# 密钥管理系统（KMS）

## 产品文档

腾讯云TCE

版权所有：腾讯云金融专区

# 文档目录

- 产品简介
  - 产品概览
  - 产品功能
  - 产品优势
  - 使用场景
  - 名词解释
- 快速入门
  - 入门概述
  - 创建密钥
  - 管理密钥
  - 加密密钥
  - 密钥归档
- 最佳实践
  - 敏感信息加密
  - 信封加密
  - 访问控制
    - 概述
    - KMS访问控制策略示例
    - KMS API操作支持的资源级权限
  - 外部密钥导入
    - 概述
    - 操作指南
  - 国密 Encryption SDK
    - SDK 概览
    - SDK 接入指南
    - SDK 接口
      - 旗舰版 C 接口文档
      - 旗舰版 GO 接口文档
      - 旗舰版Python2接口文档
      - 旗舰版Python3接口文档
- API文档
  - 密钥管理系统（kms）
    - 版本（2019-01-18）
      - API概览
      - 调用方式
        - 接口签名v1
        - 接口签名v3
        - 请求结构
        - 返回结果
        - 公共参数
      - 其他接口
        - 获取TSM SDK授权的各个版本License
      - 密钥相关接口

密钥归档
绑定密钥和云产品资源的使用关系
取消密钥归档
取消CMK计划删除操作
创建主密钥
解密
删除导入的密钥材料
获取主密钥属性
获取多个主密钥属性
禁用主密钥
禁止密钥轮换
批量禁用主密钥
启用主密钥
开启密钥轮换
批量启动主密钥
加密
生成非对称密钥对
生成数据密钥
随机数生成接口
查询密钥轮换状态
获取导入主密钥（CMK）材料的参数
获取支持的地域列表
查询服务状态
导入密钥材料
列出当前Region支持的加密方式
获取主密钥列表详情
获取主密钥列表
密文刷新
CMK计划删除接口
签名
解绑CMK和云资源的关联关系
修改别名
修改主密钥描述信息
非对称密钥相关接口
非对称密钥RSA解密
非对称密钥Sm2解密
加密
获取非对称密钥的公钥
数据结构
错误码

# 产品简介
# 产品概览

最近更新时间: 2025-02-18 16:02:00

密钥管理系统（Key Management Service，KMS）是一款安全管理类服务，使用经过第三方认证的硬件安全模块 HSM（Hardware Security Module） 来生成和保护密钥。帮助用户轻松创建和管理密钥，满足用户多应用多业务的密钥管理需求，符合监管和合规要求。
下图所示为密钥管理系统（KMS）产品架构图：

# 产品功能

最近更新时间: 2025-02-18 16:02:00

### 安全合规

密钥管理系统底层使用国家密码局或 FIPS-140-2 认证的硬件安全模块（HSM）来保护密钥的安全，确保密钥的保密性、完整性和可用性。

### 托管式密钥管理

密钥管理系统为您提供了丰富的管理功能，包括密钥创建、启用、禁用、轮换设置、别名设置、密钥归档、查看密钥详情、修改相关信息等功能 。您可以依托 腾讯云金融专区 密钥管理系统轻松的创建、保护以及执行您的各项密钥管理策略。

### 密钥轮换

提供 CMK 密钥交换能力，默认关闭，由用户设置是否打开，开启后 CMK 会一年交换一次，密钥交换由密钥管理系统系统管理，对上透明，交换后使用该 CMK 加密的旧的密文依然可以解密。新的加密则使用新的 CMK。

### 权限控制

与 腾讯云金融专区 访问管理集成，通过身份管理和策略管理控制哪些账户，哪些角色可以访问或管理您的敏感密钥。

### 内置审计

与 腾讯云金融专区 审计集成，可记录所有 API 请求，包括密钥管理操作和密钥使用情况。

### 稳定可靠

采用多机房分布式集群化的业务部署和热备份，底层 HSM 设备采用双机房冷备份部署，确保密钥管理系统的高可用性。

### 无缝集成服务

密钥管理系统与对象存储、分布式数据库、云硬盘等服务的加密特性无缝集成，您可以轻松地应用密钥管理系统来管理这些服务内所存储数据的加密。

### 集中化密钥管理

您可以通过 API、SDK、云产品等多种方式调用并集成密钥管理系统，实现对各类应用程序的密钥的集中管理，无论这些业务应用在 腾讯云金融专区 内或是 腾讯云金融专区 外。

### 敏感数据加密

敏感信息加密是密钥管理系统核心的能力，实际应用中主要用来保护服务器硬盘上敏感数据的安全（小于4KB），如密钥、证书、配置文件等。

### 信封加密

信封加密（Envelope Encryption）是一种应对海量数据的高性能加解密方案。在密钥管理系统信封加密场景中，只需要传输数据加密密钥 DEK 到密钥管理系统服务端（通过 CMK 进行加解密），所有的业务数据都是采用高效的本地对称加密处理，对业务的访问体验影响很小。

### 加密算法

密钥管理系统 KMS 支持对称加密算法：SM4，AES256；非对称加密算法：RSA，SM2，ECC；您可根据实际业务情况自主选择。

版权所有：腾讯云金融专区

# 产品优势

最近更新时间: 2025-02-18 16:02:00

### 安全合规

KMS 使用经过第三方认证的硬件安全模块（HSM）来生成和保护密钥，安全和质量控制已通过多种合规性计划认证。您的主密钥的的创建、管理等操作都将在合规的 HSM 硬件中进行，任何人都无法获取到您的明文主密钥。

### 高可用

在服务架构方层面，KMS 服务通过单地域多机房提供可靠性，其底层使用的 HSM 设备也采用多机房集群化部署，并提供双机房冷备份设备，确保服务的高可用性。在接入层面，KMS 通过云 API3.0 提供对外接入服务。云 API3.0 分地域部署，接入域名提供统一域名和地域独立域名两种方式，确保服务接入的高可用性。

### 集中化密钥管理

您可以通过 API、SDK 及已经对接的云产品接入 KMS 服务，并使用 KMS 集中管理您业务应用的密钥策略。

### 成本低廉

无须购买专门的硬件加密设备，一键部署，按量付费，我们将提供所有后端服务维护。

版权所有：腾讯云金融专区

# 使用场景

最近更新时间: 2025-02-18 16:02:00

密钥管理系统 KMS 解决用户敏感数据加密需求，满足安全合规，同时帮助不同行业数据加密痛点问题。

### 金融等行业敏感数据保护

痛点：金融等行业机构任何的通信和存储数据都具有高价值性和高保密性，需要考虑加密的安全性及合规性。 方案：通过信封加密对协议通信内容、重要文件和资料提供加密服务及密钥保护和权限管理，满足安全性及合规性要求。

### 后台服务开发配置信息保护

痛点：应用开发配置文件需要进行加密以保护程序数据安全。 方案：通过 KMS 对敏感配置信息、数据库连接信息、数据库密码、登录密钥、后台服务的配置信息进行加密及完整性保护。

### 企业核心数据保护

痛点：核心知识产权、用户手机号、身份证号、银行账号、口令等隐私数据做严格保护，将敏感数据加密后保存，但是无法保证数据密钥的安全。 方案：以信封加密方式，将所有核心数据通过数据密钥加密，数据密钥再经过 KMS 加密，为核心数据提供双重保护。

### 网站或应用开发安全

痛点：提供 HTTPS 等服务时需要使用到证书、密钥，这些信息若以明文保存本地，攻击者可以轻易获取。 方案：通过 KMS 对密钥进行加解密，加密后本地保存密钥的密文文件，使用时解密且不保存本地，使得攻击者难以获取，从而保证网页和应用的安全性。

### 集中管理密码策略

痛点：应用统一的密钥管理策略至分散的业务系统。 方案：通过 SDK、云产品或 API 调用 KMS 服务，对云上及本地应用系统数据应用统一的密钥管理策略。

# 名词解释

最近更新时间: 2025-02-18 16:02:00

### 辅助校验数据

辅助校验数据（Encryption Context）是 JSON 格式的一段数据，如果在调用加密接口时传入这段数据，解密时必须提供等价的 JSON 数据，否则解密失败，您可以通过定时更新 Encryption Context 来提高业务安全性，也可以在不禁用主密钥的情况下，快速阻止非法访问。

### 数据加密密钥

数据加密密钥 （Data Encryption Keys，DEK）是信封加密流程中，通过数据加密密钥进行本地业务数据的加密。数据加密密钥DEK受用户主密钥CMK保护，可以自定义，也可以通过 KMS 的 API 来创建新的数据密钥。

### 信封加密

信封加密（Envelope Encryption）是一种应对海量数据的高性能加解密方案。对于较大的文件或者对性能敏感的数据加密，使用 GenerateDataKey 接口生成 AES 数据加密密钥 DEK，只需要传输数据加密密钥 DEK 到 KMS 服务端（通过CMK 进行加解密），所有的业务数据都是采用高效的本地对称加密处理，对业务的访问体验影响很小。

### 用户主密钥

用户主密钥（Customer Master Keys，CMK）受到经过第三方认证硬件安全模块（HSM）的保护，通过它来加解密业务使用到的密码、证书、数据密钥等敏感数据，可以通过控制台和 API 来创建和 管理主密钥。 用户主密钥（CMK）包括用户密钥和云产品密钥两种类型。

### 用户密钥

用户密钥是用户通过控制台和 API 来创建的用户主密钥。您可以对用户密钥进行创建/启用/禁用/轮换/权限控制等操作。

### 云产品密钥

云产品密钥是云产品/服务在调用密钥管理服务时，自动为用户创建的用户主密钥。您可以对云产品密钥进行查询及开启密钥轮换操作，不支持禁用、计划删除操作。

版权所有：腾讯云金融专区

# 快速入门
# 入门概述

最近更新时间: 2025-02-18 16:02:00

密钥管理系统KMS提供安全合规的密钥的全生命周期管理和数据加解密能力。

对于用户而言，KMS服务中涉及的核心密钥组件包括用户主密钥CMK（Customer Master Key，CMK）、数据加密密钥DEK（Data Encryption Key，DEK）。其中CMK属于用户的一级密钥，CMK 用于对敏感数据的加解密以及DEK的派生。DEK是信封加密流程中的二级密钥，用于加密业务数据的密钥，受用户主密钥CMK的保护。

关于使用CMK及DEK进行业务加解密的场景，请参见敏感数据加密和信封加密最佳实践。

## 密钥概述

**用户主密钥CMK**

用户主密钥是KMS中的核心资源，这些主密钥经过第三方认证硬件安全模块（HSM）的保护，作为用户加密解密的一级密钥。KMS服务主要是针对用户主密钥的管理服务。

用户主密钥CMK是主密钥的逻辑表示。CMK包含元数据，例如密钥ID、创建日期、描述和密钥状态等。通常情况下您可以使用 KMS 的自动生成用户主密钥功能来生成CMK，同时支持您自有密钥的导入来形成CMK。

用户主密钥CMK包括用户密钥和云产品密钥两种类型：

- **用户密钥**是用户通过控制台或API来创建的用户主密钥。您可以对用户密钥进行创建/启用/禁用/计划删除等操作。
- **云产品密钥**是腾讯云金融专区产品/服务在调用密钥管理系统时，自动为用户创建的CMK。CMK不支持禁用、计划删除操作。

**数据加密密钥DEK**

数据加密密钥是基于CMK 生成的二级密钥，可用于用户本地数据加密解密。 您可以使用KMS用户主密钥（CMK）生成DEK，但是，KMS不会存储、管理或跟踪您的DEK，也不会用于DEK执行加密操作。您必须在KMS之外使用和管理DEK。

一般DEK在信封加密流程中使用，通过DEK进行本地业务数据的加密。DEK受用户主密钥CMK保护，可以自定义创建，也可以通过接口创建。

# 创建密钥

最近更新时间: 2025-02-18 16:02:00

## 操作场景

本文为您详细介绍如何在密钥管理服务控制台创建密钥。

## 操作步骤

1. 登录密钥管理服务控制台。
2. 选择需要创建密钥的区域，单击【新建】。

3. 在弹出的配置框中，输入以下信息：

- 密钥名称：必填且在区域内唯一，密钥名称只能为字母、数字及字符 `_` 和 `-` ，且不能以 “KMS-” 开头。
- 描述信息：选填。

4. 单击【确定】后返回密钥列表，新创建的密钥会出现在密钥列表首位，也可以通过密钥名称来识别新创建的密钥。

# 管理密钥

最近更新时间: 2025-02-18 16:02:00

## 操作场景

本文为您详细介绍如何查看、启用/禁用，修改密钥以及密钥轮换等操作。

## 查看密钥

登录密钥管理服务控制台，注意主密钥是区分区域的，通过切换上方区域可以查看其他区域主密钥列表。

## 启用/禁用密钥

您可以通过以下方式启用/禁用密钥：单个操作和批量操作。

**单个操作**

密钥信息的右侧操作区域可以对该密钥进行启用、禁用操作。

**批量操作**

> 禁用密钥会导致所有依赖该密钥的加解密操作被同时禁用，所以在禁用密钥前，请确认没有运行中业务依赖该密钥。

找到您需要更改的密钥并勾选，单击列表上方的【启用密钥】或【禁用密钥】，页面将弹出"操作确认框"，继续单击【确认】，则可对所有选中密钥进行对应操作。

如果同时选择了不同可用状态的密钥，则会在单击批量【启用密钥】或【禁用密钥】后，页面将在弹出的"操作确认框"里进行相应提示，单击【确定】，系统只会对状态符合要求的密钥进行操作，不符合的密钥保持原有可用状态。

## 密钥轮换

在密钥列表操作栏【设置轮换策略】，您可以对该密钥进行密钥轮换操作。默认情况下，密钥轮换处于关闭状态。由用户设置是否打开。开启时可以设置自动轮换时间。

## 查看密钥详情

在密钥列表页面，单击任一密钥的【密钥ID/密钥名称】即可进入该密钥的详情页面，您可以查看和修改该密钥的信息，也可以通过该页面的在线工具进行加密或解密。

## 修改名称、用途

在密钥详情页，您可以修改密钥的名称、用途。 单击【密钥名称】或【密钥用途】，在弹出的对话框内输入需要新的内容。注意密钥名称只能为字母、数字及字符 _ 和 - ，且不能以"KMS-"开头。

# 加密密钥

最近更新时间: 2025-02-18 16:02:00

## 概述

KMS 提供加密解密接口，加密接口（Encrypt）用于加密最多为4KB的任意数据，可用于加密数据库密码，RSA Key，或其它较小的敏感信息。解密接口（Decrypt）用于对密文解密。生成的 DataKey 通过解密接口可以得到密钥的明文数据。

## 在线工具

适合处理单次或者非批量的加解密操作，比如首次生成密钥密文，开发者无需为非批量的加解密操作而去开发额外的工具，将精力集中在实现核心业务能力上。下面为您详细介绍如何使用在线加密工具对小型数据进行加密。

### 前提条件

已事先创建密钥，且保证密钥为启用状态。

### 操作步骤

1. 登录密钥管理服务控制台。
2. 找到您需要加解密的密钥，在“密钥ID/密钥名称”操作栏下，单击密钥名称，进入密钥详情页面。
3. 在“在线工具”模块下，选择【加密】或【解密】。
4. 在下方的输入框中输入待处理数据。

5. 单击【执行】，系统处理后的数据将显示在右边的灰色框中。

6. 您可以单击【下载】，将数据下载到本地电脑。

# 密钥归档

最近更新时间: 2025-02-18 16:02:00

## 操作场景

密钥归档是只能对密钥进行解密，而不能加密的存档过程。密钥管理系统（合规）KMS 为用户提供密钥归档的能力，从而实现对密钥更全面的管理。

您可以登录密钥管理系统（合规） 对已创建的用户主密钥进行启用/取消密钥归档设置。本文为您介绍如何通过控制台方式启用/取消密钥归档。

## 操作步骤

1. 登录密钥管理系统（合规） 控制台。
2. 找到您需要更改状态的密钥，在其密钥信息的右侧操作区域，可以对该密钥归档进行启用、取消归档操作。

- 密钥归档功能目前只支持**未被云产品占用的用户主密钥**，其他密钥均不支持。
- 只有当用户主密钥处于**已启用**、**禁用**状态下，才能使用密钥归档功能，其余状态均不支持。
- 当开启密钥归档功能，密钥处于**已归档**状态时：
- 该密钥只能进行**解密** ，而不能进行**加密** 。
- 该密钥可以进行**计划删除**、**取消归档** 及 **编辑标签** 操作，但不能进行**开启轮换**功能。
- 当用户主密钥处于**归档**的状态下，其密钥的存储和调用服务需要进行计费。

版权所有：腾讯云金融专区

# 最佳实践
# 敏感信息加密

最近更新时间: 2025-02-18 16:02:00

## 简介

敏感信息加密是密钥管理服务（KMS）核心的能力，实际应用中主要用来保护服务器硬盘上敏感数据的安全（小于4KB），如密钥、证书、配置文件等。 使用 CMK 加密敏感数据信息，而非直接将明文放置到云服务器上。使用时，再将密钥解密到内存，保证明文不落盘。这样，即使云服务器因为个人疏忽而遭受不明人员访问，这些数据信息也不会泄漏。

**示意图**

## 敏感信息举例

| | 密钥，证书 | 后台配置文件 |
|---|---|---|
| 用途 | 加密业务数据，通信通道，数字签名 | 保存系统架构和其他业务信息，比如数据库 IP、密码 |
| 丢失风险 | 保密信息被盗、加密通道遭监听、签名被伪造 | 业务数据被拖库、成为攻击其他系统的跳板 |

## 提前规划安全性

敏感信息是访问企业更高机密以及安全通道的钥匙，它本身的安全性尤为重要，所以在公司业务发展的阶段就应该规划其安全性。一个最基本的保护方法就是不要在云服务器硬盘上**明文放置敏感信息**，而是通过密钥管理服务将它们加密后放置，使用时再解密到内存，保证**明文不落盘。** 这样的好处是即使云服务器因为个人疏忽而遭受不明人员访问，也无法被直接获取明文敏感信息。对于攻击者来说，获取密文信息后还需要再推测密文文件用途、获取解密访问权限以及编写解密程序，这些将大大提高获取明文信息的难度和被发现的可能性。

# 信封加密

最近更新时间: 2025-02-18 16:02:00

## 简介

信封加密（Envelope Encryption）是一种应对海量数据的高性能加解密方案。对于较大的文件或者对性能敏感的数据加密，使用 GenerateDataKey 接口生成数据加密密钥 DEK，只需要传输数据加密密钥 DEK 到 KMS 服务端（通过CMK 进行加解密），所有的业务数据都是采用高效的本地对称加密处理，对业务的访问体验影响很小。 在实际业务场景中，对数据加密性能要求较高，数据加密量大的场景下，可通过生成 DEK 来对本地数据进行加解密，保证了业务加密性能的要求，同时也由 KMS 确保了数据密钥的随机性和安全性。

### 示意图

本场景中，KMS 生成的 CMK 作为重要资源，通过 CMK 生成和获取 DEK 的明文和密文。用户根据实际业务场景，首先在内存中通过 DEK 明文来对本地数据进行加密，然后将 DEK 密文和密文数据落盘，其次在业务解密场景中需通过 KMS 来解密 DEK 密文，最后通过解密出来的 DEK 明文在内存中解密。

## 操作步骤

### 创建明文 DEK

1. API 调用 KMS GenerateDataKey 接口生成数据密钥，
2. 用户通过第三方工具或者开发库创建（比如 OpenSSL）。

### 创建和保存密文 DEK

1. 密文 DEK 可以通过 KMS 云 API 对明文加密生成，也可以通过在线工具来处理。
2. 密文 DEK 由用户自行保存，常见的实现方案中，密文 DEK 会和密文业务数据保存在一起，比如存储场景下保存在一个或类似访问途径的存储容器，通信场景下与密文 DEK 和密文业务数据共同组成一个报文。

## 优势

### 高效

所有的业务数据都是采用高效的本地对称加密处理，对业务的访问体验影响很小。而对于 DEK 的创建和加解密开销，除了非常极端的情况下，您需要采用"一次一钥"的方案，大部分场景下可以在一段时间内复用一个 DEK 的明文和密文，所以大多数情况下这部分开销非常小。

### 安全易用

信封加密的安全性类似于常见公钥体系，DEK 保护业务数据，而KMS 则保护 DEK 并提供更好的可用性，您的主密钥无论如何都不会被泄露，只有有用密钥访问权限的对象才有能力操作 CMK。

## 何时在云上使用信封加密？

1. **较大体积**：目前 KMS API 支持4KB以下数据加解密。

2. **海量数据，低延迟**：想对业务数据加解密，但是又比较在乎访问延迟。KMS 后台虽然拥有非常高的性能，但是是远程调用且采用非对称加密，而信封加密方案大多数操作使用高性能的本地对称加密。

### 常见方案对比

| | 敏感信息加密 | 信封加密 |
|---|---|---|
| 相关密钥 | CMK | CMK、DEK |
| 性能 | 非对称加密，远程调用 | 少量远程非对称加密，海量本地对称加密 |
| 主要场景 | 密钥、证书、小型数据 | 海量大型数据 |

# 访问控制
# 概述

最近更新时间: 2025-02-18 16:02:00

如果您使用到了密钥管理系统（KMS）、私有网络（VPC）、云服务器、数据库等服务，这些服务由不同的人管理，但都共享您的云账号密钥，将存在以下问题：

- 您的密钥由多人共享，泄密风险高。
- 您无法限制其它人的访问权限，易产生误操作造成安全风险。

访问控制（CAM）用于管理账户下资源访问权限，通过 CAM，您可以通过身份管理和策略管理控制哪些子账号有哪些资源的操作权限。

例如，您的根账户下有个主密钥，您只想让子帐号 A 使用该主密钥，而让子账号 B 不能使用，就可以通过在 CAM 中配置策略，对子账号的权限进行控制。

如果您不需要对子账户进行 KMS 相关资源的访问控制，您可以跳过此章节。跳过这些部分并不影响您对文档中其余部分的理解和使用。

### CAM 基本概念

根账户通过给子账户绑定策略实现授权，策略设置可精确到（**API，资源，用户/用户组，允许/拒绝，条件**）维度。

- **账户**
  - **根账号**： 腾讯云金融专区 资源归属、资源使用计量计费的基本主体，可登录 腾讯云金融专区 服务。
  - **子账号**：由根账号创建账号，有确定的身份 ID 和身份凭证，且能登录到 腾讯云金融专区 控制台。根账号可以创建多个子账号(用户)。**子账号默认不拥有资源，必须由所属根账号进行授权。**
  - **身份凭证**：包括登录凭证和访问证书两种，**登录凭证**是指用户登录名和密码，**访问证书**是指云 API 密钥（SecretId 和 SecretKey）。
- **资源与权限**
  - **资源**：资源是云服务中被操作的对象，如一个 KMS 的一个主密钥，云服务器实例，COS 存储桶，VPC 实例等。
  - **权限**：权限是指允许或拒绝某些用户执行某些操作。默认情况下，**根账号拥有其名下所有资源的访问权限**，而**子账号没有根账号下任何资源的访问权限。**
  - **策略**：策略是定义和描述一条或多条权限的语法规范。**根账号**通过将**策略关联**到用户/用户组完成授权。

# KMS访问控制策略示例

最近更新时间: 2025-02-18 16:02:00

**KMS 的全读写策略**

以下策略允许子账号有所有操作的权限。Action 元素指定所有 KMS 相关 API。

```
{
"version": "2.0",
"statement": [
{
"action": [
"name/kms:*"
],
"resource": "*",
"effect": "allow"
}
]
}
```

**KMS 的只读策略**

以下策略允许子账号查询您的 KMS 资源。但子账号无法创建、更新或删除它们。 在控制台，操作一个资源的前提是可以查看该资源，所以建议您为用户开通 KMS 全读权限。

```
{
"version": "2.0",
"statement": [
{
"action": [
"name/kms:ListKey",
"name/kms:GetKeyAttributes"
],
"resource": "*",
"effect": "allow"
}
]
}
```

**允许子账号做管理类操作**

```
{
"version": "2.0",
"statement": [
{
"action": [
"name/kms:CreateKey",
"name/kms:ListKey",
"name/kms:GetKeyAttributes",
"name/kms:SetKeyAttributes"
],
"resource": "*",
"effect": "allow"
```

```
}
]
}
```

**允许子账号做数据类操作，但不允许其做管理类操作**

```
{
"version": "2.0",
"statement": [
{
"action": [
"name/kms:*"
],
"resource": "*",
"effect": "allow"
},
{
"action": [
"name/kms:CreateKey",
"name/kms:ListKey",
"name/kms:GetKeyAttributes",
"name/kms:SetKeyAttributes"
],
"resource": "*",
"effect": "deny"
}
]
}
```

# KMS API操作支持的资源级权限

最近更新时间: 2025-02-18 16:02:00

在 CAM 中，可对主密钥资源进行以下 API 操作的授权， 具体 API 支持的资源和条件的对应关系如下：

| API 操作 | 资源 | 备注 |
| --- | --- | --- |
| kms:CreateKey | qcs::kms:$region:$account:key/* | creatorUin 表示资源创建者的 uin，资源创建者可为根帐号或子账号。 |
| kms:ListKey | qcs::kms:$region:$account:key/* | |
| kms:ListKeys | qcs::kms:$region:$account:key/* | |
| kms:ListKeyDetail | qcs::kms:$region:$account:key/* | |
| kms:GetServiceStatus | qcs::kms:$region:$account:key/* | |
| kms:Encrypt | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权单个资源） qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） | |
| kms:Decrypt | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权单个资源） qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） | |
| kms::GenerateDataKey | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权单个资源） qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） | |
| kms::EnableKey | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权单个资源） qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） | |
| kms::DisableKey | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权单个资源） qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） | |
| kms::GetKeyAttributes | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权单个资源） qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） | |
| kms::SetKeyAttributes | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权单个资源） qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） | |
| kms:GetKeyRotationStatus | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权 | |

| | |
|---|---|
| | 单个资源）<br>qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） |
| kms:ReEncrypt | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权单个资源） qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） |
| kms:DescribeKey | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权单个资源）<br>qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） |
| kms:UpdateKeyDescription | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权单个资源） qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） |
| kms:UpdateAlias | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权单个资源）<br>qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） |
| kms:DisableKeyRotation | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权单个资源） qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） |
| kms:EnableKeyRotation | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权单个资源）<br>qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） |
| kms:EnableKeys | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权单个资源） qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） |
| kms:DisableKeys | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权单个资源）<br>qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） |
| kms:DescribeKeys | qcs::kms:$region:$account:key/creatorUin/$creatorUin/$keyid（授权单个资源） qcs::kms:$region:$account:key/creatorUin/$creatorUin/*（授权某个创建者的所有资源） qcs::kms:$region:$account:key/*（授权某个根帐号的所有资源） |

# 外部密钥导入

## 概述

最近更新时间: 2025-02-18 16:02:00

用户主密钥（CMK，Customer Master Key）是 KMS 服务的基本元素，它包含密钥 ID、密钥元数据（别名、描述、状态等）和用于加解密数据的密钥材料。

默认情况下，通过 KMS 服务创建 CMK 密钥时，由 KMS 服务底层加密机生成安全的密钥材料。当您希望使用自己的密钥材料时，也就是希望实施 BYOK（Bring Your Own Key）方案时，可通过 KMS 服务生成一个密钥材料为空的 CMK，并将自己的密钥材料导入到该用户主密钥中，形成一个外部密钥 CMK（EXTERNAL CMK），再由 KMS 服务进行该外部密钥的分发管理。

## 功能特点

- 在 腾讯云金融专区 上实施 BYOK（Bring Your Own Key）方案，即允许您在 腾讯云金融专区 架构上使用您自有的密钥材料进行敏感数据加解密服务。
- 完全掌控并管理您在 腾讯云金融专区 上使用的密钥服务，包括按需导入或删除密钥材料。
- 您可以在本地密钥管理基础设施中备份一份密钥材料，作为 腾讯云金融专区 密钥管理系统的额外灾备措施。
- 通过支持在云上使用您自有的密钥材料进行加解密操作，满足相关行业合规要求。

