法：RSA_2048 ,RSA_3072 ,RSA_4096
  - 允许的密钥使用模式组合：{ Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } , { Encrypt = true, Wrap = true } ,{ Decrypt = true, Unwrap = true }
  - 注意：: {encrypt = true，Wrap = true} 是导入用于加密数据或封装密钥的公钥时唯一有效的选项
- TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE
  - 允许的密钥算法：RSA_2048 ,RSA_3072 ,RSA_4096
  - 允许的按键使用模式组合：{Sign = true}、{Verify = true}
  - 注意：：在导入用于签名的密钥（例如 TR-34 的根证书、中间证书或签名证书）时，{Verify = true} 是唯一有效的选项。

数据操作

在您建立 AWS 支付密码学密钥后，即可使用它来执行加密操作。不同的操作执行不同类型的活动，包括加密、哈希以及领域特定算法（例如 CVV2 生成）。

如果没有匹配的解密密钥（对称密钥或私钥，取决于加密类型），则无法解密加密数据。同样，如果没有对称密钥或公钥，则无法验证哈希算法和特定域算法。

有关特定操作的有效密钥类型信息，请参阅加密操作的有效密钥

> **Note**
>
> 我们建议在非生产环境中使用测试数据。在非生产环境中使用生产密钥和数据（PAN、BDK ID 等）可能会影响您的合规范围，例如 PCI DSS 和 PCI P2PE。

主题

加密、解密和重新加密数据

加密和解密方法可用于使用各种对称和非对称技术（包括 TDES、AES 和 RSA）来加密或解密数据。这些方法还支持使用 DUKPT 和 EMV 技术派生的密钥。对于希望在新密钥下保护数据而不暴露底层数据的用例，也可以使用该 ReEncrypt 命令。

> **Note**
>
> 使用加密/解密函数时，假设所有输入均采用 HexBinary 格式——例如，值 1 将输入为 31（十六进制），小写的 t 表示为 74（十六进制）。所有输出也都采用十六进制格式。

有关所有可用选项的详细信息，请参阅加密、解密和重新加密的 API 指南。

主题

- 加密数据
- 解密数据

加密数据

该 Encrypt Data API 用于使用对称和非对称数据加密密钥以及 DUKPT 和 EMV 派生的密钥对数据进行加密。支持各种算法和变体，包括 TDES、RSA 和 AES。

主要输入是用于加密数据的加密密钥、要加密的 HexBinary 格式的纯文本数据以及诸如 TDES 之类的分组密码的初始化向量和模式等加密属性。纯文本数据必须是 8 字节TDES、16 字节AES和密钥长度的倍数（如果是）。RSA如果输入数据不符合这些要求，则应填充对称密钥输入（TDES、AES、DUKPT、EMV）。下表显示了每种密钥类型的最大明文长度以及您在中EncryptionAttributes为 RSA 密钥定义的填充类型。

填充类型	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

主要输出包括十六进制格式加密文字形式的加密数据以及加密密钥的校验和值。有关所有可用选项的详细信息，请参阅加密 API 指南。

示例

- 使用 AES 对称密钥加密数据
- 使用 DUKPT 密钥加密数据
- 使用 EMV 派生的对称密钥加密数据

- 使用 RSA 密钥加密会话数据

使用 AES 对称密钥加密数据

> **Note**
>
> 所有示例都假设相关密钥已经存在。可以使用该操作创建密钥，也可以使用该CreateKey操作导入密钥。ImportKey

Example

在此示例中，我们将使用使用操作创建或使用CreateKey操作导入的对称密钥对纯文本数据进行加密。ImportKey要执行此操作，密钥必须 KeyModesOfUse 设置为，Encrypt并 KeyUsage 设置为TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY。有关更多选项，请参阅加密操作密钥。

```
$ 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"
}
```

使用 DUKPT 密钥加密数据

Example

在此示例中，我们将使用 DU KPT 密钥对纯文本数据进行加密。 AWS 支付密码学支持TDES和 AES DUKPT 密钥。要执行此操作，密钥必须 KeyModesOfUse 设置为，DeriveKey并 KeyUsage 设置为TR31_B0_BASE_DERIVATION_KEY。有关更多选项，请参阅加密操作密钥。

```
$ 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"
}
```

使用 EMV 派生的对称密钥加密数据

Example

在此示例中，我们将使用已创建的 EMV 派生的对称密钥对明文数据进行加密。你可以使用这样的命令将数据发送到 EMV 卡。要执行此操作，密钥必须 KeyModesOfUse 设置为，`Derive`且必须 KeyUsage 设置为`TR31_E1_EMV_MKEY_CONFIDENTIALITY`或`TR31_E6_EMV_MKEY_OTHER`。有关更多详细信息，请参阅加密操作密钥。

```
$ 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"
}
```

使用 RSA 密钥加密会话数据

Example

在此示例中，我们将使用使用操作导入的 RSA 公钥对纯文本数据进行加密。ImportKey要执行此操作，密钥必须 KeyModesOfUse 设置为，Encrypt并 KeyUsage 设置为TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION。有关更多选项，请参阅加密操作密钥。

对于 PKCS #7 或其他目前不支持的填充方案，请在调用服务之前申请，并通过省略填充指示器 'Asymmetric={}' 来选择无填充

```
$ 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=OAEP_SHA256}'
```

```
{
 "CipherText":
 "12DF6A2F64CC566D124900D68E8AFEAA794CA819876E258564D525001D00AC93047A83FB13 \
 E73F06329A100704FA484A15A49F06A7A2E55A241D276491AA91F6D2D8590C60CDE57A642BC64A897F4832A3930
 \
 0FAEC7981102CA0F7370BFBF757F271EF0BB2516007AB111060A9633D1736A9158042D30C5AE11F8C5473EC70F067
 \
 72590DEA1638E2B41FAE6FB1662258596072B13F8E2F62F5D9FAF92C12BB70F42F2ECDCF56AADF0E311D4118FE3591
 \
 FB672998CCE9D00FFFE05D2CD154E3120C5443C8CF9131C7A6A6C05F5723B8F5C07A4003A5A6173E1B425E2B5E42AD
 \
 7A2966734309387C9938B029AFB20828ACFC6D00CD1539234A4A8D9B94CDD4F23A",
 "KeyArn": "arn:aws:payment-cryptography:us-east-1:529027455495:key/5dza7xqd6soanjtb",
 "KeyCheckValue": "FF9DE9CE"
}
```

解密数据

该 Decrypt Data API 用于使用对称和非对称数据加密密钥以及 DUK PT 和 EMV 派生的密钥来解密数据。支持各种算法和变体，包括 TDES、RSA 和 AES。

主要输入是用于解密数据的解密密钥、要解密的十六进制格式的加密文字数据以及解密属性，例如初始化向量、分组密码模式等。主要输出包括十六进制格式的明文解密数据以及解密密钥的校验和值。有关所有可用选项的详细信息，请查阅解密 API 指南。

示例

- 使用 AES 对称密钥解密数据
- 使用 DUKPT 密钥解密数据
- 使用 EMV 派生的对称密钥解密数据
- 使用 RSA 密钥解密数据

使用 AES 对称密钥解密数据

Example

在此示例中，我们将使用对称密钥解密密文数据。此示例显示了一个AES密钥，但TDES_3KEY也支持TDES_2KEY和。要执行此操作，密钥必须 KeyModesOfUse 设置为，Decrypt并 KeyUsage 设置为TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY。有关更多选项，请参阅加密操作密钥。

```
$ 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"
}
```

使用 DUKPT 密钥解密数据

> **Note**
>
> 将解密数据与 DUKPT 一起用于 P2PE 交易，可能会将信用卡 PAN 和其他持卡人数据返回到您的应用程序，在确定其 PCI DSS 范围时需要考虑这些数据。

Example

在此示例中，我们将使用使用操作创建或使用CreateKey操作导入的 DUKPT 密钥解密密文数据。ImportKey要执行此操作，密钥必须 KeyModesOfUse 设置为，DeriveKey并 KeyUsage 设置为TR31_B0_BASE_DERIVATION_KEY。有关更多选项，请参阅加密操作密钥。使用 DUKPT 时，对于 TDES 算法，加密文字数据长度必须是 16 字节的倍数。对于 AES 算法，加密文字数据长度必须是 32 字节的倍数。

```
$ 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"
}
```

使用 EMV 派生的对称密钥解密数据

Example

在此示例中，我们将使用 EMV 派生的对称密钥解密密文数据，该密钥是使用操作创建或使用操作导入CreateKey的。ImportKey要执行此操作，密钥必须 KeyModesOfUse 设置为，Derive且必须

KeyUsage 设置为TR31_E1_EMV_MKEY_CONFIDENTIALITY或TR31_E6_EMV_MKEY_OTHER。有关更多详细信息，请参阅加密操作密钥。

```
$ 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=1500000000000999,Mode=CBC}'
```

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

使用 RSA 密钥解密数据

Example

在此示例中，我们将使用使用操作创建的 RSA 密钥对来解密密文数据。CreateKey要执行此操作，必须 KeyModesOfUse 将密钥设置为启用Decrypt并 KeyUsage 设置为TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION。有关更多选项，请参阅加密操作密钥。

对于 PKCS #7 或当前不支持的其他填充方案，请通过省略填充指示符“Asymmetry={}”来选择无填充，并在调用服务后删除填充。

```
$ 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=OAEP_SHA256}'
```

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

生成并验证卡数据

生成并验证卡数据，包含从卡数据派生的数据，例如 CVV、CVV2、CVC 和 DCVV。

主题

生成卡数据

`Generate Card Data` API 用于使用 CVV、CVV2 或动态 CVV2 等算法生成卡数据。要查看此命令可以使用哪些密钥，请参阅加密操作的有效密钥部分。

许多加密值，例如 CVV、CVV2、iCVV、CAVV V8，都使用相同的加密算法，但输入值会有所不同。例如，CardVerificationValue1 的输入为 ServiceCode、卡号和到期日期。虽然 V CardVerificationalue2 只有两个这样的输入，但这是因为对于 CVV2/CVC2，固定为 000。ServiceCode 同样，对于iCvV，固定 ServiceCode 为999。某些算法可能会重新利用现有字段，例如 CAVV V8，在这种情况下，您需要查阅提供商手册以获取正确的输入值。

> **Note**
>
> 必须以相同的格式（例如 MMYY 与 YYMM）输入生成和验证的到期日期，才能生成正确的结果。

生成 CVV2

Example

在此示例中，我们将为给定的 PAN 生成一个 CVV2，其输入值为PAN和卡片到期日期。这假设您已生成信用卡验证密钥。

```
$ 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}
```

```
  {
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
      "KeyCheckValue": "CADDA1",
      "ValidationData": "801"
  }
```

生成 iCvV

Example

在此示例中，我们将为给定的 PAN 生成一个 ICvVPAN，其输入为，服务代码为 999，卡到期日期。这假设您已生成信用卡验证密钥。

有关所有可用参数，请参阅 API 参考指南中的 CardVerificationValue1。

```
$ 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
 CardVerificationValue1='{CardExpiryDate=1127,ServiceCode=999}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "CADDA1",
    "ValidationData": "801"
}
```

验证卡数据

`Verify Card Data` 用于验证使用依赖于加密主体的支付算法创建的数据，例如 `DISCOVER_DYNAMIC_CARD_VERIFICATION_CODE`。

输入值通常作为入站交易的一部分提供给发卡方或支持平台合作伙伴。要验证 ARQC 密码（用于 EMV 芯片卡），请参阅验证 ARQC。

有关更多信息，请参阅 API 指南VerifyCardValidationData中的。

如果该值经过验证，则 api 将返回 http/200。如果该值未经过验证，它将返回 http/400。

验证 CVV2

Example

在此示例中，我们将验证给定 PAN 的 CVV/CVV2。CVV2 通常由持卡人或用户在交易期间提供以供验证。为了验证他们的输入，将在运行时提供以下值：用于验证的密钥 (CVK)、PAN、信用卡到期日期和输入的 CVV2。卡到期格式必须与初始值生成中使用的格式匹配。

有关所有可用参数，请参阅 API 参考指南中的 CardVerificationValue2。

```
$ 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": "CADDA1"
  }
```

验证 iCvV

Example

在此示例中，我们将使用用于验证的密钥 (C VK)、服务代码为 999 PAN、卡到期日期以及交易提供的待验证的 IC V V 来验证给定 PAN 的 ICVV。

iCvV 不是用户输入的值（如 CVV2），而是嵌入在 EMV 卡上。应考虑在提供时是否应始终进行验证。

有关所有可用参数，请参阅 API 参考指南中的 CardVerificationValue1。

```
$ 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
 CardVerificationValue1='{CardExpiryDate=1127,ServiceCode=999}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "CADDA1",
    "ValidationData": "801"
}
```

生成、转换和验证 PIN 数据

PIN 数据功能允许您生成随机 pin、pin 验证值 (PVV) 并根据 PVV 或 PIN 偏移验证入站加密 PIN。

Pin 转换允许您将 pin 从一个工作密钥转换为另一个工作密钥，而无需按照 PCI PIN 要求 1 的规定以明文形式显示 pin。

> **Note**
>
> 由于 PIN 生成和验证通常是发行方功能，而 PIN 转换是典型的收单方功能，因此我们建议您考虑最低特权访问并为您的系统使用案例设置相应的策略。

主题

转换 PIN 数据

转换 PIN 数据功能用于将加密的 PIN 数据从一组密钥转换为另一组密钥，而加密数据不会离开 HSM。它用于 P2PE 加密，其中工作密钥应该更改，但处理系统不需要或不允许解密数据。主要输入是加密数据、用于加密数据的加密密钥以及用于生成输入值的参数。另一组输入是请求的输出参数，例如用于加密输出的密钥和用于创建该输出的参数。主要输出是新加密的数据集以及用于生成该数据集的参数。

> **Note**
>
> AES 密钥类型仅支持 ISO 格式 4 pin 块。

主题

从 PEK 到 DUKPT 的 PIN

Example

在此示例中，我们将使用 ISO 0 PIN 块将 PIN 从 PEK TDES 加密转换为使用 DUKPT 算法的 AES ISO 4 PIN 块。通常，此操作可能是反向进行的，即支付终端对 ISO 4 中的 pin 进行加密，然后将其转换回 TDES 进行下游处理。

```
$ 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",
            "KeyCheckValue": "7CC9E2"
        }
```

