# DSVS 客户端组件 API 说明

# （Java 版）

二〇一六年 十一月

数字认证 | 安信天行

北京市海淀区北四环西路 68 号双桥大厦 15 层

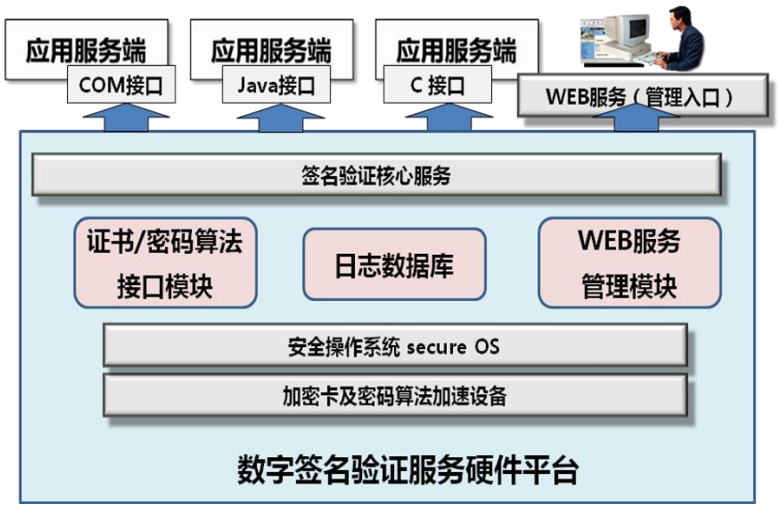
**TEL**：86-10-58045600 **FAX**：86-10-58045678

**邮政编码**：100080

# 目录

# 1 组件介绍

数字签名验证服务器主要包括签名验证核心服务模块和安全管理模块。签名验证核心服务通过应用端部署的 API，接收应用端发送的签名服务请求，并返回签名或验证服务结果。安全管理以 B/S 方式提供，管理员可以通过 WEB 浏览器直接对各种证书服务和系统进行集中管理和配置。主要结构图如下：

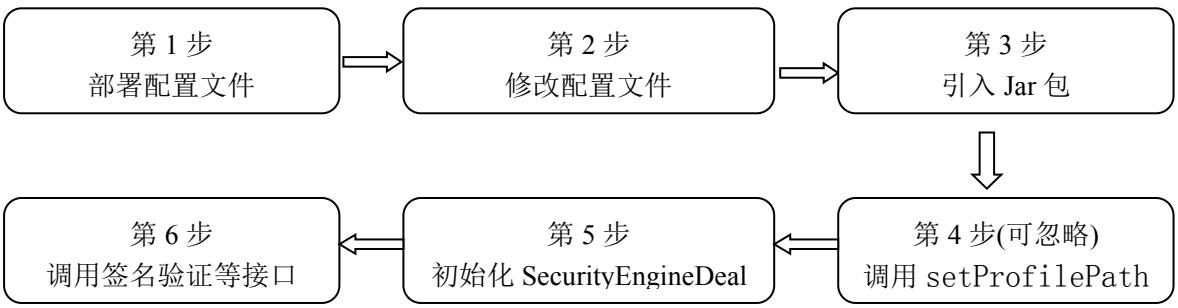



主要的使用方式为业务系统通过调用客户端组件，向数字签名验证服务器发送请求，数字签名验证服务器将请求处理结果再返还给客户端组件，客户端组件将结果返回给业务系统，目前支持的客户端组件包括 COM 组件、C 组件、Java 组件等，本文档主要介绍 Java 组件接口使用，其他类型客户端组件请参考相应的文档。

# 2 组件使用说明

## 2.1 使用流程说明

### 2.1.1 通过配置文件设置客户端参数



1） 将配置文件中的 SVSClient.properties 放在客户端机器当前用户目录的 BJCAROOT 中，比如 WindowsXP 平台下 Administrator 用户为 C:\Documents and Settings\Administrator\BJCAROOT ； WIN7 平台下 Administrator 用户为 C:\Users\Administrator\BJCAROOT；Linux 平台下 root 用户为/root/BJCAROOT。

2） 根据 DSVS 服务器的配置，修改 SVSClient.properties 文件中的 address1 和 port1（默认是 8000）。如双机并行部署，请将另一台 DSVS 的 IP 及端口号填写在“address2”和“port2”的对应项中，并修改服务器个数，若需要设置长短链接可根据配置文件说明修改。

3） 将 java 组件中的 SVSClient.jar、BJCA_LOG.jar 2 个 jar，引入到应用开发的 lib 加载。

4） 编写代码调用 setProfilePath()接口，设置客户端配置文件。

5） 实例化 SecurityEngineDeal，调用 getInstance()接口。

6） 调用 signData()等接口。

## 2.2 设置根路径

功能：

设置 SVS 客户端配置文件和日志文件路径（一个应用服务中调用一次即可，默认路径是当前用户目录 BJCAROOT 文件夹下，比如 WindowsXP 平台下 Administrator 用户为 C:\Documents and Settings\Administrator\BJCAROOT； WIN7 平台下 Administrator 用户为 C:\Users\Administrator\BJCAROOT；Linux 平台下 root 用户为/root/BJCAROOT）。

函数定义：

```
public static void setProfilePath(String rootPath);
```

参数：

- rootPath：[IN] 路径（如：c:/）

返回： 无

调用过程：`SecurityEngineDeal.setProfilePath("C:\\BJCAROOT");`

## 2.3 初始化 SecurityEngineDeal 实例

功能：

获得一个对象实例，初始化对象。

函数定义：

```
public static SecurityEngineDeal getInstance(String appName);
```

参数：

- appName：[IN] 应用名

返回：

SecurityEngineDeal 对象

抛出：

SVSConnectException： 连接服务端异常

ApplicationNotFoundException： 此应用名不存在

InitException： 初始化异常

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

# 3 推荐接口列表

## 3.1 数据签名

功能：

对二进制数据进行数字签名，签名格式为 Pkcs1，签名算法的选择(RSA 或者 SM2)取决于所配置的应用的服务器证书类型。

函数定义：

```
String signData(byte[] inData);
```

参数：

- inData：[IN] 待签名的数据原文。

返回值：

成功返回 pkcs1 格式的签名值的 base64 编码。失败返回空值（NULL）

抛出：

SVSConnectException 连接服务端异常

ParameterTooLongException 参数过长

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    byte[] data = "test".getBytes();
    String signedValue = sed.signData(data);
    System.out.println(signedValue);
```

```
        } catch (SVSConnectException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ApplicationNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InitException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ParameterTooLongException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

## 3.2 验证数据签名

功能：

验证数字签名，格式为 Pkcs1

函数定义：

```
boolean verifySignedData(String base64EncodeCert, byte[] inData, byte[] signValue);
```

参数：

- base64EncodeCert：[IN] base64 编码的签名证书。
- inData：[IN] 待验证的原文。
- signValue：[IN] 签名值。

返回值：

true：验证成功。false：验证失败。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterInvalidException | 参数无效 |
| UnkownException | 未知异常 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    byte[] data = "test".getBytes();
    String signedValue = sed.signData(data);
    byte[] signedValueByte = sed.base64Decode(signedValue);
    String cert = sed.getServerCertificate();
    boolean verifyRes = sed.verifySignedData(cert, data,
signedValueByte);
    System.out.println(verifyRes);
```

```
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterInvalidException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (UnkownException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 3.3 PKCS7 签名（带原文）

功能：

对二进制数据进行数字签名，签名格式为 Pkcs7（带原文）。

函数定义：

```
String signDataByP7Attach (byte[] inData);
```

参数：

- inData：[IN] 待签名的数据原文。

返回值：

成功返回 pkcs7 格式的签名值的 base64 编码（带原文）。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    byte[] data = "test".getBytes();
    String signedValue = sed.signDataByP7Attach(data);
    System.out.println(signedValue);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
```

```
        } catch (ApplicationNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InitException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ParameterTooLongException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

