ICS 35.040
L 80
备案号：38317—2013

GM

中华人民共和国密码行业标准

GM/T 0019—2012

通用密码服务接口规范

Universal cryptography service interface specification

2012-11-22 发布　　2012-11-22 实施

国家密码管理局　发布

中国防伪315产品信息查询系统
手机发送防伪码至13720082315
网站www.cntw315.com
电话4006982315
刮涂层 查真伪

目　次

前　言

本标准依据 GB/T 1.1—2009 给出的规则起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

本标准的附录 A 为规范性附录。

本标准由国家密码管理局提出并归口。

本标准起草单位：北京数字认证股份有限公司、上海格尔软件股份有限公司、北京海泰方圆科技有限公司、无锡江南信息安全工程技术中心、上海数字证书认证中心有限公司、卫士通信息产业股份有限公司、山东得安信息技术有限公司、国家信息安全工程技术研究中心。

本标准主要起草人：刘平、李述胜、谭武征、柳增寿、徐强、刘承、李元正、高志权、孔凡玉、袁峰。

本标准凡涉及密码算法相关内容，按照国家有关法规实施。

引　言

本标准依托于密码设备层的GM/T 0018《密码设备应用接口规范》和GM/T 0016《智能密码钥匙密码应用接口规范》，为典型密码服务层和应用层规定了统一的通用密码服务接口。

通用密码服务接口在公钥密码基础设施支撑的前提下，向应用系统和典型密码服务层提供各类通用的密码服务，有利于密码服务接口产品的开发，有利于应用系统在密码服务过程中的集成和实施，有利于实现各应用系统的互联互通。

通用密码服务接口规范

1 范围

本标准规定了统一的通用密码服务接口。

本标准适用于公开密钥应用技术体系下密码应用服务的开发，密码应用支撑平台的研制及检测，也可用于指导直接使用密码设备的应用系统的开发。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

GM/T 0006 密码应用标识规范

GM/T 0015 基于SM2密码算法的数字证书格式规范

GM/T 0018 密码设备应用接口规范

GM/T 0016 智能密码钥匙密码应用接口规范

GM/T 0010 SM2密码算法加密签名消息语法规范

GM/T 0009 SM2密码算法使用规范

PKCS #7:Cryptographic Message Syntax

3 术语和定义

下列术语和定义适用于本文件。

3.1

数字证书 digital certificate

由认证权威数字签名的包含公开密钥拥有者信息、公开密钥、签发者信息、有效期以及一些扩展信息的数字文件。

3.2

用户密钥 user key

存储在设备内部的用于应用密码运算的非对称密钥对，包含签名密钥对和加密密钥对。

3.3

容器 container

密码设备中用于保存密钥所划分的唯一性存储空间。

4 符号和缩略语

下列缩略语适用于本文件：

API Application Program Interface 应用程序接口，简称应用接口

CA Certification Authority 证书认证机构

CN Common Name 通用名

CRL Certificate Revocation List 证书撤销列表
DER Distinguished Encoding Rules 可区分编码规则
DN Distinguished Name 可识别名
ECC Elliptic Curve Cryptography 椭圆曲线密码
LDAP Lightweight Directory Access Protocol 轻量级目录访问协议
OID Object Identifier 对象标识符
PKCS the Public-Key Cryptography Standard 公钥密码标准

5 算法标识和数据结构

5.1 算法标识与常量定义

本规范所使用常量定义、各类算法标识和证书解析标识的具体定义见 GM/T 0006。

5.2 密码服务接口数据结构定义和说明

5.2.1 常量定义

数据常量标识定义了在规范中用到的常量的取值。
数据常量标识的定义如表 1 所示。

表 1 常量定义

常量名	取值	描述
SGD_MAX_COUNT	64	枚举出的对象数量最大值
SGD_MAX_NAME_SIZE	256	证书某项信息的字符串长度最大值

5.2.2 用户证书列表(见表 2)

表 2 用户证书列表

字段名称	数据长度(字节)	含义
certCount	4	证书总数
Certificate	4	DER 编码的数字证书
certificateLen	4	数字证书的长度
containerName	4	容器名称
containerNameLen	4	容器名称的长度
keyUsage	4	密钥用途

实际数据结构定义：

```
typedef struct SGD_USR_CERT_ENUMLIST_ {
    unsigned int certCount;
    unsigned char * certificate[SGD_MAX_COUNT];
    unsigned int certificateLen[SGD_MAX_COUNT];
    unsigned char * containerName[SGD_MAX_COUNT];
    unsigned int containerNameLen[SGD_MAX_COUNT];
```

```
    unsigned int keyUsage[SGD_MAX_COUNT];
} SGD_USR_CERT_ENUMLIST;
```

5.2.3 密钥容器信息列表(见表3)

表3 密钥容器信息列表

字段名称	数据长度(字节)	含义
keyPairCount	4	密钥容器信息总数
containerName	256	容器名称
containerNameLen	256	容器名称的长度
keyUsage	4	密钥用途:1:加密;2:签名;3:密钥交换
keyType	4	密钥类型:1:SM2;2:RSA1024;3:RSA2048;4:RSA3072;5:RSA4096

实际数据结构定义:

```
typedef struct SGD_ KEYCONTAINERINFO_ENUMLIST_ {
    unsigned int keyPairCount;
    unsigned char * containerName[SGD_MAX_COUNT];
    unsigned int containerNameLen[SGD_MAX_COUNT];
    unsigned int keyUsage[SGD_MAX_COUNT];
    unsigned int keyType[SGD_MAX_COUNT];
} SGD_ KEYCONTAINERINFO_ENUMLIST;
```

5.2.4 证书中DN的结构(见表4)

表4 证书中DN的结构

字段名称	数据长度(字节)	含义
dn_c	256	国家名称
dn_c_len	1	国家名称的长度
dn_s	256	省份或直辖市名称
dn_s_len	1	省份或直辖市名称的长度
dn_l	256	城市或地区的名称
dn_l_len	1	城市或地区的名称的长度
dn_o	256	机构名称数组
dn_o_len	20	机构名称数组的长度
dn_ou	256	机构单位名称数组
dn_ou_len	20	机构单位名称数组的长度
dn_cn	256	证书拥有者名称数组
dn_cn_len	8	证书拥有者名称数组的长度
dn_email	256	电子邮件数组
dn_email_len	8	电子邮件数组的长度

实际数据结构定义：

```
typedef struct{
    unsigned char dn_c [SGD_MAX_NAME_SIZE];
    unsigned char dn_c_len[1];
    unsigned char dn_s [SGD_MAX_NAME_SIZE];
    unsigned char dn_s_len[1];
    unsigned char dn_l [SGD_MAX_NAME_SIZE];
    unsigned char dn_l_len[1];
    unsigned char dn_o[5][SGD_MAX_NAME_SIZE];
    unsigned int dn_o_len[5];
    unsigned char dn_ou[5][SGD_MAX_NAME_SIZE];
    unsigned int dn_ou_len[5];
    unsigned char dn_cn[2][SGD_MAX_NAME_SIZE];
    unsigned int dn_cn_len[2];
    unsigned char dn_email[2][SGD_MAX_NAME_SIZE];
    unsigned int dn_email_len[2];
}SGD_NAME_INFO;
```

6 密码服务接口

6.1 通用密码服务接口在公钥密码基础设施应用技术体系框架中的位置

通用密码服务通过统一的密码服务接口，向典型密码服务层和应用层提供证书解析、证书认证、信息的机密性、完整性和不可否认性等通用密码服务，将上层应用的密码服务请求转化为具体的基础密码操作请求，通过统一的密码设备应用接口调用相应的密码设备实现具体的密码运算和密钥操作。

通用密码服务在公钥密码基础设施应用技术体系框架内的位置如图 1 所示。

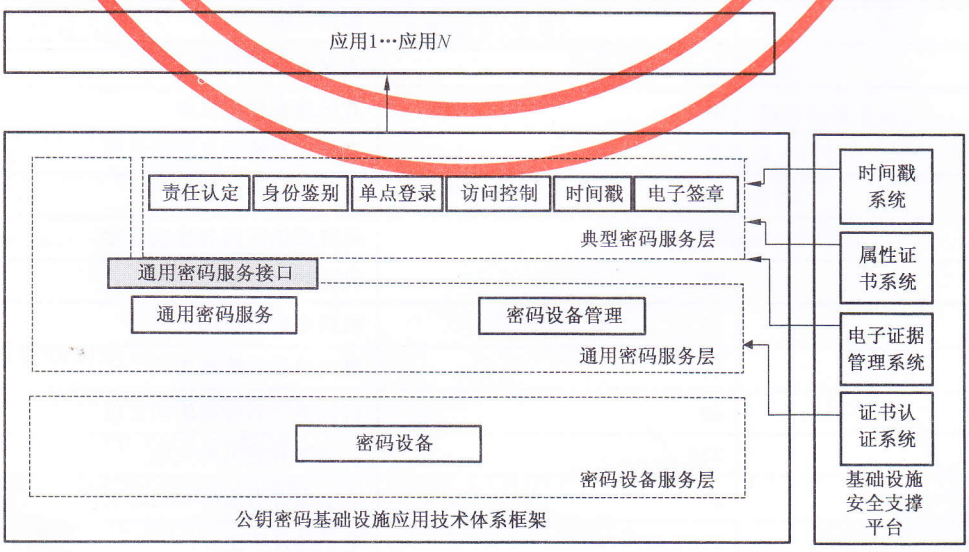

图 1 通用密码服务接口在公钥密码基础设施应用技术体系框架内的位置

6.2 密码服务接口组成和功能说明

6.2.1 概述

通用密码服务接口由以下部分组成：

a) 环境类函数；

b) 证书类函数；

c) 密码运算类函数；

d) 消息类函数。

6.2.2 环境类函数

环境类函数负责创建和管理程序空间、负责创建和管理安全程序空间中所需的各种资源、信号，并确保安全程序空间在应用程序运行期间不会被非法访问，造成信息泄漏。环境类函数负责完成与密码设备的安全连接，确保后续的安全操作是在安全、可信的程序空间中进行。环境类函数还负责在用户与密码设备之间创建和管理安全访问令牌。可创建两种类型的用户安全访问令牌，一类是普通用户，该类型的安全访问令牌标识该用户是普通用户，只能访问密码设备中属于自己的信息和数据；另一类是管理员，该类型的安全访问令牌标识该用户是管理员，可以管理普通用户的安全令牌。

应用程序在使用密码服务接口时，首先要调用初始化环境函数(SAF_Initialize)创建和初始化安全的应用程序空间，完成与密码设备连接和初始化工作。在中止应用程序之前，应调用清除环境函数(SAF_Finalize)，中止与密码设备的连接，销毁所创建的安全程序空间，防止由于内存残留所带来的安全风险。应用程序在调用任何密码服务函数，进行任何密码运算之前应首先调用用户登录函数(SAF_Login)，建立安全访问令牌。建立了安全访问令牌后，则可以调用任何密码服务函数。当不再调用任何密码服务函数时，应调用注销登录函数(SAF_Logout)注销安全访问令牌，确保密码设备不被非法访问。

6.2.3 证书类函数

证书类函数设置各类数字证书到应用接口会话环境中、验证用户证书和获取数字证书或 CRL，提供包括证书获取、CRL 获取、CA 根证书设置、用户证书验证和用户证书信息获取等一系列具体函数。应用程序通过调用证书函数，实现基于数字证书的身份认证，从证书中获取有关信息，实现授权管理、访问控制等安全机制。本标准中涉及的数字证书格式应遵循 GM/T 0015。