从 DUKPT 到 AWK 的 PIN

Example

在此示例中，我们将 PIN 从 AES DUKPT 加密 PIN 转换为在 AWK 下加密的 PIN。从功能上讲，它与前面的示例相反。

```
$ 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/4pmyquwjs3yj4vwe --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"
        }
```

生成 PIN 数据

生成 PIN 数据函数用于生成与 PIN 相关的值，例如 PVV 和 pin 块偏移，用于验证用户在交易或授权期间输入的 PIN 码。此 API 还可以使用各种算法生成新的随机 pin 码。

为密码生成 Visa PVV

Example

在此示例中，我们将生成一个新的（随机）引脚，其中的输出将是加密的 PIN block (PinData. PinBlock) 和 a PVV (pindata.Offset)。密钥输入是PAN、Pin Verification Key、Pin Encryption Key、和PIN block format。

此命令要求密钥的类型为TR31_V2_VISA_PIN_VERIFICATION_KEY。

```
$ 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"
            }
        }
```

为引脚生成 IBM3624 引脚偏移

IBM 3624 PIN Offset 有时也称为 IBM 方法。此方法使用验证数据（通常是 PAN）和 PIN 密钥 (PVK) 生成自然/中间 PIN 码。自然密码实际上是一种衍生值，对于发卡机构来说，确定性非常有效，因为不需要在持卡人级别存储密码数据。最明显的缺点是，该方案不考虑持卡人可选择的密码或随机密码。为了允许使用这些类型的引脚，在该方案中添加了偏移算法。偏移量表示用户选择（或随机）引脚与自然密钥之间的差异。抵消值由发卡机构或发卡商存储。在交易时，P AWS ayment Cryptography 服务会在内部重新计算自然密码并应用偏移量来找到密码。然后，它将其与交易授权提供的值进行比较。

IBM3624 有几个选项可供选择：

- `Ibm3624NaturalPin`将输出自然引脚和加密密码块
- `Ibm3624PinFromOffset`给定偏移量后将生成一个加密的密码块
- `Ibm3624RandomPin`将生成一个随机引脚，然后生成匹配的偏移量和加密的引脚块。
- `Ibm3624PinOffset`根据用户选择的引脚生成引脚偏移。

在 AWS 支付密码学的内部，执行以下步骤：

- 将提供的平移填充到 16 个字符。如果提供 <16，则使用提供的填充字符在右侧填充。
- 使用 PIN 生成密钥加密验证数据。
- 使用十进制表对加密的数据进行十进制化。这会将十六进制数字映射到十进制数字，例如 “A” 可能映射到 9，而 1 可能映射到 1。
- 从输出的十六进制表示形式中获取前 4 位数字。这是自然的别针。
- 如果用户选择或生成了随机引脚，则用客户引脚模数减去自然引脚。结果是引脚偏移。

示例

- 为引脚生成 IBM3624 引脚偏移

为引脚生成 IBM3624 引脚偏移

在此示例中，我们将生成一个新的（随机）引脚，其中的输出将是加密的 PIN block (PinData.PinBlock）和IBM3624偏移值（pindata.Offset）。输入是PAN验证数据（通常是平移）、填充字符Pin Verification Key、、Pin Encryption Key和。PIN block format

此命令要求密码生成密钥的类型为TR31_V1_IBM3624_PIN_VERIFICATION_KEY，加密密钥的类型为 TR31_P0_PIN_ENCRYPTION_KEY

Example

以下示例显示生成一个随机引脚，然后使用 Ibm3624 输出加密引脚块和 IBM3624 偏移值 RandomPin

```
$ 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
 Ibm3624RandomPin="{DecimalizationTable=9876543210654321,PinValidationDataPadCharacter=D,PinVal
```

```
{
              "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": {
                  "PinOffset": "5507"
              }
          }
```

验证 PIN 数据

验证 PIN 数据功能用于验证 pin 是否正确。这通常涉及将之前存储的 PIN 值与持卡人在 POI 输入的值进行比较。这些函数比较两个值，但不暴露任一来源的底层值。

使用 PVV 方法验证加密的 PIN

Example

在此示例中，我们将验证给定 PAN 的 PIN。PIN 通常由持卡人或用户在交易期间提供以进行验证，并与存档的值进行比较（持卡人的输入以加密值形式由终端或其他上游提供商提供）。为了验证此输入，还将在运行时提供以下值 - 用于加密输入 pin 的密钥（通常是 IWK）、PAN、以及要验证的值（PVV 或 PIN offset）。

如果 AWS 支付密码学能够验证密码，则会返回 http/200。如果 pin 未经过验证，将返回 http/400。

```
$ 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",

        }
```

根据先前存储的 IBM3624 引脚偏移量验证 PIN

在此示例中，我们将根据发卡机构/处理商存档的密码偏移量对持卡人提供的 PIN 进行验证。这些输入类似于???支付终端（或其他上游提供商，例如信用卡网络）提供的额外加密密码。如果引脚匹配，api 将返回 http 200。其中输出将是加密的 PIN block (PinData. PinBlock）和IBM3624偏移值（pindata.Offset）。

此命令要求密码生成密钥的类型为TR31_V1_IBM3624_PIN_VERIFICATION_KEY，加密密钥的类型为 TR31_P0_PIN_ENCRYPTION_KEY

Example

```
$ 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
 Ibm3624RandomPin="{DecimalizationTable=9876543210654321,PinValidationDataPadCharacter=D,PinVal
```

```
{
"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": {
  "PinOffset": "5507"
}
}
```

验证身份验证请求 (ARQC) 密码

验证身份验证请求密码 API 用于验证 ARQC。ARQC的生成不在 AWS 支付密码学的范围之内，通常在交易授权期间使用EMV芯片卡（或数字等效物，例如移动钱包）执行。ARQC 对于每笔交易都是唯一的，旨在以加密方式显示卡的有效性并确保交易数据与当前（预期）交易完全匹配。

AWS 支付密码学为验证ARQC和生成可选的ARPC值提供了多种选项，包括 EMV 4.4 Book 2 中定义的值以及Visa和Mastercard使用的其他方案。有关所有可用选项的完整列表，请参阅 API 指南中的 VerifyCardValidationData 部分。

ARQC 密码通常需要以下输入（尽管这可能因实现而异）：

- PAN-在 PrimaryAccountNumber 字段中指定
- PAN 序列号 (PSN)-在字段中指 PanSequenceNumber 定
- 密钥派生方法，例如通用会话密钥 (CSK)-在 SessionKeyDerivationAttributes
- 主密钥派生模式（例如 EMV 选项 A）-在 MajorKeyDerivationMode

- 交易数据-在 TransactionData 字段中指定的各种交易、终端和银行卡数据的字符串，例如金额和日期
- 颁发者主密钥-用于派生用于保护个人交易并在字段中指定的密码 (AC) 密钥的主密钥 KeyIdentifier

主题

建立交易数据

交易数据字段的确切内容（和顺序）因实现和网络方案而异，但最低推荐字段（和串联顺序）在 EMV 4.4 Book 2 第 8.1.1 节 ——数据选择中定义。如果前三个字段是金额 (17.00)、其他金额 (0.00) 和购买国家，则交易数据将按以下方式开始：

- 000000001700 - 金额 - 12 位隐含两位小数
- 000000000000 - 其他金额 - 12 位隐含两位小数
- 0124 - 四位数国家代码
- 输出（部分）交易数据 - 00000000170000000000000000124

交易数据填充

在将交易数据发送到服务之前，应先填充交易数据。大多数方案使用 ISO 9797 方法 2 填充，其中十六进制字符串后面加上十六进制 80，然后加上 00，直到该字段是加密块大小的倍数；TDES 为 8 字节或 16 个字符，AES 为 16 字节或 32 个字符。替代方案（方法 1）并不常见，但仅使用 00 作为填充字符。

ISO 9797 方法 1 填充

未填充：
00000000170000000000000000840008000800008401605170000000009380000B03011203（74 个字符或 37 个字节）

填充：
00000000170000000000000000840008000800008401605170000000009380000B03011203000000（80 个字符或 40 个字节）

ISO 9797 方法 2 填充

未填充：
0000000017000000000000000840008000800008401605170000000093800000B1F220103000000（80 个字符或 40 个字节）

填充：
0000000017000000000000000840008000800008401605170000000093800000B1F220103000000800000 个字符或 44 个字节）

示例

Visa CVN10

Example

在此示例中，我们将验证使用 Visa CVN10 生成的 ARQC。

如果 AWS 支付密码学能够验证 ARQC，则会返回 http/200。如果 arqc 未经过验证，将返回 http/400 响应。

```
$ aws payment-cryptography-data verify-auth-request-cryptogram --auth-request-
cryptogram D791093C8A921769 \
--key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk \
--major-key-derivation-mode EMV_OPTION_A \
--transaction-data
 0000000017000000000000000840008000800008401605170000000093800000B03011203000000 \
--session-key-derivation-attributes='{"Visa":{"PanSequenceNumber":"01" \
,"PrimaryAccountNumber":"9137631040001422"}}'
```

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

Visa CVN18 和 Visa CVN22

Example

在此示例中，我们将验证使用 Visa CVN18 或 CVN22 生成的 ARQC。CVN18 和 CVN22 之间的加密操作相同，但交易数据中包含的数据有所不同。与 CVN10 相比，即使输入相同，也会生成完全不同的密码。

如果 AWS 支付密码学能够验证 ARQC，则会返回 http/200。如果 arqc 未经过验证，将返回 http/400。

```
$ aws payment-cryptography-data verify-auth-request-cryptogram \
--auth-request-cryptogram 61EDCC708B4C97B4
--key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk \
--major-key-derivation-mode EMV_OPTION_A
--transaction-data
 00000000170000000000000008400080008000084016051700000000093800000B1F22010300000000000
 \
00000000000000000000000000000000000000000000800000000000000000
--session-key-derivation-attributes='{"EmvCommon":
{"ApplicationTransactionCounter":"000B", \
"PanSequenceNumber":"01","PrimaryAccountNumber":"9137631040001422"}}'
```

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

生成并验证 MAC

消息验证码 (MAC) 通常用于验证消息的完整性（是否已被修改）。HMAC（基于哈希的消息验证代码）、CBC-MAC 和 CMAC（基于密码的消息验证代码）等加密哈希还利用加密技术为 MAC 发送者提供额外的保证。HMAC 基于哈希函数，而 CMAC 基于分组密码。

此服务的所有 MAC 算法结合了加密哈希函数和共享密钥。他们获取消息和密钥，例如密钥中的密钥材料，然后返回一个唯一的标签或 MAC。即使是消息的一个字符发生了变化，或者密钥发生变化，生成的标签也会完全不同。通过要求提供密钥，加密 MAC 还提供了真实性；如果没有密钥，就不可能生成

相同的 MAC。加密 MAC 有时被称为对称签名，因为它们像数字签名一样工作，但使用单个密钥进行签名和验证。

AWS Payment Cryptography 支持多种类型的 MAC：

ISO9797 算法 1

由 ISO9797_ALGORITHM1 的 KeyUsage 表示

ISO9797 算法 3（零售 MAC）

由 ISO9797_ALGORITHM3 的 KeyUsage 表示

ISO9797 算法 5 (CMAC)

由 TR31_M6_ISO_9797_5_CMAC_KEY 的 KeyUsage 表示

HMAC

用 TR31_M7_HMAC_KEY 的 KeyUsage 表示，包括 HMAC_SHA224、HMAC_SHA256、HMAC_SHA384 和 HMAC_SHA512

主题

生成 MAC

Generate MAC API 用于验证与卡相关的数据，例如来自卡片磁条的跟踪数据，方法是使用已知的数据值生成 MAC（消息验证码），以在发送方和接收方之间进行数据验证。用于生成 MAC 的数据包括消息数据、机密 MAC 加密密钥和 MAC 算法，用于生成用于传输的唯一 MAC 值。MAC 的接收方将使用相同的 MAC 消息数据、MAC 加密密钥和算法来重现另一个 MAC 值以进行比较和数据验证。即使是消息的一个字符发生了变化，或者用于验证的 MAC 密钥不完全相同，生成的MAC 也会完全不同。该 API 支持用于此操作的 DUPKT MAC、HMAC 和 EMV MAC 加密密钥。

message-data 的输入值必须是十六进制数据。

在此示例中，我们将使用 HMAC 算法 HMAC_SHA256 和 HMAC 加密密钥生成用于卡数据身份验证的 HMAC（基于哈希的消息身份验证代码）。该密钥必须将 KeyUsage 设置为 TR31_M7_HMAC_KEY，并将 KeyModesOfUse 设置为 Generate。MAC 密钥可以使用 AWS Payment Cryptography 创建，通过调用 CreateKey 或调用 ImportKey 导入。

Example

```
$ aws payment-cryptography-data generate-mac \
    --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
qnobl5lghrzunce6 \
    --message-data
 "3b3130383834393030313037333934313533d32343038323236303030373030303f33" \
    --generation-attributes Algorithm=HMAC_SHA256
```

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

验证 MAC

Verify MAC API 用于验证与卡片相关的数据身份验证的 MAC（消息身份验证代码）。它必须使用生成 MAC 期间使用的相同加密密钥来重新生成用于身份验证的 MAC 值。MAC 加密密钥可以使用 AWS Payment Cryptography 创建，通过调用 CreateKey 或调用 ImportKey 导入。该 API 支持用于此操作的 DUPKT MAC、HMAC 和 EMV MAC 加密密钥。

如果该值已通过验证，则响应参数`MacDataVerificationSuccessful` 将返回 `Http/200`，否则将为 `Http/400`，消息表示 `Mac verification failed`。

在此示例中，我们将使用 HMAC 算法 HMAC_SHA256 和 HMAC 加密密钥来验证用于卡数据身份验证的 HMAC（基于哈希的消息身份验证代码）。该密钥必须将 KeyUsage 设置为 TR31_M7_HMAC_KEY，并将 KeyModesOfUse 设置为 `Verify`。

Example

```
$ aws payment-cryptography-data verify-mac \
     --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
qnobl5lghrzunce6 \
     --message-data
 "3b3430383834393030313037333934313533d32343038323236303030373030303f33" \
     --verification-attributes='Algorithm=HMAC_SHA256' \
     --mac ED87F26E961C6D0DDB78DA5038AA2BDDEA0DCE03E5B5E96BDDD494F4A7AA470C
```