## 3.4 验证 PKCS7 签名（带原文）

功能：

验证数字签名，格式为 PKCS7（带原文）

函数定义：

```
boolean verifySignedDataByP7Attach (String  pkcs7SignData);
```

参数：

- pkcs7SignData：[IN] p7 签名值（带原文）。

返回：

成功返回 true，否则返回 false。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
    SecurityEngineDeal sed = null;
    try {
        sed = SecurityEngineDeal.getInstance("SVSDefault");
        String data = "test";
        String signedValue = sed.signDataByP7Attach(data);
        boolean verifyRes =
sed.verifySignedDataByP7Attach(signedValue);
        System.out.println(verifyRes);
    } catch (SVSConnectException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ApplicationNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (InitException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterTooLongException e) {
```

```
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterInvalidException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

## 3.5 PKCS7 签名（不带原文）

功能：

对二进制数据进行数字签名，签名格式为 Pkcs7（不带原文）

函数定义：

```
String signDataByP7Detach(byte[] inData);
```

参数：

- inData：[IN] 待签名的数据原文。

返回值：

成功返回 pkcs7 格式的签名值的 base64 编码（不带原文）。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    byte[] inData = "abc".getBytes();
    String signedValue = sed.signDataByP7Detach(inData);
    System.out.println(signedValue);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 3.6 验证 PKCS7 签名（不带原文）

功能：

验证数字签名，格式为 PKCS7（不带原文）,本接口验证证书有效性

函数定义：

```
String verifySignedDataByP7Detach(byte[] originalData,byte[] pkcs7SignData);
```

参数：

- originalData：[IN] 原文
- pkcs7SignData：[IN]  p7 签名值（不带原文）。

返回：

证书（BASE64） 为验证通过， -1 为不是所信任的根，-2 为超过有效期，-3 为作废证书，-4 已加入黑名单, -5 证书未生效,  0 验证 P7 签名不通过

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    byte[] inData = "abc".getBytes();
    String signedValue = sed.signDataByP7Detach(inData);
   System.out.println(sed.verifySignedDataByP7Detach(inData,
signedValue));
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

# 3.7 获取 PKCS7 签名值的签名信息

功能：

解析 p7 签名包的信息，可获得原文、签名值、签名证书等信息

函数定义：

```
byte[] getP7SignatureInfo (byte[] pkcs7SignedData, int type)
```

参数：

- pkcs7SignedData：[IN] p7 签名包。
- type：[IN] 类型 1：原文 2：签名者证书 3：签名值。

返回值：

成功返回 type 对应的值，失败返回 null。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterOutRangeException | 参数超出范围 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
SecurityEngineDeal sed = null;
```

```
    try {
        sed = SecurityEngineDeal.getInstance("SVSDefault");
        String data = "test";
        String signedValue = sed.signDataByP7Attach(data);
        byte[] info =
sed.getP7SignatureInfo(sed.base64Decode(signedValue), 1);
        System.out.println(new String(info));
    } catch (SVSConnectException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ApplicationNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (InitException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterTooLongException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterOutRangeException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterInvalidException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

## 3.8 文件签名

功能：

对文件数字签名。得到 base64 编码后的 Pkcs1 格式的签名数据。

函数定义：

```
String  signFile(String  inFile);
```

参数：

- inFile：[IN] 待签名的文件路径。

返回值：

base64 编码后的 Pkcs1 格式的签名数据。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
```

```
        sed = SecurityEngineDeal.getInstance("SVSDefault");
        String inFile = "C:\\temp\\test.txt";
        String signedValue = sed.signFile(inFile);
        System.out.println(signedValue);
    } catch (SVSConnectException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ApplicationNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (InitException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

## 3.9 验证文件签名

功能：

验证文件数字签名。

函数定义：

```
boolean verifySignedFile(String base64EncodeCert, String inFile, String signValue);
```

参数：

- base64EncodeCert：[IN] base64 编码的签名证书。
- inFile：[IN] 待验证的原文。
- signValue：[IN] 签名值。

返回值：

true：验证成功。false：验证失败。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterInvalidException | 参数无效 |
| UnkownException | 未知异常 |

调用过程：

```
    SecurityEngineDeal sed = null;
    try {
        sed = SecurityEngineDeal.getInstance("SVSDefault");
        String inFile = "C:\\temp\\test.txt";
        String signedValue = sed.signFile(inFile);
        String cert = sed.getServerCertificate();
        boolean verifyRes = sed.verifySignedFile(cert, inFile,
signedValue);
        System.out.println(verifyRes);
    } catch (SVSConnectException e) {
```

```
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ApplicationNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (InitException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterTooLongException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterInvalidException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (UnkownException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

## 3.10 获取服务器证书

功能：

获取当前应用的服务器证书。

函数定义：

```
String getServerCertificate ();
```

参数：无

返回值：

Base64 编码的服务器证书。出错则返回空值。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String serverCert = sed.getServerCertificate();
    System.out.println(serverCert);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
```

```
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

## 3.11 获取证书基本信息

功能：

获取证书基本信息。

函数定义：

```
String getCertInfo(String base64EncodeCert, int type)
```

参数：

- base64EncodeCert：[IN] base64 编码的证书。
- type：[IN] 证书信息类型。

  1 证书版本
  2 证书序列号
  8 证书发放者通用名
  11 证书有效期起始
  12 证书有效期截止
  13 用户国家名
  14 用户组织名
  15 用户部门名
  16 用户省州名
  17 用户通用名
  18 用户城市名
  19 用户 EMAIL 地址
  20 证书颁发者DN
  21 证书主题（DN）
  23 用户国家名(备用名C)
  24 用户组织名(备用名O)
  25 用户部门名(备用名OU)
  26 用户省州名(备用名S)
  27 用户通用名(备用名CN)
  28 用户城市名(备用名L)
  29 用户EMAIL地址(备用名E)
  30 证书公钥（base64）
  31 证书密钥类型（返回RSA or ECC）

返回值：

证书信息。出错返回 NULL。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterOutRangeException | 参数超出范围 |
| ParameterInvalidException | 无效参数 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String cert = sed.getServerCertificate();
    String info = sed.getCertInfo(cert, 2);
    System.out.println(info);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterOutRangeException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterInvalidException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 3.12 获取证书扩展信息

功能：

根据 OID 获取证书私有扩展项信息。

函数定义：

```
String getCertInfoByOid(String base64EncodeCert, String oid)
```

参数：

- base64EncodeCert：[IN] base64 编码的证书。
- oid：[IN] 私有扩展对象 ID，比如“1.2.18.21.88.2”。

返回值：

证书 OID 对应的值。没有此扩展项或出错返回 NULL。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterOutRangeException | 参数超出范围 |

ParameterInvalidException　　无效参数

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String cert = sed.getServerCertificate();
    String info = sed.getCertInfoByOid(cert, "2.16.840.1.113732.2");
    System.out.println(info);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterOutRangeException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterInvalidException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 3.13　验证证书

功能：

根据应用的策略验证证书有效性。

函数定义：

```
int validateCert(String base64EncodeCert)
```

参数：

- base64EncodeCert：[IN] 待验证的 base64 编码证书。

返回值：int

证书有效返回 1,证书无效情况下： -1 为不是所信任的根，-2 为超过有效期，-3 为作废证书，-4 已加入黑名单，-5 证书未生效， 0 未知错误

抛出：

SVSConnectException　　连接服务端异常

ParameterTooLongException　　参数过长

ParameterOutRangeException　　参数超出范围

ParameterInvalidException　　无效参数

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String cert = sed.getServerCertificate();
    int validRes = sed.validateCert(cert);
    System.out.println(validRes);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterOutRangeException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterInvalidException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 3.14　产生随机数

功能：

产生指定长度的随机数。

函数定义：

```
String genRandom(int len)
```

参数：

- len：[IN] 待产生的随机数长度（bytes，字节长度）。

返回值：

随机数值（Base64 编码后的）。

抛出：

SVSConnectException　　连接服务端异常

ParameterOutRangeException　　参数超出范围

调用过程：

