关于上海海关放行电子信息加验签技术实施的简要说明

[v1.1]

上海市数字证书认证中心有限公司

2013-12-10

版本信息：

当前版本 1.1

目　录

一．术语说明

（1）数字证书：

由国家认可的，具有权威性、可信性和公正性的第三方证书认证机构（CA）进行数字签名的一个可信的数字化文件。

（2）根证书：

根证书是一份特殊的证书，它的签发者是它本身，是 CA 认证中心给自己颁发的证书,同时也是信任链的起始点，证书的验证追溯至根证书即为结束。

（3）数字签名：

可靠电子签名的一种技术实现形式，是以电子形式存在于数据信息之中的，或作为其附件的或逻辑上与之有联系的数据，可用于辨别数据签署人的身份，并表明签署人对数据信息中包含的信息的认可。数字签名是可靠的电子签名。

（4）签名验证

数字签名验证是验证者使用签名者的公开密钥对数字签名进行验证的过程。

（5）SM2

国家密码管理局在 2010 年 12 月份公布了《SM2 椭圆曲线公钥密码算法》，SM2 算法本质上是一种椭圆曲线算法（ECC）。

（6）SM3

国家密码管理局编制的商用算法，用于密码应用中的数字签名和验证、消息认证码的生成与验证以及随机数的生成，可满足多种密码应用的安全需求。

（7）RSA

RSA 算法是一种非对称密码算法。

（8）SHA1

SHA1 算法是一种杂凑算法。

（9）加密机

加密机是通过国家商用密码主管部门鉴定并批准使用的国内自主开发的主机加密设备。

（10）签名验签服务器

提供数字签名服务并对数据签名结果进行真实性、有效性验证的设备。设备可对关键敏感数据进行签名并验证，保证数据的真实性、完整性、合法性。

（11）USB KEY

USB KEY 是一种 USB 接口的硬件设备。它内置单片机或智能卡芯片，有一定的存储空间，可以存储用户的私钥以及数字证书，利用 USB KEY 内置的公钥算法实现对用户身份的认证。

二．应用过程

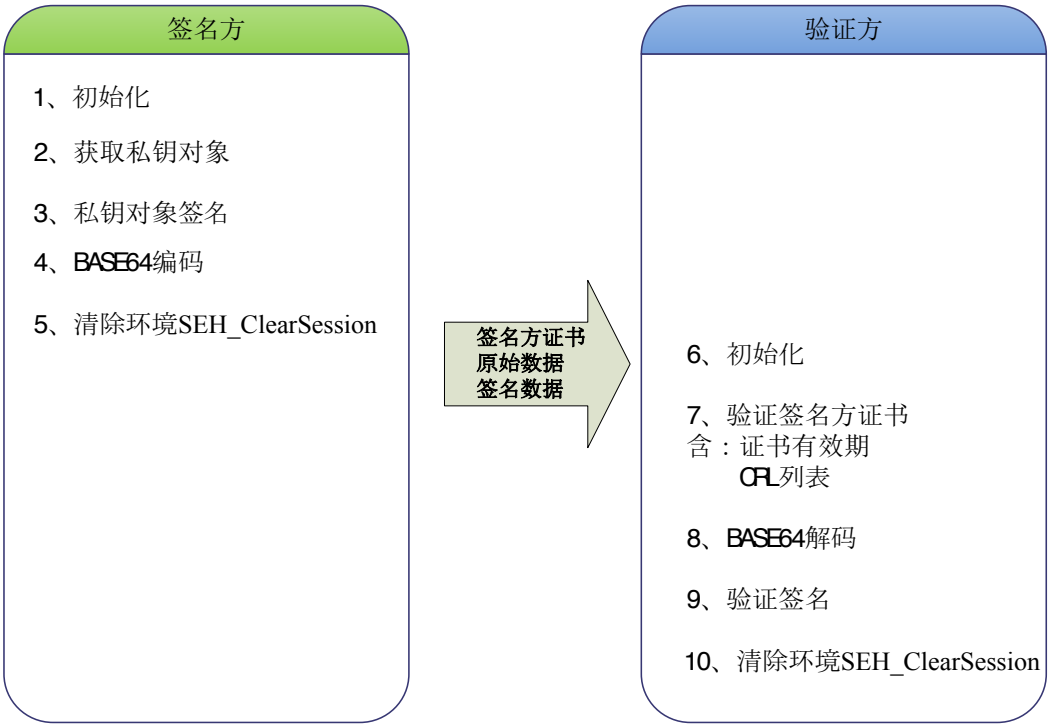
（1） 数据源发送方（签名方）调用上海数字认证中心（以下简称：上海 CA 中心）提供的 API 接口，执行初始化操作；