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

加密操作的有效密钥

某些密钥只能用于某些操作。此外，某些操作可能会限制密钥的密钥使用模式。请查看下表中允许的组合。

> **Note**
>
> 某些组合虽然允许，但可能会造成无法使用的情况，例如生成 CVV 代码(`generate`)但随后无法验证(`verify`)。

主题

- GenerateCard数据
- VerifyCard数据
- GeneratePinData （适用于签证/ABA计划）
- GeneratePinData （对于IBM3624）
- VerifyPinData （适用于签证/ABA计划）
- VerifyPinData （对于IBM3624）
- 解密数据
- 加密数据
- 转换 PIN 数据
- 生成/验证 MAC
- VerifyAuthRequestCryptogram
- Import/Export 密钥
- 未使用的密钥类型

GenerateCard数据

API 端点	加密操作或算法	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
GenerateCard数据	• AMEX_CARD_SECURITY_CODE_VERSION_1 • AMEX_CARD_SECURITY_CODE_VERSION_2	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2KEY • TDES_3KEY	{ Generate = true },{ Generate = true, Verify = true }
GenerateCard数据	• CARD_VERIFICATION_VALUE_1 • CARD_VERIFICATION_VALUE_2	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2KEY	{ Generate = true },{ Generate = true, Verify = true }
GenerateCard数据	• CARDHOLDER_AUTHENTICATION_VERIFICATION_VALUE	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{ DeriveKey = true}
GenerateCard数据	• DYNAMIC_CARD_VERIFICATION_CODE	TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS	• TDES_2KEY	{ DeriveKey = true}
GenerateCard数据	• DYNAMIC_CARD_VERIFICATION_VALUE	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{ DeriveKey = true}

VerifyCard数据

加密操作或算法	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
• AMEX_CARD_SECURITY_CODE_VERSION_1 • AMEX_CARD_SECURITY_CODE_VERSION_2	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2KEY • TDES_3KEY	{ Generate = true }, { Generate = true, Verify = true }
• CARD_VERIFICATION_VALUE_1 • CARD_VERIFICATION_VALUE_2	TR31_C0_CARD_VERIFICATION_KEY	• TDES_2KEY	{ Generate = true }, { Generate = true, Verify = true }
• CARDHOLDER_AUTHENTICATION_VERIFICATION_VALUE	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{ DeriveKey = true}
• DYNAMIC_CARD_VERIFICATION_CODE	TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS	• TDES_2KEY	{ DeriveKey = true}
• DYNAMIC_CARD_VERIFICATION_VALUE	TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	{ DeriveKey = true}

GeneratePinData（适用于签证/ABA计划）

`VISA_PIN or VISA_PIN_VERIFICATION_VALUE`

密钥类型	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
KMS 加密密钥	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY	• { Encrypt = true, Wrap = true } • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } • { NoRestrictions = true}
PIN 生成密钥	TR31_V2_VISA_PIN_VERIFICATION_KEY	• TDES_3KEY	• { Generate = true } • { Generate = true, Verify = true }

GeneratePinData（对于**IBM3624**）

`IBM3624_PIN_OFFSET,IBM3624_NATURAL_PIN,IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET)`

密钥类型	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
KMS 加密密钥	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY	适用于 IBM3624_NATURAL_PIN, IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET

密钥类型	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
			• { Encrypt = true, Wrap = true } • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } • { NoRestrictions = true} 适用于 IBM3624_PIN_OFFSET • { Encrypt = true, Unwrap = true } • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } • { NoRestrictions = true}
PIN 生成密钥	TR31_V1_IBM3624_PIN_VERIFICATION_KEY	• TDES_3KEY	• { Generate = true } • { Generate = true, Verify = true }

VerifyPinData（适用于签证/ABA计划）

`VISA_PIN`

密钥类型	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
KMS 加密密钥	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY	• { Decrypt = true, Unwrap = true } • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } • { NoRestrictions = true}
PIN 生成密钥	TR31_V2_VISA_PIN_VERIFICATION_KEY	• TDES_3KEY	• { Verify = true } • { Generate = true, Verify = true }

VerifyPinData（对于**IBM3624**）

`IBM3624_PIN_OFFSET,IBM3624_NATURAL_PIN,IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET)`

密钥类型	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
KMS 加密密钥	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY	适用于 IBM3624_NATURAL_PIN, IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET • { Decrypt = true, Unwrap = true } • { Encrypt = true, Decrypt = true,

密钥类型	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
			Wrap = true, Unwrap = true } • { NoRestrictions = true}
PIN 验证密钥	TR31_V1_IBM3624_PIN_VERIFICATION_KEY	• TDES_3KEY	• { Verify = true } • { Generate = true, Verify = true }

解密数据

密钥类型	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
DUKPT	TR31_B0_BASE_DERIVATION_KEY	• TDES_2KEY • AES_128 • AES_192 • AES_256	• { DeriveKey = true} • { NoRestrictions = true}
EMV	TR31_E1_EMV_MKEY_CONFIDENTIALITY TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	• { DeriveKey = true}
RSA	TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION	• RSA_2048 • RSA_3072 • RSA_4096	• { Decrypt = true, Unwrap=true} • {Encrypt=true, Wrap=true,Decrypt = true, Unwrap=true}

密钥类型	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
对称密钥	TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• {Decrypt = true, Unwrap=true} • {Encrypt=true, Wrap=true,Decrypt = true, Unwrap=true} • { NoRestrictions = true}

加密数据

密钥类型	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
DUKPT	TR31_B0_BASE_DERIVATION_KEY	• TDES_2KEY • AES_128 • AES_192 • AES_256	• { DeriveKey = true} • { NoRestrictions = true}
EMV	TR31_E1_EMV_MKEY_CONFIDENTIALITY TR31_E6_EMV_MKEY_OTHER	• TDES_2KEY	• { DeriveKey = true}
RSA	TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION	• RSA_2048 • RSA_3072 • RSA_4096	• { Encrypt = true, Wrap=true} • {Encrypt=true, Wrap=true,Decrypt = true, Unwrap=true}

密钥类型	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
对称密钥	TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• {Encrypt = true, Wrap=true} • {Encrypt=true, Wrap=true,Decrypt = true, Unwrap=true} • { NoRestrictions = true}

转换 PIN 数据

方向	密钥类型	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
入站数据来源	DUKPT	TR31_B0_BASE_DERIVATION_KEY	• TDES_2KEY • AES_128 • AES_192 • AES_256	• { DeriveKey = true} • { NoRestrictions = true}
入站数据来源	非 DUKPT（PEK、AWK、IWK 等）	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• { Decrypt = true, Unwrap = true } • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } • { NoRestrictions = true}

方向	密钥类型	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
出站数据目标	DUKPT	TR31_B0_BASE_DERIVATION_KEY	• TDES_2KEY • AES_128 • AES_192 • AES_256	• { DeriveKey = true} • { NoRestrictions = true}
出站数据目标	非 DUKPT（PEK、IWK、AWK 等）	TR31_P0_PIN_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• { Encrypt = true, Wrap = true } • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true } • { NoRestrictions = true}

生成/验证 MAC

MAC 密钥用于创建消息/数据正文的加密哈希。不建议创建密钥使用模式有限的密钥，因为您将无法执行匹配操作。但是，如果另一个系统打算执行另一半的操作对，则只能通过一个操作来导入/导出密钥。

允许的密钥用法	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
MAC 密钥	TR31_M1_ISO_9797_1_MAC_KEY	• TDES_2KEY • TDES_3KEY	• { Generate = true } • { Generate = true, Verify = true } • { Verify = true } • { Generate = true }

允许的密钥用法	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
MAC 密钥（零售 MAC）	TR31_M1_ISO_9797_3_MAC_KEY	• TDES_2KEY • TDES_3KEY	• { Generate = true } • { Generate = true, Verify = true } • { Verify = true } • { Generate = true }
MAC 密钥 (CMAC)	TR31_M6_ISO_9797_5_CMAC_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• { Generate = true } • { Generate = true, Verify = true } • { Verify = true } • { Generate = true }
MAC 密钥 (HMAC)	TR31_M7_HMAC_KEY	• TDES_2KEY • TDES_3KEY • AES_128 • AES_192 • AES_256	• { Generate = true } • { Generate = true, Verify = true } • { Verify = true } • { Generate = true }

VerifyAuthRequestCryptogram

允许的密钥用法	EMV 选项	允许的密钥算法	允许的密钥使用模式组合
• 选项 A • 选项 B	TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS	• TDES_2KEY	• { DeriveKey = true}

Import/Export 密钥

操作类型	允许的密钥用法	允许的密钥算法	允许的密钥使用模式组合
TR-31 包装钥匙	TR31_K1_KEY_BLOCK_PROTECTION_KEY TR31_K0_KEY_ENCRYPTION_KEY	• TDES_2KEY • TDES_3KEY • AES_128	• {encrypt = true，Wrap = true}（仅导出） • {Decrypt = true，Unwrap = true}（仅限导入） • { Encrypt = true, Decrypt = true, Wrap = true, Unwrap = true }
导入可信 CA	TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE	• RSA_2048 • RSA_3072 • RSA_4096	• { Verify = true }
导入用于非对称加密的公钥证书	TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION	• RSA_2048 • RSA_3072 • RSA_4096	• {encrypt= True，wrap=True

未使用的密钥类型

AWS 支付密码学目前未使用以下密钥类型

- TR31_P1_PIN_PIN_GENERATION_KEY
- TR31_K3_ASYMMETRIC_KEY_FOR_KEY_ACTREMENT

AWS 支付密码学的安全性

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将此描述为云的安全性和云中的安全性：

- 云安全 —AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于 AWS 支付加密的合规计划，请参阅按合规计划划分的 AWS 范围内的服务 AWS 按合规计划。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

本主题可帮助您了解在使用 AWS 支付密码学时如何应用分担责任模型。它向您展示了如何配置 AWS 支付密码以满足您的安全和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 AWS 支付密码学资源。

主题

- AWS 支付密码学中的数据保护
- AWS 支付密码学的弹性
- 中的基础设施安全 AWS Payment Cryptography
- 通过 VPC 终端节点连接到 AWS 支付加密
- AWS 支付密码学安全最佳实践

AWS 支付密码学中的数据保护

分 AWS 担责任模型适用于 AWS 支付密码学中的数据保护。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础设施上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS 安全性博客 上的 AWS 责任共担模式和 GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证 (MFA)。
- 使用 SSL/TLS 与资源通信。 AWS 我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您使用控制台、API 或 AWS SDK AWS 服务 使用 AWS 支付加密或其他工具包的情况。 AWS CLI在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

AWS Payment Cryptography 存储和保护您的支付加密密钥，使其具有高度可用性，同时为您提供强大而灵活的访问控制。

主题

保护密钥材料

默认情况下，AWS Payment Cryptography 会保护该服务管理的支付密钥的加密密钥材料。此外，AWS Payment Cryptography 还提供用于导入在服务外部创建的密钥材料的选项。有关支付密钥和密钥材料的技术详细信息，请参阅 AWS Payment Cryptography 加密详细信息。

数据加密

AWS Payment Cryptography 中的数据包括 AWS Payment Cryptography 密钥、它们所代表的加密密钥材料及其使用属性。此密钥材料仅在其使用时以明文形式存在于 AWS Payment Cryptography 硬件安全模块 (HSM) 中。否则，密钥材料和属性将被加密并存储在持久性存储中。

AWS Payment Cryptography 生成或加载的支付密钥的密钥材料永远不会以未加密的形式离开 AWS Payment Cryptography HSM 的边界。它可以通过 AWS Payment Cryptography API 操作加密导出。

静态加密

AWS Payment Cryptography 为 PCI PTS HSM 列出的 HSM 中的支付密钥生成密钥材料。密钥材料未使用时，会利用 HSM 密钥进行加密，并写入耐久的持久性存储中。Payment Cryptography 密钥的密钥材料和保护密钥材料的加密密钥永远不会以明文形式离开 HSM。

Payment Cryptography 密钥的密钥材料的加密和管理完全由服务处理。

有关更多信息，请参阅 AWS Key Management Service 加密详情。

传输中加密

AWS Payment Cryptography 为支付密钥生成或加载的密钥材料绝不会在 AWS Payment Cryptography API 操作中以明文形式导出或传输。AWS Payment Cryptography 使用密钥标识符表示 API 操作中的密钥。

但是，某些 AWS Payment Cryptography API 操作会导出由先前共享或非对称密钥交换密钥加密的密钥。此外，客户可以使用 API 操作来为支付密钥导入密钥材料。

所有 AWS Payment Cryptography API 调用必须使用传输层安全性协议 (TLS) 进行签名和传输。AWS Payment Cryptography 需要 PCI 定义为“强加密技术”的 TLS 版本和密码套件。所有服务端点均支持 TLS 1.0-1.3 和混合后量子 TLS。

有关更多信息，请参阅 AWS Key Management Service 加密详情。

互联网络流量隐私保护

AWS Payment Cryptography 支持 Amazon Web Services 管理控制台和一组使您能够创建和管理支付密钥并在加密操作中使用它们的 API 操作。

AWS Payment Cryptography 支持两种网络连接选项，从您的私有网络到 AWS。

- Internet 上的 IPsec VPN 连接
- AWS Direct Connect，该服务通过标准的以太网光纤电缆将您的内部网络链接到 AWS Direct Connect 位置。

所有 Payment Cryptography API 调用必须使用传输层安全性协议 (TLS) 进行签名和传输。这些调用还需要一个现代化的密码套件，该套件支持完美向前保护。仅允许通过 AWS 内部网络从已知的 AWS Payment Cryptography API 主机向存储密钥的密钥材料的硬件安全模块 (HSM) 传输流量。

要从您的虚拟私有云 (VPC) 直接连接到 AWS Payment Cryptography，而不通过公共互联网发送流量，请使用由 AWS PrivateLink 提供支持的 VPC 终端节点。有关更多信息，请参阅通过 VPC 端点连接到 AWS Payment Cryptography。

AWS Payment Cryptography 还支持对传输层安全性协议 (TLS) 网络加密协议使用混合后量子密钥交换选项。当您连接到 AWS Payment Cryptography API 端点时，可以结合使用此选项与 TLS。

AWS 支付密码学的弹性

AWS 全球基础设施是围绕 AWS 区域和可用区构建的。区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅AWS 全球基础设施。