```
SecurityEngineDeal sed = null;
```

```
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String random = sed.genRandom(24);
    System.out.println(random);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterOutRangeException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 3.15 根据指定算法对称加密

功能：

根据指定对称加密。

函数定义：

```
String symmEncryptData(byte[] key, byte[] inData,int encryptType)
```

参数：

- key：[IN] 加密密钥（Base64 编码后的密钥），算法为 BJCA_Type.BCA_ALGO_SM4_CBC 时，密钥长度需要是 16 位，算法为 BJCA_Type.BCA_ALGO_3DES_3KEY 时，密钥长度需要是 24 位
- inData: [IN] 待加密的明文
- encryptType: [IN] 对称算法，目前可选值为 BJCA_Type.BCA_ALGO_SM4_CBC 和 BJCA_Type.BCA_ALGO_3DES_3KEY

返回值：

成功返回加密后的密文（Base64 编码后的）。出错返回空值。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterOutRangeException | 参数超出范围 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String key = sed.genRandom(24);
    String data = "test";
    byte[] encData =
```

```
sed.symmEncryptData(sed.base64Decode(key), sed.base64Decode(data),
BJCA_Type.BCA_ALGO_3DES_3KEY);
                System.out.println(sed.base64Encode(encData));
            } catch (SVSConnectException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (ApplicationNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InitException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (ParameterOutRangeException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (ParameterTooLongException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
```

## 3.16 根据指定算法对称解密

功能：

根据指定算法对称解密。

函数定义：

```
byte[] symmDecryptData(byte[] key, byte[] inData, int encryptType)
```

参数：

- key：[IN] 解密密钥（Base64 编码后的密钥），算法为 BJCA_Type.BCA_ALGO_SM4_CBC 时，密钥长度需要是 16 位，算法为 BJCA_Type.BCA_ALGO_3DES_3KEY 时，密钥长度需要是 24 位
- inData: [IN] 待解密的明文
- encryptType: [IN] 对称算法，目前可选值为 BJCA_Type.BCA_ALGO_SM4_CBC 和 BJCA_Type.BCA_ALGO_3DES_3KEY

返回值：

解密后的明文。出错返回空值。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterOutRangeException | 参数超出范围 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String key = sed.genRandom(24);
```

```
                byte[] data = "test".getBytes();
                byte[] encData =
sed.symmEncryptData(sed.base64Decode(key), data,
BJCA_Type.BCA_ALGO_3DES_3KEY);
                byte[] decData =
sed.symmDecryptData(sed.base64Decode(key), encData,
BJCA_Type.BCA_ALGO_3DES_3KEY);
                String decDataStr = new String(decData);
                System.out.println(decDataStr);
            } catch (SVSConnectException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (ApplicationNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InitException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (ParameterOutRangeException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (ParameterTooLongException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
```

## 3.17 非对称加密

功能：

使用证书公钥对数据加密。

函数定义：

```
Byte[] pubKeyEncrypt(String base64EncodeCert, byte[] inData);
```

参数：

- 输入参数 ：base64EncodeCert BASE64 编码证书。
- 输入参数 ：inData 待加密的数据（所能加密的最大数据长度和使用的密钥长度有关，如 RSA1024 则最大长度为 117）。

返回：

成功返回加密后的密文。否则返回空。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
SecurityEngineDeal sed = null;
```

```
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    byte[] data = "test".getBytes();
    String cert = sed.getServerCertificate();
    byte[] encData = sed.pubKeyEncrypt(cert, data);
    String encDataStr = sed.base64Encode(encData);
    System.out.println(encDataStr);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterInvalidException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 3.18 非对称解密

功能：

私钥解密（Pkcs1 格式）。

函数定义：

```
Byte[] priKeyDecrypt (byte[] inData);
```

参数：

- 输入参数 ：inData：待解密的数据。

返回：

成功返回解密后的明文。否则返回空。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    byte[] data = "test".getBytes();
    String cert = sed.getServerCertificate();
    byte[] encData = sed.pubKeyEncrypt(cert, data);
```

```
            byte[] decData = sed.priKeyDecrypt(encData);
            String decDataStr = new String(decData);
            System.out.println(decDataStr);
        } catch (SVSConnectException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ApplicationNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InitException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ParameterTooLongException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ParameterInvalidException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

## 3.19 数字信封加密

功能：

数字信封加密，RSA 算法符合 PKCS7 标准规范，国产算法符合消息语法规范。

函数定义：

```
Byte[] encodeEnvelopedData (String base64EncodeCert, byte[] inData);
```

参数：

- 输入参数 ：base64EncodeCert BASE64 编码证书。
- 输入参数 ：inData 待加密的数据。

返回：

成功返回加密后的密文(base64 编码后的)。否则返回空。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    byte[] data = "test".getBytes();
    String cert = sed.getServerCertificate();
    byte[] encData = sed.encodeEnvelopedData(cert, data);
    String encDataStr = sed.base64Encode(encData);
    System.out.println(encDataStr);
```

```
    } catch (SVSConnectException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ApplicationNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (InitException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterTooLongException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterInvalidException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

## 3.20 解密数字信封

功能：

解密数字信封，RSA 算法符合 PKCS7 标准规范，国产算法符合消息语法规范。

函数定义：

```
byte[] decodeEnvelopedData (byte[] inData);
```

参数：

- 输入参数 ：inData：待解密的数据。

返回：

成功返回解密后的明文。否则返回空。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |

调用过程：

```
    SecurityEngineDeal sed = null;
    try {
        sed = SecurityEngineDeal.getInstance("SVSDefault");
        byte[] data = "test".getBytes();
        String cert = sed.getServerCertificate();
        byte[] encData = sed.encodeEnvelopedData(cert, data);
        byte[] decData = sed.decodeEnvelopedData(encData);
        String decDataStr = new String(decData);
        System.out.println(decDataStr);
    } catch (SVSConnectException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
```

```
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterInvalidException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 3.21 BASE64 编码

功能：

对数据进行 base64 解码。

函数定义：

```
String base64Encode (byte[]  inData);
```

参数：

- 输入参数 ：byte[]  inData：原文。

返回：

编码后的数据。

抛出：

无。

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    byte[] inData = "test".getBytes();
    String encData = sed.base64Encode(inData);
    System.out.println(encData);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 3.22 BASE64 解码

功能：

对数据进行 base64 解码。

函数定义：

String base64Decode (String inData);

参数：

- 输入参数 ：String inData：待解码的数据。

返回：

解码后的原文。

抛出：

无。

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    byte[] inData = "test".getBytes();
    String encData = sed.base64Encode(inData);
    byte[] decData = sed.base64Decode(encData);
    String decDataStr = new String(decData);
    System.out.println(decDataStr);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 3.23 根据算法对数据进行摘要运算

功能：

对数据做摘要运算。

函数定义：

byte[] hashData (String alg, byte[] inData, String cert);

参数：

- 输入参数 ：String alg：算法 可选值为 SHA1，SHA256，SM3
- 输入参数 ：byte[] inData：原文。
- 输入参数 ：String cert：证书 算法为 RSA 时，该值无效，算法为 ECC 时，如果该值不为空，则公钥参与运算，如果该值为空，则公钥不参与运算。

返回：

hash 后的数据。

抛出：

无。

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    byte[] inData = "test".getBytes();
    String alg = "SHA1";
    String cert = sed.getServerCertificate();
    byte[] hashData = sed.hashData(alg, inData, cert);
    String hashDataStr = sed.base64Encode(hashData);
    System.out.println(hashDataStr);
} catch (SVSConnectException e) {
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    e.printStackTrace();
} catch (InitException e) {
    e.printStackTrace();
}
```