## 注意事项

- 需要确保导入密钥材料的安全性：
- 在使用密钥导入功能时，您需要确保自己生成密钥材料的随机源的安全可靠性。目前 KMS 国密版本仅允许导入128位对称密钥，FIPS 版本仅允许导入256位对称密钥。
- 需要确保导入密钥材料的可用性：
- KMS 服务提供自身服务的高可用及备份恢复能力，但导入的密钥材料的可用性需由用户来管理。强烈建议您采用安全可靠的方式保存密钥材料的原始备份，以便在意外删除密钥材料或密钥材料过期时，能及时将备份的密钥材料重新导入KMS服务。
- 需注意密钥导入操作的规范性：
- 当您将密钥材料导入CMK 时，该 CMK 与该密钥材料永久关联，即不能将其他密钥材料导入该外部密钥 CMK 中。当使用该外部密钥 CMK 加密数据时，加密后的数据必须使用加密时采用的 CMK（即 CMK 的元数据及密钥材料与导入的密钥匹配）才能解密数据，否则解密将失败。请谨慎处理密钥材料、CMK 的删除操作。
- 需要注意密钥导入的状态： 待导入状态的密钥属于启用状态的密钥，该启用状态密钥需付费使用。

版权所有：腾讯云金融专区

# 操作指南

最近更新时间: 2025-02-18 16:02:00

## 操作流程

创建外部密钥 CMK，您可以通过以下4个步骤完成操作。

1. 通过控制台或 API 创建一个密钥来源为“外部”的用户主密钥 CMK，即创建外部密钥 CMK。
2. 通过 API 操作获取密钥材料导入的参数，包括一个用于加密密钥材料的公钥，以及一个导入令牌。
3. 在本地通过加密机或其他安全的加密措施，利用步骤2获取的加密公钥对您的密钥材料进行加密。
4. 通过 API 操作，将加密后的密钥材料及步骤2获取的导入令牌，导入创建的外部密钥 CMK 中，至此导入外部密钥完成。

## 操作步骤

### 步骤1：创建外部密钥 CMK

创建外部密钥 CMK 有两种方式，控制台方式和调用 API 的方式。

- **控制台方式**（1）登录控制台。（2）选择需要创建密钥的区域，单击【新建】开始创建密钥。（3）在新建密钥窗口，输入密钥名称，选择密钥材料来源为外部，阅读导入外部密钥材料的方法及注意事项并勾选确认框。

  （4）单击【确定】，即可创建外部密钥 CMK。您可在控制台看到已创建的外部密钥 CMK，“密钥来源”显示为“外部”。

- 调用API方式

  您可以使用任何受支持的编程语言调用API。 请求 CreateKey API 时指定参数 Type为2，CreateKey 函数源码示例：

```
def create_external_key(client, alias):
"""
生成 BYOK 密钥，
:param Type = 2
"""
try:
req = models.CreateKeyRequest()
req.Alias = alias
req.Type = 2
rsp = client.CreateKey(req)
return rsp, None
except TencentCloudSDKException as err:
return None, err
```

### 步骤2：获取导入密钥材料参数

为确保密钥材料的安全性，需要先对您的密钥材料进行加密后再导入。您可以通过 API 获取导入密钥材料的参数，其中包括一个用于加密密钥材料的公钥，以及一个导入令牌。

GetParametersForImport 函数源码示例：

```
def get_parameters_for_import(client, keyid):
"""
获取导入主密钥（CMK）材料的参数，
返回的Token作为执行ImportKeyMaterial的参数之一，
返回的PublicKey用于对自主导入密钥材料进行加密。
返回的Token和PublicKey 24小时后失效，失效后如需重新导入，需要再次调用该接口获取新的 Token 和 PublicKey。
WrappingAlgorithm 指定加密密钥材料的算法，目前支持 RSAES_PKCS1_V1_5、RSAES_OAEP_SHA_1、RSAES_OAEP_SHA_256。
WrappingKeySpec 指定加密密钥材料的类型，目前只支持 RSA_2048。
"""
try:
req = models.GetParametersForImportRequest()
req.KeyId = keyid
req.WrappingAlgorithm = 'RSAES_PKCS1_V1_5' # RSAES_PKCS1_V1_5 | RSAES_OAEP_SHA_1 | RSAES_OAEP_SHA_256
req.WrappingKeySpec = 'RSA_2048' # RSA_2048
rsp = self.client.GetParametersForImport(req)
return rsp, None
except TencentCloudSDKException as err:
return None, err
```

### 步骤3：本地加密您的密钥材料

在本地利用获取的加密公钥对您的密钥材料进行加密。加密公钥是一个2048比特的 RSA 公钥，使用的加密算法需要与获取导入密钥材料参数时指定的一致。 由于 API 返回的加密公钥经过 base64 编码，因此在使用时需要先进行 base64 解码。目前 KMS 支持的加密算法有 RSAES_OAEP_SHA_1、RSAES_OAEP_SHA_256 与 RSAES_PKCS1_V1_5。

以下为通过 openssl 加密密钥材料的测试示例。在实际使用中，建议您通过加密机或其他更为安全的加密措施对密钥材料进行加密。

（1）调用 GetParametersForImport 接口获取Token 和 PublicKey。将 PublicKey 写入文件 public_key.base64。

（2）使用 openssl 生成随机数。

```
openssl rand -out raw_material.bin 16
```

您也可以通过 GenerateRandom 生成随机数进行 base64 解码。

国密版本 key material 长度必须为 128， FIPS 版本为 256。

（3）Decode Public Key 获取公钥原文。

```
openssl enc -d -base64 -A -in public_key.base64 -out public_key.bin
```

（4）使用公钥对 key material 进行加密。

```
# RSAES_OAEP_SHA_1 对应的命令行如下
openssl pkeyutl -in raw_material.bin -out encrypted_key_material.bin -inkey public_key.bin -keyform DER -pubin -encrypt -pkeyopt rsa_padding_mode:oaep -pkeyopt rsa_oaep_md:sha1

# RSAES_PKCS1_V1_5 对应的命令行如下
openssl pkeyutl -in raw_material.bin -out encrypted_key_material.bin -inkey public_key.bin -keyform DER -pubin -encrypt -pkeyopt rsa_padding_mode:pkcs1

# RSAES_OAEP_SHA_256 对应的命令行如下
openssl pkeyutl -in raw_material.bin -out encrypted_key_material.bin -inkey public_key.bin -keyform DER -pubin -encrypt -pkeyopt rsa_padding_mode:oaep -pkeyopt rsa_oaep_md:sha256
```

（5）将密文编码后作为参数可以导入 KMS。

```
openssl enc -e -base64 -A -in encrypted_key_material.bin -out encrypted_material.base64
```

encrypted_material.base64 为最终输出，作为 EncryptedKeyMaterial 导入 KMS。

### 步骤4：导入密钥材料

最后，将加密后的密钥材料及 步骤2 获取的导入令牌，通过 API 操作，一起导入至 步骤1 创建的外部密钥 CMK中。

- 导入令牌与加密密钥材料的公钥具有绑定关系，同时一个令牌只能为其生成时指定的主密钥导入密钥材料。导入令牌的有效期为24小时，在有效期内可以重复使用，失效以后需要获取新的导入令牌和加密公钥。
- 如果多次调用 GetParametersForImport 获取导入材料，只有最后一次调用的 token 和 publicKey 有效，历史调用返回的将自动过期。
- 可以为从未导入过密钥材料的外部密钥导入密钥材料，也可以重新导入已经过期和已被删除的密钥材料，或者重置密钥材料的过期时间。

ImportKeyMaterial 函数源码示例：

```
def import_key_material(client, material, token, keyid):
try:
req = models.ImportKeyMaterialRequest()
req.EncryptedKeyMaterial = material
req.ImportToken = token
req.KeyId = keyid
rsp = client.ImportKeyMaterial(req)
return rsp, None
except TencentCloudSDKException as err:
return None, err
```

至此，导入外部密钥操作完成。您可以像使用普通密钥一样使用外部密钥 CMK。

## 更多操作

### 删除外部密钥 CMK

外部密钥 CMK 的删除操作涉及到两类操作，一类操作为计划删除 CMK，一类操作为删除密钥材料，两类操作将会有不同的操作效果。

#### 计划删除 CMK

通过计划删除功能，删除外部密钥 CMK。计划删除的操作强制要求有7 - 30天的等待期，当达到到期时间后，外部密钥CMK 将被彻底删除。已经删除的 CMK 将无法恢复，通过其加密的数据也将无法解密，请谨慎操作。

#### 删除密钥材料

您可以通过两种方式删除密钥材料。当密钥材料过期或者被删除以后，外部密钥 CMK 将无法继续使用，由该 CMK 加密的密文也无法被解密，除非您重新导入相同的密钥材料。

- 通过 API 操作 DeleteImportedKeyMaterial 完成。当操作删除密钥材料后，密钥状态将变为等待导入（PendingImport）。
- 在导入密钥材料 API 操作中，将 ImportKeyMaterial 设置 ValidTo 输入参数设置过期时间完成，KMS 服务将自动删除到达过期时间的密钥材料。

等待密钥材料到期失效与手动删除密钥材料所达到的效果是一样的。

DeleteImportedKeyMaterial 函数源码示例：

```
def delete_key_material(client, keyid):
try:
req = models.DeleteImportedKeyMaterialRequest()
req.KeyId = keyid
rsp = client.DeleteImportedKeyMaterial(req)
return rsp, None
except TencentCloudSDKException as err:
return None, err
```

- 当您将密钥材料导入 CMK 时，该 CMK 与该密钥材料永久关联，即不能将其他密钥材料导入该外部密钥 CMK 中。当您删除密钥材料后，若需要重新导入密钥材料，导入的密钥材料必须与删除的密钥材料完全相同，才能导入成功。
- 当使用外部密钥 CMK 加密数据时，加密后的数据必须使用加密时采用的 CMK（即 CMK 的元数据及密钥材料与导入的密钥匹配）才能解密数据，否则解密会失败。请谨慎处理密钥材料、CMK 的删除操作。

# 国密 Encryption SDK
# SDK 概览

最近更新时间: 2025-02-18 16:02:00

密钥管理系统旗舰版提供通过商用密码产品认证的国密 Encryption SDK，国密 Encryption SDK 采用信封加密的方式，结合 KMS 多级密钥，灵活指定数据加密密钥的管理策略。SDK 支持一文一密、多文一密等多种加密方式，并能够支持 CMK 跨地域级的容灾。用户只需调用加解密接口和关注 CMK 的权限控制，即可轻松实现本地高性能海量数据加解密。

如下为使用国密 Encryption SDK 进行数据加解密的原理图：

## 功能特点

**极简加解密服务**

采用信封加密，复杂的密钥管理全由国密 Encryption SDK 进行封装，用户仅需调用加解密接口和关注 CMK 的权限控制即可实现本地海量数据加解密。

**多密钥容灾保护**

国密 Encryption SDK 支持 CMK 跨地域级的容灾，对数据加密可指定多个 CMK（建议指定不同地域的 CMK），任意 CMK 均可解密 DEK 从而对数据进行解密。

**数据密钥缓存机制**

国密 Encryption SDK 具备 DEK 缓存管理功能，使用缓存机制能够有效降低加密过程中导致的性能损耗。将 DEK 缓存在本地，在进行加密时，用户可以通过接口参数指定是否使用数据密钥缓存。

**安全合规**

国密 Encryption SDK 可以指定由 KMS 生成的加密密钥，KMS 底层使用国家密码局或 FIPS-140-2 认证的硬件安全模块 HSM 来保护密钥的安全，确保密钥的保密性、完整性和可用性。

**国密算法支持**

国密 Encryption SDK 具备商用密码产品认证证书，支持基于 SM4 的多种模式加解密，满足密码算法合规要求，算法详情请参见 **各语言接口文档** 。

# SDK 接入指南

最近更新时间: 2025-02-18 16:02:00

国密Encryption SDK目前支持 **Linux** 和 **Windows** 系统的 C 语言、 Go 语言、Python2和Python3语言。 其中Go语言SDK，底层使用 C 来实现，上层通过 cgo 封装后，提供接口供 Go 语言调用；而Python语言SDK，底层也是使用C来实现，上层通过ctypes封装后，提供接口供Python语言调用。 本文档以 C 语言作为代码示例，介绍如何接入使用国密 Encryption SDK，其他语言可以参考SDK包中具体的示例代码。

## 环境依赖

- Linux 系统支持情况，已经在下述平台验证：

| 系统版本 | 位数 | 支持情况 |
| --- | --- | --- |
| Tencent Linux release 2.4（Final） | 64 | 支持 |
| CentOS 8.2 | 64 | 支持 |
| CentOS 8.0 | 64 | 支持 |
| CentOS 7.8 | 64 | 支持 |
| CentOS 7.7 | 64 | 支持 |
| CentOS 7.6 | 64 | 支持 |
| CentOS 7.5 | 64 | 支持 |
| CentOS 7.4 | 64 | 支持 |
| CentOS 7.3 | 64 | 支持 |
| CentOS 7.2 | 64 | 支持 |
| CentOS 6.9 | 64 | 支持 |
| CentOS 6.8 | 64 | 支持 |
| Debian 9.0 | 64 | 支持 |
| Ubuntu Server 20.04.1 LTS | 64 | 支持 |
| Ubuntu Server 18.04.1 LTS | 64 | 支持 |
| Ubuntu Server 16.04.1 LTS | 64 | 支持 |
| Ubuntu Server 14.04.1 LTS | 64 | 支持 |
| CoreOS 1745.5.0 | 64 | 支持 |
| Debian 10.2 | 64 | 支持 |
| Debian 8.2 | 64 | 支持 |

| 系统版本 | 位数 | 支持情况 |
| --- | --- | --- |
| openSUSE Leap 15.1 | 64 | 支持 |
| openSUSE 42.3 | 64 | 支持 |
| SUSE Linux Enterprise Server 12 SP3 | 64 | 支持 |

> ！Linux开发环境仅支持 glibc 2.12 及其以上版本

- Windows 系统支持情况，已经在下述平台验证：

| 系统版本 | 位数 | 支持情况 |
| --- | --- | --- |
| Windows Server 2019 数据中心版 64位 中文版 | 64 | 支持 |
| Windows Server 2016 数据中心版 64位中文版 | 64 | 支持 |
| Windows Server 2012 R2 数据中心版 64位中文版 | 64 | 支持 |

- SDK 基于 OpenSSL1.1.1 改造。

# 接入指引

### 步骤1：开通 KMS 旗舰版

国密 Encryption SDK 仅适用于密钥管理系统旗舰版，请升级为 KMS 旗舰版。

> ！KMS版本的升级操作目前是在TCE运营端平台中进行。

### 步骤2：创建用户主密钥

登录 密钥管理系统控制台，创建用户主密钥，并保证其状态为已启用。

> ?为了容灾互备，建议设置至少2个可用区的用户主密钥 CMK，国密 Encryption SDK 最多支持设置5个用户主密钥 CMK（调用原生接口可忽略）。

### 步骤3：获取 SDK

SDK获取的方式：先咨询交付人员或者项目经理，然后由交付人员再向产品接口人获取。

### 步骤4：在代码中引用加密 SDK

1. 加密 SDK 依赖 curl，如果没有，请提前安装，各不同操作系统的安装命令如下：

- **Ubuntu**

```
sudo apt-get install libcurl4-openssl-dev
```

- **CentOS**

```
yum install libcurl-devel
```

2. 将下载的 tar 包解压到本地，进入 src 目录。
3. 编辑 setenv.sh，配置环境变量。setenv.sh 包含的操作指令如下：

```
export LD_LIBRARY_PATH=../lib:../lib/proto
export OPENSSL_ENGINES=../lib/engines-1.1
```

4. 环境变量配置完成后保存，执行source ./setenv.sh，使配置生效。
5. 修改 src 路径下的 demo_kms_pro.c 和 demo_original.c 文件。国密 Encryption SDK 支持基于 KMS 的密钥保护（ demo_kms_pro.c ）和原生加密（ demo_original.c ）的两种加密方式，两种模式的差异请参见 接口文档，用户根据需要修改其中一个即可，参数替换如下：
   - 使用主账号登录 API 密钥管理控制台 获取您的 secretId 和 secretKey，并替换为文件中对应的 "replace-with-real-secretId"、"replace-with-real-secretKey" 字符串。
   - 将 步骤2 创建的主密钥 ID 替换文件中的 "replace-with-realkeyid" 字符串。
6. 编译 src 路径下的 make 文件。
7. 运行可执行文件。

> !使用正确的 secretId、secretKey 和主密钥 ID ，Demo 才可以正常运行。

# C SDK KMS 示例

### 加解密函数

- **InitSdk**：初始化函数，用于检验用户是否已开通 KMS 旗舰版服务。
- **InitKeyManager**：用户主密钥初始化函数。
- **NewMasterKey**：设定主 CMK，在调用加解密函数时，会优先使用主 CMK，建议设置和 SDK 运行环境相同区域的 CMK 以减少延时。当主 CMK 不可用时，会使用 AddMasterKey 函数设定的备用CMK。
- **AddMasterKey**：设定备用的 CMK，备用 CMK 建议设定在不同区域，其备用 CMK 与 NewMasterKey 中设定的密钥形成 CMK 密钥列表，目的是为了容灾互备，以防首要主密钥无法使用时，可以使用密钥列表中的其他密钥。
- **Encrypt**：加密函数。
- **Decrypt**：解密函数。

以上函数详细的参数说明请参见 C SDK 接口文档。

> ?
>
> 1. 示例代码中，CBCEnAndDeTest 函数包含加密、解密的调用，其中采用的算法是 **SM4_CBC_128**。
> 2. 原生加密方法包含 SM2、SM3 及 SM4，详细的函数及参数说明请参见 步骤3 下载的 SDK 头文件 kms_enc_sdk.h。

### 示例代码

KMS 密钥保护方式接口调用示例如下：

```
#include<stdio.h>
#include "kms_enc_sdk.h"

int CBCEnAndDeTest(struct KeyManager *p,unsigned char plaintext[],char masterKeys[])
```

```
{
int i_ret = 0;
unsigned char ch_orig[128];
unsigned char ch_enheader[128];
unsigned char ch_cipher[1024];
size_t i_cipherlen = 0;

unsigned char ch_dedata[1024];
size_t i_dedatalen = 0;
size_t sourceLength = 0;

struct KeyManager keymanager;
struct MsgHead enheader,deheader;

char masterKeys[1024];
memset(masterKeys,0,sizeof(masterKeys));

memset(&enheader,0,sizeof(enheader));
memset(&deheader,0,sizeof(deheader));

memset(ch_orig,0,sizeof(ch_orig));
strcpy(ch_orig,plaintext);
sourceLength = strlen(ch_orig);

memset(ch_dedata,0,sizeof(ch_dedata));
memset(ch_cipher,0,sizeof(ch_cipher));
memset(ch_dedata,0,sizeof(ch_dedata));

unsigned char encryptionContext[1024];
memset(encryptionContext,0,sizeof(encryptionContext));
i_cipherlen = 0;
i_dedatalen = 0;

/*分片大小*/
size_t blockSize = 0;
/*选择加解密算法*/
enum Algorithm al_en = SM4_CBC_128;

strcpy(encryptionContext,"{\"name\":\"test\",\"date\":\"20200228\"}");

NewMasterKey(masterKeys,"ap-guangzhou","replace-with-realkeyid");
AddMasterKey(masterKeys,"ap-beijing","replace-with-realkeyid");

/*初始化 可加密的消息数量设置为0即缓存不过期*/
i_ret = InitKeyManager(&keymanager,masterKeys,0,0,0,p->secretId,p->secretKey);

if ( 0 != i_ret )
{
printf("InitKeyManager error\n");
return ( 0 );
}

/*ch_cipher 是加密后的密文内容 */
i_ret = Encrypt(ch_orig,sourceLength,&keymanager,masterKeys,al_en,encryptionContext,blockSize,&enheader,ch_cipher,&i
_cipherlen);
if (i_ret == 0)
{
```

```
i_ret = Decrypt(ch_cipher,i_cipherlen,&keymanager,&deheader,ch_dedata,&i_dedatalen);
if ( 0 == i_ret )
{
printf("ch_dedata[%s] i_dedatalen[%d]\n",ch_dedata,i_dedatalen);
}
else
{
printf("Decrypt is err!!!!!!!![%d]\n\n",i_ret);
}
}
else
{
printf("Encrypt is err!!!!!!!![%d]\n\n",i_ret);
}
return ( 0 );
}

int main()
{
int i_ret = 0;
unsigned char plaintext[128];
char region[128];
char masterKeys[1024];
char domainName[128];

memset(plaintext,0,sizeof(plaintext));
memset(region,0,sizeof(region));
memset(masterKeys,0,sizeof(masterKeys));
memset(domainName,0,sizeof(domainName));

struct KeyManager keymanager;

/*domainName 请填入TCE的域名地址*/
strcpy(domainName,"replace-with-real-domainName");
strcpy(region,"ap-guangzhou");
strcpy(keymanager.secretId,"replace-with-real-secretId");
strcpy(keymanager.secretKey,"replace-with-real-secretKey");
strcpy(plaintext,"abcdefg123456789abcdefg123456789abcdefg");

i_ret = InitSdk(region,keymanager.secretId,keymanager.secretKey,domainName);
if ( 0 != i_ret )
{
printf("InitSdk error\n");
return ( -1 );
}

NewMasterKey(masterKeys,"ap-guangzhou","replace-with-real-secretKey");
AddMasterKey(masterKeys,"ap-beijing","replace-with-real-secretKey");

CBCEnAndDeTest(&keymanager,plaintext,masterKeys);

return ( 0 );
}
```

# SDK 接口
# 旗舰版 C 接口文档

最近更新时间: 2025-02-18 16:02:00

国密 Encryption SDK 集成了 **KMS**，帮助用户解决密钥的生命周期管理问题，用户只需设置相关参数，调用加解密接口，便可实现本地、高效、稳定的国密加解密。

为了提供更好的服务，国密 Encryption SDK 支持基于 KMS 的密钥保护和原生加密的两种加密方式。

- 基于 KMS 的密钥保护方式：是指调用 KMS 平台生成加密密钥，由 KMS 提供密钥的全生命周期的管理，用户通过接口设置密钥的使用和替换策略，详情请参见 InitKeyManager 接口。
- 原生的加密方式：是指用户自行创建加密密钥，传入接口进行加解密，密钥的整个生命周期由用户管理，用户需要自己保证密钥的安全。

出于安全和合规考虑，建议用户使用基于 KMS 的密钥保护方式。

## 前提条件

- 国密 Encryption SDK 仅适用于密钥管理系统旗舰版，请升级为 KMS 旗舰版。
- 用户需要确保本机支持 CPU 支持 SDK 指令集优化，执行以下命令进行验证：

```
cat /proc/cpuinfo|grep aes
cat /proc/cpuinfo|grep avx
```

若可以查询到内容，则说明机器支持指令集加速。

## 初始化 SDK 接口

### InitSdk

- 功能描述：检验用户是否已开通 KMS 旗舰版服务。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | region | 是 | char * | CMK 地域信息字符串，详见产品支持的 地域列表 |
| | secretId | 是 | char * | 云账户 API 密钥 ID |
| | secretKey | 是 | char * | 云账户 API 密钥 Key |
| | domainName | 是 | char * | 域名信息字符串 |

- 返回值：初始化成功返回0，否则返回相应的 错误码。

> **说明：**

> - 需注意 SecretId 和 SecretKey 的保密存储：云服务商接口认证主要依靠 SecretID 和 SecretKey，SecretID 和 SecretKey 是用户的唯一认证凭证。业务系统需要该凭证调用云服务商接口。
> - 需注意 SecretId 和 SecretKey 的权限控制：建议使用子账号，根据业务需要进行接口授权的方式管控风险。
> - 需注意 domainName 的设置：如果domainName入参为""，则从环境变量TENCENT_SDK_DOMAIN中读取值，反之，则以入参为准。

# KMS 密钥保护方式接口说明

KMS 密钥保护方式基于 KMS 密钥管理平台实现，由 KMS 提供密钥的全生命周期管理，其中接口包括主密钥信息列表的新建添加、KeyManager 的初始化、加解密接口等。

### NewMasterKey

- 功能描述：把用户首个主密钥加入主密钥信息列表。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 出参 | masterKeys | 是 | char * | 主密钥信息列表，长度根据用户加入的密钥数量来确定，每个CMK 占用的空间为 region 和 KeyId 长度。 |
| 入参 | cmkRegion | 是 | char * | 主密钥 CMK 地域信息 |
| | cmkKeyId | 是 | char * | 主密钥 CMK 的 ID，从 KMS 控制台中查询 |

- 返回值：加入主密钥调用成功返回0，否则返回相应的 错误码。

> !用于加密的首个主密钥，在 KMS 平台中是处于**生效**的状态。

### AddMasterKey

- 功能描述：加入备用的用户主密钥，目的是为了灾备，当首个主密钥无法使用时，将会使用的备用密钥，最多支持加入4个。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 出参 | masterKeys | 是 | char * | 主密钥信息列表，长度根据用户加入的密钥数量来确定，每个 CMK 占用的空间为 region 和 KeyId 长度。 |
| 入参 | cmkRegion | 是 | char * | 主密钥 CMK 地域信息 |
| 入参 | cmkKeyId | 是 | char * | 主密钥 CMK 的 ID，从 KMS 控制台中查询。 |

- 返回值：加入主密钥调用成功返回0，否则返回相应的 错误码。

> !请保证 masterKeys 至少保留有512字节的空间，否则可能会产生内存错误。

### InitKeyManager

- 功能描述：初始化 KeyManager 的结构体，KeyManager 用来保存密钥管理相关参数，包含主密钥信息、密钥加密次数、密钥生效时间等。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 出参 | keyManager | 是 | struct of KeyManager * | KeyManager 结构体指针 |
| 入参 | masterKeys | 是 | char * | 主密钥 CMK 信息列表 |
| 入参 | msgCount | 是 | int | 每个缓存 DataKey 可加密的消息数量，加密的数量达到后，会重新向 KMS 后台请求，生成新的 DataKey，设置为 0 表示没有限制使用次数 |
| 入参 | enExpiretime | 是 | int | 加密使用的 DataKey 在缓存中的有效期，单位为秒。和消息数量一起生效，消息数量超过或者超时时间达到，都会触发 DataKey 的替换，0 表示不过期 |
| 入参 | deExpiretime | 是 | int | 解密使用的 DataKey 缓存的有效期，单位为秒，0 表示不过期 |
| 入参 | secretId | 是 | char * | 云账户 API 密钥 ID |
| 入参 | secretKey | 是 | char * | 云账户 API 密钥 Key |

- 返回值：初始化成功返回0，否则返回相应的 错误码。

### Encrypt

- 功能描述：使用 KMS 平台创建的 DataKey，进行本地数据加密。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | source | 是 | unsigned char * | 待加密的明文数据 |
| 入参 | sourceLength | 是 | size_t | 待加密数据长度，单位 byte |
| 入参 | keyManager | 是 | struct of KeyManager * | 已经初始化的 KeyManager 结构体指针 |
| 入参 | masterKeys | 是 | char * | 主密钥 CMK 信息列表 |
| 入参 | algorithm | 是 | enum | 算法枚举值，参照后面算法列表 |
| 入参 | encryptionContext | 是 | char * | 用于标识 DataKey 的辅助字段，key/value 对的 JSON 字符串格式，最大支持 1024 字节。例如{"name":"test","date":"20200228"} |

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | blockSize | 是 | size_t | 0 表示加密时不分块加密，非 0 表示分块加密以及分块大小，单位 byte |
| 出参 | header | 是 | struct of MsgHead * | 头部数据结构体，用于返回本次加密的一些基本信息，具体请查看后续描述 |
| 出参 | cipher | 是 | unsigned char * | 加密后的密文内容 |
| 出参 | cipherLength | 是 | size_t | 密文长度，单位 byte |

- 返回值：加密成功返回0，否则返回相应的 错误码。

> !加密后的数据，会加入DataKey相关信息，只能使用KMS密钥保护方式的接口进行解密。

### 支持的加密算法列表

| 枚举值 | 数值 | 说明 |
| --- | --- | --- |
| SM4_CBC_128_WITH_SIGNATURE | 1 | 使用 SM3 HAC 签名的 SM4 CBC 模式 |
| SM4_CBC_128 | 2 | 不使用签名的 SM4 CBC 模式加密 |
| SM4_GCM_128_WITH_SIGNATURE | 3 | 使用 SM3 HAC 签名的 SM4 GCM 模式 |
| SM4_GCM_128 | 4 | 不使用签名的 SM4 GCM 模式加密算法 |
| SM4_CTR_128_WITH_SIGNATURE | 5 | 使用 SM3 HAC 签名的 SM4 CTR 模式 |
| SM4_CTR_128 | 6 | 不使用签名的 SM4 CTR 模式 |
| SM4_ECB_128_WITH_SIGNATURE | 7 | 使用 SM3 HAC 签名的 SM4 ECB 模式 |
| SM4_ECB_128 | 8 | 不使用签名的 SM4 ECB 模式 |