（2） 使用自己的私钥对数据或其他与数据内容有关的变量进行加密处理，完成对数据的合法“签名”，并对签名数据进行 BASE64 编码；

（3） 将数字证书、签名原始数据以及签名数据发送给验证方；

（4） 数据接收方（验证方）接收到对方的数字证书后，检验证书的合法性（包括：证书有效期、CRL 列表）

（5） 利用对方数字证书的公钥来解读收到的“数字签名”，并将解读结果用于对数据完整性的检验，以确认签名的合法性。



3-1 签名验证基本流程图

数字签名技术是在网络系统虚拟环境中确认身份的重要技术，在数字签名应用中，发送者的公钥可以公开，但私钥则需要严格保密。

备注：（1）在验证签名需要验证证书有效性；

（2）验证证书有效性需参照《上海海关放行电子信息安全认证规范V1.0》第3.4.3中的要求实施；

三、技术比较

内容	加密机	签名验签服务器	USB Key
技术特点	提供多密钥管理、数字签名验签服务、数据加密服务	提供多密钥管理、数字签名验签服务	提供单密钥管理、数字签名验签服务
加验签性能比较	高	高	低
密钥	多密钥	多密钥	单密钥
兼容平台	Windows / Linux/Unix/Solaris	Windows / Linux/Unix/Solaris	Windows

开发难易度	较难	较易	较易
支持算法	SM2 / RSA	SM2 / RSA	SM2 / RSA
成本投入	高	中	低

四. 接口说明

（a）加密机

调用上海数字认证中心的 SHECA _ SafeEngine 接口(以 java 为例)，连接加密机设备，实现数据的加密、签名操作。

1. 简述

简述上海数字认证中心的 SHECA_SafeEngine_Java 接口中常用的函数（SM2 与 RSA 算法)，详细内容请查询我司提供的 SHECA _SafeEngine_Java 接口文档。

加密机的接口文档及 API 接口包，我司将在我司网站上予以公布。

2. 主要 API 函数说明

基本项获取

密钥库

public KeyStore getKeyStore(String sStoreType,String sProviderName,String sStoreName,String sStorePin)

功能：

获取指定的密钥库。

参数：

参数名	含义	In/out	参数选项
sStoreType	密钥库类型	In	
sProviderName	算法提供者名称	In	
sStoreName	密钥库名称	In	
sStorePin	密钥库密码	In	

密钥库元素别名

public Enumeration getAliasEnum(KeyStore oStore)
功能：
根据密钥库获取别名枚举。
参数：

参数名	含义	In/out	参数选项
oStore	密钥库对象	In	

证书

密钥库对象证书

public java.security.cert.Certificate getCertFromStore(KeyStore oStore,String sAlias,String sPin)
功能：
从密钥库获取指定别名的证书对象。
参数：

参数名	含义	In/out	参数选项
oStore	密钥库对象	In	
sAlias	证书别名	In	
sPin	密钥库证书密码	In	

缓存证书（RSA）

publicjava.security.cert.CertificategetCertFromBuffer(byte[] bCert)
功能：
转换缓存证书数据为证书对象。
参数：

参数名	含义	In/out	参数选项
bCert	缓存证书数据	In	

数字签名

签名

密钥库隐式签名（RSA）

public byte[] sign(byte[] bData, KeyStore oStore, String sAlias, String sKeyPin, String sAlg, String sProviderName)

