密钥管理服务KMS介绍

密码产品

202308

目录

云时代下的数据安全风险

- 密码泄漏与合规风险
- 密码技术演进

“云”时代面临的数据安全威胁

数据泄露/破坏

服务故障

勒索蠕虫

脚本黑客

专业黑灰产

间谍、内鬼

网络战

云上业务高度集中，数据高度集中，在数据流转不断产生新价值的同时，面临来自内外部各种安全威胁，从而带来合规和安全防护方面挑战。

“云”时代面临的合规挑战

中华人民共和国
网络安全法

2016年11月

为了保障网络安全,维护网络空间主权和国家安全、社会公共利益,保护公民、法人和其他组织的合法权益,促进经济社会信息化健康发展,制定本法。

中华人民共和国
密码法

2019年10月

为了规范密码应用和管理，促进密码事业发展，保障网络与信息安全，维护国家安全和社会公共利益，保护公民、法人和其他组织的合法权益，制定本法。

中华人民共和国
数据安全法

2021年6月

为了保障数据安全，促进数据开发利用，保护公民、组织的合法权益，维护国家主权、安全和发展利益，制定本法。

中华人民共和国
个人信息保护法

2021年8月

为了保护个人信息权益，规范个人信息处理活动，保障个人信息依法有序自由流动，促进个人信息合法利用，制定本法。

《国家政务信息化项目建设管理办法》：明确政务信息化项目建设时，网络安全必须同步规划、同步建设，系统建成后须提交等保测评报告、密码应用安全性评估报告。

密码在网络安全的重要作用

密码技术是对抗网络攻击、维护网络空间安全的核心技术和基础支撑

中华人民共和国网络安全法

重要数据备份和加密，防止网络数据泄露或者被窃取、篡改。

中华人民共和国密码法

关键基础设施应使用商用密码进行保护，并进行商用密码应用安全性评估

中华人民共和国个人信息保护法

针对个人信息应采取相应的加密、去标识化等安全技术措施防止未经授权的访问以及个人信息泄露、篡改、丢失

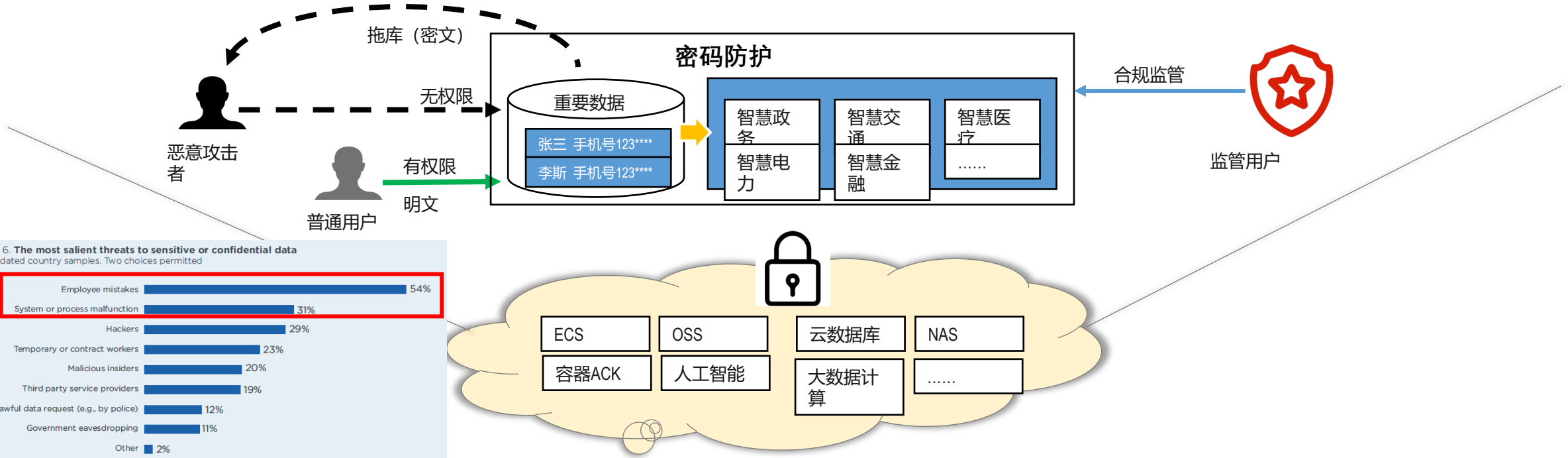
关键信息基础设施安全保护条例

运营者应保障关键信息基础设施安全稳定运行，维护数据的完整性、保密性和可用性。

商用密码管理条例

等保三级以上网络、国家政务信息系统等网络与信息系统应使用商用密码进行保护，制定商用密码应用方案，运行商用密码保障系统

监管合规、防攻击拖库



密码服务演进：传统密码盒子产品到云原生密码服务

传统密码产品

- 硬件盒子
- 提供独立的密码应用能力

- 密码资源无法按需弹性伸缩
- 密码资源服务运维运营复杂
- 资源管控和服务接口不统一

传统密码产品虚拟化

- 密码服务虚拟化
- 密码资源池化
- 提供综合的密码应用能力

- 资源的隔离性与网络的安全性难以保障
- 与云产品、云上云原生安全体系融合程度低
- 同城容灾、信创混部等云平台特性难拉齐

云原生密码服务

- Cloud Native
- 密码产品saas化
- 提供丰富与云深度融合的弹性密码服务能力

- 密码能力融入云和业务，成为与云环境中计算、存储、网络资源并列的云资源，实现密码服务与云平台的无缝集成，实现云密码资源的统一管理、使用和运维。

密码产品简介

- 产品架构
- 核心功能
- 产品优势

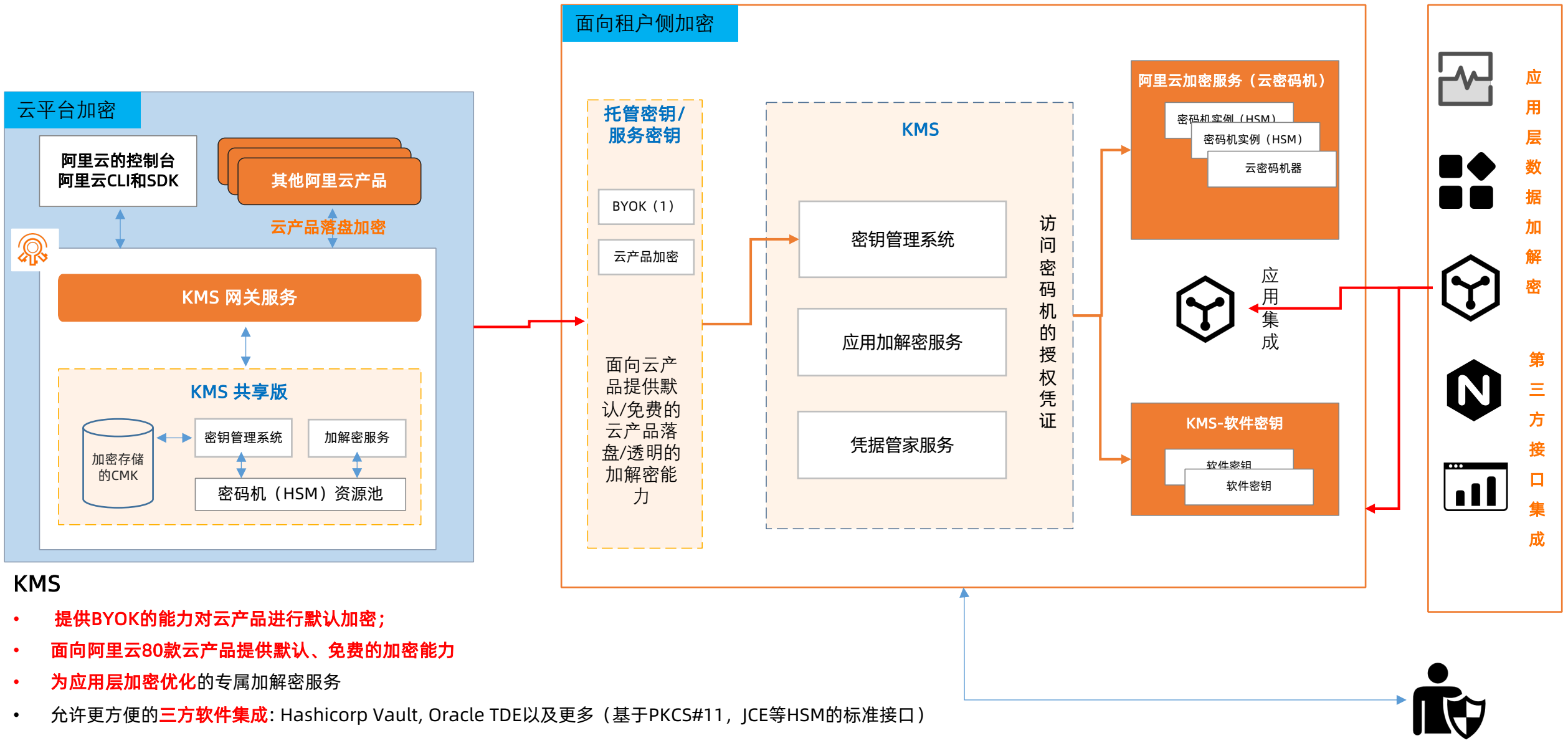
阿里云云上密码服务产品思路

全面能力：满足云平台和云租户应用密评需求以及资源独立；
安全隔离：租户独享密码资源和密码服务，租户间资源隔离；

统一管理：支持与云管平台集成，统一管理密码资源；
统一权限：对接云管平台账户体系、统一权限控制；
云原生：支持密码资源服务自动化部署、配置、监控、调度
态势感知：提供密码服务运行情况感知能力，支持对接密码监管

