阿里云
密钥管理服务

用户指南

文档版本：20191108

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. **您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。**
2. **未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。**
3. **由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。**
4. **本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。**
5. **阿里云文档中所有内容，包括但不限于图片、架构设计、页面布局、文字描述，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。**
6. **如若发现本文档存在任何错误，请与阿里云取得直接联系。**

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告:** 重启操作将导致业务中断，恢复业务时间约十分钟。
	用于警示信息、补充说明等，是用户必须了解的内容。	 **注意:** 权重设置为0，该服务器不会再接受新请求。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明:** 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	单击设置 > 网络 > 设置网络类型。
粗体	表示按键、菜单、页面名称等UI元素。	在结果确认页面，单击**确定**。
`Courier字体`	命令。	执行`cd /d C:/window`命令，进入Windows系统文件夹。
*## *	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{}或者{a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1 使用RAM实现访问控制

密钥管理服务（KMS）通过访问控制（RAM）实现对资源的访问控制。本文为您介绍KMS定义的资源类型、操作和策略条件。

云帐号对自己的资源拥有完整的操作权限，RAM用户和角色则需要通过显示授权获取对应资源的操作权限。在了解如何使用RAM授权和访问主密钥之前，请确保您已详细阅读了*#unique_4*和*#unique_5*。

KMS定义的资源类型

下表列出了KMS定义的所有资源类型以及对应的资源名称（ARN），用于RAM权限策略的`Resource`元素。

资源类型	ARN
抽象密钥容器	acs:kms:${region}:${account}:key
抽象别名容器	acs:kms:${region}:${account}:alias
密钥	acs:kms:${region}:${account}:key/${key-id}
别名	acs:kms:${region}:${account}:alias/${alias-name}

KMS定义的操作

针对每一个需要进行访问控制的接口，KMS都定义了用于RAM权限策略的操作（Action），通常为`kms:${api-name}`。

> **说明:**
> `DescribeRegions`不需要进行访问控制，只要请求可以通过认证校验，便可以调用。调用者可以是云帐号、RAM用户或RAM角色。

下表列出了KMS接口的对应RAM Action，以及接口所访问的资源类型。

KMS接口	Action	资源类型
`ListKeys`	kms:ListKeys	抽象密钥容器
`CreateKey`	kms:CreateKey	抽象密钥容器
`DescribeKey`	kms:DescribeKey	密钥

KMS接口	Action	资源类型
UpdateKeyDescription	**kms:UpdateKeyDescription**	**密钥**
EnableKey	**kms:EnableKey**	**密钥**
DisableKey	**kms:DisableKey**	**密钥**
ScheduleKeyDeletion	**kms:ScheduleKeyDeletion**	**密钥**
CancelKeyDeletion	**kms:CancelKeyDeletion**	**密钥**
GetParametersForImport	**kms:GetParametersForImport**	**密钥**
ImportKeyMaterial	**kms:ImportKeyMaterial**	**密钥**
DeleteKeyMaterial	**kms:DeleteKeyMaterial**	**密钥**
Encrypt	**kms:Encrypt**	**密钥**
GenerateDataKey	**kms:GenerateDataKey**	**密钥**
GenerateDataKeyWithoutPlaintext	**kms:GenerateDataKeyWithoutPlaintext**	**密钥**
Decrypt	**kms:Decrypt**	**密钥**
ListAliases	**kms:ListAliases**	**抽象别名容器**
CreateAlias	**kms:CreateAlias**	**别名，密钥**
UpdateAlias	**kms:UpdateAlias**	**别名，密钥**
DeleteAlias	**kms:DeleteAlias**	**别名，密钥**
ListAliasesByKeyId	**kms:ListAliasesByKeyId**	**密钥**
TagResource	**kms:TagResource**	**密钥**
UntagResource	**kms:UntagResource**	**密钥**
ListResourceTags	**kms:ListResourceTags**	**密钥**
DescribeKeyVersion	**kms:DescribeKeyVersion**	**密钥**
ListKeyVersions	**kms:ListKeyVersions**	**密钥**
UpdateRotationPolicy	**kms:UpdateRotationPolicy**	**密钥**

KMS支持的策略条件

您可以在RAM权限策略中设定条件控制对KMS的访问，只有当条件满足时，权限验证才能通过。例如，您可以使用`acs:CurrentTime`条件限制权限策略有效的时间。

除了阿里云全局条件，您也可以使用标签作为条件关键字，限制对Encrypt、Decrypt、GenerateDataKey等密码运算API的使用。条件关键字的格式为`kms:tag/${tag-key}`。

详情请参见*#unique_6*。

常见的授权策略示例

- **允许访问所有的KMS资源**

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

- **只读访问权限，即列出、查看密钥、查看别名与使用密钥权限**

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:List*", "kms:Describe*",
        "kms:Encrypt", "kms:Decrypt", "kms:GenerateDataKey"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

- **允许使用含有下列标签的密钥进行密码运算：**
  - **标签键：**`Project`
  - **标签值：**`Apollo`

```
{
    "Version": "1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt", "kms:Decrypt", "kms:GenerateDataKey"
            ],
            "Resource": [
```

```
                "*"
            ],
            "Condition": {
                "StringEqualsIgnoreCase": {
                    "kms:tag/Project": [
                        "Apollo"
                    ]
                }
            }
        }
    ]
}
```

2 使用ActionTrail记录操作事件

本文为您介绍如何使用ActionTrail记录密钥管理服务（KMS）操作事件。

KMS已经与操作审计（*ActionTrail*）服务进行了集成。您可以在ActionTrail中查看所有用户（主账号/RAM用户）对您的资源实例所进行的操作记录。

ActionTrail记录的KMS信息包括除DescribeRegions的所有API。详情请参见*#unique_8*。

关于操作记录的详细信息，请参见*#unique_9*。

3 别名使用说明

别名是用户主密钥的可选标识。

别名在一个阿里云账户、一个地域中具有唯一性。同一个账户在不同地域下可以拥有相同的别名。每个别名只能指向同地域的一个用户主密钥，但是每个用户主密钥可以拥有多个别名。

一个别名一定会指向一个用户主密钥，但是别名是独立于用户主密钥存在的一个资源。别名具有以下特点：

- 可以通过*#unique_11*更改别名关联的用户主密钥，而不会影响用户主密钥。
- 删除别名不会删除其关联的用户主密钥。
- 子账户操作别名，需要有别名相关资源的授权。详情参见使用*RAM*实现访问控制。
- 别名不可更改。可以通过为一个用户主密钥创建新的别名，并且删除旧的别名来达到修改别名的目的。