功能：

根据密钥库对象及指定签名算法进行签名运算。

参数：

参数名	含义	In/out	参数选项
bData	原文	In	
oStore	密钥库对象	In	
sAlias	私钥别名	In	
sKeyPin	私钥密码	In	
sAlg	签名算法	In	
sProviderName	签名算法提供者名称	In	

SM2 数字签名

public byte[] signSM2(String sAlg,String sProviderName,byte[] data,int keynum)

功能：

通过 JCE 调用加密机 SM2 算法进行数字签名运算。

参数：

参数名	含义	In/out	参数选项
sAlg	SM2 算法名	In	
sProviderName	加密机 JCE 提供者名称	In	
data	原文	In	
keynum	内部密钥号	In	

验证签名

对象证书（RSA）

public boolean verifySign(byte[] bData,byte[] bSign, String sAlg, java.security.cert.Certificate oCert, String sProviderName)

功能：

使用对象证书验证签名。

参数：

参数名	含义	In/out	参数选项
bData	原文	In	
bSign	签名数据	In	
sAlg	签名算法	In	
oCert	证书对象	In	
sProviderName	算法提供者名称	In	

返回：
验签成功（true）与否（false）。

缓存证书

public boolean verifySign(byte[] bData,byte[] bSign, String sAlg, byte[] bCert, String sProviderName)
功能：
使用缓存证书验证签名。
参数：

参数名	含义	In/out	参数选项
bData	原文	In	
bSign	签名数据	In	
sAlg	签名算法	In	
bCert	缓存证书	In	
sProviderName	算法提供者名称	In	

SM2 数字验签

```
public boolean verifySignSM2ByCert(String sAlg, String sProviderName,String sCert, byte[] bData, byte[] sign)
```

功能：
通过 JCE 调用厂商加密机基于 SM2 算法验签，用证书进行验签。
参数：

参数名	含义	In/out	参数选项
sAlg	SM2 算法名	In	
sProviderName	加密机 JCE 提供者名称	In	
sCert	证书	In	
bData	原文	In	
sign	验签内容	In	

返回：
验签成功（true）与否（false）。
Example:

```
javasafeengine ose = new javasafeengine();
ose.verifySignSM2ByCert("SM3withSM2", "FishermanJCE", cert,
                bData.getBytes(),sign);
```

通过黑名单验证证书

对象证书（RSA）

public int verifyCert(X509Certificate oCert,byte[] bChain,int iCfg)
功能：
验证证书有效性，先验有效期，ca 签发，根据配置决定是否验证本地黑名单，ca 黑名单。配置方式见参数说明。
参数：

参数名	含义	In/out	参数选项
oCert	证书对象	In	
bChain	PKCS#7 格式的证书链数据	In	
iCfg	验证方式配置参数	In	iCfg=000（二进制) 验证证书有效性及证书链后不验证黑名单 iCfg=001 验证证书有效性及证书链，继续验证本地 CRL 失败，继续到 CA 服务器下载并验证 CRL iCfg=101 验证证书有效性及证书链后继续验证本地 CRL，如果本地验证失败，直接返回错误. 出于兼容性和扩展性考虑 Bit 1，Bit 3 - Bit 31: 保留.

对象证书（SM2）

public int verifyCertSM2 (X509Certificate oCert,byte[] bChain,int iCfg, ,String sProviderName)
功能：
验证证书有效性，先验有效期，ca 签发，根据配置决定是否验证本地黑名单，ca 黑名单。配置方式见参数说明。
参数：

参数名	含义	In/out	参数选项
oCert	证书对象	In	
bChain	PKCS#7 格式的证书链数据	In	

iCfg	验证方式配置参数	In	iCfg=000（二进制) 验证证书有效性及证书链后不验证黑名单 iCfg=001 验证证书有效性及证书链，继续验证本地 CRL 失败，继续到 CA 服务器下载并验证 CRL iCfg=101 验证证书有效性及证书链后继续验证本地 CRL，如果本地验证失败，直接返回错误. 出于兼容性和扩展性考虑 Bit 1，Bit 3 - Bit 31: 保留.