## 3.24 对摘要数据进行签名

功能：

对摘要数据进行签名运算。

函数定义：

```
byte[] signHashedData(byte[] hashData);
```

参数：

- 输入参数 ：byte[] hashData：摘要数据。

返回：

签名数据。

抛出：

无。

调用过程：

```
SecurityEngineDeal securityEngineDeal = null;
try {
    securityEngineDeal = SecurityEngineDeal.getInstance("BJCADevice");
    byte[] hash = securityEngineDeal.hashData("SHA256", "abc".getBytes(), null);
    byte[] signValue = securityEngineDeal.signHashedData(hash);
```

```
System.out.println(securityEngineDeal.base64Encode(signValue));
        } catch (SVSConnectException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ApplicationNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InitException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ParameterTooLongException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

# 4 扩展接口列表

## 4.1 数据签名

功能：

对字符串数据进行数字签名，签名格式为 Pkcs1，签名算法的选择(RSA 或者 SM2)取决于所配置的应用的服务器证书类型。

函数定义：

```
String signData(String inData);
```

参数：

- inData：[IN] 待签名的数据原文。

返回值：

成功返回 pkcs1 格式的签名值的 base64 编码。失败返回空值（NULL）

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String data = "test";
    String signedValue = sed.signData(data);
    System.out.println(signedValue);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        } catch (ApplicationNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InitException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ParameterTooLongException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

## 4.2 验证数据签名

（一）

功能：

验证数字签名，格式为 Pkcs1

函数定义：

```
boolean verifySignedData(String base64EncodeCert, String inData, String signValue);
```

参数：

- base64EncodeCert：[IN] base64 编码的签名证书。
- inData：[IN] 待验证的原文。
- signValue：[IN] 签名值。

返回值：

true：验证成功。false：验证失败。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterInvalidException | 参数无效 |
| UnkownException | 未知异常 |

调用过程：

```
        SecurityEngineDeal sed = null;
        try {
            sed = SecurityEngineDeal.getInstance("SVSDefault");
            String data = "test";
            String signedValue = sed.signData(data);
            String cert = sed.getServerCertificate();
            boolean verifyRes = sed.verifySignedData(cert, data,
signedValue);
            System.out.println(verifyRes);
        } catch (SVSConnectException e) {
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        } catch (ApplicationNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InitException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ParameterTooLongException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ParameterInvalidException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (UnkownException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

（二）

功能：

验证数字签名，格式为 Pkcs1

函数定义：

```
boolean verifySignedData(String base64EncodeCert, byte[] inData, String signValue);
```

参数：

- base64EncodeCert：[IN] base64 编码的签名证书。
- inData：[IN] 待验证的原文。
- signValue：[IN] 签名值。

返回值：

true：验证成功。false：验证失败。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterInvalidException | 参数无效 |
| UnkownException | 未知异常 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    byte[] data = "test".getBytes();
    String signedValue = sed.signData(data);
    String cert = sed.getServerCertificate();
    boolean verifyRes = sed.verifySignedData(cert, data,
signedValue);
    System.out.println(verifyRes);
```

```
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterInvalidException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (UnkownException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 4.3 PKCS7 签名（带原文）

功能：

对字符串数据进行数字签名，签名格式为 Pkcs7（带原文）。

函数定义：

```
String signDataByP7Attach (String inData);
```

参数：

- inData：[IN] 待签名的数据原文。

返回值：

成功返回 pkcs7 格式的签名值的 base64 编码（带原文）。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String data = "test";
    String signedValue = sed.signDataByP7Attach(data);
    System.out.println(signedValue);
} catch (SVSConnectException e) {
```

```
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ApplicationNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InitException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ParameterTooLongException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

## 4.4 验证 PKCS7 签名（不带原文）

功能：

验证数字签名，格式为 PKCS7（不带原文），本接口不验证证书有效性

函数定义：

```
boolean verifyP7DetachedSignature(byte[] originalData,byte[] pkcs7SignData);
```

参数：

- originalData：[IN] 原文
- pkcs7SignData：[IN] p7 签名值（不带原文）。

返回：

成功返回 true，否则返回 false。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
    SecurityEngineDeal sed = null;
    try {
        sed = SecurityEngineDeal.getInstance("SVSDefault");
        byte[] inData = "abc".getBytes();
        String signedValue = sed.signDataByP7Detach(inData);
        System.out.println(sed.verifyP7DetachedSignature(inData,
sed.base64Decode(signedValue)));
    } catch (SVSConnectException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ApplicationNotFoundException e) {
        // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        } catch (InitException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

## 4.5 XML 签名

该接口目前尚未支持 SM2 算法

（一）

功能：

对 XML 数据进行数字签名,输出符合国际标准的 XML 签名结果(暂不支持 utf-8 字符集)。

函数定义：

```
String signDataByXML (String inData);
```

参数：

- inData：[IN] 签名原文，XML 格式。

返回：

签名结果。出错返回空值。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String data = "<?xml version=\"1.0\"
encoding=\"gbk\"?><test>test</test>";
    String signedValue = sed.signDataByXML(data);
    System.out.println(signedValue);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
```

```
        } catch (ParameterInvalidException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

(二)

功能：

对 XML 数据进行数字签名，输出符合国际标准的 XML 签名结果。

函数定义：

```
String signDataByXML (byte[] inData);
```

参数：

- inData：[IN] 签名原文，XML 格式。

返回：

签名结果。出错返回空值。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
        SecurityEngineDeal sed = null;
        try {
            sed = SecurityEngineDeal.getInstance("SVSDefault");
            byte[] data = "<?xml version=\"1.0\"
encoding=\"gbk\"?><test>test</test>".getBytes();
            String signedValue = sed.signDataByXML(data);
            System.out.println(signedValue);
        } catch (SVSConnectException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ApplicationNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InitException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ParameterTooLongException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ParameterInvalidException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

# 4.6 验证 XML 签名

该接口目前尚未支持 SM2 算法

（一）

功能：

验证 xml 签名。

函数定义：

```
boolean  verifySignedDataByXML (String inData)
```

参数：

- InData：[IN] XML 签名值.

返回：

成功返回 true，否则返回 false。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String data = "<?xml version=\"1.0\"
encoding=\"gbk\"?><test>test</test>";
    String signedValue = sed.signDataByXML(data);
    boolean verifyRes =
sed.verifySignedDataByXML(signedValue);
    System.out.println(verifyRes);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterInvalidException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

（二）

功能：

验证 xml 签名。

函数定义：

```
boolean  verifySignedDataByXML (byte[] inData)
```

参数：

- InData：[IN]  XML 签名值.

返回：

成功返回 true，否则返回 false。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String data = "<?xml version=\"1.0\"
encoding=\"gbk\"?><test>test</test>";
    String signedValue = sed.signDataByXML(data);
    byte[] signedValueByte = signedValue.getBytes();
    boolean verifyRes =
sed.verifySignedDataByXML(signedValueByte);
    System.out.println(verifyRes);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterInvalidException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 4.7 对称加密

（一）

功能：

对称加密。

函数定义：

```
String symmEncryptData(String key, String inData)
```

参数：

- key：[IN] 加密密钥 （Base64 编码后的密钥），密钥长度需要是 24 位
- inData：[IN] 待加密的明文

返回值：

成功返回加密后的密文（Base64 编码后的）。出错返回空值。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterOutRangeException | 参数超出范围 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String key = sed.genRandom(24);
    String data = "test";
    String encData = sed.symmEncryptData(key, data);
    System.out.println(encData);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterOutRangeException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

（二）

功能：

对称加密。

函数定义：

```
Byte[] symmEncryptData(String key, byte[] inData)
```

参数：

- key：[IN] 加密密钥 （Base64 编码后的密钥），密钥长度需要是 24 位
- inData：[IN] 待加密的明文

返回值：

成功返回加密后的密文。出错返回空值。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterOutRangeException | 参数超出范围 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String key = sed.genRandom(24);
    byte[] data = "test".getBytes();
    byte[] encData = sed.symmEncryptData(key, data);
    String encDataStr = sed.base64Encode(encData);
    System.out.println(encDataStr);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterOutRangeException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 4.8 对称解密

(一)

功能：

对称解密。

函数定义：

```
String symmDecryptData(String key, String inData)
```

参数：

- key：[IN] 解密密钥 （Base64 编码后的密钥），密钥长度需要是 24 位
- inData：[IN] 待解密的密文

返回值：

解密后的明文。出错返回空值。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterOutRangeException | 参数超出范围 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String key = sed.genRandom(24);
    String data = "test";
    String encData = sed.symmEncryptData(key, data);
    String decData = sed.symmDecryptData(key, encData);
    System.out.println(decData);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterOutRangeException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