当一个别名与某一个用户主密钥相关联时，在以下操作中，可以将访问参数中的密钥ID用别名代替：

- DescribeKey
- Encrypt
- GenerateDataKey
- GenerateDataKeyWithoutPlaintext

当子账户使用别名代替密钥ID进行上述操作时，子账户必须拥有对应密钥的权限，无需拥有对应别名的权限。

您可以对别名进行以下操作：

- 创建别名
- 更新别名
- 删除别名
- 列出别名
- 列出指定密钥关联的别名

任何情况下使用别名时，都必须使用完整的别名，即别名前缀`alias/`与别名。

```
//包含前缀“alias/”的完整别名
```

```
alias/example
```

创建别名

- 别名必须拥有前缀alias/。除前缀以外，支持字母、数字、下划线（_）、连字符（-）以及斜杠（/）。除前缀以外，长度限制在1~255个字符。
- 当子账户访问创建别名时，需要同时拥有别名以及其关联密钥的权限。
- 为同一个用户主密钥创建新的别名不会影响已有别名。
- 创建别名的API是*#unique_17*。

```
//创建别名的RAM Policy示例：用户123456可以为密钥08ec3bb9-034f-485b-b1cd-
3459baa889c7创建别名alias/example
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateAlias"
      ],
      "Resource": [
        "acs:kms:cn-hangzhou:123456:key/08ec3bb9-034f-485b-b1cd-
3459baa889c7",
        "acs:kms:cn-hangzhou:123456:alias/example"
      ]
    }
  ]
}
//创建别名
aliyuncli kms CreateAlias --KeyId 08ec3bb9-034f-485b-b1cd-3459baa889c7
 --AliasName alias/example
```

更新别名

- 更新别名可以将已有的别名关联到其他用户主密钥，对应的API为*#unique_11*。
- 当子账户访问更新别名时，需要同时拥有原密钥、目标密钥以及别名的权限。

```
//更新别名的RAM Policy示例：用户123456可以将密钥08ec3bb9-034f-485b-b1cd-
3459baa889c7关联的别名alias/example管理到新密钥127d2f84-ee5f-4f4d-9d41-
dbc1aca28788
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:UpdateAlias"
      ],
      "Resource": [
        "acs:kms:cn-hangzhou:123456:key/08ec3bb9-034f-485b-b1cd-
3459baa889c7",
        "acs:kms:cn-hangzhou:123456:key/127d2f84-ee5f-4f4d-9d41-
dbc1aca28788",
        "acs:kms:cn-hangzhou:123456:alias/example"
      ]
    }
```

```
  ]
}
//更新别名
aliyuncli kms UpdateAlias --AliasName alias/example --KeyId 127d2f84-
ee5f-4f4d-9d41-dbc1aca28788
```

删除别名

- **删除别名不会影响其关联的密钥，对应的API为**#unique_18。
- **当子账户访问删除别名时，需要同时拥有别名以及其关联密钥的权限。**

```
//删除别名的RAM Policy示例：用户123456可以删除别名alias/example，其关联的密钥
是127d2f84-ee5f-4f4d-9d41-dbc1aca28788
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:DeleteAlias"
      ],
      "Resource": [
        "acs:kms:cn-hangzhou:123456:key/127d2f84-ee5f-4f4d-9d41-
dbc1aca28788",
        "acs:kms:cn-hangzhou:123456:alias/example"
      ]
    }
  ]
}
//删除别名
aliyuncli kms DeleteAlias --AliasName alias/example
```

列出别名

- **列出别名可以获取所有的别名信息，对应的API为**#unique_19。
- **当子账户访问列出别名时，需要拥有别名资源类型的权限。**

```
//列出别名的RAM Policy示例
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:ListeAliases"
      ],
      "Resource": [
        "acs:kms:cn-hangzhou:123456:alias"
      ]
    }
  ]
}
//列出别名
aliyuncli kms ListAliases
```

列出指定密钥关联的别名

- **列出指定密钥关联的别名只会返回与指定密钥相关联的别名信息，对应的API为**#unique_20。

- 当子账户访问列出指定密钥关联的别名时，只需拥有指定密钥的权限即可访问。

```
//出指定密钥关联的别名的RAM Policy示例：列出密钥127d2f84-ee5f-4f4d-9d41-
dbc1aca28788关联的所有别名
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:DeleteAlias"
      ],
      "Resource": [
        "acs:kms:cn-hangzhou:123456:key/127d2f84-ee5f-4f4d-9d41-
dbc1aca28788"
      ]
    }
  ]
}
//列出指定密钥关联的别名
aliyuncli kms ListAliasesByKeyId --KeyId 127d2f84-ee5f-4f4d-9d41-
dbc1aca28788
```

4 EncryptionContext说明

Encryption Context 是在 KMS 的 Encrypt、GenerateDataKey、Decrypt 这些 API 中可能会用到的 JSON 字符串。

Encryption Context的作用

Encryption Context 只能是 String-String 形式的 JSON，用于保护数据的完整性。

当在加密（Encrypt、GenerateDataKey）时指定了该参数时，解密（Decrypt）密文时，需要传入等价的*Encryption Context*的参数，才能正确的解密。Encryption Context 虽然与解密相关，但是并不会存在密文（CipherBlob）中。

Encryption Context有效值

Encryption Context 的有效值是一个总长度在 8192 个字符数以内的 JSON 字符串，并且只能是 String-String 形式的。 当您直接调用 API 填 Encryption Context 的时候，请注意转义的问题。

有效的Encryption Context示例

```
{"ValidKey":"ValidValue"}
{"Key1":"Value1","Key2":"Value2"}
```

无效的Encryption Context (部分)示例

```
[{"Key":"Value"}] //json数组
{"Key":12345} //String-int
{"Key":["value1","value2"]} //String-数组
```

等价的Encryption Context

Encryption Context的本质是一个 String-String的map（hashtable）, 因此在作为参数时，只需要保证 JSON 字符串所表示的key-value含义是一致的，则 Encryption Context 是等价的。与加密时输入的 Encryption Context 等价的 Encryption Context 就可以用于正确的解密，而不用保持完全一致的字符串。

等价的Encryption Context示例

```
{"Key1":"Value1","Key2":"Value2"} 与 {"Key2":"Value2","Key1":"Value1
"} 等价
```

5 导入密钥材料

本文为您介绍如何导入外部密钥材料以及删除已导入的外部密钥材料。

前提条件

进行操作前，请确保您已经注册了阿里云账号。如还未注册，请先完成账号注册。

背景信息

用户主密钥（CMK）是KMS的基本资源，是由密钥ID、基本元数据（如密钥状态等）以及用于加密、解密数据的密钥材料组成。默认情况下，当您*#unique_23*时，会由KMS生成密钥材料。 您可以选择创建密钥材料来源为外部的密钥，此时，KMS将不会为您创建的用户主密钥生成密钥材料，您可以将自己的密钥材料导入到CMK中。 您可以通过*#unique_24*来判断密钥材料来源情况。当KeyMetadata中`Origin`为`Aliyun_KMS`时，说明密钥材料由KMS生成，称为普通密钥；当`Origin`为`EXTERNAL`时，说明密钥材料由外部导入，称为外部密钥。