6.2.4 密码运算类函数

密码运算类函数负责具体与密码设备交互实现具体的密码运算，并将密码运算后的结果返回给应用程序，是应用程序实现数据保密性、完整性和不可抵赖性等安全机制的基础。

密码运算类函数提供包括 base64 编解码、随机数生成、数字摘要以及各种对称和非对称密码运算等。密码服务函数支持定长数据和不定长数据的密码运算，对于定长数据可以直接调用相关函数进行处理；对于不定长数据，需要先创建相应的密码运算对象，然后调用相应的函数对数据进行持续处理。当数据处理完时，要调用相应的函数表示数据处理完，最后要调用相应函数销毁相应的密码运算对象。

6.2.5 消息类函数

消息类函数主要是将数据按照 PKCS#7 格式进行封装，实现数据封装格式与应用系统无关性，实现应用系统互联互通和信息共享。

消息类函数提供了 PKCS#7 格式的数据编解码、PKCS#7 格式的签名数据编解码和 PKCS#7 格式的数字信封编解码，能够非常方便应用程序实现身份认证、保密性、完整性和不可否认性等安全措施。

7 密码服务接口函数定义

7.1 环境类函数

7.1.1 概述

环境类函数包括以下具体函数，各函数返回值见附录 A 错误代码定义：

a) 初始化环境：SAF_Initialize
b) 清除环境：SAF_Finalize
c) 获取接口版本信息：SAF_GetVersion
d) 用户登录：SAF_Login
e) 修改 PIN：SAF_ChangePin
f) 注销登录：SAF_Logout

7.1.2 初始化环境

原型：int SAF_Initialize(void ** phAppHandle,char * pucCfgFilePath);
描述：初始化密码服务程序空间。
参数：phAppHandle[in/out] 输入并返回应用接口句柄
pucCfgFilePath[in] 配置文件全路径名，配置信息自定义，建议包括：密码设备类型、密码设备动态库名称、设备的配置文件、应用策略等。
返回值：0 成功
非 0 失败，返回错误代码

7.1.3 清除环境

原型：int SAF_Finalize(void * hAppHandle);
描述：清除应用程序空间。
参数：hAppHandle[in] 应用接口句柄
返回值：0 成功
非 0 失败，返回错误代码

7.1.4 获取接口版本信息

原型：int SAF_GetVersion(unsigned int * puiVersion)
描述：取接口对应标准的版本号。
参数：puiVersion [out] 版本号
返回值：0 成功
非 0 失败，返回错误代码
备注：版本号的格式为：0xAABBCCCC，其中 AA 为主版本号，BB 为次版本号，CCCC 为修改号。

7.1.5 用户登录

原型：int SAF_Login (
void * hAppHandle,

```
        unsigned int uiUsrType,
        unsigned char * pucContainerName,
        unsigned int uiContainerNameLen,
        unsigned char * pucPin,
        unsigned int uiPinLen,
        unsigned int * puiRemainCount)
```

描述：用户登录密码设备，建立安全令牌。

参数：

hAppHandle[in]	应用接口句柄
uiUsrType[in]	用户类型，当为 0 时表示管理员登录，为 1 时表示用户登录
pucContainerName [in]	容器名或密钥检索号
uiContainerNameLen[in]	容器名或密钥检索号的长度
pucPin[in]	设备口令
uiPinLen[in]	设备口令长度
puiRemainCount[out]	口令剩余重试次数

返回值：

0	成功
非 0	失败，返回错误代码

备注：在服务器端使用加密机或加密卡时，pucContainerName 用于标识密码设备内部的密钥位置。客户端使用智能密码钥匙或智能 IC 卡时，pucContainerName 用于标识指定的容器名。

7.1.6 修改 PIN

原型：

```
int SAF_ChangePin (
        void * hAppHandle,
        unsigned int uiUsrType,
        unsigned char * pucContainerName,
        unsigned int uiContainerNameLen,
        unsigned char * pucOldPin,
        unsigned int uiOldPinLen,
        unsigned char * pucNewPin,
        unsigned int uiNewPinLen,
        unsigned int * puiRemainCount);
```

描述：修改设备 PIN，本函数仅对 USBkey 或 IC 设备等客户端密码设备有用。

参数：

hAppHandle[in]	应用接口句柄
uiUsrType[in]	用户类型，当为 0 时表示管理员登录，为 1 时表示用户登录
pucContainerName [in]	容器名或密钥检索号
uiContainerNameLen[in]	容器名或密钥检索号的长度
pucOldPin[in]	设备当前口令
uiOldPinLen[in]	设备当前口令长度
pucNewPin[in]	设备新口令
uiNewPinLen[in]	设备新口令长度
puiRemainCount[out]	口令剩余重试次数

返回值：　0　　成功
　　非 0　　失败，返回错误代码

7.1.7　注销登录

原型：　int SAF_Logout(
　　void　* hAppHandle,
　　unsigned int uiUsrType)

描述：　设备注销登录。

参数：　hAppHandle[in]　　应用接口句柄
　　uiUsrType [in]　　用户类型，当为 0 时表示管理员登录，为 1 时表示用户登录

返回值：　0　　成功
　　非 0　　失败，返回错误代码

7.2　证书类函数

7.2.1　概述

证书类函数包括以下具体函数，各函数返回值见附录 A 错误代码定义：

a)　添加根 CA 证书：SAF_AddTrustedRootCaCertificate
b)　获取根 CA 证书个数：SAF_GetRootCaCertificateCount
c)　获取根 CA 证书：SAF_GetRootCaCertificate
d)　删除根 CA 证书：SAF_RemoveRootCaCertificate
e)　添加 CA 证书：SAF_AddCaCertificate
f)　获取 CA 证书个数：SAF_GetCaCertificateCount
g)　获取 CA 证书：SAF_GetCaCertificate
h)　删除 CA 证书：SAF_RemoveCaCertificate
i)　添加 CRL：SAF_AddCrl
j)　验证用户证书：SAF_VerifyCertificate
k)　根据 CRL 文件获取用户证书注销状态：SAF_VerifyCertificateByCrl
l)　根据 OCSP 获取证书状态：SAF_GetCertificateStateByOCSP
m)　通过 LDAP 方式获取证书：SAF_GetCertificateFromLdap
n)　通过 LDAP 方式获取证书对应的 CRL：SAF_GetCrlFromLdap
o)　取证书信息：SAF_GetCertificateInfo
p)　取证书扩展信息：SAF_GetExtTypeInfo
q)　列举用户证书：SAF_EnumCertificates
r)　列举用户的密钥容器信息：SAF_EnumKeyContainerInfo
s)　释放列举用户证书的内存：SAF_EnumCertificatesFree
t)　释放列举密钥容器信息的内存：SAF_EnumkeyContainerInfoFree

7.2.2　添加信任的 CA 根证书

原型：　int SAF_AddTrustedRootCaCertificate(
　　void　* hAppHandle,
　　unsigned char　* pucCertificate,

unsigned int uiCertificateLen);

描述：添加信任的 CA 根证书。

参数：

hAppHandle[in]	应用接口句柄
pucCertificate[in]	DER 编码的证书
uiCertificateLen[in]	证书长度

返回值：

0	成功
非 0	失败，返回错误代码

备注：本函数仅限于管理员使用。

7.2.3 获取根 CA 证书个数

原型：
```
int SAF_GetRootCaCertificateCount(
    void *hAppHandle,
    unsigned int *puiCount);
```

描述：获取信任根 CA 证书的个数。

参数：

hAppHandle[in]	应用接口句柄
puiCount[out]	信任根 CA 证书个数

返回值：

0	成功
非 0	失败，返回错误代码

7.2.4 获取根 CA 证书

原型：
```
int SAF_GetRootCaCertificate(
    void *hAppHandle,
    unsigned int uiIndex,
    unsigned char *pucCertificate,
    unsigned int *puiCertificateLen);
```

描述：获取指定位置 CA 根证书。

参数：

hAppHandle[in]	应用接口句柄
uiIndex[in]	根证书索引
pucCertificate[out]	DER 编码的证书
puiCertificateLen[in,out]	证书长度

返回值：

0	成功
非 0	失败，返回错误代码

7.2.5 删除根 CA 证书

原型：
```
int SAF_RemoveRootCaCertificate(
    void *hAppHandle,
    unsigned int uiIndex);
```

描述：删除指定位置 CA 根证书。

参数：

hAppHandle[in]	应用接口句柄
uiIndex[in]	根证书索引

返回值：

0	成功
非 0	失败，返回错误代码

备注：本函数仅限于管理员使用。

7.2.6 添加 CA 证书

原型：
```
int SAF_AddCaCertificate(
    void * hAppHandle,
    unsigned char * pucCertificate,
    unsigned int uiCertificateLen);
```
描述：添加 CA 证书。

参数：		
	hAppHandle[in]	应用接口句柄
	pucCertificate[in]	DER 编码的证书
	uiCertificateLen[in]	证书长度
返回值：	0	成功
	非 0	失败，返回错误代码

备注：本函数仅限于管理员使用。

7.2.7 获取 CA 证书个数

原型：
```
int SAF_GetCaCertificateCount(
    void * hAppHandle,
    unsigned int * puiCount);
```
描述：获取信任 CA 证书的个数。

参数：		
	hAppHandle[in]	应用接口句柄
	puiCount[out]	信任 CA 证书个数
返回值：	0	成功
	非 0	失败，返回错误代码

7.2.8 获取 CA 证书

原型：
```
int SAF_GetCaCertificate(
    void * hAppHandle,
    unsigned int uiIndex,
    unsigned char * pucCertificate,
    unsigned int * puiCertificateLen);
```
描述：获取指定位置 CA 证书。

参数：		
	hAppHandle[in]	应用接口句柄
	uiIndex[in]	CA 证书索引
	pucCertificate[out]	DER 编码的证书
	puiCertificateLen[in,out]	证书长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.2.9 删除 CA 证书

原型：
```
int SAF_RemoveCaCertificate(
    void * hAppHandle,
    unsigned int uiIndex);
```
描述：删除指定位置 CA 证书。

	hAppHandle[in]	应用接口句柄
参数：	uiIndex[in]	证书位置索引
返回值：	0	成功
	非 0	失败，返回错误代码
备注：	本函数仅限于管理员使用。	

7.2.10 添加 CRL

原型：
```
int SAF_AddCrl(
    void *hAppHandle,
    unsigned char *pucDerCrl,
    unsigned int uiDerCrlLen);
```

描述：添加 CRL。

参数：	hAppHandle[in]	应用接口句柄
	pucDerCrl[in]	DER 编码的 CRL
	uiDerCrlLen[in]	DER 编码 CRL 的长度
返回值：	0	成功
	非 0	失败，返回错误代码
备注：	本函数仅限于管理员使用。	

7.2.11 验证用户证书

原型：
```
int SAF_VerifyCertificate(
    void *hAppHandle,
    unsigned char *pucUsrCertificate,
    unsigned int uiUsrCertificateLen);
```

描述：验证用户证书的有效性，包括验证有效期、证书信任列表、吊销状态等。

参数：	hAppHandle[in]	应用接口句柄
	pucUsrCertificate[in]	DER 编码的证书
	uiUsrCertificateLen[in]	证书长度
返回值：	0	成功
	非 0	失败，返回错误代码，详见附录 A

7.2.12 根据 CRL 文件获取用户证书注销状态

原型：
```
int SAF_VerifyCertificateByCrl(
    void *hAppHandle,
    unsigned char *pucUsrCertificate,
    unsigned int uiUsrCertificateLen,
    unsigned char *pucDerCrl,
    unsigned int uiDerCrlLen);
```