弹性：快速扩容和收缩能力，资源利用最大化
高可用：基于云服务保障高可用服务能力
多类型：基于伙伴丰富的密码服务提供多类型的服务能力

阿里云安全密码产品简介

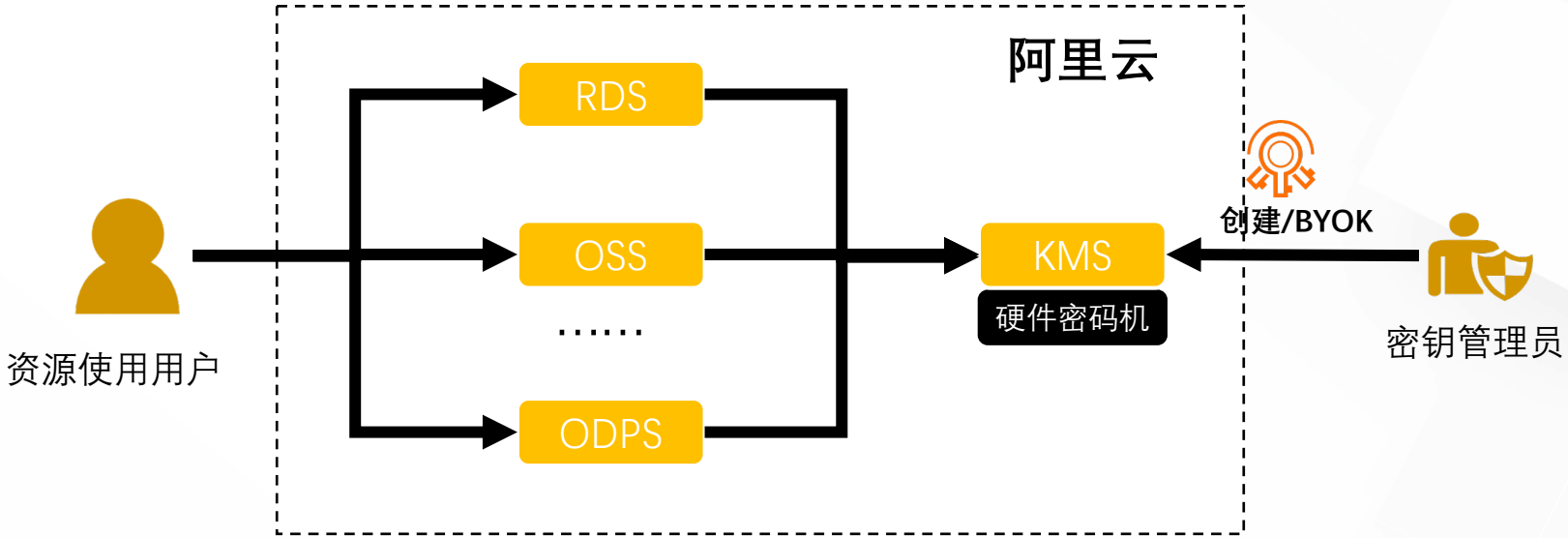


KMS

- **提供BYOK的能力对云产品进行默认加密；**
- **面向阿里云80款云产品提供默认、免费的加密能力**
- **为应用层加密优化**的专属加解密服务
- 允许更方便的**三方软件集成**: Hashicorp Vault, Oracle TDE以及更多（基于PKCS#11，JCE等HSM的标准接口）

“一键加密” 云产品：构建云上默认安全能力

- 与阿里云ECS、RDS、OSS、Maxcompute等云产品集成，解决云产品中原生数据的加密保护问题。支持阿里云79款云产品默认加密的能力；
- 支持自带密钥（BYOK）导入KMS用作用户主密钥（CMK），从而更好的满足用户密钥管理需求。



