公钥密码基础设施应用技术体系

通用密码服务接口规范

Public Key Infrastructure Application Technology

Interface Specifications of Cryptograph Service

国 家 密 码 管 理 局

2010 年 8 月

目　　次

前　　言 II
引　　言 III
1 范围 1
2 规范性引用文件 1
3 术语和定义 1
4 符号和缩略语 1
5 算法标识和数据结构 2
5.1 密码服务接口定义 2
5.2 密码服务接口数据结构定义和说明 3
6 密码服务接口 5
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 5
6.2 密码服务接口接口组成和功能说明 6
7 密码服务接口函数定义 7
7.1 环境类函数 7
7.2 证书类函数 9
7.3 密码运算类函数 15
7.4 消息类函数 37
附录 A (规范性附录) 密码服务接口错误代码定义 43

前　言

本规范是《公钥密码基础设施应用技术体系框架规范》系列规范之一，本规范为应用技术体系框架的典型密码服务层和应用层定义了统一的密码服务接口。

本规范的附录A为规范性附录，附录B为资料性附录。

本规范由国家密码管理局提出并归口。

本规范主要起草单位：北京数字证书认证中心、上海数字证书认证中心、卫士通信息产业股份有限公司、兴唐通信科技股份有限公司、济南得安计算机技术有限公司、北京海泰方圆科技有限公司。

本规范主要起草人：李述胜、崔久强、高志权、李元正、徐强、柳增寿等。

本规范责任专家：刘平。

本规范凡涉及密码算法相关内容，按国家有关法规实施。

引　　言

本规范依托于密码设备层的《公钥密码基础设施应用技术体系 密码设备应用接口规范》，向上为典型密码服务层和应用层规定了统一的、与密码协议无关、与密钥管理无关、与密码设备管理无关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。

本规范不涉及任何具体的密码运算，所有密码运算均在符合国家有关法规的密码设备中进行。

本规范编制过程中得到了国家商用密码应用技术体系总体工作组的指导。

公钥密码基础设施应用技术体系

通用密码服务接口规范

1 范围

本规范规定了与密码协议无关，与密钥管理无关，与密码设备管理无关的通用密码服务接口。

本规范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

下列标准所包含的条文，通过本规范中的引用而构成本规范的条文。考虑到标准的修订，使用本规范时，应研究使用下列标准最新版本的可能性。

GB/T 20518-2006 信息安全技术　公钥基础设施　数字证书格式

GB/T xxxxx 证书认证系统密码及其相关安全技术规范

GB/T AAAAA 信息技术 安全技术 密码术语

GB/T BBBBB 公钥密码基础设施应用技术体系 框架规范

GB/T CCCCC 公钥密码基础设施应用技术体系 密码设备应用接口规范

GB/T DDDDD 智能IC及智能密码钥匙密码应用接口规范

3 术语和定义

以下术语和定义适用于本规范。

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 pair

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存储空间的唯一性编号。

4 符号和缩略语

下列缩略语适用于本部分：

API	应用程序接口（Application Program Interface），简称应用接口
CA	证书认证机构（Certification Authority）
CN	通用名（Common Name）
CRL	证书撤销列表（Certificate Revocation List）
CSP	加密服务提供者（Cryptographic Service Provider）
DER	可区分编码规则（Distinguished Encoding Rules）
DN	可识别名（Distinguished Name）
ECC	椭圆曲线算法（Elliptic Curve Cryptography）
LDAP	轻量级目录访问协议（Lightweight Directory Access Protocol）
OID	对象标识符（Object Identifier）

PKCS　公钥密码标准(the Public-Key Cryptography Standard)
SDS　国标密码设备应用接口

5 算法标识和数据结构

5.1 密码服务接口定义

5.1.1 常量定义

常量定义		
宏描述	预定义值	说明
#define SGD_TRUE	0x00000001	布尔值为真
#define SGD_FALSE	0x00000000	布尔值为假