在您选择密钥材料来源为外部，使用您自己导入的密钥材料的时候，您需要注意以下几点：

- 您需要保证使用了符合要求的随机源生成密钥材料。
- 您需要保证密钥材料的可靠性。
  - KMS确保导入密钥材料高可用，但是不能保证导入密钥材料与KMS生成的密钥材料具有相同的可靠性。
  - 您导入的密钥材料可以直接通过*#unique_25*进行删除，也可以设置过期时间，在密钥材料过期后也会被删除（CMK不会被删除）。KMS生成的密钥材料无法直接被删除，只能通过*#unique_26*，在等待7到30天后，随着CMK一起被删除。
  - 当导入的密钥材料被删除后，可以再次导入相同的密钥材料使得CMK再次可用，因此您需要自行保存密钥材料的副本。
- 每个CMK只能拥有一个导入密钥材料。一旦您将一个密钥材料导入CMK，该CMK将与该密钥材料绑定，即便密钥材料已经过期或者被删除，也不能导入其他密钥材料。如果您需要轮换使用外部密钥材料的CMK，只能创建一个新的CMK然后导入新的密钥材料。
- CMK具有独立性。您使用一个CMK加密的数据，无法使用其他CMK进行解密，即便这些CMK都使用相同的密钥材料。
- 只能导入256位对称密钥作为密钥材料。

导入密钥材料

1. 创建外部密钥。

   - 在控制台创建密钥的高级选项中，选择密钥材料来源为外部。
   - 调用CreateKey，指定参数Origin为EXTERNAL。

   ```
   aliyuncli kms CreateKey --Origin EXTERNAL --Description "External
   key"
   ```

2. 获取导入密钥材料参数。

   导入密钥材料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。

   - 通过控制台导入密钥材料的参数。
   - 调用*#unique_27*。

   ```
   aliyuncli kms GetParametersForImport --KeyId 1339cb7d-54d3-
   47e0-b595-c7d3dba8**** --WrappingAlgorithm RSAES_OAEP_SHA_1 --
   WrappingKeySpec RSA_2048
   ```

3. 导入密钥材料。

   导入密钥材料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。 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。

   a) 使用加密公钥对密钥材料进行加密。

      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返回的加密公钥是经过Base64编码的，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。

   b) 将加密后的密钥材料进行Base64编码。

   c) 将编码后的密钥材料与导入令牌一起，作为*#unique_28*的参数导入KMS。

   ```
   aliyuncli kms ImportKeyMaterial --KeyId 1339cb7d-54d3-47e0-b595-
   c7d3dba8**** --EncryptedKeyMaterial xxx --ImportToken xxxx
   ```

4. 删除密钥材料。

   当您导入了密钥材料以后，您就可以像使用普通密钥一样使用外部密钥了。与普通密钥不同的是，外部密钥的密钥材料可能会过期，也可以由您手工删除。当密钥材料过期或者被删除以

后，密钥将无法继续使用，由该密钥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。

如果密钥在密钥材料过期或者密钥材料删除时处于待删除（PendingDeletion）状态时，密钥状态不会改变，否则密钥状态会变为等待导入（PendingImport）。

- 通过控制台删除密钥。
- 调用*#unique_25*。

```
aliyuncli kms DeleteKeyMaterial --KeyId 1339cb7d-54d3-47e0-b595-
c7d3dba8****
```

使用OPENSSL加密密钥材料并上传

1. 创建一个外部密钥。
2. 创建一个密钥材料。

   密钥材料必须是256位的对称密钥，使用OPENSSL产生一个32字节的随机数进行演示。

```
openssl rand -out KeyMaterial.bin 32
```

3. 获取导入密钥材料参数。
4. 加密密钥材料。

   a) 将加密公钥进行Base64解码。

   b) 根据指定的加密算法（以RSAES_OAEP_SHA_1为例）加密密钥材料。

   c) 将加密后的密钥材料进行Base64编码，保存为文本文件。

```
openssl rand -out KeyMaterial.bin 32
openssl enc -d -base64 -A -in PublicKey_base64.txt -out PublicKey.
bin
openssl rsautl -encrypt -in KeyMaterial.bin -oaep -inkey PublicKey
.bin  -keyform DER  -pubin -out EncryptedKeyMaterial.bin
openssl enc -e -base64 -A -in EncryptedKeyMaterial.bin -out
EncryptedKeyMaterial_base64.txt
```

   d) 上传加密后的密钥材料与导入令牌。

使用JAVA SDK加密密钥材料并上传。

```
//使用最新KMS JAVA SDK
//KmsClient.java

import com.aliyuncs.kms.model.v20160120.*;
import com.aliyuncs.profile.DefaultProfile;

//KMS API封装
public class KmsClient {
        DefaultAcsClient client;
```

```
        public KmsClient( String region_id, String ak,
String secret) {
                DefaultProfile profile = DefaultProfile.
getProfile(region_id, ak, secret);
                this.client = new DefaultAcsClient(profile);
        }

        public CreateKeyResponse createKey() throws
Exception {
                CreateKeyRequest request = new CreateKeyR
equest();
                request.setOrigin("EXTERNAL"); //创建外部密钥
                return this.client.getAcsResponse(request);
        }
        //... 省略，其余API类似
}
//example.java
import com.aliyuncs.kms.model.v20160120.*;
import KmsClient
import java.security.KeyFactory;
import java.security.PublicKey;
import java.security.spec.MGF1ParameterSpec;
import javax.crypto.Cipher;
import javax.crypto.spec.OAEPParameterSpec;
import javax.crypto.spec.PSource.PSpecified;
import java.security.spec.X509EncodedKeySpec;
import java.util.Random;
import javax.xml.bind.DatatypeConverter;

public class CreateAndImportExample {
        public static void main(String[] args) {
                String regionId = "cn-hangzhou";
        String accessKeyId = "*** Provide your AccessKeyId
 ***";
        String accessKeySecret = "*** Provide your
AccessKeySecret ***";
        KmsClient kmsclient = new KmsClient(regionId,
accessKeyId,accessKeySecret);
        //Create External Key
        try {
                CreateKeyResponse keyResponse = kmsclient.
createKey();
                String keyId = keyResponse.KeyMetadata.
getKeyId();
                //产生一个32位随机数
                byte[] keyMaterial = new byte[32];
                new Random().nextBytes(keyMaterial);
                //获取导入密钥材料参数
                GetParametersForImportResponse paramRespo
nse = kmsclient.getParametersForImport(keyId,"RSAES_OAEP
_SHA_256");
                String importToekn = paramResponse.
getImportToken();
                String encryptPublicKey = paramResponse.
getPublicKey();
                //base64解码加密公钥
                byte[] publicKeyDer = DatatypeConverter.
parseBase64Binary(encryptPublicKey);
                //解析成RSA的公钥
                KeyFactory keyFact = KeyFactory.getInstance
("RSA");
                X509EncodedKeySpec spec = new X509Encode
dKeySpec(publicKeyDer);
```

```
                PublicKey publicKey = keyFact.generatePublic
(spec);
                //加密密钥材料
                Cipher oaepFromAlgo = Cipher.getInstance("
RSA/ECB/OAEPWithSHA-1AndMGF1Padding");
                String hashFunc = "SHA-256";
                OAEPParameterSpec oaepParams = new
OAEPParameterSpec(hashFunc, "MGF1", new MGF1ParameterSpec(
hashFunc), PSpecified.DEFAULT);
                oaepFromAlgo.init(Cipher.ENCRYPT_MODE,
publicKey, oaepParams);
                byte[] cipherDer = oaepFromAlgo.doFinal(
keyMaterial);
                //加密后的密钥材料，需要进行base64编码
                String encryptedKeyMaterial = DatatypeCo
nverter.printBase64Binary(cipherDer);
                //导入密钥材料
                Long expireTimestamp = 1546272000L; //Unix时
间戳，精确到秒，0表示永不过期
                        kmsClient.importKeyMaterial(keyId,
encryptedKeyMaterial, expireTimestamp);
        } catch(Exception e) {
                //... 省略
        }
        }
}
```