描述：根据 CRL 文件验证用户证书是否被吊销。

参数：	hAppHandle[in]	应用接口句柄
	pucUsrCertificate[in]	DER 编码的证书
	uiUsrCertificateLen[in]	证书长度
	pucDerCrl[in]	DER 编码的 CRL

	uiDerCrlLen	CRL 长度
返回值：	0	成功，用户证书有效
	其他	失败，返回附录 A 的错误代码

7.2.13 根据 OCSP 获取证书状态

原型：
```
int SAF_GetCertificateStateByOCSP(
    void * hAppHandle,
    unsigned char * pcOcspHostURL,
    unsigned int uiOcspHostURLLen,
    unsigned char * pucUsrCertificate,
    unsigned int uiUsrCertificateLen,
    unsigned char * pucCACertificate,
    unsigned char uiCACertificateLen);
```

描述：从 OCSP 获取用户证书的实时状态。

参数：

hAppHandle[in]	应用接口句柄
pcOcspHostURL [in]	ocsp 服务的 URL
uiOcspHostURLLen [in]	URL 长度
pucUsrCertificate[in]	DER 编码的用户证书
uiUsrCertificateLen[in]	证书长度
pucCACertificate[in]	DER 编码的颁发者证书
uiCACertificateLen[in]	颁发者证书长度

返回值：

0	成功，用户证书有效
其他	失败，返回错误代码，详见附录 A

7.2.14 通过 LDAP 方式获取证书

原型：
```
int SAF_GetCertFromLdap(
    void * hAppHandle,
    char * pcLdapHostURL,
    unsigned int uiLdapHostURLLen,
    unsigned char * pucQueryDN,
    unsigned int uiQueryDNLen,
    unsigned char * pucOutCert,
    unsigned int * puiOutCertLen);
```

描述：通过 LDAP 方式获取证书。

参数：

hAppHandle[in]	应用接口句柄
pcLdapHostURL [in]	ldap 服务器 IP 地址
uiLdapHostURLLen [in]	ldap 服务器端口
pucQueryDN [in]	需要查找的证书的查询条件
uiQueryDNLen [in]	查询条件长度
pucOutCert[out]	查询到的 base64 编码的证书，如查询出多证书，则证书信息中间以“&”分割
puiOutCertLen[in,out]	找到的证书长度

返回值：

0	成功

	非 0	失败，返回错误代码

7.2.15 通过 LDAP 方式获取证书对应的 CRL

原型：
```
int SAF_GetCrlFromLdap (
    void * hAppHandle,
    char * pcLdapHostURL,
    unsigned int uiLdapHostURLLen,
    unsigned char * pucCertificate,
    unsigned int uiCertificateLen,
    unsigned char * pucCrlData,
    unsigned int * puiCrlDataLen);
```

描述：通过 LDAP 方式根据证书获取对应的 CRL。

参数：

hAppHandle[in]	应用接口句柄
pcLdapHostURL [in]	ldap 服务 URL
uiLdapHostURLLen [in]	URL 长度
pucCertificate [in]	DER 编码的数字证书
uiCertificateLen [in]	证书长度
pucCrlData [out]	获取的 DER 编码的 CRL 数据
puiCrlDataLen [in,out]	CRL 数据长度

返回值：

0	成功
非 0	失败，返回错误代码

7.2.16 取证书信息

原型：
```
int SAF_GetCertificateInfo(
    void * hAppHandle,
    unsigned char * pucCertificate,
    unsigned int uiCertificateLen,
    unsigned int uiInfoType,
    unsigned char * pucCnfo,
    unsigned int * puiInfoLen);
```

描述：解析证书，获取证书中的信息。

参数：

hAppHandle[in]	应用接口句柄
pucCertificate[in]	DER 编码的证书
uiCertificateLen[in]	证书长度
uiInfoType[in]	指定的证书解析标识，详见 GM/T 0006
pucInfo[out]	获取的证书信息
puiInfoLen[in,out]	获取的证书信息长度

返回值：

0	成功
非 0	失败，返回错误代码

7.2.17 取证书扩展信息

原型：
```
int SAF_GetExtTypeInfo (
    void * hAppHandle,
```

```
    unsigned char * pucDerCert,
    unsigned int uiDerCertLen,
    unsigned int uiInfoType,
    unsigned char * pucPriOid,
    unsigned int uiPriOidLen,
    unsigned char * pucInfo,
    unsigned int * puiInfoLen);
```

描述：　获取证书的扩展信息。

参数：

hAppHandle[in]	应用接口句柄
pucDerCert[in]	Der 格式的数字证书
uiDerCertLen[in]	数字证书长度
uiInfoType[in]	指定的证书扩展项解析标识，详见 GM/T 0006
pucPriOid[in]	扩展项的 OID，如果不是私有扩展项类型，该参数无效
uiPriOidLen[in]	扩展项 OID 长度，如果不是私有扩展项类型，该参数无效
pucInfo[out]	获取的证书信息
puiInfoLen[in,out]	获取的证书信息长度

返回值：

0	成功
非 0	失败，返回错误代码

7.2.18　列举用户证书

```
原型：　int SAF_EnumCertificates(
            void * hAppHandle,
            SGD_USR_CERT_ENUMLIST * usrCerts);
```

描述：　列举出插入当前计算机的所有密码设备内的有效证书。

参数：

hAppHandle[in]	应用接口句柄
usrCerts[out]	返回的用户证书列表

返回值：

0	成功
非 0	失败，返回错误代码

备注：　usrCerts 结构内的对象数据由本函数分配空间。本函数用于客户端枚举证书。

7.2.19　列举用户的密钥容器信息

```
原型：　Int SAF_EnumKeyContainerInfo(
            void * hAppHandle,
            SGD_ KEYCONTAINERINFO_ENUMLIST * keyContainerInfo);
```

描述：　列举插入当前计算机的所有密码设备的容器信息。

参数：

hAppHandle[in]	应用接口句柄
keyContainerInfo [out]	返回的密钥容器信息

返回值：

SAR_OK	成功
其他	失败

备注：　keyContainerInfo 结构内的对象数据由本函数分配空间。仅在客户端使用。

7.2.20 释放列举用户证书的内存

原型：int SAF_EnumCertificatesFree(
    void * hAppHandle,
    SGD_USR_CERT_ENUMLIST * usrCerts);

描述：释放 SAF_EnumCertificates 函数中分配的内存。

参数：		
	hAppHandle[in]	应用接口句柄
	usrCerts[in]	证书信息
返回值：	0	成功
	非 0	失败，返回错误代码

7.2.21 释放列举密钥容器信息的内存

原型：int SAF_EnumkeyContainerInfoFree(
    void * hAppHandle,
    SGD_ KEYCONTAINERINFO_ENUMLIST * keyContainerInfo);

描述：释放 SAF_EnumKeyContainerInfo 函数中分配的内存。

参数：		
	hAppHandle[in]	应用接口句柄
	usrKeyPairs [in]	密钥容器信息
返回值：	0	成功
	非 0	失败，返回错误代码

7.3 密码运算类函数

7.3.1 概述

密码运算类函数包括以下具体函数，各函数返回值见附录 A 错误代码定义：

a) 单块 base64 编码：SAF_Base64_Encode
b) 单块 base64 解码：SAF_Base64_Decode
c) 创建 base64 对象：SAF_Base64_CreateBase64Obj
d) 销毁 base64 对象：SAF_Base64_DestroyBase64Obj
e) 通过 base64 对象继续编码：SAF_Base64_EncodeUpdate
f) 通过 base64 对象编码结束：SAF_Base64_EncodeFinal
g) 通过 base64 对象继续解码：SAF_Base64_DecodeUpdate
h) 通过 base64 对象解码结束：SAF_Base64_DecodeFinal
i) 生成随机数：SAF_GenRandom
j) HASH 运算：SAF_Hash
k) 创建 HASH 对象：SAF_CreateHashObj
l) 删除 HASH 对象：SAF_DestroyHashObj
m) 通过对象多块 HASH 运算：SAF_HashUpdate
n) 结束 HASH 运算：SAF_HashFinal
o) 生成 RSA 密钥对：SAF_GenRsaKeyPair
p) 获取 RSA 公钥：SAF_GetPublicKey
q) RSA 签名运算：SAF_RsaSign
r) 对文件进行 RSA 签名运算：SAF_RsaSignFile

s) RSA 验证签名运算:SAF_RsaVerifySign
t) 对文件及其签名进行 RSA 验证:SAF_RsaVerifySignFile
u) 基于证书的 RSA 公钥验证:SAF_VerifySignByCert
v) 生成 ECC 密钥对:SAF_GenEccKeyPair
w) 获取 ECC 公钥:SAF_GetEccPublicKey
x) ECC 签名:SAF_EccSign
y) ECC 验证:SAF_EccVerifySign
z) ECC 公钥加密:SAF_EccPublicKeyEnc
aa) 基于证书的 ECC 公钥加密:SAF_ EccPublicKeyEncByCert
bb) 基于证书的 ECC 公钥验证:SAF_ EccVerifySignByCert
cc) 创建对称算法对象:SAF_CreateSymmAlgoObj
dd) 生成会话密钥并用外部公钥加密输出:SAF GenerateKeyWithEPK
ee) 导入加密的会话密钥:SAF_ImportEncedKey
ff) 生成密钥协商参数并输出:SAF_GenerateAgreementDataWithECC
gg) 计算会话密钥:SAF_GenerateKeyWithECC
hh) 产生协商数据并计算会话密钥:SAF_GenerateAgreementDataAndKeyWithECC
ii) 销毁对称算法对象:SAF_DestroySymmAlgoObj
jj) 销毁会话密钥句柄:SAF_DestroyKeyHandle
kk) 单块加密运算:SAF_SymmEncrypt
ll) 多块加密运算:SAF_SymmEncryptUpdate
mm) 结束加密运算:SAF_SymmEncryptFinal
nn) 单块解密运算:SAF_SymmDecrypt
oo) 多块解密运算:SAF_SymmDecryptUpdate
pp) 结束解密运算:SAF_SymmDecryptFinal
qq) 单组数据消息鉴别码运算:SAF_Mac
rr) 多组数据消息鉴别码运算:SAF_MacUpdate
ss) 结束消息鉴别码运算:SAF_MacFinal

7.3.2 单块 base64 编码

原型:
```
int SAF_Base64_Encode(
    unsigned char *pucInData,
    unsigned int puiInDataLen,
    unsigned char *pucOutData,
    unsigned int *puiOutDataLen);
```

描述: 对输入的数据进行 base64 编码。

参数:

	pucInData[in]	编码前的数据
	puiInDataLen[in]	编码前的数据长度
	pucOutData[out]	编码后的数据
	puiOutDataLen[in,out]	编码后的数据长度

返回值:

0	成功
非 0	失败,返回错误代码

7.3.3 单块 base64 解码

原型：
```
int SAF_Base64_Decode(
    unsigned char * pucInData,
    unsigned int uiInDataLen,
    unsigned char * pucOutData,
    unsigned int * puiOutDataLen);
```
描述：对输入的数据进行 base64 解码。

参数：

pucInData[in]	解码前的数据
uiInDataLen[in]	解码前的数据长度
pucOutData[out]	解码后的数据
puiOutDataLen[in,out]	解码后的数据长度

返回值：

0	成功
非 0	失败，返回错误代码

7.3.4 创建 base64 对象

原型：
```
int SAF_Base64_CreateBase64Obj(
    void ** phBase64Obj);
```
描述：为任意长度数据的 base64 编解码创建 base64 对象。