区域隔离

AWS Payment Cryptography 是一项区域性服务，可在多个区域使用。

AWS Payment Cryptography 的区域隔离设计可确保一个 Amazon Web Services 区域的可用性问题不会影响任何其他区域的 AWS Payment Cryptography 操作。AWS Payment Cryptography 旨在确保零计划停机，所有软件更新和扩展操作都在不知不觉中无缝执行。

AWS Payment Cryptography 服务水平协议 (SLA) 为所有 Payment Cryptography API 提供 99.99% 的服务承诺。为履行这一承诺，AWS Payment Cryptography 会确保执行 API 请求所需的所有数据和授权信息在接收该请求的所有区域主机上都可用。

AWS Payment Cryptography 基础设施会在每个区域的至少三个可用区 (AZ) 中复制。为确保多个主机故障不会影响 AWS Payment Cryptography 性能，AWS Payment Cryptography 旨在服务于来自区域中任何 AZ 的客户流量。

您对支付密钥属性或权限所做的更改将复制到该区域中的所有主机，以确保该区域中的任何主机都能正确处理后续请求。有关使用支付密钥的加密操作请求将会转发给某个 AWS Payment Cryptography 硬件安全模块（HSM）队列，其中任何一个模块都可以使用支付密钥执行操作。

多租户设计

AWS Payment Cryptography 的多租户设计使其能够达到可用性 SLA，并保持较高的请求率，同时保护密钥和数据的保密性。

通过部署多个完整性控制执行机制，以确保实际用于执行加密操作的支付密钥始终是您为该操作指定的密钥。

Payment Cryptography 密钥的明文密钥材料受到全面保护。密钥材料在创建后将立即在 HSM 中加密，并且加密后的密钥材料会立即移动到安全的存储中。加密后的密钥仅在使用时才在 HSM 中检索和解密。明文密钥仅在完成加密操作所需的时间内驻留在 HSM 内存中。明文密钥材料永远不会离开 HSM；并且永远不会写入持久性存储。

要详细了解 AWS Payment Cryptography 使用的密钥保护机制，请参阅 AWS Payment Cryptography 加密详细信息。

中的基础设施安全 AWS Payment Cryptography

作为一项托管服务， AWS Payment Cryptography 受到《Amazon Web Services：安全流程概述》白皮书中描述的 AWS 全球网络安全程序的保护。

您可以使用 AWS 已发布的 API 调用 AWS Payment Cryptography 通过网络进行访问。客户端必须支持传输层安全性（TLS）1.2 或更高版本。客户端还必须支持具有完全向前保密（PFS）的密码套件，例如 Ephemeral Diffie-Hellman（DHE）或 Elliptic Curve Ephemeral Diffie-Hellman（ECDHE）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service（AWS STS）生成临时安全凭证来对请求进行签名。

物理主机的隔离

AWS Payment Cryptography 使用的物理基础设施的安全性受 Amazon Web Services：安全流程概述的物理和环境安全部分中所述的控制措施约束。您可以在上一节中列出的合规性报告和第三方审计结果中找到更多详细信息。

AWS Payment Cryptography 由 commercial-off-the-shelf PCI PTS HSM 列出的专用硬件安全模块 (HSM) 提供支持。AWS Payment Cryptography 的密钥材料仅存储在 HSM 上的易失存储器中，并且仅在 Payment Cryptography 密钥正在使用时才存储。HSM 位于 Amazon 数据中心内的访问控制机架中，对任何物理访问实施双重控制。有关 AWS Payment Cryptography HSM 的操作的详细信息，请参阅 AWS Payment Cryptography 加密详细信息。

通过 VPC 终端节点连接到 AWS 支付加密

您可以通过虚拟私有云 (VPC) 中的私有接口终端节点直接连接到 AWS 支付加密。当您使用接口 VPC 终端节点时，您的 VPC 和 AWS 支付加密之间的通信完全在 AWS 网络内进行。

AWS Payment Cryptography 支持由AWS PrivateLink提供支持的亚马逊虚拟私有云（亚马逊 VPC）终端节点。每个 VPC 终端节点都由您的 VPC 子网中一个或多个使用私有 IP 地址的弹性网络接口 (ENI) 代表。

接口 VPC 终端节点将您的 VPC 直接连接到 AWS 支付加密，无需互联网网关、NAT 设备、VPN 连接或 AWS Direct Connect 连接。您的 VPC 中的实例不需要公有 IP 地址即可与 AWS 支付加密进行通信。

区域

AWS 支付密码学支持 VPC 终端节点和 VPC 终端节点策略，所有这些 AWS 区域 策略都支持AWS 支付加密。

主题

- AWS 支付加密 VPC 终端节点的注意事项
- 为 AWS 支付加密创建 VPC 终端节点
- 连接到 AWS 支付加密 VPC 终端节点
- 控制对 VPC 终端节点的访问
- 在策略语句中使用 VPC 终端节点
- 记录您的 VPC 终端节点

AWS 支付加密 VPC 终端节点的注意事项

在为 AWS 支付加密设置接口 VPC 终端节点之前，请查看AWS PrivateLink 指南中的接口终端节点属性和限制主题。

AWS VPC 终端节点的支付加密支持包括以下内容。

- 您可以使用您的 VPC 终端节点从 VPC 调用所有AWS 支付加密控制面板操作和AWS 支付加密数据平面操作。
- 您可以创建连接 AWS 支付加密区域终端节点的接口 VPC 终端节点。
- AWS 支付密码学由控制平面和数据平面组成。您可以选择设置一个或两个子服务，但每个子服务都是单独配置的。
- 您可以通过 VPC 终端节点使用 AWS CloudTrail 日志来审核您对 AWS 支付加密密钥的使用情况。有关更多信息，请参阅 记录您的 VPC 终端节点。

为 AWS 支付加密创建 VPC 终端节点

您可以使用亚马逊 VPC 控制台或亚马逊 VPC API 为 AWS 支付加密创建 VPC 终端节点。有关更多信息，请参阅《AWS PrivateLink 指南》中的创建接口端点。

- 要为 AWS 支付加密创建 VPC 终端节点，请使用以下服务名称：

```
com.amazonaws.region.payment-cryptography.controlplane
```

```
com.amazonaws.region.payment-cryptography.dataplane
```

例如，在美国西部（俄勒冈）区域 (us-west-2) 中，服务名称将是：

```
com.amazonaws.us-west-2.payment-cryptography.controlplane
```

```
com.amazonaws.us-west-2.payment-cryptography.dataplane
```

为了更轻松地使用 VPC 终端节点，您可以为 VPC 终端节点启用私有 DNS 名称。如果您选择启用 DNS 名称选项，则标准 AWS 支付加密 DNS 主机名将解析到您的 VPC 终端节点。例如，`https://controlplane.payment-cryptography.us-west-2.amazonaws.com` 将解析为连接到服务名称 `com.amazonaws.us-west-2.payment-cryptography.controlplane` 的 VPC 端点。

此选项可让您更轻松地使用 VPC 终端节点。默认情况下， AWS 软件开发工具包和 AWS CLI 使用标准 AWS 支付加密 DNS 主机名，因此您无需在应用程序和命令中指定 VPC 终端节点 URL。

有关更多信息，请参阅 AWS PrivateLink 指南中的通过接口端点访问服务。

连接到 AWS 支付加密 VPC 终端节点

您可以使用 AWS SDK（ AWS CLI 或 AWS Tools for PowerShell ）通过 VPC 终端节点连接到 AWS 支付加密。要指定 VPC 终端节点，请使用其 DNS 名称。

例如，此 list-keys 命令使用 `endpoint-url` 参数指定 VPC 终端节点。要使用类似命令，请将示例中的 VPC 终端节点 ID 替换为您账户中的 ID。

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

如果在创建 VPC 终端节点时启用了私有主机名，则无需在 CLI 命令或应用程序配置中指定 VPC 终端节点 URL。标准 AWS 支付加密 DNS 主机名解析到您的 VPC 终端节点。 AWS CLI 和软件开发工具包默认使用此主机名，因此您可以开始使用 VPC 终端节点连接到 AWS 支付加密区域终端节点，而无需更改脚本和应用程序中的任何内容。

要使用私有主机名，您的 VPC 的 `enableDnsHostnames` 和 `enableDnsSupport` 属性必须设置为 `true`。要设置这些属性，请使用ModifyVpc属性操作。有关详细信息，请参阅《Amazon VPC 用户指南》中的查看和更新 VPC 的 DNS 属性。

控制对 VPC 终端节点的访问

要控制对 AWS 支付加密的 VPC 终端节点的访问，请将 V PC 终端节点策略附加到您的 VPC 终端节点。终端节点策略决定委托人是否可以使用 VPC 终端节点通过特定的 AWS 支付加密资源调用 AWS 支付加密操作。

您可以在创建终端节点时创建 VPC 终端节点策略，并且可以随时更改 VPC 终端节点策略。使用 VPC 管理控制台或CreateVpc终端节点或终ModifyVpc端节点操作。您也可以使用 AWS CloudFormation 模板创建和更改 VPC 终端节点策略。有关使用 VPC 管理控制台的帮助，请参阅 AWS PrivateLink 指南中的创建接口终端节点和修改接口终端节点。

主题

- 关于 VPC 终端节点策略
- 默认的 VPC 终端节点策略
- 创建 VPC 端点策略
- 查看 VPC 终端节点策略

关于 VPC 终端节点策略

要使使用 VPC 终端节点的 AWS 支付加密请求成功，委托人需要来自两个来源的权限：

- 基于身份的策略必须授予委托人对资源调用操作的权限（AWS 付款加密密钥或别名）。
- VPC 终端节点策略必须授予委托人使用终端节点发出请求的权限。

例如，密钥策略可能允许委托人对特定的 AWS 支付加密密钥调用 Decrypt。但是，VPC 终端节点策略可能不允许该委托人使用终端节点调`Decrypt`用该 AWS 支付加密密钥。

或者，VPC 终端节点策略可能允许委托人使用终端节点调StopKey用某些 AWS 支付加密密钥的 Usage。但是，如果委托人没有 IAM 策略中的这些权限，则请求将失败。

默认的 VPC 终端节点策略

每个 VPC 终端节点都有 VPC 终端节点策略，但您无需指定策略。如果未指定策略，则默认的终端节点策略允许所有委托人对终端节点上的所有资源执行所有操作。

但是，对于 AWS 支付加密资源，委托人还必须拥有从 IAM 策略调用操作的权限。因此，在实践中，默认策略表示，如果委托人有权对资源调用操作，他们也可以通过使用终端节点调用该操作。

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Principal": "*",
      "Resource": "*"
    }
  ]
}
```

要允许主体仅将 VPC 终端节点用于其允许操作的子集，请创建或更新改 VPC 终端节点策略。

创建 VPC 端点策略

VPC 终端节点策略确定委托人是否有权使用 VPC 终端节点对资源执行操作。对于 AWS 支付密码学资源，委托人还必须有权执行 IAM 策略中的操作。

每个 VPC 终端节点策略语句都需要以下元素：

- 可执行操作的委托人

- 可执行的操作
- 可对其执行操作的资源

策略语句不指定 VPC 终端节点。它适用于策略所附加到的任何 VPC 终端节点。有关更多信息，请参阅《Amazon VPC 用户指南》中的使用 VPC 端点控制对服务的访问权限。

以下是 AWS 支付加密的 VPC 终端节点策略示例。当连接到 VPC 终端节点时，此策略ExampleUser允许使用 VPC 终端节点对指定的 AWS 支付加密密钥调用指定操作。在使用此类政策之前，请将示例委托人和密钥标识符替换为账户中的有效值。

```
{
   "Statement":[
      {
         "Sid": "AllowDecryptAndView",
         "Principal": {"AWS": "arn:aws:iam::111122223333:user/ExampleUser"},
         "Effect":"Allow",
         "Action": [
             "payment-cryptography:Decrypt",
             "payment-cryptography:GetKey",
             "payment-cryptography:ListAliases",
             "payment-cryptography:ListKeys",
             "payment-cryptography:GetAlias"
          ],
         "Resource": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
      }
   ]
}
```

AWS CloudTrail 记录使用 VPC 终端节点的所有操作。但是，您的 CloudTrail 日志不包括其他账户中的委托人请求的操作或其他账户中 AWS 支付密码密钥的操作。

因此，您可能需要创建一个 VPC 终端节点策略，以防止外部账户中的委托人使用 VPC 终端节点对本地账户中的任何密钥调用任何 AWS 支付加密操作。

以下示例使用 a ws: g PrincipalAccount lobal 条件密钥拒绝所有委托人访问所有 AWS 支付密码密钥的所有操作，除非委托人位于本地账户中。使用类似于此策略的策略之前，请使用有效值替换示例账户 ID。

```
{
  "Statement": [
```

```
    {
      "Sid": "AccessForASpecificAccount",
      "Principal": {"AWS": "*"},
      "Action": "payment-cryptography:*",
      "Effect": "Deny",
      "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*",
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalAccount": "111122223333"
        }
      }
    }
  ]
}
```

查看 VPC 终端节点策略

要查看终端节点的 VPC 终端节点策略，请使用 VPC 管理控制台或DescribeVpc终端节点操作。

以下 AWS CLI 命令获取具有指定 VPC 终端节点 ID 的终端节点的策略。

在使用此命令之前，请将示例终端节点 ID 替换为您账户中的有效终端节点 ID。

```
$ aws ec2 describe-vpc-endpoints \
--query 'VpcEndpoints[?VpcEndpointId==``].[PolicyDocument]'
--output text
```

在策略语句中使用 VPC 终端节点

当请求来自 VPC 或使用 VPC 终端节点时，您可以控制对 AWS 支付加密资源和操作的访问权限。为此，请使用一个 IAM 策略

- 使用 aws:sourceVpce 条件键基于 VPC 终端节点授予或限制访问。
- 使用 aws:sourceVpc 条件键基于托管私有终端节点的 VPC 授予或限制访问。

> **Note**
>
> 当请求来自 Amazon VPC 终端节点时，aws:sourceIP条件密钥无效。要限制对 VPC 终端节点的请求，请使用 aws:sourceVpce 或 aws:sourceVpc 条件键。有关更多信息，请参阅《AWS PrivateLink 指南》中的 VPC 终端节点和 VPC 终端节点服务的身份和访问管理。

您可以使用这些全局条件密钥来控制对 P AWS ayment Cryptography 密钥、别名以及不依赖任何特定资源的此类CreateKey操作的访问权限。

例如，以下示例密钥策略仅允许用户在请求使用指定的 VPC 终端节点时使用 AWS 支付加密密钥执行特定的加密操作，从而阻止来自互联网和连接的访问（如果已设置）。当用户向 P AWS ayment Cryptography 发出请求时，会将请求中的 VPC 终端节点 ID 与策略中的aws:sourceVpce条件密钥值进行比较。如果它们不匹配，则请求会被拒绝。

要使用类似的策略，请将占位符 AWS 账户 ID 和 VPC 终端节点 ID 替换为账户的有效值。