（二）

功能：

对称解密。

函数定义：

```
Byte[] symmDecryptData(String key, byte[] inData)
```

参数：

- key：[IN] 解密密钥 （Base64 编码后的密钥），密钥长度需要是 24 位
- inData：[IN] 待解密的明文

返回值：

解密后的明文。出错返回空值。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterOutRangeException | 参数超出范围 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String key = sed.genRandom(24);
    byte[] data = "test".getBytes();
    byte[] encData = sed.symmEncryptData(key, data);
    byte[] decData = sed.symmDecryptData(key, encData);
    String decDataStr = new String(decData);
    System.out.println(decDataStr);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterOutRangeException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 4.9 获取 XML 签名值的签名信息

该接口目前尚未支持 SM2 算法

（一）

功能：

解析 XML 签名数据，获取签名值、XML 原文、证书等信息。

函数定义：

```
String getXMLSignatureInfo (String xmlSignedData, int type)
```

参数：

- xmlSignedData：[IN] xml 签名数据。
- type：[IN] 类型待解析的参数类型。可选的参数和意义 1：xml 原文 2：

摘要 3：签名值 4：签名证书 5：摘要算法 6：签名算法。

返回值：

成功返回 type 对应的值，失败返回 null。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterOutRangeException | 参数超出范围 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String data = "<?xml version=\"1.0\"
encoding=\"gbk\"?><test>test</test>";
    String signedValue = sed.signDataByXML(data);
    String info = sed.getXMLSignatureInfo(signedValue, 1);
    System.out.println(info);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterInvalidException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterOutRangeException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

（二）

功能：

解析 XML 签名数据，获取签名值、XML 原文、证书等信息。

函数定义：

```
String getXMLSignatureInfo (byte[] xmlSignedData, int type)
```

参数：

- xmlSignedData：[IN] xml 签名数据。

- type：[IN] 类型待解析的参数类型。可选的参数和意义 1：xml 原文 2：摘要 3：签名值 4：签名证书 5：摘要算法 6：签名算法。

返回值：

成功返回 type 对应的值，失败返回 null。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterOutRangeException | 参数超出范围 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    byte[] data = "<?xml version=\"1.0\"
encoding=\"gbk\"?><test>test</test>".getBytes();
    String signedValue = sed.signDataByXML(data);
    byte[] signedValueByte = signedValue.getBytes();
    String info = sed.getXMLSignatureInfo(signedValueByte,
1);

    System.out.println(info);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterInvalidException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterOutRangeException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 4.10 非对称加密

功能：

使用证书公钥对数据加密。

函数定义：

```
String pubKeyEncrypt(String base64EncodeCert, String inData);
```

参数：

- 输入参数 ：base64EncodeCert BASE64 编码证书。
- 输入参数 ：inData 待加密的数据（所能加密的最大数据长度和使用的密钥长度有关，如 RSA1024 则最大长度为 117）。

返回：

成功返回加密后的密文(base64 编码后的)。否则返回空。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String data = "test";
    String cert = sed.getServerCertificate();
    String encData = sed.pubKeyEncrypt(cert, data);
    System.out.println(encData);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterInvalidException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 4.11 非对称解密

功能：

私钥解密（Pkcs1 格式）。

函数定义：

```
String priKeyDecrypt (String inData);
```

参数：

- 输入参数 ：inData：待解密的数据。

返回：

成功返回解密后的明文。否则返回空。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String data = "test";
    String cert = sed.getServerCertificate();
    String encData = sed.pubKeyEncrypt(cert, data);
    String decData = sed.priKeyDecrypt(encData);
    System.out.println(decData);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterInvalidException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 4.12 文件数字信封加密

功能：

对文件进行数字信封加密，RSA 算法符合 PKCS7 标准规范，国产算法符合消息语法规范。

函数定义：

```
boolean encodeEnvelopedFile(String base64EncodeCert, String inFile, String outFile) throws SVSConnectException, ParameterTooLongException, ParameterInvalidException, ParameterOutRangeException, InitException;
```

参数：

- 输入参数 ：base64EncodeCert BASE64 编码证书。
- 输入参数 ：inFile 待加密的文件(最大支持 500M 文件)。
- 输入参数 ：outFile 加密后的文件。

返回：

成功返回 true，失败返回 false

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterInvalidException | 参数无效 |
| ParameterOutRangeException | 参数越界 |
| InitException | 初始化异常 |

调用过程(示例代码未捕获异常，实际使用需要捕获处理)：

```
    SecurityEngineDeal sed =
SecurityEngineDeal.getInstance("SVSDefault");
    String cert = sed.getServerCertificate();
    boolean envRes = sed.encodeEnvelopedFile(cert, "E:\\temp\\test.txt",
"E:\\temp\\test.txt.env");
    System.out.println(envRes);
```

## 4.13 文件数字信封解密

功能：

解密文件数字信封，RSA 算法符合 PKCS7 标准规范，国产算法符合消息语法规范。

函数定义：

```
    boolean decodeEnvelopedFile(String inFile, String outFile) throws SVSConnectException,
ParameterTooLongException, ParameterInvalidException,
ParameterOutRangeException, InitException;
```

参数：

- 输入参数 ：inFile：待解密的文件。
- 输入参数 ：outFile：解密后的文件。

返回：

成功返回 true，失败返回 false。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterInvalidException | 参数无效 |
| ParameterOutRangeException | 参数越界 |
| InitException | 初始化异常 |

调用过程(示例代码未捕获异常，实际使用需要捕获处理)：

```
    SecurityEngineDeal sed =
SecurityEngineDeal.getInstance("SVSDefault");
```

```
    boolean envRes = sed.decodeEnvelopedFile("E:\\temp\\test.txt.env",
"E:\\temp\\test.txt");
    System.out.println(envRes);
```

## 4.14 文件加密

功能：

使用对称算法加密文件

函数定义：

```
boolean encryptFile(String key, String inFile,String outFile);
```

参数：

- 输入参数： String key，加密密钥（BASE64 编码）。
- 输入参数： String inFile，待加密的明文文件路径(最大支持 500M 文件)。
- 输入参数： String outFile，密文文件保存路径。

返回值：

成功返回 true，否则返回 false。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterOutRangeException | 参数超出范围 |
| InitException | 初始化异常 |

调用过程(示例代码未捕获异常，实际使用需要捕获处理)：

```
    SecurityEngineDeal sed =
SecurityEngineDeal.getInstance("SVSDefault");
    String key = sed.genRandom(24);
    String inFile = "C:\\temp\\test.txt";
    String outFile = "C:\\temp\\test_enc.txt";
    boolean encRes = sed.encryptFile(key, inFile, outFile);
    System.out.println(encRes);
```

## 4.15 文件解密

功能：

使用对称算法解密文件

函数定义：

```
boolean decryptFile(String key, String inFile, String outFile);
```

参数：

- 输入参数： String key，解密密钥（BASE64 编码）
- 输入参数： String inFile，待解密的密文文件路径
- 输入参数： String outFile，明文文件保存路径

返回值：

成功返回 true，否则返回 false。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterOutRangeException | 参数超出范围 |
| InitException | 初始化异常 |

调用过程(示例代码未捕获异常，实际使用需要捕获处理)：

```
    SecurityEngineDeal sed =
SecurityEngineDeal.getInstance("SVSDefault");
    String key = sed.genRandom(24);
    String inFile = "C:\\temp\\test.txt";
    String outFile = "C:\\temp\\test_enc.txt";
    String decFile = "C:\\temp\\test_dec.txt";
    boolean encRes = sed.encryptFile(key, inFile, outFile);
    boolean decRes = sed.decryptFile(key, outFile, decFile);
    System.out.println(decRes);
```