参数：

phBase64Obj[out]	指向创建的 base64 对象句柄的指针

返回值：

0	成功
非 0	失败，返回错误代码

备注：本函数与 SGD_Base64_DestroyBase64Obj，SGD_Base64_EncodeUpdate，SGD_Base64_EncodeFinal，SGD_Base64_DecodeUpdate，SGD_Base64_DecodeFinal 等函数共同使用以支持任意长度数据的 base64 编解码。

7.3.5 销毁 base64 对象

原型：
```
int SAF_Base64_DestroyBase64Obj(
    void * hBase64Obj);
```
描述：删除 base64 对象。

参数：

hBase64Obj[in]	需要删除的 base64 对象句柄

返回值：

0	成功
非 0	失败，返回错误代码

7.3.6 通过 base64 对象多块编码

原型：
```
int SAF_Base64_EncodeUpdate(
    void * hBase64Obj,
    unsigned char * pucInData,
    unsigned int uiInDataLen,
    unsigned char * pucOutData,
    unsigned int * puiOutDataLen);
```
描述：通过 base64 对象对数据多块 base64 编码。

参数：

hBase64Obj[in]	base64 对象
pucInData[in]	编码前的数据
uiInDataLen[in]	编码前的数据长度
pucOutData[out]	编码后的数据
puiOutDataLen[in,out]	返回编码后的数据长度

返回值：

0	成功
非 0	失败，返回错误代码

备注：编码完成后，需调用 SAF_Base64_EncodeFinal

7.3.7 通过 base64 对象编码结束

原型：

```
int SAF_Base64_EncodeFinal(
    void * hBase64Obj,
    unsigned char * pucOutData,
    unsigned int * puiOutDataLen);
```

描述：通过 base64 对象对数据编码结束。

参数：

hBase64Obj[in]	base64 对象
pucOutData[out]	编码后的数据
puiOutDataLen[in,out]	返回编码后的数据长度

返回值：

0	成功
非 0	失败，返回错误代码

7.3.8 通过 base64 对象多块解码

原型：

```
int SAF_Base64_DecodeUpdate(
    void * hBase64Obj,
    unsigned char * pucInData,
    unsigned int uiInDataLen,
    unsigned char * pucOutData,
    unsigned int * puiOutDataLen);
```

描述：通过 base64 对象对数据多块 base64 解码。

参数：

hBase64Obj[in]	base64 对象
pucInData[in]	解码前的数据
uiInDataLen[in]	解码前的数据长度
pucOutData[out]	解码后的数据
puiOutDataLen[in,out]	返回解码后的数据长度

返回值：

0	成功
非 0	失败，返回错误代码

备注：解码完成后，需调用 SAF_Base64_DecodeFinal 结束。

7.3.9 通过 base64 对象解码结束

原型：

```
int SAF_Base64_DecodeFinal(
    void * hBase64Obj,
    unsigned char * pucOutData,
    unsigned int * puiOutDataLen);
```

描述：　通过 base64 对象对数据解码结束。

参数：		
	hBase64Obj[in]	base64 对象
	pucOutData[out]	解码后的数据
	puiOutDataLen[in,out]	返回解码后的数据长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.3.10　生成随机数

```
原型：　int SAF_GenRandom(
            unsigned int uiRandLen,
            unsigned char *pucRand);
```

描述：　生成指定长度的随机数。

参数：		
	uiRandLen[in]	随机数长度
	pucRand[out]	指定长度的随机数，内存由外部调用者分配
返回值：	0	成功
	非 0	失败，返回错误代码

7.3.11　HASH 运算

```
原型：　int SAF_Hash(
            unsigned int uiAlgoType,
            unsigned char *pucInData,
            unsigned int uiInDataLen,
            unsigned char *pucPublicKey,
            unsigned int ulPublicKeyLen,
            unsigned char *pucID,
            unsigned int ulIDLen,
            unsigned char *pucOutData,
            unsigned int *puiOutDataLen);
```

描述：　HASH 运算，对给定长度数据的 HASH 运算。

参数：		
	uiAlgoType[in]	HASH 算法
	pucInData[in]	输入数据
	uiInDataLen[in]	输入数据长度
	pucPublicKey[in]	签名者公钥，当 alAlgID 为 SGD_SM3 时有效
	ulPublicKeyLen[in]	签名者公钥长度
	pucID[in]	签名者的 ID 值，当 alAlgID 为 SGD_SM3 时有效
	ulIDLen[in]	签名者 ID 的长度，当 alAlgID 为 SGD_SM3 时有效
	pucOutData[out]	HASH
	puiOutDataLen[out]	HASH 长度
返回值：	0	成功
	非 0	失败，返回错误代码

备注：　当 ulAlgID 为 SGD_SM3 且 ulIDLen 不为 0 的情况下 pPubKey、pucID 有效，执行 SM2 算法签名预处理 1 操作。计算过程遵循 GM/T 0009。

7.3.12 创建 HASH 对象

原型：
```
int SAF_CreateHashObj(
    void **phHashObj,
    unsigned int uiAlgorithmType,
    unsigned char *pucPublicKey,
    unsigned int ulPublicKeyLen,
    unsigned char *pucID,
    unsigned int ulIDLen);
```

描述：创建 HASH 对象，对任意长度数据的 HASH 运算。

参数：

phHashObj[out]	创建的 HASH 对象
uiAlgorithmType[in]	HASH 算法
pucPublicKey[in]	签名者公钥，当 alAlgID 为 SGD_SM3 时有效
ulPublicKeyLen[in]	签名者公钥长度
pucID[in]	签名者的 ID 值，当 alAlgID 为 SGD_SM3 时有效
ulIDLen[in]	签名者 ID 的长度，当 alAlgID 为 SGD_SM3 时有效

返回值：

0	成功
非 0	失败，返回错误代码

备注：当 ulAlgID 为 SGD_SM3 且 ulIDLen 不为 0 的情况下 pPubKey、pucID 有效，执行 SM2 算法签名预处理 1 操作。计算过程遵循 GM/T 0009。

7.3.13 删除 HASH 对象

原型：
```
int SAF_DestroyHashObj(
    void *hHashObj);
```

描述：删除 HASH 对象。

参数：

hHashObj[in]	需要删除的 HASH 对象

返回值：

0	成功
非 0	失败，返回错误代码

7.3.14 通过对象进行多块 HASH 运算

原型：
```
int SAF_HashUpdate(
    void *hHashObj,
    unsigned char *pucInData,
    unsigned int uiInDataLen);
```

描述：继续 HASH 运算。

参数：

hHashObj[in]	HASH 对象
pucInData[in]	输入数据
uiInDataLen[in]	输入数据长度

返回值：

0	成功
非 0	失败，返回错误代码

备注：运算完成后，需调用 SAF_HashFinal 结束。

7.3.15 结束 HASH 运算

原型：
```
int SAF_HashFinal(
    void * hHashObj,
    unsigned char * pucOutData,
    unsigned int * uiOutDataLen);
```

描述：结束 HASH 运算。

参数：

hHashObj[in]	HASH 对象
pucOutData[out]	输出的 HASH 值
uiOutDataLen[out]	HASH 值的长度

返回值：

0	成功
非 0	失败，返回错误代码

7.3.16 生成 RSA 密钥对

原型：
```
int SAF_GenRsaKeyPair (
    void * hAppHandle,
    unsigned char * pucContainerName,
    unsigned int uiContainerNameLen,
    unsigned int uiKeyBits,
    unsigned int uiKeyUsage,
    unsigned int uiExportFlag);
```

描述：产生指定名称的容器并在该容器内生成 RSA 密钥对。

参数：

hAppHandle[in]	应用接口句柄
pucContainerName[in]	密钥的容器名
uiContainerNameLen[in]	密钥的容器名长度
uiKeyBits[in]	密钥模长
uiKeyUsage[in]	密钥用途
SGD_KEYUSAGE_SIGN	签名
SGD_KEYUSAGE_KEYEXCHANGE	密钥交换(加密)
uiExportFlag[in]	1 表示生成的密钥对可导出
	0 表示不可导出

返回值：

0	成功
非 0	失败，返回错误代码

备注：如果指定的容器名已存在，则在该容器内增加或者替换密钥对。

7.3.17 获取 RSA 公钥

原型：
```
int SAF_GetRsaPublicKey(
    void * hAppHandle,
    unsigned char * pucContainerName,
    unsigned int uiContainerLen,
    unsigned int uiKeyUsage,
    unsigned char * pucPublicKey,
    unsigned int * puiPublicKeyLen);
```

描述：　取出符合 PKCS1 的 RSA 公钥。

参数：		
	hAppHandle[in]	应用接口句柄
	pucContainerName[in]	密钥的容器名或密码机的密码号
	uiContainerLen[in]	密钥的容器名长度
	uiKeyUsage[in]	密钥用途
	SGD_KEYUSAGE_SIGN	签名
	SGD_KEYUSAGE_KEYEXCHANGE	加密
	pucPublicKey[out]	输出的 DER 格式的公钥数据
	puiPublicKeyLen[in,out]	输出公钥数据的长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.3.18 RSA 签名运算

```
原型：　int SAF_RsaSign(
            void * hAppHandle,
            unsigned char * pucContainerName,
            unsigned int uiContainerNameLen,
            unsigned int uiHashAlgorithmID,
            unsigned char * pucInData,
            unsigned int uiOnDataLen,
            unsigned char * pucSignData,
            unsigned int * puiSignDataLen);
```

描述：　按照 PKCS#1 的要求对一定长度的字符串进行签名运算。

参数：		
	hAppHandle[in]	应用接口句柄
	pucContainerName[in]	密钥的容器名或密钥号
	uiContainerNameLen[in]	密钥的容器名长度
	uiHashAlgorithmID[in]	HASH 算法
	pucInData[in]	原始数据
	uiOnDataLen[in]	原始数据的长度
	pucSignData[out]	输出的 DER 格式的签名结果数据
	puiSignDataLen[in,out]	输出的签名结果数据长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.3.19 对文件进行 RSA 签名运算

```
原型：　int SAF_RsaSignFile(
            void * hAppHandle,
            unsigned char * pucContainerName,
            unsigned int uiContainerNameLen,
            unsigned int uiHashAlgorithmID,
            unsigned char * pucFileName,
            unsigned char * pucSignData,
            unsigned int * puiSignDataLen);
```

描述：　按照 PKCS#1 的要求对指定的文件进行签名运算。

参数：

hAppHandle[in]	应用接口句柄
pucContainerName[in]	密钥的容器名
uiContainerNameLen[in]	密钥的容器名长度
uiHashAlgorithmID[in]	HASH 算法
pucFileName[in]	要签名的全路径文件名
pucSignData[out]	输出的 DER 格式的签名结果数据
puiSignDataLen[in,out]	输出的签名结果数据长度

返回值：

0	成功
非 0	失败，返回错误代码

备注：　此函数主要用于对附件的签名，也适用于对大文件的签名。

7.3.20　RSA 验证签名运算

原型：

```
int SAF_RsaVerifySign(
    unsigned int uiHashAlgorithmID,
    unsigned char * pucPublicKey,
    unsigned int uiPublicKeyLen,
    unsigned char * pucInData,
    unsigned int uiInDataLen,
    unsigned char * pucSignData,
    unsigned int uiSignDataLen);
```

