ALIBABA CLOUD

阿里云

加密服务

最佳实践

文档版本：20220113

阿里云

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. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，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
危险	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。	危险 重置操作将丢失用户配置数据。
警告	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。
注意	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。
说明	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	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	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.敏感信息加密

1.1. 概述

本教程提供了在应用系统中借助阿里云密码机实现应用层敏感数据加密的操作说明。

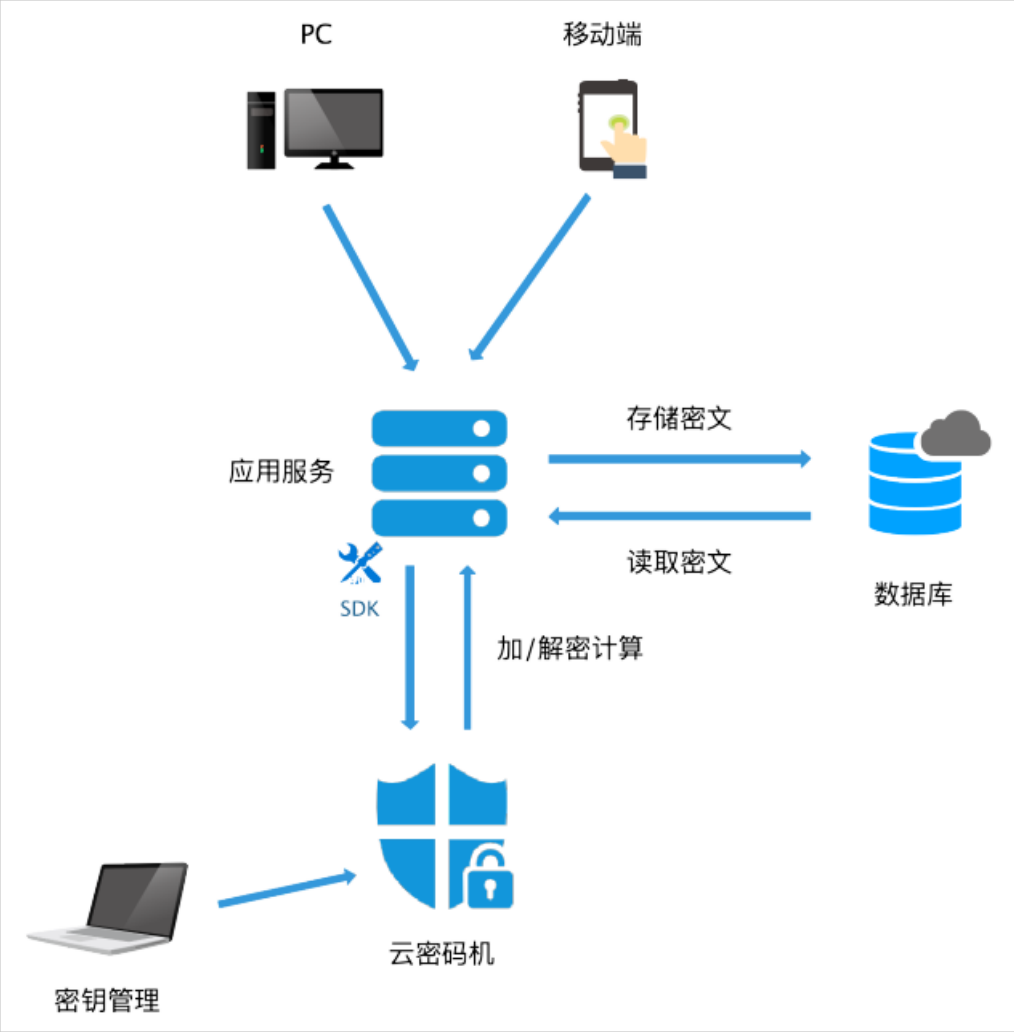
使用阿里云密码机对应用数据加密的过程如下：

1. 通过云密码机产生加密密钥。
2. 使用加密密钥对应用数据明文进行加密产生密文，并将应用数据密文返回给应用系统。
3. 应用系统把密文数据存储到数据库。

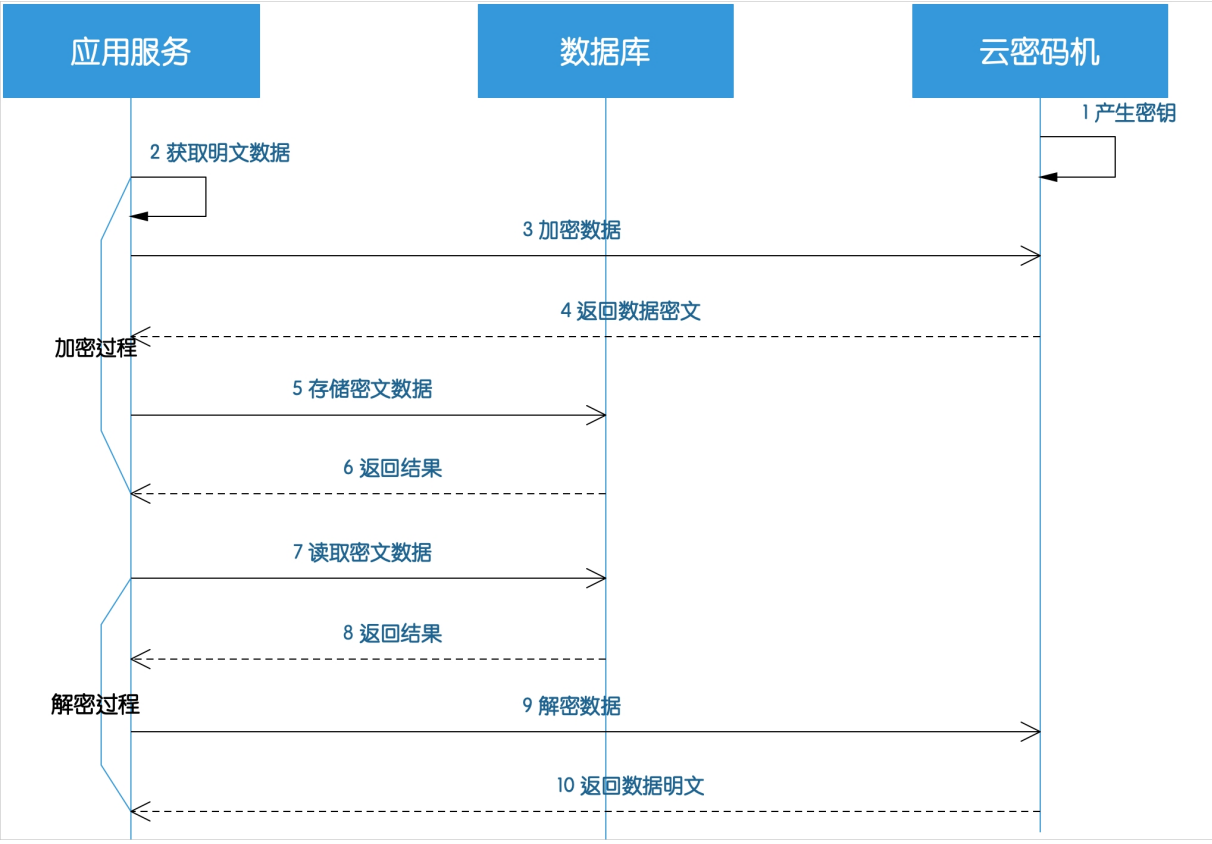
使用阿里云密码机对应用数据解密的过程如下：

1. 应用系统从数据库读取已加密的密文数据，并将应用数据密文透传给云密码机进行解密。
2. 密文解密后云密码机将明文信息返回给应用系统。

下图为在应用系统中实现敏感数据加密和解密的典型部署方案。



云密码机提供密钥生成和管理、数据加密和解密功能。整个密码算法运算过程都在云密码机中完成，数据通过云密码机处理后把密文数据存储在数据库中，提高了系统的安全性。下图为敏感数据加密的时序图。



资源准备

您需要准备以下云资源：

云资源类型	规格	说明
数据库	无要求	用于对应用数据密文进行安全存储和读取。
应用服务器	64位Linux系统	用于部署用户的业务系统。
EVSM	TASS EVSM	用于提供加密和解密等密码服务。