6 托管密码机（公测）

6.1 托管密码机简介

托管密码机是密钥管理服务提供的一项重要功能，帮您在阿里云上轻松使用具有合规资质的硬件密码机。

硬件密码机是一种执行密码运算、安全生成和存储密钥的硬件设备。通过将密钥托管在这些高安全等级的硬件设备中，您可以保护您在阿里云上最敏感的计算任务和资产。

> **说明:**
> **硬件密码机也叫硬件安全模块（Hardware Security Module），通常缩写为HSM。**

支持的地域

您可以在下列地域使用托管密码机，同时我们计划在更多的地域逐步推出这一重要功能。

地域名称	所在城市	地域标识符
中国香港	香港	cn-hongkong
亚太东南1	新加坡	ap-southeast-1

合规

托管密码机能帮您满足严格的合规要求。根据各地区监管机构要求，阿里云提供的多种密码机分别由不同的三方机构认证，从而适应不同市场的地区性差异，满足您的本地化和国际化需求。

对中国大陆之外的地域：

- **硬件的FIPS认证：阿里云运营的密码机，包含它们的硬件和固件，获得了FIPS 140-2第三级认证。链接*Certificate #3254***
- **FIPS 140-2 第三级合规：阿里云的托管密码机运行在FIPS许可的第三级模式下。**
- **PCI-DSS合规：阿里云的托管密码机符合PCI DSS合规的要求。**

高安全保证

- **硬件保护**

  托管密码机可以帮助您通过安全的硬件机制来保护KMS中的密钥。用户主密钥的明文密钥材料只会在密码机的内部被处理，用于密码运算，而永远不会离开密码机硬件的安全边界。

- 安全的密钥生成

  随机性是密钥强度的关键。借助使用托管密码机，密钥材料的产生基于安全、许可、且以高系统熵值为种子的随机数生成算法，从而保护密钥不被攻击者恢复或者预判。

易运维

阿里云提供密码机硬件的全托管，您完全免去了自己管理硬件带来的如下运维开销。

- 硬件生命周期的管理
- 密码机集群管理
- 高可用和可伸缩性管理
- 系统修补（Patching）
- 大部分灾备工作

易集成

通过原生的密钥管理能力，您也可以从以下丰富的功能中获得极大的方便。

- 密钥版本管理
- 自动密钥轮转
- 资源标签管理
- 可控制的授权机制

这些功能赋能您的应用和密码机快速集成，同样也赋能云服务器ECS、关系型数据库RDS等其他云服务和托管密码机集成，实现云上数据静态加密，而您无需付出研发成本。

保持对密钥的控制

借助托管密码机，您能更好的控制云上的加密密钥，赋能您将最具敏感性的计算任务和资产移动到云端。

同时使用托管密码机和*BYOK（Bring Your Own Key）*，您能够在以下方面达成更强的控制力。

- 您完全控制密钥材料的生成方式
- 您导入到托管HSM的密钥材料只能被销毁而无法被导出
- 您完全控制密钥的生命周期
- 您完全控制密钥的持久性

低成本

您也能从云计算“用多少花多少”的计费模式中，享受极大的低成本好处。相比于通过本地密码机自建密钥基础设施，您免去了硬件采购的初始成本，同时也免去了后续研发和运维带来的持续性投入。

6.2 使用托管密码机

本文为您介绍如何使用托管密码机创建并使用密钥。

前提条件

进行操作前，请确保您已经注册了阿里云账号。如还未注册，请先完成账号注册。

背景信息

- **您需要在当前已经支持的地域使用托管密码机。支持的地域详情，请参见支持的地域。**
- **您可以联系您的售前或者售后顾问，开通免费试用托管密码机的功能。**

通过KMS控制台创建密钥

1. **登录*KMS*控制台。**
2. **在控制台左上角选择合适的地域，单击创建密钥。**
3. **在下拉框中，选择HSM作为保护级别。**
4. **填写描述后，单击确定即可。**

   创建完成后，在密钥详情和密钥列表页均可以查看到密钥的保护级别。

通过阿里云CLI创建密钥

1. **在阿里云CLI中，输入以下命令。**

```
aliyun kms CreateKey --ProtectionLevel HSM --Description "Key1 in
Managed HSM"
```

2. **调用DescribeKey，查看密钥的保护级别。**

```
{
  "KeyMetadata": {
    "CreationDate": "2019-07-04T13:14:15Z",
    "Description": "Key1 in Managed HSM",
    "KeyId": "1234abcd-12ab-34cd-56ef-12345678****",
    "KeyState": "Enabled",
    "KeyUsage": "ENCRYPT/DECRYPT",
    "DeleteDate": "",
    "Creator": "111122223333",
    "Arn": "acs:kms:cn-hongkong:111122223333:key/1234abcd-12ab-34cd-
56ef-12345678****",
    "Origin": "Aliyun_KMS",
    "MaterialExpireTime": "",
    "ProtectionLevel": "HSM"
  },
  "RequestId": "8eaeaa8b-4491-4f1e-a51e-f95a4e54620c"
```

```
}
```