5.1.2 全局参数定义

对称算法标识		
宏描述	预定义值	说明
#define SGD_SM1_ECB	0x00000101	SM1 算法 ECB 加密模式
#define SGD_SM1_CBC	0x00000102	SM1 算法 CBC 加密模式
#define SGD_SM1_CFB	0x00000104	SM1 算法 CFB 加密模式
#define SGD_SM1_OFB	0x00000108	SM1 算法 OFB 加密模式
#define SGD_SM1_MAC	0x00000110	SM1 算法 MAC 加密模式
#define SGD_SSF33_ECB	0x00000201	SSF33 算法 ECB 加密模式
#define SGD_SSF33_CBC	0x00000202	SSF33 算法 CBC 加密模式
#define SGD_SSF33_CFB	0x00000204	SSF33 算法 CFB 加密模式
#define SGD_SSF33_OFB	0x00000208	SSF33 算法 OFB 加密模式
#define SGD_SSF33_MAC	0x00000210	SSF33 算法 MAC 加密模式
非对称算法标识		
宏描述	预定义值	说明
#define SGD_RSA	0x00010000	RSA 算法
#define SGD_SM2_1	0x00020100	椭圆曲线签名算法
#define SGD_SM2_2	0x00020200	椭圆曲线密钥交换协议
#define SGD_SM2_3	0x00020400	椭圆曲线加密算法
杂凑算法标识		
宏描述	预定义值	说明
#define SGD_SM3	0x00000001	SM3 杂凑算法
#define SGD_SHA1	0x00000002	SHA1 杂凑算法
#define SGD_SHA256	0x00000004	SHA256 杂凑算法
接口描述标识		
宏描述	预定义值	说明
#define SGD_PROVIDER_CSP	0x00000001	CSP 接口
#define SGD_PROVIDER_PKCS11	0x00000002	PKCS#11 接口
#define SGD_PROVIDER_SDS	0x00000003	国标密码设备应用接口
#define SGD_KEYUSAGE_SIGN	0x00000001	签名/验证的密钥用途
#define SGD_KEYUSAGE_KEYEXCHANGE	0x00000002	加/解密的密钥用途
#define SGD_MODE_ECB	0x00000001	ECB 模式

#define SGD_MODE_CBC	0x00000002	CBC 模式
#define SGD_MODE_CFB	0x00000003	CFB 模式
#define SGD_MODE_OFB	0x00000004	OFB 模式
#define SGD_KEYINFO_DEV_GENERATE	0x00000001	设备产生
#define SGD_KEYINFO_KEY	0x00000002	外部输入 KEY
#define SGD_KEYINFO_IV	0x00000003	外部输入 IV
#define SGD_KEYINFO_PASSWORD_DERIVE_KEY	0x00000004	通过口令生成 KEY
证书解析标识		
宏描述	预定义值	说明
#define SGD_GET_CERT_VERSION	0x00000001	证书版本
#define SGD_GET_CERT_SERIAL	0x00000002	证书序列号
#define SGD_GET_CERT_ISSUER	0x00000005	证书颁发者信息
#define SGD_GET_CERT_VALID_TIME	0x00000006	证书有效期
#define SGD_GET_CERT_SUBJECT	0x00000007	证书拥有者信息
#define SGD_GET_CERT_DER_PUBLIC_KEY	0x00000008	证书公钥信息
#define SGD_GET_CERT_DER_EXTENSIONS	0x00000009	证书扩展项信息
#define SGD_EXT_AUTHORITYKEYIDENTIFIER	0x00000011	颁发者密钥标示符
#define SGD_EXT_SUBJECTKEYIDENTIFIER	0x00000012	证书持有者密钥标示符
#define SGD_EXT_KEYUSAGE	0x00000013	密钥用途
#define SGD_EXT_PRIVATEKEYUSAGEPERIOD	0x00000014	私钥有效期
#define SGD_EXT_CERTIFICATEPOLICIES	0x00000015	证书策略
#define SGD_EXT_POLICYMAPPINGS	0x00000016	策略映射
#define SGD_EXT_BASICCONSTRAINTS	0x00000017	基本限制
#define SGD_EXT_POLICYCONSTRAINTS	0x00000018	策略限制
#define SGD_EXT_EXTKEYUSAGE	0x00000019	扩展密钥用途
#define SGD_EXT_CRLDISTRIBUTIONPO	0x00000020	CRL 发布点
#define SGD_EXT_NETSCAPE_CERT_TYPE	0x00000021	netscape 属性
#define SGD_EXT_SELFDEFINED_EXTENSION	0x00000022	私有的自定义扩展项
#define SGD_EXT_IDENTIFYCARDNUMBER	0x00000023	个人身份证号码
#define SGD_EXT_INURANCENUMBER	0x00000024	个人社会保险号
#define SGD_EXT_ICREGISTRATIONNUMBER	0x00000025	企业工商注册号
#define SGD_EXT_ORGANIZATIONCODE	0x00000026	企业组织机构代码
#define SGD_EXT_TAXATIONNUMBER	0x00000027	企业税号
#define SGD_MAX_NAME_SIZE	0x00000080	名称最大长度
#define SGD_MAX_COUNT	0x00000100	列表最大长度
#define SGD_MAX_CONTAINER	0x00000800	容器容量