您需要准备的软件资源：

VsmManager.exe：EVSM管理工具，请您提交工单获取该工具。

1.2. 步骤1：配置密码机客户端

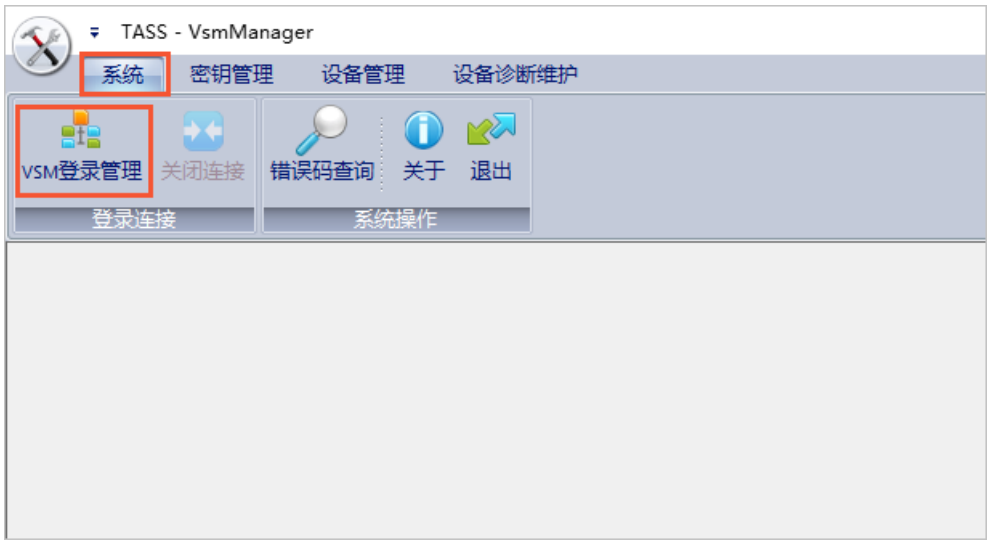
本文介绍了配置密码机客户端的具体操作。

背景信息

在加密服务试用阶段，您无须配置密码机客户端。如果您的业务需要正式上线，在购买加密服务实例后，您需要配置密码机客户端。

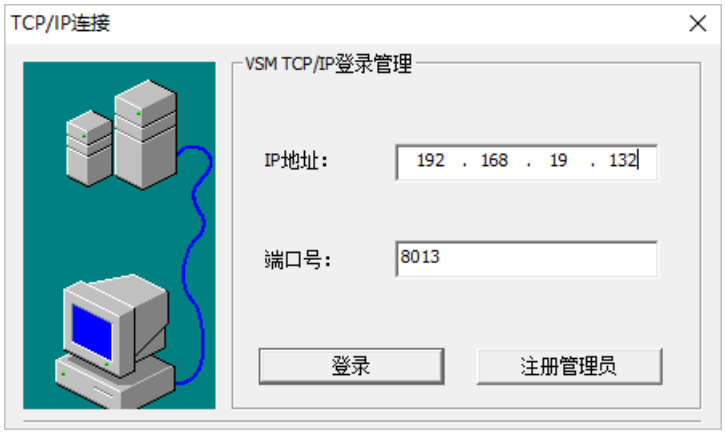
操作步骤

1. 打开密码机客户端。
2. 在顶部菜单栏单击**系统**页签，并单击**VSM登录管理**。



3. 在**TCP/IP连接**对话框中，单击**注册管理员**，注册第一个管理员UKEY并使用第一个管理员UKEY登录EVSM。

   具体操作，请参见登录EVSM（有USB KEY）。



4. 在**设备管理**页签，单击**UKEY管理**，至少再添加一个管理员UKEY。

   具体操作，请参见登录EVSM（有USB KEY）。

5. 在**密钥管理**页签，单击**原始初始化**，产生至少2个域名主密钥DMK（Domain Master Key）成份的UKEY。

   > **说明** 建议您采用3选2授权控制机制并制作3个授权UKEY。

6. 在**设备管理**页签，单击**主机端口属性**。

   您需要配置消息报文头长度为 `0` 、消息报文编码格式为 `ASCII` 、主机服务通讯方式为 `明文` 。

7. 在**设备管理**页签，单击**操作授权**，为主机服务的密钥管理类别授权。

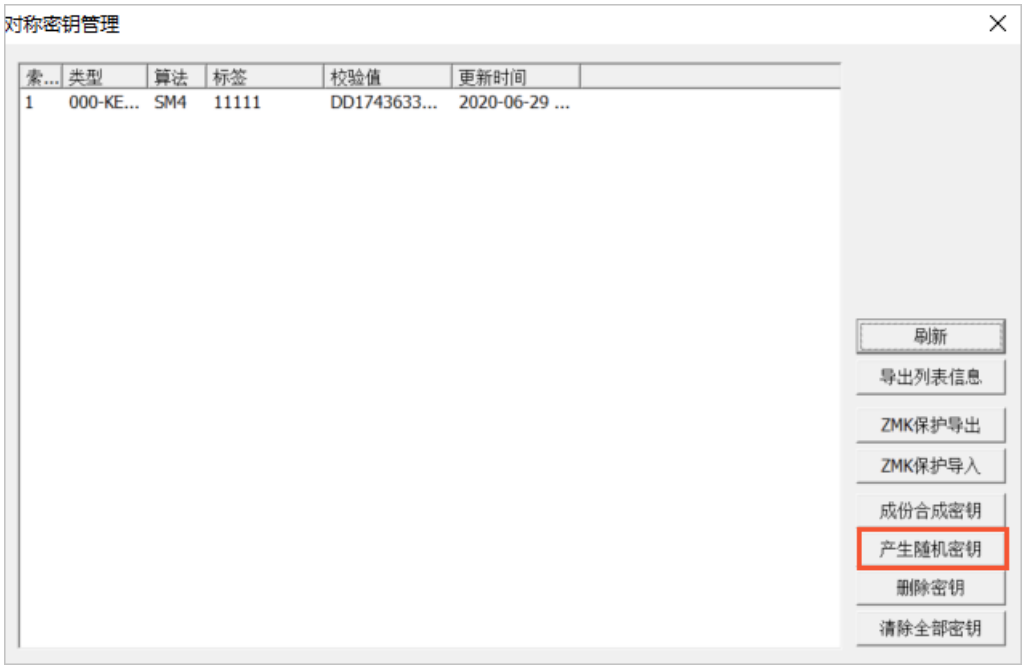
1.3. 步骤2：生成密钥

本文介绍了通过VSM管理工具或调用Java接口来生成工作密钥的操作方法。

通过VSM管理工具生成工作密钥

1. 登录EVSM。
2. 在顶部菜单中，单击**密钥管理**。
3. 在**密钥管理**菜单下，单击**对称密钥管理**。

4. 在**对称密钥管理**对话框中，单击**产生随机密钥**。



5. 在**产生随机对称密钥**对话框中，根据需要选择要产生密钥的**算法标识**、**密钥类型**，输入**密钥索引**、**密钥标签**，单击**产生**。

> **说明** 随机对称密钥默认存储到VSM内，您如果选择了**存储到密码机内索引**，则根据输入的密钥索引值，存储到指定索引中，覆盖并删除原有密钥。

通过执行上述步骤，EVSM将产生新的随机密钥，同时会输出和显示密文和校验值。

调用Java接口生成工作密钥

您可以调用 `hsm.genWorkKey(keyType,algFlag,keyIndex,keyLable)` 接口生成工作密钥。具体操作，请参见调用Java接口生成工作密钥。

1.4. 步骤3：调用Java接口

本文档主要介绍了通过调用Java接口生成工作密钥和进行通用数据加解密的操作方法。

整体流程

1. 初始化SDK。具体内容请参见初始化SDK。
2. 调用Java接口生成工作密钥、进行数据加密和解密。具体内容，请参见调用Java接口生成工作密钥、调用Java接口进行数据加密、调用Java接口进行数据解密。
3. 配置Java接口。具体内容，请参见配置Java接口。

初始化SDK

使用以下方法初始化SDK。

```
hsmGeneralFinancehsm=hsmGeneralFinance.getInstance("./cacipher.ini");
```

调用Java接口生成工作密钥

调用以下Java接口生成工作密钥。

```
hsm.genWorkKey(keyType,algFlag,keyIndex,keyLable);
```

返回值

返回两个值：

- 0号索引下：密钥在LMK下加密的密文。
- 1号索引下：密钥校验值。

抛出异常

```
cn.tass.exceptions.TAException //接口自定义异常。
```

接口定义

```
public String[] genWorkKey(String keyType,
                                    char keyFlag,
                                    int storeKeyIndex,
                                    String storeKeyLabel)
                             throws cn.tass.exceptions.TAException
```