描述：　符合 PKCS1 的验证签名运算。

参数：

uiHashAlgorithmID[in]	HASH 算法
pucPublicKey[in]	DER 编码的公钥
uiPublicKeyLen[in]	DER 编码的公钥长度
pucInData[in]	原始数据
uiInDataLen[in]	原始数据的长度
pucSignData[in]	DER 编码的签名数据
uiSignDataLen[in]	签名数据长度

返回值：

0	成功
非 0	失败，返回错误代码

7.3.21　对文件及其签名进行 RSA 验证

原型：

```
int SAF_RsaVerifySignFile(
    unsigned int uiHashAlgorithmID,
    unsigned char * pucPublicKey,
    unsigned int uiPublicKeyLen,
    unsigned char * pucFileName,
    unsigned char * pucSignData,
    unsigned int uiSignDataLen);
```

描述：　对文件及其签名值，进行符合 PKCS#1 的验证签名运算。

参数：

uiHashAlgorithmID[in]	HASH 算法
pucPublicKey[in]	DER 编码的公钥

	uiPublicKeyLen[in]	DER 编码的公钥长度
	pucFileName[in]	需要验证签名的文件名
	pucSignData[in]	DER 编码的签名数据
	uiSignDataLen[in]	签名数据长度
返回值：	0	成功
	非 0	失败，返回错误代码

备注： 本函数用于对附件的签名进行验证，也可对大文件的签名进行验证。

7.3.22 基于证书的 RSA 公钥验证

原型：
```
int SAF_VerifySignByCert (
    unsigned int uiHashAlgorithmID,
    unsigned char * pucCertificate,
    unsigned int uiCertificateLen,
    unsigned char * pucInData,
    unsigned int uiInDataLen,
    unsigned char * pucSignData,
    unsigned int uiSignDataLen);
```

描述： 按照 PKCS#1 的要求使用数字证书对签名值进行验证。

参数：	uiHashAlgorithmID[in]	HASH 算法标识
	pucCertificate[in]	DER 编码的数字证书
	uiCertificateLen[in]	DER 编码的数字证书长度
	pucInData[in]	原始数据
	uiInDataLen[in]	原始数据的长度
	pucSignData[in]	签名数据
	uiSignDataLen[in]	签名数据长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.3.23 生成 ECC 密钥对

原型：
```
int SAF_GenEccKeyPair(
    void * hAppHandle,
    unsigned char * pucContainerName,
    unsigned int uiContainerLen,
    unsigned int uiKeyBits,
    unsigned int uiKeyUsage,
    unsigned int uiExportFlag);
```

描述： 生成指定名称的容器，并在该容器内生成 ECC 密钥对。

参数：	hAppHandle[in]	应用接口句柄
	pucContainerName[in]	密钥的容器名
	uiContainerLen[in]	密钥的容器名长度
	uiKeyBits[in]	密钥模长
	uiKeyUsage[in]	密钥用途
	SGD_SM2_1	椭圆曲线签名

	SGD_SM2_2	椭圆曲线密钥交换协议
	SGD_SM2_3	椭圆曲线加密
	uiExportFlag[in]	1 表示生成的密钥对可导出 0 表示不可导出
返回值：	0	成功
	非 0	失败，返回错误代码

7.3.24 获取 ECC 公钥

原型：
```
int SAF_GetEccPublicKey(
    void * hAppHandle,
    unsigned char * pucContainerName,
    unsigned int uiContainerLen,
    unsigned int uiKeyUsage,
    unsigned char * pucPublicKey,
    unsigned int * puiPublicKeyLen);
```

描述：取出 Ecc 公钥。

参数：

hAppHandle[in]	应用接口句柄
pucContainerName[in]	密钥的容器名
uiContainerLen[in]	密钥的容器名长度
uiKeyUsage[in]	密钥用途
SGD_SM2_1	签名公钥
SGD_SM2_2	密钥交换协议公钥
SGD_SM2_3	加密公钥
pucPublicKey[out]	输出的 DER 编码的公钥数据
puiPublicKeyLen[in,out]	输出公钥数据的长度

返回值：

0	成功
非 0	失败，返回错误代码

7.3.25 ECC 签名

原型：
```
int SAF_EccSign (
    void * hAppHandle,
    unsigned char * pucContainerName,
    unsigned int uiContainerNameLen,
    unsigned int uiAlgorithmID,
    unsigned char * pucInData,
    unsigned int uiInDataLen,
    unsigned char * pucSignData,
    unsigned int * puiSignDataLen);
```

描述：使用 ECC 私钥对数据进行签名运算。

参数：

hAppHandle[in]	应用接口句柄
pucContainerName[in]	密钥的容器名
uiContainerNameLen[in]	密钥的容器名长度
uiAlgorithmID[in]	签名算法标识
pucInData[in]	待签名数据

	uiInDataLen[in]	待签名数据的长度
	pucSignData[out]	输出的 DER 编码的签名数据
	puiSignDataLen[in,out]	输出的签名结果数据长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.3.26 ECC 验证

原型：
```
int SAF_ EccVerifySign(
    unsigned char * pucPublicKey,
    unsigned int uiPublicKeyLen,
    unsigned int uiAlgorithmID,
    unsigned char * pucInData,
    unsigned int uiInDataLen,
    unsigned char * pucSignData,
    unsigned int uiSignDataLen);
```

描述：利用 ECC 公钥验证签名。

参数：	pucPublicKey[in]	DER 编码的公钥
	uiPublicKeyLen[in]	公钥长度
	uiAlgorithmID[in]	ECC 签名算法
	pucInData[in]	待验证数据
	uiInDataLen[in]	待验证数据的长度
	pucSignData[in]	DER 编码的签名数据
	uiSignDataLen[in]	签名数据长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.3.27 ECC 公钥加密

原型：
```
int SAF_EccPublicKeyEnc(
    unsigned char * pucPublicKey,
    unsigned int uiPublicKeyLen,
    unsigned int uiAlgorithmID,
    unsigned char * pucInData,
    unsigned int uiInDataLen,
    unsigned char * pucOutData,
    unsigned int * puiOutDataLen);
```

描述：ECC 公钥加密运算。

参数：	pucPublicKey[in]	DER 编码的公钥
	uiPublicKeyLen[in]	DER 编码的公钥长度
	uiAlgorithmID[in]	ECC 算法标识
	pucInData[in]	输入数据，该数据不能为密钥
	uiInDataLen[in]	输入数据长度
	pucOutData[out]	输出数据，DER 编码
	puiOutDataLen[in,out]	输出数据长度

返回值：0　　成功
非 0　　失败，返回错误代码

7.3.28 基于证书的 ECC 公钥加密

原型：
```
int SAF_EccPublicKeyEncByCert(
    unsigned char * pucCertificate,
    unsigned int uiCertificateLen,
    unsigned int uiAlgorithmID,
    unsigned char * pucInData,
    unsigned int uiInDataLen,
    unsigned char * pucOutData,
    unsigned int * puiOutDataLen);
```
描述：基于证书的 ECC 公钥加密。

参数：

pucCertificate[in]	DER 编码的数字证书
uiCertificateLen[in]	数字证书长度
uiAlgorithmID[in]	ECC 算法标识
pucInData[in]	输入数据
uiInDataLen[in]	输入数据长度
pucOutData[out]	输出 DER 编码的数据
puiOutDataLen[in,out]	输出数据长度

返回值：0　　成功
非 0　　失败，返回错误代码

7.3.29 基于证书的 ECC 公钥验证

原型：
```
int SAF_EccVerifySignByCert(
    unsigned int uiAlgorithmID,
    unsigned char * pucCertificate,
    unsigned int uiCertificateLen,
    unsigned char * pucInData,
    unsigned int uiInDataLen,
    unsigned char * pucSignData,
    unsigned int uiSignDataLen);
```
描述：基于证书的 ECC 公钥验证。

参数：

uiAlgorithmID[in]	ECC 签名算法标识
pucCertificate[in]	DER 编码的数字证书
uiCertificateLen[in]	数字证书长度
pucInData[in]	待验证数据
uiInDataLen[in]	待验证数据的长度
pucSignData[in]	DER 编码的签名数据
uiSignDataLen[in]	签名数据长度

返回值：0　　成功
非 0　　失败，返回错误代码

7.3.30 创建对称算法对象

原型：
```
int SAF_CreateSymmKeyObj(
    void           * hAppHandle,
    void           ** phSymmKeyObj,
    unsigned char  * pucContainerName,
    unsigned int     uiContainerLen,
    unsigned char  * pucIV,
    unsigned int     uiIVLen,
    unsigned int     uiEncOrDec,
    unsigned int     uiCryptoAlgID);
```

描述：本地产生对称算法对象。

参数：

hAppHandle[in]	应用接口句柄
phSymmKeyObj[out]	返回的对称算法对象
pucContainerName[in]	密钥的容器名
uiContainerLen[in]	密钥的容器名长度
pucIV[in]	初始向量，ECB模式时此参数忽略
uiIVLen[in]	初始向量长度，ECB模式时此参数忽略
uiEncOrDec[in]	1 encrypt 0 decrypt
uiCryptoAlgID[in]	加密算法标识

返回值：

0	成功
非0	失败，返回错误代码

7.3.31 生成会话密钥并用外部公钥加密输出

原型：
```
int SAF_GenerateKeyWithEPK(
    void  * hSymmKeyObj,
    unsigned char  * pucPublicKey,
    unsigned int uiPublicKeyLen,
    unsigned char  * pucSymmKey,
    unsigned int uiSymmKeyLen,
    void  ** phKeyHandle);
```

描述：生成会话密钥并用外部公钥加密输出。

参数：

hSymmKeyObj[in]	对称算法对象
pucPublicKey[in]	输入的 DER 编码的公钥数据
uiPublicKeyLen[in]	输入公钥数据的长度
pucSymmKey[out]	输出的加密后的会话密钥
uiSymmKeyLen[out]	输出的加密会话密钥长度
phKeyHandle[out]	输出会话密钥的句柄

返回值：

0	成功
非0	失败，返回错误代码

7.3.32 导入加密的会话密钥

原型：
```
int SAF_ImportEncedKey(
    void          * hSymmKeyObj,
    unsigned char * pucSymmKey,
    unsigned int uiSymmKeyLen,
    void          ** phKeyHandle);
```

描述：导入加密的会话密钥，使用指定的私钥解密，产生会话密钥句柄并输出。

参数：

hSymmKeyObj[in]	对称算法对象
pucSymmKey[in]	输入的加密后的会话密钥
uiSymmKeyLen[in]	输入的加密会话密钥长度
phKeyHandle[out]	输出会话密钥的句柄

返回值：

0	成功
非 0	失败，返回错误代码

7.3.33 生成密钥协商参数并输出

原型：
```
int SAF_GenerateAgreementDataWithECC (
    void * hSymmKeyObj,
    unsigned int uiISKIndex,
    unsigned int uiKeyBits,
    unsigned char * pucSponsorID,
    unsigned int uiSponsorIDLength,
    unsigned char * pucSponsorPublicKey,
    unsigned int * puiSponsorPublicKeyLen,
    unsigned char * pucSponsorTmpPublicKey,
    unsigned int * puiSponsorTmpPublicKeyLen,
    void ** phAgreementHandle);
```

描述：使用 ECC 密钥协商算法，为计算会话密钥而产生协商参数，同时返回指定索引位置的 ECC 公钥、临时 ECC 密钥对的公钥及协商句柄。

参数：

hSymmKeyObj [in]	对称算法对象
uiISKIndex[in]	密码设备内部存储加密私钥的索引值，该私钥用于参与密钥协商
uiKeyBits[in]	要求协商的密钥长度
pucSponsorID[in]	参与密钥协商的发起方 ID 值
uiSponsorIDLength[in]	发起方 ID 长度
pucSponsorPublicKey[out]	返回的发起方 ECC 公钥
puiSponsorPublicKeyLen[in,out]	返回的发起方 ECC 公钥长度
pucSponsorTmpPublicKey[out]	返回的发起方临时 ECC 公钥
puiSponsorTmpPublicKeyLen [in,out]	返回的发起方临时 ECC 公钥长度
phAgreementHandle[out]	返回的密钥协商句柄