Base64 编码/解码

Base64 编码

public String base64Encode(byte[] in)
功能：
将二进制数据经 Base64 编码转成可见字符串。
参数：

参数名	含义	In/out	参数选项
in	二进制数据	In	

返回：
Base64 编码字符数据。

Base64 解码

public byte[] base64Decode(String in)
功能：
将 Base64 编码字符数据解码成二进制数据。
参数：

参数名	含义	In/out	参数选项
in	待解码的数据	In	

返回：
Base64 解码数据。

3. Demo 示例

```
public static void main(String[] args) throws Exception {
        javasafeengine oSE = new javasafeengine();
        String sKeyPin = "12345678";
        String sStoreName = "";
        String sProviderName = "FishermanJCE";
        String sAlias = "key55";
        String sStoreType = "FMKS";
        System.out.println("begin");
        KeyStore oKeyStore = oSE.getKeyStore(sStoreType, sProviderName,
                sStoreName, sKeyPin);
        Certificate oCert = oSE.getCertFromStore(oKeyStore, sAlias,
sKeyPin);
        File file = new File("c:\\CertChain.spc");
        FileInputStream fin = new FileInputStream(file);
        byte[] bChain = readStream(fin);
        int iRet = sm2.verifyCertSM2((X509Certificate)oCert, bChain,
000,"FishermanJCE")
        if(iRet==1){
            System.out.println("Verify cert succeed.");
        }
        else {
            System.out.println("Verify cert fail.");
        }
        String bSign1 = "12345678";
        byte[] bsign = oSE.signSM2("SM3withSM2", "FishermanJCE", bSign1
                .getBytes(), 55);
        String ssign = new String(Base64.encode(bsign));
        System.out.println(ssign);
        System.out.println(ssign.length());
       boolean flag = oSE.verifySignSM2("SM3withSM2", "FishermanJCE", 55,
                bSign1.getBytes(), bsign);
        System.out.println(flag);
```

（b）签名验签服务器

1. 简述

调用上海数字认证中心的 webservice 接口，实现签名与验签操作。

接口协议：HTTP+SOAP

2. 接口说明

签名验签接口

签名

1） 请求

参数类型	参数名称	字段类型	字段说明	必填
应用系统标识	AppCode	VARCHAR(32)	应用系统标识	是
应用系统密码	AppPWD	VARCHAR(32)	应用系统密码	是
容器名	ContainerName	VARCHAR(32)	对应容器名	是
签名算法	SignAlg	VARCHAR(32)	签名算法： SHA1withRSA MD5withRSA SM3withSM2	是
原文	InData	VARCHAR	待签名原文	是

示例：

```
<?xml version="1.0" encoding="UTF-8" ?>
<Root>
    <AppCode>1</AppCode>
    <AppPWD>12345678</AppPWD>
    <Request>
        <ContainerName>容器名</ContainerName>
        <SignAlg> SHA1withRSA </SignAlg>
        <InData>待签名数据</InData>
    </Request>
```

2） 响应

参数类型	参数名称	字段类型	字段说明	必填
返回值	RetCode	VARCHAR(2)	1 成功，其他失败	是
返回消息	RetMsg	VARCHAR(128)	返回说明	是
签名值	SignData	VARCHAR	Base64 编码签名值	成功时

示例：

```
<?xml version="1.0" encoding="UTF-8" ?>
<Root>
    <RetCode>1</RetCode>
    <RetMsg>成功</RetMsg>
    <Response>
        <SignData>签名值</SignData>
    </Response>
</Root>
```