请求参数

参数名称	参数类型	参数描述
keyType	String	密钥类型。支持密钥类型编码和密钥类型名称两种格式。例如：ZEK/DEK可以传"00A"和"ZEK/DEK"两种格式。取值： • 000：ZMK/KEK • 001：ZPK • 002：PVK/TPK/TMK • 003：TAK • 008：ZAK • 009：BDK • 00A：ZEK/DEK • 00B：TEK • 011：KMC • 109：MK-AC/MDK • 10C：HMAC • 209：MK-SMI • 309：MK-SMC • 402：CVK • 409：MK-DAK • 509：MK-DN
algFlag	char	在LMK下加密的密钥密文标识。取值： • Z：单倍长DES密钥 • X：双倍长3DES密钥 • Y：三倍长3DES密钥 • U：双倍长的3DES算法密钥 • T：三倍长的3DES算法密钥 • R：16字节SM4密钥 • P：16字节SM1密钥 • L：16字节AES密钥 • M：AES-192算法密钥 • N：AES-256算法密钥

参数名称	参数类型	参数描述
KeyIndex	int	密钥存储索引。取值范围：1~2048。
KeyLabel	String	密钥存储标签。包含0~16个ASII字符。 **说明** 在云密码机内部存储密钥时标记密钥的标签说明。

请求示例

```
keyType: 00A
algFlag: R
keyIndex: 1
KeyLabel: test
```

调用Java接口进行数据加密

调用以下Java接口进行数据加密。

```
hsm.generalDataEnc(algType,keyType,sm4SymmKey,disperFactor,sessionType,sessionFactor,padFla
g,inData,IV);
```

返回值

加密之后的密文数据。

抛出异常

```
cn.tass.exceptions.TAException //接口自定义异常。
```

接口定义

```
public byte[] generalDataEnc(int algType,
                             String keyType,
                             Object key,
                             String disperFactor,
                             Int sessionType,
                             String sessionFactor,
                             Int padFlag,
                             byte[] inData,
                             String IV)
                     throws cn.tass.exceptions.TAException
```

请求参数

参数名称	参数类型	参数描述

参数名称	参数类型	参数描述
algType	int	加密算法模式。取值： • 0：ECB模式加密 • 1：CBC模式加密 • 2：CFB模式加密 • 3：OFB模式加密 • 4：CTR模式加密（16字节分组长度）
keyType	String	加密数据的源密钥类型，支持密钥类型名称和密钥类型编码两种格式。例如：ZEK/DEK可以传"00A"和"ZEK/DEK"两种格式。取值： • 000：KEK • 109：MDK • 309：MK-SMC • 00A：ZEK/DEK • 00B：TEK • 011：KMC
key	Object	加密数据密钥的索引或密文。 • 当随机对称密钥的入参数据类型为int时，通过密钥索引调用密钥。 • 当随机对称密钥的入参数据类型为String时，按LMK加密的密钥密文调用密钥。
disperFactor	String	密钥分散因子的n级分散因子进行串联，且每级分散因子必须为16个字节。 **说明** 针对敏感数据加密场景该参数取值为空字符串或NULL。
sessionType	int	密钥分散因子的n级分散因子进行串联，且每级分散因子必须为16个字节。 **说明** 针对敏感数据加密场景该参数取值为空字符串或NULL。
sessionFactor	String	会话密钥因子。 • 当sessionType为1时，该参数为8字节（16H）。 • 当sessionType为2时，该参数为16字节（32H）。 • 当sessionType为5时，该参数为16字节（32H）。 **说明** 针对敏感数据加密场景该参数取值为空字符串或NULL。

参数名称	参数类型	参数描述
padFlag	int	PAD填充标识。取值： • **0**：PBOC 2 .0填充模式 • **1**：ISO/IEC 9797-1的PADDING模式2 • **2**：ISO/IEC 9797-1的PADDING模式1 • **3**：ANSI X9.23 • **4**：PKCS#5 • **5**：NoPadding模式 • **10**：PBOC 3.0填充模式 • **11**：左填充+ISO/IEC 9797-1 **说明** 针对敏感数据加密场景该参数取值为1。
inData	byte[]	输入的明文数据。 **说明** 您可以根据应用业务自行输入数据内容。
IV	String	初始向量。仅当**algType**取值为**1**、**2**、**3**、**4**时支持该参数。 • 密钥算法为64分组，该参数为8字节（16H）。 • 密钥算法为128分组，该参数为16字节（32H）。 **说明** ECB加密算法模式不需要**IV**，该参数取值为空字符串或NULL。

请求示例

```
algType: 1
keyType: "00A"
key:symmKey[0] //加密的对称密钥密文，使用EVSM内部的索引密钥，使用内部密钥时为int类型。
disperFactor: null
sessionType: 0
sessionFactor: null
padFlag: 1
inData: 根据用户业务自行输入数据内容。
IV: "00000000000000000000000000000000"
```

调用Java接口进行数据解密

调用以下Java接口进行数据解密。

```
hsm.generalDataDec(algType,keyType,sm4SymmKey,disperFactor,sessionType,sessionFactor,padFlag,symmEnc,IV);
```

返回值

解密后的数据。

抛出异常

```
cn.tass.exceptions.TAException //程序运行异常。
```

接口定义

```
public byte[] generalDataDec(int algType,
                             String keyType,
                              Object key,
                             String disperFactor,
                             int sessionType,
                             String sessionFactor,
                              int padFlag,
                             byte[] inData,
                              String IV)
                      throws cn.tass.exceptions.TAException
```

请求参数

参数名称	参数类型	参数描述
algType	int	加密算法模式。取值： • 0：ECB模式加密 • 1：CBC模式加密 • 2：CFB模式加密 • 3：OFB模式加密 • 4：CTR模式加密（16字节分组长度）
keyType	String	密钥类型。取值： • MK-SMC • ZEK • DEK • TEK
key	Object	加密数据的密钥索引或密文。
disperFactor	String	密钥分散因子的n级分散因子进行串联，且每级分散因子必须为16个字节。 **说明** 针对敏感数据加密场景该参数取值为空字符串或NULL。

参数名称	参数类型	参数描述
sessionType	int	会话密钥的产生模式。取值： • 0：不产生会话密钥。 • 1：ECB模式加密8字节会话密钥因子，得到8字节会话密钥。 • 2：ECB模式加密16字节会话密钥因子，得到16字节会话密钥。 • 3：密钥的左右8字节异或，得到8字节会话密钥。 • 4：取密钥的左8字节作为会话密钥。 • 5：CBC模式加密16字节会话密钥因子，得到16字节会话密钥。 **说明** 针对敏感数据加密场景该参数取值为空字符串或NULL。
sessionFactor	String	会话密钥因子。 • 当sessionType为1时，该参数为8字节（16H）。 • 当sessionType为2时，该参数为16字节（32H）。 • 当sessionType为5时，该参数为16字节（32H）。 **说明** 针对敏感数据加密场景该参数取值为空字符串或NULL。
padFlag	int	PAD填充标识。取值： • 0：PBOC 2 .0填充模式 • 1：ISO/IEC 9797-1的PADDING模式2 • 2：ISO/IEC 9797-1的PADDING模式1 • 3：ANSI X9.23 • 4：PKCS#5 • 5：NoPadding模式 • 10：PBOC 3.0填充模式 • 11：左填充+ISO/IEC 9797-1 **说明** 针对敏感数据加密场景该参数取值为1。
inData	byte[]	待解密的数据。
IV	String	初始向量。仅当algType取值为1、2、3时支持该参数。 • 密钥算法为64分组，该参数为8字节（16H）。 • 密钥算法为128分组，该参数为16字节（32H）。

请求示例

```
algType: 1
keyType: "00A"
key:symmKey[0] //加密的对称密钥密文，使用EVSM内部的索引密钥，使用内部密钥时为int类型。
disperFactor: null
sessionType: 0
sessionFactor: null
padFlag: 1
inData: 自行输入加密后的数据。
IV: "00000000000000000000000000000000"
```

配置Java接口

您可以通过**文件形式**或者**内容形式**配置Java接口。

- **文件形式**：支持直接将配置文件绝对路径传入初始化接口内。示例：

```
[LOGGER]
logsw=error
logPath=./
[HOST1]
hsmModel=SJJ1310
linkNum=-5
host=192.168.19.19
port=8018
timeout=5
ifHeart=yes
```

文件格式要求：

分类	要求	示例
注释	注释行以符号“#”起始，不支持行内注释。	#内容形式中的属性字段与文件形式中的属性字段保持一致。
配置域	配置域以方括号“[”和“]”标识。 **说明** **配置域**与**键名**不区分大小写，为了便于区分建议**配置域**使用大写。	[LOGGER]
配置项	配置项格式：“键名（Key）=键值（Value）”。	linkNum=-5

分类	要求	示例
配置内容	支持使用空白字符（空格或制表符）等对内容进行对齐操作。 **说明** ○ 您可以在接口内拼装字符串传递配置。 ○ 使用“{”和“}”，表示包括所有内容；使用“;”，表示换行。	无