5.2 密码服务接口数据结构定义和说明

5.2.1 用户证书列表

字段名称	含义
certCount	证书总数
certificate	DER 编码的数字证书
certificateLen	数字证书的长度

containerName	容器名称
containerNameLen	容器名称的长度
keyUsage	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;
```

5.2.2 用户密钥列表

字段名称	含义
keyPairCount	密钥对总数
containerName	容器名称
containerNameLen	容器名称的长度
keyUsage	密钥用途

实际数据结构定义：

```
typedef struct SGD_USR_KEYPAIR_ENUMLIST_ {
    unsigned int keyPairCount;
    unsigned char *containerName[SGD_MAX_COUNT];
    unsigned int containerNameLen[SGD_MAX_COUNT];
    unsigned int keyUsage[SGD_MAX_COUNT];
} SGD_USR_KEYPAIR_ENUMLIST;
```

5.2.3 证书中 DN 的结构

字段名称	数据长度（字节）	含义
dn_c	256	国家名称数组
dn_c_len	8	国家数组的长度
dn_s	256	省份或直辖市名称数组
dn_s_len	8	省份或直辖市名称数组的长度
dn_l	256	城市或地区的名称数组
dn_l_len	8	城市或地区的名称数组的长度
dn_o	640	机构名称数组
dn_o_len	20	机构名称数组的长度
dn_ou	640	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[2][SGD_MAX_NAME_SIZE];
    unsigned int dn_c_len[2];
    unsigned char dn_s[2][SGD_MAX_NAME_SIZE];
    unsigned int dn_s_len[2];
    unsigned char dn_l[2][SGD_MAX_NAME_SIZE];
    unsigned int dn_l_len[2];
    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：

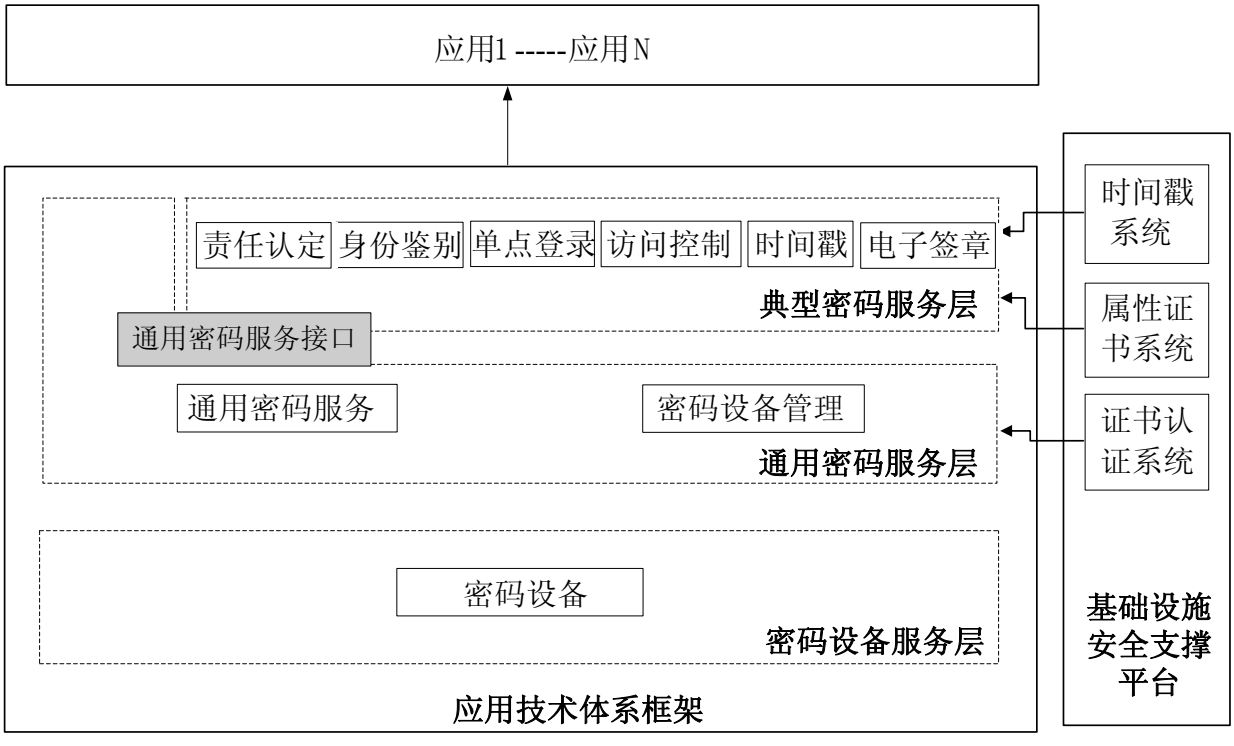

图1 通用密码服务接口在公钥密码基础设施应用技术体系框架内的位置

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安全程序空间，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，一种是普通用户，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私有的信息和数据；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，可以访问密码设备中的公有信息，设置和修改普通用户的PIN。

应用程序在使用密码服务接口时，要首先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。在不再调用任何密码服务函数，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、可信根CA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。

证书类函数在使用中，应先通过调用证书和CRL获取函数或其他途径，获取相关证书和CRL,然后调用相关地证书设置函数：添加信任的CA根证书、添加CA证书、设置用户证书、设置证书应用环境，然后调用相关函数进行证书验证，实现用户身份认证，还可以调用取证书信息和取证书扩展信息获取证书中有关信息，用于实现授权管理和访问控制等安全机制。

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地密码运算，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。

密码服务类函数在整个密码服务接口中属于底层函数，并不是所有的具体函数都必须在应用程序中使用。应用程序可在必要时候直接调用密码服务函数，根据业务需要，结合消息函数来使用。

6.2.5 消息类函数

消息类函数主要是将数据按照PKCS7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PKCS7格式的数据编解码、PKCS7格式的签名数据编解码和PKCS7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 环境类函数概述

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);
描述： 初始化密码服务程序空间
参数： phAppHandle[out] 返回应用接口句柄
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：0xAAAABBBB,其中 AAAA 为主版本号，BBBB 为次版本号。

7.1.5 用户登录