返回值：

0	成功
非 0	失败，返回错误代码

备注：　为协商会话密钥，协商的发起方应首先调用本函数计算会话密钥。会话密钥的计算过程遵循 GM/T 0009。

7.3.34　计算会话密钥

原型：
```
int SAF_GenerateKeyWithECC(
    void * hAgreementHandle,
    unsigned char * pucResponseID,
    unsigned int uiResponseIDLength,
    unsigned char * pucResponsePublicKey,
    unsigned int uiResponsePublicKeyLen,
    unsigned char * pucResponseTmpPublicKey,
    unsigned int uiResponseTmpPublicKeyLen,
    void ** phKeyHandle);
```

描述：　使用 ECC 密钥协商算法，使用自身协商句柄和响应方的协商参数计算会话密钥，同时返回会话密钥句柄。

参数：

hAgreementHandle [in]	密钥协商句柄
pucResponseID[in]	外部输入的响应方 ID 值
uiResponseIDLength[in]	外部输入的响应方 ID 长度
pucResponsePublicKey[in]	外部输入的响应方 ECC 公钥
uiResponsePublicKeyLen[in]	外部输入的响应方 ECC 公钥长度
pucResponseTmpPublicKey[in]	外部输入的响应方临时 ECC 公钥
uiResponseTmpPublicKeyLen[in]	外部输入的响应方临时 ECC 公钥长度
phKeyHandle[out]	返回的密钥句柄

返回值：

0	成功
非 0	失败，返回错误代码

备注：　协商的发起方获得响应方的协商参数后调用本函数，计算会话密钥。会话密钥的计算过程遵循 GM/T 0009。

7.3.35　产生协商数据并计算会话密钥

原型：
```
int SAF_GenerateAgreementDataAndKeyWithECC (
    void * hSymmKeyObj,
    unsigned int uiISKIndex,
    unsigned int uiKeyBits,
    unsigned char * pucResponseID,
    unsigned int uiResponseIDLength,
    unsigned char * pucSponsorID,
    unsigned int uiSponsorIDLength,
    unsigned char * pucSponsorPublicKey,
    unsigned int uiSponsorPublicKeyLen,
    unsigned char * pucSponsorTmpPublicKey,
    unsigned int uiSponsorTmpPublicKeyLen,
    unsigned char * pucResponsePublicKey,
    unsigned int * puiResponsePublicKeyLen,
```

```
    unsigned char * pucResponseTmpPublicKey
    unsigned int * puiResponseTmpPublicKeyLen,
    void ** phKeyHandle);
```

描述：使用 ECC 密钥协商算法，产生协商参数并计算会话密钥，同时返回产生的协商参数和密钥句柄。

参数：

hSymmKeyObj[in]	对称算法对象
uiISKIndex[in]	密码设备内部存储加密私钥的索引值，该私钥用于参与密钥协商
uiKeyBits[in]	协商后要求输出的密钥长度
pucResponseID[in]	响应方 ID 值
uiResponseIDLength[in]	响应方 ID 长度
pucSponsorID[in]	发起方 ID 值
uiSponsorIDLength[in]	发起方 ID 长度
pucSponsorPublicKey[in]	外部输入的发起方 ECC 公钥
pucSponsorPublicKeyLen[in]	外部输入的发起方 ECC 公钥长度
uiSponsorTmpPublicKey[in]	外部输入的发起方临时 ECC 公钥
uiSponsorTmpPublicKeyLen[in]	外部输入的发起方临时 ECC 公钥长度
pucResponsePublicKey[out]	返回的响应方 ECC 公钥
puiResponsePublicKeyLen[in,out]	返回的响应方 ECC 公钥长度
pucResponseTmpPublicKey[out]	返回的响应方临时 ECC 公钥
puiResponseTmpPublicKeyLen[in,out]	返回的响应方临时 ECC 公钥长度
phKeyHandle[out]	返回的对称算法密钥句柄

返回值：

0	成功
非 0	失败，返回错误代码

备注：本函数由响应方调用。会话密钥的计算过程遵循 GM/T 0009。

7.3.36 销毁对称算法对象

```
原型：int SAF_DestroySymmKeyObj(
        void * hSymmKeyObj);
```

描述：销毁对称算法对象。

参数：

hSymmKeyObj[in]	对称算法密钥句柄

返回值：

0	成功
非 0	失败，返回错误代码

7.3.37 销毁会话密钥句柄

```
原型：int SAF_DestroyKeyHandle(
        void * hKeyHandle);
```

描述：销毁会话密钥句柄。

参数：

hKeyHandle [in]	会话密钥句柄

返回值：

0	成功
非 0	失败，返回错误代码

7.3.38 单块加密运算

原型：
```
int SAF_SymmEncrypt(
    void * hKeyHandle,
    unsigned char * pucInData,
    unsigned int uiInDataLen,
    unsigned char * pucOutData,
    unsigned int * puiOutDataLen);
```

描述：加密运算。

参数：

hKeyHandle[in]	对称算法密钥句柄
pucInData[in]	输入数据
uiInDataLen[in]	输入数据长度
pucOutData[out]	输出数据
puiOutDataLen[in,out]	输出数据长度

返回值：

0	成功
非 0	失败，返回错误代码

7.3.39 多块加密运算

原型：
```
int SAF_SymmEncryptUpdate(
    void * hKeyHandle,
    unsigned char * pucInData,
    unsigned int uiInDataLen,
    unsigned char * pucOutData,
    unsigned int * puiOutDataLen);
```

描述：多块加密运算。

参数：

hKeyHandle[in]	对称算法密钥句柄
pucInData[in]	输入数据
uiInDataLen[in]	输入数据长度
pucOutData[out]	输出数据
puiOutDataLen[in,out]	输出数据长度

返回值：

0	成功
非 0	失败，返回错误代码

备注：运算完成后，需调用 SAF_SymmEncryptFinal 结束。

7.3.40 结束加密运算

原型：
```
int SAF_SymmEncryptFinal(
    void * hKeyHandle,
    unsigned char * pucOutData,
    unsigned int * puiOutDataLen);
```

描述：结束加密运算。

参数：

phSymmAlgoObj [in]	对称算法密钥句柄
pucOutData[out]	输出数据
puiOutDataLen[in,out]	输出数据长度

返回值：0　　成功
非 0　　失败，返回错误代码

7.3.41　单块解密运算

原型：
```
int SAF_SymmDecrypt(
    void * hKeyHandle,
    unsigned char * pucInData,
    unsigned int uiInDataLen,
    unsigned char * pucOutData,
    unsigned int * puiOutDataLen);
```
描述：解密运算。
参数：

hKeyHandle[in]	对称算法密钥句柄
pucInData[in]	输入数据
uiInDataLen[in]	输入数据长度
pucOutData[out]	输出数据
puiOutDataLen[in,out]	输出数据长度

返回值：0　　成功
非 0　　失败，返回错误代码

7.3.42　多块解密运算

原型：
```
int SAF_SymmDecryptUpdate(
    void * hKeyHandle,
    unsigned char * pucInData,
    unsigned int uiInDataLen,
    unsigned char * pucOutData,
    unsigned int * puiOutDataLen);
```
描述：继续解密运算。
参数：

hKeyHandle[in]	对称算法密钥句柄
pucInData[in]	输入数据
uiInDataLen[in]	输入数据长度
pucOutData[out]	输出数据
puiOutDataLen[in,out]	输出数据长度

返回值：0　　成功
非 0　　失败，返回错误代码
备注：运算完成后，需调用 SAF_SymmDecryptFinal 结束。

7.3.43　结束解密运算

原型：
```
int SAF_SymmDecryptFinal(
    void * hKeyHandle,
    unsigned char * pucOutData,
    unsigned int * puiOutDataLen);
```
描述：结束解密运算。
参数：

hKeyHandle[in]	对称算法密钥句柄

	pucOutData[out]	输出数据
	puiOutDataLen[in,out]	输出数据长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.3.44 单组数据消息鉴别码运算

```
原型：  int SAF_Mac(
            void * hKeyHandle,
            unsigned char * pucOnData,
            unsigned int uiInDataLen,
            unsigned char * pucOutData,
            unsigned int uiOutDataLen);
```

描述： 单组数据消息鉴别码运算。

参数：	hKeyHandle[in]	对称算法密钥句柄
	pucOnData[in]	输入数据
	uiInDataLen[in]	输入数据长度
	pucOutData[out]	输出的 MAC
	uiOutDataLen[out]	MAC 长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.3.45 多组数据消息鉴别码运算

```
原型：  int SAF_MacUpdate(
            void * hKeyHandle,
            unsigned char * pucInData,
            unsigned int uiInDataLen);
```

描述： 连续多组数据消息鉴别码运算。

参数：	phSymmAlgoObj[in]	对称算法密钥句柄
	pucInData[in]	输入数据
	uiInDataLen[in]	输入数据长度
返回值：	0	成功
	非 0	失败，返回错误代码

备注： 运算完成后，需调用 SAF_SymmDecryptFinal 结束。

7.3.46 结束消息鉴别码运算

```
原型：  int SAF_MacFinal(
            void * hKeyHandle,
            unsigned char * pucOutData,
            unsigned int uiOutDataLen);
```

描述： 结束消息鉴别码运算。

参数：	hKeyHandle[in]	对称算法密钥句柄
	pucOutData[out]	输出的 MAC
	uiOutDataLen[out]	MAC 长度

返回值： 0 成功

非 0 失败，返回错误代码

7.4 消息类函数

7.4.1 概述

消息类函数包含以下具体函数，各函数返回值见附录 A 错误代码定义：

a) 编码 PKCS#7 格式的带签名的数字信封数据：SAF_Pkcs7_EncodeData

b) 解码 PKCS#7 格式的带签名的数字信封数据：SAF_Pkcs7_DecodeData

c) 编码 PKCS#7 格式的签名数据：SAF_Pkcs7_EncodeSignedData

d) 解码 PKCS#7 格式的签名数据：SAF_Pkcs7_DecodeSignedData

e) 编码 PKCS#7 格式的数字信封数据：SAF_Pkcs7_EncodeEnvelopedData

f) 解码 PKCS#7 格式的数字信封数据：SAF_Pkcs7_DecodeEnvelopedData

g) 编码 PKCS#7 格式的摘要数据：SAF_Pkcs7_EncodeDigestedData

h) 解码 PKCS#7 格式的摘要数据：SAF_Pkcs7_DecodeDigestedData

i) 编码基于 SM2 算法的带签名的数字信封数据：SAF_SM2_EncodeSignedAndEnvelopedData

j) 解码基于 SM2 算法的带签名的数字信封数据：SAF_SM2_DecodeSignedAndEnvelopedData

k) 编码基于 SM2 算法的签名数据：SAF_SM2_EncodeSignedData

l) 解码基于 SM2 算法的签名数据：SAF_SM2_DecodeSignedData

m) 编码基于 SM2 算法的数字信封：SAF_SM2_EncodeEnvelopedData

n) 解码基于 SM2 算法的数字信封：SAF_SM2_DecodeEnvelopedData

7.4.2 编码 PKCS#7 格式的带签名的数字信封数据

原型：

```
int SAF_Pkcs7_EncodeData(
    void * hAppHandle,
    unsigned char * pucSignContainerName,
    unsigned int uiSignContainerNameLen,
    unsigned int uiSignKeyUsage,
    unsigned char * pucSignerCertificate,
    unsigned int uiSignerCertificateLen,
    unsigned int uiDigestAlgorithms,
    unsigned char * pucEncCertificate,
    unsigned int uiEncCertificateLen,
    unsigned int uiSymmAlgorithm,
    unsigned char * pucData,
    unsigned int uiDataLen,
    unsigned char * pucDerP7Data,
    unsigned int * puiDerP7DataLen);
```