## 4.16 获取证书

功能：

根据证书序列号从 DSVS 中获取证书。

函数定义：

```
String getCert(String sn);
```

参数：

- 输入参数 ：String sn：证书序列号。

返回：

Base64 证书。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |

调用过程：

```
    SecurityEngineDeal sed = null;
    try {
        sed = SecurityEngineDeal.getInstance("SVSDefault");
        String sn = "20000000000001125373";
        String cert = sed.getCert(sn);
        System.out.println(cert);
    } catch (SVSConnectException e) {
```

```
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ApplicationNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (InitException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterTooLongException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

## 4.17 查询证书是否存在

功能：

根据证书序列号查询证书是否在 DSVS 中。

函数定义：

```
boolean containsCert(String sn);
```

参数：

- 输入参数 ：String sn：证书序列号。

返回：

查询结果，true 表示 DSVS 中有该证书，false 表示 DSVS 中没有该证书。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |

调用过程：

```
    SecurityEngineDeal sed = null;
    try {
        sed = SecurityEngineDeal.getInstance("SVSDefault");
        String sn = "20000000000001125373";
        boolean contRes = sed.containsCert(sn);
        System.out.println(contRes);
    } catch (SVSConnectException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ApplicationNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (InitException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterTooLongException e) {
```

```
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

## 4.18 根据证书序列号验证签名

功能：

根据证书序列号验证签名。

函数定义：

```
int verifySignatureBySN(String sn,String inData, String signedValue);
```

参数：

- 输入参数 ：String sn：证书序列号。
- 输入参数 ：String inData：原文。
- 输入参数 ：String signedValue：签名值。

返回：

验证结果，1 表示验证通过，-1 表示验证未通过，-2 表示未找到证书。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterOutRangeException | 参数超出范围 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
    SecurityEngineDeal sed = null;
    try {
        sed = SecurityEngineDeal.getInstance("SVSDefault");
        String data = "test";
        String signedValue = sed.signData(data);
        String cert = sed.getServerCertificate();
        String sn = sed.getCertInfo(cert, 2);
        int verifyRes = sed.verifySignatureBySN(sn, data, signedValue);
        System.out.println(verifyRes);
    } catch (SVSConnectException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ApplicationNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (InitException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterTooLongException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
```

```
    } catch (ParameterOutRangeException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterInvalidException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

## 4.19 验证并保存证书

功能：

验证证书有效性并将证书保存到 DSVS。

函数定义：

```
int validateAndSaveCert(String cert);
```

参数：

- 输入参数 ：String cert：base64 证书。

返回：

验证并保存的结果，证书有效并且已经存在 DSVS 中返回 1，证书有效并保存成功返回 2，证书无效情况下： -1 为不是所信任的根，-2 为超过有效期，-3 为作废证书，-4 已加入黑名单，-5 证书未生效， 0 未知错误。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterOutRangeException | 参数超出范围 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
    SecurityEngineDeal sed = null;
    try {
        sed = SecurityEngineDeal.getInstance("SVSDefault");
        String cert =
"MIIE9TCCA92gAwIBAgIKIAAAAAAAARJTczANBgkqhkiG9w0BAQUFADA6MQswCQYDVQQG
EwJDTjENMAsGA1UECgwEQkpDQTENMAsGA1UECwwEQkpDQTENMAsGA1UEAwwEQkpDQTAeF
w0wOTEwMjExNjAwMDBaFw0xMTEwMDExNTU5NTlaMEgxCzAJBgNVBAYTAkNOMQ0wCwYDVQ
QKDARCSkNBMQ0wCwYDVQQLDARCSkNBMRswGQYDVQQDDBLmnI3liqHlmajor4HkuabkuIA
wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAInLo4VW6RLWDkzxgn3gkYzvzPWOgG0J
HVG1KWb18oGd+lo5RNOERn6rj6qmso7M1C5El5Zsu6jPjz0EL7/XmfUKhFhv7raii+MOl
M89PFe/8ifd0e5FoW7733pGua3MDUGxySdD73rU9JHnr60FRTuyz2zus0vForyWZumzny
S/AgMBAAGjggJxMIICbTAfBgNVHSMEGDAWgBTBzihoGF2OgzPxlaoIwz2KCJqddjAMBgN
VHQ8EBQMDB/gAMCsGA1UdEAQkMCKADzIwMDkxMDIyMDAwMDAwWoEPMjAxMTEwMDEyMzU5
NTlaMAkGA1UdEwQCMAAwgZkGA1UdHwSBkTCBjjBWoFSgUqRQME4xCzAJBgNVBAYTAkNOM
Q0wCwYDVQQKDARCSkNBMQ0wCwYDVQQLDARCSkNBMQ0wCwYDVQQDDARCSkNBMRIwEAYDVQ
QDEwljYTJjcmwzNDkwNKAyoDCGLmh0dHA6Ly9sZGFwLmJqY2Eub3JnLmNuL2NybC9iamN
hL2NhMmNybDM0OS5jcmwwEQYJYIZIAYb4QgEBBAQDAgD/MCoGC2CGSAFlAwIBMAkKBBto
```

```
dHRwOi8vYmpjYS5vcmcuY24vYmpjYS5jcnQwGgYFKlYLBwkEEUpKMDExMDAwMTAwMDE1M
TExMB0GCGCGSAGG+EQCBBFKSjAxMTAwMDEwMDAxNTExMTAbBggqVoZIAYEwAQQPMDExMD
AwMTAwMDE1MTExMB4GBipWCwcBCAQUMUJASkowMTEwMDAxMDAwMTUxMTEwgbAGA1UdIAS
BqDCBpTA1BgkqgRwBxTiBFQEwKDAmBggrBgEFBQcCARYaaHR0cDovL3d3dy5iamNhLm9y
Zy5jbi9jcHMwNQYJKoEcAcU4gRUCMCgwJgYIKwYBBQUHAgEWGmh0dHA6Ly93d3cuYmpjY
S5vcmcuY24vY3BzMDUGCSqBHAHFOIEVAzAoMCYGCCsGAQUFBwIBFhpodHRwOi8vd3d3Lm
JqY2Eub3JnLmNuL2NwczANBgkqhkiG9w0BAQUFAAOCAQEAgjF3O4M0Ztt1ykvbpMLWtCM
lJ4sXZh6ABh/XW3tC72wldX9KwJZIHG1lLYgumWHJdZjKL1AzayUntl7ifjkNEIMjU1oq
ajAAMXSPIPV84hgRCVCx54ue+udknVFQhO1dfAl5cdc4SU2rftIUmx8FG0BF9qwNwE7Gd
Ix8cdYusdXFivYKcExypBQrRS284QOlB1a4GEoU1Pf7OvZ/86wzMRwic3DxM4iZjvJv+G
4okR0w2HvRaYO7fsY4H2yBvjtiKoiuryA6pqHIRtcgQvpxOdlRmxZgL5x4Ss3aJcIxUSJ
7SF6w/nF5ywD/VSZZ8/jo7/avISOEuqDmvkqDYn0Txw==";
        int saveRes = sed.validateAndSaveCert(cert);
        System.out.println(saveRes);
    } catch (SVSConnectException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ApplicationNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (InitException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterTooLongException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterOutRangeException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterInvalidException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

## 4.20 通过证书或证书序列号验证签名

功能：

通过证书或者证书序列号验证签名(证书和序列号只能有一个，否则视为无效参数)。

函数定义：

```
int verifySignatureByCertOrSN(String cert, String inData,String signedValue,String sn);
```

参数：

- 输入参数 ： String cert：base64 证书。
- 输入参数 ： String inData：原文

- 输入参数 ：String signedValue：签名值
- 输入参数 ：String sn：证书序列号

返回：

1 验证签名成功，证书存储成功
0 验证签名成功，证书已存在
-1 证书不被信任
-2 证书超过有效期
-3 证书已作废
-4 证书已加入黑名单
-5 证书未生效
-6 验证签名失败
-7 证书不存在
其他 异常错误，表示失败

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| ParameterOutRangeException | 参数超出范围 |
| ParameterInvalidException | 参数无效 |