Java接口中配置文件的基本属性配置域包括**日志属性**、**EVSM属性**和**应用属性**等，**日志属性**和**EVSM属性**的属性字段说明请参见下表。

属性分类	配置域	属性字段	属性字段说明
日志属性	[LOGGER]	**logsw**	设置日志类别的开关，每种日志通过独立的关键字开启。取值： ○ **error**：错误日志 ○ **debug**：调试日志
		logPath	设置日志文件的保存目录。 **说明** 您需确保配置文件的目录已经存在，且应用系统具有写入权限。
EVSM属性	[HOST n]	**hsmModel**	VSM类型标识，用于指定EVSM驱动。默认值SJJ1310。
		linkNum	与云密码机建立长连接的数量。默认值-10。 **说明** 数字前的负号（-）表示仅使用连接池，如果数字前没有负号表示优先使用连接池。优先使用连接池时，业务并发过高则Java接口中可能会创建短连接处理业务，对系统资源造成较大的影响。
		host	EVSM主机服务IP地址，支持设置成域名形式。
		port	EVSM主机服务端口。
		timeout	超时时间。单位为秒，默认值6秒。

属性分类	配置域	属性字段	属性字段说明
	说明 n为该EVSM在当前配置文件中从1开始的序号，Java接口会按顺序读取多个EVSM的属性，直到最后1个序号。	connTimeout	指定Java接口与逃生服务器建立网络连接的超时时间。该属性字段不存在时，使用timeout的取值。
		readTimeout	指定Java接口从逃生服务器读取信息的超时时间。该属性字段不存在时，使用timeout的取值。
		socketProtocol	通讯协议，支持TLSv1.2等版本协议。默认值TCP。

- **内容形式**：支持直接将配置信息以字符串的形式传入初始化接口内。示例：

```
# 内容形式中的属性字段与文件形式中的属性字段保持一致。
Stringconfig=
"{"
+"[LOGGER];"
+"logsw=error;logPath=./;"
+"[HOST1];"
+"hsmModel=SJJ1310;"
+"host=192.168.19.19;"
+"port=8018;"
+"connTimeout=5;"
+"}";
```

示例

```
public class SensitiveDataEnc {
    public static void main(String[] args) throws TAException {
        // 接口初始化，采用配置文件的方式。
        hsmGeneralFinance hsm = hsmGeneralFinance.getInstance("./cacipher.ini");
        // 接口初始化2，采用配置内容的方式。
//        String config =
//                "{"
//                        + "[LOGGER];"
//                        + "logsw=error;logPath=./;"
//                        + "[HOST 1];"
//                        + "hsmModel=SJJ1310;"
//                        + "host=192.168.19.19;"
//                        + "port=8018;"
//                        + "connTimeout=5;"
//                        + "}";
//
//        hsmGeneralFinance hsm = hsmGeneralFinance.getInstance(config);
        // 测试1产生随机密钥keyIndex为可变参数。
        // 当该密钥索引值取值为0时，表示不需要加密机保存生成的该条随机对称密钥。
        // 当keyIndex取值为1~2048时，表示将对称密钥储存在加密机中，且执行覆盖操作(相同索引执行覆盖操作)。
        int keyIndex = 0;
        switch (keyIndex) {
            case 0:
                // 产生随机SM4算法对称密钥。
```

```
                String keyType = "00A";
                char algFlag = 'R';
                String keyLable = "sm4Key";
                // 调用产生随机密钥接口。
                String[] symmKey = hsm.genWorkKey(keyType, algFlag, keyIndex, keyLable);
                System.out.println("对称密钥lmk下的密文值:" + symmKey[0]);
                System.out.println("对称密钥校验值:" + symmKey[1]);
                // 测试2。
                String str = "要加密的数据";
                // CBC模式。
                int algType = 1;
                // 密钥类型固定。
                keyType = "00A";
                // 执行加密的对称密钥密文。
                String sm4SymmKey = symmKey[0];
                // 也可以使用加密机内部的索引密钥，使用内部密钥时为int类型。
                String disperFactor = null;
                int sessionType = 0;
                String sessionFactor = null;
                // 遵循强制80填充。
                int padFlag = 1;
                byte[] inData = str.getBytes();
                String IV = "00000000000000000000000000000000";
                // 调用数据加密接口。
                byte[] symmEnc = hsm.generalDataEnc(algType, keyType, sm4SymmKey, disperFac
tor, sessionType, sessionFactor,
                        padFlag, inData, IV);
                System.out.println("16进制字符串输出对称加密结果: " + Forms.byteToHexString(sym
mEnc)
                        + ",如果进行解密，可使用接口功能函数'Forms.hexStringToByte()'将16进制字
符串转换为byte[]参与解密。");
                // 调用数据解密接口。
                byte[] symmDec = hsm.generalDataDec(algType, keyType, sm4SymmKey, disperFac
tor, sessionType, sessionFactor,
                        padFlag, symmEnc, IV);
                System.out.println("解密结果与加密数据比较结果:" + Arrays.equals(symmDec, inDa
ta));
                System.out.println("还原解密结果，通过字符集还原原文:" + new String(symmDec));
                break;
            // 此时生成对称密钥到1号索引位置，若已经存在，会执行覆盖操作。
            case 1:
                str = "要加密的数据";
                // CBC模式。
                algType = 1;
                // 密钥类型固定。
                keyType = "00A";
                // 执行加密的对称密钥密文。
                int sm4SymmKeyIndex = 1;
                // 也可以使用加密机内部的索引密钥，使用内部密钥时为int类型。
                disperFactor = null;
                sessionType = 0;
                sessionFactor = null;
                // 遵循强制80填充。
                padFlag = 1;
```

```
                inData = str.getBytes();
                IV = "00000000000000000000000000000000";
                // 调用数据加密接口。
                symmEnc = hsm.generalDataEnc(algType, keyType, sm4SymmKeyIndex, disperFacto
r, sessionType, sessionFactor,
                        padFlag, inData, IV);
                System.out.println("16进制字符串输出对称加密结果: " + Forms.byteToHexString(sym
mEnc)
                        + ",如果进行解密，可使用接口功能函数'Forms.hexStringToByte()'将16进制字
符串转换为byte[]参与解密。");
                // 调用数据解密接口。
                symmDec = hsm.generalDataDec(algType, keyType, sm4SymmKeyIndex, disperFacto
r, sessionType, sessionFactor,
                        padFlag, symmEnc, IV);
                System.out.println("解密结果与加密数据比较结果:" + Arrays.equals(symmDec, inDa
ta));
                System.out.println("还原解密结果，通过字符集还原原文:" + new String(symmDec));
                break;
            default:
                break;
        }
    }
}
```

1.5. 步骤4：部署密码机实例

在部署密码机实例时，您需要同步应用系统的密钥和配置Java接口。本文介绍了同步应用系统密钥和配置Java接口的具体操作。

背景信息

当您在同步应用系统的密钥时，操作步骤根据应用密钥存储在密码机实例内部还是外部系统是不同的。

- 密钥存储在密码机实例内部

  您需要根据密钥索引将应用密钥通过UKEY备份导出，然后通过密钥恢复导入功能将备份密钥导入到其他密码机实例中，完成密钥同步操作。

  > **说明** 应用密钥由随机产生的备份密钥加密，您可以将加密后的密文以文件的形式导出或存储在UKEY中。同步密钥时，您需要将UKEY插入需要同步的设备中同步密钥。

- 密钥存储在密码机实例外部系统

  当应用密钥经过本地主密钥LMK（Local Master Key）分组加密保护后存储在外部系统，如果需要同步应用系统密钥，您还需要同步密码机实例的域名主密钥DMK（Domain Master Key）。您可以通过密码机实例的原始初始化，来产生DMK成分的UKEY。当多个密码机实例进行密钥备份时，只需要在第一个密码机实例上完成原始初始化后，对其他的密码机实例进行恢复初始化操作，即可完成多个密码机实例的密钥同步。