应用加密：对称加密（信封加密、实时加密）非对称加密（签名）

- **KMS提供密钥安全分发能力，应用可在本地实现数据加解密**
- **业务层加密可以更好的防范复杂的安全威胁，按需构建层次化数据防护策略**

KMS支持加密算法

- 对称加密： AES, SM4
- 非对称加密： RSA, SM2
- 数字签名： RSA, P-256, secp256k1, SM2

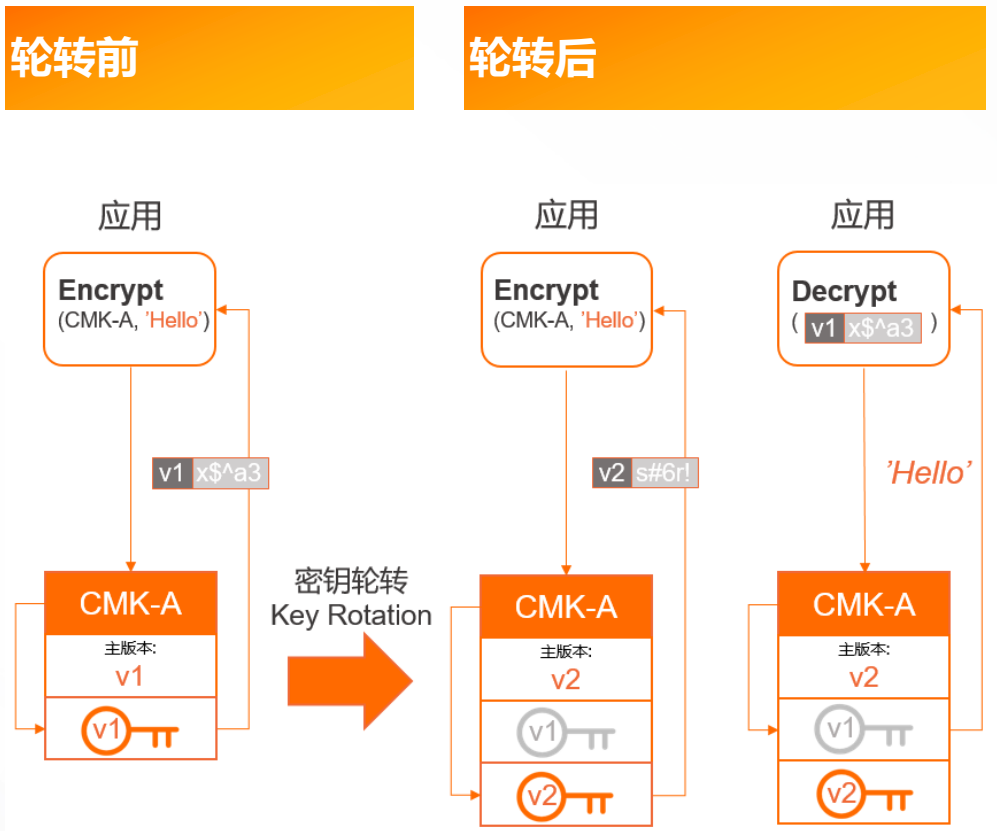
应用示例

- 各语言示例

  GitHub Samples for KMS SDK

密钥轮转：自动化密钥轮转能力，有效解决密钥到期或合规要求

- 周期性轮转密钥，减小密码攻击面，提升系统密码安全应用能力



轮转方式

◆ 自动轮转

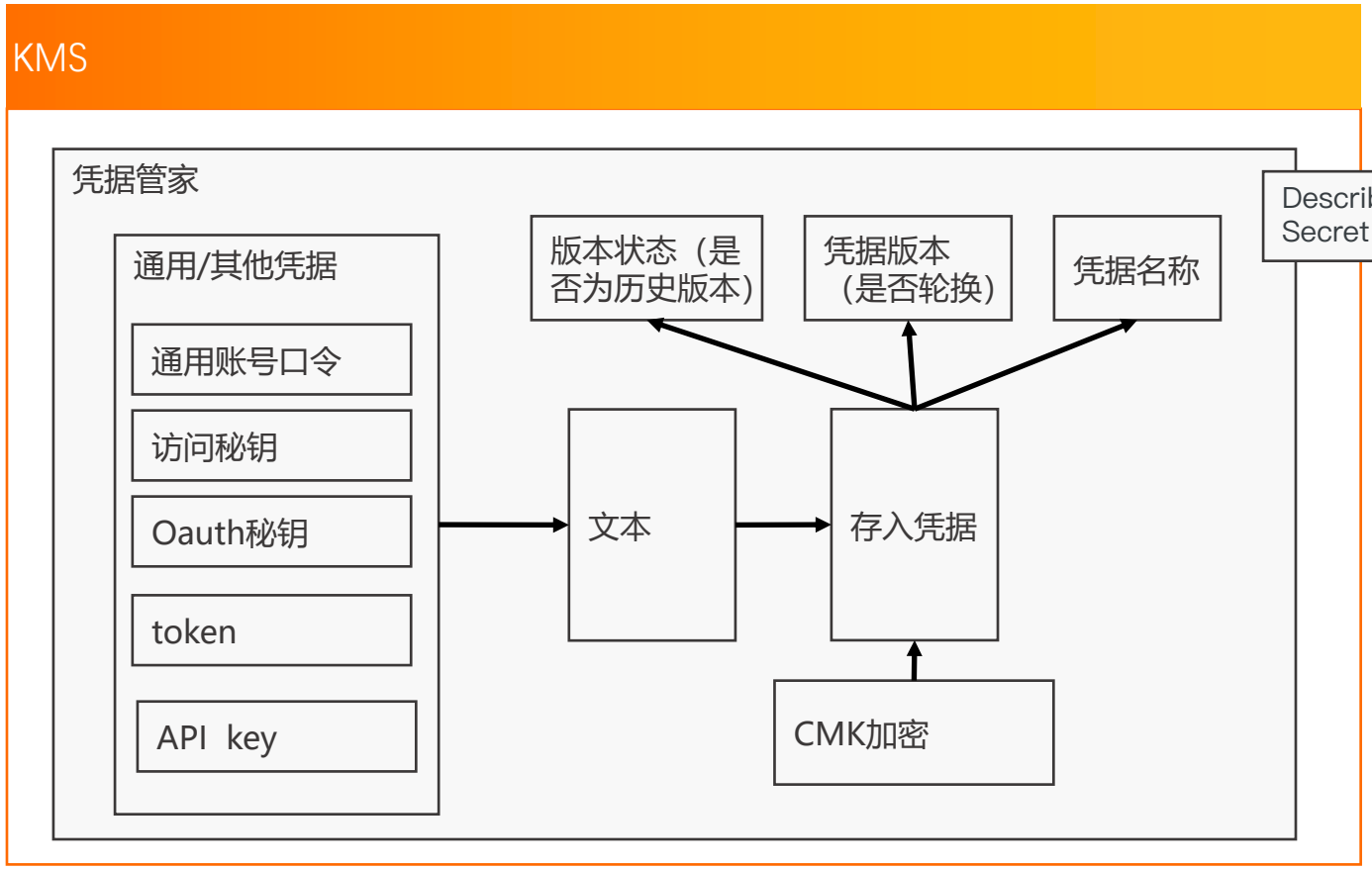
调用CreateKey创建CMK时，可以指定CMK的自动轮转策略，也可以调用UpdateRotationPolicy变更当前的自动轮转策略，设定自动轮转的周期。

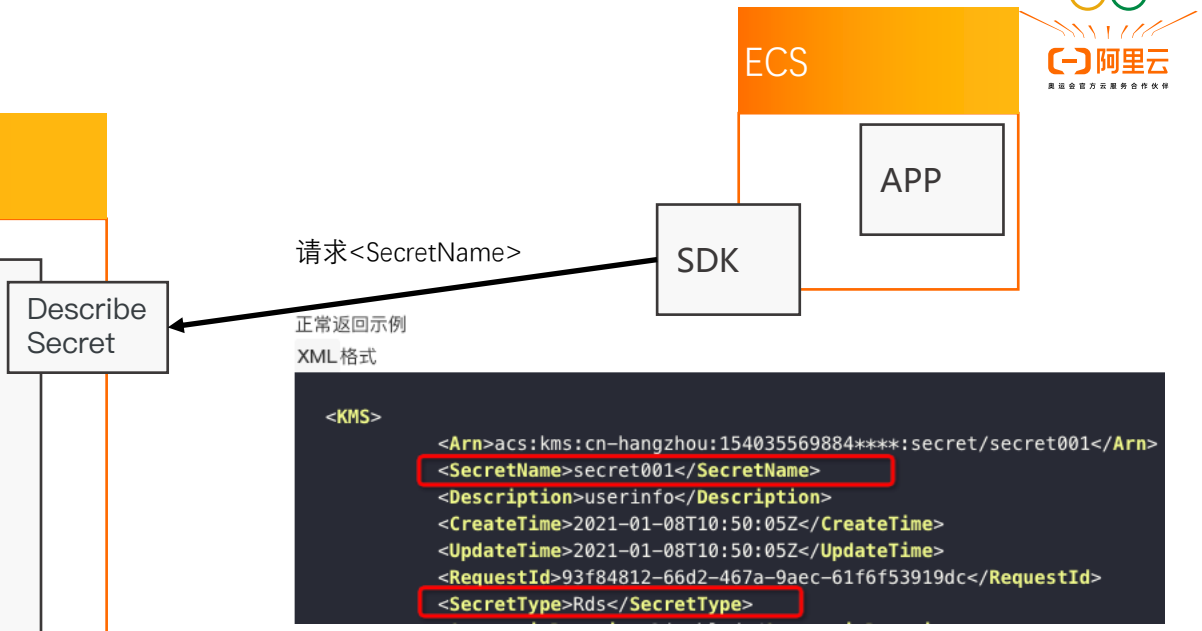
◆ 人工轮转

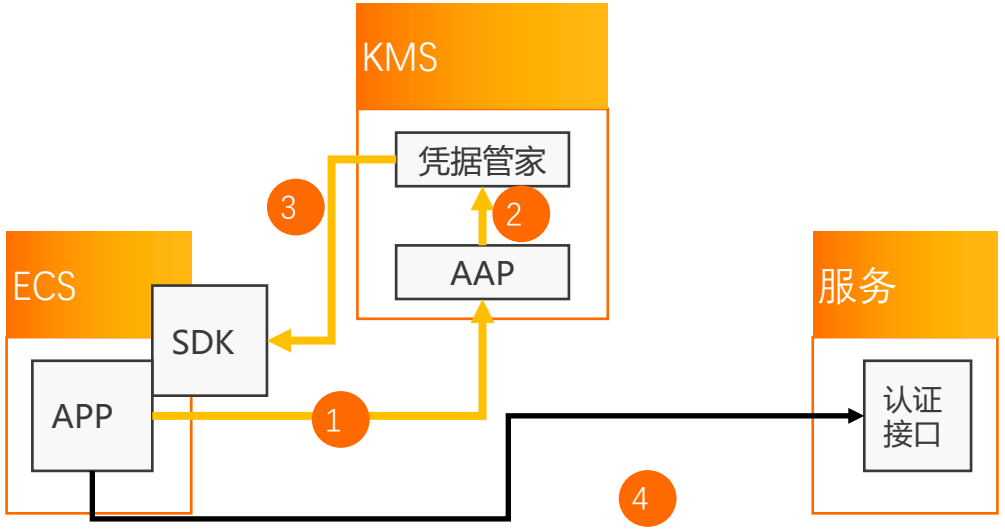
1. 加密时cmkid参数使用别名
2. 产生新cmk，将别名绑定到新cmk
3. 解密时api自动寻找cmkid解密

KMS凭据管家与密钥托管内部实现逻辑

普通凭据生成过程







1. APP向凭据管家完成AAP应用接入认证，确保APP端可信
2. AAP认证通过后根据RBAC获取相关凭据，通过【DS】接口获取<SecretName>
3. 凭据管家返回APP相关凭据信息（<SecretName>对应的登录口令）
4. APP在登录配置文件中通过<SecretName>字段获取登录口令并完成登录

阿里云KMS产品优势

- **天然集成阿里云ECS、RDS、OSS、RAM等众多云产品，满足用户云产品加密安全需求。**
- **结合阿里云安全稳定能力，为应用提供安全、合规、易用和可控的密钥管理服务。**