验签

1） 请求

参数类型	参数名称	字段类型	字段说明	必填
应用系统标识	AppCode	VARCHAR(32)	应用系统标识	是
应用系统密码	AppPWD	VARCHAR(32)	应用系统密码	是
证书	Cert	VARCHAR	Base64 编码公钥证书	是
签名算法	SignAlg	VARCHAR(32)	签名算法： SHA1withRSA MD5withRSA	是
原文	InData	VARCHAR	签名原文	是
签名值	SignData	VARCHAR	Base64 编码签名值	是

示例：

```
<?xml version="1.0" encoding="UTF-8" ?>
<Root>
    <AppCode>1</AppCode>
    <AppPWD>12345678</AppPWD>
    <Request>
        <Cert>Base64 编码证书</Cert>
        <SignAlg> SHA1withRSA </SignAlg>
        <InData>签名原文</InData>
        <SignData>签名值</SignData>
    </Request>
</Root>
```

2） 响应

参数类型	参数名称	字段类型	字段说明	必填
返回值	RetCode	VARCHAR(2)	1 成功，其他失败	是
返回消息	RetMsg	VARCHAR(128)	返回说明	是

示例：

```
<?xml version="1.0" encoding="UTF-8" ?>
<Root>
    <RetCode>1</RetCode>
    <RetMsg>成功</RetMsg>
</Root>
```

证书解析接口

获取证书基本项信息

1） 请求

参数类型	参数名称	字段类型	字段说明	必填
应用系统标识	AppCode	VARCHAR(32)	应用系统标识	是
应用系统密码	AppPWD	VARCHAR(32)	应用系统密码	是
证书	Cert	VARCHAR	Base64 编码证书	是

示例：

```
<?xml version="1.0" encoding="UTF-8" ?>
<Root>
    <AppCode>1</AppCode>
    <AppPWD>12345678</AppPWD>
    <Request>
        <Cert>Base64 编码证书</Cert>
    </Request>
</Root>
```

2） 响应

参数类型	参数名称	字段类型	字段说明	必填
返回值	RetCode	VARCHAR(2)	1 成功，其他失败	是
返回消息	RetMsg	VARCHAR(128)	返回说明	是
版本	Verison	VARCHAR(2)	版本	成功时
证书序列号	SerialNumber	VARCHAR(128)	证书序列号	成功时
证书颁发者	Issuer	VARCHAR(128)	证书颁发者	成功时
有效起始时间	NotBefore	VARCHAR(64)	有效起始时间，格式 yyyyMMddHHmmss	成功时
有效终止时间	NotAfter	VARCHAR(64)	有效终止时间，格式 yyyyMMddHHmmss	成功时
证书主题	Subject	VARCHAR(128)	证书主题	成功时
公钥	PublicKey	VARCHAR	Base64 编码公钥	成功时
签名值	Signature	VARCHAR	Base64 编码签名值	成功时

示例：

```
<?xml version="1.0" encoding="UTF-8" ?>
<Root>
    <RetCode>1</RetCode>
    <RetMsg>成功</RetMsg>
    <Response>
        <Verison>版本</ Verison>
        <SerialNumber>证书序列号</ SerialNumber>
        <Issuer>证书颁发者</ Issuer>
```

```
        <NotBefore>有效起始时间</ NotBefore>
        <NotAfter>有效终止时间</ NotAfter>
        <Subject>证书主题</ Subject>
        <PublicKey>公钥</ PublicKey>
        <Signature>签名值</ Signature>
    </Response>
</Root>
```

（c）USB KEY

1. 简述

调用上海数字认证中心的 SHECA_SafeEngine 接口(以 COM 为例)，调用 USB KEY 设备，实现数据的加密、签名操作。

2. 主要 API 函数说明

初始化环境

HRESULT ESE_InitialSession(long PriKeyDevType, BSTR strPrivKeyDevParam, BSTR strPriKeyPass, long lPriKeyTimeout, long CertChainDevType, BSTR strCertChainDevParam, BSTR strCertChainPass)

功能：初始化环境. 从设备中读取私钥, 根证书.

在调用以下方法前必须先调用此方法初始化.