### EncryptedDataKey 结构体说明

| 参数名称 | 类型 | 说明 |
| --- | --- | --- |
| cmkRegion | char * | 主密钥 CMK 地域信息 |
| cmkKeyId | char * | 主密钥 CMK 的 ID，从 KMS 控制台中查询 |
| dataKey | char * | 存储的 DataKey 对应的密文 |

### MsgHead结构体说明

| 参数名称 | 类型 | 说明 |
| --- | --- | --- |
| algorithm | enum | 算法枚举值，请参见 加密算法列表 |

| 参数名称 | 类型 | 说明 |
| --- | --- | --- |
| encryptionContext | char * | 用于标识 DataKey 的辅助字段，key/value 对的 JSON 字符串格式，最大支持1024字节。例如{"name":"test","date":"20200228"} |
| dataKeyNum | int | 使用的加密后 DataKey 数量和有效的主密钥 CMK 数量相关，由各个地域的主密钥加密产生 |
| dataKey | Array of EncryptedDataKey | DataKey 的信息列表，详情请参见 EncryptedDataKey 结构体说明 |
| blockType | enum | 密文加密分块的枚举值，用于标识该密文是否被分块，详情请参见 BlockType 结构体说明 |
| blockLength | int | 分块的长度 |

### BlockType 结构体说明

| 枚举值 | 数值 | 说明 |
| --- | --- | --- |
| WITHOUT_BLOCK | 1 | 密文加密未做分块 |
| WITH_BLOCK | 2 | 密文加密开设分块 |

### Decrypt

- 功能描述：方法用于解密密文，得到明文数据。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | source | 是 | unsigned char * | 加密后的数据 |
| 入参 | sourceLength | 是 | size_t | 加密后的数据长度，单位 byte |
| 入参 | keyManager | 是 | struct of KeyManager * | 已经初始化的 KeyManager 结构体指针 |
| 出参 | header | 是 | struct of MsgHead * | 头部数据结构体，用于返回本次解密的一些基本信息，具体请看关于结构体的描述 |
| 出参 | plainText | 是 | unsigned char * | 解密后的明文数据 |
| 出参 | plainTextLength | 是 | size_t | 明文长度，单位 byte |

- 返回值：解密成功则返回0，否则返回相应的 错误码。

### KMS 加密方式接口调用示例

KMS 密钥保护方式接口调用示例如下：

```
#include<stdio.h>
#include "kms_enc_sdk.h"

int CBCEnAndDeTest(struct KeyManager *p,unsigned char plaintext[],char masterKeys[])
```

```
{
int i_ret = 0;
unsigned char ch_orig[128];
unsigned char ch_enheader[128];
unsigned char ch_cipher[1024];
size_t i_cipherlen = 0;

unsigned char ch_dedata[1024];
size_t i_dedatalen = 0;
size_t sourceLength = 0;

struct KeyManager keymanager;
struct MsgHead enheader,deheader;

char masterKeys[1024];
memset(masterKeys,0,sizeof(masterKeys));

memset(&enheader,0,sizeof(enheader));
memset(&deheader,0,sizeof(deheader));

memset(ch_orig,0,sizeof(ch_orig));
strcpy(ch_orig,plaintext);
sourceLength = strlen(ch_orig);

memset(ch_dedata,0,sizeof(ch_dedata));
memset(ch_cipher,0,sizeof(ch_cipher));
memset(ch_dedata,0,sizeof(ch_dedata));

unsigned char encryptionContext[1024];
memset(encryptionContext,0,sizeof(encryptionContext));
i_cipherlen = 0;
i_dedatalen = 0;

/*分片大小*/
size_t blockSize = 0;
/*选择加解密算法*/
enum Algorithm al_en = SM4_CBC_128;

strcpy(encryptionContext,"{\"name\":\"test\",\"date\":\"20200228\"}");

NewMasterKey(masterKeys,"ap-guangzhou","replace-with-realkeyid");
AddMasterKey(masterKeys,"ap-beijing","replace-with-realkeyid");

/*初始化 可加密的消息数量设置为0即缓存不过期*/
i_ret = InitKeyManager(&keymanager,masterKeys,0,0,0,p->secretId,p->secretKey);

if ( 0 != i_ret )
{
printf("InitKeyManager error\n");
return ( 0 );
}

/*ch_cipher 是加密后的密文内容 */
i_ret = Encrypt(ch_orig,sourceLength,&keymanager,masterKeys,al_en,encryptionContext,blockSize,&enheader,ch_cipher,&i
_cipherlen);
if (i_ret == 0)
{
```

```
i_ret = Decrypt(ch_cipher,i_cipherlen,&keymanager,&deheader,ch_dedata,&i_dedatalen);
if ( 0 == i_ret )
{
printf("ch_dedata[%s] i_dedatalen[%d]\n",ch_dedata,i_dedatalen);
}
else
{
printf("Decrypt is err!!!!!!!![%d]\n\n",i_ret);
}
}
else
{
printf("Encrypt is err!!!!!!!![%d]\n\n",i_ret);
}
return ( 0 );
}

int main()
{
int i_ret = 0;
unsigned char plaintext[128];
char region[128];
char masterKeys[1024];
char domainName[128];

memset(plaintext,0,sizeof(plaintext));
memset(region,0,sizeof(region));
memset(masterKeys,0,sizeof(masterKeys));
memset(domainName,0,sizeof(domainName));

struct KeyManager keymanager;

/*domainName 请填入TCE的域名地址*/
strcpy(domainName,"replace-with-real-domainName");
strcpy(region,"ap-guangzhou");
strcpy(keymanager.secretId,"replace-with-real-secretId");
strcpy(keymanager.secretKey,"replace-with-real-secretKey");
strcpy(plaintext,"abcdefg123456789abcdefg123456789abcdefg");

i_ret = InitSdk(region,keymanager.secretId,keymanager.secretKey,domainName);
if ( 0 != i_ret )
{
printf("InitSdk error\n");
return ( -1 );
}

NewMasterKey(masterKeys,"ap-guangzhou","replace-with-realkeyid");
AddMasterKey(masterKeys,"ap-beijing","replace-with-realkeyid");

CBCEnAndDeTest(&keymanager,plaintext,masterKeys);

return ( 0 );
}
```

# 原生加密方式的接口说明

原生加密方式对应的服务也需要升级为 KMS 旗舰版，与 KMS 密钥保护方式相比，原生加密方式需要用户本身生成加密密钥进行加解密，由用户保证密钥的安全性。出于安全与合规的考虑，建议用户使用 KMS 密钥保护方式。

> ?其中 CTR 模式加密没有填充，其他的模式加密采用 PKCS#7 标准进行填充。

## Sm2GetKey

- 功能描述：使用SM2算法生成密钥对。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 出参 | pubKey | 是 | unsigned char * | 公钥内容，长度为 64 字节 |
| 出参 | priKey | 是 | unsigned char * | 私钥内容，长度为 32 字节 |

- 返回值：密钥对生成成功返回0，否则返回相应的错误码。

## Sm2Sign

- 功能描述：使用 SM2 算法进行签名。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | pubKey | 是 | unsigned char * | 未编码的公钥内容，数据长度固定为 64 字节。 |
| 入参 | priKey | 是 | unsigned char * | 未编码的私钥内容，数据长度固定为 32 字节。 |
| 入参 | msg | 是 | unsigned char * | 原文数据 |
| 入参 | msgLen | 是 | int | 原文数据的长度，单位 byte |
| 出参 | sig | 是 | unsigned char * | 生成的签名 |
| 出参 | sigLen | 是 | int * | 签名数据的长度，单位 byte |

- 返回值：数据签名成功返回0，否则返回相应的 错误码。

> !公钥和私钥的长度为固定长度，用户如果输入长度不一致的数据，可能导致内存访问异常。

## Sm2Verify

- 功能描述：使用 SM2 算法进行验签。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | pubKey | 是 | unsigned char * | 未编码的公钥内容，数据长度固定为 64 字节 |
| 入参 | sig | 是 | unsigned char * | 签名后的数据 |
| 入参 | sigLen | 是 | int | 签名后的数据长度，单位 byte |

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 出参 | msg | 是 | unsigned char * | 原文数据 |
| 出参 | msgLen | 是 | int * | 原文数据的长度，单位 byte |

- 返回值：验签成功返回0，否则返回相应的 错误码。

> !公钥长度为固定长度64字节，用户如果输入长度不一致的数据，可能导致内存访问异常。

### Sm2PemChangeToPubkey

- 功能描述：对pem格式的公钥内容进行转换。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | pemPubKeyInfo | 是 | unsigned char * | pem 格式的公钥信息 |
| 出参 | pubKey | 是 | unsigned char * | 转换后的公钥信息 |

- 返回值：转换成功返回0，否则返回相应的错误码。

### HashForSM3WithSM2

- 功能描述：使用 **Sm2GetKey** 接口生成的公钥，并基于SM3算法生成信息摘要。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | msg | 是 | unsigned char * | 原文数据 |
| 入参 | msgLen | 是 | int | 原文数据的长度 |
| 入参 | pubKey | 是 | unsigned char * | 公钥内容，数据长度固定为 64 字节 |
| 入参 | id | 是 | unsigned char * | id 值 |
| 入参 | idLen | 是 | int | id 值的长度 |
| 出参 | digest | 是 | unsigned char * | 生成的摘要 |
| 出参 | digestLen | 是 | int * | 摘要数据的长度 |

- 返回值：生成摘要成功返回0，否则返回相应的错误码。

> 注意：公钥的长度为固定长度，用户如果输入长度不一致的数据，可能导致内存访问异常。

### Sm2SignWithDigest

- 功能描述：使用本地生成的消息摘要生成签名
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | pubKey | 是 | unsigned char * | 公钥内容，数据长度固定为 64 字节 |
| 入参 | priKey | 是 | unsigned char * | 私钥内容，数据长度固定为 32 字节 |
| 入参 | digest | 是 | unsigned char * | 摘要数据 |
| 入参 | digestLen | 是 | int | 摘要数据的长度 |
| 出参 | sig | 是 | unsigned char * | 生成的签名值 |
| 出参 | sigLen | 是 | int * | 签名数据的长度 |

- 返回值：生成签名成功返回0，否则返回相应的错误码。

> 注意：公钥和私钥的长度为固定长度，用户如果输入长度不一致的数据，可能导致内存访问异常。

### Sm2VerifyWithDigest

- 功能描述：通过生成的摘要内容进行验签。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | pubKey | 是 | unsigned char * | 公钥内容，数据长度为 64 字节 |
| 入参 | sig | 是 | unsigned char * | 签名内容 |
| 入参 | sigLen | 是 | int | 签名数据的长度 |
| 入参 | digest | 是 | unsigned char * | 摘要数据 |
| 入参 | digestLen | 是 | int | 摘要数据的长度 |

- 返回值：验签成功返回0，否则返回相应的错误码。

### Sm2Encrypt

- 功能描述：使用 SM2 算法进行加密。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | pubKey | 是 | unsigned char * | 未编码的公钥内容，数据长度为 64 字节 |

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | source | 是 | unsigned char * | SM2 加密的源数据 |
| 入参 | sourceLength | 是 | int | 源数据长度，单位 byte |
| 出参 | cipherText | 是 | unsigned char * | 密文数据 |
| 出参 | cipherLength | 是 | int * | 密文数据长度，单位 byte |

- 返回值：加密成功返回0，否则返回相应的 错误码。

> !SM2 加密适用于小数据的场景，不建议加密超过256k的数据。

### Sm2Decrypt

- 功能描述：使用SM2算法进行解密。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | priKey | 是 | unsigned char * | 未编码的私钥内容，数据长度固定为 32 字节 |
| 入参 | source | 是 | unsigned char * | 加密后的数据 |
| 入参 | sourceLength | 是 | int | 加密后的数据长度，单位 byte |
| 出参 | plainText | 是 | unsigned char * | 明文数据 |
| 出参 | plainTextLength | 是 | int * | 明文数据长度，单位 byte |

- 返回值：解密成功返回0，否则返回相应的 错误码。

### Sm3Hmac

- 功能描述：使用 SM3 哈希运算 Hmac 计算。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | data | 是 | unsigned char * | 原文数据 |
| 入参 | dataLen | 是 | int | 原文数据的长度，单位 byte |
| 入参 | hmacKey | 是 | unsigned char * | 计算 Hmac 的密钥内容 |
| 入参 | keyLen | 是 | int | 计算 Hmac 的密钥长度，单位 byte |
| 出参 | hmac | 是 | unsigned char * | 生成的 Hmac 值 |
| 出参 | hmacLen | 是 | int * | Hmac 长度，单位 byte |

版权所有：腾讯云金融专区

- 返回值：接口调用成功返回0，否则返回相应的 错误码。

### Sm3Digest

- 功能描述：使用SM3生成摘要。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | data | 是 | unsigned char * | 原文数据 |
| 入参 | dataLen | 是 | int | 原文数据的长度 |
| 出参 | digest | 是 | unsigned char * | 生成的摘要 |
| 出参 | digestLen | 是 | int | 生成的摘要的长度 |

### Sm4CbcEncrypt/Sm4CtrEncrypt

- 功能描述：使用 SM4 加密算法 CBC、CTR 模式的加密。

- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | source | 是 | unsigned char * | 原文数据 |
| 入参 | sourceLength | 是 | int | 原文数据的长度，单位 byte |
| 入参 | key | 是 | int char * | 用户自定义的 SM4 密钥，长度固定为 128 位(16 字节) |
| 入参 | iv | 是 | unsigned char * | 初始化向量，固定为 128 位(16 字节) |
| 出参 | cipherText | 是 | unsigned char * | 密文数据 |
| 出参 | cipherLength | 是 | int * | 密文数据长度，单位 byte |

- 返回值：加密成功返回0，否则返回相应的 错误码。

### Sm4CbcDecrypt/Sm4CtrDecrypt

- 功能描述：用于 SM4 加密算法 CBC、CTR 模式下的解密。

- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | source | 是 | unsigned char * | 加密后的数据 |
| 入参 | sourceLength | 是 | int | 加密后的数据长度，单位 byte |
| 入参 | key | 是 | unsigned char * | 用户自定义的 SM4 密钥，长度固定为 128 位(16 字节) |
| 入参 | iv | 是 | unsigned char * | 初始化向量，固定为 128 位(16 字节) |

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 出参 | plainText | 是 | unsigned char * | 解密后的明文数据 |
| 出参 | plainTextLength | 是 | int * | 解密后的明文数据长度，单位 byte |

- 返回值：解密成功返回0，否则返回相应的 错误码。

### Sm4EcbEncrypt

- 功能描述：方法是用于 SM4 加密算法 ECB 模式下的加密。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | source | 是 | unsigned char * | 原文数据 |
| 入参 | sourceLength | 是 | int | 原文数据的长度，单位 byte |
| 入参 | key | 是 | unsigned char * | 用户自定义的 SM4 密钥，长度固定为 128 位(16 字节) |
| 出参 | cipherText | 是 | unsigned char * | 密文数据 |
| 出参 | cipherLength | 是 | int * | 密文数据长度，单位 byte |

- 返回值：加密成功返回0，否则返回相应的 错误码。

### Sm4EcbDecrypt

- 功能描述：使用 SM4 加密算法 ECB 模式下的解密。
- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | source | 是 | unsigned char * | 加密后的数据 |
| 入参 | sourceLength | 是 | int | 加密后的数据长度，单位 byte |
| 入参 | key | 是 | unsigned char * | 用户自定义的 SM4 密钥，长度固定为 128 位(16 字节) |
| 出参 | plainText | 是 | unsigned char * | 解密后的明文数据 |
| 出参 | plainTextLength | 是 | int * | 解密后的明文数据长度，单位 byte |

- 返回值：解密成功返回0，否则返回相应的 错误码。

### Sm4GcmEncrypt

- 功能描述：用于 SM4 加密算法 GCM 模式下的加密。

- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | source | 是 | unsigned char * | 原文数据 |
| 入参 | sourceLength | 是 | int | 原文数据的长度，单位 byte |
| 入参 | key | 是 | unsigned char * | 用户自定义的 SM4 密钥，长度固定为 128 位(16 字节) |
| 入参 | iv | 是 | unsigned char * | 初始化向量，固定为 128 位(16 字节) |
| 入参 | ivLen | 是 | int | 向量内容的长度，单位 byte |
| 入参 | add | 是 | unsigned char * | 附加校验信息 |
| 入参 | addLen | 是 | int | 附加校验信息长度，单位 byte |
| 入参 | tag | 是 | unsigned char * | tag 值，即校验码 |
| 入参 | tagLen | 是 | int | 校验码的长度，单位 byte |
| 出参 | cipherText | 是 | unsigned char * | 密文数据 |
| 出参 | cipherLength | 是 | int * | 密文数据长度，单位 byte |

- 返回值：加密成功返回0，否则返回相应的 错误码。

### Sm4GcmDecrypt

- 功能描述：使用 SM4 加密算法 GCM 模式下的解密。

- 参数说明：

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 入参 | source | 是 | unsigned char * | 加密后的数据 |
| 入参 | sourceLength | 是 | int | 加密后的数据长度，单位 byte |
| 入参 | key | 是 | unsigned char * | 用户自定义的 SM4 密钥，长度固定为 128 位(16 字节) |
| 入参 | iv | 是 | unsigned char * | 初始化向量，固定为 128 位(16 字节) |
| 入参 | ivLen | 是 | int | 初始化向量内容的长度，单位 byte |
| 入参 | add | 是 | unsigned char * | 附加校验信息 |
| 入参 | addLen | 是 | int | 附加校验信息的长度，单位 byte |
| 入参 | tag | 是 | unsigned char * | tag 值，即校验码 |
| 入参 | tagLen | 是 | int | 校验码的长度，单位 byte |
| 出参 | plainText | 是 | unsigned char * | 解密后的明文数据 |

版权所有：腾讯云金融专区

| 属性 | 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| 出参 | plainTextLength | 是 | int * | 解密后的明文数据长度，单位 byte |

- 返回值：解密成功返回0，否则返回相应的 错误码。

### 原生加密方式的接口调用示例

原生加密方式的接口调用示例代码如下：

```
#include<stdio.h>
#include "kms_enc_sdk.h"

int Sm4CbcTest()
{
int i_ret = 0;

unsigned char ch_orig[1024];
int i_orig = 0;
unsigned char ch_en[1024];
int i_en = 0;
unsigned char ch_de[1024];
int i_de = 0;

memset(ch_orig,0,sizeof(ch_orig));
memset(ch_en,0,sizeof(ch_en));
memset(ch_de,0,sizeof(ch_de));

unsigned char key[16] = {
0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef,
0xfe, 0xdc, 0xba, 0x98, 0x76, 0x54, 0x32, 0x10};

unsigned char iv[16] = {
0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef,
0xfe, 0xdc, 0xba, 0x98, 0x76, 0x54, 0x32, 0x10};

strcpy(ch_orig,"this Sm4CbcTest");
i_orig = strlen(ch_orig);

i_ret = Sm4CbcEncrypt(ch_orig,i_orig,ch_en,&i_en,key,iv);
if ( 0 != i_ret )
{
printf("Sm4CbcEncrypt error\n");
return ( -1 );
}

i_ret = Sm4CbcDecrypt(ch_en,i_en,ch_de,&i_de,key,iv);
if ( 0 != i_ret )
{
printf("Sm4CbcDecrypt error\n");
return ( -1 );
}
printf("Sm4CbcDecrypt data is [%s]\n",ch_de);

return ( 0 );
```

```
}

int main()
{
int i_ret = 0;

char region[128];
char secretId[128];
char secretKey[128];

memset(region,0,sizeof(region));
memset(secretId,0,sizeof(secretId));
memset(secretKey,0,sizeof(secretKey));
memset(domainName,0,sizeof(domainName));

/*domainName 请填入TCE的域名地址*/
strcpy(domainName,"replace-with-real-domainName");
strcpy(region,"ap-guangzhou");
strcpy(secretId,"replace-with-real-secretId");
strcpy(secretKey,"replace-with-real-secretKey");

i_ret = InitSdk(region,secretId,secretKey,domainName);
if ( i_ret != 0 )
{
printf("InitSdk error\n");
return ( -1 );
}

Sm4CbcTest();

return ( 0 );
}
```

## 错误码

| 返回值 | 枚举值 | 说明 |
| --- | --- | --- |
| 0 | ES_OK | 正常返回 |
| -1 | ES_ERROR | 一般错误 |
| -2 | ES_ENCRYPT_SOURCE_EMPTY | 加密的原文为空 |
| -3 | ES_NO_CMKEY | 未设置主密钥 |
| -4 | ES_ALGRITHM_ERR | 算法不支持 |
| -5 | ES_GENERATE_DATAKEY_ERR | 产生 DataKey 错误 |
| -6 | ES_ENCRYPT_DATA_ERR | 加密 DataKey 错误 |
| -7 | ES_MARSHAL_PROTOBUF_ERR | 序列化 ProtoBuf 出错 |
| -8 | ES_CIPHER_TEXT_TOO_SHORT | 密文数据太短 |

| 返回值 | 枚举值 | 说明 |
| --- | --- | --- |
| -9 | ES_GET_PROTO_ERR | 获取 ProtoBuf 报文出错 |
| -10 | ES_PARSE_PROTO_ERR | 解析 ProtoBuf 报文出错 |
| -11 | ES_DECRYPT_DATAKEY_ERR | 解密 DataKey 错误 |
| -12 | ES_SET_DATAKEY_ERR | 设置 DataKey 错误 |
| -13 | ES_DIGEST_INVALIDATE | 签名不合法，导致校验不通过 |
| -14 | ES_MEMORY_ERR | 内存错误 |
| -15 | ES_KMSSERVICE_ERR | KMS 服务未开通 |
| -16 | ES_USEREDITION_ERR | 未升级为 KMS 旗舰版 |

# 旗舰版 GO 接口文档

最近更新时间: 2025-02-18 16:02:00

Go 语言 SDK，底层使用 C 语言实现，上层通过 cgo 封装后，提供接口供 Go 语言调用。

## 接口返回错误码说明

大部分接口的返回值为 EncryptSDKError 类型结构体（Code：错误码，Message：错误消息）。详情如下：

| 错误码 | 错误消息 |
| --- | --- |
| InvalidParameter | 参数错误 |
| KmsAccessError | 访问 KMS 出错 |
| GenerateDataKeyError | 产生 DataKey 错误 |
| EncryptDataKeyError | 加密 DataKey 错误 |
| LocalEncryptError | 本地加密出错 |
| UnknownError | 未知错误 |
| CheckAlgorithmError | 加密算法出错 |
| InvalidMessage | 获取 ProtoBuf 报文出错 |
| DecryptDataKeyError | 解密 DataKey 出错 |
| SignCheckFail | 签名校验失败 |
| LocalDecryptError | 本地解密出错 |
| KmsServiceError | KMS 服务未开通 |
| UserEditionError | KMS 未升级为旗舰版 |

## 初始化SDK接口

### InitSdk

功能描述：检验用户是否已开通 KMS 旗舰版服务。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| region | 是 | string | CMK 地域信息字符串 |
| secretId | 是 | string | 云账户 API 密钥 ID 值 |
| secretKey | 是 | string | 云账号 API 密钥 Key 值 |

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| domainName | 是 | string | 域名信息字符串 |

返回值：接口返回 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示初始化成功。
- 当接口返回值为非nil，代表初始化失败， 详情请参见错误码。

> **注意：**
>
> - 需注意 SecretId 和 SecretKey 的保密存储：云服务商接口认证主要依靠 SecretID 和 SecretKey，SecretID 和 SecretKey 是用户的唯一认证凭证。业务系统需要该凭证调用云服务商接口。
> - 需注意 SecretID 和 SecretKey 的权限控制：建议使用子账号，根据业务需要进行接口授权的方式管控风险。
> - 需注意 domainName 的设置：如果domainName入参为""，则从环境变量TENCENT_SDK_DOMAIN中读取值，反之，则以入参为准。

# KMS加密方式的接口说明

## NewMasterKey

功能描述：将用户首个主密钥加入主密钥信息列表。

参数说明：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| masterKeys | 是 | []byte | 主密钥信息列表，长度根据用户加入的密钥数量来确定，每个 CMK 占用的空间为 Region 和 KeyId 长度。 |
| cmkRegion | 是 | string | 主密钥 CMK 地域信息 |
| cmkKeyId | 是 | string | 主密钥 CMK的 ID，从 KMS 控制台中查询 |

返回值：接口返回 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示添加成功。
- 当接口返回值为非 nil，代表添加失败， 详情请参参见错误码。

> **注意：**
>
> 请确保用于加密的首个主密钥，在 KMS 平台中是处于生效的状态。

## AddMasterKey

功能描述：加入备用的用户主密钥，目的是为了灾备，当首个主密钥无法使用时，会使用的备用密钥，最多支持加入4个。

参数说明：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| masterKeys | 是 | []byte | 主密钥信息列表，长度根据用户加入的密钥数量来确定，每个CMK占用的空间为Region和KeyId长度。 |
| cmkRegion | 是 | string | 主密钥（CMK）地域信息 |
| cmkKeyId | 是 | string | 主密钥（CMK）的ID，从KMS控制台中查询 |

返回值：接口返回 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示添加成功。
- 当接口返回值为非 nil，代表添加失败， 详情请参参见错误码。

### InitKeyManager

功能描述：初始化 KeyManager 的结构体，KeyManager 用来保存密钥管理相关参数，包含主密钥信息、密钥加密次数、密钥生效时间等。

参数说明：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| keyManager | 是 | * C.struct_KeyManager | KeyManager 结构体指针，使用 C 语言中的 KeyManager 结构体进行创建 |
| masterKeys | 是 | string | 主密钥 CMK 信息列表 |
| msgCount | 是 | int | 每个缓存 DataKey 可加密的消息数量，加密的数量达到后，会重新向 KMS 后台请求，生成新的 DataKey，设置为0表示没有限制使用次数。 |
| enExpiretime | 是 | int | 加密使用的DataKey在缓存中的有效期，单位为秒。和消息数量一起生效，消息数量超过或者超时时间达到，都会触发DataKey的替换，0表示不过期。 |
| deExpiretime | 是 | int | 解密使用的 DataKey 缓存的有效期，单位为秒，0表示不过期。 |
| secretId | 是 | string | 云账户 API 密钥 ID 值 |
| secretKey | 是 | string | 云账号 API 密钥 Key 值 |

返回值：接口返回 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示初始化成功。
- 当接口返回值为非 nil，代表初始化失败， 详情请参参见错误码。

### Encrypt

功能描述：使用 KMS 平台创建的 DataKey，进行本地数据加密。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | []byte | 待加密的明文数据 |

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| keyManager | 是 | * C.struct_KeyManager | 已经初始化的KeyManager结构体指针 |
| masterKeys | 是 | string | 主密钥（CMK）信息列表 |
| algorithm | 是 | C.enum_Algorithm | 算法枚举值，参照后面算法列表 |
| encryptionContext | 是 | string | 用于标识DataKey的辅助字段，key/value对的json字符串格式,最大支持1024字节。如：{"name":"test","date":"20200228"} |
| blockSize | 是 | int | 0 表示加密时不分块加密，非0表示分块加密以及分块大小，单位 byte |
| header | 是 | * C.struct_MsgHead | 头部数据结构体，用于返回本次加密的一些基本信息，具体请看关于结构体的描述 |

返回值：接口返回两个内容，一个字符数组和一个 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示加密成功。
- 当接口返回值为非 nil，代表加密失败， 详情请参见错误码。

> **注意：**
>
> 加密后的数据，会加入 DataKey 相关信息，只能使用 KMS 密钥保护方式的接口进行解密。

### Decrypt

功能描述：方法用于解密密文，得到明文数据。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | []byte | 加密后的数据 |
| keyManager | 是 | * C.struct_KeyManager | 已经初始化的 KeyManager 结构体指针 |
| header | 是 | * C.struct_MsgHead | 头部数据结构体，用于返回本次解密的一些基本信息，具体请看关于结构体的描述 |