安全合规

KMS经过严格安全设计和审核，使用监管机构许可认证的密码设施，保证您的密钥在阿里云得到最严格的保护。

云产品默认集成

KMS具备统一管理租户云产品密钥和应用密钥的能力，是云上租户开箱即用的密钥管理中心，支持ECS、OSS、RDS、Maxcompute等云产品落盘加密。

高可靠、高可用

依托阿里云高可靠和高可用的云平台底座及KMS自身高可用架构设计，提供了99.9% 的服务可用性保障。

易用、易运维

KMS提供密码机全托管所需的自动运维能力，租户只需关注加密业务，无需关注繁琐密码机运维操作。

最佳实践

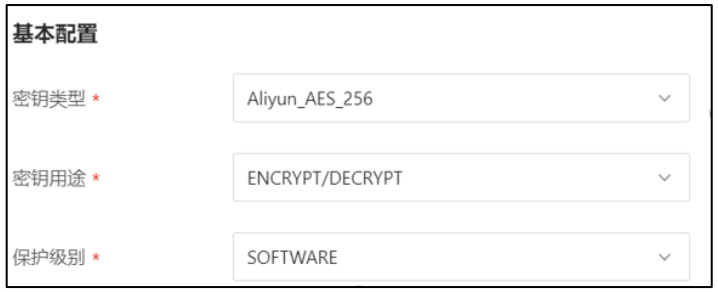
- KMS密钥加密
- KMS-凭据管家

KMS密钥管理---业务加解密

KMS典型应用场景-云产品加密及应用加密

云产品透明加密实践

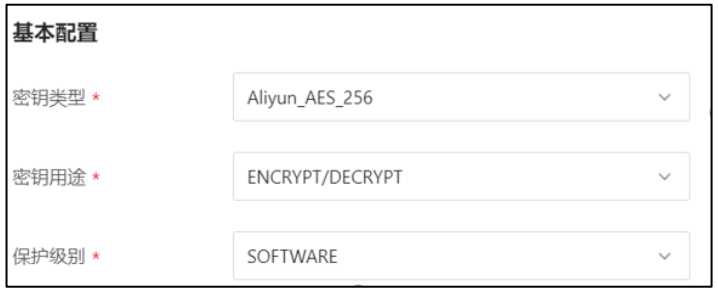
1 使用kms创建用户主密钥



2 在云产品控制台开启云产品加密

应用加密实践

① 使用kms控制台创建用户主密钥CMK



② 应用系统调用KMS sdk获取密钥在本地实现数据加解密

- **信封加密**

1. **生成数据加密秘钥：GenerateDataKey(cmk-id) ->{DK, EDK}**
2. **加密数据：本地加解密**

③ 应用系统将加密后的数据存入数据库

KMS应用层数据加密-云产品加密

应用场景

1、黑客攻击数据库沦陷导致数据被脱库。

2、存储数据的磁盘处理不规范（损坏、社工攻击）导致数据泄露。

数据库防明文脱库方案

使用KMS针对所有云上数据启用全盘加密防存储设备数据泄露，针对高敏感数据启用应用层字段级加密，防应用层脱库。

应用实践：

① 云产品透明加密

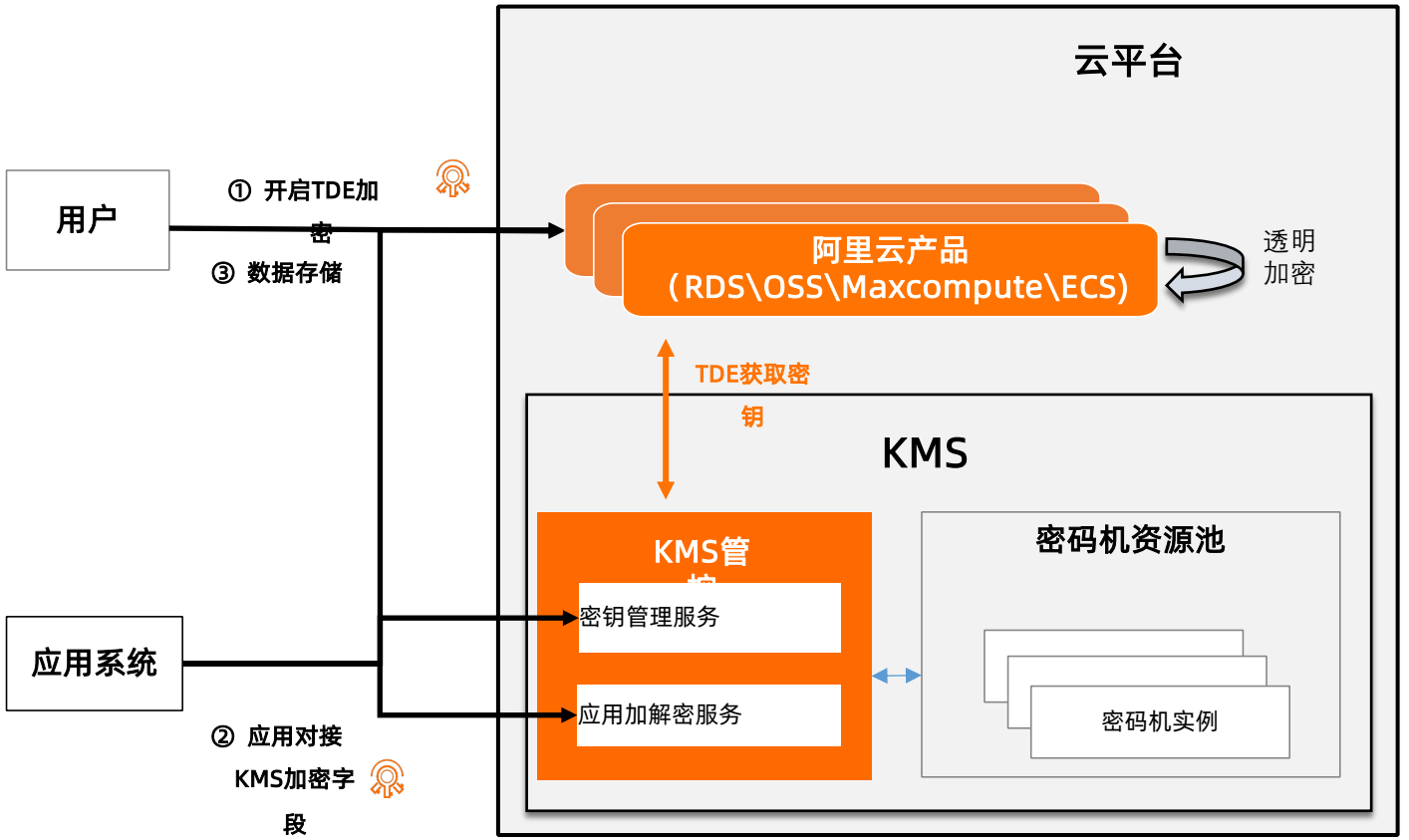
- 开启云产品“一键”默认加密能力，全盘透明加密在ECS、RDS、OSS、Maxcompute内的所有用户数据，防止存储设备泄露风险。