```
原型： int SAF_Login (void *hAppHandle,
            unsigned int uiUsrType,
            unsigned int uiKeyIndex,
            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。
	uiKeyIndex[in]	密钥检索号
	pucPin[in]	设备口令
	uiPinLen[in]	设备口令长度
	puiRemainCount[out]	口令剩余尝试次数
返回值：	0	成功
	非 0	失败，返回错误代码

备注： uiKeyIndex 参数只有在服务器端使用才有效，用于标识密码设备内部的密钥位置。

7.1.6 修改 PIN

原型： int SAF_ChangePin (void *hAppHandle,
unsigned int uiUsrType,
unsigned char *pucOldPin,
unsigned int uiOldPinLen,
unsigned char *pucNewPin,
unsigned int uiNewPinLen,
unsigned int *puiRemainCount);

描述： 修改设备 PIN

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
	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]	口令剩余尝试次数
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 证书类函数概述

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 方式根据证书获取 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_EnumKeyPairs
S. 释放列举用户证书的内存：SAF_EnumCertificatesFree

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[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[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： 使用证书链验证用户证书的有效性。
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

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注销
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
非 0 失败，返回错误代码

7.2.13 根据 OCSP 获取证书状态

原型： int SAF_GetCertificateStateByOCSP(
void *hAppHandle,
char *pcOcspHostIp,
unsigned int uiOcspPort,
unsigned char *pucUsrCertificate,
unsigned int uiUsrCertificateLen);
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
pcOcspHostIp[in] ocsp 服务器 IP 地址
uiOcspPort[in] ocsp 服务器端口
pucUsrCertificate[in] DER 编码证书
uiUsrCertificateLen[in] 证书长度
返回值： 0 成功
非 0 失败，返回错误代码