同步应用系统密钥-密钥存储在密码机实例内部

1. 使用拥有**应用密钥管理**类别授权许可的UKEY登录EVSM。

   具体操作，请参见登录EVSM。

2. 在顶部菜单栏，单击**密钥管理**，然后在**密钥管理**菜单中，单击**备份导出**。

3. 在**选择导出密钥类型和索引**对话框中，根据实际需要选择密钥类型并输入密钥索引，单击**确认**。

您可以选择将密钥备份**保存到文件**或**保存到UKEY内**。本文以将密钥备份**保存到文件**为例进行描述。

4. 按照系统提示依次插入3个空UKEY并输入口令，单击**下一步**。

密码机实例将依次制作出3个密钥备份密钥（KBK）UKEY。

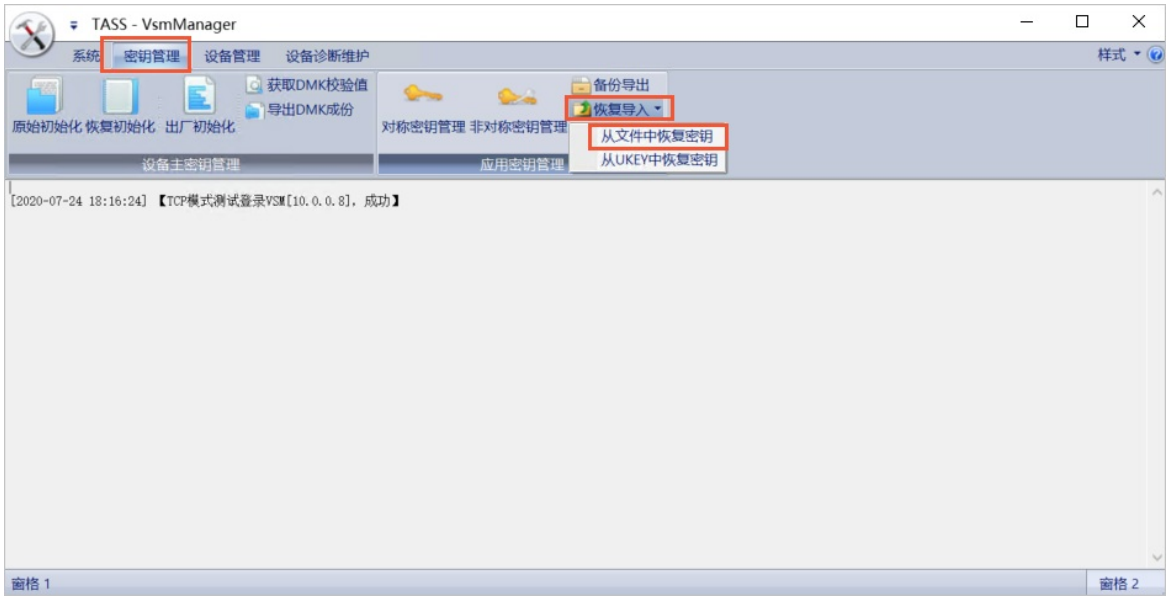
> **说明** 建议3个KBK UKEY由3个密钥管理员分别保管。

5. 选择要保存密钥密文的文件，单击**下一步**。

   EVSM将导出全部应用密钥并保存到您选择的文件中。

> **说明** 密钥备份完成后，请您妥善保管3个KBK UKEY和密钥备份文件，待密钥恢复时使用。

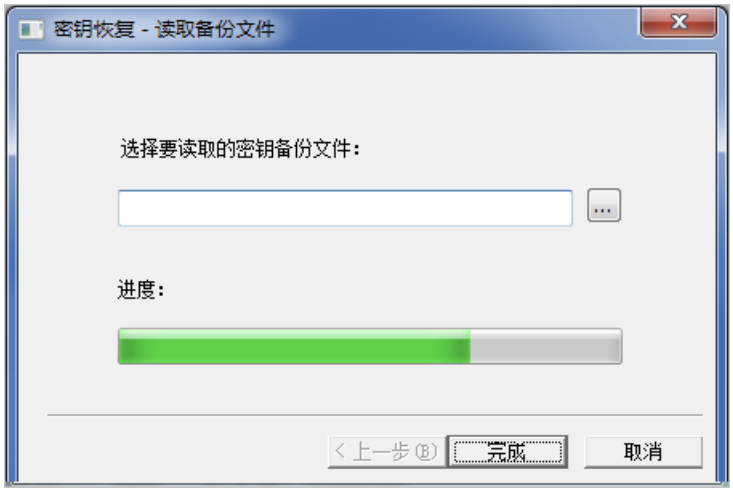
6. 登录EVSM，在顶部菜单栏单击**密钥管理**。
7. 在**密钥管理**页签中，选择**恢复导入 > 从文件中恢复密钥**。



> **说明** 恢复密钥时使用任意2个密钥备份UKEY即可还原出原始的KBK文件，然后将密钥恢复到其他密码机实例内部或者同步到热备份的其他密码机实例内部。

8. 按照系统提示依次插入任意2个KBK UKEY并输入口令，单击**下一步**。

9. 选择要读取的密钥备份文件，等待系统完成应用密钥的恢复，单击**完成**。



同步应用系统密钥-密钥存储在密码机实例外部系统

1. 在第一个密码机实例上进行原始初始化。

   具体操作，请参见步骤1：配置密码机客户端。

2. 对其他密码机实例进行恢复初始化（即导入DMK）。

   i. 登录EVSM，在顶部菜单栏，单击**密钥管理**，在密钥管理菜单中，单击**恢复初始化**。

   ii. 在**安全操作警示**对话框中，单击**下一步**，清除密码机实例内的全部密钥。

iii. 在**恢复初始化-第一步**对话框中，输入DMK成份UKEY数目，单击**下一步**。

iv. 在**恢复初始化-第二步**对话框中，依次插入n个成份UKEY并输入UKEY口令，单击**导入成份UKEY**，密码机实例将读取UKEY内的DMK成份数据。

v. DMK成份导入完成后，单击**合成DMK**。

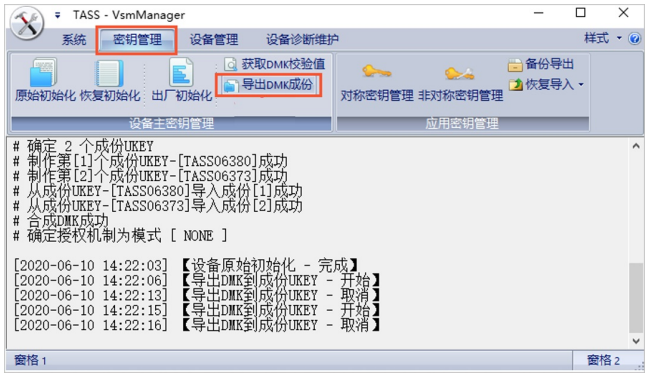
vi. DMK合成成功后，确定授权机制。

- 如果您的其他密码机实例共用一套授权UKEY，请选择**同步授权信息**。您只需插入有效授权的UKEY并输入口令，单击**完成**，完成恢复初始化。
- 如果您的每个密码机实例都需要使用独立的授权UKEY，选中**制作新的授权UKEY**，并从选择授权机制列表中选择请选择**1选1授权控制机制**，制作授权UKEY完成恢复初始化。

3. 在顶部菜单栏，单击**密钥管理**，在密钥管理菜单中，单击**获取DMK校验值**。

当DMK同步到多个密码机实例时，可以通过比对多个密码机实例的DMK校验值来确定同步后的DMK是否一致。

4. 在顶部菜单栏，单击**密钥管理**，在密钥管理菜单中，单击**导出DMK成份**。



您可以将DMK成份导出到多个UKEY中，可防止原有密码机实例的成份UKEY丢失或损坏的情况下，能够重新合成出与原有密码机实例同样的DMK成份。

> **说明** 导出DMK成份功能不能保证DMK成份UKEY中的密钥备份与原有密码机实例的成份UKEY中的密钥备份完全相同。

配置Java接口

您可以通过**文件形式**或者**内容形式**配置Java接口。

- **文件形式**：支持直接将配置文件绝对路径传入初始化接口内。示例：

```
[LOGGER]
logsw=error
logPath=./
[HOST1]
hsmModel=SJJ1310
linkNum=-15
host=192.168.XX.XX
port=8018
timeout=5
[HOST2]
hsmModel=SJJ1310
linkNum=-15
host=192.168.XX.XX
port=8018
timeout=5
```

- **内容形式**：支持直接将配置信息以字符串的形式传入初始化接口内。示例：

```
# 内容形式中的属性字段与文件形式中的属性字段保持一致。
Stringconfig=
"{"
+"[LOGGER];"
+"logsw=error;logPath=./;"
+"[HOST1];"
+"hsmModel=SJJ1310;"
+"host=192.168.XX.XX;"
+"port=8018;"
+"connTimeout=5;"
+"[HOST2];"
+"hsmModel=SJJ1310;"
+"host=192.168.XX.XX;"
+"port=8018;"
+"connTimeout=5;"
+"}";
```