② 应用层加密

- 高敏感数据调用信封加密接口获取密钥，在应用系统内完成加密，加密后存储到ECS、RDS、OSS、Maxcompute中，防止脱库泄露威胁。

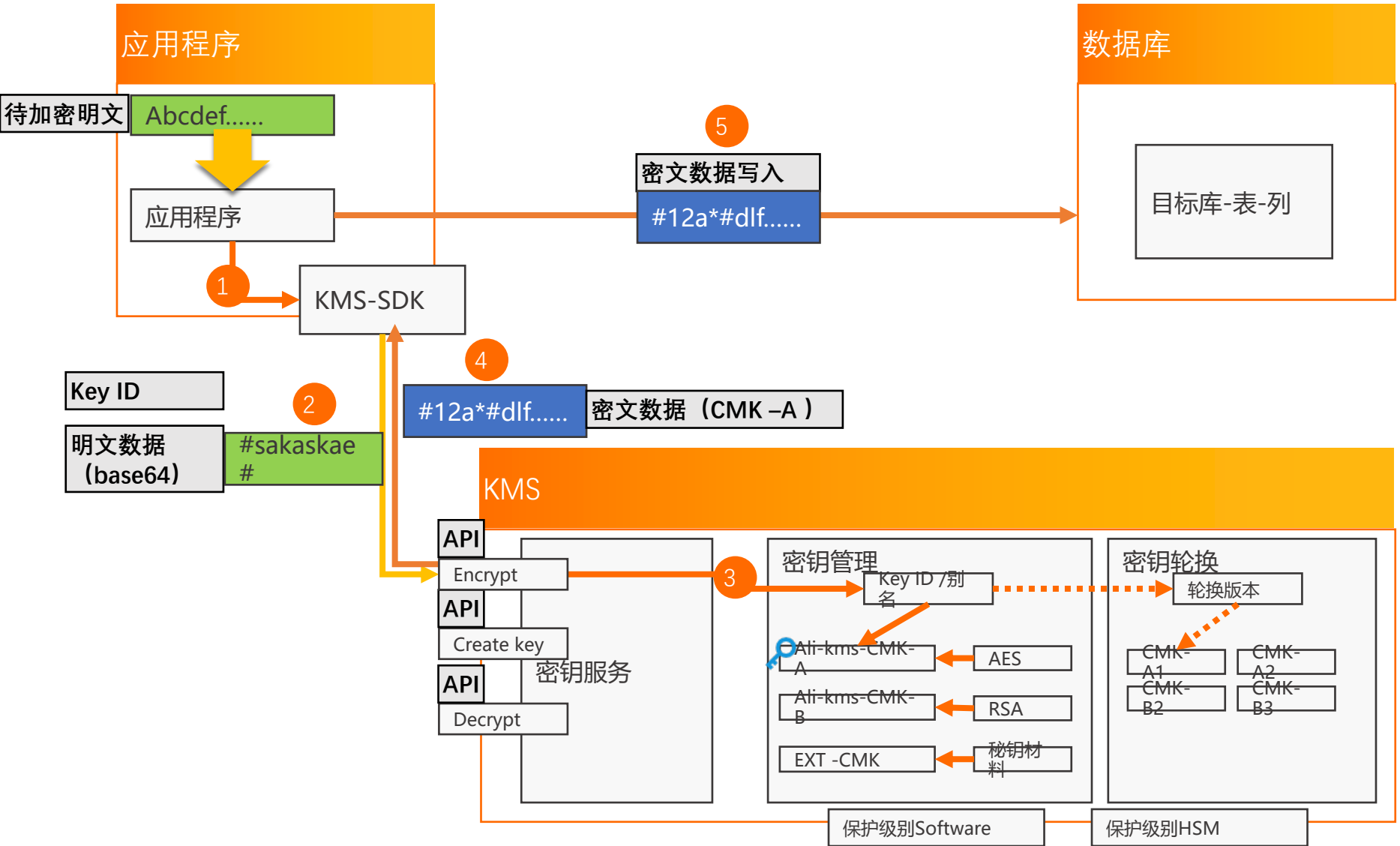
③ 密钥管理

- 密钥生命周期管理，密钥硬件级安全防护能力，黑客拿不到密钥就解不出数据。



场景1：使用KMS在线加密数据（对称加密）

应用场景：当有少量数据（小于6k）需要加密时，可直接使用KMS进行加密。



加密过程

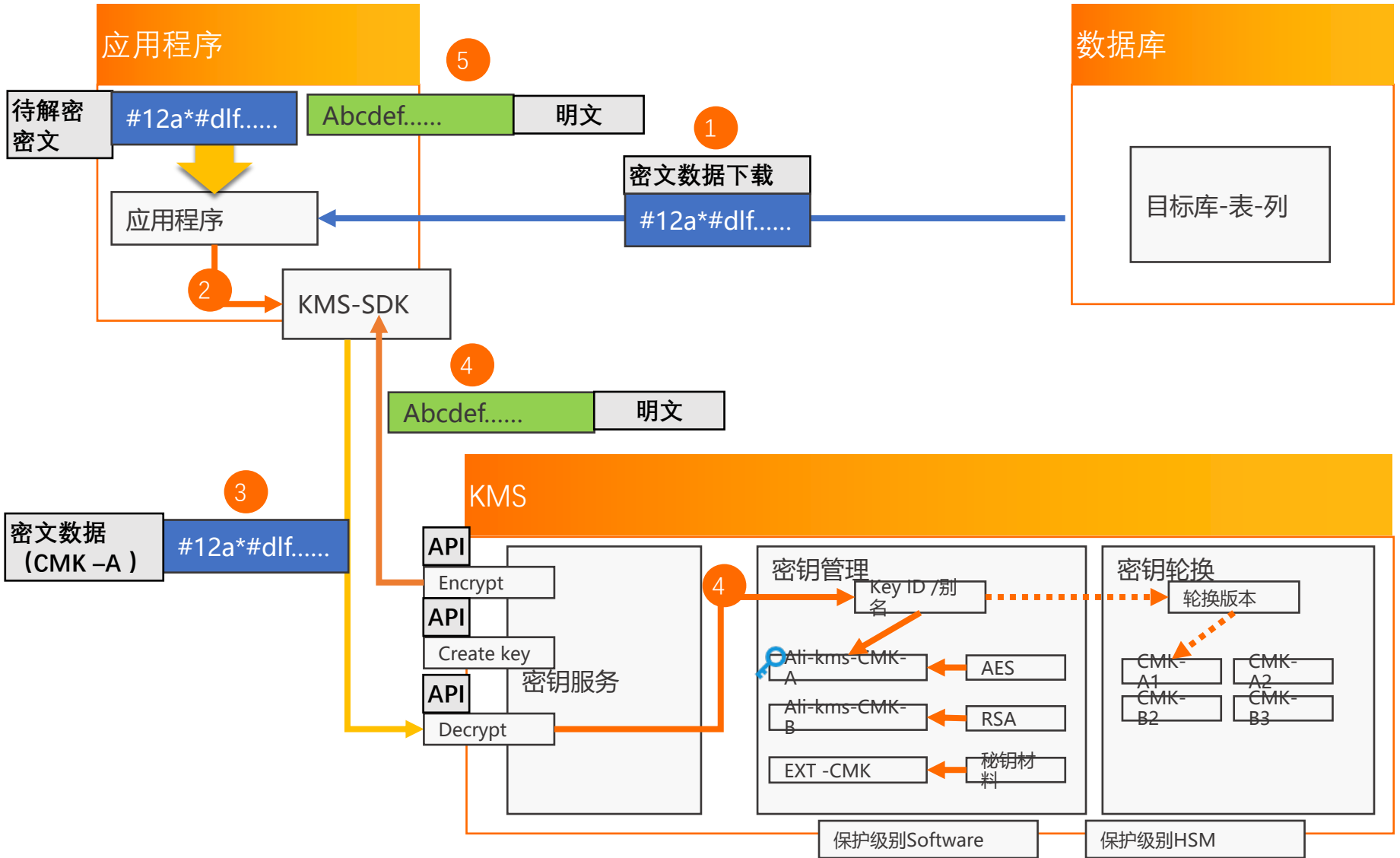
1. 应用程序调用【Encrypt】接口，请求KMS进行数据加密；
2. 传参“KeyID”“明文base64”到KMS的服务端；
3. KMS通过识别Key ID 或别名，对明文数据进行加密；
4. KMS通过【Encrypt】接口返回密文数据；
5. 应用程序收到密文数据后在将密文数据（字符串）写入数据库。

注意事项

1. 确保加密数据一定不能过大；
2. 加密过程发生在KMS内；

场景2：使用KMS在线解密数据（对称加密）

应用场景：当有少量数据（小于6k）需要解密时，可直接使用KMS进行解密。



加密过程

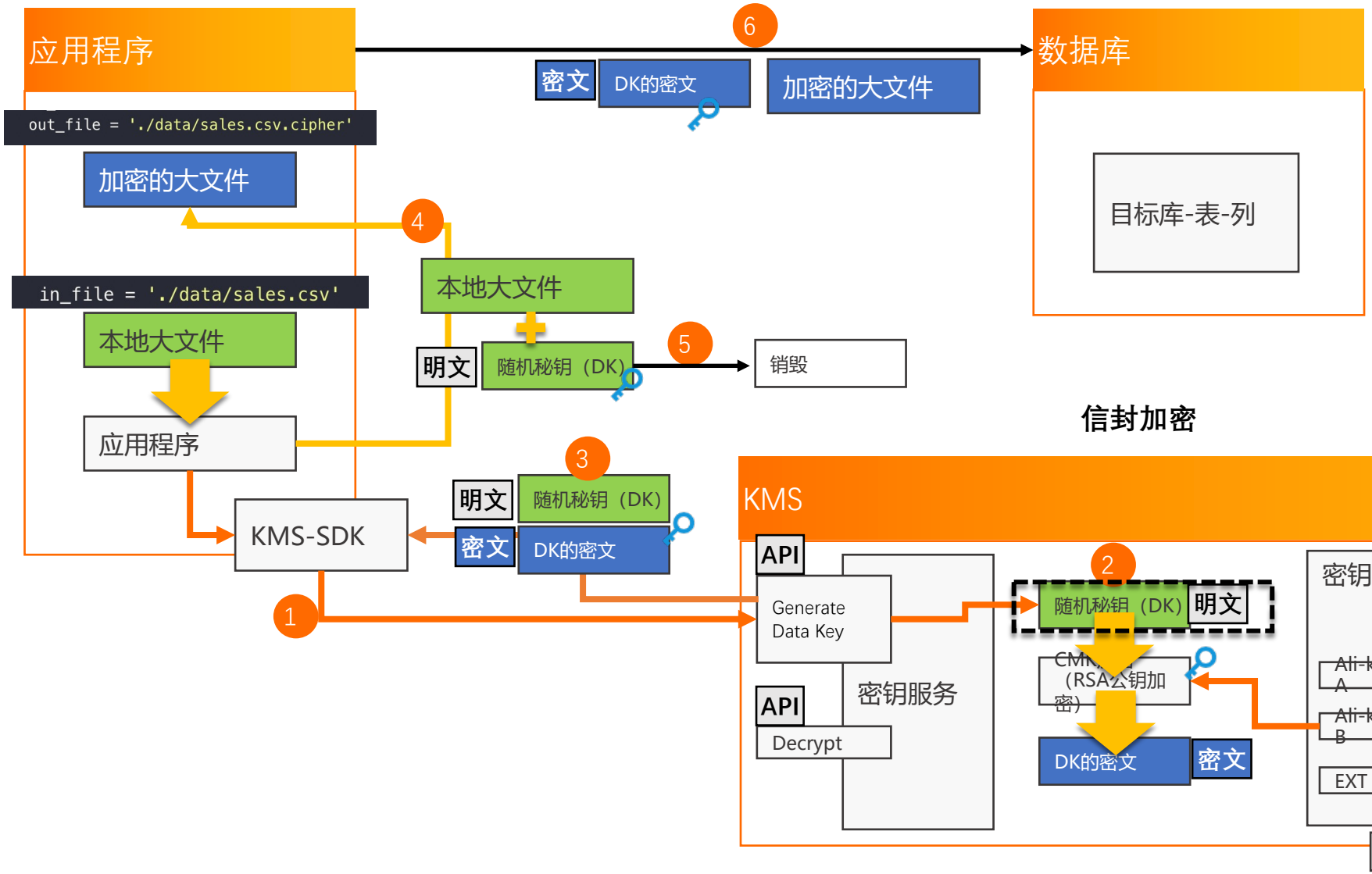
1. 应用程序获取密文；
2. 应用程序调用【Decrypt】接口请求数据解密；
3. 传参“密文数据”至KMS；
4. KMS对密文数据进行解密，直接返回明文；
5. 应用程序收到明文数据；