返回值：接口返回两个内容，一个字符数组和一个 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示解密成功，解密后的明文内容在返回的字符数组中。
- 当接口返回值为非 nil，代表解密失败， 详情请参见错误码。

### C.enum_Algorithm 支持的加密算法列表

| 枚举值 | 数值 | 说明 |
| --- | --- | --- |
| C.SM4_CBC_128_WITH_SIGNATURE | 1 | 使用 SM3 HAC 签名的 SM4 CBC模式 |
| C.SM4_CBC_128 | 2 | 不使用签名的SM4 CBC模式加密 |

| 枚举值 | 数值 | 说明 |
| --- | --- | --- |
| C.SM4_GCM_128_WITH_SIGNATURE | 3 | 使用 SM3 HAC 签名的 SM4 GCM 模式 |
| C.SM4_GCM_128 | 4 | 不使用签名的 SM4 GCM 模式加密算法 |
| C.SM4_CTR_128_WITH_SIGNATURE | 5 | 使用 SM3HAC 签名的 SM4 CTR 模式 |
| C.SM4_CTR_128 | 6 | 不使用签名的 SM4 CTR 模式 |
| C.SM4_ECB_128_WITH_SIGNATURE | 7 | 使用 SM3 HAC 签名的 SM4 ECB 模式 |
| C.SM4_ECB_128 | 8 | 不使用签名的 SM4 ECB 模式 |

### C.EncryptedDataKey 结构体说明

| 参数名称 | 类型 | 说明 |
| --- | --- | --- |
| cmkRegion | C.char * | 主密钥 CMK 地域信息 |
| cmkKeyId | C.char * | 主密钥 CMK 的 ID，从 KMS 控制台中查询 |
| dataKey | C.char * | 存储的 Datakey 对应的密文 |

### C.struct_MsgHead 结构体说明

| 参数名称 | 类型 | 说明 |
| --- | --- | --- |
| algorithm | C.enum_Algorithm | 算法枚举值，详情请参见加密算法列表 |
| encryptionContext | C.char * | 用于标识 DataKey 的辅助字段，key/value 对的 JSON 字符串格式，最大支持1024字节。如：{"name":"test","date":"20200228"} |
| dataKeyNum | C.int | 使用的加密后 DataKey 数量，和有效的主密钥 CMK 数量相关，由各个地域的主密钥加密产生 |
| dataKey | Array of C.EncryptedDataKey | DataKey 的信息列表，详情请参见C.EncryptedDataKey 结构体说明 |
| blockType | C.enum_BlockType | 密文加密分块的枚举值，用于标识该密文是否被分块，详情请参见C.enum_BlockType 结构体说明 |
| blockLength | C.int | 分块的长度 |

### C.enum_BlockType 结构体说明

| 枚举值 | 数值 | 说明 |
| --- | --- | --- |
| C.WITHOUT_BLOCK | 1 | 密文加密未做分块 |
| C.WITH_BLOCK | 2 | 密文加密开设分块 |

### KMS 加密方式接口调用示例

KMS 密钥保护方式接口调用示例如下：

```
package main

/*
#include "kms_enc_sdk.h"
*/
import "C"

import (
"fmt"
// "time"
// "encoding/hex"
)

func ECBEnAndDeWithSignTest(){
masterKeys := make([]byte, 1024)
NewMasterKey(masterKeys,"ap-guangzhou","replace-with-realkeyid")
AddMasterKey(masterKeys,"ap-shanghai","replace-with-realkeyid")

f := &C.struct_KeyManager{}
header_en := &C.struct_MsgHead{}
header_de := &C.struct_MsgHead{}

error := InitKeyManager(f,string(masterKeys),0,0,0,"replace-with-real-secretId"," replace-with-real-secretKey ")
if ( nil != error ){
fmt.Println(error.Error())
return
}

source := []byte("hello world!")
encryptionContext := "{\"test\":\"nihao\"}"

var algorithm C.enum_Algorithm = C.SM4_CBC_128_WITH_SIGNATURE
cipher,err_en := Encrypt(source,f,string(masterKeys),algorithm,encryptionContext,0,header_en)
if err_en != nil {
fmt.Println(err_en.Error())
return
}
plainTexttest,err_de := Decrypt(cipher,f,header_de)
if err_de != nil {
fmt.Println(err_de.Error())
return
}
fmt.Println(string(plainTexttest))
}

func main() {
error := InitSdk("ap-guangzhou","replace-with-real-secretId","replace-with-real-secretKey ","replace-with-real-domainNa
me")
if(nil != error){
fmt.Println(error.Eoor())
return
}

ECBEnAndDeWithSignTest()

}
```

# 原生加密方式的接口说明

原生加密方式对应的服务也需要升级为旗舰版，与 KMS 密钥保护方式相比，原生加密方式需要用户自己生成加密密钥进行加解密，由用户保证密钥的安全性。出于安全与合规的考虑，建议用户使用 KMS 密钥保护方式。

> **注意：**
>
> 其中CTR模式加密没有填充，其他的模式加密采用 PKCS#7 标准进行填充。

## Sm2GetKey

功能描述：使用SM2算法生成密钥对。

输入参数：无需填充输入参数。

返回值：接口返回三个内容，两个字节数组（分别是生成的公钥值、私钥值）和一个EncryptSDKError类型结构体，具体请查看开头说明。

- 当接口返回的结构体信息为nil，表示获取密钥对成功；
- 非nil，代表获取失败，具体查看结构体中的错误码Code和错误信息Message。

## Sm2Sign

功能描述：使用 SM2 算法进行签名。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pubKey | 是 | []byte | 未编码的公钥内容，数据长度固定为64字节 |
| priKey | 是 | []byte | 未编码的私钥内容，数据长度固定为32字节 |
| msg | 是 | []byte | 原文数据 |

返回值：接口返回两个内容，一个字符数组和一个 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示签名成功，签名内容在返回的字符数组中。
- 当接口返回值为非 nil，代表签名失败， 详情请参见错误码。

> **注意：**
>
> 公钥和私钥的长度为固定长度，用户如果输入长度不一致的数据，可能导致内存访问异常。

## Sm2Verify

功能描述：使用 SM2 算法进行验签。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pubkey | 是 | []byte | 未编码的公钥内容，数据长度固定为64字节 |

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| msg | 是 | []byte | 原文数据 |
| sig | 是 | []byte | 签名后的数据 |

返回值：接口返回两个内容，一个字符数组和一个 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示验签成功，签名内容在返回的字符数组中。
- 当接口返回值为非 nil，代表验签失败， 详情请参见错误码。

> **注意：**
>
> 公钥长度为固定长度64字节，用户如果输入长度不一致的数据，可能导致内存访问异常。

### Sm2Encrypt

功能描述：使用 SM2 算法进行加密。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pubkey | 是 | []byte | 未编码的公钥内容，数据长度为64字节 |
| source | 是 | []byte | 源数据 |

返回值：接口返回两个内容，一个字符数组和一个 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示加密成功，加密后的密文内容在返回的字符数组中。
- 当接口返回值为非 nil，代表加密失败， 详情请参见错误码。

> **注意：**
>
> SM2 加密适用于小数据的场景，不建议加密超过256k的数据。

### Sm2Decrypt

功能描述：使用 SM2 算法进行解密

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| prikey | 是 | []byte | 未编码的私钥内容，数据长度固定为32字节 |
| source | 是 | []byte | 加密后的数据 |

返回值：接口返回两个内容，一个字符数组和一个 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示解密成功，解密后的明文内容在返回。
- 当接口返回值为非 nil，表示解密失败， 详情请参见错误码。

### Sm2PemChangeToPubkey

功能描述：对pem格式的公钥内容进行转换。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pemPubKeyInfo | 是 | \[]byte | pem格式的公钥信息 |

返回值：接口返回两个内容，一个字符数组和一个EncryptSDKError类型结构体，具体请查看开头说明。

- 当接口返回的结构体信息为nil，代表转换成功，转换后的公钥内容在返回的字节数组中；
- 非nil，代表转换失败，具体查看结构体中的错误码Code和错误信息Message。

### HashForSM3WithSM2

功能描述：使用 **Sm2GetKey** 接口生成的公钥，并基于SM3算法生成信息摘要。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| msg | 是 | \[]byte | 原文数据 |
| pubKey | 是 | \[]byte | 公钥内容，数据长度固定为64字节 |
| id | 是 | \[]byte | id值 |

返回值：接口返回两个内容，一个字节数组和一个EncryptSDKError类型结构体，具体请查看开头说明。

- 当接口返回的结构体信息为nil，代表摘要生成成功，生成的摘要内容在返回的字节数组中；
- 非nil，代表摘要生成失败，具体查看结构体中的错误码Code和错误信息Message。

### Sm2SignWithDigest

功能描述：使用本地生成的消息摘要生成签名

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pubKey | 是 | \[]byte | 公钥内容，数据长度固定为64字节 |
| priKey | 是 | \[]byte | 私钥内容，数据长度固定为32字节 |
| digest | 是 | \[]byte | **HashForSM3WithSM2** 生成的摘要信息内容 |

返回值：接口返回两个内容，一个字节数组和一个EncryptSDKError类型结构体，具体请查看开头说明。

- 当接口返回的结构体信息为nil，代表签名成功，生成的签名内容在返回的字节数组中；
- 非nil，代表摘要签名失败，具体查看结构体中的错误码Code和错误信息Message。

### Sm2VerifyWithDigest

功能描述：通过生成的摘要内容进行验签。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pubKey | 是 | \[]byte | 公钥内容，数据长度固定为64字节 |
| sig | 是 | \[]byte | 签名的内容 |
| digest | 是 | \[]byte | **HashForSM3WithSM2** 生成的摘要信息内容 |

返回值：接口返回一个EncryptSDKError类型结构体，具体请查看开头说明。

- 当接口返回的结构体信息为nil，代表验签成功；
- 非nil，代表摘要验签失败，具体查看结构体中的错误码Code和错误信息Message。

### Sm3Hmac

功能描述：使用 SM3 哈希运算 Hmac 计算。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| data | 是 | []byte | 原文数据 |
| hmacKey | 是 | []byte | 计算Hmac的密钥内容 |

返回值：接口返回两个内容，一个字符数组和一个 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示 Hmac 计算成功，Hmac 内容在返回的字符数组中。
- 当接口返回值为非 nil，表示Hmac计算失败， 详情请参见错误码。

### Sm3Digest

功能描述：使用SM3生成摘要。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| msg | 是 | \[]byte | 原文数据 |

返回值：接口返回两个内容，一个字节数组和一个EncryptSDKError类型结构体，具体请查看开头说明。

- 当接口返回的结构体信息为nil，代表生成摘要成功，生成的摘要内容在返回的字节数组中；
- 非nil，代表生成摘要失败，具体查看结构体中的错误码Code和错误信息Message。

### Sm4CbcEncrypt/Sm4CtrEncrypt

功能描述：方法是用于 SM4 加密算法 CBC、CTR 模式下的加密。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | []byte | 原文数据 |

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| Key | 是 | []byte | 用户自定义的SM4密钥，长度固定为128位(16字节)，不能设置为空 |
| iv | 是 | []byte | 初始化向量，固定为128位(16字节)，不能设置为空 |

返回值：接口返回两个内容，一个字符数组和一个 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示加密成功，加密后的密文内容在返回的字符数组中。
- 当接口返回值为非 nil，表示加密失败， 详情请参见错误码。

### Sm4CbcDecrypt/Sm4CtrDecrypt

功能描述：方法是用于 SM4 加密算法 CBC、CTR 模式下的解密。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | []byte | 加密后的数据 |
| key | 是 | []byte | 用户自定义的 SM4 密钥，长度固定为128位(16字节)，不能设置为空 |
| iv | 是 | []byte | 初始化向量，固定为128位(16字节)，不能设置为空 |

返回值：接口返回两个内容，一个字符数组和一个 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示解密成功，解密后的明文内容在返回的字符数组中。
- 当接口返回值为非 nil，表示解密失败， 详情请参见错误码。

### Sm4EcbEncrypt

功能描述：方法是用于 SM4 加密算法 ECB 模式下的加密。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | []byte | 原文数据 |
| key | 是 | []byte | 用户自定义的 SM4 密钥，长度固定为128位(16字节)，不能设置为空 |

返回值：接口返回两个内容，一个字符数组和一个 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示加密成功，加密后的密文内容在返回的字符数组中。
- 当接口返回值为非 nil，表示加密失败， 详情请参见错误码。

### Sm4EcbDecrypt

功能描述：方法是用于 SM4 加密算法 ECB 模式下的解密。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | []byte | 加密后的数据 |
| key | 是 | []byte | 用户自定义的SM4密钥，长度固定为128位(16字节)，不能设置为空 |

返回值：接口返回两个内容，一个字符数组和一个 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示解密成功，解密后的密文内容在返回的字符数组中。
- 当接口返回值为非 nil，表示解密失败， 详情请参见错误码。

### Sm4GcmEncrypt

功能描述：方法是用于 SM4 加密算法 GCM 模式下的加密。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | []byte | 加密后的数据 |
| key | 是 | []byte | 用户自定义的 SM4 密钥，长度固定为128位(16字节)，不能设置为空 |
| iv | 是 | []byte | 初始化向量，不能设置为空 |
| aad | 是 | []byte | 附加校验信息 |
| tag | 是 | []byte | tag 值，即校验码 |

返回值：接口返回两个内容，一个字符数组和一个 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示加密成功，加密后的密文内容在返回的字符数组中。
- 当接口返回值为非 nil，表示加密失败， 详情请参见错误码。

### Sm4GcmDecrypt

功能描述：方法是用于 SM4 加密算法 GCM 模式下的解密。

输入参数：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | []byte | 加密后的数据 |
| key | 是 | []byte | 用户自定义的 SM4 密钥，长度固定为128位(16字节)，不能设置为空 |
| iv | 是 | []byte | 初始化向量，不能设置为空 |
| aad | 是 | []byte | 附加校验信息 |
| tag | 是 | []byte | tag 值，即校验码 |

返回值：接口返回两个内容，一个字符数组和一个 EncryptSDKError 类型结构体。

- 当接口返回值为 nil，表示解密成功，解密后的明文内容在解密后的字符数组中。

- 当接口返回值为非 nil，表示解密失败， 详情请参见错误码。

### 原生加密方式的接口调用示例

原生加密方式的接口调用示例如下：

```
package main

import (
"fmt"
"encoding/hex"
)

func Sm4EcbTest(){
key := []byte("1234567890abcdef")
msg := []byte("hello world!")

cipherText,enerr := Sm4EcbEncrypt(msg,key)
if enerr != nil {
fmt.Println(enerr.Error())
}else{
plainText,deerr := Sm4EcbDecrypt(cipherText,key)
if deerr != nil {
fmt.Println(deerr.Error())
}else{
fmt.Print("Sm4EcbDecrypt:")
fmt.Println(string(plainText))
}
}
}

func main(){
error := InitSdk("ap-guangzhou","replace-with-real-secretId","replace-with-real-secret","replace-with-real-domainName")
if (nil != error){
fmt.Println("InitSdk err",error.Error())
return
}
fmt.Println("InitSdk is ok")

Sm4CbcTest()

}
```

# 旗舰版Python2接口文档

最近更新时间: 2025-02-18 16:02:00

Python2 语言 SDK，底层使用 C 语言实现，上层通过 ctypes 封装后，提供接口供 Python2 语言调用。

## 错误码

| 错误码 | 说明 |
| --- | --- |
| 0 | 正常返回 |
| -1 | 一般错误 |
| -2 | 加密的原文为空 |
| -3 | 未设置主密钥 |
| -4 | 算法不支持 |
| -5 | 产生 DataKey 错误 |
| -6 | 加密 DataKey 错误 |
| -7 | 序列化 ProtoBuf 出错 |
| -8 | 密文数据太短 |
| -9 | 获取 ProtoBuf 报文出错 |
| -10 | 解析 ProtoBuf 报文出错 |
| -11 | 解密 DataKey 错误 |
| -12 | 设置 DataKey 错误 |
| -13 | 签名不合法，导致校验不通过 |
| -14 | 内存错误 |
| -15 | KMS 服务未开通 |
| -16 | 未升级为 KMS 旗舰版 |

## 初始化 SDK 接口

### InitSdk

**功能描述**：检验用户是否开通 KMS 旗舰版服务。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| region | 是 | str | CMK 地域信息字符串，详见产品支持的地域列表 |
| secretId | 是 | str | 云账户 API 密钥 ID 值 |
| secretKey | 是 | str | 云账号 API 密钥 Key 值 |
| domainName | 是 | str | 域名信息字符串 |

**返回值**：接口返回一个整数。

- 当接口返回值为0，表示初始化成功。
- 当接口返回值为非0，代表初始化失败。

> **注意：**
>
> - 需注意 SecretId 和 SecretKey 的保密存储：云服务商接口认证主要依靠 SecretID 和 SecretKey，SecretID 和 SecretKey 是用户的唯一认证凭证。业务系统需要该凭证调用云服务商接口。
> - 需注意 SecretID 和 SecretKey 的权限控制：建议使用子账号，根据业务需要进行接口授权的方式管控风险。
> - 需注意 domainName 的设置：如果 domainName 入参为""，则从环境变量 TENCENT_SDK_DOMAIN 中读取值，反之，则以入参为准。

# KMS 加密方式的接口说明

## NewMasterKey

**功能描述**：将用户首个主密钥加入主密钥信息列表。

**参数说明**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| cmkRegion | 是 | str | 主密钥（CMK）地域信息 |
| cmkKeyId | 是 | str | 主密钥（CMK）的 ID，从 KMS 控制台中查询 |

**返回值**：接口返回整数、主密钥信息列表。

- 当接口返回整数值为0，表示添加成功。
- 当接口返回整数值为非0，代表添加失败。

> **注意：**
>
> 请确保用于加密的首个主密钥，在KMS平台中是处于**生效**的状态。

## AddMasterKey

**功能描述**：加入备用的用户主密钥，目的是为了灾备，当首个主密钥无法使用时，会使用的备用密钥，最多支持加入4个备用密钥。

**参数说明**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| masterKeys | 是 | list | 主密钥信息列表，长度根据用户加入的密钥数量来确定 |
| cmkRegion | 是 | str | 主密钥（CMK）地域信息 |
| cmkKeyId | 是 | str | 主密钥（CMK）的 ID，从 KMS 控制台中查询 |

**返回值**：接口返回整数、主密钥信息列表。

- 当接口返回值为0，表示添加成功。
- 当接口返回值为非0，代表添加失败。

### InitKeyManager

**功能描述**：初始化用户的主密钥，包含主密钥信息、密钥加密次数、密钥生效时间等，具体看后续参数。

**参数说明**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| masterKey | 是 | list | 主密钥（CMK）信息列表 |
| msgCount | 是 | int | 每个缓存 DataKey 可加密的消息数量，加密的数量达到后，会重新向 KMS 后台请求，生成新的 DataKey，设置为0表示没有限制使用次数 |
| enExpiretime | 是 | int | 加密使用的 DataKey 在缓存中的有效期，单位为秒，和消息数量一起生效，消息数量超过或者超时时间达到，都会触发 DataKey 的替换，0表示不过期 |
| deExpiretime | 是 | int | 解密使用的 DataKey 缓存的有效期，单位为秒，0表示不过期。 |
| secretId | 是 | str | 云账户 API 密钥 ID 值 |
| secretKey | 是 | str | 云账号 API 密钥 Key 值 |

**返回值**：接口返回 KeyManager 对象。

- 当接口返回值不为 None，表示初始化成功。
- 当接口返回值为 None，代表初始化失败。

### Encrypt

**功能描述**：使用 KMS 平台创建的 DataKey，进行本地数据加密。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | str | 待加密的明文数据 |
| keyManager | 是 | kms_enc_sdk.KeyManager | 已经初始化的 KeyManager 结构体指针 |
| masterKeys | 是 | list | 主密钥（CMK）信息列表 |
| algorithm | 是 | Algorithm | 算法枚举值，参照后面算法列表 |

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| encryptionContext | 是 | str | 用于标识 DataKey 的辅助字段，key/value 对的 json 字符串格式，最大支持2048字节。如：{"name":"test","date":"20200228"} |
| blockSize | 是 | int | 0 表示加密时不分块加密，非0表示分块加密以及分块大小，单位 byte |

**返回值**：接口返回三个内容，一个整数、一个 MsgHead 对象、一个字符串类型的加密后的密文。

- 当接口返回的整数信息为0，代表加密成功。
- 当接口返回的整数信息为非0，代表加密失败。

> **注意：**
>
> 加密后的数据，会加入 DataKey 相关信息，只能使用 KMS 密钥保护方式的接口进行解密。

### Decrypt

**功能描述**：方法用于解密密文，得到明文数据。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | str | 加密后的数据 |
| keyManager | 是 | kms_enc_sdk.KeyManager | 已经初始化的 KeyManager 对象 |

**返回值**：接口返回三个内容，一个整数、一个 MsgHead 结构体、一个字符串类型的解密后的明文。

- 当接口返回的整数信息为0，代表解密成功。
- 当接口返回的整数信息为非0，代表解密失败。

### Algorithm 支持的加密算法列表

| 枚举值 | 数值 | 说明 |
| --- | --- | --- |
| SM4_CBC_128_WITH_SIGNATURE | 1 | 使用 SM3HAC 签名的 SM4 CBC 模式 |
| SM4_CBC_128 | 2 | 不使用签名的 SM4 CBC 模式加密 |
| SM4_GCM_128_WITH_SIGNATURE | 3 | 使用 SM3HAC 签名的 SM4 GCM 模式 |
| SM4_GCM_128 | 4 | 不使用签名的 SM4 GCM 模式加密算法 |
| SM4_CTR_128_WITH_SIGNATURE | 5 | 使用 SM3HAC 签名的 SM4 CTR 模式 |
| SM4_CTR_128 | 6 | 不使用签名的 SM4 CTR 模式 |
| SM4_ECB_128_WITH_SIGNATURE | 7 | 使用 SM3HAC 签名的 SM4 ECB 模式 |
| SM4_ECB_128 | 8 | 不使用签名的 SM4 ECB 模式 |

### KMS 加密方式接口调用示例

```
#coding=utf-8
import sys
import os
import time

from kms_enc_sdk import *

region1 = 'replace-with-real-region1'
keyId1 = 'replace-with-realkeyid1'
region2 = 'replace-with-real-region2'
keyId2 = 'replace-with-realkeyid2'
secretId = 'replace-with-real-secretId'
secretKey = 'replace-with-real-secretKey'
domainName = 'replace-with-real-domainName'

def ECBEnAndDeWithSignTest():
ret,masterKey = NewMasterKey(region1,keyId1)
if ret != 0:
print('NewMasterKey error:',ret)
return
ret,masterKey = AddMasterKey(masterKey,region2,keyId2)
if ret != 0:
print('AddMasterKey error:',ret)
return

keymanager = InitKeyManager(masterKey,0,0,0,secretId,secretKey)
if keymanager == None:
print('KeyManager error')
return

ret,enheader,endata = Encrypt('hello',keymanager,masterKey,Algorithm.SM4_ECB_128_WITH_SIGNATURE,'{}',0)
if ret == 0:
print('Encrypt ok')
else:
print('Encrypt error:',ret)
return

ret,deheader,dedata = Decrypt(endata,keymanager)
if ret == 0:
print 'dedata:',
print dedata
else:
print('Decrypt error:',ret)



ret = InitSdk(region1,secretId,secretKey,domainName)
if ret != 0:
print('InitSdk error:',ret)
sys.exit(1)

ECBEnAndDeWithSignTest()
```

## 原生加密方式的接口说明

原生加密方式对应的服务也需要升级为 **旗舰版**，与 KMS 密钥保护方式相比，原生加密方式需要用户自己生成加密密钥进行加解密，由用户保证密钥的安全性。出于安全与合规的考虑，建议用户使用 KMS 密钥保护方式。

> **说明：**
>
> 其中 CTR 模式加密没有填充，其他的模式加密采用 PKCS#7 标准进行填充。

### Sm2GetKey

**功能描述**：使用 SM2 算法生成密钥对。

**输入参数**：无需填充输入参数。

**返回值**：接口返回三个内容，一个整数和两个字符串（分别是生成的公钥值及私钥值）。

- 当接口返回的整数信息为0，表示获取密钥对成功；
- 当接口返回的整数信息为非0，代表获取失败。

### Sm2Sign

**功能描述**：使用 SM2 算法进行签名。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pubKey | 是 | str | 公钥内容，数据长度固定为64字节 |
| priKey | 是 | str | 私钥内容，数据长度固定为32字节 |
| source | 是 | str | 原文数据 |

**返回值**：接口返回两个内容，一个整数和一个包含签名内容的字符串。

- 当接口返回的整数信息为0，代表签名成功。
- 当接口返回的整数信息为非0，代表签名失败。

> **注意：**
>
> 公钥和私钥的长度为固定长度，用户如果输入长度不一致的数据，可能导致内存访问异常。

### Sm2Verify

**功能描述**：使用 SM2 算法进行验签。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pubkey | 是 | str | 公钥内容，数据长度固定为64字节 |
| sign | 是 | str | 签名后的数据 |
| source | 是 | str | 原文数据 |

- **返回值**：接口返回一个整数。

- 当接口返回的整数信息为0，表示验签成功。
- 当接口返回的整数信息为非0，代表验签失败。

> !公钥长度为固定长度64字节，用户如果输入长度不一致的数据，可能导致内存访问异常。

### Sm2Encrypt

- **功能描述**：使用 SM2 算法进行加密。
- **输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pubkey | 是 | str | 公钥内容，数据长度为64字节 |
| source | 是 | str | 源数据 |

**返回值**：接口返回两个内容，一个整数和一个包含加密后的密文内容的字符串。

- 当接口返回的整数信息为0，代表加密成功。
- 当接口返回的整数信息为非0，代表加密失败。

> **注意**：
>
> SM2 加密适用于小数据的场景，不建议加密超过256k的数据。

### Sm2Decrypt

**功能描述**：使用SM2算法进行解密

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| prikey | 是 | str | 私钥内容，数据长度固定为32字节 |
| source | 是 | str | 加密后的数据 |

**返回值**：接口返回两个内容，一个整数和一个包含解密后的明文内容的字符串。

- 当接口返回的整数信息为0，代表解密成功。
- 当接口返回的整数信息为非0，代表解密失败。

### Sm2PemChangeToPubkey

**功能描述**：对 pem 格式的公钥内容进行转换。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pemPubKeyInfo | 是 | str | pem 格式的公钥信息 |

**返回值**：接口返回两个内容，一个整数和一个包含转换后的公钥内容的字符串。

- 当接口返回的整数信息为0，代表转换成功。
- 当接口返回的整数信息为非0，代表转换失败。

### HashForSM3WithSM2

**功能描述**：使用 **Sm2GetKey** 接口生成的公钥，并基于SM3算法生成信息摘要。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| msg | 是 | str | 原文数据 |
| pubKey | 是 | str | 公钥内容，数据长度固定为64字节 |
| id | 是 | str | id值 |

**返回值**：接口返回两个内容，一个整数和一个包含生成的摘要内容的字符串。

- 当接口返回的整数信息为0，代表摘要生成成功。
- 当接口返回的整数信息为非0，代表摘要生成失败。

### Sm2SignWithDigest

**功能描述**：使用本地生成的消息摘要生成签名

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pubKey | 是 | str | 公钥内容，数据长度固定为64字节 |
| priKey | 是 | str | 私钥内容，数据长度固定为32字节 |
| digest | 是 | str | **HashForSM3WithSM2** 生成的摘要信息内容 |

**返回值**：接口返回两个内容，一个整数和一个包含生成签名内容的字符串。

- 当接口返回的整数信息为0，代表签名成功。
- 当接口返回的整数信息为非0，代表摘要签名失败。

### Sm2VerifyWithDigest

**功能描述**：通过生成的摘要内容进行验签。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pubKey | 是 | str | 公钥内容，数据长度固定为64字节 |
| sig | 是 | str | **Sm2SignWithDigest** 生成的签名内容 |
| digest | 是 | str | **HashForSM3WithSM2** 生成的摘要信息内容 |