调用过程：

通过证书验证：

```
    SecurityEngineDeal sed = null;
    try {
        sed = SecurityEngineDeal.getInstance("SVSDefault");
        String data = "test";
        String signedValue = sed.signData(data);
        String cert = sed.getServerCertificate();
        int verifyRes = sed.verifySignatureByCertOrSN(cert, data,
signedValue, "");
        System.out.println(verifyRes);
    } catch (SVSConnectException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ApplicationNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (InitException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterTooLongException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (ParameterOutRangeException e) {
        // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        } catch (ParameterInvalidException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

通过序列号验证：

```
        SecurityEngineDeal sed = null;
        try {
            sed = SecurityEngineDeal.getInstance("SVSDefault");
            String data = "test";
            String signedValue = sed.signData(data);
            String cert = sed.getServerCertificate();
            String sn = sed.getCertInfo(cert, 2);
            int verifyRes = sed.verifySignatureByCertOrSN("", data,
signedValue, sn);
            System.out.println(verifyRes);
        } catch (SVSConnectException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ApplicationNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InitException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ParameterTooLongException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ParameterOutRangeException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (ParameterInvalidException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

## 4.21 签名（返回值：签名值#base64(sn#原文)）

功能：

对原文数据进行签名。

函数定义：

```
String signDataReAllInfo(String inData);
```

参数：

- 输入参数 ：String inData：原文

返回：

签名值#base64(sn#原文)。

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String data = "test";
    String signedValue = sed.signDataReAllInfo(data);
    System.out.println(signedValue);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 4.22 验证签名（输入参数：签名值#base64(sn#原文)）

功能：

通过特殊格式的输入参数验证签名值。

函数定义：

```
int verifySignedDataByAllInfo(String verifySignedData);
```

参数：

- 输入参数 ：签名值#base64(sn#原文)。

返回：

1 验证签名成功

-1 验证签名失败

-2 未找到证书

其他 异常错误，表示失败

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |

| | |
|---|---|
| ParameterTooLongException | 参数过长 |
| UnkownException | 未知异常 |
| ParameterInvalidException | 参数无效 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String data = "test";
    String signedValue = sed.signDataReAllInfo(data);
    int verifyRes = sed.verifySignedDataByAllInfo(signedValue);
    System.out.println(verifyRes);
} catch (SVSConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (InitException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterTooLongException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (ParameterInvalidException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (UnkownException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

## 4.23 获取 P10 数据

功能：

返回 P10 数据，信手书产品一次性密钥对使用

函数定义：

```
Map getP10(byte[] image, byte[] pdfHash, int[] rv) throws SVSConnectException, UnkownException, ParameterOutRangeException, ParameterInvalidException;
```

参数：

- 输入参数 ：image：手写笔迹
- 输入参数 ：pdfHash：PDF 文档摘要
- 输入参数 ：rv：调用结果

返回：

返回 Map 中 key 为缓存 P10Info 对象描述：p10info， value 为 P10Info 对象

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterOutRangeException | 参数过长 |
| UnkownException | 未知异常 |
| ParameterInvalidException | 参数无效 |

调用过程：(编写用例时将异常抛出了，实际使用时可以根据需要捕获，另外 image 和 pdfHash 以及 rv 值，实际使用时都是从信手书传递过来)

```
        SecurityEngineDeal.setProfilePath("C:\\BJCAROOT");
        SecurityEngineDeal securityEngineDeal =
SecurityEngineDeal.getInstance("SVSDefault");
        // 实际使用时由信手书传入
        byte[] image = "abc".getBytes();
        FileUtil fileUtil = new FileUtil();
        String pdfFilePath = "E:\\temp\\aaa.pdf";
        // 实际使用时由信手书传入
        byte[] pdfHash =
securityEngineDeal.hashData(fileUtil.readFile(pdfFilePath));
        int[] rv = new int[1];
        // map中包含ID和P10
        String keyPairIdString = "";
        String p10String = "";
        Map p10ResMap = securityEngineDeal.getP10(image, pdfHash, rv);
        if (rv[0] != 0) {
            System.out.println("genP10返回结果为" + rv[0]);
            return;
        }
        P10Info p10Info = (P10Info) p10ResMap.get("p10info");

        // 取出ID和P10
        keyPairIdString = p10Info.getKeyPairID();
        p10String =
securityEngineDeal.base64Encode(p10Info.getP10Data());
        System.out.println("ID=" + keyPairIdString);
        System.out.println("P10=" + p10String);
```

## 4.24 获取签名数据

功能：

返回签名数据，信手书产品使用一次性密钥签名使用

函数定义：

```
byte[] getSignData(String respSessionID, byte[] realHash, byte[] onceCert, int[] rv) throws
SVSConnectException, UnkownException, ParameterOutRangeException,ParameterInvalidException;
```

参数：

- 输入参数 ：respSessionID：获取 keyPair 对象 ID
- 输入参数 ：realHash：PDF 文档摘要
- 输入参数 ：onceCert：证书数据
- 输入参数 ：rv：调用结果

返回：

成功返回签名数据

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterOutRangeException | 参数过长 |
| UnkownException | 未知异常 |
| ParameterInvalidException | 参数无效 |

调用过程：(编写用例时将异常抛出了，实际使用时可以根据需要捕获，该接口一般结合 getP10 一起使用，onceCert 根据 P10 生成)

```
        SecurityEngineDeal.setProfilePath("C:\\BJCAROOT");
        SecurityEngineDeal securityEngineDeal =
SecurityEngineDeal.getInstance("SVSDefault");

        // 该参数后台未使用，因此随便传了一个
        byte[] image = "abc".getBytes();
        // 随便传入一个pdf，然后做hash
        FileUtil fileUtil = new FileUtil();
        String pdfFilePath = "E:\\temp\\aaa.pdf";
        byte[] pdfHash =
securityEngineDeal.hashData(fileUtil.readFile(pdfFilePath));
        int[] rv = new int[1];
        // map中包含ID和P10
        String keyPairIdString = "";
        String p10String = "";
        Map p10ResMap = securityEngineDeal.getP10(image, pdfHash, rv);
        if (rv[0] != 0) {
            System.out.println("genP10返回结果为" + rv[0]);
            return;
        }
        P10Info p10Info = (P10Info) p10ResMap.get("p10info");

        // 取出ID和P10
        keyPairIdString = p10Info.getKeyPairID();
        p10String =
securityEngineDeal.base64Encode(p10Info.getP10Data());
        System.out.println("ID=" + keyPairIdString);
        System.out.println("P10=" + p10String);

        byte[] onceCert = securityEngineDeal
                .base64Decode("MIIEajCCA9OgAwIBAgIKEAAAAAAAAATgDANBgkq
```

```
hkiG9w0BAQUFADBbMQswCQYDVQQGEwJDTjELMAkGA1UECAwCQkoxCzAJBgNVBAcMAkJKM
Q0wCwYDVQQKDARBWFRYMQ0wCwYDVQQLDARCSkNBMRQwEgYDVQQDDAtCSkNBIENBLVJTQT
AeFw0xNDEwMDgwNTAwMDBaFw0xNjEwMDkwNDU5NTlaME8xCzAJBgNVBAYMAkNOMQ0wCwY
DVQQKDARiamNhMTEwLwYDVQQDDCg3MzkyYTYwNjA5ZWM0ZjkzYjIyYWE5N2U2OWQ3NDVk
MmI1YzQ4NDRkMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCB0U5BUuh8Y8h6SYPUU
ik/GJ6OAbmIFo98g1B/O/lP4yZbY3HWF2a7gI+ZPnXhDSVYGwJA1ZnjfQWpFhpjrj36XT
ROirwJ8b+Gt59AowMlLStA212SausnbPVtK/nNhCY6ssx2p8ohXECVP6qH89Xjm119WBI
UcCQq9Co6eDDRsQIDAQABo4ICPzCCAjswHwYDVR0jBBgwFoAUE+HxRvN0jvzfmLXUitiP
f3+OAwcwgbcGA1UdHwSBrzCBrDB2oHSgcqRwMG4xCzAJBgNVBAYTAkNOMQswCQYDVQQID
AJCSjELMAkGA1UEBwwCQkoxDTALBgNVBAoMBEFYVFgxDTALBgNVBAsMBEJKQ0ExFDASBg
NVBAMMC0JKQ0EgQ0EtUlNBMREwDwYDVQQDEwhjYTIyY3JsMTAyoDCgLoYsaHR0cDovLzE
5Mi4xNjguMTM2LjI0MDo4MDAwL2NybC9jYTIyY3JsMS5jcmwwCQYDVR0TBAIwADARBglg
hkgBhvhCAQEEBAMCAP8wEgYIYIZIAYb4RAIEBkpKYmpjYTAbBggqVoZIAYEwAQQPODA3M
DAwMTAwMDAxMjAyMA8GBSpWCwcJBAZKSmJqY2EwEgYGKlYLBwEIBAgxQEpKYmpjYTAqBg
tghkgBZQMCATAJCgQbaHR0cDovL2JqY2Eub3JnLmNuL2JqY2EuY3J0MIGwBgNVHSAEgag
wgaUwNQYJKoEcAcU4gRUBMCgwJgYIKwYBBQUHAgEWGmh0dHA6Ly93d3cuYmpjYS5vcmcu
Y24vY3BzMDUGCSqBHAHFOIEVAjAoMCYGCCsGAQUFBwIBFhpodHRwOi8vd3d3LmJqY2Eub
3JnLmNuL2NwczA1BgkqgRwBxTiBFQMwKDAmBggrBgEFBQcCARYaaHR0cDovL3d3dy5iam
NhLm9yZy5jbi9jcHMwCwYDVR0PBAQDAgP4MA0GCSqGSIb3DQEBBQUAA4GBAA85kDUJ+wN
svywaeAhVZe1EA2v4C7W4j+iWcxRdCWogUwPbwGdpJTlUlZLg4hA/cPfIX8EsuqRkzpav
YNau8EqfUbTddjLvxNQi0FliH0wu3HlhEY7I4t7b8kKfKyTzrJrk5yYLlrOeDl877THdm
/8J9Ge0XUsdO0jMhGrnxUbj");
        byte[] signValue =
securityEngineDeal.getSignData(keyPairIdString, pdfHash, onceCert,
rv);
        if (rv[0] != 0) {
            System.out.println("getSignData返回结果为" + rv[0]);
            return;
        }
        System.out.println("signValue=" +
securityEngineDeal.base64Encode(signValue));
```