将外部密钥导入托管密码机

您还可以将来自于您的自建密钥基础设施中的密钥导入到托管密码机中，您只需要在导入密钥材料的流程中第一步，即创建外部密钥时，指定保护级别为HSM，剩余的流程对您来说保持不变。

而在阿里云侧会发生如下的事情：

- **您调用`GetParametersForImport`时：阿里云将根据您指定的保护级别，HSM，在托管密码机中生成一个用于导入外部密钥的密钥对，并把密钥对的公钥返回给您**
- **您调用`ImportKeyMaterial`时：阿里云将加密的外部密钥材料导入到托管密码机的内部，并通过HSM的密钥反打包（Unwrap）机制获取密钥材料本身，而导入的密钥材料明文不能被任何人导出。**

管理和使用密钥

密钥管理服务支持的所有管理类功能和密码运算功能都适用于您在托管密码机中创建的密钥，主要包括：

- **密钥状态的开启和禁用**
- **密钥的生命周期管理**
- **密钥的别名管理**
- **密钥的云标签管理**
- **密码运算接口的调用**

和其他云产品的集成

托管密码机中的密钥，可以通过密钥管理服务的标准接口和ECS、RDS、OSS等其它云产品实现无缝集成，用于您在阿里云上的原生数据保护。这需要云产品支持用户自选密钥进行服务端加密的能力。在此基础上，您只需要在云产品侧配置用于服务端加密的CMK时，选择一个创建在托管密码机中的密钥。

7 密钥的轮转

7.1 密钥轮转概述

密钥常用于保护特定的数据，因此，数据的安全依赖于密钥的安全。您可以通过密钥版本化和定期轮转来加强密钥使用的安全性，实现数据保护的安全策略和最佳实践。

实现安全目标

您可以通过周期性轮转密钥，达成以下安全目标：

- **减少每个密钥加密的数据量**

  一个密钥的安全性与被它加密的数据量呈反相关。数据量通常是指同一个密钥加密的数据总字节数或总消息数。例如，NIST将GCM模式下一个密钥的安全生命周期定义为基于其加密的总消息数。通过定期轮转密钥而改变加密密钥的方式，可以使得每个密钥具有更高的安全阈值和更小的密码分析攻击面。

- **提前具备响应安全事件的能力**

  在系统设计的早期，引入密钥轮转的功能并将其作为日常运维手段。这样可以使系统在特定安全事件发生时具备实际执行能力，符合软件工程中Fail Early、Fail Often的原则。如果第一次执行（突发性）密钥轮转是在响应具体事件的情形下，并且发生在运行中的系统上，则发生故障的概率会被无限放大。

- **对数据形成逻辑上的隔离**

  轮转加密密钥使得轮转前后产生的密文数据形成事实上的隔离效果。特定密钥的安全事件可以被快速定义影响范围，从而采取进一步措施。

- **减小破解密钥的时间窗口**

  如果在定期轮转加密密钥的基础上，将旧的加密密钥产生的密文数据用新的加密密钥轮转加密，则轮转周期即为一个密钥的破解时间窗口。这意味着恶意者只有在两次轮转事件之间完成破解，才能拿到数据。这对于保护数据不受密码分析（Cryptanalysis）攻击具有很强的实践意义。

满足合规规范

密钥的周期性轮转功能可以方便企业符合各种合规规范。与密钥轮转相关的合规规范包含（但不限于）：

- **支付卡行业数据安全标准（PCI DSS）**
- **中国国家密码管理局发布的密码行业相关标准，例如GM/T 0051-2016**

- 美国国家标准技术委员会（NIST）发布的密码使用相关标准，例如NIST Publication 800-38D

7.2 自动轮转密钥

本文为您介绍如何对密钥管理服务（KMS）中的用户主密钥（CMK）进行自动轮转。

密钥版本

KMS中的CMK支持多个密钥版本。每一个密钥版本是一个独立生成的密钥，同一个CMK下的多个密钥版本在密码学上互不相关。KMS通过生成一个新的密钥版本来实现密钥的自动轮转。

CMK的密钥版本分为以下2种：

- 主版本（Primary Key Version）
  - 主版本是CMK的活跃加密密钥（Active Encryption Key）。每个CMK在任何时间点上有且仅有1个主版本。
  - 调用GenerateDataKey、Encrypt等加密API时，KMS使用指定CMK的主版本对明文进行加密。
  - 可以通过DescribeKey返回的响应，查看PrimaryKeyVersion属性。
- 非主版本（Non-primary Key Version）
  - 非主版本是CMK的非活跃加密密钥（Inactive Encryption Key）。每个CMK可以有0到多个非主版本。
  - 非主版本历史上曾经是主版本，当时被用作活跃加密密钥。
  - 密钥轮转产生新的主版本后，KMS不会删除或禁用非主版本，它们需要被用作解密数据。

说明：

加密API使用指定CMK的主版本，解密API使用传入密文的对应加密密钥版本。

您可以通过以下两种方式生成密钥版本：

- 创建CMK

  通过调用CreateKey创建CMK。如果指定参数Origin为Aliyun_KMS，则KMS会生成初始密钥版本，并且将其设置为主版本。

- KMS执行自动轮转策略

  CMK配置自动轮转策略，配置的策略由KMS定期执行，周期性产生新的密钥版本。

自动轮转

配置轮转策略

当您调用CreateKey创建CMK时，可以指定CMK的自动轮转策略，也可以调用UpdateRotationPolicy变更当前的自动轮转策略。调用时，需配置以下参数：

- EnableAutomaticRotation：是否开启自动轮转
- RotationInterval：自动轮转的周期

您可以通过调用DescribeKey查询所配置的轮转策略，返回的相关参数为：

- AutomaticRotation：自动轮转的开启状态。Disabled表示用户未开启自动轮转；Enabled表示用户已开启自动轮转；Suspended表示用户开启了自动轮转，但是KMS暂停执行。详情请参见主密钥状态对轮转的影响。
- RotationInterval：自动轮转的周期

KMS执行策略

如果您指定开启自动轮转，则KMS会按照以下方式计算下次轮转时间。

```
${NextRotationTime} = ${LastRotationTime} + ${RotationInterval}
```

其中：

- LastRotationTime：生成上一个密钥版本的时间。您可以通过DescribeKey返回的LastRotationDate字段查询此参数值。
- NextRotationTime：KMS会在计算出来的下一次轮转时间到来之后，执行轮转任务，创建新的密钥版本。您可以通过DescribeKey返回的NextRotationDate字段查询此参数值。

注意：

更新轮转策略所指定的轮转周期，可能会导致KMS计算出来的下次轮转时间成为过去的某个时间点。但这并不影响KMS执行轮转策略的规则，即在下一次轮转时间到来之后生成新的密钥版本。如果更新后的策略指向的下一次轮转时间已经在当前时间之前，则立即满足了KMS触发密钥轮转的标准。