注意事项

1. 确保加密数据一定不能过大；
2. 解密过程发生在KMS内；

场景3：使用KMS加密本地大文件数据（信封加密）

应用场景：当有大量数据要做加密时，推荐本地加密，KMS通过信封加密完成data key(DK)的安全交换。

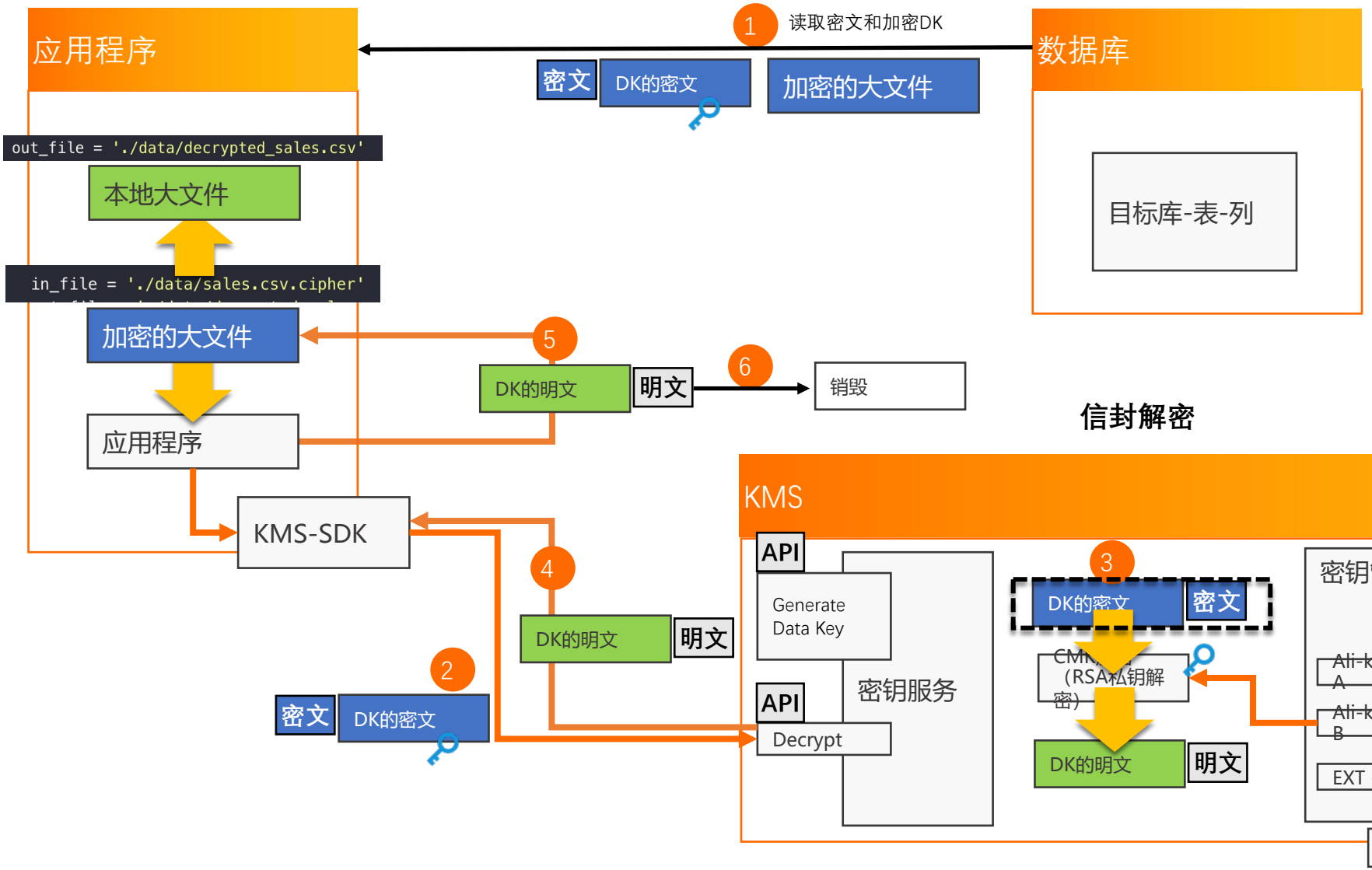


加密过程

1. 应用程序调用KMS【GDK】接口生成随机秘钥DK
2. KMS内部完成信封加密，使用RSA-CMK公钥对DK进行加密；
3. KMS返回“明文”“密文”
4. 应用程序用“明文”DK进行加密本地文件；
5. 应用程序加密后销毁“明文”DK；
6. 应用程序将DK密文和加密文件同时存储；

场景4：使用KMS解密本地大文件数据（信封解密）

应用场景：当有大量数据要做加密时，推荐本地加密，KMS通过信封加密完成data key(DK)的安全交换。



加密过程

1. 从数据库读取密文+加密的DK
2. 请求KMS【decrypt】接口解密DK
3. KMS内完成信封解密（RSA私钥解密DK）生成明文DK;
4. 通过接口返回应用程序DK（Plaintext）
5. 应用程序利用明文对加密文件执行本地解密生成明文数据
6. 应用程序立刻销毁DK明文

场景5：使用KMS完成签名验签（签名过程）

应用场景：使用KMS完成数字签名

签名的目的：

- ✓ 是为了证明是谁发送了消息，证明消息的可靠性和不可否认性，提供了身份验证。
- ✓ 部分web业务、app业务要求客户端提供签名，以防止中间人劫持。

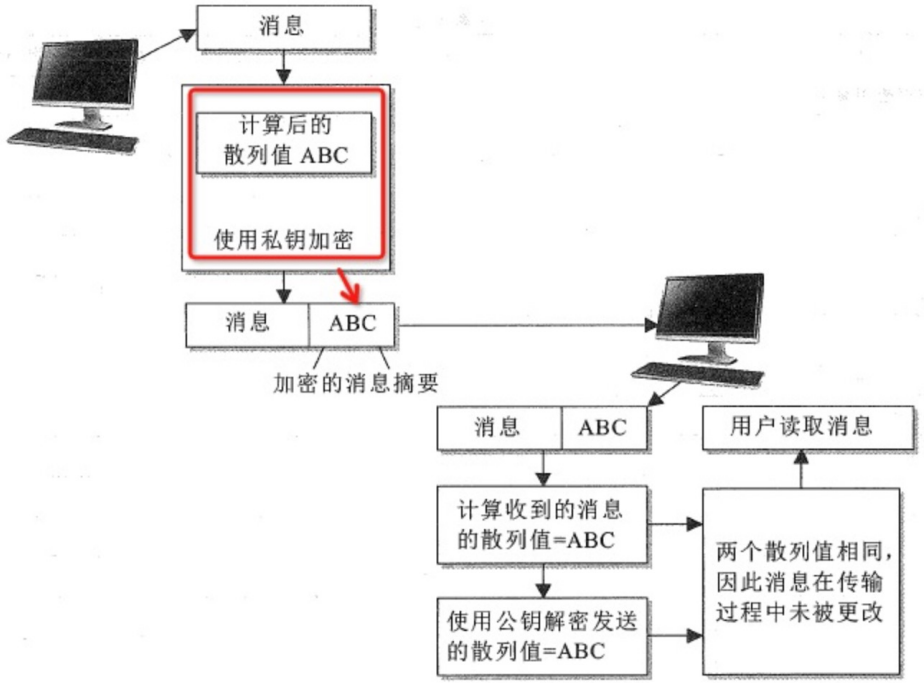
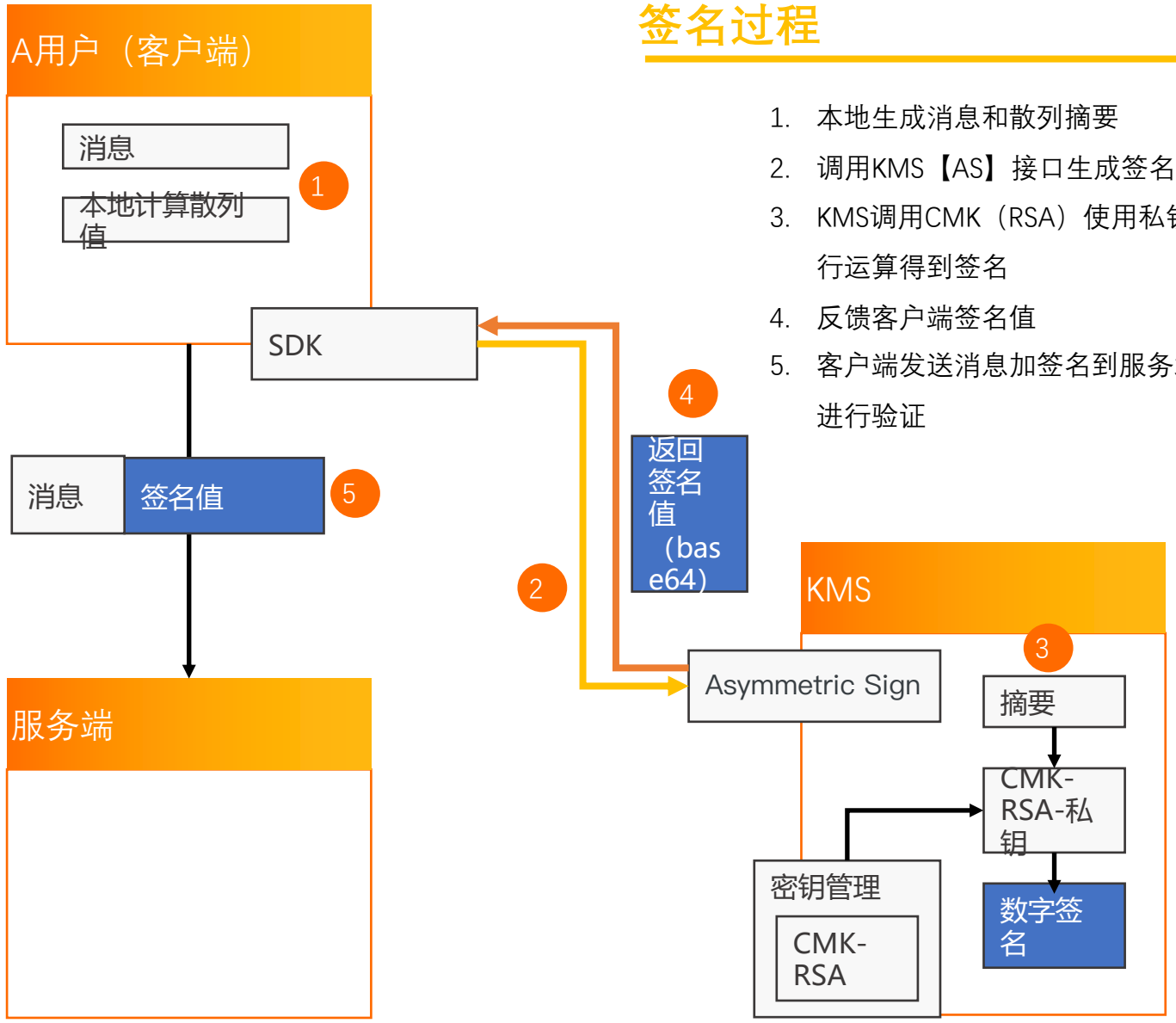