## 4.25 对数据做摘要运算

（一）

功能：

对数据做摘要运算。只支持 SHA1 和 SHA256 算法，根据应用对应的摘要算法类型选择

函数定义：

byte[] hashData (byte[]  inData);

参数：

- 输入参数 ：byte[]  inData：原文。

返回：

编码后的数据。

抛出：

无。

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    byte[] inData = "test".getBytes();
    byte[] hashData = sed.hashData(inData);
    String hashDataStr = sed.base64Encode(hashData);
    System.out.println(hashDataStr);
} catch (SVSConnectException e) {
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    e.printStackTrace();
} catch (InitException e) {
    e.printStackTrace();
}
```

（二）

功能：

对数据做摘要运算。只支持 SHA1 和 SHA256 算法，根据应用对应的摘要算法类型选择

函数定义：

byte[] hashData (String  inData);

参数：

- 输入参数 ：String  inData：原文。

返回：

编码后的数据。

抛出：

无。

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String inData = "test";
    byte[] hashData = sed.hashData(inData);
    String hashDataStr = sed.base64Encode(hashData);
    System.out.println(hashDataStr);
} catch (SVSConnectException e) {
    e.printStackTrace();
```

```
} catch (ApplicationNotFoundException e) {
    e.printStackTrace();
} catch (InitException e) {
    e.printStackTrace();
}
```

## 4.26 获取签名业务日志

功能：

查询签名的业务日志

函数定义：

```
List  getBusinessLog(byte[] hashData);
```

参数：

- 输入参数 ：String  hashData：原文的 hash 值。

返回：

业务日志的集合。

抛出：

无。

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("SVSDefault");
    String inData = "test";
    List list = new ArrayList();
    list = sed.getBusinessLog(sed.hashData(inData));
    String signedValue = (String) list.get(0);
    String sn = (String) list.get(1);
    String time = (String) list.get(2);
    System.out.println("base64签值:" + signedValue);
    System.out.println("签名服务器证书序列号:" + sn);
    System.out.println("签名时间:" + time);
} catch (SVSConnectException e) {
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    e.printStackTrace();
} catch (InitException e) {
    e.printStackTrace();
}
```

## 4.27 根据业务 ID 生成密钥对

功能：

根据业务 ID 生成密钥对接口，移动招投标定制接口

函数定义：

```
String generateCert(String uid, int type, int length);
```

参数：

- 输入参数 ：String uid：移动招投标 ID 值。
- 输入参数 ：int type：密钥对类型 0 为 RSA 密钥对 1 为 SM2 密钥对
- 输入参数 ：int length：密钥模长 当 type 为 0 时有效，可选值为 1024 和 2048

返回：

生成的密钥对对应的公钥证书

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("rsa");
    String uid = "20160627";
    String genCert = sed.generateCert(uid,1 , 1024);
    System.out.println(genCert);
} catch (SVSConnectException e) {
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    e.printStackTrace();
} catch (InitException e) {
    e.printStackTrace();
}
```

## 4.28 获取公钥

功能：

根据业务 ID 从数据库中获取公钥，移动招投标定制接口

函数定义：

```
String getPubKey(String uid);
```

参数：

- 输入参数 ：String uid：移动招投标 ID 值。

返回：

公钥

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| UnsupportedEncodingException | 不支持的编码格式异常 |

CommonClientException　　客户端异常

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("rsa");
    String uid = "20160627";
    System.out.println(sed.getPubKey(uid));
} catch (SVSConnectException e) {
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    e.printStackTrace();
} catch (InitException e) {
    e.printStackTrace();
}
```

## 4.29　获取公钥证书

功能：

根据业务 ID 从数据库中查询公钥证书，移动招投标定制接口

函数定义：

```
String getGenerateCert(String uid);
```

参数：

- 输入参数 ：String　uid：移动招投标 ID 值。

返回：

查询到的公钥证书

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| UnsupportedEncodingException | 不支持的编码格式异常 |
| CommonClientException | 客户端异常 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("rsa");
    String uid = "20160627";
    System.out.println(sed.getGenerateCert(uid));
} catch (SVSConnectException e) {
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    e.printStackTrace();
} catch (InitException e) {
    e.printStackTrace();
```

```
}
```

## 4.30 根据业务 ID 解密数字信封

功能：

根据业务 ID 生成的密钥解密数字信封，移动招投标定制接口

函数定义：

```
byte[] decodeP7Envelope(String uid, String encodeData);
```

参数：

- 输入参数 ：String uid：移动招投标 ID 值。
- 输入参数 ：String encodeData：待解密数字信封

返回：

保存结果

抛出：

| | |
|---|---|
| SVSConnectException | 连接服务端异常 |
| ParameterTooLongException | 参数过长 |
| UnsupportedEncodingException | 不支持的编码格式异常 |
| CommonClientException | 客户端异常 |

调用过程：

```
SecurityEngineDeal sed = null;
try {
    sed = SecurityEngineDeal.getInstance("rsa");
    String uid = "201606271800";
    System.out.println(sed.generateCert(uid, 1, 1024));
    String genCert = sed.getGenerateCert(uid);
    System.out.println(genCert);
    byte[] encData = sed.encodeEnvelopedData(genCert,
"abc".getBytes());
    System.out.println(sed.base64Encode(encData));
    System.out.println(new String(sed.decodeP7Envelope(uid,
sed.base64Encode(encData))));
} catch (SVSConnectException e) {
    e.printStackTrace();
} catch (ApplicationNotFoundException e) {
    e.printStackTrace();
} catch (InitException e) {
    e.printStackTrace();
}
```