主密钥状态对轮转的影响

只有在启用状态的主密钥（KeyState为Enabled）才能产生新的密钥版本。您需要注意以下情形：

- 如果一个主密钥处于Disabled或者PendingDeletion状态，请勿调用UpdateRotationPolicy变更主密钥的轮转策略。
- 如果一个主密钥在用户开启自动轮转后，进入到Disabled或者PendingDeletion状态，则KMS暂停执行自动轮转。此时通过DescribeKey查看到的轮转状态（AutomaticR

otation属性）为Suspended。当主密钥重新回到启用状态时，之前配置的轮转策略会重新生效。

不适用范围

KMS管理的以下类型的密钥不支持多个版本：

- 云产品托管密钥：特定云产品托管在KMS上的、用于加密保护您的数据的默认密钥。这类密钥由特定云产品为用户代为管理，为您的数据提供最基本的加密保护。
- 用户自带密钥：您导入到KMS中的密钥（详情请参见导入密钥材料）。这类CMK的Origin属性为External，KMS不负责为用户生成密钥材料，无法自动发起轮转行为。

因此，以上2种类型的密钥不支持以自动或人工的方式进行基于版本的密钥轮转。此外，KMS也不支持自带密钥的多版本功能。一方面，自带密钥的持久性和生命周期由用户强管控，本身就具有较高的管理难度和易错风险（例如您需要有云下的密钥管理设施，云上云下信息需要同步，云上删除密钥材料没有任何缓冲期），而多版本带来的复杂度升级会超线性地升高易错性，从而带来数据风险；另一方面，每个密钥版本，包括用于加密和解密的主版本以及仅用于解密的非主版本，都可能在不同的时间点上不可用（例如在不同时间点上过期而被KMS删除，或者过期后被重新分别导入），导致无法同步主密钥和被保护数据的可用性，很难保证系统设计的完整性。

说明:

对于上述不支持基于版本进行密钥轮转的情景，替代方案请参见人工轮转主密钥。

7.3 人工轮转主密钥

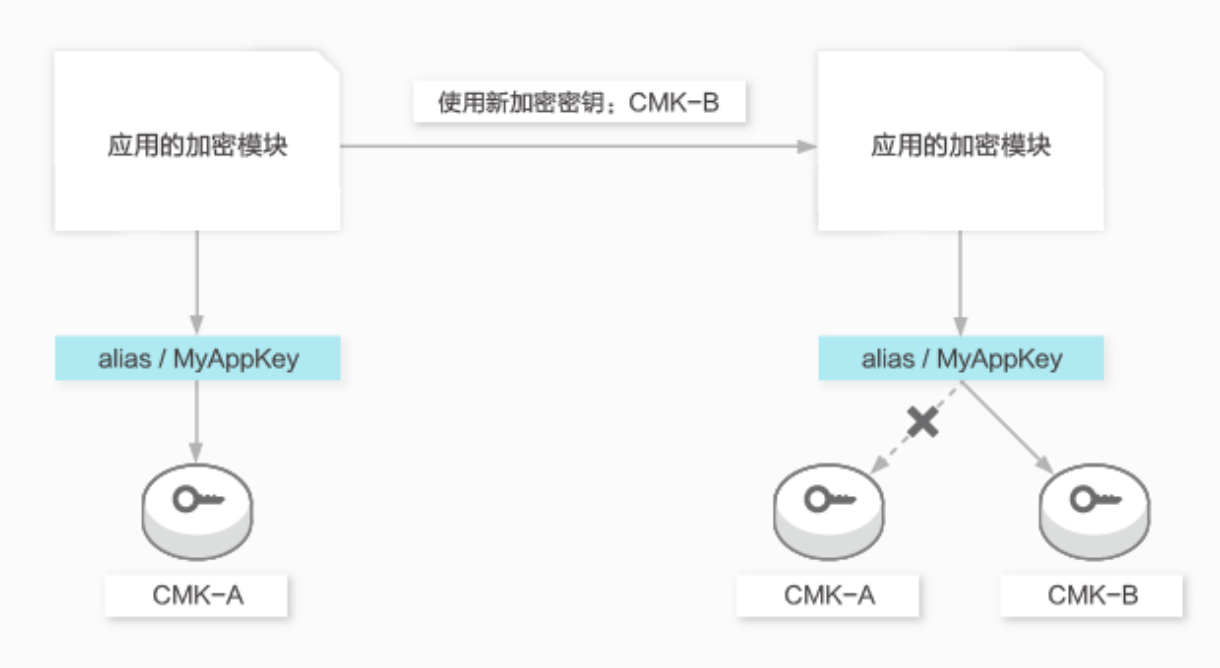
如果您使用的密钥类型不支持基于密钥版本的自动轮转，您可以基于使用场景，直接轮转使用的用户主密钥（CMK），通过人工的方式实现密钥轮转。由于这种方式建立在CMK之上，对于支持和不支持自动轮转的CMK而言，都可以作为特殊场景下的替代技术方案。

应用自定义加密场景

云上或者云下的应用可以调用KMS API实现自定义的数据加密，例如：

- 在将身份证号、信用卡号、家庭地址等敏感信息写入数据库之前进行加密
- 在上传数据到OSS之前，进行客户端加密
- 对包含敏感信息的应用配置文件、SSL私钥证书等服务配置文件进行加密

您可以通过KMS提供的密钥别名功能，实现对应用中加密密钥的轮转。KMS的解密API不需要您提供密钥的ID或者别名。



这一抽象的轮转场景包括以下步骤：

1. **初始设置**
   a. **管理员创建CMK，假定ID为CMK-A。**
   b. **管理员为上述CMK-A绑定别名alias/MyAppKey。**
   c. **应用加密模块调用Encrypt接口时，在KeyId参数传入别名alias/MyAppKey：KMS发现alias/MyAppKey绑定了ID为CMK-A的主密钥，因此使用CMK-A执行加密。**
   d. **应用解密模块调用Decrypt接口时，不传入KeyId参数，KMS使用实际用于加密数据的CMK执行解密。**
2. **人工轮转密钥**
   a. **管理员创建新的CMK，假定ID为CMK-B。**
   b. **管理员调用UpdateAlias，将别名alias/MyAppKey绑定到CMK-B。**
   c. **应用加密模块调用Encrypt接口时，在KeyId参数传入别名alias/MyAppKey：KMS发现alias/MyAppKey绑定了ID为CMK-B的主密钥，因此使用CMK-B执行加密。**
   d. **应用解密模块调用Decrypt接口时，不传入KeyId参数，KMS使用实际用于加密数据的CMK执行解密。**