**返回值**：接口返回一个整数。

- 当接口返回的整数信息为0，代表验签成功。
- 当接口返回的整数信息为非0，代表摘要验签失败。

#### Sm3Hmac

**功能描述**：使用 SM3 哈希运算 Hmac 计算。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | str | 原文数据 |
| hmacKey | 是 | str | 计算 Hmac 的密钥内容 |

**返回值**：接口返回两个内容，一个整数和一个包含 Hmac 内容的字符串。

- 当接口返回的整数信息为0，代表 Hmac 计算成功。
- 当接口返回的整数信息为非0，代表 Hmac 计算失败。

### Sm3Digest

**功能描述**：使用 SM3 生成摘要。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | str | 原文数据 |

**返回值**：接口返回两个内容，一个整数和一个包含摘要内容的字符串。

- 当接口返回的整数信息为0，代表生成摘要成功。
- 当接口返回的整数信息为非0，代表生成摘要失败。

### Sm4CbcEncrypt/Sm4CtrEncrypt

**功能描述**：方法是用于 SM4 加密算法 CBC、CTR 模式下的加密。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | str | 原文数据 |
| Key | 是 | str | 用户自定义的 SM4 密钥，长度固定为128位(16字节)，不能设置为空 |
| iv | 是 | str | 初始化向量，固定为128位(16字节)，不能设置为空 |

**返回值**：接口返回两个内容，一个整数和一个包含加密后的密文内容的字符串。

- 当接口返回的整数信息为0，代表加密成功。
- 当接口返回的整数信息为非0，代表加密失败。

### Sm4CbcDecrypt/Sm4CtrDecrypt

**功能描述**：方法是用于 SM4 加密算法 CBC、CTR 模式下的解密。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | str | 加密后的数据 |
| Key | 是 | str | 用户自定义的 SM4 密钥，长度固定为128位(16字节)，不能设置为空 |
| iv | 是 | str | 初始化向量，固定为128位(16字节)，不能设置为空 |

**返回值**：接口返回两个内容，一个整数和一个包含解密后的明文内容的字符串。

- 当接口返回的整数信息为0，代表解密成功。
- 当接口返回的整数信息为非0，代表解密失败。

### Sm4EcbEncrypt

**功能描述**：方法是用于 SM4 加密算法 ECB 模式下的加密。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | str | 原文数据 |
| Key | 是 | str | 用户自定义的 SM4 密钥，长度固定为128位(16字节)，不能设置为空 |

**返回值**：接口返回两个内容，一个整数和一个包含加密后的密文内容的字符串。

- 当接口返回的整数信息为0，代表加密成功。
- 当接口返回的整数信息为非0，代表加密失败。

### Sm4EcbDecrypt

**功能描述**：方法是用于 SM4 加密算法 ECB 模式下的解密。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | str | 加密后的数据 |
| Key | 是 | str | 用户自定义的 SM4 密钥，长度固定为128位(16字节)，不能设置为空 |

**返回值**：接口返回两个内容，一个整数和一个包含解密后的明文内容的字符串。

- 当接口返回的整数信息为0，代表解密成功。
- 当接口返回的整数信息为非0，代表解密失败。

### Sm4GcmEncrypt

**功能描述**：方法是用于 SM4 加密算法 GCM 模式下的加密。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | str | 原文数据 |
| key | 是 | str | 用户自定义的 SM4 密钥，长度固定为128位(16字节)，不能设置为空 |
| iv | 是 | str | 初始化向量，不能设置为空 |
| aad | 是 | str | 附加校验信息 |

**返回值**：接口返回三个内容，一个整数、一个包含加密后的密文内容的字符串、一个包含加密后的 tag 内容的字符串。

- 当接口返回的整数信息为0，代表加密成功。
- 当接口返回的整数信息为非0，代表加密失败。

### Sm4GcmDecrypt

**功能描述**：方法是用于 SM4 加密算法 GCM 模式下的解密。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | str | 加密后的数据 |
| key | 是 | str | 用户自定义的 SM4 密钥，长度固定为128位(16字节)，不能设置为空 |
| iv | 是 | str | 初始化向量，不能设置为空 |
| aad | 是 | str | 附加校验信息 |
| tag | 是 | str | tag 值，即校验码 |

**返回值**：接口返回两个内容，一个整数和一个包含解密后的明文内容的字符串。

- 当接口返回的整数信息为0，代表解密成功。
- 当接口返回的整数信息为非0，代表解密失败。

### 原生加密方式的接口调用示例

```
#coding=utf-8
import sys
import os
import time

from kms_enc_sdk import *

key = "1234567890abcdef"
iv = "1234567890abcdef"
aad = "1234567890abcdef"

region = 'replace-with-real-region'
secretId = 'replace-with-real-secretId'
```

```
secretKey = 'replace-with-real-secretKey'
domainName = 'replace-with-real-domainName'

ret = InitSdk(region,secretId,secretKey,domainName)
if ret != 0:
print('InitSdk error:',ret)
sys.exit(1)

ret,pubKey,priKey = Sm2GetKey()
if ret == 0:
print ("Sm2GetKey ok")

ret,sm2endata = Sm2Encrypt(pubKey,b'this Sm2Test')
if ret == 0:
print ("Sm2Encrypt:",sm2endata)
ret,sm2dedata = Sm2Decrypt(priKey,sm2endata)
if ret == 0:
print ("Sm2Decrypt:",sm2dedata)
else:
print ("Sm2Decrypt is error",ret)
else:
print ("Sm2Encrypt is error",ret)


ret,sign = Sm2Sign(pubKey,priKey,b'hello world')
if ret == 0:
print ("Sm2Sign is ok",sign)
ret = Sm2Verify(pubKey,sign,b'hello world')
if ret == 0:
print ("Sm2Verify is ok")
else:
print ("Sm2Verify is error",ret)
else:
print ("Sm2Sign is error",ret)
else:
print ("Sm2GetKey is error",ret)


pemInfo = "-----BEGIN PUBLIC KEY-----\n"
pemInfo += "MFkwEwYHKoZIzj0CAQYIKoEcz1UBgi0DQgAE5rwEIw9e5fX87uSN7C/vy6lyEZ2R\n"
pemInfo += "gzLqWLnY8EPN1C+nJP2v4rLgaQCbHV38+vBVLimbLmdccLM69R83JxpxuQ==\n"
pemInfo += "-----END PUBLIC KEY-----\n"

ret,pubKey = Sm2PemChangeToPubkey(pemInfo)
if ret == 0:
print('Sm2PemChangeToPubkey ok')
else:
print('Sm2PemChangeToPubkey error')

priKey = "B8F66F0097488D8FA91FE857DFC9886356BA26A48C23F44271DD702F374174F0".decode('hex')

ret,digest = HashForSM3WithSM2('hello',pubKey,'1234567812345678')
if ret == 0:
print('HashForSM3WithSM2 ok')
else:
print('HashForSM3WithSM2 error')
```

```
ret,sign = Sm2SignWithDigest(pubKey,priKey,digest)
if ret == 0:
print('Sm2SignWithDigest ok')
else:
print('Sm2SignWithDigest error')

ret = Sm2VerifyWithDigest(pubKey,sign,digest)
if ret == 0:
print('Sm2VerifyWithDigest ok')
else:
print('Sm2VerifyWithDigest error')


ret,hmac = Sm3Hmac("hello",key)
if ret == 0:
print ("Sm3Hmac is ok:",hmac)
else:
print ("Sm3Hmac is error",ret)

ret,digest = Sm3Digest('hello')
if ret == 0:
print ('Sm3Digest is ok:',digest)
else:
print ('Sm3Digest is error',ret)

ret,en_data = Sm4CbcEncrypt("this Sm4CbcTest",key,iv)
if ret == 0:
ret,de_data = Sm4CbcDecrypt(en_data,key,iv)
if ret == 0:
print (de_data)
else:
print ("Sm4CbcDecrypt is error",ret)
else:
print ("Sm4CbcEncrypt is error",ret)



ret,en_data = Sm4CtrEncrypt("this Sm4CtrTest",key,iv)
if ret == 0:
ret,de_data = Sm4CtrDecrypt(en_data,key,iv)
if ret == 0:
print (de_data)
else:
print ("Sm4CtrDecrypt is error",ret)
else:
print ("Sm4CtrEncrypt is error",ret)



ret,en_data = Sm4EcbEncrypt("this Sm4EcbTest",key)
if ret == 0:
ret,de_data = Sm4EcbDecrypt(en_data,key)
if ret == 0:
print (de_data)
else:
print ("Sm4EcbDecrypt is error",ret)
else:
```

```
print ("Sm4EcbEncrypt is error",ret)

ret,en_data,tag = Sm4GcmEncrypt("this Sm4GcmTest",key,iv,aad)
if ret == 0:
ret,de_data = Sm4GcmDecrypt(en_data,key,iv,aad,tag)
if ret == 0:
print (de_data)
else:
print ("Sm4GcmDecrypt is error",ret)
else:
print ("Sm4GcmEncrypt is error",ret)
```

# 旗舰版Python3接口文档

最近更新时间: 2025-02-18 16:02:00

Python3 语言 SDK，底层使用 C 语言实现，上层通过 ctypes 封装后，提供接口供 Python3 语言调用。

## 错误码

| 错误码 | 说明 |
| --- | --- |
| 0 | 正常返回 |
| -1 | 一般错误 |
| -2 | 加密的原文为空 |
| -3 | 未设置主密钥 |
| -4 | 算法不支持 |
| -5 | 产生 DataKey 错误 |
| -6 | 加密 DataKey 错误 |
| -7 | 序列化 ProtoBuf 出错 |
| -8 | 密文数据太短 |
| -9 | 获取 ProtoBuf 报文出错 |
| -10 | 解析 ProtoBuf 报文出错 |
| -11 | 解密 DataKey 错误 |
| -12 | 设置 DataKey 错误 |
| -13 | 签名不合法，导致校验不通过 |
| -14 | 内存错误 |
| -15 | KMS 服务未开通 |
| -16 | 未升级为 KMS 旗舰版 |

## 初始化 SDK 接口

### InitSdk

**功能描述**：检验用户是否开通 KMS 旗舰版服务。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| region | 是 | str | CMK 地域信息字符串，详见产品支持的地域列表 |
| secretId | 是 | str | 云账户 API 密钥 ID 值 |
| secretKey | 是 | str | 云账号 API 密钥 Key 值 |
| domainName | 是 | str | 域名信息字符串 |

**返回值**：接口返回一个整数。

- 当接口返回值为0，表示初始化成功。
- 当接口返回值为非0，代表初始化失败。

> **注意**：
>
> - SecretId 和 SecretKey 需保密存储：云服务商接口认证主要依靠 SecretID 和 SecretKey，SecretID 和 SecretKey 是用户的唯一认证凭证。业务系统需要该凭证调用云服务商接口。
> - SecretID 和 SecretKey 的权限控制：建议使用子账号，根据业务需要进行接口授权的方式管控风险。
> - 在设置 domainName 时：如果 domainName 入参为""，则从环境变量 TENCENT_SDK_DOMAIN 中读取值，反之，则以入参为准。

# KMS加密方式的接口说明

## NewMasterKey

**功能描述**：将用户首个主密钥加入主密钥信息列表。

**参数说明**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| cmkRegion | 是 | str | 主密钥（CMK）地域信息 |
| cmkKeyId | 是 | str | 主密钥（CMK）的 ID，从 KMS 控制台中查询 |

**返回值**：接口返回整数、主密钥信息列表。

- 当接口返回整数值为0，表示添加成功。
- 当接口返回整数值为非0，代表添加失败。

> **注意**：
>
> 请确保用于加密的首个主密钥，在 KMS 平台中是处于生效的状态。

## AddMasterKey

**功能描述**：加入备用的用户主密钥，目的是为了灾备，当首个主密钥无法使用时，会使用的备用密钥，最多支持加入4个备用密钥。

**参数说明**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| masterKeys | 是 | list | 主密钥信息列表，长度根据用户加入的密钥数量来确定 |
| cmkRegion | 是 | str | 主密钥（CMK）地域信息 |
| cmkKeyId | 是 | str | 主密钥（CMK）的 ID，从 KMS 控制台中查询 |

**返回值**：接口返回整数、主密钥信息列表。

- 当接口返回值为0，表示添加成功。
- 当接口返回值为非0，代表添加失败。

### InitKeyManager

**功能描述**：初始化用户的主密钥，包含主密钥信息、密钥加密次数、密钥生效时间等，具体参见下方参数说明。

**参数说明**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| masterKey | 是 | list | 主密钥（CMK）信息列表 |
| msgCount | 是 | int | 每个缓存 DataKey 可加密的消息数量，加密的数量达到后，会重新向KMS后台请求，生成新的 DataKey，设置为0表示没有限制使用次数。 |
| enExpiretime | 是 | int | 加密使用的 DataKey 在缓存中的有效期，单位为秒，和消息数量一起生效，消息数量超过或者超时时间达到，都会触发 DataKey 的替换，0表示不过期 |
| deExpiretime | 是 | int | 解密使用的 DataKey 缓存的有效期，单位为秒，0表示不过期 |
| secretId | 是 | str | 云账户 API 密钥 ID 值 |
| secretKey | 是 | str | 云账号 API 密钥 Key 值 |

**返回值**：接口返回 KeyManager 对象。

- 当接口返回值不为 None，表示初始化成功。
- 当接口返回值为None，代表初始化失败。

### Encrypt

**功能描述**：使用 KMS 平台创建的 DataKey，进行本地数据加密。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | bytes | 待加密的明文数据 |
| keyManager | 是 | kms_enc_sdk.KeyManager | 已经初始化的 KeyManager 结构体指针 |
| masterKeys | 是 | list | 主密钥（CMK）信息列表 |
| algorithm | 是 | Algorithm | 算法枚举值，参照后面算法列表 |

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| encryptionContext | 是 | str | 用于标识 DataKey 的辅助字段，key/value 对的 json 字符串格式，最大支持2048字节。如：{"name":"test","date":"20200228"} |
| blockSize | 是 | int | 0 表示加密时不分块加密，非0表示分块加密以及分块大小，单位 byte |

**返回值**：接口返回三个内容，一个整数、一个 MsgHead 对象、一个字节类型的加密后的密文。

- 当接口返回的整数信息为0，代表加密成功。
- 当接口返回的整数信息为非0，代表加密失败。

> **注意：**
>
> 加密后的数据，会加入 DataKey 相关信息，只能使用 KMS 密钥保护方式的接口进行解密。

### Decrypt

**功能描述**：方法用于解密密文，得到明文数据。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | bytes | 加密后的数据 |
| keyManager | 是 | kms_enc_sdk.KeyManager | 已经初始化的 KeyManager 对象 |

**返回值**：接口返回三个内容，一个整数、一个 MsgHead 结构体、一个字节类型的解密后的明文。

- 当接口返回的整数信息为0，代表解密成功。
- 当接口返回的整数信息为非0，代表解密失败。

### Algorithm 支持的加密算法列表

| 枚举值 | 数值 | 说明 |
| --- | --- | --- |
| SM4_CBC_128_WITH_SIGNATURE | 1 | 使用 SM3HAC 签名的 SM4 CBC 模式 |
| SM4_CBC_128 | 2 | 不使用签名的 SM4 CBC 模式加密 |
| SM4_GCM_128_WITH_SIGNATURE | 3 | 使用 SM3HAC 签名的 SM4 GCM 模式 |
| SM4_GCM_128 | 4 | 不使用签名的 SM4 GCM 模式加密算法 |
| SM4_CTR_128_WITH_SIGNATURE | 5 | 使用 SM3HAC 签名的 SM4 CTR 模式 |
| SM4_CTR_128 | 6 | 不使用签名的 SM4 CTR 模式 |
| SM4_ECB_128_WITH_SIGNATURE | 7 | 使用 SM3HAC 签名的 SM4 ECB 模式 |
| SM4_ECB_128 | 8 | 不使用签名的 SM4 ECB 模式 |

### KMS加密方式接口调用示例

```
#coding=utf-8
import sys
import os
import time

from kms_enc_sdk import *

region1 = 'replace-with-real-region1'
keyId1 = 'replace-with-realkeyid1'
region2 = 'replace-with-real-region2'
keyId2 = 'replace-with-realkeyid2'
secretId = 'replace-with-real-secretId'
secretKey = 'replace-with-real-secretKey'
domainName = 'replace-with-real-domainName'

def ECBEnAndDeWithSignTest():
ret,masterKey = NewMasterKey(region1,keyId1)
if ret != 0:
print('NewMasterKey error:',ret)
return
ret,masterKey = AddMasterKey(masterKey,region2,keyId2)
if ret != 0:
print('AddMasterKey error:',ret)
return

keymanager = InitKeyManager(masterKey,0,0,0,secretId,secretKey)
if keymanager == None:
print('KeyManager error')
return

ret,enheader,endata = Encrypt(b'hello',keymanager,masterKey,Algorithm.SM4_ECB_128_WITH_SIGNATURE,'{}',0)
if ret == 0:
print('Encrypt ok')
else:
print('Encrypt error:',ret)
return

ret,deheader,dedata = Decrypt(endata,keymanager)
if ret == 0:
print ('dedata:',)
print (dedata)
else:
print('Decrypt error:',ret)

ret = InitSdk(region1,secretId,secretKey,domainName)
if ret != 0:
print('InitSdk error:',ret)
sys.exit(1)

ECBEnAndDeWithSignTest()
```

## 原生加密方式的接口说明

原生加密方式对应的服务也需要升级为 **旗舰版**，与 KMS 密钥保护方式相比，原生加密方式需要用户自己生成加密密钥进行加解密，由用户保证密钥的安全性。出于安全与合规的考虑，建议用户使用 KMS 密钥保护方式。

> **说明：**
>
> 其中 CTR 模式加密没有填充，其他的模式加密采用 PKCS#7 标准进行填充。

### Sm2GetKey

**功能描述**：使用 SM2 算法生成密钥对。

**输入参数**：无需填充输入参数。

**返回值**：接口返回三个内容，一个整数和两个字节列表（分别是生成的公钥值、私钥值）。

- 当接口返回的整数信息为0，表示获取密钥对成功。
- 当接口返回的整数信息为非0，代表获取失败。

### Sm2Sign

**功能描述**：使用 SM2 算法进行签名。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pubKey | 是 | bytes | 公钥内容，数据长度固定为64字节 |
| priKey | 是 | bytes | 私钥内容，数据长度固定为32字节 |
| source | 是 | bytes | 原文数据 |

**返回值**：接口返回两个内容，一个整数和一个包含签名内容的字节列表。

- 当接口返回的整数信息为0，代表签名成功。
- 当接口返回的整数信息为非0，代表签名失败。

> **注意：**
>
> 公钥和私钥的长度为固定长度，用户如果输入长度不一致的数据，可能导致内存访问异常。

### Sm2Verify

**功能描述**：使用 SM2 算法进行验签。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pubkey | 是 | bytes | 公钥内容，数据长度固定为64字节 |
| sign | 是 | bytes | 签名后的数据 |
| source | 是 | bytes | 原文数据 |

**返回值**：接口返回一个整数。

- 当接口返回的整数信息为0，表示验签成功。
- 当接口返回的整数信息为非0，代表验签失败。

> **注意：**
>
> 公钥长度为固定长度64字节，用户如果输入长度不一致的数据，可能导致内存访问异常。

### Sm2Encrypt

**功能描述**：使用 SM2 算法进行加密。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pubkey | 是 | bytes | 公钥内容，数据长度为64字节 |
| source | 是 | bytes | 源数据 |

**返回值**：接口返回两个内容，一个整数和一个包含加密后的密文内容的字节列表。

- 当接口返回的整数信息为0，代表加密成功。
- 当接口返回的整数信息为非0，代表加密失败。

> **注意：**
>
> SM2 加密适用于小数据的场景，不建议加密超过256k的数据。

### Sm2Decrypt

**功能描述**：使用 SM2 算法进行解密。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| prikey | 是 | bytes | 私钥内容，数据长度固定为32字节 |
| source | 是 | bytes | 加密后的数据 |

**返回值**：接口返回两个内容，一个整数和一个包含解密后的明文内容的字节列表。

- 当接口返回的整数信息为0，代表解密成功。
- 当接口返回的整数信息为非0，代表解密失败。

### Sm2PemChangeToPubkey

**功能描述**：对 pem 格式的公钥内容进行转换。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pemPubKeyInfo | 是 | bytes | pem 格式的公钥信息 |

**返回值**：接口返回两个内容，一个整数和一个包含转换后的公钥内容的字节列表。

- 当接口返回的整数信息为0，代表转换成功。
- 当接口返回的整数信息为非0，代表转换失败。

### HashForSM3WithSM2

**功能描述：**使用 **Sm2GetKey** 接口生成的公钥，并基于SM3算法生成信息摘要。

**输入参数：**

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| msg | 是 | bytes | 原文数据 |
| pubKey | 是 | bytes | 公钥内容，数据长度固定为64字节 |
| id | 是 | bytes | ID 值 |

**返回值：**接口返回两个内容，一个整数和一个包含生成的摘要内容的字节列表。

- 当接口返回的整数信息为0，代表摘要生成成功。
- 当接口返回的整数信息为非0，代表摘要生成失败。

### Sm2SignWithDigest

**功能描述**：使用本地生成的消息摘要生成签名。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pubKey | 是 | bytes | 公钥内容，数据长度固定为64字节 |
| priKey | 是 | bytes | 私钥内容，数据长度固定为32字节 |
| digest | 是 | bytes | **HashForSM3WithSM2** 生成的摘要信息内容 |

**返回值**：接口返回两个内容，一个整数和一个包含生成的签名内容的字节列表。

- 当接口返回的整数信息为0，代表签名成功。
- 当接口返回的整数信息为非0，代表摘要签名失败。

### Sm2VerifyWithDigest

**功能描述**：通过生成的摘要内容进行验签。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| pubKey | 是 | bytes | 公钥内容，数据长度固定为64字节 |
| sig | 是 | bytes | **Sm2SignWithDigest** 生成的签名内容 |
| digest | 是 | bytes | **HashForSM3WithSM2** 生成的摘要信息内容 |

**返回值**：接口返回一个整数。

- 当接口返回的整数信息为0，代表验签成功。
- 当接口返回的整数信息为非0，代表摘要验签失败。

### Sm3Hmac

**功能描述**：使用 SM3 哈希运算 Hmac 计算。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | bytes | 原文数据 |
| hmacKey | 是 | bytes | 计算 Hmac 的密钥内容 |

**返回值**：接口返回两个内容，一个整数和一个包含 Hmac 内容的字节列表。

- 当接口返回的整数信息为0，代表 Hmac 计算成功。
- 当接口返回的整数信息为非0，代表 Hmac 计算失败。

### Sm3Digest

**功能描述**：使用 SM3 生成摘要。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | bytes | 原文数据 |

**返回值**：接口返回两个内容，一个整数和一个包含摘要内容的字节列表。

- 当接口返回的整数信息为0，代表生成摘要成功。
- 当接口返回的整数信息为非0，代表生成摘要失败。

### Sm4CbcEncrypt/Sm4CtrEncrypt

**功能描述：**方法适用于 SM4 加密算法 CBC、CTR 模式下的加密。

**输入参数：**

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | bytes | 原文数据 |
| Key | 是 | bytes | 用户自定义的 SM4 密钥，长度固定为128位(16字节)，不能设置为空 |
| iv | 是 | bytes | 初始化向量，固定为128位(16字节)，不能设置为空 |

**返回值：**接口返回两个内容，一个整数和一个包含加密后的密文内容的字节列表。

- 当接口返回的整数信息为0，代表加密成功。
- 当接口返回的整数信息为非0，代表加密失败。

### Sm4CbcDecrypt/Sm4CtrDecrypt

**功能描述**：方法是用于 SM4 加密算法 CBC、CTR 模式下的解密。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | bytes | 加密后的数据 |
| Key | 是 | bytes | 用户自定义的 SM4 密钥，长度固定为128位(16字节)，不能设置为空 |
| iv | 是 | bytes | 初始化向量，固定为128位(16字节)，不能设置为空 |

**返回值**：接口返回两个内容，一个整数和一个包含解密后的明文内容的字节列表。

- 当接口返回的整数信息为0，代表解密成功。
- 当接口返回的整数信息为非0，代表解密失败。

### Sm4EcbEncrypt

**功能描述**：方法是用于 SM4 加密算法 ECB 模式下的加密。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | bytes | 原文数据 |
| Key | 是 | bytes | 用户自定义的 SM4 密钥，长度固定为128位(16字节)，不能设置为空 |

**返回值**：接口返回两个内容，一个整数和一个包含加密后的密文内容的字节列表。

- 当接口返回的整数信息为0，代表加密成功。
- 当接口返回的整数信息为非0，代表加密失败。

### Sm4EcbDecrypt

**功能描述**：方法是用于 SM4 加密算法 ECB 模式下的解密。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | bytes | 加密后的数据 |
| Key | 是 | bytes | 用户自定义的 SM4 密钥，长度固定为128位(16字节)，不能设置为空 |

**返回值**：接口返回两个内容，一个整数和一个包含解密后的明文内容的字节列表。

- 当接口返回的整数信息为0，代表解密成功。
- 当接口返回的整数信息为非0，代表解密失败。

### Sm4GcmEncrypt

**功能描述**：方法是用于 SM4 加密算法 GCM 模式下的加密。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | bytes | 原文数据 |
| key | 是 | bytes | 用户自定义的 SM4 密钥，长度固定为128位(16字节)，不能设置为空 |
| iv | 是 | bytes | 初始化向量，不能设置为空 |
| aad | 是 | bytes | 附加校验信息 |

**返回值**：接口返回三个内容，一个整数、一个包含加密后的密文内容的字节列表、一个包含加密后的 tag 内容的字节列表。

- 当接口返回的整数信息为0，代表加密成功。
- 当接口返回的整数信息为非0，代表加密失败。

### Sm4GcmDecrypt

**功能描述**：方法是用于 SM4 加密算法 GCM 模式下的解密。

**输入参数**：

| 参数名称 | 必选 | 类型 | 描述 |
| --- | --- | --- | --- |
| source | 是 | bytes | 加密后的数据 |
| key | 是 | bytes | 用户自定义的 SM4 密钥，长度固定为128位(16字节)，不能设置为空 |
| iv | 是 | bytes | 初始化向量，不能设置为空 |
| aad | 是 | bytes | 附加校验信息 |
| tag | 是 | bytes | tag 值，即校验码 |