```
{
    "Id": "example-key-1",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable IAM policies",
            "Effect": "Allow",
            "Principal": {"AWS":["111122223333"]},
            "Action": ["payment-cryptography:*"],
            "Resource": "*"
        },
        {
            "Sid": "Restrict usage to my VPC endpoint",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "payment-cryptography:Encrypt",
                "payment-cryptography:Decrypt"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:sourceVpce": ""
                }
            }
        }

    ]
}
```

您还可以使用aws:sourceVpc条件密钥根据 VPC 终端节点所在的 VPC 限制对您的 AWS 支付加密密钥的访问。

以下示例密钥策略允许命令仅在 AWS 付款加密密钥来自vpc-12345678时才对其进行管理。此外，它允许使用 AWS 支付密码学密钥进行加密操作的命令，但仅当它们来自时。vpc-2b2b2b2b如果应用程序在一个 VPC 中运行，但您使用第二个隔离的 VPC 执行管理功能，则可以使用这样的策略。

要使用类似的策略，请将占位符 AWS 账户 ID 和 VPC 终端节点 ID 替换为账户的有效值。

```
{
    "Id": "example-key-2",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow administrative actions from vpc-12345678",
            "Effect": "Allow",
            "Principal": {"AWS": "111122223333"},
            "Action": [
                "payment-cryptography:Create*","payment-
cryptography:Encrypt*","payment-cryptography:ImportKey*","payment-
cryptography:GetParametersForImport*",
                "payment-cryptography:TagResource", "payment-
cryptography:UntagResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpc": "vpc-12345678"
                }
            }
        },
        {
            "Sid": "Allow key usage from vpc-2b2b2b2b",
            "Effect": "Allow",
            "Principal": {"AWS": "111122223333"},
            "Action": [
                "payment-cryptography:Encrypt","payment-cryptography:Decrypt"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpc": "vpc-2b2b2b2b"
                }
            }
        },
        {
```

```
            "Sid": "Allow list/read actions from everywhere",
            "Effect": "Allow",
            "Principal": {"AWS": "111122223333"},
            "Action": [
                "payment-cryptography:List*","payment-cryptography:Get*"
            ],
            "Resource": "*",
        }
    ]
}
```

记录您的 VPC 终端节点

AWS CloudTrail 记录使用 VPC 终端节点的所有操作。当向 AWS 支付加密发出的请求使用 VPC 终端节点时，VPC 终端节点 ID 会出现在记录该请求的AWS CloudTrail 日志条目中。您可以使用终端节点 ID 来审核您的 AWS 支付加密 VPC 终端节点的使用情况。

为了保护您的 VPC，如果请求被 VPC 终端节点策略拒绝，但本来会被允许，则不会记录在中AWS CloudTrail。

例如，此示例日志条目记录了使用 VPC 终端节点的GenerateMac请求。vpcEndpointId 字段出现在日志条目的末尾。

```
{
      "eventVersion": "1.08",
      "userIdentity": {
          "principalId": "TESTXECZ5U9M4LGF2N6Y5:",
          "arn": "arn:aws:sts::111122223333:assumed-role//",
          "accountId": "111122223333",
          "accessKeyId": "TESTXECZ5U2ZULLHHMJG",
          "sessionContext": {
              "sessionIssuer": {
                  "type": "Role",
                  "principalId": "TESTXECZ5U9M4LGF2N6Y5",
                  "arn": "arn:aws:iam::111122223333:role/",
                  "accountId": "111122223333",
                  "userName": ""
              },
              "webIdFederationData": {},
              "attributes": {
                  "creationDate": "2024-05-27T19:34:10Z",
                  "mfaAuthenticated": "false"
              },
```

```
            "ec2RoleDelivery": "2.0"
        }
    },
    "eventTime": "2024-05-27T19:49:54Z",
    "eventSource": "payment-cryptography.amazonaws.com",
    "eventName": "CreateKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "172.31.85.253",
    "userAgent": "aws-cli/2.14.5 Python/3.9.16 Linux/6.1.79-99.167.amzn2023.x86_64
 source/x86_64.amzn.2023 prompt/off command/payment-cryptography.create-key",
    "requestParameters": {
        "keyAttributes": {
            "keyUsage": "TR31_M1_ISO_9797_1_MAC_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
            }
        },
        "exportable": true
    },
    "responseElements": {
        "key": {
            "keyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
            "keyAttributes": {
                "keyUsage": "TR31_M1_ISO_9797_1_MAC_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": "A486ED",
            "keyCheckValueAlgorithm": "ANSI_X9_24",
            "enabled": true,
            "exportable": true,
            "keyState": "CREATE_COMPLETE",
            "keyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
            "createTimestamp": "May 27, 2024, 7:49:54 PM",
            "usageStartTimestamp": "May 27, 2024, 7:49:54 PM"
        }
    },
    "requestID": "f3020b3c-4e86-47f5-808f-14c7a4a99161",
    "eventID": "b87c3d30-f3ab-4131-87e8-bc54cfef9d29",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "vpcEndpointId": "",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "-oo28vrvr.controlplane.payment-cryptography.us-
east-1.vpce.amazonaws.com"
    }
}
```


AWS 支付密码学安全最佳实践

AWS Payment Cryptography 支持许多内置安全功能，或者您可以选择实施这些功能，以增强对加密密钥的保护并确保其用于预期目的，包括 IAM 策略、用于完善密钥策略和 IAM 策略的大量策略条件密钥以及有关密钥块的 PCI PIN 规则的内置强制执行。

> **Important**
>
> 提供的这些一般准则并不代表完整的安全解决方案。由于并非所有最佳实践都适用于所有情况，因此这些做法并不是规范性的。

- 密钥使用和使用模式： AWS 支付密码学遵循并强制执行密钥使用和使用模式限制，如ANSI X9 TR 31-2018互操作安全密钥交换密钥区块规范中所述，并符合PCI PIN安全要求18-3。这限制了将单个密钥用于多种目的的能力，并以加密方式将密钥元数据（例如允许的操作）绑定到密钥材料本身。 AWS 支付密码学会自动强制执行这些限制，例如密钥加密密钥（TR31_K0_KEY_ENCRYPTION_KEY）也不能用于数据解密。有关更多信息，请参阅了解 AWS 支付密码学密钥的关键属性。
- 限制对称密钥材料的共享：最多只能与其他一个实体共享对称密钥材料（例如 Pin 加密密钥或密钥加密密钥）。如果需要将敏感材料传输给更多实体或合作伙伴，请创建其他密钥。 AWS 支付密码学从不公开对称密钥材料或非对称私钥材料。
- 使用别名或标签将密钥与某些用例或合作伙伴相关联：别名可用于轻松表示与密钥关联的用例，例如 alias/BIN_12345_CVK，以表示与 BIN 12345 关联的卡片验证密钥。为了提供更大的灵活性，可以考虑创建诸如 bin=12345、use_case=acquiring、country=us,partner=foo 之类的标签。别名和标签还可用于限制访问权限，例如在发布和获取用例之间实施访问控制。
- 实行最低权限访问：IAM 可用于限制对系统而非个人的生产访问，例如禁止个人用户创建密钥或运行加密操作。IAM 还可用于限制对可能不适用于您的用例的命令和密钥的访问权限，例如限制为收单机构生成或验证密码的能力。使用最低权限访问的另一种方法是将敏感操作（例如密钥导入）限制为特定的服务账户。有关示例，请参阅 AWS 支付密码学基于身份的策略示例。

另请参阅

- AWS 支付密码学的身份和访问管理
- IAM 用户指南中的 IAM 安全最佳实践

AWS Payment Cryptography 的合规性验证

作为多个 AWS 合规性计划的一部分，第三方审计员将评估 AWS Payment Cryptography 服务的安全性和合规性。其中包括 SOC、PCI 等。

除了 PCI DSS 之外，AWS Payment Cryptography 还针对多个 PCI 标准进行了评估。其中包括 PCI PIN 安全 (PCI PIN) 和 PCI 点对点 (P2PE) 加密。有关可用的认证和合规指南，请参阅 AWS Artifact。

有关特定合规性计划范围内的 AWS 服务列表，请参阅合规性计划范围内的 Amazon Web Services 服务。有关常规信息，请参阅AWS合规性计划。

您可以使用 AWS Artifact 下载第三方审计报告。有关更多信息，请参阅下载 AWS Artifact 中的报告。

您使用 AWS Payment Cryptography 的合规性责任取决于您数据的敏感度、贵公司的合规性目标以及适用的法律法规。AWS 提供以下资源来帮助满足合规性：

- 安全性与合规性 Quick Start 指南 - 这些部署指南讨论了架构注意事项，并提供了在 AWS 上部署基于安全性和合规性的基准环境的步骤。
- AWS合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。
- 《AWS Config 开发人员指南》中的使用规则评估资源 – AWS Config；评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- AWS Security Hub 此 AWS 服务提供了 AWS 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践。

AWS 支付密码学的身份和访问管理

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以进行身份验证（登录）和授权（有权限）使用 AWS 支付加密资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

受众

您的使用方式 AWS Identity and Access Management (IAM) 会有所不同，具体取决于您在 AWS 支付加密领域所做的工作。

服务用户-如果您使用 AWS 支付密码服务完成工作，则您的管理员会为您提供所需的凭证和权限。当你使用更多的 AWS 支付密码学功能来完成工作时，你可能需要额外的权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 AWS Payment Cryptography 的功能，请参阅 疑难解答 AWS 支付密码学身份和访问权限。

服务管理员 — 如果您负责公司的 AWS 支付密码学资源，则可能拥有对 AWS 支付密码学的完全访问权限。您的工作是确定您的服务用户应访问哪些 AWS 支付密码学功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要详细了解贵公司如何将 IAM 与 AWS 支付加密结合使用，请参阅AWS 支付密码学如何与 IAM 配合使用。

IAM 管理员 — 如果您是 IAM 管理员，则可能需要详细了解如何编写策略来管理对 AWS 付款加密的访问权限。要查看您可以在 IAM 中使用的基于身份的 AWS 支付加密策略示例，请参阅。AWS 支付密码学基于身份的策略示例

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份或通过担 AWS 账户根用户任 IAM 角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （IAM Identity Center）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅 IAM 用户指南中的签署 AWS API 请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的多重身份验证和《IAM 用户指南》中的在 AWS中使用多重身份验证（MFA）。

AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关需要您以根用户身份登录的任务的完整列表，请参阅 IAM 用户指南 中的 需要根用户凭证的任务。

IAM 用户和群组

I AM 用户是您 AWS 账户 内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的何时创建 IAM 用户（而不是角色）。

IAM 角色

I AM 角色是您内部具有特定权限 AWS 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。您可以 AWS Management Console 通过切换角色在中临时担任 IAM 角色。您可以通过调用 AWS CLI 或 AWS API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的 为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的权限集。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。
  - 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。
  - 服务相关角色-服务相关角色是一种与服务相关联的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

- 在 A@@ mazon EC2 上运行的应用程序 — 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时证书。这优先于在 EC2 实例中存储访问密钥。要向 EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建附加到该实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的 使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅《IAM 用户指南》中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管式策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（Amazon S3）存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

访问控制列表 (ACL)

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持 ACL 的服务示例。 AWS WAF要了解有关 ACL 的更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》 中的访问控制列表（ACL）概览。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 - 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCP)-SCP 是 JSON 策略，用于指定组织或组织单位 (OU) 的最大权限。 AWS Organizations AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中的实体（包括每个 AWS 账户根用户实体）的权限。有关 Organizations 和 SCP 的更多信息，请参阅《AWS Organizations 用户指南》中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

AWS 支付密码学如何与 IAM 配合使用

在使用 IAM 管理对 AWS 支付加密的访问权限之前，您应该了解有哪些 IAM 功能可用于 AWS 支付加密。要全面了解 AWS 支付加密和其他 AWS 服务如何与 IAM 配合使用，请参阅 IAM 用户指南中的与 IAM 配合使用的AWS 服务。

主题

AWS 支付密码学基于身份的政策

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。AWS 支付密码学支持特定的操作、资源和条件密钥。要了解在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》 中的 IAM JSON 策略元素参考。

操作

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么 条件 下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

AWS 支付密码学中的策略操作在操作前使用以下前缀:`payment-cryptography:`. 例如，要授予某人执行 AWS 支付加密 `VerifyCardData` API 操作的权限，您需要将该`payment-cryptography:VerifyCardData`操作包含在他们的策略中。策略语句必须包含 `Action` 或 `NotAction` 元素。 AWS Payment Cryptography 定义了自己的一组操作，这些操作描述了您可以使用此服务执行的任务。

要在单个语句中指定多项操作，请使用逗号将它们隔开，如下所示：

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

您也可以使用通配符（*）指定多个操作。例如，要指定以单词 List 开头的所有操作（例如 ListKeys 和 ListAliases），包括以下操作：

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

要查看 AWS 支付加密操作列表，请参阅 IAM 用户指南中的 AWS 支付加密定义的操作。

资源

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么条件 下执行。

Resource JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

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

Payment Cryptography 密钥资源具有以下 ARN：

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

有关 ARN 格式的更多信息，请参阅 Amazon 资源名称 (ARN) 和 AWS 服务命名空间。

例如，要在语句中指定 arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h 实例，请使用以下 ARN：