云产品服务端加密场景

云产品通过集成KMS API，可以对其管理的数据进行服务端加密。云产品服务端加密的场景，从密钥轮转角度可能出现以下情形：

- KMS侧配置自动轮转策略对云产品服务端加密产生效果

  这种情形通常是因为云产品配置指定CMK加密之后，会持续性调用KMS的GenerateDataKey产生新的数据密钥，因此KMS产生新的主版本后，云产品会使用到新的主版本来加密新产生的数据密钥。这类云产品包括OSS等。对于这类情况，如果您希望具备自动轮转密钥的能力，则不能使用服务托管的默认密钥，也不能使用自带密钥（您导入到KMS的外部密钥），因为它们并不支持密钥自动轮转。
- KMS侧配置自动轮转策略对云产品服务端加密不产生效果

  这种情形通常是因为云产品配置指定CMK加密之后，只会对特定资源调用一次KMS的加密API，因此KMS即便轮转CMK产生新的密钥版本，云产品也无法使用。例如ECS对云盘的加密，只会一次性调用KMS的GenerateDataKey产生数据密钥用于卷加密，后续不会再对整个磁盘的卷加密密钥（Volume Encryption Key）进行更新。

如果您在第一种情形中使用了自带密钥或者希望对数据密钥也进行变更轮转，或者应用场景属于第二种情形，则可以通过变更配置、拷贝数据等方式达成密钥轮转的目的。这类解决方案依赖于云产品的特定功能，详情请参见相关云产品文档。

8 云产品与KMS的集成

8.1 服务端集成加密概述

阿里云为您提供云产品落盘存储加密服务，并统一使用密钥管理服务（KMS）进行密钥管理。阿里云的存储加密功能提供了256位密钥的存储加密强度（AES256），满足敏感数据的加密存储需求。

云产品和KMS在服务端的集成为您带来了如下收益：

- 增加云上数据的安全和隐私

  通过使用您在KMS中管理的密钥，云产品可以加密任何归属于您的数据。这些数据既可以是您可以直接访问的数据，也可以是您无法直接访问的云产品内部数据（例如数据库引擎产生的文件），从而为您提供对云上数据更大的控制权，保证数据的安全性和隐私性。
- 降低自研数据加密带来的研发成本

  自研数据加密往往会带来如下研发成本：

  - 需要考虑合理的密钥层次结构和对应的数据划分方式，以平衡加密性能和安全性
  - 需要考虑密钥轮转、数据重加密
  - 需要掌握密码学技术，保证加密算法的健壮性、安全性以及防篡改能力等
  - 需要考虑自研系统的工程健壮性和可靠性，以确保数据持久性

  云服务端集成加密为您考虑和解决了所有这些复杂的工程和安全问题，降低了研发成本。

选择合适的密钥

您可以根据数据保护需求，选择托管在KMS中的不同类型密钥用于服务端加密：

- 服务托管的密钥

  如果您介意密钥管理带来的开销，云产品可以为您在KMS中托管一个用于服务端加密的默认密钥，称为服务密钥。服务密钥由对应云产品管理，您对云产品使用此服务密钥的授权是隐式的。通过操作审计服务，您可以对云产品使用服务密钥的情况进行审计。

  为了方便用户识别，服务密钥的Creator属性为对应云产品的代码，同时服务密钥被关联了特殊的别名，格式为`acs/<云产品代码>`。例如，对象存储服务OSS为用户创建的服务密钥，Creator属性为OSS，同时被关联了别名acs/oss。

- 用户托管的密钥

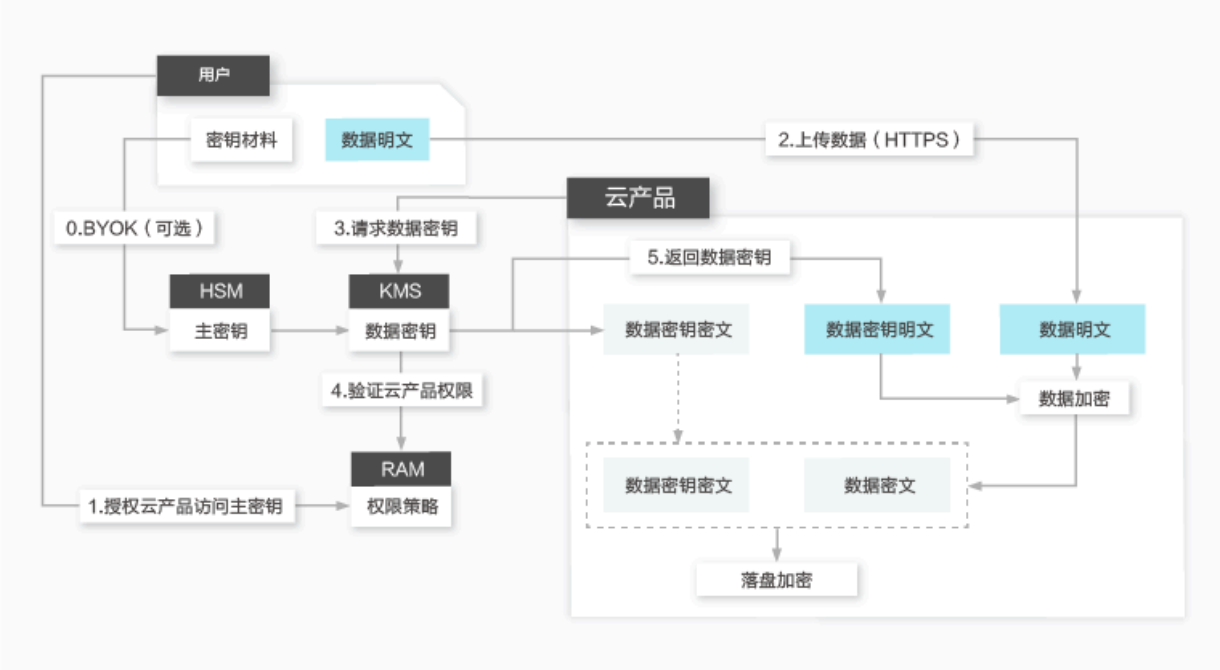
  您可以选择自己托管的密钥进行云产品服务端加密，从而获取对数据加密行为更大的控制权。由于云产品并不是密钥托管者，对密钥的使用需要获得您的显式授权。RAM服务是云产品和KMS之间的权限鉴别仲裁者，您通过RAM服务配置权限策略，允许或者拒绝云产品使用特定的主密钥，云产品访问KMS时，KMS向RAM服务验证云产品是否具备使用特定主密钥的权限。

  您除了可以让KMS生成密钥，还可以通过导入自带密钥（BYOK）的方式，将线下的密钥材料安全的导入KMS的主密钥，以获得对密钥的更大控制权。例如，您并不可以立即删除KMS生成的密钥材料，但是可以立即删除导入到KMS中的密钥材料。使用自带密钥将产生额外的管理成本，需谨慎使用，详情请参见导入密钥材料。