参数：

参数名	含义	In/out	参数选项
PriKeyDevType	存储私钥的设备类型	in	见设备类型说明文件
StrPrivKeyDevParam	存储私钥设备的参数	In	可为”com1”, “com2” 若不需要私钥, 可设为空字符串 “”,表示不取私钥
strPriKeyPass	私钥密码	In	
LpriKeyTimeout	私钥超时时间. 秒为单位	In	若 = 0，则私钥永久有效。
CertChainDevType	存储证书链的设备类型	In	见设备类型说明文件
StrCertChainDevParam	存储证书链设备的参数	In	可为”com1”, “com2” 若不需要根证书, 可设为空字符串 “”,表示不取证书链
StrCertChainPa	证书链密码	In	

ss			

返回：

S_OK	正常返回,
S_FALSE	错误。 错误代码见错误代码表

清除环境

HRESULT ESE_ClearSession ()
功能: 清除环境变量.
在程序结束前应调用此方法.
返回：

S_OK	正常返回,
S_FALSE	错误。 错误代码见错误代码表

数字签名

签名

HRESULT ESE_SignData(BSTR strOrgData, BSTR alg, [out, retval] BSTR *strSignedData)
参数：

参数名	含义	In/out	参数选项
strOrgData	原始数据块	In	
alg	摘要和签名算法	In	参见头文件定义
strSignedData	处理后数据块	Out，ret	

返回：

S_OK	正常返回,
S_FALSE	错误。 错误代码见错误代码表

签名并编码成 PKCS7 格式

HRESULT ESE_SignData_P7（BSTR strOrgData, BSTR SignMethod, BSTR strCertificate, [out, retval] BSTR *strP7SignedData）
说明：签名并编码成 PKCS7 格式
参数：

参数名	含义	In/out	参数选项
strOrgData	原始数据块	In	
SignMethod	摘要和签名算法	In	参见头文件定义
strCertificate	证书内容	In	
strP7SignedData	处理后数据块	Out,ret	

返回：

S_OK	正常返回,
S_FALSE	错误。 错误代码见错误代码表

注意：对于 RSA 签名，需要增加将签名结果打包成 P7 的处理。

验证签名

HRESULT ESE_VerifySignData (BSTR strOrgData, BSTR SignMethod, BSTR strSignedData, BSTR strCertificate)

参数：

参数名	含义	In/out	参数选项
strOrgData	原始数据块	In	
SignMethod	摘要签名算法	In	参见头文件定义
strSignedData	签名数据块	In	
strCertificate	证书内容	In	

返回：

S_OK	正常返回,
S_FALSE	错误。 错误代码见错误代码表

解 PKCS7 并验证签名

HRESULT ESE_VerifySignData_P7(BSTR strP7SignedData, [out, retval] BSTR * strOrgData)

说明：验证签名后输出原文和证书。

参数：

参数名	含义	In/out	参数选项
strP7SignedData	P7 数据包	In	
strOrgData	原文	Out,ret	

返回：

S_OK	正常返回,
S_FALSE	错误。 错误代码见错误代码表

注意：对于 RSA 签名，需要增加解析 P7 的处理。

证书

从介质中获取证书

HRESULT ESE_GetSelfCertificate((long DevType, BSTR strDevParam, [out, retval] BSTR * strCertificate)

功能：从设备中读取证书，证书的来源可以是 IC 卡，磁盘。

参数：

参数名	含义	In/out	参数选项
DevType	设备类型	in	见设备类型说明文件
strDevParam	设备参数	In	可为”com1”, “com2”
strCertificate	证书内容	Out,ret	

返回：

S_OK	正常返回,
S_FALSE	错误。 错误代码见错误代码表

通过黑名单验证证书

HRESULT ESE _VerifyCertificate(BSTR strCertificate)