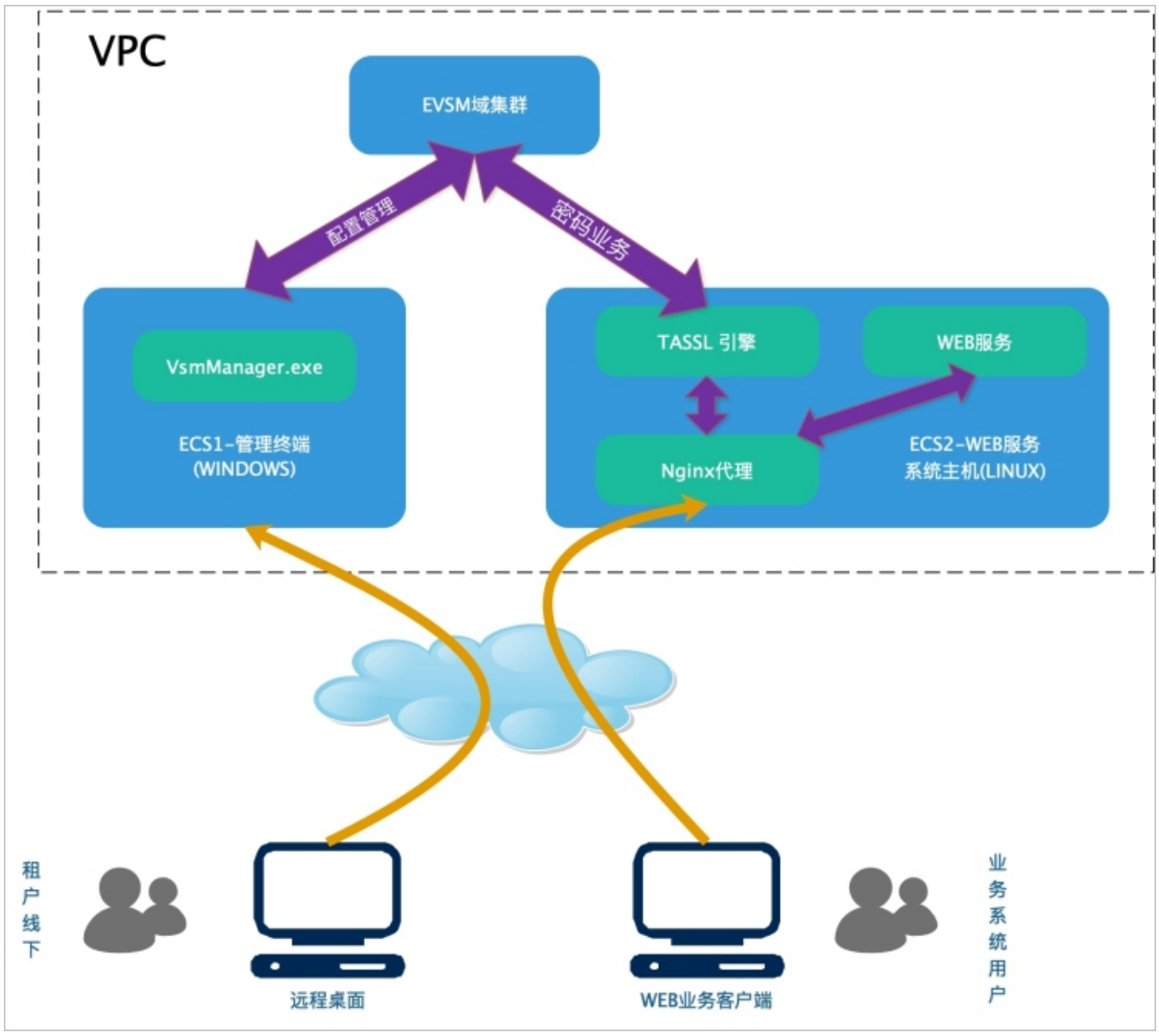
2.基于加密服务的SSL安全卸载

2.1. 概述

本教程提供了在阿里云ECS上借助阿里云加密服务实现SSL卸载的操作说明。

概述

阿里云加密服务全面支持国密算法证书和国密SSL协议，符合监管合规要求。通过使用阿里云加密服务和配套接口TASSL，可以实现SSL卸载；SSL证书私钥通过EVSM产生和存储，可以提升系统的安全性。在阿里云ECS上实现SSL卸载的典型部署方案，请参见下图：



资源准备

您需要准备以下云资源：

云资源类型	规格	说明

云资源类型	规格	说明
ECS1	64位Windows 10系统	对租用的EVSM进行管理配置。 说明 您需要设置ECS1访问EVSM的管理端口为8013。
ECS2	64位Linux系统	用于部署用户的业务系统、TASSL、Nginx。 说明 • 您需要设置ECS2访问EVSM的服务端口为8018。 • 您需要设置ECS2的SSL端口服务。
EVSM	TASS EVSM	用于完成SSL卸载相关的密码运算。

您需要准备的软件资源：

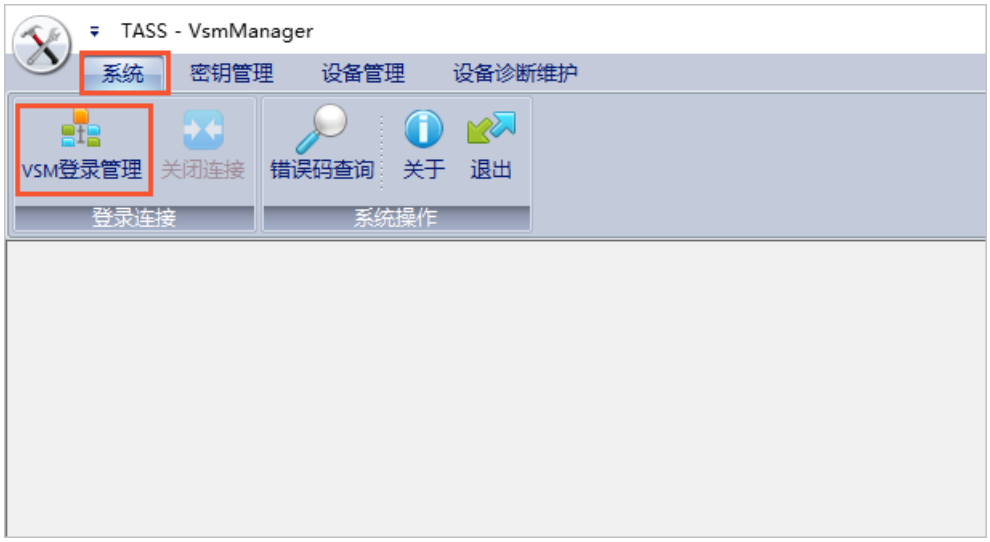
- **VsmManager.exe**：EVSM管理工具，请您提交工单获取该工具。
- **TASSL引擎**：支持调用EVSM的TASSL引擎文件包*tasshsm_engine.tgz*，支持国密算法和国外算法。请您提交工单获取该引擎包。
- **Nginx代理**：配合使用TASSL引擎的Nginx服务包*nginx-1.16.0_tassl_hsm.tgz*。请您提交工单获取该服务包。

2.2. 步骤1：配置EVSM

您需要登录ECS1，安装EVSM客户端，并正确配置EVSM。

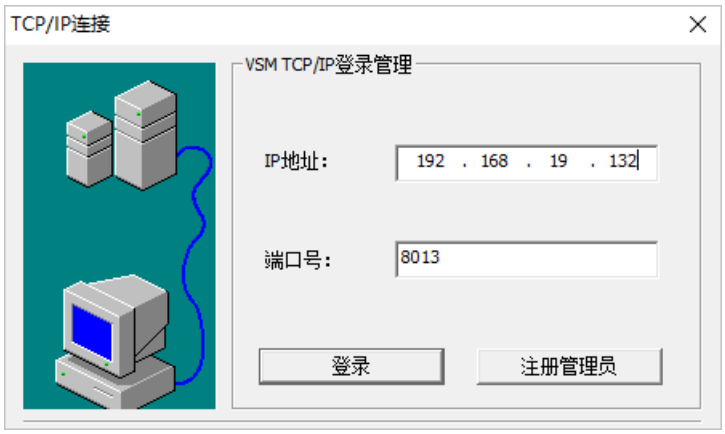
操作步骤

1. 登录ECS1后，打开EVSM客户端，在顶部菜单栏单击**系统 > VSM登录管理**。



2. 在**TCP/IP连接**对话框中，单击**注册管理员**，注册第一个管理员UKEY并使用第一个管理员UKEY登录EVSM。

   具体操作，请参见登录EVSM（有USB KEY）。



3. 在**设备管理**页签，单击**UKEY管理**，至少再添加一个管理员UKEY。

   具体操作，请参见登录EVSM（有USB KEY）。

4. 在**密钥管理**页签，单击**原始初始化**，产生至少2个域名主密钥DMK（Domain Master Key）成份的UKEY。

   > **说明** 建议您采用3选2授权控制机制并制作3个授权UKEY。

5. 在**设备管理**页签，单击**主机端口属性**。

   您需要配置消息报文头长度为`0`、消息报文编码格式为`ASCII`、主机服务通讯方式为`明文`。

6. 在**设备管理**页签，单击**操作授权**，为主机服务的密钥管理类别授权。

2.3. 步骤2：部署TASSL

本文档主要介绍部署调用EVSM的TASSL引擎文件包的方法。

操作步骤

1. 请您将支持调用EVSM的TASSL引擎文件包*tasshsm_engine.tgz*上传至ECS2并解压，例如解压到*/root/t*

*asshsm_engine*目录下。

2. 配置TASSL引擎要访问的EVSM信息。
   - 您如果使用的是RSA算法，将*/root/tasshsm_engine/cfg/tasshsm_rsa_engine.ini*文件中的 `IP` 和 `PORT` 修改为EVSM的IP地址和主机服务端口号。