云产品加密的方式

不同产品基于业务形态和客户需求，其加密的具体设计略有不同。通常，存储加密中密钥层次结构会至少分为两层，并通过信封加密的机制实现对数据的加密。

第一层为KMS中的用户主密钥（CMK），第二层为数据密钥（DK）。CMK对DK进行加解密操作和保护，DK对业务数据进行加解密操作和保护。在业务数据落盘存储时，云产品会将DK的密文（由KMS使用CMK加密）与业务数据的密文（由云产品使用DK加密）一同写入持久化存储介质中。信封加密中的“信封”是指在概念上DK的密文和业务数据的密文被打包在一个“信封”（Envelope）中。在读取加密数据时，DK的密文也会一同被读取，并先于业务数据进行解密。只有在DK被解密后，密文的业务数据才能够被正常解密读取。



在信封加密机制中，CMK受KMS提供的密钥管理基础设施的保护。云产品必须通过恰当的用户授权，才可以使用对应的主密钥来产生DK，用于业务数据的加密，也只有通过用户授权，才可以使

用对应的主密钥来解密DK的密文，用于业务数据的解密。DK的明文仅会在您使用的云产品服务实例所在的宿主机的内存中使用，不会以明文形式存储在永久介质上。

8.2 服务端集成加密的云产品

本文介绍了当前支持服务端集成加密的云产品。云产品通过使用服务托管密钥或者允许用户自选密钥（包括BYOK - 自带密钥）对数据进行服务端加密保护。

服务端集成加密功能可以以低成本实现云上数据的加密保护，为业务数据提供安全边界，提升阿里云对您业务安全的保障能力。这不仅适用于您可直接获取的业务数据，也适用于您只能间接访问的业务数据。

块存储云盘

块存储-云盘的加密功能默认使用服务密钥为用户的数据进行加密，同时也支持使用用户自选密钥为用户的数据进行加密。云盘的加密机制中，每一块云盘（Disk）会有相对应的用户主密钥（CMK）和数据密钥（DK），并通过信封加密机制对用户数据进行加密。

使用ECS云盘加密功能，系统会将从ECS实例传输到云盘的数据自动加密，并在读取数据时自动解密。加密解密在ECS实例所在的宿主机上进行，在加密解密的过程中，云盘的性能几乎没有衰减。

在创建加密云盘并将其挂载到ECS实例后，系统将对以下数据进行加密：

- 云盘中的静态数据
- 云盘和实例间传输的数据（实例操作系统内数据不加密）
- 从加密云盘创建的所有快照（即加密快照）

说明:

块存储云盘加密也适用于容器服务，详情请参见*#unique_40*。

对象存储 OSS

OSS支持在服务器端对上传的数据进行加密（Server-Side Encryption）：上传数据时，OSS对收到的用户数据进行加密，然后将得到的加密数据持久化保存下来；下载数据时，OSS自动对保存的加密数据进行解密并把原始数据返回给用户，并在返回的HTTP请求Header中，声明该数据进行了服务器端加密。

OSS在支持集成KMS之前就支持了自研的加密体系，称之为SSE-OSS，即：使用OSS私有密钥体系进行服务端加密。这种方式并不使用归属用户的密钥，因此用户无法通过ActionTrail服务审计密钥使用情况。

OSS支持集成KMS进行服务端加密，称之为SSE-KMS。OSS支持使用服务密钥和用户自选密钥两种方式进行服务端加密。同时OSS既支持在桶级别配置默认加密CMK，也支持在上传每个对象时使用特定的CMK。

请参见OSS的*#unique_41*和*SDK参考*获取更多详情。

云数据库

- 关系型数据库 RDS

  RDS数据加密提供以下两种方式：

  - 云盘落盘加密

    针对RDS云盘版实例，阿里云免费提供云盘加密功能，基于块存储对整个数据盘进行加密。云盘加密使用的密钥由KMS服务加密保护，RDS只在启动实例和迁移实例时，动态读取一次密钥。

  - 透明数据加密TDE

    RDS提供MySQL和SQL Server的透明数据加密（Transparent Data Encryption，简称TDE）功能。TDE加密使用的密钥由KMS服务加密保护，RDS只在启动实例和迁移实例时动态读取一次密钥。当RDS实例开启TDE功能后，用户可以指定参与加密的数据库或者表。这些数据库或者表中的数据在写入到任何设备（磁盘、SSD、PCIe卡）或者服务（对象存储OSS、归档存储OAS）前都会进行加密，因此实例对应的数据文件和备份都是以密文形式存在的。

  请参见以下文档获取更多详情：

  - MySQL：*#unique_43*、*#unique_44*
  - SQL Server：*#unique_45*、*#unique_46*

- MongoDB

  MongoDB加密和RDS类似，请参见*#unique_47*获取更多详情。

应用配置管理 ACM

ACM通过和KMS集成，对应用配置进行加密，确保敏感配置（数据源、Token、用户名、密码等）的安全性，降低用户配置的泄露风险。ACM服务端和KMS的集成有以下两种方式：

- 在KMS服务端直接加密

  ACM服务通过KMS的数据加密API，将配置传送到KMS端，指定CMK完成对配置的加密。

- **在ACM服务端信封加密**

  通过KMS的API，使用指定CMK来保护生成的DK，使用DK在ACM服务端完成对配置的加密。

请参见*#unique_48*获取更多详情。

文件存储 NAS

NAS的加密功能默认使用服务密钥为用户的数据进行加密。NAS的加密机制中，每一卷（Volume）会有相对应的CMK（目前只支持NAS的服务密钥，后续会对自选密钥进行支持）和DK，并通过信封加密机制对用户数据进行加密。

表格存储 Table Store

Table Store的加密功能默认使用服务密钥为用户的数据进行加密，同时也支持使用用户自选密钥为用户的数据进行加密。表格存储的加密机制中，每一个表格（Table）会有相对应的CMK和DK，并通过信封加密机制对用户数据进行加密。

大数据计算 MaxCompute

MaxCompute支持使用服务密钥作为主密钥进行数据加密。

云存储网关 CSG

CSG支持基于OSS对数据进行加密，详情请参见*#unique_49*。

媒体处理 MTS

MTS支持私有加密和HLS标准加密两种方式，均可以集成KMS对视频内容进行保护，详情请参见*MTS*官方文档。

视频点播 VOD

VOD支持阿里云视频加密和*HLS*标准加密两种方式，均可以集成KMS对视频内容进行保护。

Web应用托管服务 Web+

Web+使用KMS对应用托管服务中使用的敏感配置数据进行加密保护，这包含了类似*RDS*数据库的访问凭证等机密信息。