功能：验证证书有效性，先验有效期，ca 签发，根据配置决定是否验证本地黑名单，ca 黑名单。配置方式见 SE_SetConfiguration 说明。

参数：

参数名	含义	In/out	参数选项
strCertificate	证书内容	In	证书内容

返回：

S_OK	正常返回,
S_FALSE	错误。 错误代码见错误代码表

Demo 示例

SM2

摘取部分代码，供参考。具体内容详见 demo 示例


```
//初始化参数
SafeEngineCtl.ESE_InitialSession(41,"com1", usbkeyPassword, 0, 41,
 "com1", "");
if(SafeEngineCtl.ErrorCode!=0)
 {
        alert("ESE_InitialSession Error. Return:\n" +
 GetErrCode(SafeEngineCtl.ErrorCode));
        SafeEngineCtl.ESE_ClearSession();
        return;
 }

 if(!confirm("ESE_InitialSession ok\n 初始化成功."))
        {
        return;
        }

//证书验证
var strCert="XXXXXXX";
 SafeEngineCtl.ESE_VerifyCertificate(strCert);
 if(SafeEngineCtl.ErrorCode!=0)
 {
        alert("ESE_VerifyCertificate Error. Return:\n" +
 GetErrCode(SafeEngineCtl.ErrorCode));
        SafeEngineCtl.ESE_ClearSession();
        return;
 }
 if(!confirm("ESE_VerifyCertificate ok.函数成功\n"))
        {
        return;
        }
```

```
//OCSP 验证
SafeEngineCtl.ESE_VerifyCertificateOnline(strCert);
if(SafeEngineCtl.ErrorCode!=0)
{
        alert("ESE_VerifyCertificateOnline Error. Return:" +
SafeEngineCtl.ErrorCode);
        SafeEngineCtl.ESE_ClearSession();
}
if(!confirm("ESE_VerifyCertificateOnline ok.\n 证书验证函数成功\n"))
{
        return;
}

//验证签名

strRandom 签名原文（详见海关报文原文规则）、strSigned 海关签名值、
 strCert 海关证书
SafeEngineCtl.ESE_VerifySignData(strRandom, "", strSigned,
strCert);
if(SafeEngineCtl.ErrorCode!=0)
{
        alert("ESE_VerifySignData Error. Return:\n" +
GetErrCode(SafeEngineCtl.ErrorCode));
        SafeEngineCtl.ESE_ClearSession();
        return;
}
if(!confirm("ESE_VerifySignData ok.\n 验证签名函数成功\n 原文:"+
strRandom))
        {
        return;
        }
```

RSA

摘取部分代码，供参考。具体内容详见 demo 示例

SafeEngineEccCtl(证书验证_RSA).htm

```
//初始化函数
 SafeEngineCtl.SEH_InitialSession(10,"com1", strpassword, 0, 10,
 "com1", "");
 if(SafeEngineCtl.ErrorCode!=0)
 {
        alert("SEH_InitialSession Error. Return:\n" +
 GetErrCode(SafeEngineCtl.ErrorCode));
        return;
 }

 if(!confirm("SEH_InitialSession ok\n 初始化成功."))
        {
        return;
        }


//验证证书
var strCert="XXXXXXXXXX";
SafeEngineCtl.SEH_VerifyCertificate(strCert);
 if(SafeEngineCtl.ErrorCode!=0)
 {
        alert("SEH_VerifyCertificate Error. Return:\n" +
 GetErrCode(SafeEngineCtl.ErrorCode));
        SafeEngineCtl.SEH_ClearSession();
        return;
 }
 if(!confirm("SEH_VerifyCertificate ok.\n 证书验证函数成功\n"))
        {
        return;
        }
```

```
//验证签名
strRandom 签名原文（详见海关报文原文规则）、strSigned 海关签名值、
strCert 海关证书
SafeEngineCtl.SEH_VerifySignData(strRandom, 3, strSigned, strCert);

if(SafeEngineCtl.ErrorCode!=0)
{
        alert("SEH_VerifySignData Error. Return:\n" +
GetErrCode(SafeEngineCtl.ErrorCode));
        SafeEngineCtl.SEH_ClearSession();
        return;
}
if(!confirm("SEH_VerifySignData ok.\n 验证签名函数成功\n 原文:"+
strRandom))
        {
        return;
        }
```