```
"Resource": "arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h"
```

要指定属于特定账户的所有密钥，请使用通配符 (*)：

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

某些 AWS 支付加密操作（例如用于创建密钥的操作）无法对特定资源执行。在这些情况下，您必须使用通配符（*)。

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

要在单个语句中指定多个资源，请使用逗号，如下所示：

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

示例

要查看基于身份的 AWS 支付加密政策的示例，请参阅。AWS 支付密码学基于身份的策略示例

基于 AWS Payment Cryptography 标签的授权

AWS 支付密码学基于身份的策略示例

默认情况下，IAM 用户和角色没有创建或修改 AWS Payment Cryptography 资源的权限。他们也无法使用 AWS Management Console AWS CLI、或 AWS API 执行任务。IAM 管理员必须创建 IAM 策略，以便为用户和角色授予权限以对所需的指定资源执行特定的 API 操作。然后，管理员必须将这些策略附加到需要这些权限的 IAM 用户或组。

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅《IAM 用户指南》中的 在 JSON 选项卡上创建策略。

主题

- 策略最佳实践
- 使用 AWS Payment Cryptography 控制台
- 允许用户查看他们自己的权限

- 能够访问 AWS 支付密码学的各个方面
- 使用指定的密钥调用 API 的能力
- 明确拒绝资源的能力

策略最佳实践

基于身份的策略决定了某人是否可以在您的账户中创建、访问或删除 AWS 支付加密资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的AWS 托管策略或工作职能的AWS 托管策略。
- 应用最低权限 – 在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定 AWS 服务的（例如）使用的，则也可以使用条件来授予对服务操作的访问权限 AWS CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Acess Analyzer 策略验证。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的 配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实操。

使用 AWS Payment Cryptography 控制台

要访问 AWS 支付加密控制台，您必须拥有一组最低权限。这些权限必须允许您列出和查看有关您 AWS 账户中 AWS 支付加密资源的详细信息。如果您创建的基于身份的策略比所需的最低权限更严格，则无法为具有该策略的实体（IAM 用户或角色）正常运行控制台。

为确保这些实体仍然可以使用 AWS 支付密码控制台，还需要将以下 AWS 托管策略附加到这些实体。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限。

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与您尝试执行的 API 操作相匹配的操作。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "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": "*"
        }
    ]
}
```

能够访问 AWS 支付密码学的各个方面

> **Warning**
>
> 此示例提供了广泛的权限，因此不建议这样操作。相反，请考虑最低特权访问模型。

在此示例中，您希望授予 AWS 账户中的 IAM 用户访问您的所有 AWS 支付加密密钥的权限，以及调用所有 AWS 支付密码学 api（包括 ControlPlane 和 DataPlane 操作）的能力。

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

使用指定的密钥调用 API 的能力

在此示例中，您想向 AWS 账户中的 IAM 用户授予访问您的 AWS 付款加密密钥之一的权限，`arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h`然后在两个 API 中使用该资源，`GenerateCardData`以及`VerifyCardData`。相反，IAM 用户将无权在其他操作（例如 `DeleteKey` 或 `ExportKey`）中使用此密钥

资源可以是前缀为 `key` 的密钥，也可以是前缀为 `alias` 的别名。

```
{
   "Version": "2012-10-17",
   "Statement": [
         {
            "Effect": "Allow",
```

```
                "Action": [
                    "payment-cryptography:VerifyCardData",
                    "payment-cryptography:GenerateCardData"
                ],
                "Resource": [
                    "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
                ]
            }
        ]
    }
```

明确拒绝资源的能力

> **Warning**
>
> 请仔细考虑授予通配符访问权限的影响。考虑改为最低权限模型。

在此示例中，您希望允许 AWS 账户中的 IAM 用户访问您的任何 AWS 付款加密密钥，但希望拒绝对一个特定密钥的权限。用户将有权访问用所有密钥的 `VerifyCardData` 和 `GenerateCardData`，但拒绝语句中指定的密钥除外。

```
{
    "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/
kwapwa6qaifllw2h"
            ]
        }
    ]
}
```

疑难解答 AWS 支付密码学身份和访问权限

在确定特定于 AWS 支付密码学的 IAM 相关问题后，将向本节添加主题。有关 IAM 主题的一般故障排除内容，请参阅 IAM 用户指南的故障排除部分。

监控 AWS Payment Cryptography

监控是保持 AWS Payment Cryptography 和您的其他 AWS 解决方案的可靠性、可用性和性能的重要环节。AWS 提供以下监控工具来监控 AWS Payment Cryptography、在出现错误时进行报告并在适当的时候采取自动化措施：

- Amazon CloudWatch 实时监控您的 AWS 资源以及在 AWS 上运行的应用程序。您可以收集和跟踪指标，创建自定义的控制面板，以及设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。例如，您可以具有 Amazon EC2 实例的 CloudWatch 跟踪 CPU 使用率或其他指标并且在需要时自动启动新实例。有关更多信息，请参阅 Amazon CloudWatch 用户指南。
- Amazon CloudWatch Logs 使您能够监控、存储和访问来自 Amazon EC2 实例、CloudTrail 和其他来源的日志文件。CloudWatch Logs 可以监控日志文件中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。有关更多信息，请参阅 Amazon CloudWatch Logs 用户指南。
- 您可以使用 Amazon EventBridge 自动执行您的 AWS 服务并自动响应系统事件，例如应用程序可用性问题或资源更改。AWS 服务中的事件将近乎实时传输到 EventBridge。您可以编写简单的规则来指示您关注的事件，并指示要在事件匹配规则时执行的自动化操作。有关更多信息，请参阅 Amazon EventBridge 用户指南。
- AWS CloudTrail 捕获由您的 AWS 账户或代表该账户发出的 API 调用和相关事件，并将日志文件传输到您指定的 Simple Storage Service（Amazon S3）存储桶。您可以标识哪些用户和账户调用了 AWS、从中发出调用的源 IP 地址以及调用的发生时间。有关更多信息，请参阅 AWS CloudTrail 用户指南。

> **Note**
>
> AWS CloudTrail 日志支持 CreateKey 等控制面板操作，但不支持生成卡片数据等数据面板操作

使用 AWS CloudTrail 记录 AWS Payment Cryptography API 调用

AWS Payment Cryptography 已与 AWS CloudTrail 集成，后者作为一项服务，提供 AWS Payment Crypto 中由用户、角色或 AWS 服务所执行的操作记录。CloudTrail 将 AWS Payment Cryptography 的所有 API 调用作为事件捕获。捕获的调用包含来自 AWS Payment Cryptography 控制台的调用和对 AWS Payment Cryptography API 操作的代码调用。如果您创建跟踪，则可以将 CloudTrail 事件持续交付到 Amazon S3 存储桶，包括 AWS Payment Cryptography 事件。如果您不配置跟踪，则仍可

在 CloudTrail 控制台中的 Event history（事件历史记录）中查看最新事件。使用 CloudTrail 收集的信息，您可以确定向 AWS Payment Cryptography 发出了什么请求、发出请求的 IP 地址、请求方、请求时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅《AWS CloudTrail 用户指南》。

> **Note**
>
> Cloudtrail 集成当前仅支持控制面板操作。

CloudTrail 中的 AWS Payment Cryptography 信息

在您创建 AWS 账户时，将在该账户上启用 CloudTrail。当 AWS Payment Cryptography 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他 AWS 服务事件一同记录在事件历史记录中。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录 AWS 账户中的事件（包括 AWS Payment Cryptography 的事件），请创建跟踪。通过跟踪记录，CloudTrail 可将日志文件传送至 Simple Storage Service（Amazon S3）存储桶。预设情况下，在控制台中创建跟踪时，此跟踪应用于所有 AWS 区域。此跟踪记录在 AWS 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Simple Storage Service（Amazon S3）桶。此外，您可以配置其他 AWS 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅下列内容：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域接收 CloudTrail 日志文件
- 从多个账户接收 CloudTrail 日志文件

CloudTrail 记录 AWS Payment Cryptography 操作，例如 CreateKey、ImportKey、DeleteKey、ListKeys、TagResource 及所有其他控制面板操作。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 AWS Identity and Access Management（IAM）用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。

- 请求是否由其它 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 AWS Payment Cryptography 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 桶。CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 AWS Payment Cryptography `CreateKey` 操作。

```
      {
    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,
```

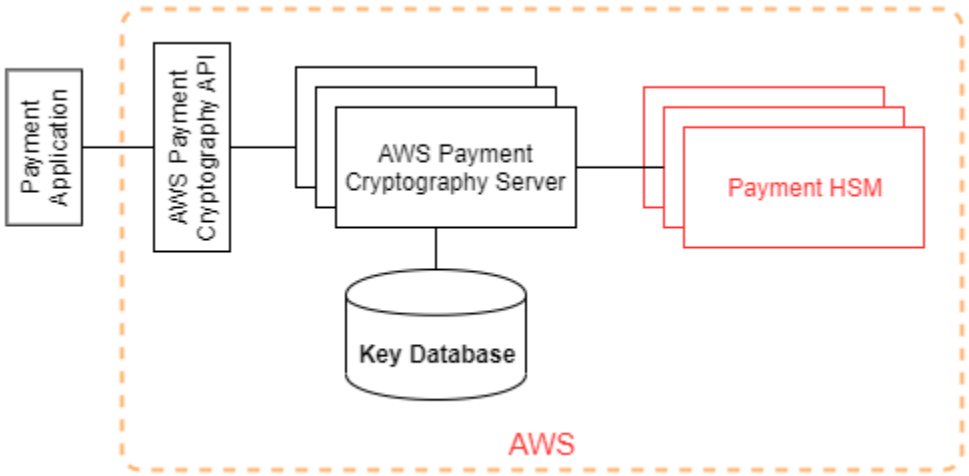
```
      exportable=true,
      enabled=true,
      tags=null),
    eventName=CreateKey,
    userAgent=Coral/Apache-HttpClient5,
    responseElements=CreateKeyOutput(
      key=Key(
        keyArn=arn:aws:payment-cryptography:us-
east-2:111122223333:key/5rplquuwozodpwsp,
          keyAttributes=KeyAttributes(
            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={
    }
  }
}
```

加密详细信息

AWS Payment Cryptography 提供了一个 Web 界面，来生成和管理支付交易的加密密钥。AWSPayment Cryptography 提供标准密钥管理服务和支付交易加密，以及可用于集中管理和审计的工具。本文档详细说明了您在 AWS Payment Cryptography 中可以使用的加密操作，以帮助您评估该服务提供的功能。

AWS Payment Cryptography 包含多个接口（包括 RESTful API、通过 AWS CLI、AWS SDK 和 AWS Management Console），用于请求经过 PCI PTS HSM 验证的分布式硬件安全模块实例集的加密操作。



AWS Payment Cryptography 是一种分层服务，由面向 Web 的 AWS 主机和一层 HSM 组成。这些分层主机的分组形成 AWS Payment Cryptography 堆栈。对 AWS Payment Cryptography 的所有请求都必须使用传输层安全协议（TLS）发出，并在 AWS Payment Cryptography 主机上终止。服务主机仅允许使用提供完美前向保密性的密码套件的 TLS。该服务使用适用于所有其他 AWS API 操作的相同 IAM 凭证和策略机制对您的请求进行身份验证和授权。

AWS Payment Cryptography 服务器通过专有非虚拟网络连接到底层 HSM。服务组件和 HSM 之间的连接使用双向 TLS (mTLS) 进行身份验证和加密。

设计目标

AWS Payment Cryptography 设计为满足以下要求。

- 值得信赖 —密钥的使用受您定义和管理的访问控制策略的保护。没有导出明文 AWS Payment Cryptography 密钥的机制。加密密钥的机密性至关重要。要对 HSM 执行管理操作，需要对具有基

于仲裁的访问控制的角色特定访问权限的多名 Amazon 员工。任何 Amazon 员工都无法访问 HSM 主（或主要）密钥或备份。主密钥无法与不属于 AWS Payment Cryptography 区域的 HSM 同步。所有其他密钥都受到 HSM 主密钥的保护。因此，客户 AWS Payment Cryptography 密钥在客户账户内运行的 AWS Payment Cryptography 服务之外无法使用。

- 低延迟和高吞吐量 — AWS Payment Cryptography 提供的加密操作的延迟和吞吐量级别适用于管理支付加密密钥和处理支付交易。
- 持久性—加密密钥的持久性设计为等同于 中服务最高的持久性。单个加密密钥可以与支付终端、EMV 芯片卡或其他已使用多年的安全加密设备 (SCD) 共享。
- 独立的区域 — AWS 为需要在不同区域限制数据访问或需要遵守数据驻留要求的客户提供独立的区域。可以在 Amazon Web Services 区域内隔离密钥使用。
- 随机数的安全来源 — 由于强加密依赖于真正不可预测的随机数生成，因此 AWS Payment Cryptography 提供优质且经过验证的随机数来源。AWS Payment Cryptography 的所有密钥生成均使用 PCI PTS HSM 列出的 HSM，在 PCI 模式下运行。
- 审计 — AWS Payment Cryptography 在 CloudTrail 日志和通过 Amazon CloudWatch 提供的服务日志中记录加密密钥的使用和管理。您可以使用 CloudTrail 日志来检查加密密钥的使用情况，包括与您共享密钥的账户的密钥使用情况。 AWSPayment Cryptography 由第三方评估机构根据适用的 PCI、卡品牌和区域支付安全标准进行审核。AWS Artifact 上提供了证明和责任共担指南。
- 弹性 — AWS Payment Cryptography 可根据您的需求进行横向扩展。AWS Payment Cryptography 不是预测和预留 HSM 容量，而是按需提供支付加密。 AWSPayment Cryptography 负责维护 HSM 的安全性和合规性，以提供足够的容量来满足客户的峰值需求。

基本原理

本章中的主题描述了 AWS 支付密码学的加密原语及其用途。它们还介绍了服务的基本元素。

主题

- 密钥导出
- 每笔交易派生唯一密钥 (DUKPT) 协议
- 密钥层次结构

加密基元

AWS Payment Cryptography 使用可参数化的标准加密算法，因此应用程序可以实现其用例所需的算法。这组加密算法由 PCI、ANSI X9、EMVco 和 ISO 标准定义。所有加密均由在 PCI 模式下运行的 PCI PTS HSM 标准列出的 HSM 执行。

熵和随机数生成

AWS 支付密码学密钥生成是在 AWS 支付密码学 HSM 上执行的。HSM 实现一个随机数生成器，该生成器满足 PCI PTS HSM 对所有支持的密钥类型和参数的要求。