图 3-54　为消息创建数字签名

签名的标准过程



签名过程

1. 本地生成消息和散列摘要
2. 调用KMS【AS】接口生成签名
3. KMS调用CMK（RSA）使用私钥进行运算得到签名
4. 反馈客户端签名值
5. 客户端发送消息加签名到服务端进行验证

KMS密钥管理---凭据管家

KMS凭据管家接入方案

方法一：使用 KMS SDK

使用KMS托管凭据

Name	Value （加密存储）
MyDbCreds	{“username”:“Alice”,“password”:“123456”}
MySSHKey	A-BINARY-BLOB

配置文件：MyApp.ini

```
[mysql]
secret_name=MyDbCreds
```

应用代码：MyApp.java

```
String secretValue = kmsClient.GetSecretValue(
          config.Read(“secret_name”));
JSONObject obj = new JSONObject(secretValue);
String userName = obj.getString(“username”);
String password = obj.getString(“password”);
Connection conn = DriverManager.getConnection(
          url, userName, password);
```

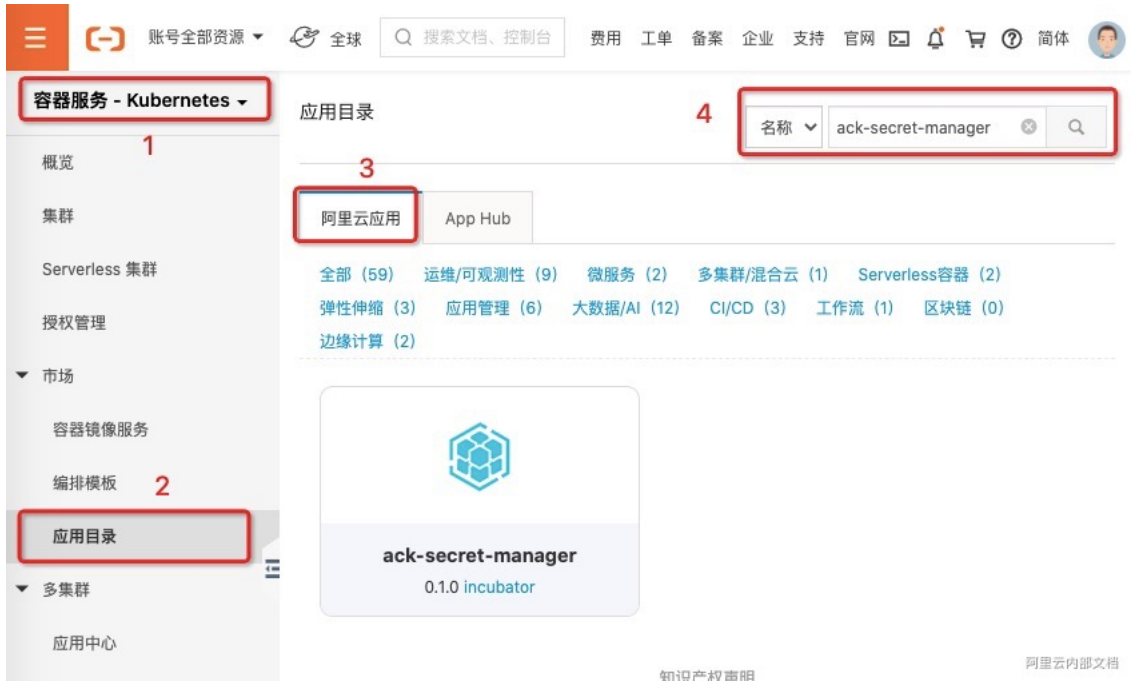

方法二：使用 SecretsManager Client

- 应用代码和使用KMS SDK类似
- 客户端缓存、从服务端自动获取更新
- 可以配置多地域读取
- 默认错误重试策略

方法三：使用 Kubernetes 插件

提供 K8s原生Secret的使用体验，动态获取使用凭据管家中的Secret

- **选项 1：ACK Kubernetes插件 ack-secrets-manager**



- **选项 2：Kubernetes 三方插件 kubernetes-external-secrets**

注：K8s的SecretsManager插件和K8s集群的Secrets加密功能并不冲突。使用KMS一键加密Kubernetes集群Secret，加密的对象为K8s Etcd中的静态Secret、和SecretsManager动态Secret的本地缓存。

场景1：使用KMS凭据管家实现AK的托管

参考官网文档： [动态RAM凭据]

➢ 管理员（管理系统）操作

1. 创建和管理

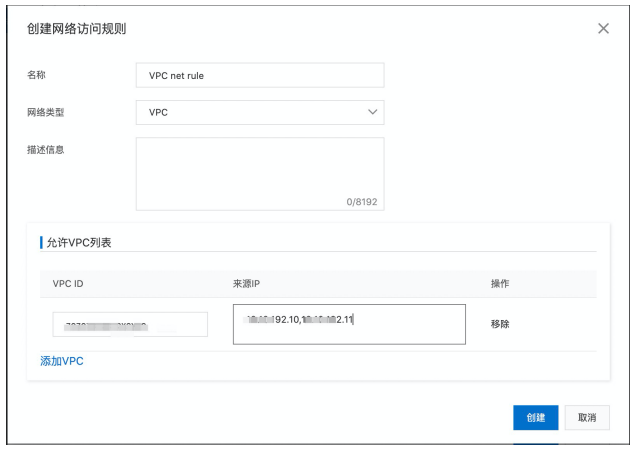
❑ 最佳实践

1. 建议用户以“新建RAM用户”方式进行创建，避免AK轮转对使用其的未改造用户造成影响；
2. 设定自动轮换周期，应用程序通过SDK自动获取轮换后的AK

❑ 管理系统集成

通过OpenAPI和企业自身的账号管理系统集成，收敛所有云账号、子账号的凭据生命周期和应急响应

2. 配置AK到应用的分发规则：应用接入点（AAP）



➢ 业务研发如何使用

```
String secretName = "******";
String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";
String accessKeyId = "LTAI5tNGr...B";
String secretAccessKey = "JSDGnRkG...";

OSSClient ossClient = new OSSClient(endpoint, accessKeyId, secretAccessKey);
// business code
List<Bucket> buckets = ossClient.listBuckets();
for (Bucket bucket : buckets) {
    if (bucket != null) {
        // ...
    }
}
```

```
String secretName = "******";
String endpoint = "https://oss-cn-hangzhou.aliyuncs.com";

// must use the follow method to in the client after the process starts
AliyunSDKSecretsManagerPluginsManager.init();
// get Oss Client
OSSClient ossClient = SecretsManagerOssPluginManager.getOssClient(endpoint, secretName);
// business code
List<Bucket> buckets = ossClient.listBuckets();
for (Bucket bucket : buckets) {
    if (bucket != null) {
        // ...
    }
}
```

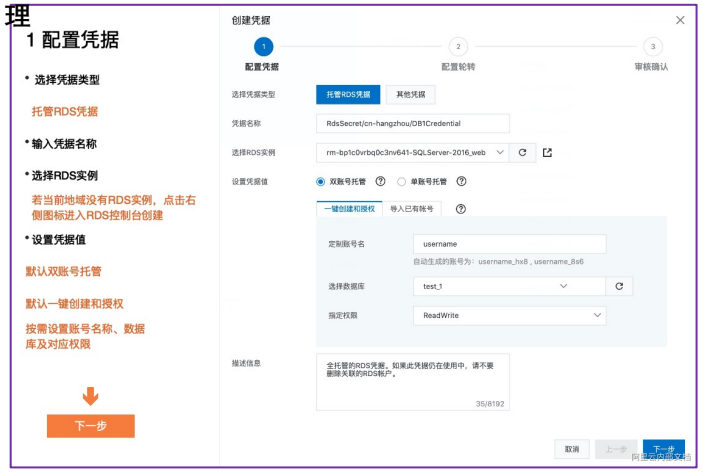
代码仅需配置AK对应的“Secret Name”