   - 您如果使用的是国密SM2算法，将*/root/tasshsm_engine/cfg/tasshsm_sm2_engine.ini*文件中的 `IP` 和 `PORT` 修改为EVSM的IP地址和主机服务端口号。

   - 您如果使用的是ECC算法，将*/root/tasshsm_engine/cfg/tasshsm_ecc_engine.ini*文件中的 `IP` 和 `PORT` 修改为EVSM的IP地址和主机服务端口号。

2.4. 步骤3：申请和签发证书

您可以根据需要生成RSA算法、SM2算法、ECC算法服务器的证书申请文件并签发证书。在试用阶段建议采用自签发证书，生产阶段建议从CA中心签发合格的服务器证书，或者通过阿里云证书服务去签发证书。

RSA算法服务端证书（单证）申请和签发

1. 生成证书申请文件。
   - 方式一：通过EVSM的管理工具生成。
     a. 登录EVSM，在**密钥管理**页签中单击**非对称密钥管理**。

     b. 在**非对称密钥管理**对话框中，单击**产生新密钥**。
     c. 在**产生非对称密钥**对话框中，配置**算法标识**为 `RSA` 、**密钥模长**为 `2048` 、**幂指数**为 `65537` 。

d. 单击**产生**。

EVSM将产生新的非对称密钥并输出显示公钥明文和私钥密文。

e. 在**非对称密钥管理**对话框中，单击**生成RSA请求**。

f. 在**生成RSA**对话框中，输入合法的**主题**，选择**是否使用内部索引**，并输入**密钥索引**，单击**确定**。

- 方式二：通过Tassl配套脚本生成。

进入ECS2的*/root/tasshsm_engine/cert/server/rsa*目录，设置环境变量*source ./setting*，产生证书申请文件*S_RSA_HSM.csr*。

```
[root@localhost rsa]#./gen_rsaf_csr_with_hsm -r S_RSA_HSM.csr
请输入DN：/C=CN/ST=BJ/L=HaiDian/O=Beijing
      JNTA Technology LTD./OU=BSRC of TASS/CN=rsa_commoname/
请输入密钥模长[1024 - 2048]:2048
请选择摘要算法:1)SHA1
               2)SHA224
               3)SHA256
               4)SHA384
               5)SHA512
请输入:3
请输入加密机存储私钥的索引号：15
```

2. 签发证书。

- 在试用阶段，如果您采用自签发证书，自签发证书通过Tassl配套脚本产生。以下步骤为自签发证书的签发过程：

a. 进入ECS2的*/root/tasshsm_engine/cert/server/rsa*目录。

b. 设置环境变量*source ./setting*。

c. 签发*S_RSA_HSM.crt*。

```
./sign_cert.sh S_RSA_HSM.csr S_RSA_HSM.crt
```

- 如果您在生产阶段，建议从CA中心签发合格的服务器证书，或者登录阿里云SSL证书控制台去签发证书。

国密SM2算法的服务端证书（双证）申请和签发

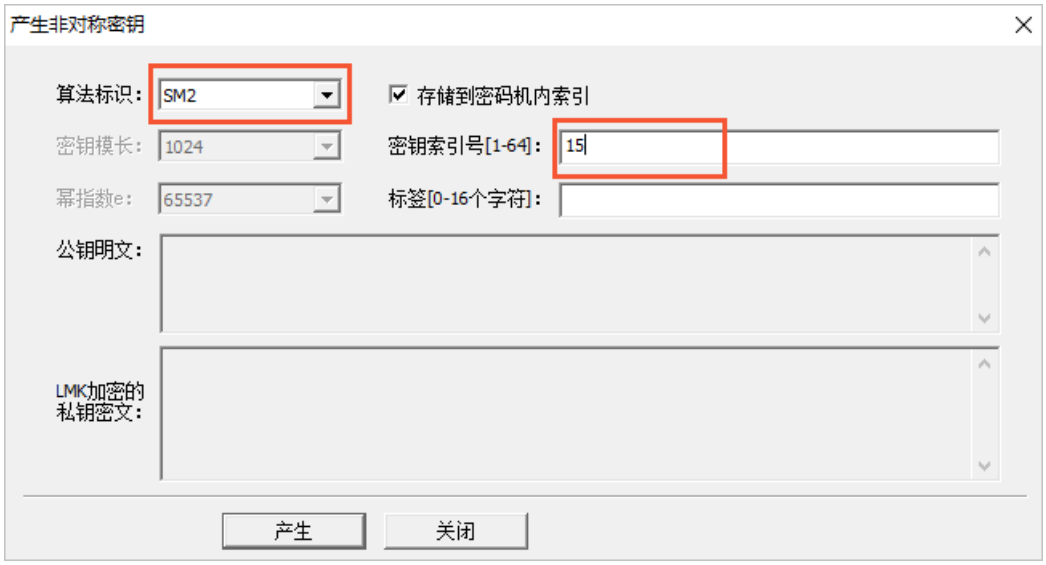
1. 生成证书申请文件 。

- 方式一：通过EVSM的管理工具生成。

a. 登录EVSM，在**密钥管理**菜单中单击**非对称密钥管理**。

b. 在**非对称密钥管理**对话框中，单击**产生新密钥**。

c. 在**产生非对称密钥**对话框中，配置**算法标识**为 `SM2` 、**密钥索引号**为 `15` 。



d. 单击**产生**，EVSM将产生新的非对称密钥并输出显示公钥明文和私钥密文。

e. 在**非对称密钥管理**对话框中，单击**生成ECC请求**。

f. 在**生成ECC**对话框中，选择**算法标识**、**主题标识**，输入**主题**、**密钥索引**，单击**确定**。

- 方式二：通过Tassl配套脚本产生。

  您可以在ECS2的*/root/tasshsm_engine/cert/server/sm2*目录下，设置环境变量*source ./setting*，产生证书申请文件*SS_SM2_HSM.csr*和加密证书申请文件*SE_SM2_HSM.csr*。

```
#产生证书申请文件SS_SM2_HSM.csr
[root@localhost rsa]# ./gen_sm2_csr_with_hsm -r SS_SM2_HSM.csr
请输入DN：/C=CN/ST=BJ/L=HaiDian/O=Beijing
JNTA Technology LTD./OU=BSRC of TASS/CN=sm2_commoname/
请输入加密机存储私钥的索引号：15
#产生加密证书申请文件SE_SM2_HSM.csr
[root@localhost rsa]# ./gen_sm2_csr_with_hsm -r SE_SM2_HSM.csr
请输入DN：/C=CN/ST=BJ/L=HaiDian/O=Beijing
JNTA Technology LTD./OU=BSRC of TASS/CN=sm2_commoname/
请输入加密机存储私钥的索引号：16
```

2. 签发证书。
   - 如果您在试用阶段可以采用自签发证书，自签发证书通过Tassl配套脚本产生。以下步骤为自签发证书的签发过程：
     a. 进入ECS2的*/root/tasshsm_engine/cert/server/sm2*目录。
     b. 设置环境变量*source ./setting*。
     c. 签发签名证书文件*SS_SM2_HSM.crt*和加密证书文件*SE_SM2_HSM.crt*。

     ```
     #签发签名证书文件SS_SM2_HSM.cr
     ./sign_cert_s.sh SS_SM2_HSM.csr SS_SM2_HSM.crt
     #签发加密证书文件SE_SM2_HSM.crt
     ./sign_cert_e.sh SE_SM2_HSM.csr SE_SM2_HSM.crt
     ```