对称密钥操作

支持 ANSI X9 TR 31、ANSI X9.24 和 PCI PIN 附录 C 中定义的对称密钥算法和密钥强度：

- 哈希函数 — 来自 SHA2 和 SHA3 系列的算法，其输出大小大于 2551。与 PCI 之前的 PTS POI v3 终端向后兼容除外。
- 加密和解密 — 密钥大小大于或等于 128 位的 AES，或密钥大小大于或等于 112 位（2 个密钥或 3 个密钥）的 TDEA。
- 消息验证代码 (MAC) 使用 AES 的 CMAC 或 GMAC，以及具有批准的哈希函数且密钥大小大于或等于 128 的 HMAC。

AWS 支付密码学使用 AES 256 作为 HSM 主密钥、数据保护密钥和 TLS 会话密钥。

非对称密钥操作

支持 ANSI X9 TR 31、ANSI X9.24 和 PCI PIN 附录 C 中定义的非对称密钥算法和密钥强度：

- 经批准的密钥建立方案 — 如 NIST SP800-56A（基于 ECC/FCC2 的密钥协议）、NIST SP800-56B（基于 IFC 的密钥协议）和 NIST SP800-38F（基于 AES 的密钥加密/封装）中所述。

AWS Payment Cryptography 主机仅允许使用 TLS 连接服务，其密码套件可提供完美的前向保密。

密钥存储

AWS 支付密码学密钥受 HSM AES 256 主密钥保护，并存储在加密数据库的 ANSI X9 TR 31 密钥块中。该数据库被复制到 AWS 支付密码服务器上的内存数据库。

根据 PCI PIN 安全规范附录 C，AES 256 密钥的强度等于或强于：

- 3 密钥 TDEA
- RSA 15360 位
- ECC 512 位
- DSA、DH、和 MQV 15360/512

使用对称密钥导入密钥

AWS Payment Cryptography 支持导入带有对称密钥或公钥的密码和密钥块，其对称密钥加密密钥 (KEK) 的强度与受保护的密钥一样强或更强。

使用非对称密钥导入密钥

AWS Payment Cryptography 支持导入带有对称或公钥的密码和密钥块，这些密钥由私钥加密密钥 (KEK) 保护，该密钥的强度与受保护的密钥一样强或更强，用于导入。用于解密的公钥必须具有由客户信任的机构颁发的证书，以确保其真实性和完整性。

P AWS ayment Cryptography 提供的公共 KEK 具有证书颁发机构 (CA) 的身份验证和完整性保护，经证实符合 PCI PIN 安全和 PCI P2PE 附录 A。

密钥导出

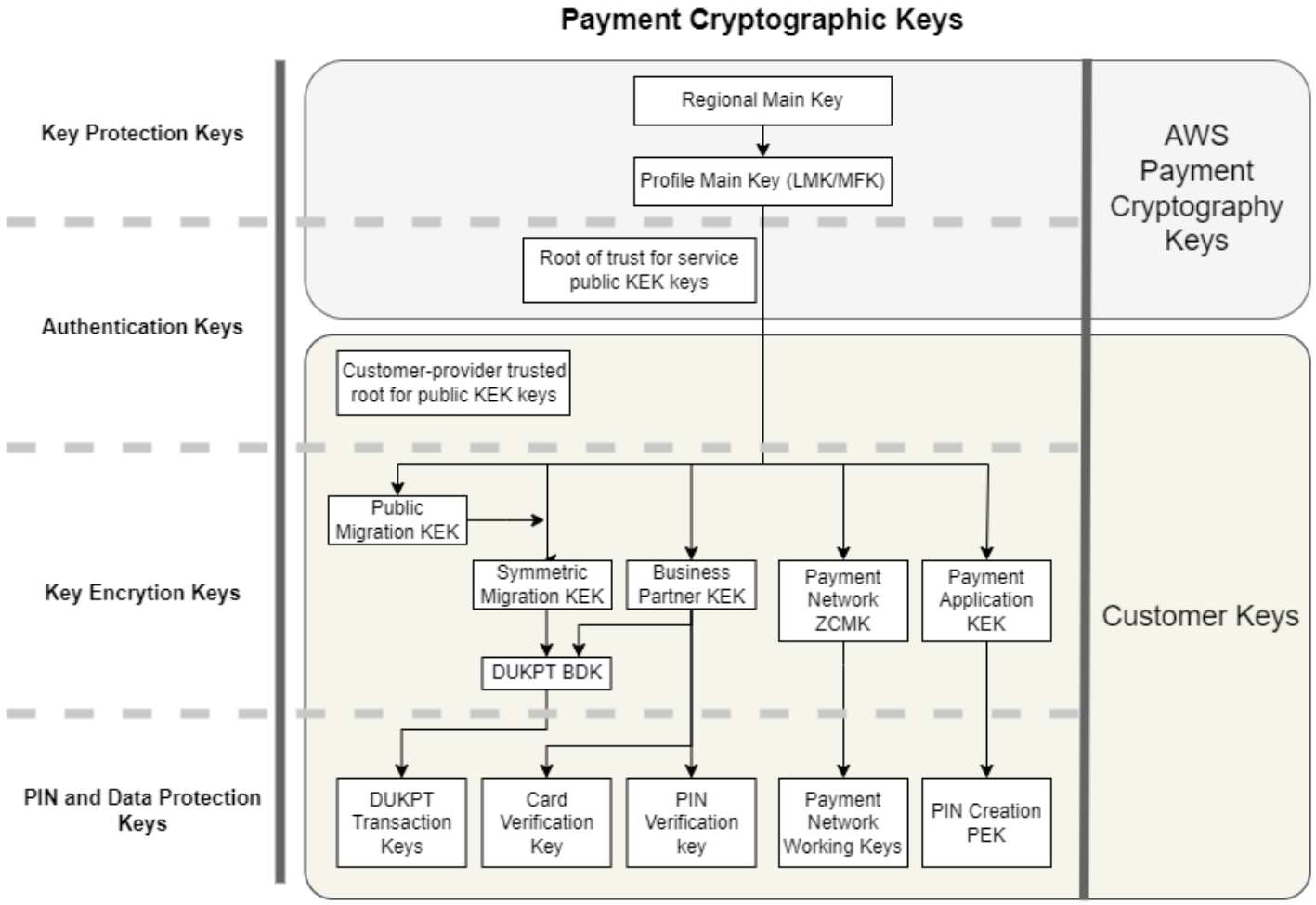
密钥可以导出并使用相应的 KeyUsage 密钥进行保护，这些密钥的强度与要导出的密钥一样强或强。

每笔交易派生唯一密钥 (DUKPT) 协议

AWS 支付密码学支持 TDEA 和 AES 基础派生密钥 (BDK)，如 ANSI X9.24-3 所述。

密钥层次结构

Pay AWS ment Cryptography 密钥层次结构可确保密钥始终受到与其保护的密钥一样强或更强的密钥保护。



AWS 支付密码学密钥用于服务内的密钥保护：

键	描述
区域主密钥	保护用于加密处理的虚拟 HSM 映像或配置文件。此密钥仅存在于 HSM 和安全备份中。
个人资料主密钥	顶级客户密钥保护密钥，传统上称为客户密钥的本地主密钥 (LMK) 或主文件密钥 (MFK)。此密钥仅存在于 HSM 和安全备份中。配置文件根据支付用例安全标准的要求定义不同的 HSM 配置。

键	描述
AWS 支付密码学公钥加密密钥 (KEK) 密钥的信任根	受信任的根公钥和证书，用于验证和验证 Payment Cryptography 提供的用于使用 AWS 非对称密钥导入和导出密钥的公钥。

客户密钥按用于保护其他密钥的密钥和保护支付相关数据的密钥进行分组。以下是两种类型的客户密钥示例：

键	描述
客户提供的公共 KEK 密钥的可信根	您提供的公钥和证书作为信任根，用于验证和核实您为使用非对称密钥导入和导出而提供的公钥。
密钥加密密钥 (KEK)	KEK 仅用于加密其他密钥，以便在外部密钥存储和 AWS 支付密码学、业务合作伙伴、支付网络或组织内的不同应用程序之间进行交换。
每笔交易派生唯一密钥 (DUKPT) 基本派生密钥 (BDK)	BDK 用于为每个支付终端创建唯一密钥，并将交易从多个终端转换为单个收单银行或收单机构工作密钥。PCI 点对点加密 (P2PE) 要求的最佳实践是，不同的终端模型、密钥注入或初始化服务或其他分段使用不同的 BDK，以限制破坏 BDK 的影响。
支付网络区域控制主密钥 (ZCMK)	ZCMK，也称为区域密钥或区域主密钥，由支付网络提供，用于建立初始工作密钥。
DUKPT 交易密钥	为 DUKPT 配置的支付终端会为终端和交易生成唯一的密钥。接收交易的 HSM 可以根据终端标识符和交易序列号确定密钥。
卡片数据准备密钥	EMV 发卡机构主密钥、EMV 卡密钥和验证值以及卡片个性化数据文件保护密钥用于为单张卡片创建数据，供卡片个性化提供商使用。发卡银行

键	描述
	或发卡机构也使用这些密钥和加密验证数据来验证卡数据，作为授权交易的一部分。
卡片数据准备密钥	EMV 发卡机构主密钥、EMV 卡密钥和验证值以及卡片个性化数据文件保护密钥用于为单张卡片创建数据，供卡片个性化提供商使用。发卡银行或发卡机构也使用这些密钥和加密验证数据来验证卡数据，作为授权交易的一部分。
支付网络工作密钥	这些密钥通常被称为发行方工作密钥或收单方工作密钥，用于对发送到支付网络或从支付网络接收的交易进行加密。这些密钥由网络频繁轮换，通常每天或每小时轮换。这些是用于 PIN/借记交易的 PIN 加密密钥 (PEK)。
个人识别码 (PIN) 加密密钥 (PEK)	创建或解密 PIN 块的应用程序使用 PEK 来防止存储或传输明文 PIN。

内部操作

本主题介绍服务实现的内部要求，以保护全球分布式且可扩展的支付加密和密钥管理服务的客户密钥和加密操作。

HSM 规格和生命周期

AWS 支付密码学使用大量市售的 HSM。HSM 已通过 FIPS 140-2 3 级验证，还使用 PCI 安全标准委员会批准的 PCI PTS 设备列表中列出的固件版本和安全策略，符合 PCI HSM v3 的要求。PCI PTS HSM 标准包括对 HSM 硬件的制造、运输、部署、管理和销毁的附加要求，这些要求对于支付安全性和合规性非常重要，但 FIPS 140 并未解决。

所有 HSM 均在 PCI 模式下运行，并配置了 PCI PTS HSM 安全策略。仅启用支持 AWS 支付加密用例所需的功能。 AWS Payment Cryptography 不提供打印、显示或返回明文 PIN。

HSM 设备物理安全

该服务只能使用在交付前由制造商通过 AWS 支付密码学证书颁发机构 (CA) 签署的设备密钥的 HSM。AWS 支付密码学是制造商 CA 的子 CA，是 HSM 制造商和设备证书的信任根源。制造商的 CA 实施了

ANSI TR 34，并已证明符合 PCI PIN 安全附录 A 和 PCI P2PE 附录 A。制造商验证所有带有由 AWS 支付密码学 CA 签署的设备密钥的 HSM 均已运送到 AWS 的指定接收方。

根据 PCI PIN 安全性的要求，制造商通过与 HSM 发货不同的通信通道提供序列号列表。在将 HSM 安装到 AWS 数据中心的过程中的每个步骤都会检查这些序列号。最后， AWS 支付密码操作员将已安装的 HSM 列表与制造商的列表进行验证，然后再将序列号添加到允许接收 AWS 支付加密密钥的 HSM 列表中。

HSM 始终处于安全存储或双重控制之下，其中包括：

- 从制造商运送到 AWS 机架组装设施。
- 在机架组装期间。
- 从机架装配设施运送到数据中心。
- 接收并安装到数据中心安全处理室。HSM 机架通过卡门禁锁、报警门传感器和摄像头实现双重控制。
- 操作期间。
- 在停用和销毁期间。

为每个 HSM 维护和监控完整的 chain-of-custody个人问责制。

HSM 初始化

只有通过序列号、制造商安装的设备密钥和固件校验和验证其身份和完整性后，HSM 才会作为 AWS 支付密码群的一部分进行初始化。在验证 HSM 的真实性和完整性后，对其进行配置，包括启用 PCI 模式。然后，建立 AWS 支付密码区域主密钥和配置文件主密钥，HSM 可供该服务使用。

HSM 服务和维修

HSM 具有无需违反设备加密边界的可维护组件。这些组件包括冷却风扇、电源和电池。如果 HSM 或 HSM 机架内的其他设备需要维修，则在机架打开的整个期间都将保持双重控制。

HSM 停用

停用是由于 HSM end-of-life 的故障造成的。从逻辑上讲，HSM 在从机架上移除之前会被归零，如果可以正常运行，则在 AWS 数据中心的安全处理室中销毁。在销毁之前，其永远不会退回制造商进行维修，也不会用于其他目的，也不会以其他方式从安全的加工室中移走。

HSM 固件更新

如果更新与安全有关，或者确定客户可以从新版本中的功能中受益，则在需要时应用 HSM 固件更新，以与 PCI PTS HSM 和 FIPS 140-2（或 FIPS 140-3）列出的版本保持一致。 AWS 支付密码学 HSM 运行 off-the-shelf 固件，与 PCI PTS HSM 列出的版本相匹配。使用经过 PCI 或 FIPS 认证的固件版本对新固件版本进行完整性验证，然后在向所有 HSM 推出之前测试其功能性。

操作员访问权限

在极少数情况下，在正常操作期间从 HSM 收集的信息不足以识别问题或计划更改，操作员可以非控制台访问 HSM 进行故障排除。执行以下步骤：

- 已制定并批准了故障排除活动，并安排了非控制台会话。
- HSM 已从客户处理服务中删除。
- 在双重控制下，主密钥被删除。
- 允许操作员通过非控制台访问 HSM，以在双重控制下执行批准的故障排除活动。
  - 非控制台会话终止后，在 HSM 上执行初始配置过程，返回标准固件和配置，然后同步主密钥，再将 HSM 交还给服务客户。
  - 会话记录记录在变更跟踪中。
  - 从会话中获得的信息用于规划未来的更改。