**返回值**：接口返回两个内容，一个整数和一个包含解密后的明文内容的字节列表。

- 当接口返回的整数信息为0，代表解密成功。
- 当接口返回的整数信息为非0，代表解密失败。

### 原生加密方式的接口调用示例

```
#coding=utf-8
import sys
import os
import time

from kms_enc_sdk import *

key = b'1234567890abcdef'
iv = b'1234567890abcdef'
aad = b'1234567890abcdef'

region = 'replace-with-real-region'
secretId = 'replace-with-real-secretId'
```

```
secretKey = 'replace-with-real-secretKey'
domainName = 'replace-with-real-domainName'

ret = InitSdk(region,secretId,secretKey,domainName)
if ret != 0:
print('InitSdk error:',ret)
sys.exit(1)

ret,pubKey,priKey = Sm2GetKey()
if ret == 0:
print ('Sm2GetKey ok')

ret,sm2endata = Sm2Encrypt(pubKey,b'this Sm2Test')
if ret == 0:
print ('Sm2Encrypt:',sm2endata)
ret,sm2dedata = Sm2Decrypt(priKey,sm2endata)
if ret == 0:
print ('Sm2Decrypt:',sm2dedata)
else:
print ('Sm2Decrypt is error',ret)
else:
print ('Sm2Encrypt is error',ret)


ret,sign = Sm2Sign(pubKey,priKey,b'hello world')
if ret == 0:
print ('Sm2Sign is ok',sign)
ret = Sm2Verify(pubKey,sign,b'hello world')
if ret == 0:
print ('Sm2Verify is ok')
else:
print ('Sm2Verify is error',ret)
else:
print ('Sm2Sign is error',ret)
else:
print ('Sm2GetKey is error',ret)



pemInfo = "-----BEGIN PUBLIC KEY-----\n"
pemInfo += "MFkwEwYHKoZIzj0CAQYIKoEcz1UBgi0DQgAE5rwEIw9e5fX87uSN7C/vy6lyEZ2R\n"
pemInfo += "gzLqWLnY8EPN1C+nJP2v4rLgaQCbHV38+vBVLimbLmdccLM69R83JxpxuQ==\n"
pemInfo += "-----END PUBLIC KEY-----\n"

ret,pubKey = Sm2PemChangeToPubkey(pemInfo.encode())
if ret == 0:
print('Sm2PemChangeToPubkey ok')
else:
print('Sm2PemChangeToPubkey error')

priKey = base64.b16decode("B8F66F0097488D8FA91FE857DFC9886356BA26A48C23F44271DD702F374174F0")

ret,digest = HashForSM3WithSM2(b'hello',pubKey,b'1234567812345678')
if ret == 0:
print('HashForSM3WithSM2 ok')
else:
print('HashForSM3WithSM2 error')
```

```
ret,sign = Sm2SignWithDigest(pubKey,priKey,digest)
if ret == 0:
print('Sm2SignWithDigest ok')
else:
print('Sm2SignWithDigest error')

ret = Sm2VerifyWithDigest(pubKey,sign,digest)
if ret == 0:
print('Sm2VerifyWithDigest ok')
else:
print('Sm2VerifyWithDigest error')


ret,hmac = Sm3Hmac(b'hello',key)
if ret == 0:
print ('Sm3Hmac is ok:',hmac)
else:
print ('Sm3Hmac is error',ret)

ret,digest = Sm3Digest(b'hello')
if ret == 0:
print ('Sm3Digest is ok:',digest)
else:
print ('Sm3Digest is error',ret)

ret,en_data = Sm4CbcEncrypt(b'this Sm4CbcTest',key,iv)
if ret == 0:
ret,de_data = Sm4CbcDecrypt(en_data,key,iv)
if ret == 0:
print (de_data)
else:
print ('Sm4CbcDecrypt is error',ret)
else:
print ('Sm4CbcEncrypt is error',ret)



ret,en_data = Sm4CtrEncrypt(b'this Sm4CtrTest',key,iv)
if ret == 0:
ret,de_data = Sm4CtrDecrypt(en_data,key,iv)
if ret == 0:
print (de_data)
else:
print ('Sm4CtrDecrypt is error',ret)
else:
print ('Sm4CtrEncrypt is error',ret)



ret,en_data = Sm4EcbEncrypt(b'this Sm4EcbTest',key)
if ret == 0:
ret,de_data = Sm4EcbDecrypt(en_data,key)
if ret == 0:
print (de_data)
else:
print ('Sm4EcbDecrypt is error',ret)
```

```
else:
print ('Sm4EcbEncrypt is error',ret)

ret,en_data,tag = Sm4GcmEncrypt(b'this Sm4GcmTest',key,iv,aad)
if ret == 0:
ret,de_data = Sm4GcmDecrypt(en_data,key,iv,aad,tag)
if ret == 0:
print (de_data)
else:
print ('Sm4GcmDecrypt is error',ret)
else:
print ('Sm4GcmEncrypt is error',ret)
```

# API文档
# 密钥管理系统（kms）
# 版本（2019-01-18）
# API概览

最近更新时间: 2025-02-18 17:50:22

## API版本

V3

## 其他接口

| 接口名称 | 接口功能 |
| --- | --- |
| DescribeTSMLicenses | 获取TSM SDK授权的各个版本License |

## 密钥相关接口

| 接口名称 | 接口功能 |
| --- | --- |
| ArchiveKey | 密钥归档 |
| BindCloudResource | 绑定密钥和云产品资源的使用关系 |
| CancelKeyArchive | 取消密钥归档 |
| CancelKeyDeletion | 取消CMK计划删除操作 |
| CreateKey | 创建主密钥 |
| Decrypt | 解密 |
| DeleteImportedKeyMaterial | 删除导入的密钥材料 |
| DescribeKey | 获取主密钥属性 |
| DescribeKeys | 获取多个主密钥属性 |
| DisableKey | 禁用主密钥 |
| DisableKeyRotation | 禁止密钥轮换 |
| DisableKeys | 批量禁用主密钥 |
| EnableKey | 启用主密钥 |

版权所有：腾讯云金融专区

| 接口名称 | 接口功能 |
| --- | --- |
| EnableKeyRotation | 开启密钥轮换 |
| EnableKeys | 批量启动主密钥 |
| Encrypt | 加密 |
| GenerateAsymmetricKeyPair | 生成非对称密钥对 |
| GenerateDataKey | 生成数据密钥 |
| GenerateRandom | 随机数生成接口 |
| GetKeyRotationStatus | 查询密钥轮换状态 |
| GetParametersForImport | 获取导入主密钥（CMK）材料的参数 |
| GetRegions | 获取支持的地域列表 |
| GetServiceStatus | 查询服务状态 |
| ImportKeyMaterial | 导入密钥材料 |
| ListAlgorithms | 列出当前Region支持的加密方式 |
| ListKeyDetail | 获取主密钥列表详情 |
| ListKeys | 获取主密钥列表 |
| ReEncrypt | 密文刷新 |
| ScheduleKeyDeletion | CMK计划删除接口 |
| SignByAsymmetricKey | 签名 |
| UnbindCloudResource | 解绑CMK和云资源的关联关系 |
| UpdateAlias | 修改别名 |
| UpdateKeyDescription | 修改主密钥描述信息 |

## 非对称密钥相关接口

| 接口名称 | 接口功能 |
| --- | --- |
| AsymmetricRsaDecrypt | 非对称密钥RSA解密 |
| AsymmetricSm2Decrypt | 非对称密钥Sm2解密 |
| AsymmetricSm2Encrypt | 加密 |
| GetPublicKey | 获取非对称密钥的公钥 |

# 调用方式
# 接口签名v1

最近更新时间: 2025-02-18 17:50:22

tcecloud API 会对每个访问请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature）以验证请求者身份。 签名信息由安全凭证生成，安全凭证包括 SecretId 和 SecretKey；若用户还没有安全凭证，请前往云API密钥页面申请，否则无法调用云API接口。

## 1. 申请安全凭证

在第一次使用云API之前，请前往云API密钥页面申请安全凭证。 安全凭证包括 SecretId 和 SecretKey：

- SecretId 用于标识 API 调用者身份
- SecretKey 用于加密签名字符串和服务器端验证签名字符串的密钥。
- 用户必须严格保管安全凭证，避免泄露。

申请安全凭证的具体步骤如下：

1. 登录tcecloud管理中心控制台。
2. 前往云API密钥的控制台页面
3. 在云API密钥页面，点击【新建】即可以创建一对SecretId/SecretKey

注意：开发商帐号最多可以拥有两对 SecretId / SecretKey。

## 2. 生成签名串

有了安全凭证SecretId 和 SecretKey后，就可以生成签名串了。以下是生成签名串的详细过程：

假设用户的 SecretId 和 SecretKey 分别是：

- SecretId: AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE
- SecretKey: Gu5t9xGARNpq86cd98joQYCN3EXAMPLE

**注意：这里只是示例，请根据用户实际申请的 SecretId 和 SecretKey 进行后续操作！**

以云服务器查看实例列表(DescribeInstances)请求为例，当用户调用这一接口时，其请求参数可能如下:

| 参数名称 | 中文 | 参数值 |
| --- | --- | --- |
| Action | 方法名 | DescribeInstances |
| SecretId | 密钥Id | AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE |
| Timestamp | 当前时间戳 | 1465185768 |
| Nonce | 随机正整数 | 11886 |
| Region | 实例所在区域 | ap-guangzhou |

| 参数名称 | 中文 | 参数值 |
| --- | --- | --- |
| InstanceIds.0 | 待查询的实例ID | ins-09dx96dg |
| Offset | 偏移量 | 0 |
| Limit | 最大允许输出 | 20 |
| Version | 接口版本号 | 2017-03-12 |

### 2.1. 对参数排序

首先对所有请求参数按参数名的字典序（ ASCII 码）升序排序。注意：1）只按参数名进行排序，参数值保持对应即可，不参与比大小；2）按 ASCII 码比大小，如 InstanceIds.2 要排在 InstanceIds.12 后面，不是按字母表，也不是按数值。用户可以借助编程语言中的相关排序函数来实现这一功能，如 php 中的 ksort 函数。上述示例参数的排序结果如下:

```
{
'Action' : 'DescribeInstances',
'InstanceIds.0' : 'ins-09dx96dg',
'Limit' : 20,
'Nonce' : 11886,
'Offset' : 0,
'Region' : 'ap-guangzhou',
'SecretId' : 'AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE',
'Timestamp' : 1465185768,
'Version': '2017-03-12',
}
```

使用其它程序设计语言开发时，可对上面示例中的参数进行排序，得到的结果一致即可。

### 2.2. 拼接请求字符串

此步骤生成请求字符串。 将把上一步排序好的请求参数格式化成"参数名称"="参数值"的形式，如对 Action 参数，其参数名称为 "Action" ，参数值为 "DescribeInstances" ，因此格式化后就为 Action=DescribeInstances 。 **注意：“参数值”为原始值而非url编码后的值。**

然后将格式化后的各个参数用"&"拼接在一起，最终生成的请求字符串为:

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Timestamp=1465185768&Version=2017-03-12
```

### 2.3. 拼接签名原文字符串

此步骤生成签名原文字符串。 签名原文字符串由以下几个参数构成:

1. 请求方法: 支持 POST 和 GET 方式，这里使用 GET 请求，注意方法为全大写。
2. 请求主机:查看实例列表(DescribeInstances)的请求域名为：cvm.finance.cloud.tencent.com。实际的请求域名根据接口所属模块的不同而不同，详见各接口说明。
3. 请求路径: 当前版本云API的请求路径固定为 / 。
4. 请求字符串: 即上一步生成的请求字符串。

签名原文串的拼接规则为: 请求方法 + 请求主机 +请求路径 + ? + 请求字符串

示例的拼接结果为：

```
GETcvm.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Timestamp=1465185768&Version=2017-03-12
```

### 2.4. 生成签名串

此步骤生成签名串。 首先使用 HMAC-SHA1 算法对上一步中获得的**签名原文字符串**进行签名，然后将生成的签名串使用 Base64 进行编码，即可获得最终的签名串。

具体代码如下，以 PHP 语言为例:

```
$secretKey = 'Gu5t9xGARNpq86cd98joQYCN3EXAMPLE';
$srcStr = 'GETcvm.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Timestamp=1465185768&Version=2017-03-12';
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

最终得到的签名串为:

```
EliP9YW3pW28FpsEdkXt/+WcGeI=
```

使用其它程序设计语言开发时，可用上面示例中的原文进行签名验证，得到的签名串与例子中的一致即可。

## 3. 签名串编码

生成的签名串并不能直接作为请求参数，需要对其进行 URL 编码。

如上一步生成的签名串为 EliP9YW3pW28FpsEdkXt/+WcGeI= ，最终得到的签名串请求参数（Signature）为：EliP9YW3pW28FpsEdkXt%2f%2bWcGeI%3d，它将用于生成最终的请求 URL。

**注意：如果用户的请求方法是 GET，或者请求方法为 POST 同时 Content-Type 为 application/x-www-form-urlencoded，则发送请求时所有请求参数的值均需要做 URL 编码，参数键和=符号不需要编码。非 ASCII 字符在 URL 编码前需要先以 UTF-8 进行编码。**

**注意：有些编程语言的 http 库会自动为所有参数进行 urlencode，在这种情况下，就不需要对签名串进行 URL 编码了，否则两次 URL 编码会导致签名失败。**

**注意：其他参数值也需要进行编码，编码采用 RFC 3986。使用 %XY 对特殊字符例如汉字进行百分比编码，其中"X"和"Y"为十六进制字符（0-9 和大写字母 A-F），使用小写将引发错误。**

## 4. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理

| 错误代码 | 错误描述 |
| --- | --- |
| AuthFailure.SignatureExpire | 签名过期 |
| AuthFailure.SecretIdNotFound | 密钥不存在 |
| AuthFailure.SignatureFailure | 签名错误 |

| 错误代码 | 错误描述 |
| --- | --- |
| AuthFailure.TokenFailure | token 错误 |
| AuthFailure.InvalidSecretId | 密钥非法（不是云 API 密钥类型） |

## 5. 签名演示

在实际调用 API 3.0 时，推荐使用配套的tcecloud SDK 3.0 ，SDK 封装了签名的过程，开发时只关注产品提供的具体接口即可。详细信息参见 SDK 中心。当前支持的编程语言有：

- Python
- Java
- PHP
- Go
- JavaScript
- .NET

为了更清楚的解释签名过程，下面以实际编程语言为例，将上述的签名过程具体实现。请求的域名、调用的接口和参数的取值都以上述签名过程为准，代码只为解释签名过程，并不具备通用性，实际开发请尽量使用 SDK 。

最终输出的 url 可能为： http://imgcache.finance.cloud.tencent.com:80cvm.finance.cloud.tencent.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&Region=ap-guangzhou&SecretId=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE&Signature=EliP9YW3pW28FpsEdkXt%2F%2BWcGeI%3D&Timestamp=1465185768&Version=2017-03-12

注意：由于示例中的密钥是虚构的，时间戳也不是系统当前时间，因此如果将此 url 在浏览器中打开或者用 curl 等命令调用时会返回鉴权错误：签名过期。为了得到一个可以正常返回的 url ，需要修改示例中的 SecretId 和 SecretKey 为真实的密钥，并使用系统当前时间戳作为 Timestamp 。

注意：在下面的示例中，不同编程语言，甚至同一语言每次执行得到的 url 可能都有所不同，表现为参数的顺序不同，但这并不影响正确性。只要所有参数都在，且签名计算正确即可。

注意：以下代码仅适用于 API 3.0，不能直接用于其他的签名流程，即使是旧版的 API ，由于存在细节差异也会导致签名计算错误，请以对应的实际文档为准。

### Java

```
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.Random;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;

public class TceCloudAPIDemo {
private final static String CHARSET = "UTF-8";

public static String sign(String s, String key, String method) throws Exception {
Mac mac = Mac.getInstance(method);
```

```
SecretKeySpec secretKeySpec = new SecretKeySpec(key.getBytes(CHARSET), mac.getAlgorithm());
mac.init(secretKeySpec);
byte[] hash = mac.doFinal(s.getBytes(CHARSET));
return DatatypeConverter.printBase64Binary(hash);
}

public static String getStringToSign(TreeMap<String, Object> params) {
StringBuilder s2s = new StringBuilder("GETcvm.finance.cloud.tencent.com/?");
// 签名时要求对参数进行字典排序，此处用TreeMap保证顺序
for (String k : params.keySet()) {
s2s.append(k).append("=").append(params.get(k).toString()).append("&");
}
return s2s.toString().substring(0, s2s.length() - 1);
}

public static String getUrl(TreeMap<String, Object> params) throws UnsupportedEncodingException {
StringBuilder url = new StringBuilder("http://imgcache.finance.cloud.tencent.com:80cvm.finance.cloud.tencent.com/?");
// 实际请求的url中对参数顺序没有要求
for (String k : params.keySet()) {
// 需要对请求串进行urlencode，由于key都是英文字母，故此处仅对其value进行urlencode
url.append(k).append("=").append(URLEncoder.encode(params.get(k).toString(), CHARSET)).append("&");
}
return url.toString().substring(0, url.length() - 1);
}

public static void main(String[] args) throws Exception {
TreeMap<String, Object> params = new TreeMap<String, Object>(); // TreeMap可以自动排序
// 实际调用时应当使用随机数，例如：params.put("Nonce", new Random().nextInt(java.lang.Integer.MAX_VALUE));
params.put("Nonce", 11886); // 公共参数
// 实际调用时应当使用系统当前时间，例如： params.put("Timestamp", System.currentTimeMillis() / 1000);
params.put("Timestamp", 1465185768); // 公共参数
params.put("SecretId", "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"); // 公共参数
params.put("Action", "DescribeInstances"); // 公共参数
params.put("Version", "2017-03-12"); // 公共参数
params.put("Region", "ap-guangzhou"); // 公共参数
params.put("Limit", 20); // 业务参数
params.put("Offset", 0); // 业务参数
params.put("InstanceIds.0", "ins-09dx96dg"); // 业务参数
params.put("Signature", sign(getStringToSign(params), "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE", "HmacSHA1")); // 公共
参数
System.out.println(getUrl(params));
}
}
```

### Python

注意：如果是在 Python 2 环境中运行，需要先安装 requests 依赖包： `pip install requests` 。

```
# -*- coding: utf8 -*-
import base64
import hashlib
import hmac
import time

import requests
```

```
secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

def get_string_to_sign(method, endpoint, params):
s = method + endpoint + "/?"
query_str = "&".join("%s=%s" % (k, params[k]) for k in sorted(params))
return s + query_str

def sign_str(key, s, method):
hmac_str = hmac.new(key.encode("utf8"), s.encode("utf8"), method).digest()
return base64.b64encode(hmac_str)

if __name__ == '__main__':
endpoint = "cvm.finance.cloud.tencent.com"
data = {
'Action' : 'DescribeInstances',
'InstanceIds.0' : 'ins-09dx96dg',
'Limit' : 20,
'Nonce' : 11886,
'Offset' : 0,
'Region' : 'ap-guangzhou',
'SecretId' : secret_id,
'Timestamp' : 1465185768, # int(time.time())
'Version': '2017-03-12'
}
s = get_string_to_sign("GET", endpoint, data)
data["Signature"] = sign_str(secret_key, s, hashlib.sha1)
print(data["Signature"])
# 此处会实际调用，成功后可能产生计费
# resp = requests.get("http://imgcache.finance.cloud.tencent.com:80" + endpoint, params=data)
# print(resp.url)
```

# 接口签名v3

最近更新时间: 2025-02-18 17:50:22

tcecloud API 会对每个访问请求进行身份验证，即每个请求都需要在公共请求参数中包含签名信息（Signature）以验证请求者身份。 签名信息由安全凭证生成，安全凭证包括 SecretId 和 SecretKey；若用户还没有安全凭证，请前往云API密钥页面申请，否则无法调用云API接口。

## 1. 申请安全凭证

在第一次使用云API之前，请前往云API密钥页面申请安全凭证。 安全凭证包括 SecretId 和 SecretKey：

- SecretId 用于标识 API 调用者身份
- SecretKey 用于加密签名字符串和服务器端验证签名字符串的密钥。
- 用户必须严格保管安全凭证，避免泄露。

申请安全凭证的具体步骤如下：

1. 登录tcecloud管理中心控制台。
2. 前往云API密钥的控制台页面
3. 在云API密钥页面，点击【新建】即可以创建一对SecretId/SecretKey

注意：开发商帐号最多可以拥有两对 SecretId / SecretKey。

## 2. TC3-HMAC-SHA256 签名方法

注意：对于GET方法，只支持 Content-Type: application/x-www-form-urlencoded 协议格式。对于POST方法，目前支持 Content-Type: application/json 以及 Content-Type: multipart/form-data 两种协议格式，json 格式默认所有业务接口均支持，multipart 格式只有特定业务接口支持，此时该接口不能使用 json 格式调用，参考具体业务接口文档说明。

下面以云服务器查询广州区实例列表作为例子，分步骤介绍签名的计算过程。我们仅用到了查询实例列表的两个参数：Limit 和 Offset，使用 GET 方法调用。

假设用户的 SecretId 和 SecretKey 分别是：AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE 和 Gu5t9xGARNpq86cd98joQYCN3EXAMPLE

### 2.1. 拼接规范请求串

按如下格式拼接规范请求串（CanonicalRequest）：

```
CanonicalRequest =
HTTPRequestMethod + '\n' +
CanonicalURI + '\n' +
CanonicalQueryString + '\n' +
CanonicalHeaders + '\n' +
SignedHeaders + '\n' +
HashedRequestPayload
```

- HTTPRequestMethod：HTTP 请求方法（GET、POST ），本示例中为 GET；

- CanonicalURI：URI 参数，API 3.0 固定为正斜杠（/）；
- CanonicalQueryString：发起 HTTP 请求 URL 中的查询字符串，对于 POST 请求，固定为空字符串，对于 GET 请求，则为 URL 中问号（?）后面的字符串内容，本示例取值为：Limit=10&Offset=0。注意：CanonicalQueryString 需要经过 URL 编码。
- CanonicalHeaders：参与签名的头部信息，至少包含 host 和 content-type 两个头部，也可加入自定义的头部参与签名以提高自身请求的唯一性和安全性。拼接规则：1）头部 key 和 value 统一转成小写，并去掉首尾空格，按照 key:value\n 格式拼接；2）多个头部，按照头部 key（小写）的字典排序进行拼接。此例中为：`content-type:application/x-www-form-urlencoded\nhost:cvm.finance.cloud.tencent.com\n`
- SignedHeaders：参与签名的头部信息，说明此次请求有哪些头部参与了签名，和 CanonicalHeaders 包含的头部内容是一一对应的。content-type 和 host 为必选头部。拼接规则：1）头部 key 统一转成小写；2）多个头部 key（小写）按照字典排序进行拼接，并且以分号（;）分隔。此例中为：`content-type;host`
- HashedRequestPayload：请求正文的哈希值，计算方法为 Lowercase(HexEncode(Hash.SHA256(RequestPayload)))，对 HTTP 请求整个正文 payload 做 SHA256 哈希，然后十六进制编码，最后编码串转换成小写字母。注意：对于 GET 请求，RequestPayload 固定为空字符串，对于 POST 请求，RequestPayload 即为 HTTP 请求正文 payload。

根据以上规则，示例中得到的规范请求串如下（为了展示清晰，\n 换行符通过另起打印新的一行替代）：

```
GET
/
Limit=10&Offset=0
content-type:application/x-www-form-urlencoded
host:cvm.finance.cloud.tencent.com

content-type;host
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

### 2.2. 拼接待签名字符串

按如下格式拼接待签名字符串：

```
StringToSign =
Algorithm + \n +
RequestTimestamp + \n +
CredentialScope + \n +
HashedCanonicalRequest
```

- Algorithm：签名算法，目前固定为 TC3-HMAC-SHA256；
- RequestTimestamp：请求时间戳，即请求头部的 X-TC-Timestamp 取值，如上示例请求为 1539084154；
- CredentialScope：凭证范围，格式为 Date/service/tc3_request，包含日期、所请求的服务和终止字符串（tc3_request）。**Date 为 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致**；service 为产品名，必须与调用的产品域名一致，例如 cvm。如上示例请求，取值为 2018-10-09/cvm/tc3_request；
- HashedCanonicalRequest：前述步骤拼接所得规范请求串的哈希值，计算方法为 Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))。

注意：

> 1. Date 必须从时间戳 X-TC-Timestamp 计算得到，且时区为 UTC+0。如果加入系统本地时区信息，例如东八区，将导致白天和晚上调用成功，但是凌晨时调用必定失败。假设时间戳为 1551113065，在东八区的时间是 2019-02-26 00:44:25，但是计算得到的 Date 取 UTC+0 的日期应为 2019-02-25，而不是 2019-02-26。

> 2. Timestamp 必须是当前系统时间，且需确保系统时间和标准时间是同步的，如果相差超过五分钟则必定失败。如果长时间不和标准时间同步，可能导致运行一段时间后，请求必定失败（返回签名过期错误）。

根据以上规则，示例中得到的待签名字符串如下（为了展示清晰，\n 换行符通过另起打印新的一行替代）：

```
TC3-HMAC-SHA256
1539084154
2018-10-09/cvm/tc3_request
91c9c192c14460df6c1ffc69e34e6c5e90708de2a6d282cccf957dbf1aa7f3a7
```

### 2.3. 计算签名

1）计算派生签名密钥，伪代码如下

```
SecretKey = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"
SecretDate = HMAC_SHA256("TC3" + SecretKey, Date)
SecretService = HMAC_SHA256(SecretDate, Service)
SecretSigning = HMAC_SHA256(SecretService, "tc3_request")
```

- SecretKey：原始的 SecretKey；
- Date：即 Credential 中的 Date 字段信息，如上示例，为2018-10-09；
- Service：即 Credential 中的 Service 字段信息，如上示例，为 cvm；

2）计算签名，伪代码如下

```
Signature = HexEncode(HMAC_SHA256(SecretSigning, StringToSign))
```

- SecretSigning：即以上计算得到的派生签名密钥；
- StringToSign：即步骤2计算得到的待签名字符串；

### 2.4. 拼接 Authorization

按如下格式拼接 Authorization：

```
Authorization =
Algorithm + ' ' +
'Credential=' + SecretId + '/' + CredentialScope + ', ' +
'SignedHeaders=' + SignedHeaders + ', '
'Signature=' + Signature
```

- Algorithm：签名方法，固定为 TC3-HMAC-SHA256；
- SecretId：密钥对中的 SecretId；
- CredentialScope：见上文，凭证范围；
- SignedHeaders：见上文，参与签名的头部信息；
- Signature：签名值

根据以上规则，示例中得到的值为：

```
TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
```

最终完整的调用信息如下：

```
http://imgcache.finance.cloud.tencent.com:80cvm.finance.cloud.tencent.com/?Limit=10&Offset=0

Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE/2018-10-09/cvm/tc3_request, SignedHeaders=content-type;host, Signature=5da7a33f6993f0614b047e5df4582db9e9bf4672ba50567dba16c6ccf174c474
Content-Type: application/x-www-form-urlencoded
Host: cvm.finance.cloud.tencent.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1539084154
X-TC-Region: ap-guangzhou
```

## 3. 签名失败

根据实际情况，存在以下签名失败的错误码，请根据实际情况处理

| 错误代码 | 错误描述 |
| --- | --- |
| AuthFailure.SignatureExpire | 签名过期 |
| AuthFailure.SecretIdNotFound | 密钥不存在 |
| AuthFailure.SignatureFailure | 签名错误 |
| AuthFailure.TokenFailure | token 错误 |
| AuthFailure.InvalidSecretId | 密钥非法（不是云 API 密钥类型） |

## 4. 签名演示

### Java

```
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Map;
import java.util.TimeZone;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.net.ssl.HttpsURLConnection;
import javax.xml.bind.DatatypeConverter;

import org.apache.commons.codec.digest.DigestUtils;

public class TceCloudAPITC3Demo {
private final static String CHARSET = "UTF-8";
```

```
private final static String ENDPOINT = "cvm.finance.cloud.tencent.com";
private final static String PATH = "/";
private final static String SECRET_ID = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE";
private final static String SECRET_KEY = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE";
private final static String CT_X_WWW_FORM_URLENCODED = "application/x-www-form-urlencoded";
private final static String CT_JSON = "application/json";
private final static String CT_FORM_DATA = "multipart/form-data";

public static byte[] sign256(byte[] key, String msg) throws Exception {
Mac mac = Mac.getInstance("HmacSHA256");
SecretKeySpec secretKeySpec = new SecretKeySpec(key, mac.getAlgorithm());
mac.init(secretKeySpec);
return mac.doFinal(msg.getBytes(CHARSET));
}

public static void main(String[] args) throws Exception {
String service = "cvm";
String host = "cvm.finance.cloud.tencent.com";
String region = "ap-guangzhou";
String action = "DescribeInstances";
String version = "2017-03-12";
String algorithm = "TC3-HMAC-SHA256";
String timestamp = "1539084154";
//String timestamp = String.valueOf(System.currentTimeMillis() / 1000);
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
// 注意时区，否则容易出错
sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
String date = sdf.format(new Date(Long.valueOf(timestamp + "000")));

// ************ 步骤 1：拼接规范请求串 ************
String httpRequestMethod = "GET";
String canonicalUri = "/";
String canonicalQueryString = "Limit=10&Offset=0";
String canonicalHeaders = "content-type:application/x-www-form-urlencoded\n" + "host:" + host + "\n";
String signedHeaders = "content-type;host";
String hashedRequestPayload = DigestUtils.sha256Hex("");
String canonicalRequest = httpRequestMethod + "\n" + canonicalUri + "\n" + canonicalQueryString + "\n"
+ canonicalHeaders + "\n" + signedHeaders + "\n" + hashedRequestPayload;
System.out.println(canonicalRequest);

// ************ 步骤 2：拼接待签名字符串 ************
String credentialScope = date + "/" + service + "/" + "tc3_request";
String hashedCanonicalRequest = DigestUtils.sha256Hex(canonicalRequest.getBytes(CHARSET));
String stringToSign = algorithm + "\n" + timestamp + "\n" + credentialScope + "\n" + hashedCanonicalRequest;
System.out.println(stringToSign);