描述： 编码 PKCS#7 格式的带签名的数字信封数据。

参数：

hAppHandle[in]	应用接口句柄
pucSignContainerName[in]	签名私钥的容器名
uiSignContainerNameLen[in]	签名私钥的容器名长度
uiSignKeyUsage[in]	私钥的用途

	pucSignerCertificate[in]	签名者证书
	uiSignerCertificateLen[in]	签名者证书长度
	uiDigestAlgorithms[in]	HASH 算法
	pucEncCertificate[in]	接收者证书
	uiEncCertificateLen[in]	接收者证书长度
	uiSymmAlgorithm[in]	对称算法参数
	pucData[in]	原始数据
	uiDataLen[in]	原始数据长度
	pucDerP7Data[out]	编码后的数据
	puiDerP7DataLen[in,out]	编码后的数据长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.4.3 解码 PKCS#7 格式的带签名的数字信封数据

原型：

```
int SAF_Pkcs7_DecodeData(
    void * hAppHandle,
    unsigned char * pucDecContainerName,
    unsigned int uiDecContainerNameLen,
    unsigned int uiDecKeyUsage,
    unsigned char * pucDerP7Data,
    unsigned int uiDerP7DataLen,
    unsigned char * pucData,
    unsigned int * puiDataLen,
    unsigned char * pucSignerCertificate,
    unsigned int * puiSignerCertificateLen,
    unsigned int * puiDigestAlgorithms,
    unsigned char * pucSignedData,
    unsigned int * puiSignedDataLen);
```

描述：解码 PKCS#7 格式的带签名的数字信封数据。

参数：	hAppHandle[in]	应用接口句柄
	pucDecContainerName[in]	解密用私钥的容器名
	uiDecContainerNameLen[in]	解密用私钥的容器名长度
	uiDecKeyUsage[in]	解密用私钥的用途
	pucDerP7Data[in]	编码后的数据
	uiDerP7DataLen[in]	编码后的数据长度
	pucData[out]	原始数据
	puiDataLen[in,out]	原始数据长度
	pucSignerCertificate[out]	签名者证书
	puiSignerCertificateLen[in,out]	签名者证书长度
	puiDigestAlgorithms[out]	HASH 算法
	pucSignedData[out]	Base64 编码的签名值
	puiSignedDataLen[in,out]	签名值长度
返回值：	0	成功

非 0	失败，返回错误代码

7.4.4 编码 PKCS#7 格式的签名数据

原型：

```
int SAF_Pkcs7_EncodeSignedData(
    void *hAppHandle,
    unsigned char *pucSignContainerName,
    unsigned int uiSignContainerNameLen,
    unsigned int uiSignKeyUsage,
    unsigned char *pucSignerCertificate,
    unsigned int uiSignerCertificateLen,
    unsigned int uiDigestAlgorithms,
    unsigned char *pucData,
    unsigned int uiDataLen,
    unsigned char *pucDerP7SignedData,
    unsigned int *dpuiDerP7SignedDataLen);
```

描述：编码 PKCS#7 格式的签名数据。

参数：

hAppHandle[in]	应用接口句柄
pucSignContainerName[in]	签名私钥的容器名
uiSignContainerNameLen[in]	签名私钥的容器名长度
uiSignKeyUsage[in]	私钥的用途
pucSignerCertificate[in]	签名者证书
uiSignerCertificateLen[in]	签名者证书长度
uiDigestAlgorithms[in]	HASH 算法
pucData[in]	需要签名的数据
uiDataLen[in]	需要签名的数据长度
pucDerP7SignedData[out]	带签名值的 P7 数据
dpuiDerP7SignedDataLen[in,out]	带签名值的 P7 数据长度

返回值：

0	成功
非 0	失败，返回错误代码

7.4.5 解码 PKCS#7 格式的签名数据

原型：

```
int SAF_Pkcs7_DecodeSignedData(
    void *hAppHandle,
    unsigned char *pucDerP7SignedData,
    unsigned int uiDerP7SignedDataLen,
    unsigned char *pucSignerCertificate,
    unsigned int *puiSignerCertificateLen,
    unsigned int *puiDigestAlgorithms,
    unsigned char *pucData,
    unsigned int *puiDataLen,
    unsigned char *pucSign,
    unsigned int *puiSignLen);
```

描述：解码 PKCS#7 格式的签名数据。

参数：	hAppHandle[in]	应用接口句柄
	pucDerP7SignedData[in]	签名后的数据
	uiDerP7SignedDataLen[in]	签名后的数据长度
	pucSignerCertificate[out]	签名者证书
	puiSignerCertificateLen[in,out]	签名者证书长度
	puiDigestAlgorithms[out]	HASH 算法
	pucData[out]	被签名的数据
	puiDataLen[in,out]	被签名的数据长度
	pucSign[out]	签名值
	puiSignLen[in,out]	签名值的长度
返回值：	0	成功
	非 0	失败,返回错误代码

7.4.6 编码 PKCS#7 格式的数字信封

原型：

```
int SAF_Pkcs7_EncodeEnvelopedData(
    void * hAppHandle,
    unsigned char * pucData,
    unsigned int uiDataLen,
    unsigned char * pucEncCertificate,
    unsigned int uiEncCertificateLen,
    unsigned int uiSymmAlgorithm,
    unsigned char * pucDerP7EnvelopedData,
    unsigned int * puiDerP7EnvelopedDataLen);
```

描述：编码 PKCS#7 格式的数字信封数据。

参数：	hAppHandle[in]	应用接口句柄
	pucData[in]	需要做数字信封的数据
	uiDataLen[in]	需要做数字信封的数据长度
	pucEncCertificate[in]	接收者证书
	uiEncCertificateLen[in]	接收者证书长度
	uiSymmAlgorithm[in]	对称算法参数
	pucDerP7EnvelopedData[out]	编码后的数字信封数据
	puiDerP7EnvelopedDataLen[in,out]	编码后的数字信封数据长度
返回值：	0	成功
	非 0	失败,返回错误代码

7.4.7 解码 PKCS#7 格式的数字信封

原型：

```
int SAF_Pkcs7_DecodeEnvelopedData(
    void * hAppHandle,
    unsigned char * pucDecContainerName,
    unsigned int uiDecContainerNameLen,
    unsigned int uiDecKeyUsage,
    unsigned char * pucDerP7EnvelopedData,
    unsigned int uiDerP7EnvelopedDataLen,
```

```
    unsigned char * pucData,
    unsigned int * puiDataLen);
```

描述：解码 PKCS#7 格式的数字信封数据。

参数：

hAppHandle[in]	应用接口句柄
pucDecContainerName[in]	解密用私钥的容器名
uiDecContainerNameLen[in]	解密用私钥的容器名长度
uiDecKeyUsage[in]	解密用私钥的用途
pucDerP7EnvelopedData[in]	数字信封数据
uiDerP7EnvelopedDataLen[in]	数字信封数据长度
pucData[out]	解码后得到的数据原文
puiDataLen [in,out]	解码后得到的数据原文数据长度

返回值：

0	成功
非 0	失败，返回错误代码

7.4.8 编码 PKCS#7 格式的摘要数据

原型：

```
int SAF_Pkcs7_EncodeDigestedData(
    void * hAppHandle,
    unsigned int uiDigestAlgorithm,
    unsigned char * pucData,
    unsigned int puiDataLen,
    unsigned char * pucDerP7DigestedData,
    unsigned int * puiDerP7DigestedDataLen);
```

描述：使用指定的杂凑算法计算原文的摘要，并打包成符合 PKCS#7 摘要数据格式的数据包。

参数：

hAppHandle[in]	应用接口句柄
uiDigestAlgorithm[in]	杂凑算法标识
pucData[in]	原文数据
puiDataLen [in]	原文数据长度
pucDerP7DigestedData[out]	符合 PKCS#7 摘要数据格式的数据包
puiDerP7DigestedDataLen[in,out]	符合 PKCS#7 摘要数据格式的数据包长度

返回值：

0	成功
非 0	失败，返回错误代码

7.4.9 解码 PKCS#7 格式的摘要数据

原型：

```
int SAF_Pkcs7_DecodeDigestedData(
    void * hAppHandle,
    unsigned int uiDigestAlgorithm,
    unsigned char * pucDerP7DigestedData,
    unsigned int puiDerP7DigestedDataLen,
    unsigned char * pucData,
    unsigned int * puiDataLen,
    unsigned char * pucDigest,
    unsigned int * puiDigestLen);
```

描述：　从符合 PKCS#7 摘要数据格式的数据包中解析出原文及摘要值。

参数：

hAppHandle[in]	应用接口句柄
uiDigestAlgorithm[in]	杂凑算法标识
pucDerP7DigestedData[in]	符合 PKCS#7 摘要数据格式的数据包
puiDerP7DigestedDataLen[in]	符合 PKCS#7 摘要数据格式的数据包长度
pucData[out]	原文数据
puiDataLen [in,out]	原文数据长度
pucDigest[out]	摘要值
puiDigestLen[in,out]	摘要值长度

返回值：

0	成功
非 0	失败，返回错误代码

7.4.10　编码基于 SM2 算法的带签名的数字信封数据

原型：

```
int SAF_SM2_EncodeSignedAndEnvelopedData(
    void * hAppHandle,
    unsigned char * pucSignContainerName,
    unsigned int uiSignContainerNameLen,
    unsigned int uiSignKeyUsage,
    unsigned char * pucSignerCertificate,
    unsigned int uiSignerCertificateLen,
    unsigned int uiSigestAlgorithms,
    unsigned char * pucEncCertificate,
    unsigned int uiEncCertificateLen,
    unsigned int uiSymmAlgorithm,
    unsigned char * pucData,
    unsigned int uiDataLen,
    unsigned char * pucDerSignedAndEnvelopedData,
    unsigned int * puiDerSignedAndEnvelopedDataLen);
```

描述：　编码基于 SM2 算法的带签名的数字信封数据。

参数：

hAppHandle[in]	应用接口句柄
pucSignContainerName[in]	签名私钥的容器名
uiSignContainerNameLen[in]	签名私钥的容器名长度
uiSignKeyUsage[in]	私钥的用途
pucSignerCertificate[in]	签名者证书
uiSignerCertificateLen[in]	签名者证书长度
uiSigestAlgorithms[in]	HASH 算法
pucEncCertificate[in]	接收者证书
uiEncCertificateLen[in]	接收者证书长度
uiSymmAlgorithm[in]	对称算法参数
pucData[in]	原始数据
uiDataLen[in]	原始数据长度
pucDerSignedAndEnvelopedData [out]	Der 编码后的 SignedAndEnvelopedData 数字信封数据，SignedAndEnvelopedData 数据格式定义遵循 GM/T 0010

	puiDerSignedAndEnvelopedDataLen[in,out]	带签名的数字信封数据长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.4.11 解码基于 SM2 算法的带签名的数字信封数据