对所有非控制台访问记录进行审查，以确定流程合规性以及 HSM 监控、 non-console-access 管理流程或操作员培训可能发生的变化。

密钥管理

一个区域中的所有 HSM 都与区域主密钥同步。区域主密钥可以保护至少一个配置文件主密钥。配置文件主密钥可保护客户密钥。

所有主密钥均由 HSM 生成，并使用非对称技术通过对称密钥分配进行分发，符合 ANSI X9 TR 34 和 PCI PIN 附录 A。

主题

- 配置文件主密钥同步
- 配置文件主密钥轮换
- 保护
- 持久性
- 通信安全
- 客户密钥的管理
- 日记账记录和监控

生成

AES 256 位主密钥是使用 PCI PTS HSM 随机数生成器在为服务 HSM 实例集配置的 HSM 上生成的。

区域主密钥同步

HSM 区域主密钥由跨区域实例集的服务同步，机制由 ANSI X9 TR-34 定义，其中包括：

- 使用密钥分配主机 (KDH) 和密钥接收设备 (KRD) 密钥和证书进行相互身份验证，以提供公钥的身份验证和完整性。
- 证书由符合 PCI PIN 附录 A2 要求的证书颁发机构 (CA) 签名，但适用于保护 AES 256 位密钥的非对称算法和密钥强度除外。
- 分布式对称密钥的识别和密钥保护符合 ANSI X9 TR-34 和 PCI PIN 附录 A1，但适用于保护 AES 256 位密钥的非对称算法和密钥强度除外。

区域主密钥是通过以下方式为已通过身份验证和为区域配置的 HSM 建立的：

- 主密钥是在该区域的 HSM 上生成的。该 HSM 被指定为密钥分配主机。
- 该区域中所有已配置的 HSM 都会生成 KRD 身份验证令牌，其中包含 HSM 的公钥和不可重播的身份验证信息。
- 在 KDH 验证 HSM 的身份和接收密钥的许可后，KRD 令牌将添加到 KDH 允许列表中。
- KDH 为每个 HSM 生成一个可验证的主密钥令牌。令牌包含 KDH 身份验证信息和加密的主密钥，这些密钥只能在为其创建的 HSM 上加载。
- 每个 HSM 都会收到为其构建的主密钥令牌。在验证 HSM 自己的身份验证信息和 KDH 身份验证信息后，主密钥由 KRD 私钥解密并加载到主密钥中。

如果必须将单个 HSM 与某个区域重新同步：

- 它会经过重新验证并配备固件和配置。
- 如果是该地区的新用户：
  - HSM 会生成 KRD 身份验证令牌。
  - KDH 将令牌添加到其允许列表中。
  - KDH 为 HSM 生成主密钥令牌。
  - HSM 加载主密钥。
  - HSM 可供该服务使用。

这可以保证：

- 只有在一个区域内通过 AWS 支付加密处理验证的 HSM 才能接收该地区的主密钥。
- 只有来自 AWS 支付密码学 HSM 的主密钥才能分发给队列中的 HSM。

区域主密钥轮换

区域主密钥将在加密期限到期时轮换，以防万一发生可疑的密钥泄露事件，或者在确定会影响密钥安全性的服务更改之后进行轮换。

与初始配置一样，将生成和分发新的区域主密钥。保存的配置文件主密钥必须转换为新的区域主密钥。

区域主密钥轮换不会影响客户处理。

配置文件主密钥同步

配置文件主密钥受区域主密钥保护。这会将配置文件限制在特定区域。

相应地配备了配置文件主密钥：

- 配置文件主密钥是在已同步区域主密钥的 HSM 上生成的。
- 配置文件主密钥与配置文件配置和其他上下文一起存储和加密。
- 该区域中任何具有区域主密钥的 HSM 都将该配置文件用于客户加密功能。

配置文件主密钥轮换

配置文件主密钥将在加密期限到期、可疑密钥泄露后或在确定会影响密钥安全性的服务更改后进行轮换。

轮换步骤：

- 与初始配置一样，将生成新的配置文件主密钥作为待处理的主密钥进行分发。
- 后台流程将客户密钥材料从已建立的配置文件主密钥转换为待处理的主密钥。
- 使用待处理密钥加密所有客户密钥后，待处理密钥将升级为配置文件主密钥。
- 后台流程会删除受过期密钥保护的客户密钥材料。

配置文件主密钥轮换不会影响客户处理。

保护

密钥仅依赖于密钥层次结构进行保护。保护主密钥对于防止所有客户密钥丢失或泄露至关重要。

区域主密钥只能从备份中恢复到为服务进行身份验证和配置的 HSM。这些密钥只能存储为来自特定 HSM 的特定 KDH 的相互身份验证的加密主密钥令牌。

配置文件主密钥与按区域加密的配置文件配置和上下文信息一起存储。

客户密钥存储在密钥块中，由配置文件主密钥保护。

所有密钥都只存在于 HSM 中，或者由另一个具有同等或更强加密强度的密钥保护。

持久性

即使在通常会导致中断的极端情况下，也必须提供用于交易加密和业务功能的客户密钥。 AWS 支付密码学利用跨可用区域和区域的多级冗余模型。 AWS 如果客户要求支付加密操作的可用性和耐久性高于服务所能提供的范围，则应实施多区域架构。

HSM 身份验证和主密钥令牌已保存，如果必须重置 HSM，则可用于恢复主密钥或与新的主密钥同步。令牌已存档，仅在需要时在双重控制下使用。

通信安全

外部

AWS Payment Cryptography API 端点符合 AWS 安全标准，包括 1.2 或以上的 TLS 和用于请求身份验证和完整性的签名版本 4。

传入的 TLS 连接在网络负载均衡器上终止，并通过内部 TLS 连接转发给 API 处理程序。

Internal

服务组件之间以及服务组件与其他 Amazon Web Service 服务之间的内部通信由 TLS 使用强大的加密技术进行保护。

HSM 位于只能通过服务组件访问的专用非虚拟网络上。HSM 和服务组件之间的所有连接均通过 TLS 1.2 或更高版本的相互 TLS (mTLS) 进行保护。TLS 和 mTLS 的内部证书由 Amazon Certificate Manager 使用 AWS 私有证书颁发机构进行管理。监控内部 VPC 和 HSM 网络是否存在无异常的活动和配置更改。

客户密钥的管理

在 AWS，客户信任是我们的首要任务。您可以完全控制自己在 Amazon Web Services 账户下的服务中上传或创建的密钥，并负责配置对密钥的访问。

AWS Payment Cryptography 对服务管理的密钥的 HSM 物理合规性和密钥管理负全部责任。这需要拥有和管理 HSM 主密钥，并将受保护的客户密钥存储在 AWS 付款密码学密钥数据库中。

客户密钥空间分离

AWS Payment Cryptography 对所有密钥的使用都强制执行密钥政策，包括将委托人限制为拥有密钥的账户，除非明确与其他账户共享密钥。

备份和恢复

区域的密钥和密钥信息由 AWS备份到加密存档中。存档需要双重控制 AWS 才能恢复。

密钥块

所有密钥都存储在 ANSI X9 TR-31 格式的密钥块中。

密钥可以从支持的密码或其他密钥块格式导入到服务中。 ImportKey同样，如果密钥可导出，也可以将其导出为其他密钥块格式或密钥导出配置文件支持的密码。

密钥用途

密钥的使用仅限于服务 KeyUsage 所配置的。如果密钥使用、使用模式或所请求的加密操作算法不当，该服务将使任何请求失败。

密钥交换关系

PCI PIN Security 和 PCI P2PE 要求共享加密 PIN 的密钥的组织（包括用于共享这些密钥的 KEK）不得与任何其他组织共享这些密钥。最佳做法是，对称密钥仅在两方之间共享，包括在同一个组织内。这可以最大限度地减少可疑密钥泄露的影响，从而强制更换受影响的密钥。

即使业务案例需要在超过 2 方之间共享密钥，也应将参与方数量保持在最低数量。

AWS Payment Cryptography 提供了密钥标签，可用于在这些要求范围内跟踪和强制使用密钥。

例如，可以通过为与该服务提供商共享的所有密钥设置“KIF”=“POSStation”来识别不同密钥注入设施的 KEK 和 BDK。另一个例子是将与支付网络共享的密钥标记为 “网络” = “PayCard”。使用标记，您可以创建访问控制并创建审计报告，以强制执行和演示您的密钥管理实践。

删除密钥

DeleteKey 将数据库中的密钥标记为在客户可配置的时间段后删除。在这段时间之后，密钥将被不可挽回地删除。这是一种防止意外或恶意删除密钥的安全机制。标记为删除的密钥不可用于任何操作，除 RestoreKey了。

删除的密钥将在删除后的服务备份中保留 7 天。在此期间，它们无法修复。

属于已关闭 Amazon Web Services 账户的密钥被标记为删除。如果在达到删除期限之前重新激活帐户，则所有标记为删除的密钥都将被恢复，但会被禁用。您必须重新启用才能将其用于加密操作。

密钥共享

密钥可以使用 AWS Resource Access Manager (https://docs.aws.amazon.com/ARG/index.html) 与您组织内部或外部的其他账户共享。密钥可以分组到资源共享中，然后与账户或账户中的特定 IAM 用户和角色共享。您可以为每个资源共享指定使用权限。共享权限受密钥资源策略的限制。共享密钥不允许执行受其自身策略限制的操作。共享许可可以随时撤回。

日记账记录和监控

内部服务日志包括：

- CloudTrail 该服务发出的 AWS 服务调用的日志
- CloudWatch 两个事件的日志都直接记录到 CloudWatch 日志或从 HSM 的事件中
- 来自 HSM 和服务系统的日志文件
- 日志档案

所有日志源都会监控和过滤敏感信息，包括有关密钥的信息。日志会经过系统性审查，以确保其中包含或不包含敏感客户信息。

对日志的访问仅限于完成工作角色所需的个人。

所有日志均按照 AWS 日志保留策略进行保留。

客户操作

AWS 根据PCI标准，支付密码学对HSM的物理合规性负全部责任。该服务还提供安全的密钥存储，并确保密钥只能用于 PCI 标准允许、且由您在创建或导入时指定的用途。您负责配置关键属性和访问权限，以利用服务的安全性和合规性功能。

主题

生成密钥

创建密钥时，您可以设置服务用于强制执行密钥的合规使用的属性：

- 算法和密钥长度
- 使用量
- 可用性和过期

用于基于属性的访问权限控制（ABAC）的标签用于限制与特定合作伙伴或应用程序一起使用的密钥，也应在创建过程中设置。请务必包括限制允许删除或更改标签的角色的政策。

您应确保在创建密钥之前设置了确定可以使用和管理密钥的角色的策略。

> **Note**
>
> CreateKey 命令上的 IAM 策略可用于强制执行和演示对密钥生成的双重控制。

导入密钥

导入密钥时，服务使用密钥块中的加密绑定信息来设置强制使用密钥的属性。设置基本密钥上下文的机制是使用通过源 HSM 创建、并受共享或非对称 KEK 保护的密钥块。这符合 PCI PIN 要求，并保留了源应用程序的用法、算法和密钥强度。

除了密钥块中的信息外，还必须在导入时建立重要的密钥属性、标签和访问控制策略。

使用密码导入密钥不会传输源应用程序中的密钥属性。您必须使用此机制相应地设置属性。

通常，密钥是使用明文组件进行交换的，由密钥保管人传输，然后加载仪式，在安全房间中实现双重控制。 AWS 支付密码学不直接支持这一点。API 将导出带有证书的公钥，该证书可由您自己的 HSM 导入，以导出可由服务导入的密钥块。允许使用您自己的 HSM 来加载明文组件。

您应该使用密钥检查值 (KCV) 来验证导入的密钥是否与源密钥匹配。

ImportKey API 上的 IAM 策略可用于强制执行和演示对密钥导入的双重控制。

导出密钥

与合作伙伴或本地应用程序共享密钥可能需要导出密钥。使用密钥块进行导出可以使用加密的密钥材料维护基本密钥上下文。

密钥标签可用于限制向共享相同标签和值的 KEK 导出密钥。

AWS 支付密码学不提供或显示明文密钥组件。这需要密钥保管人直接访问 PCI PTS HSM 或经过 ISO 13491 测试的安全加密设备 (SCD) 以进行显示或打印。您可以使用 SCD 建立非对称 KEK 或对称 KEK，以便在双重控制下进行明文密钥组件创建仪式。

应使用密钥检查值 (KCV) 来验证目标 HSM 导入的密钥是否与源密钥匹配。

删除密钥

您可以使用删除密钥 API 安排在您配置的一段时间后删除密钥。在此之前，密钥是可以恢复的。一旦密钥被删除，就会从服务中永久移除。

DeleteKey API 上的 IAM 策略可用于强制执行和演示对密钥删除的双重控制。

轮换 密钥

可以使用密钥别名实现密钥轮换的效果，方法是创建或导入新密钥，然后修改密钥别名以引用新密钥。根据您的管理惯例，旧密钥将被删除或禁用。

的配额 AWS Payment Cryptography

您的 Amazon Web Services 账户对于每个 AWS 服务都具有默认配额（以前称为限制）。除非另有说明，否则，每个配额是区域特定的。您可以请求增加某些限额，但其他一些限额无法增加。

名称	默认值	可调整	描述
别名	每个受支持的区域：2000 个	是	在当前区域内的此账户中，您可以拥有的别名的最大数量。
控制面板请求的组合速率	每个支持的区域：每秒 5 个	是	在当前区域内的此账户中，您每秒可以发出的控制面板请求的最大数量。此配额适用于所有控制面板操作的组合。
数据面板请求的合并速率（非对称）	每个支持的区域：每秒 20 个	是	在当前区域中，您在此账户中使用非对称密钥进行数据平面操作的每秒最大请求次数。此配额适用于所有数据面板操作的总和。
数据面板请求的组合速率（对称）	每个支持的区域：每秒 500 个	是	在当前区域中，您在此账户中使用对称密钥进行数据平面操作的每秒最大请求次数。此配额适用于所有数据面板操作的总和。
键	每个受支持的区域：2000 个	是	在当前区域内的此账户中，您可以拥有的密钥的最大数量，不包括已删除的密钥。

《 AWS 支付密码学用户指南》的文档历史记录

下表描述了 AWS 支付密码学的文档版本。

变更	说明	日期
功能发布	添加有关 VPC 终端节点 (PrivateLink) 和 ICvV 示例的信息。	2024 年 5 月 30 日
功能发布	添加了有关使用 RSA 导入/导出密钥和导出 DUKPT IPEK/IK 密钥的新功能的信息。	2024 年 1 月 15 日
初始版本	《 AWS 支付密码学用户指南 》的首次发布	2023 年 6 月 8 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。