// ************ 步骤 3：计算签名 ************
byte[] secretDate = sign256(("TC3" + SECRET_KEY).getBytes(CHARSET), date);
byte[] secretService = sign256(secretDate, service);
byte[] secretSigning = sign256(secretService, "tc3_request");
String signature = DatatypeConverter.printHexBinary(sign256(secretSigning, stringToSign)).toLowerCase();
System.out.println(signature);

// ************ 步骤 4：拼接 Authorization ************
String authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
+ "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
System.out.println(authorization);
```

```
TreeMap<String, String> headers = new TreeMap<String, String>();
headers.put("Authorization", authorization);
headers.put("Host", host);
headers.put("Content-Type", CT_X_WWW_FORM_URLENCODED);
headers.put("X-TC-Action", action);
headers.put("X-TC-Timestamp", timestamp);
headers.put("X-TC-Version", version);
headers.put("X-TC-Region", region);
}
}
```

### Python

```
# -*- coding: utf-8 -*-
import hashlib, hmac, json, os, sys, time
from datetime import datetime

# 密钥参数
secret_id = "AKIDz8krbsJ5yKBZQpn74WFkmLPx3EXAMPLE"
secret_key = "Gu5t9xGARNpq86cd98joQYCN3EXAMPLE"

service = "cvm"
host = "cvm.finance.cloud.tencent.com"
endpoint = "http://imgcache.finance.cloud.tencent.com:80" + host
region = "ap-guangzhou"
action = "DescribeInstances"
version = "2017-03-12"
algorithm = "TC3-HMAC-SHA256"
timestamp = 1539084154
date = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
params = {"Limit": 10, "Offset": 0}

# ************* 步骤 1：拼接规范请求串 *************
http_request_method = "GET"
canonical_uri = "/"
canonical_querystring = "Limit=10&Offset=0"
ct = "x-www-form-urlencoded"
payload = ""
if http_request_method == "POST":
canonical_querystring = ""
ct = "json"
payload = json.dumps(params)
canonical_headers = "content-type:application/%s\nhost:%s\n" % (ct, host)
signed_headers = "content-type;host"
hashed_request_payload = hashlib.sha256(payload.encode("utf-8")).hexdigest()
canonical_request = (http_request_method + "\n" +
canonical_uri + "\n" +
canonical_querystring + "\n" +
canonical_headers + "\n" +
signed_headers + "\n" +
hashed_request_payload)
print(canonical_request)

# ************* 步骤 2：拼接待签名字符串 *************
credential_scope = date + "/" + service + "/" + "tc3_request"
```

```
hashed_canonical_request = hashlib.sha256(canonical_request.encode("utf-8")).hexdigest()
string_to_sign = (algorithm + "\n" +
str(timestamp) + "\n" +
credential_scope + "\n" +
hashed_canonical_request)
print(string_to_sign)


# ************* 步骤 3：计算签名 *************
# 计算签名摘要函数
def sign(key, msg):
return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()
secret_date = sign(("TC3" + secret_key).encode("utf-8"), date)
secret_service = sign(secret_date, service)
secret_signing = sign(secret_service, "tc3_request")
signature = hmac.new(secret_signing, string_to_sign.encode("utf-8"), hashlib.sha256).hexdigest()
print(signature)

# ************* 步骤 4：拼接 Authorization *************
authorization = (algorithm + " " +
"Credential=" + secret_id + "/" + credential_scope + ", " +
"SignedHeaders=" + signed_headers + ", " +
"Signature=" + signature)
print(authorization)

# 公共参数添加到请求头部
headers = {
"Authorization": authorization,
"Host": host,
"Content-Type": "application/%s" % ct,
"X-TC-Action": action,
"X-TC-Timestamp": str(timestamp),
"X-TC-Version": version,
"X-TC-Region": region,
}
```

# 请求结构

最近更新时间: 2025-02-18 17:50:22

## 1. 服务地址

地域（Region）是指物理的数据中心的地理区域。tcecloud交付验证不同地域之间完全隔离，保证不同地域间最大程度的稳定性和容错性。为了降低访问时延、提高下载速度，建议您选择最靠近您客户的地域。

您可以通过 API接口 **查询地域列表** 查看完成的地域列表。

## 2. 通信协议

tcecloud API 的所有接口均通过 HTTPS 进行通信，提供高安全性的通信通道。

## 3. 请求方法

支持的 HTTP 请求方法:

- POST（推荐）
- GET

POST 请求支持的 Content-Type 类型：

- application/json（推荐），必须使用 TC3-HMAC-SHA256 签名方法。
- application/x-www-form-urlencoded，必须使用 HmacSHA1 或 HmacSHA256 签名方法。
- multipart/form-data（仅部分接口支持），必须使用 TC3-HMAC-SHA256 签名方法。

GET 请求的请求包大小不得超过 32 KB。POST 请求使用签名方法为 HmacSHA1、HmacSHA256 时不得超过 1 MB。POST 请求使用签名方法为 TC3-HMAC-SHA256 时支持 10 MB。

## 4. 字符编码

均使用UTF-8编码。

# 返回结果

最近更新时间: 2025-02-18 17:50:22

## 正确返回结果

以云服务器的接口查看实例状态列表 (DescribeInstancesStatus) 2017-03-12 版本为例，若调用成功，其可能的返回如下为：

```
{
"Response": {
"TotalCount": 0,
"InstanceStatusSet": [],
"RequestId": "b5b41468-520d-4192-b42f-595cc34b6c1c"
}
}
```

- Response 及其内部的 RequestId 是固定的字段，无论请求成功与否，只要 API 处理了，则必定会返回。
- RequestId 用于一个 API 请求的唯一标识，如果 API 出现异常，可以联系我们，并提供该 ID 来解决问题。
- 除了固定的字段外，其余均为具体接口定义的字段，不同的接口所返回的字段参见接口文档中的定义。此例中的 TotalCount 和 InstanceStatusSet 均为 DescribeInstancesStatus 接口定义的字段，由于调用请求的用户暂时还没有云服务器实例，因此 TotalCount 在此情况下的返回值为 0， InstanceStatusSet 列表为空。

## 错误返回结果

若调用失败，其返回值示例如下为：