场景2：使用KMS凭据管家实现RDS的托管

参考官网文档：[动态RDS凭据]

➢ 管理员（管理系统）操作

1. 创建和管理



❑ 最佳实践

1. 建议用户采用"一键创建和授权"方式进行创建；
2. 如果是在应用中使用RDS，建议采用**双账号托管**；
3. 如果用户在后续使用过程自定义授权方式，**"指定权限"处建议选择"暂不授权"**；

❑ 管理系统集成

通过OpenAPI和企业自身的云资源管理系统集成，收敛资源对应的凭据生命周期和应急响应

➢ 业务研发如何使用

[开源项目Secrets Manager JDBC]：可以快速集成动态RDS凭据方案，减少用户开发成本。

- JDBC、C3p0、 MyBatis、JPA、Jdbc Template 主流框架
- 下面示例标红部分为此方案修改内容部分

```
    Class.forName("com.aliyun.kms.secretsmanager.MysqlSecretsManagerSimpleDriver");
    Connection connect = null;
    try {
      connect = DriverManager.getConnection("secrets-manager:mysql://<host>:
<port>/database>","#your-mysql-secret-name#","");
    } catch(SQLException e) {
      e.printStackTrace();
    }
```

RDS托管凭据Java JDBC代码示例

```
c3p0.user=#your-mysql-secret-name#
c3p0.driverClass=com.aliyun.kms.secretsmanager.MysqlSecretsManagerSimpleDriver
c3p0.jdbcUrl=secrets-manager:mysql://<mysql-ip>:<port>/<your-database-name>
```

RDS托管凭据Java C3p0配置示例

```
  <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource" >
      <property name="driverClass"
value="com.aliyun.kms.secretsmanager.MysqlSecretsManagerSimpleDriver" />
      <property name="user" value="#your-mysql-secret-name#" />
      <property name="jdbcUrl" value="secrets-manager:mysql://<host>:
<port>/<database>" />
      <property name="maxPoolSize" value="500" />
      <property name="minPoolSize" value="5" />
      <property name="initialPoolSize" value="20" />
  </bean>

  <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate" >
      <property name="dataSource" ref="dataSource" />
  </bean>
```

RDS托管凭据Java JDBC Template配置示例

应用程序接入点AAP实现动态安全访问KMS托管凭据-1

应用场景

1. 应用动态访问KMS托管凭据的情况下，通常采用应用程序集成SDK方式访问KMS，为了安全的访问，需要配置AAP实现鉴权和访问控制
2. 凭据管家API接口不适用于应用程序动态的获取托管在KMS的凭据。需要通过SDK实现。

凭据管家SDK

```
public static void main(String[] args) throws CacheSecretException {
    String secretName = "<secret-name>";
    String roleName = "<your-ecs-ram-role-name>";
    String regionId = "<region-id>";
    SecretCacheClient client = SecretCacheClientBuilder.newCacheClientBuilder(
            DefaultSecretManagerClientBuilder.standard().withCredentialsProvider(new InstanceProfileCredentialsP
                    .build()
```

KMS-SDK

```
// 设置RAM角色。
String roleName = "<your-ecs-ram-role-name>"; // 本示例使用"EcsRamRoleTest"。
```

接入方式

接入方式

应用程序可以使用多种开发工具接入凭据管家，不同开发工具说明及其应用场景如下：

开发工具	说明	应用场景
KMS SDK	KMS SDK帮助您构造HTTPS请求，更好地使用KMS API。	• 获取凭据值频率较低 • 需要对凭据进行创建、删除、增加新版本的凭据值等操作
凭据管家客户端	凭据管家客户端（SecretsManager Client）支持简单地配置客户端缓存频率，并定时刷新存储在凭据管家的凭据。	• 在客户端周期性或者频繁获取凭据值 • 需要根据凭据值进行相关操作
凭据管家JDBC客户端	凭据管家JDBC客户端（SecretsManager JDBC）支持在JDBC连接中简单地使用托管在凭据管家的凭据。	使用动态RDS凭据通过Java程序访问数据库
多种阿里云SDK的托管凭据插件	多种阿里云SDK的托管凭据插件是一个帮助您更有效通过动态RAM凭据快速使用阿里云服务的插件。	使用动态RAM凭据获取阿里云访问凭据访问阿里云服务
凭据管家Kubernetes插件	凭据管家Kubernetes插件是一种以无代码方式快速集成凭据管家能力的插件。	以无代码的方式周期性更新配置

AAP配置应用接入点可信分发策略（RBAC和网络访问控制）-2

参考官网文档： 配置应用接入点

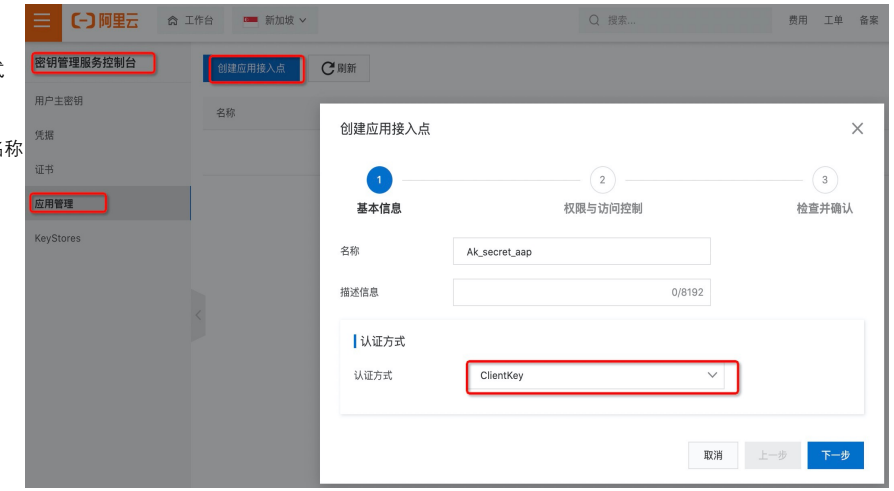
①创建应用接入点（基于加密Client 证书）

1.配置接入点

- 选择认证方式

  Client Key

- 输入接入点名称



②创建访问策略（指定访问资源）

2.配置策略

- 选择RBAC权限

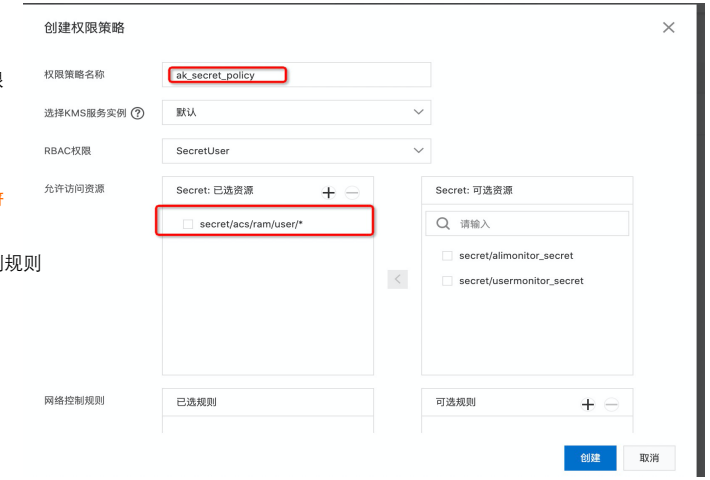
  SecretUser

- 选择访问资源

  支持资源通配符

- 选择网络控制规则

  如果没有需要的网络控制规则则创建



③创建网络访问控制

3.配置网络控制

- 选择网络类型

  VPC或公网

- 输入规则名称

- 设置ACL

  填写ACL信息

④创建加密Client 证书

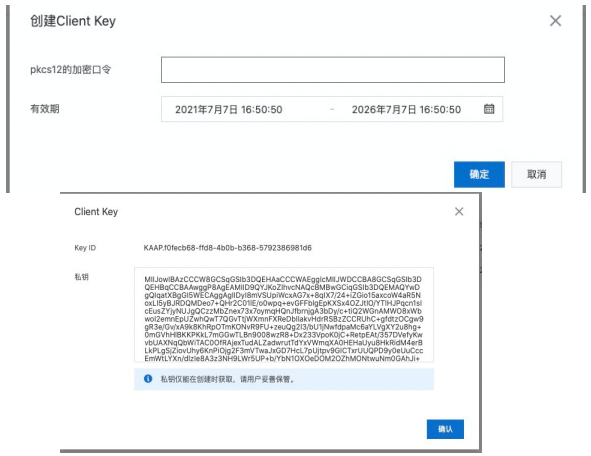
4.创建Client Key

- 输入加密口令

  此口令需要牢记

- 保存Client Key

  Client Key需要牢记



WWW.ALIYUN.COM