原型：
```
int SAF_SM2_DecodeSignedAndEnvelopedData(
    void * hAppHandle,
    unsigned char * pucDecContainerName,
    unsigned int uiDecContainerNameLen,
    unsigned int uiDecKeyUsage,
    unsigned char * pucDerSignedAndEnvelopedData,
    unsigned int uiDerSignedAndEnvelopedDataLen,
    unsigned char * pucData,
    unsigned int * puiDataLen,
    unsigned char * pucSignerCertificate,
    unsigned int * puiSignerCertificateLen,
    unsigned int * puiDigestAlgorithms);
```

描述：解码基于 SM2 算法的带签名的数字信封数据。

参数：

hAppHandle[in]	应用接口句柄
pucDecContainerName[in]	解密用私钥的容器名
uiDecContainerNameLen[in]	解密用私钥的容器名长度
uiDecKeyUsage[in]	解密用私钥的用途
pucDerSignedAndEnvelopedData[in]	Der 编码后的 SignedAndEnvelopedData 数字信封数据，SignedAndEnvelopedData 数据格式定义遵循 GM/T 0010
uiDerSignedAndEnvelopedDataLen[in]	编码后的带签名的数字信封数据长度
pucData[out]	解码后得到的原始数据
puiDataLen[in,out]	原始数据长度
pucSignerCertificate[out]	签名者证书
puiSignerCertificateLen[in,out]	签名者证书长度
puiDigestAlgorithms[out]	HASH 算法标识

返回值：0 成功；非 0 失败，返回错误代码

7.4.12 编码基于 SM2 算法的签名数据

原型：
```
int SAF_SM2_EncodeSignedData(
    void * hAppHandle,
    unsigned char * pucSignContainerName,
    unsigned int uiSignContainerNameLen,
    unsigned int uiSignKeyUsage,
    unsigned char * pucSignerCertificate,
    unsigned int uiSignerCertificateLen,
    unsigned int uiDigestAlgorithms,
```

```
    unsigned char * pucData,
    unsigned int uiDataLen,
    unsigned char * pucDerSignedData,
    unsigned int * puiDerSignedDataLen);
```

描述：编码基于 SM2 算法的签名数据。

参数：

hAppHandle[in]	应用接口句柄
pucSignContainerName[in]	签名私钥的容器名
uiSignContainerNameLen[in]	签名私钥的容器名长度
uiSignKeyUsage[in]	私钥的用途
pucSignerCertificate[in]	签名者证书
uiSignerCertificateLen[in]	签名者证书长度
uiDigestAlgorithms[in]	HASH 算法标识
pucData[in]	需要签名的数据
uiDataLen[in]	需要签名的数据长度
pucDerSignedData[out]	Der 编码的 SignedData 签名数据，SignedData 数据格式定义遵循 GM/T 0010
puiDerSignedDataLen[in,out]	带签名数据长度

返回值：

0	成功
非 0	失败，返回错误代码

7.4.13 解码基于 SM2 算法的签名数据

原型：

```
int SAF_SM2_DecodeSignedData(
    void * hAppHandle,
    unsigned char * pucDerSignedData,
    unsigned int uiDerSignedDataLen,
    unsigned char * pucSignerCertificate,
    unsigned int * puiSignerCertificateLen,
    unsigned int * puiDigestAlgorithms,
    unsigned char * pucData,
    unsigned int * puiDataLen,
    unsigned char * pucSign,
    unsigned int * puiSignLen);
```

描述：解码基于 SM2 算法的签名数据。

参数：

hAppHandle[in]	应用接口句柄
pucDerSignedData[in]	Der 编码的 SignedData 签名数据，SignedData 数据格式定义遵循 GM/T 0010
uiDerSignedDataLen[in]	签名数据的长度
pucSignerCertificate[out]	Der 编码的签名者证书
puiSignerCertificateLen[in,out]	签名者证书长度
puiDigestAlgorithms[out]	HASH 算法标识
pucData[out]	被签名的数据
puiDataLen[in,out]	被签名的数据长度
pucSign[out]	签名值

	puiSignLen[in,out]	签名值的长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.4.14 编码基于 SM2 算法的数字信封

原型：
```
int SAF_SM2_EncodeEnvelopedData(
    void *hAppHandle,
    unsigned char *pucData,
    unsigned int uiDataLen,
    unsigned char *pucEncCertificate,
    unsigned int uiEncCertificateLen,
    unsigned int uiSymmAlgorithm,
    unsigned char *pucDerEnvelopedData,
    unsigned int *puiDerEnvelopedDataLen);
```

描述：编码基于 SM2 算法的数字信封数据。

参数：	hAppHandle[in]	应用接口句柄
	pucData[in]	需要做数字信封的数据
	uiDataLen[in]	需要做数字信封的数据长度
	pucEncCertificate[in]	Der 编码的接收者证书
	uiEncCertificateLen[in]	接收者证书长度
	uiSymmAlgorithm[in]	对称算法标识
	pucDerEnvelopedData[out]	Der 编码后的 EnvelopedData 数字信封数据，EnvelopedData 数据格式定义遵循 GM/T 0010
	puiDerEnvelopedDataLen[in,out]	编码后的数字信封数据长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.4.15 解码基于 SM2 算法的数字信封

原型：
```
int SAF_SM2_DecodeEnvelopedData(
    void *hAppHandle,
    unsigned char *pucDecContainerName,
    unsigned int uiDecContainerNameLen,
    unsigned int uiDecKeyUsage,
    unsigned char *pucDerEnvelopedData,
    unsigned int uiDerEnvelopedDataLen,
    unsigned char *pucData,
    unsigned int *puiDataLen);
```

描述：解码基于 SM2 算法的数字信封数据。

参数：	hAppHandle[in]	应用接口句柄
	pucDecContainerName[in]	解密用私钥的容器名
	uiDecContainerNameLen[in]	解密用私钥的容器名长度
	uiDecKeyUsage[in]	解密用私钥的用途

	pucDerEnvelopedData[in]	Der 编码的 EnvelopedData 数字信封数据，EnvelopedData 数据格式定义遵循 GM/T 0010
	uiDerEnvelopedDataLen[in]	数字信封数据长度
	pucData[out]	解码后得到的数据原文
	puiDataLen [in,out]	解码后得到的数据原文长度
返回值：	0	成功
	非 0	失败，返回错误代码

附 录 A
（规范性附录）
密码服务接口错误代码定义

宏描述	预定义值	说 明
SAR_OK	0	成功
SAR_UnknownErr	0X02000001	异常错误
SAR_NotSupportYetErr	0X02000002	不支持的服务
SAR_FileErr	0X02000003	文件操作错误
SAR_ProviderTypeErr	0X02000004	服务提供者参数类型错误
SAR_LoadProviderErr	0X02000005	导入服务提供者接口错误
SAR_LoadDevMngApiErr	0X02000006	导入设备管理接口错误
SAR_AlgoTypeErr	0X02000007	算法类型错误
SAR_NameLenErr	0X02000008	名称长度错误
SAR_KeyUsageErr	0X02000009	密钥用途错误
SAR_ModulusLenErr	0X02000010	模的长度错误
SAR_NotInitializeErr	0X02000011	未初始化
SAR_ObjErr	0X02000012	对象错误
SAR_MemoryErr	0X02000100	内存错误
SAR_TimeoutErr	0X02000101	超时
SAR_IndataLenErr	0X02000200	输入数据长度错误
SAR_IndataErr	0X02000201	输入数据错误
SAR_GenRandErr	0X02000300	生成随机数错误
SAR_HashObjErr	0X02000301	HASH 对象错
SAR_HashErr	0X02000302	HASH 运算错误
SAR_GenRsaKeyErr	0X02000303	产生 RSA 密钥错
SAR_RsaModulusLenErr	0X02000304	RSA 密钥模长错误
SAR_CspImprtPubKeyErr	0X02000305	CSP 服务导入公钥错误
SAR_RsaEncErr	0X02000306	RSA 加密错误
SAR_RSGDecErr	0X02000307	RSA 解密错误
SAR_HashNotEqualErr	0X02000308	HASH 值不相等
SAR_KeyNotFountErr	0X02000309	密钥未发现
SAR_CertNotFountErr	0X02000310	证书未发现

表（续）

宏描述	预定义值	说　明
SAR_NotExportErr	0X02000311	对象未导出
SAR_CertRevokedErr	0X02000316	证书被吊销
SAR_CertNotYetValidErr	0X02000317	证书未生效
SAR_CertHasExpiredErr	0X02000318	证书已过期
SAR_CertVerifyErr	0X02000319	证书验证错误
SAR_CertEncodeErr	0X02000320	证书编码错误
SAR_DecryptPadErr	0X02000400	解密时做补丁错误
SAR_MacLenErr	0X02000401	MAC 长度错误
SAR_KeyInfoTypeErr	0X02000402	密钥类型错误
SAR_NotLogin	0X02000403	没有进行登录认证

参 考 文 献

[1] GB/T 17964—2008 信息安全技术 分组密码算法的工作模式

[2] GB/T 17903.1—2008 信息技术 安全技术 抗抵赖 第1部分:概述

[3] GB/T 17903.2—2008 信息技术 安全技术 抗抵赖 第2部分:使用对称技术的机制

[4] GB/T 17903.3—2008 信息技术 安全技术 抗抵赖 第3部分:使用非对称技术的机制

[5] GB/T 18238.1—2000 信息技术 安全技术 散列函数 第1部分:概述

[6] GB/T 18238.2—2002 信息技术 安全技术 散列函数 第2部分:采用n位块密码的散列函数

[7] GB/T 18238.3—2002 信息技术 安全技术 散列函数 第3部分:专用散列函数

[8] GB/T 19713—2005 信息技术 安全技术 公钥基础设施 在线证书状态协议

[9] GB/T 19771—2005 信息技术 安全技术 公钥基础设施 PKI组件最小互操作规范

[10] GB 15851 信息技术 安全技术 带消息恢复的数字签名方案

[11] RFC 2560 X.509互联网公开密钥基础设施在线证书状态协议—OCSP

[12] RFC 2459 X.509互联网公开密钥基础设施证书和CRL轮廓

[13] RSA Security:Public-Key Cryptography Standards (PKCS)

[14] Pkcs#11 Cryptographic Token Interface Standard

[15] IETF Rfc2459,Internet X.509 Public Key Infrastructure Certificate and CRL Profile

[16] IETF Rfc2560,X.509 Internet Public Key Infrastructure Online Certificate Status Protocol

[17] IETF Rfc1777,Lightweight Directory Access Protocol

[18] IETF Rfc2587 Internet X.509 Public Key Infrastructure LDAPv2 Schema

[19] IETF Rfc3647,Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework

[20] ISO/IEC 8825-1:1998 信息技术-ASN.1编码规则:基本编码规则(BER)的规范,正规编码规则(CER)和可区分编码规则(DER)

中华人民共和国密码
行业标准
通用密码服务接口规范
GM/T 0019—2012
*
中国标准出版社出版发行
北京市朝阳区和平里西街甲2号(100013)
北京市西城区三里河北街16号(100045)
网址 www.spc.net.cn
总编室:(010)64275323 发行中心:(010)51780235
读者服务部:(010)68523946
中国标准出版社秦皇岛印刷厂印刷
各地新华书店经销
*
开本 880×1230 1/16 印张 3.25 字数 94 千字
2013年1月第一版 2013年1月第一次印刷
*
书号:155066·2-24384 定价 45.00 元

如有印装差错 由本社发行中心调换
版权专有 侵权必究
举报电话:(010)68510107

GM/T 0019-2012