```
{
"Response": {
"Error": {
"Code": "AuthFailure.SignatureFailure",
"Message": "The provided credentials could not be validated. Please check your signature is correct."
},
"RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
}
}
```

- Error 的出现代表着该请求调用失败。Error 字段连同其内部的 Code 和 Message 字段在调用失败时是必定返回的。
- Code 表示具体出错的错误码，当请求出错时可以先根据该错误码在公共错误码和当前接口对应的错误码列表里面查找对应原因和解决方案。
- Message 显示出了这个错误发生的具体原因，随着业务发展或体验优化，此文本可能会经常保持变更或更新，用户不应依赖这个返回值。
- RequestId 用于一个 API 请求的唯一标识，如果 API 出现异常，可以联系我们，并提供该 ID 来解决问题。

## 公共错误码 (TODO: 重复信息, 是否真的需要?)

返回结果中如果存在 Error 字段，则表示调用 API 接口失败。 Error 中的 Code 字段表示错误码，所有业务都可能出现的错误码为公共错误码，下表列出了公共错误码。

| 错误码 | 错误描述 |
| --- | --- |
| AuthFailure.InvalidSecretId | 密钥非法（不是云 API 密钥类型）。 |
| AuthFailure.MFAFailure | MFA 错误。 |
| AuthFailure.SecretIdNotFound | 密钥不存在。 |
| AuthFailure.SignatureExpire | 签名过期。 |
| AuthFailure.SignatureFailure | 签名错误。 |
| AuthFailure.TokenFailure | token 错误。 |
| AuthFailure.UnauthorizedOperation | 请求未 CAM 授权。 |
| DryRunOperation | DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。 |
| FailedOperation | 操作失败。 |
| InternalError | 内部错误。 |
| InvalidAction | 接口不存在。 |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |
| LimitExceeded | 超过配额限制。 |
| MissingParameter | 缺少参数错误。 |
| NoSuchVersion | 接口版本不存在。 |
| RequestLimitExceeded | 请求的次数超过了频率限制。 |
| ResourceInUse | 资源被占用。 |
| ResourceInsufficient | 资源不足。 |
| ResourceNotFound | 资源不存在。 |
| ResourceUnavailable | 资源不可用。 |
| UnauthorizedOperation | 未授权操作。 |
| UnknownParameter | 未知参数错误。 |
| UnsupportedOperation | 操作不支持。 |
| UnsupportedProtocol | http(s)请求协议错误，只支持 GET 和 POST 请求。 |
| UnsupportedRegion | 接口不支持所传地域。 |

版权所有：腾讯云金融专区

# 公共参数

最近更新时间: 2025-02-18 17:50:22

公共参数是用于标识用户和接口鉴权目的的参数，如非必要，在每个接口单独的接口文档中不再对这些参数进行说明，但每次请求均需要携带这些参数，才能正常发起请求。

## 签名方法 v3

使用 TC3-HMAC-SHA256 签名方法时，公共参数需要统一放到 HTTP Header 请求头部中，如下：

| 参数名称 | 类型 | 必选 | 描述 |
| --- | --- | --- | --- |
| X-TC-Action | String | 是 | 操作的接口名称。取值参考接口文档中输入参数公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。 |
| X-TC-Region | String | 是 | 地域参数，用来标识希望操作哪个地域的数据。接口接受的地域取值参考接口文档中输入参数公共参数 Region 的说明。注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。 |
| X-TC-Timestamp | Integer | 是 | 当前 UNIX 时间戳，可记录发起 API 请求的时间。例如 1529223702。注意：如果与服务器时间相差超过5分钟，会引起签名过期错误。 |
| X-TC-Version | String | 是 | 操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。 |
| Authorization | String | 是 | HTTP 标准身份认证头部字段，例如：<br>TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024<br>其中，<br>- TC3-HMAC-SHA256：签名方法，目前固定取该值；<br>- Credential：签名凭证，AKIDEXAMPLE 是 SecretId；Date 是 UTC 标准时间的日期，取值需要和公共参数 X-TC-Timestamp 换算的 UTC 标准时间日期一致；service为产品名，必须与调用的产品域名一致，例如cvm；<br>- SignedHeaders：参与签名计算的头部信息，content-type 和 host 为必选头部；<br>- Signature：签名摘要。 |
| X-TC-Token | String | 否 | 临时证书所用的 Token ，需要结合临时密钥一起使用。临时密钥和 Token 需要到访问管理服务调用接口获取。长期密钥不需要 Token。 |

## 签名方法 v1

使用 HmacSHA1 和 HmacSHA256 签名方法时，公共参数需要统一放到请求串中，如下

| 参数名称 | 类型 | 必选 | 描述 |
| --- | --- | --- | --- |
| Action | String | 是 | 操作的接口名称。取值参考接口文档中输入参数公共参数 Action 的说明。例如云服务器的查询实例列表接口，取值为 DescribeInstances。 |

| 参数名称 | 类型 | 必选 | 描述 |
| --- | --- | --- | --- |
| Region | String | 是 | 地域参数，用来标识希望操作哪个地域的数据。接口接受的地域取值参考接口文档中输入参数公共参数 Region 的说明。注意：某些接口不需要传递该参数，接口文档中会对此特别说明，此时即使传递该参数也不会生效。 |
| Timestamp | Integer | 是 | 当前 UNIX 时间戳，可记录发起 API 请求的时间。例如1529223702，如果与当前时间相差过大，会引起签名过期错误。 |
| Nonce | Integer | 是 | 随机正整数，与 Timestamp 联合起来，用于防止重放攻击。 |
| SecretId | String | 是 | 在云API密钥上申请的标识身份的 SecretId，一个 SecretId 对应唯一的 SecretKey ，而 SecretKey 会用来生成请求签名 Signature。 |
| Signature | String | 是 | 请求签名，用来验证此次请求的合法性，需要用户根据实际的输入参数计算得出。具体计算方法参见接口鉴权文档。 |
| Version | String | 是 | 操作的 API 的版本。取值参考接口文档中入参公共参数 Version 的说明。例如云服务器的版本 2017-03-12。 |
| SignatureMethod | String | 否 | 签名方式，目前支持 HmacSHA256 和 HmacSHA1。只有指定此参数为 HmacSHA256 时，才使用 HmacSHA256 算法验证签名，其他情况均使用 HmacSHA1 验证签名。 |
| Token | String | 否 | 临时证书所用的 Token ，需要结合临时密钥一起使用。临时密钥和 Token 需要到访问管理服务调用接口获取。长期密钥不需要 Token 。 |

## 地域列表

地域（Region）是指物理的数据中心的地理区域。tcecloud交付验证不同地域之间完全隔离，保证不同地域间最大程度的稳定性和容错性。为了降低访问时延、提高下载速度，建议您选择最靠近您客户的地域。

您可以通过 API接口 **查询地域列表** 查看完成的地域列表。

# 其他接口
# 获取TSM SDK授权的各个版本License

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

获取TSM SDK授权的各个版本License

默认接口请求频率限制：20次/秒。

接口更新时间：2021-12-08 16:44:11。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeTSMLicenses |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Licenses | TsmLicense | 返回各个版本的License，如果没有配置License，则返回空数组 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InternalError | |
| UnauthorizedOperation | |

| 错误码 | 描述 |
| --- | --- |
| InvalidParameterValue | |

# 密钥相关接口
# 密钥归档

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

对密钥进行归档，被归档的密钥只能用于解密，不能加密

默认接口请求频率限制：20次/秒。

接口更新时间：2021-03-09 19:27:52。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ArchiveKey |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | CMK唯一标识符 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| UnsupportedOperation.NotUserCreatedCmk | |
| InvalidParameter | |

| 错误码 | 描述 |
| --- | --- |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |
| ResourceUnavailable.CmkStateNotSupport | |
| ResourceUnavailable.CmkNotFound | |
| UnauthorizedOperation | |
| FailedOperation.CmkUsedByCloudProduct | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |

# 绑定密钥和云产品资源的使用关系

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

记录当前key被哪个云产品的那个资源所使用。如果当前key设置了自动过期，则取消该设置，确保当前key不会自动失效。如果当前关联关系已经创建，也返回成功。

默认接口请求频率限制：20次/秒。

接口更新时间：2020-08-11 11:53:50。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：BindCloudResource |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | cmk的ID |
| ProductId | 是 | 否 | String | 云产品的唯一性标识符 |
| ResourceId | 是 | 否 | String | 资源/实例ID，由调用方根据自己的云产品特征来定义，以字符串形式做存储。 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |

| 错误码 | 描述 |
| --- | --- |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |
| ResourceUnavailable.CmkStateNotSupport | |
| FailedOperation.CmkUsedByCloudProduct | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |

# 取消密钥归档

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

取消密钥归档，取消后密钥的状态变为Enabled。

默认接口请求频率限制：20次/秒。

接口更新时间：2021-03-09 19:32:37。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：CancelKeyArchive |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | CMK唯一标识符 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |
| ResourceUnavailable.CmkStateNotSupport | |

| 错误码 | 描述 |
| --- | --- |
| ResourceUnavailable.CmkNotFound | |
| UnauthorizedOperation | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |
| UnsupportedOperation.NotUserCreatedCmk | |

# 取消CMK计划删除操作

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

取消CMK的计划删除操作

默认接口请求频率限制：20次/秒。

接口更新时间：2020-08-11 11:57:46。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：CancelKeyDeletion |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | 需要被取消删除的CMK的唯一标志 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| KeyId | String | 唯一标志被取消删除的CMK。 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |

| 错误码 | 描述 |
| --- | --- |
| UnauthorizedOperation | |
| ResourceUnavailable.CmkNotPendingDelete | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |

# 创建主密钥

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

创建用户管理数据密钥的主密钥CMK（Custom Master Key）。

默认接口请求频率限制：100次/秒。

接口更新时间：2022-09-13 18:35:13。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：CreateKey |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| Description | 否 | 否 | String | CMK 的描述，最大1024字节 |
| Alias | 是 | 否 | String | 作为密钥更容易辨识，更容易被人看懂的别名， 不可为空，1-60个字母数字 - _ 的组合，首字符必须为字母或者数字。以 kms- 作为前缀的用于云产品使用，Alias 不可重复。 |
| KeyUsage | 否 | 否 | String | 指定key的用途，默认为 "ENCRYPT_DECRYPT" 表示创建对称加解密密钥，其它支持用途 “ASYMMETRIC_DECRYPT_RSA_2048” 表示创建用于加解密的RSA2048非对称密钥，“ASYMMETRIC_DECRYPT_SM2” 表示创建用于加解密的SM2非对称密钥。可选值：ENCRYPT_DECRYPT，ASYMMETRIC_DECRYPT_RSA_2048，ASYMMETRIC_DECRYPT_SM2，ASYMMETRIC_SIGN_VERIFY_SM2，ASYMMETRIC_SIGN_VERIFY_RSA_2048，ASYMMETRIC_SIGN_VERIFY_ECC |
| Type | 否 | 否 | Uint64 | 指定key类型，默认为1，1表示默认类型，由KMS创建CMK密钥，2 表示 EXTERNAL 类型，该类型需要用户导入密钥材料，参考 GetParametersForImport 和 ImportKeyMaterial 接口 |
| Tags | 否 | 否 | Array of Tag | 标签列表 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| KeyId | String | CMK的全局唯一标识符 |
| Alias | String | 作为密钥更容易辨识，更容易被人看懂的别名 |
| CreateTime | Uint64 | 密钥创建时间，unix时间戳 |
| Description | String | CMK的描述 |
| KeyState | String | CMK的状态 |
| KeyUsage | String | CMK的用途 |
| TagCode | Uint64 | 标签操作的返回码. 0: 成功；1: 内部错误；2: 业务处理错误 |
| TagMsg | String | 标签操作的返回信息 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| InvalidParameterValue.InvalidAlias | |
| InvalidParameterValue.AliasAlreadyExists | |
| LimitExceeded.CmkLimitExceeded | |
| InvalidParameterValue.InvalidKeyUsage | |
| InvalidParameterValue.InvalidType | |
| InternalError | |
| UnauthorizedOperation | |
| UnsupportedOperation.UnsupportedKeyUsageInCurrentRegion | |
| InvalidParameterValue.TagsNotExisted | |
| InvalidParameterValue.TagKeysDuplicated | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |

# 解密

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

本接口用于解密密文，得到明文数据。

默认接口请求频率限制：300次/秒。

接口更新时间：2020-03-06 10:08:16。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：Decrypt |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| CiphertextBlob | 是 | 否 | String | 待解密的密文数据 |
| EncryptionContext | 否 | 否 | String | key/value对的json字符串，如果Encrypt指定了该参数，则在调用Decrypt API时需要提供同样的参数，最大支持1024字符 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| KeyId | String | CMK的全局唯一标识 |
| Plaintext | String | 解密后的明文。该字段是base64编码的，为了得到原始明文，调用方需要进行base64解码 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkDisabled | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidCiphertext | |
| InternalError | |
| UnauthorizedOperation | |

# 删除导入的密钥材料

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

用于删除导入的密钥材料，仅对EXTERNAL类型的CMK有效，该接口将CMK设置为PendingImport 状态，并不会删除CMK，在重新进行密钥导入后可继续使用。彻底删除CMK请使用 ScheduleKeyDeletion 接口。

默认接口请求频率限制：10次/秒。

接口更新时间：2020-08-11 11:57:01。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DeleteImportedKeyMaterial |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | 指定需要删除密钥材料的EXTERNAL CMK。 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |

| 错误码 | 描述 |
| --- | --- |
| InternalError | |
| UnsupportedOperation.NotExternalCmk | |
| ResourceUnavailable.CmkStateNotSupport | |
| FailedOperation.CmkUsedByCloudProduct | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |

# 获取主密钥属性

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

用于获取指定KeyId的主密钥属性详情信息。

默认接口请求频率限制：100次/秒。

接口更新时间：2020-01-15 14:34:52。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeKey |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | CMK全局唯一标识符 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| KeyMetadata | KeyMetadata | 密钥属性信息 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |

| 错误码 | 描述 |
| --- | --- |
| InternalError | |
| UnauthorizedOperation | |

# 获取多个主密钥属性

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

该接口用于批量获取主密钥属性信息。

默认接口请求频率限制：100次/秒。

接口更新时间：2020-01-15 20:31:52。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DescribeKeys |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyIds | 是 | 否 | Array of String | 查询CMK的ID列表，批量查询一次最多支持100个KeyId |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| KeyMetadatas | KeyMetadata | 返回的属性信息列表 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |

| 错误码 | 描述 |
| --- | --- |
| InternalError | |
| UnauthorizedOperation | |
| InvalidParameterValue.DuplicatedKeyId | |

# 禁用主密钥

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

本接口用于禁用一个主密钥，处于禁用状态的Key无法用于加密、解密操作。

默认接口请求频率限制：100次/秒。

接口更新时间：2020-08-11 11:57:32。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DisableKey |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | CMK唯一标识符 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |

| 错误码 | 描述 |
| --- | --- |
| UnauthorizedOperation | |
| ResourceUnavailable.CmkStateNotSupport | |
| FailedOperation.CmkUsedByCloudProduct | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |

# 禁止密钥轮换

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

对指定的CMK禁止密钥轮换功能。

默认接口请求频率限制：100次/秒。

接口更新时间：2020-03-06 10:18:23。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DisableKeyRotation |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | CMK唯一标识符 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |

| 错误码 | 描述 |
| --- | --- |
| UnauthorizedOperation | |

# 批量禁用主密钥

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

该接口用于批量禁止CMK的使用。

默认接口请求频率限制：100次/秒。

接口更新时间：2020-08-11 11:57:20。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：DisableKeys |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyIds | 是 | 否 | Array of String | 需要批量禁用的CMK Id 列表，CMK数量最大支持100 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |

| 错误码 | 描述 |
| --- | --- |
| UnauthorizedOperation | |
| InvalidParameterValue.DuplicatedKeyId | |
| ResourceUnavailable.CmkStateNotSupport | |
| FailedOperation.CmkUsedByCloudProduct | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |

# 启用主密钥

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

用于启用一个指定的CMK。

默认接口请求频率限制：100次/秒。

接口更新时间：2020-08-11 11:59:03。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：EnableKey |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | CMK唯一标识符 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |

| 错误码 | 描述 |
| --- | --- |
| UnauthorizedOperation | |
| ResourceUnavailable.CmkStateNotSupport | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |

# 开启密钥轮换

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

对指定的CMK开启密钥轮换功能。

默认接口请求频率限制：100次/秒。

接口更新时间：2020-12-04 15:57:09。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：EnableKeyRotation |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | CMK唯一标识符 |
| RotateDays | 否 | 否 | Uint64 | 轮换周期，最小单位：天，取值范围：7~365，默认值为365 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |

| 错误码 | 描述 |
| --- | --- |
| InternalError | |
| UnauthorizedOperation | |
| UnsupportedOperation.ExternalCmkCanNotRotate | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |
| InvalidParameterValue | |

# 批量启动主密钥

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

该接口用于批量启用CMK。

默认接口请求频率限制：100次/秒。

接口更新时间：2020-08-11 11:59:16。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：EnableKeys |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyIds | 是 | 否 | Array of String | 需要批量启用的CMK Id 列表， CMK数量最大支持100 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |

| 错误码 | 描述 |
| --- | --- |
| UnauthorizedOperation | |
| InvalidParameterValue.DuplicatedKeyId | |
| ResourceUnavailable.CmkStateNotSupport | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |

# 加密

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

本接口用于加密最多为4KB任意数据，可用于加密数据库密码，RSA Key，或其它较小的敏感信息。对于应用的数据加密，使用GenerateDataKey生成的DataKey进行本地数据的加解密操作

默认接口请求频率限制：300次/秒。

接口更新时间：2020-01-15 20:19:59。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：Encrypt |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | 调用CreateKey生成的CMK全局唯一标识符 |
| Plaintext | 是 | 否 | String | 被加密的明文数据，该字段必须使用base64编码，原文最大长度支持4K |
| EncryptionContext | 否 | 否 | String | key/value对的json字符串，如果指定了该参数，则在调用Decrypt API时需要提供同样的参数，最大支持1024个字符 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| CiphertextBlob | String | 加密后经过base64编码的密文 |
| KeyId | String | 加密使用的CMK的全局唯一标识 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkDisabled | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InvalidParameterValue.InvalidPlaintext | |
| InternalError | |
| UnauthorizedOperation | |

# 生成非对称密钥对

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

生成非对称密钥对

默认接口请求频率限制：100次/秒。

接口更新时间：2022-09-15 14:59:11。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：GenerateAsymmetricKeyPair |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | CMK全局唯一标识符 |
| KeySpec | 是 | 否 | String | 指定生成非对称密钥对算法。支持：SM2 |
| EncryptionPublicKey | 否 | 否 | String | X509格式的pem类型公钥字符串， 支持 RSA2048 和 SM2 公钥， 用于对返回的明文私钥进行加密。 若为空，则不对明文密钥对加密。 |
| EncryptionAlgorithm | 否 | 否 | String | 非对称加密算法，配合 EncryptionPublicKey 对返回数据进行加密。目前支持：SM2（C1C3C2），SM2_C1C3C2_ASN1(返回SM2C1C3C2 ASN1格式密文)，RSAES_PKCS1_V1_5， RSAES_OAEP_SHA_1，RSAES_OAEP_SHA_256。 若为空，则默认为 SM2。 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| KeyId | String | CMK的全局唯一标识 |
| PrivateKeyPlaintext | String | 若调用时未提供 EncryptionPublicKey，该字段值为P8格式PEM类型私钥明文。 若调用时提供了 EncryptionPublicKey，则该字段值为使用 EncryptionPublicKey 公钥对P8格式PEM类型私钥明文进行非对称加密后的 Base64 编码的密文。需在 Base64 解码后，使用用户上传的公钥对应的私钥进行进一步解密，以获取P8格式PEM类型私钥明文。 |

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| PublicKeyPlaintext | String | P1格式PEM类型返回的明文公钥。 |
| PrivateKeyCiphertextBlob | String | 使用 CMK 加密私钥后的密文，用户需要自行保存该密文，KMS不托管生成的密钥对。可以通过Decrypt接口从密文中获取密钥明文。 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkDisabled | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |
| UnauthorizedOperation | |

# 生成数据密钥

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

本接口生成一个数据密钥，您可以用这个密钥进行本地数据的加密。

默认接口请求频率限制：100次/秒。

接口更新时间：2020-05-07 14:31:37。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：GenerateDataKey |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | CMK全局唯一标识符 |
| KeySpec | 否 | 否 | String | 指定生成Datakey的加密算法以及Datakey大小，AES_128或者AES_256。KeySpec 和 NumberOfBytes 必须指定一个 |
| NumberOfBytes | 否 | 否 | Uint64 | 生成的DataKey的长度，同时指定NumberOfBytes和KeySpec时，以NumberOfBytes为准。最小值为1， 最大值为1024。KeySpec 和 NumberOfBytes 必须指定一个 |
| EncryptionContext | 否 | 否 | String | key/value对的json字符串，如果使用该字段，则返回的DataKey在解密时需要填入相同的字符串 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| KeyId | String | CMK的全局唯一标识 |
| Plaintext | String | 生成的数据密钥DataKey的明文，该明文使用base64进行了编码，需base64解码后作为数据密钥本地使用 |
| CiphertextBlob | String | 数据密钥DataKey加密后的密文，用户需要自行保存该密文，KMS不托管用户的数据密钥。可以通过Decrypt接口从CiphertextBlob中获取数据密钥DataKey明文 |

版权所有：腾讯云金融专区

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkDisabled | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |
| UnauthorizedOperation | |

# 随机数生成接口

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

随机数生成接口。

默认接口请求频率限制：150次/秒。

接口更新时间：2020-01-15 14:32:05。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：GenerateRandom |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| NumberOfBytes | 是 | 否 | Uint64 | 生成的随机数的长度。最小值为1， 最大值为1024。 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Plaintext | String | 生成的随机数的明文，该明文使用base64编码，用户需要使用base64解码得到明文。 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| InternalError | |
| UnauthorizedOperation | |

# 查询密钥轮换状态

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

查询指定的CMK是否开启了密钥轮换功能。

默认接口请求频率限制：100次/秒。

接口更新时间：2020-01-15 20:34:05。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：GetKeyRotationStatus |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | CMK唯一标识符 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| KeyRotationEnabled | Bool | 密钥轮换是否开启 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |

| 错误码 | 描述 |
| --- | --- |
| InternalError | |
| UnauthorizedOperation | |

# 获取导入主密钥（CMK）材料的参数

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

获取导入主密钥（CMK）材料的参数，返回的Token作为执行ImportKeyMaterial的参数之一，返回的PublicKey用于对自主导入密钥材料进行加密。返回的Token和PublicKey 24小时后失效，失效后如需重新导入，需要再次调用该接口获取新的Token和PublicKey。

默认接口请求频率限制：10次/秒。

接口更新时间：2020-08-11 11:59:37。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：GetParametersForImport |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | CMK的唯一标识，获取密钥参数的CMK必须是EXTERNAL类型，即在CreateKey时指定Type=2 类型的CMK。 |
| WrappingAlgorithm | 是 | 否 | String | 指定加密密钥材料的算法，目前支持RSAES_PKCS1_V1_5、RSAES_OAEP_SHA_1、RSAES_OAEP_SHA_256 |
| WrappingKeySpec | 是 | 否 | String | 指定加密密钥材料的类型，目前只支持RSA_2048 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| KeyId | String | CMK的唯一标识，用于指定目标导入密钥材料的CMK。 |
| ImportToken | String | 导入密钥材料需要的token，用于作为 ImportKeyMaterial 的参数。 |
| PublicKey | String | 用于加密密钥材料的RSA公钥，base64编码。使用PublicKey base64解码后的公钥将导入密钥进行加密后作为 ImportKeyMaterial 的参数。 |
| ParametersValidTo | Uint64 | 该导出token和公钥的有效期，超过该时间后无法导入，需要重新调用GetParametersForImport获取。 |

版权所有：腾讯云金融专区

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |
| UnsupportedOperation.NotExternalCmk | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |

# 获取支持的地域列表

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

获取支持的地域列表

默认接口请求频率限制：20次/秒。

接口更新时间：2020-08-24 14:14:04。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：GetRegions |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Regions | String | 可用region列表 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InternalError | |

# 查询服务状态

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

用于查询该用户是否已开通KMS服务

默认接口请求频率限制：50次/秒。

接口更新时间：2021-03-09 19:46:09。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：GetServiceStatus |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| ServiceEnabled | Bool | KMS服务是否开通， true 表示已开通 |
| InvalidType | Int64 | 服务不可用类型： 0-未购买，1-正常， 2-欠费停服， 3-资源释放 |
| UserLevel | Uint64 | 用户等级 |
| ProExpireTime | Uint64 | 过期时间 |
| ProRenewFlag | Uint64 | 标志 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |

| 错误码 | 描述 |
| --- | --- |
| InternalError | |
| UnauthorizedOperation | |

# 导入密钥材料

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

用于导入密钥材料。只有类型为EXTERNAL 的CMK 才可以导入，导入的密钥材料使用 GetParametersForImport 获取的密钥进行加密。可以为指定的 CMK 重新导入密钥材料，并重新指定过期时间，但必须导入相同的密钥材料。CMK 密钥材料导入后不可以更换密钥材料。导入的密钥材料过期或者被删除后，指定的CMK将无法使用，需要再次导入相同的密钥材料才能正常使用。CMK是独立的，同样的密钥材料可导入不同的 CMK 中，但使用其中一个 CMK 加密的数据无法使用另一个 CMK解密。 只有Enabled 和 PendingImport状态的CMK可以导入密钥材料。

默认接口请求频率限制：10次/秒。

接口更新时间：2020-08-11 12:02:05。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ImportKeyMaterial |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| EncryptedKeyMaterial | 是 | 否 | String | 使用GetParametersForImport 返回的PublicKey加密后的密钥材料base64编码。对于国密版本region的KMS，导入的密钥材料长度要求为 128 bit，FIPS版本region的KMS， 导入的密钥材料长度要求为256 bit。 |
| ImportToken | 是 | 否 | String | 通过调用GetParametersForImport获得的导入令牌。 |
| ValidTo | 否 | 否 | Uint64 | 密钥材料过期时间 unix 时间戳，不指定或者 0 表示密钥材料不会过期，若指定过期时间，需要大于当前时间点，最大支持2147443200。 |
| KeyId | 是 | 否 | String | 指定导入密钥材料的CMK，需要和GetParametersForImport 指定的CMK相同。 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkNotFound | |
| InternalError | |
| UnsupportedOperation.NotExternalCmk | |
| ResourceUnavailable.CmkStateNotSupport | |
| ResourceUnavailable.TokenExpired | |
| InvalidParameter.DecryptMaterialError | |
| InvalidParameterValue.MaterialNotMatch | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |

# 列出当前Region支持的加密方式

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

列出当前Region支持的加密方式

默认接口请求频率限制：20次/秒。

接口更新时间：2020-03-11 17:03:23。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ListAlgorithms |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| SymmetricAlgorithms | AlgorithmInfo | 本地区支持的对称加密算法 |
| AsymmetricAlgorithms | AlgorithmInfo | 本地区支持的非对称加密算法 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InternalError | |
| UnauthorizedOperation | |

# 获取主密钥列表详情

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

根据指定Offset和Limit获取主密钥列表详情。

默认接口请求频率限制：100次/秒。

接口更新时间：2020-12-02 09:56:15。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ListKeyDetail |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| Offset | 否 | 否 | Uint64 | 含义跟 SQL 查询的 Offset 一致，表示本次获取从按一定顺序排列数组的第 Offset 个元素开始，缺省为0 |
| Limit | 否 | 否 | Uint64 | 含义跟 SQL 查询的 Limit 一致，表示本次最多获取 Limit 个元素。缺省值为10，最大值为200 |
| Role | 否 | 否 | Uint64 | 根据创建者角色筛选，默认 0 表示用户自己创建的cmk， 1 表示授权其它云产品自动创建的cmk |
| OrderType | 否 | 否 | Uint64 | 根据CMK创建时间排序， 0 表示按照降序排序，1表示按照升序排序 |
| KeyState | 否 | 否 | Uint64 | 根据CMK状态筛选， 0表示全部CMK， 1 表示仅查询Enabled CMK， 2 表示仅查询Disabled CMK，3 表示查询PendingDelete 状态的CMK(处于计划删除状态的Key)，4 表示查询 PendingImport 状态的CMK |
| SearchKeyAlias | 否 | 否 | String | 根据KeyId或者Alias进行模糊匹配查询 |
| Origin | 否 | 否 | String | 根据CMK类型筛选， "TENCENT_KMS" 表示筛选密钥材料由KMS创建的CMK， "EXTERNAL" 表示筛选密钥材料需要用户导入的 EXTERNAL类型CMK，"ALL" 或者不设置表示两种类型都查询，大小写敏感。 |
| KeyUsage | 否 | 否 | String | 根据CMK的KeyUsage筛选，ALL表示筛选全部，可使用的参数为：ALL 或 ENCRYPT_DECRYPT 或 ASYMMETRIC_DECRYPT_RSA_2048 或 ASYMMETRIC_DECRYPT_SM2，为空则默认筛选ENCRYPT_DECRYPT类型 |

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TagFilters | 否 | 否 | Array of TagFilter | 标签过滤条件 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| TotalCount | Uint64 | CMK的总数量 |
| KeyMetadatas | KeyMetadata | 返回的属性信息列表。 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| InternalError | |
| UnauthorizedOperation | |

# 获取主密钥列表

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

列出账号下面状态为Enabled， Disabled 和 PendingImport 的CMK KeyId 列表

默认接口请求频率限制：50次/秒。

接口更新时间：2020-01-15 20:31:15。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ListKeys |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| Offset | 否 | 否 | Uint64 | 含义跟 SQL 查询的 Offset 一致，表示本次获取从按一定顺序排列数组的第 Offset 个元素开始，缺省为0 |
| Limit | 否 | 否 | Uint64 | 含义跟 SQL 查询的 Limit 一致，表示本次获最多获取 Limit 个元素。缺省值为10，最大值为200 |
| Role | 否 | 否 | Uint64 | 根据创建者角色筛选，默认 0 表示用户自己创建的cmk， 1 表示授权其它云产品自动创建的cmk |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Keys | Key | CMK列表数组 |
| TotalCount | Uint64 | CMK的总数量 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| InternalError | |
| UnauthorizedOperation | |

# 密文刷新

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

使用指定CMK对密文重新加密。

默认接口请求频率限制：100次/秒。

接口更新时间：2020-01-15 20:20:45。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ReEncrypt |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| CiphertextBlob | 是 | 否 | String | 需要重新加密的密文 |
| DestinationKeyId | 否 | 否 | String | 重新加密使用的CMK，如果为空，则使用密文原有的CMK重新加密（若密钥没有轮换则密文不会刷新） |
| SourceEncryptionContext | 否 | 否 | String | CiphertextBlob 密文加密时使用的key/value对的json字符串。如果加密时未使用，则为空 |
| DestinationEncryptionContext | 否 | 否 | String | 重新加密使用的key/value对的json字符串，如果使用该字段，则返回的新密文在解密时需要填入相同的字符串 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| CiphertextBlob | String | 重新加密后的密文 |
| KeyId | String | 重新加密使用的CMK |
| SourceKeyId | String | 重新加密前密文使用的CMK |
| ReEncrypted | Bool | true表示密文已经重新加密。同一个CMK进行重加密，在密钥没有发生轮换的情况下不会进行实际重新加密操作，返回原密文 |

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkDisabled | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InvalidParameterValue.InvalidCiphertext | |
| InternalError | |
| UnauthorizedOperation | |

# CMK计划删除接口

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

CMK计划删除接口，用于指定CMK删除的时间，可选时间区间为[7,30]天

默认接口请求频率限制：20次/秒。

接口更新时间：2020-08-11 11:56:46。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：ScheduleKeyDeletion |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | CMK的唯一标志 |
| PendingWindowInDays | 是 | 否 | Uint64 | 计划删除时间区间[7,30] |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| DeletionDate | Uint64 | 计划删除执行时间 |
| KeyId | String | 唯一标志被计划删除的CMK |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| ResourceUnavailable.CmkNotFound | |

| 错误码 | 描述 |
| --- | --- |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |
| UnauthorizedOperation | |
| InvalidParameter.InvalidPendingWindowInDays | |
| ResourceUnavailable.CmkShouldBeDisabled | |
| ResourceUnavailable.CmkStateNotSupport | |
| FailedOperation.CmkUsedByCloudProduct | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |

版权所有：腾讯云金融专区

# 签名

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

非对称密钥签名。 注意：只有成功创建了KeyUsage= ASYMMETRIC_SIGN_VERIFY_SM2、ASYMMETRIC_SIGN_VERIFY_ECC 或 ASYMMETRIC_SIGN_VERIFY_RSA_2048 的密钥才可以使用签名功能

默认接口请求频率限制：100次/秒。

接口更新时间：2022-09-15 14:59:41。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：SignByAsymmetricKey |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| Algorithm | 是 | 否 | String | 签名算法，支持的算法：SM2DSA（ASN1 格式），SM2DSA_ASN1（ASN1 格式），SM2DSA_RAW（原始格式），ECC_P256_R1，RSA_PSS_SHA_256，RSA_PKCS1_SHA_256 |
| Message | 是 | 否 | String | 消息原文或消息摘要。如果提供的是消息原文，则消息原文的长度（Base64编码前的长度）不超过4096字节。如果提供的是消息摘要，消息摘要长度（Base64编码前的长度）必须等于32字节 |
| KeyId | 是 | 否 | String | 密钥的唯一标识 |
| MessageType | 否 | 否 | String | 消息类型：RAW，DIGEST，如果不传，默认为RAW，表示消息原文。 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Signature | String | 签名，Base64编码 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

# 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| ResourceUnavailable.CmkStateNotSupport | |
| InvalidParameterValue.InvalidKeyId | |
| ResourceUnavailable.CmkNotFound | |
| AuthFailure | |
| InternalError | |
| InvalidParameter | |

# 解绑CMK和云资源的关联关系

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

删除指定（key, 资源，云产品）的记录，以表明：指定的云产品的资源已不再使用当前的key。

默认接口请求频率限制：20次/秒。

接口更新时间：2020-08-11 11:52:08。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：UnbindCloudResource |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | cmk的ID |
| ProductId | 是 | 否 | String | 云产品的唯一性标识符 |
| ResourceId | 是 | 否 | String | 资源/实例ID，由调用方根据自己的云产品特征来定义，以字符串形式做存储。 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| UnsupportedOperation.ServiceTemporaryUnavailable | |
| InvalidParameter | |

| 错误码 | 描述 |
| --- | --- |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |
| UnauthorizedOperation | |
| ResourceUnavailable.CloudResourceBindingNotFound | |

# 修改别名

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

用于修改CMK的别名。对于处于PendingDelete状态的CMK禁止修改。

默认接口请求频率限制：100次/秒。

接口更新时间：2020-08-11 11:59:50。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：UpdateAlias |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| Alias | 是 | 否 | String | 新的别名，1-60个字符或数字的组合 |
| KeyId | 是 | 否 | String | CMK的全局唯一标识符 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| InvalidParameterValue.InvalidAlias | |
| InvalidParameterValue.AliasAlreadyExists | |

| 错误码 | 描述 |
| --- | --- |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |
| UnauthorizedOperation | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |

# 修改主密钥描述信息

最近更新时间: 2025-02-18 17:50:22

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

该接口用于对指定的cmk修改描述信息。对于处于PendingDelete状态的CMK禁止修改。

默认接口请求频率限制：100次/秒。

接口更新时间：2020-08-11 11:58:12。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：UpdateKeyDescription |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| Description | 是 | 否 | String | 新的描述信息，最大支持1024字节 |
| KeyId | 是 | 否 | String | 需要修改描述信息的CMK ID |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |

| 错误码 | 描述 |
| --- | --- |
| InternalError | |
| UnauthorizedOperation | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |

# 非对称密钥相关接口
# 非对称密钥RSA解密

最近更新时间: 2025-02-18 17:50:23

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

使用指定的RSA非对称密钥的私钥进行数据解密，密文必须是使用对应公钥加密的。处于Enabled 状态的非对称密钥才能进行解密操作。

默认接口请求频率限制：200次/秒。

接口更新时间：2020-03-06 10:07:37。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：AsymmetricRsaDecrypt |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | CMK的唯一标识 |
| Ciphertext | 是 | 否 | String | 使用PublicKey加密的密文，Base64编码 |
| Algorithm | 是 | 否 | String | 在使用公钥加密时对应的算法：当前支持RSAES_PKCS1_V1_5、RSAES_OAEP_SHA_1、RSAES_OAEP_SHA_256 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| KeyId | String | CMK的唯一标识 |
| Plaintext | String | 解密后的明文，base64编码 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| InvalidParameterValue.InvalidKeyUsage | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |
| UnauthorizedOperation | |
| ResourceUnavailable.CmkStateNotSupport | |
| FailedOperation.DecryptError | |

版权所有：腾讯云金融专区

# 非对称密钥Sm2解密

最近更新时间: 2025-02-18 17:50:23

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

使用指定的SM2非对称密钥的私钥进行数据解密，密文必须是使用对应公钥加密的。处于Enabled 状态的非对称密钥才能进行解密操作。传入的密文的长度不能超过256字节。

默认接口请求频率限制：200次/秒。

接口更新时间：2022-09-13 14:39:30。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：AsymmetricSm2Decrypt |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | CMK的唯一标识 |
| Ciphertext | 是 | 否 | String | 使用PublicKey加密的密文，Base64编码。密文长度不能超过 2048 字节。 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| KeyId | String | CMK的唯一标识 |
| Plaintext | String | 解密后的明文，base64编码 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| InvalidParameterValue.InvalidKeyUsage | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |
| UnauthorizedOperation | |
| ResourceUnavailable.CmkStateNotSupport | |
| FailedOperation.DecryptError | |
| UnsupportedOperation.UnsupportedKeyUsageInCurrentRegion | |

# 加密

最近更新时间: 2025-02-18 17:50:23

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

本接口用于加密最多为4KB任意数据，可用于加密数据库密码，RSA Key，或其它较小的敏感信息。对于应用的数据加密，使用GenerateDataKey生成的DataKey进行本地数据的加解密操作

默认接口请求频率限制：300次/秒。

接口更新时间：2022-09-24 16:38:44。

接口既验签名又鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：AsymmetricSm2Encrypt |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | 调用CreateKey生成的CMK全局唯一标识符 |
| Plaintext | 是 | 否 | String | 被加密的明文数据，该字段必须使用base64编码，原文最大长度支持 1024 字节 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| Ciphertext | String | 加密后经过base64编码的密文 |
| KeyId | String | 加密使用的CMK的全局唯一标识 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkDisabled | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InvalidParameterValue.InvalidPlaintext | |
| InternalError | |
| UnauthorizedOperation | |

# 获取非对称密钥的公钥

最近更新时间: 2025-02-18 17:50:23

## 1. 接口描述

接口请求域名： kms.api3.finance.cloud.tencent.com。

该接口用户获取 KeyUsage为ASYMMETRIC_DECRYPT_RSA_2048 和 ASYMMETRIC_DECRYPT_SM2 的非对称密钥的公钥信息，使用该公钥用户可在本地进行数据加密，使用该公钥加密的数据只能通过KMS使用对应的私钥进行解密。只有处于Enabled状态的非对称密钥才可能获取公钥。

默认接口请求频率限制：100次/秒。

接口更新时间：2020-03-06 09:36:43。

接口只验签名不鉴权。

## 2. 输入参数

以下请求参数列表仅列出了接口请求参数和部分公共参数，完整公共参数列表见公共请求参数。

| 参数名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Action | 是 | 否 | String | 公共参数，本接口取值：GetPublicKey |
| Version | 是 | 否 | String | 公共参数，本接口取值：2019-01-18 |
| Region | 是 | 否 | String | 公共参数，详见产品支持的**地域列表**(TODO) |
| KeyId | 是 | 否 | String | CMK的唯一标识。 |

## 3. 输出参数

| 参数名称 | 类型 | 描述 |
| --- | --- | --- |
| KeyId | String | CMK的唯一标识。 |
| PublicKey | String | 经过base64编码的公钥内容。 |
| PublicKeyPem | String | PEM格式的公钥内容。 |
| RequestId | String | 唯一请求 ID，每次请求都会返回。定位问题时需要提供该次请求的 RequestId。 |

## 4. 错误码

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见公共错误码。

| 错误码 | 描述 |
| --- | --- |
| InvalidParameter | |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.InvalidKeyId | |
| InternalError | |
| UnauthorizedOperation | |
| ResourceUnavailable.CmkStateNotSupport | |

# 数据结构

最近更新时间: 2025-02-18 17:50:23

## DeviceFingerprint

设备指纹

被如下接口引用：DescribeWhiteBoxDeviceFingerprints、OverwriteWhiteBoxDeviceFingerprints

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Identity | 是 | 否 | String | 指纹信息，由设备指纹采集工具采集获得，格式满足正则表达式：^[0-9a-f]{8}[-][0-9a-f]{14}[-][0-9a-f]{14}[-][0-9a-f]{14}[-][0-9a-f]{16}$ |
| Description | 否 | 是 | String | 描述信息，如：IP，设备名称等，最大1024字节 |

## Key

返回CMK列表信息

被如下接口引用：ListKeys

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| KeyId | 是 | 否 | String | CMK的全局唯一标识。 |

## WhiteboxKeyInfo

白盒密钥信息

被如下接口引用：DescribeWhiteBoxKey、DescribeWhiteBoxKeyDetails

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| KeyId | 是 | 否 | String | 白盒密钥的全局唯一标识符 |
| Alias | 是 | 否 | String | 作为密钥更容易辨识，更容易被人看懂的别名， 不可为空，1-60个字母数字 - _ 的组合，首字符必须为字母或者数字. 不可重复 |
| CreatorUin | 是 | 否 | Uint64 | 创建者 |
| Description | 是 | 否 | String | 密钥的描述信息 |
| CreateTime | 是 | 否 | Uint64 | 密钥创建时间，Unix时间戳 |
| Status | 是 | 否 | String | 白盒密钥的状态， 取值为：Enabled \| Disabled |
| OwnerUin | 是 | 否 | Uint64 | 创建者 |
| Algorithm | 是 | 否 | String | 密钥所用的算法类型 |

版权所有：腾讯云金融专区

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| EncryptKey | 是 | 否 | String | 白盒加密密钥，base64编码 |
| DecryptKey | 是 | 否 | String | 白盒解密密钥，base64编码 |
| ResourceId | 是 | 否 | String | 资源ID，格式：creatorUin/$creatorUin/$keyId |
| DeviceFingerprintBind | 是 | 是 | Bool | 是否有设备指纹与当前密钥绑定 |

# KeyMetadata

CMK属性信息

被如下接口引用：DescribeKey、DescribeKeys、ListKeyDetail

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| KeyId | 是 | 否 | String | CMK的全局唯一标识 |
| Alias | 是 | 否 | String | 作为密钥更容易辨识，更容易被人看懂的别名 |
| CreateTime | 是 | 否 | Uint64 | 密钥创建时间 |
| Description | 是 | 否 | String | CMK的描述 |
| KeyState | 是 | 否 | String | CMK的状态， 取值为：Enabled \| Disabled \| PendingDelete \| PendingImport |
| KeyUsage | 是 | 否 | String | CMK用途，取值为: ENCRYPT_DECRYPT \| ASYMMETRIC_DECRYPT_RSA_2048 \| ASYMMETRIC_DECRYPT_SM2 |
| Type | 是 | 否 | Int64 | CMK类型，2 表示符合FIPS标准，4表示符合国密标准 |
| CreatorUin | 是 | 否 | Uint64 | 创建者 |
| KeyRotationEnabled | 是 | 否 | Bool | 是否开启了密钥轮换功能 |
| Owner | 是 | 否 | String | CMK的创建者，用户创建的为 user，授权各云产品自动创建的为对应的产品名 |
| NextRotateTime | 是 | 否 | Uint64 | 在密钥轮换开启状态下，下次轮换的时间 |
| DeletionDate | 是 | 是 | Uint64 | 计划删除的时间 |
| Origin | 是 | 是 | String | CMK 密钥材料类型，由KMS创建的为： TENCENT_KMS， 由用户导入的类型为：EXTERNAL |
| ValidTo | 是 | 是 | Uint64 | 在Origin为 EXTERNAL 时有效，表示密钥材料的有效日期， 0 表示不过期 |
| ResourceId | 是 | 否 | String | 资源ID，格式：creatorUin/$creatorUin/$keyId |
| RotateDays | 否 | 是 | Uint64 | 轮换天数 |
| LastRotateTime | 否 | 是 | Uint64 | 上一次轮换的时间 |

版权所有：腾讯云金融专区

## TsmLicense

TSM SDK授权License

被如下接口引用：DescribeTSMLicenses

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Version | 否 | 否 | String | License版本 |
| TsmAppId | 否 | 否 | String | TSM AppId |
| Cert | 否 | 否 | String | 证书，这里返回的是字符串格式 |

## Tag

标签键和标签值

被如下接口引用：CreateKey、CreateWhiteBoxKey

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TagKey | 是 | 否 | String | 标签键 |
| TagValue | 是 | 否 | String | 标签值 |

## TagFilter

标签过滤器

被如下接口引用：DescribeWhiteBoxKeyDetails、ListKeyDetail

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| TagKey | 是 | 否 | String | 标签键 |
| TagValue | 否 | 否 | Array of String | 标签值 |

## VpcZoneData

vpc区域数据详情

被如下接口引用：

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| Zone | 是 | 否 | String | 可用区 |
| Region | 是 | 否 | String | vpc节点地域 |

版权所有：腾讯云金融专区

# AlgorithmInfo

算法的名称 和 标识

被如下接口引用：ListAlgorithms

| 名称 | 必选 | 允许NULL | 类型 | 描述 |
| --- | --- | --- | --- | --- |
| KeyUsage | 是 | 否 | String | 算法的标识 |
| Algorithm | 是 | 否 | String | 算法的名称 |

# 错误码

最近更新时间: 2025-02-18 17:50:23

## 功能说明

如果返回结果中存在 Error 字段，则表示调用 API 接口失败。例如：

```
{
"Response": {
"Error": {
"Code": "AuthFailure.SignatureFailure",
"Message": "The provided credentials could not be validated. Please check your signature is correct."
},
"RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
}
}
```

Error 中的 Code 表示错误码，Message 表示该错误的具体信息。

## 错误码列表

### 公共错误码

| 错误码 | 说明 |
| --- | --- |
| AuthFailure.InvalidSecretId | 密钥非法（不是云 API 密钥类型）。 |
| AuthFailure.MFAFailure | MFA 错误。 |
| AuthFailure.SecretIdNotFound | 密钥不存在。 请在控制台检查密钥是否已被删除或者禁用，如状态正常，请检查密钥是否填写正确，注意前后不得有空格。 |
| AuthFailure.SignatureExpire | 签名过期。Timestamp 和服务器时间相差不得超过五分钟，请检查本地时间是否和标准时间同步。 |
| AuthFailure.SignatureFailure | 签名错误。 签名计算错误，请对照调用方式中的接口鉴权文档检查签名计算过程。 |
| AuthFailure.TokenFailure | token 错误。 |
| AuthFailure.UnauthorizedOperation | 请求未 CAM 授权。 |
| DryRunOperation | DryRun 操作，代表请求将会是成功的，只是多传了 DryRun 参数。 |
| FailedOperation | 操作失败。 |
| InternalError | 内部错误。 |
| InvalidAction | 接口不存在。 |
| InvalidParameter | 参数错误。 |
| InvalidParameterValue | 参数取值错误。 |

版权所有：腾讯云金融专区

| 错误码 | 说明 |
| --- | --- |
| LimitExceeded | 超过配额限制。 |
| MissingParameter | 缺少参数错误。 |
| NoSuchVersion | 接口版本不存在。 |
| RequestLimitExceeded | 请求的次数超过了频率限制。 |
| ResourceInUse | 资源被占用。 |
| ResourceInsufficient | 资源不足。 |
| ResourceNotFound | 资源不存在。 |
| ResourceUnavailable | 资源不可用。 |
| UnauthorizedOperation | 未授权操作。 |
| UnknownParameter | 未知参数错误。 |
| UnsupportedOperation | 操作不支持。 |
| UnsupportedProtocol | http(s)请求协议错误，只支持 GET 和 POST 请求。 |
| UnsupportedRegion | 接口不支持所传地域。 |

**业务错误码**

| 错误码 | 说明 |
| --- | --- |
| LimitExceeded.FingerprintsLimitExceeded | |
| ResourceUnavailable.CloudResourceBindingNotFound | |
| InvalidParameterValue.InvalidPlaintext | |
| UnsupportedOperation.NotUserCreatedCmk | |
| UnsupportedOperation.ExternalCmkCanNotRotate | |
| InvalidParameterValue.InvalidKeyUsage | |
| FailedOperation.DecryptError | |
| ResourceUnavailable.CmkDisabled | |
| InvalidParameterValue.AliasAlreadyExists | |
| InvalidParameterValue.TagKeysDuplicated | |
| InvalidParameterValue.DuplicatedKeyId | |
| UnsupportedOperation.UnsupportedKeyUsageInCurrentRegion | |
| ResourceUnavailable.CmkShouldBeDisabled | |

版权所有：腾讯云金融专区

| 错误码 | 说明 |
| --- | --- |
| ResourceUnavailable.CmkNotFound | |
| InvalidParameterValue.MaterialNotMatch | |
| InvalidParameterValue.InvalidAlias | |
| LimitExceeded.CmkLimitExceeded | |
| ResourceUnavailable.CmkNotPendingDelete | |
| FailedOperation.CmkUsedByCloudProduct | |
| InvalidParameterValue.InvalidCiphertext | |
| InvalidParameter.DecryptMaterialError | |
| InvalidParameter | |
| InvalidParameterValue | |
| ResourceUnavailable.NotPurchased | |
| ResourceUnavailable.TokenExpired | |
| InvalidParameterValue.TagsNotExisted | |
| AuthFailure | |
| LimitExceeded.KeyLimitExceeded | |
| InternalError | |
| FailedOperation | |
| UnauthorizedOperation | |
| ResourceUnavailable.CmkStateNotSupport | |
| InvalidParameterValue.InvalidType | |
| UnsupportedOperation.NotExternalCmk | |
| UnsupportedOperation.ServiceTemporaryUnavailable | |
| InvalidParameterValue.InvalidKeyId | |
| ResourceNotFound | |
| ResourceUnavailable.KeyDisabled | |
| InvalidParameter.InvalidPendingWindowInDays | |

版权所有：腾讯云金融专区