   - 如果您在生产阶段建议从CA中心签发合格的服务器证书，或者登录阿里云SSL证书控制台去签发证书。

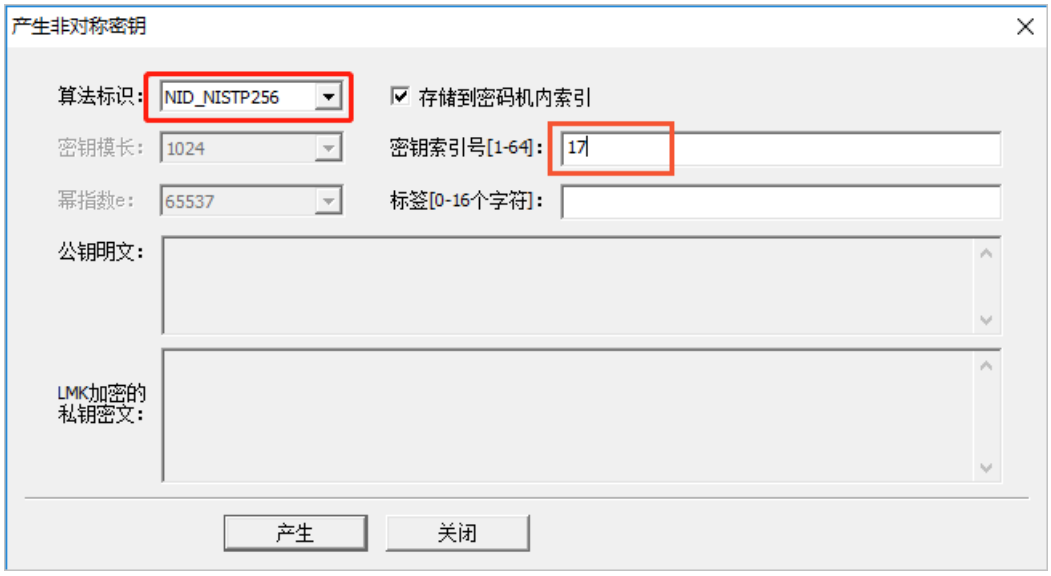
ECC算法的服务端证书申请和签发

1. 生成证书申请文件。
   - 方式一：通过EVSM的管理工具生成。
     a. 登录EVSM，在**密钥管理**菜单中单击**非对称密钥管理**。

     b. 在**非对称密钥管理**对话框中，单击**产生新密钥**。

c. 在**产生非对称密钥**对话框中，配置**算法标识**为 `NID_NISTP256` 、**密钥索引号**为 `17` 。



d. 单击**产生**，EVSM将产生新的非对称密钥并输出显示公钥明文和私钥密文。

e. 在**非对称密钥管理**对话框中，选中产生的ECC曲线密钥单击**生成ECC请求**。

f. 在**生成ECC**对话框中，选择**算法标识**，输入**主题**、**密钥索引**，单击**确定**。

- 方式二：通过Tassl配套脚本产生。

  进入ECS2的*/root/tasshsm_engine/cert/server/ecc*目录，设置环境变量*source ./setting*，产生证书申请文件*test_ecc.csr*。

```
[root@localhost rsa]# ./gen_ecc_csr_with_hsm -r test_ecc.csr
请输入DN：/C=CNST=BJ/L=HaiDian/O=Beijing
JNTA Technology LTD./OU=BSRC of TASS/CN=ecc_commoname/
请选择摘要算法:1)SHA1
               2)SHA224
               3)SHA256
               4)SHA384
               5)SHA512
请输入:3
请输入加密卡存储私钥的索引号(0代表不保存，并且输出加密私钥)： 17
```

2. 签发证书 。

- 如果您在试用阶段可以采用自签发证书，自签发证书通过Tassl配套脚本产生。以下步骤为自签发证书的签发过程：

  a. 进入ECS2的*/root/tasshsm_engine/cert/server/ecc*目录。

  b. 设置环境变量*source ./setting*。

  c. 签发证书文件*test_ecc.crt*。

```
./sign_cert.sh test_ecc.csr test_ecc.crt
```

- 如果您在生产阶段建议从CA中心签发合格的服务器证书，或者登录阿里云SSL证书控制台去签发证书。

2.5. 步骤4：部署Nginx服务

本文档主要介绍如何在您的ECS上部署Nginx服务，以及如何在Nginx服务上配置证书文件。

操作步骤

1. 请您将配合使用TASSL引擎的Nginx服务包*nginx-1.16.0_tassl_hsm.tgz*上传至ECS2上并解压，例如解压到*/root/ nginx-1.16.0_tassl*目录下。
2. 在*/root/ nginx-1.16.0_tassl*目录下安装Nginx服务。

```
./configure --with-http_ssl_module --with-stream
--with-stream_ssl_module --with-openssl=/root/tasshsm_engine/tassl
--prefix=/root/nginx
make
make install
```

3. 配置不同加密算法服务端证书的Nginx服务。

   您可以参考下表中的示例编辑*/root/nginx/conf/nginx.conf*配置文件中的证书部分。

算法	代码示例

RSA	`user root;`
	`worker_processes  1;`
	`…`
	`…`
	`    # HTTPS server`
	`    server {`
	`        listen       443 ssl;`
	`        server_name  localhost;`
	`        #use tasshsm engine by key index`
	`        ssl_certificate /root/tasshsm_engine/cert/server/rsa/S_RSA_HSM.crt;  #配置RSA证书文件。`
	`        ssl_certificate_key  engine:tasshsm_rsa:15; #配置存储RSA私钥的索引号。`
	`        ssl_verify_client off; # for one-way https #表示不验证客户端。`
	`        ssl_session_cache    shared:SSL:1m;`
	`        ssl_session_timeout  5m;`
	`        ssl_ciphers  HIGH:!aNULL:!MD5;`
	`        ssl_prefer_server_ciphers  on;`
	`        location / {`
	`            root   html;`
	`            index  index.html index.htm;`
	`        }`
	`}`
	`…`

算法	代码示例

SM2	user
	  root;
	worker_processes 1;
	…
	…
	    # HTTPS server
	    server {
	        listen 444 ssl;
	        server_name localhost;
	        #use tasshsm engine by key index
	        ssl_certificate
	/root/tasshsm_engine/cert/server/sm2/SS_SM2_HSM.crt; #配置签名证书文件。
	        ssl_certificate_key engine:tasshsm_sm2:15; #配置存储签名私钥的索引号。
	        ssl_enc_certificate
	/root/tasshsm_engine/cert/server/sm2/SE_SM2_HSM.crt; #配置加密证书文件。
	        ssl_enc_certificate_key engine:tasshsm_sm2:16; #配置存储加密私钥的索引号。
	        ssl_verify_client off; # for #表示不验证客户端。
	  one-way https
	        …
	        location / {
	            root html;
	            index index.html index.htm;
	        }
	}
	…

算法	代码示例

ECC	user
	  root;
	worker_processes 1;
	…
	…
	    # HTTPS server
	    server {
	        listen 445 ssl;
	        server_name localhost;
	        #use tasshsm engine by key index
	        ssl_certificate
	/root/tasshsm_engine/cert/server/ecc/S_ECC_HSM.crt; #配置RSA证书文件。
	        ssl_certificate_key engine:tasshsm_ecc:15; #配置存储RSA私钥的索引号。
	        ssl_verify_client off; # for #表示不验证客户端。
	  one-way https
	        ssl_session_cache shared:SSL:1m;
	        ssl_session_timeout 5m;
	        ssl_ciphers HIGH:!aNULL:!MD5;
	        ssl_prefer_server_ciphers on;
	        location / {
	            root html;
	            index index.html index.htm;
	        }
	}
	…

4. 执行以下命令启动Nginx代理。

```
cd /root/nginx/sbin
source ./setting  #设置Nginx调用ssl的动态库为tassl引擎目录。
./nginx
```

2.6. 步骤5：测试验证

本文档主要介绍了RSA算法和国密SM2算法服务端证书的验证方法。

- 如果您使用的是RSA算法服务端证书（单证），可以通过浏览器直接访问您申请的域名URL进行验证。

- 如果您使用的是国密SM2算法的服务端证书（双证），需要按照以下操作进行验证：
  i. 在客户端安装支持国密算法的浏览器（如360国密浏览器），将签发给服务端的CA证书导入到国密浏览器中，并设置为信任。
  ii. 拷贝ECS2中证书文件*/root/tasshsm_engine/cert/server/sm2/ca.crt*的内容到*C:\Users\Administrator\AppData\Roaming\360se6\User Data\Default\gmssl\ctl.dat*文件中。
  iii. 重启国密浏览器。
  iv. 在*C:\Windows\System32\drivers\etc*路径下修改hosts文件，将服务器地址指向您的测试域名。
  v. 通过浏览器访问您的测试域名进行验证。