7.2.14 通过 LDAP 方式获取证书

原型： int SAF_GetCertFromLdap(
void *hAppHandle,
char *pcLdapHostIp,
unsigned int uiLdapPort,
unsigned char *pucSubDN,
unsigned int uiSubDNLen,
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
pcLdapHostIp[in]	ldap 服务器 IP 地址
uiLdapPort[in]	ldap 服务器端口
pucSubDN[in]	需要查找的证书的 DN
uiSubDNLen[in]	需要查找的证书的 DN 长度
pucOutCert[out]	找到的 DER 编码的证书
puiOutCertLen[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 获取证书对应的 CRL

原型： int SAF_GetCrlFromLdap (
void *hAppHandle,
char * pucLdapHostIp,
unsigned int uiLdapPort,
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
pucLdapHostIp[in]	ldap 服务器 IP 地址
uiLdapPort [in]	ldap 服务器端口
pucCertificate [in]	证书
uiCertificateLen [in]	证书长度
pucCrlData [out]	获取的 DER 编码的 CRL 文件
puiCrlDataLen [out]	CRL 文件长度

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
	pucInfo[out]	获取的证书信息
	puiInfoLen[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 (
    unsigned char *pucDerCertExt,
    unsigned int uiPerCertExtLen,
    unsigned int uiInfoType,
    unsigned char * pucPriOid,
    unsigned int uiPriOidLen,
    unsigned char *pucInfo,
    unsigned int  *puiInfoLen);
```

描述： 取证书的扩展信息。

参数：		
	pucDerCertExt[in]	全体扩展项地址
	uiPerCertExtLen[in]	全体扩展项长度
	uiInfoType[in]	指定的证书扩展项解析标识
	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[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： 列举出当前设备中所有证书

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。	

7.2.19 列举用户的密钥对容器信息

原型： Int SAF_EnumKeyPairs(
    void *hAppHandle,
    SGD_USR_KEYPAIR_ENUMLIST *usrKeyPairs);

描述： 列举用户密码终端中所有密钥对所对应的容器名。

参数：	hAppHandle[in]	应用接口句柄
	usrKeyPairs[out]	用户密钥对所对应的容器名信息
返回值：	SAR_OK	成功
	其他	失败
备注：	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]	用户证书信息
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 密码运算类函数概述

密码服务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 公钥加密运算：SAF_RsaPublicKeyEnc
R. RSA 私钥解密运算：SAF_RsaPrivateKeyDec
S. RSA 的签名运算：SAF_RsaSign
T. 对文件进行 RSA 签名运算：SAF_RsaSignFile
U. RSA 验证签名运算：SAF_RsaVerifySign
V. 对文件及其签名进行 RSA 验证：SAF_RsaVerifySignFile
W. 基于证书的 RSA 公钥加密：SAF_RsaPublicKeyEncByCert
X. 基于证书的 RSA 公钥验证：SAF_VerifySignByCert
Y. 生成 ECC 密钥对:SAF_GenEccKeyPair
Z. 取出 ECC 公钥:SAF_GetEccPublicKey
AA. ECC 签名:SAF_EccSign
BB. ECC 验证:SAF_EccVerifySign
CC. ECC 公钥加密:SAF_EccPublicKeyEnc
DD. ECC 私钥解密:SAF_ EccPrivateKeyDec
EE. 基于证书的 ECC 公钥加密:SAF_ EccPublicKeyEncByCert
FF. 基于证书的 ECC 公钥验证:SAF_ EccVerifySignByCert
GG. 创建对称算法对象：SAF_CreateSymmAlgoObj
HH. 生成会话密钥并用外部公钥加密输出:SAF GenerateKeyWithEPK
II. 导入加密的会话密钥：SAF_ImportEncedKey
JJ. 生成密钥协商参数并输出：SAF_GenerateAgreementDataWithECC
KK. 计算会话密钥：SAF_GenerateKeyWithECC
LL. 产生协商数据并计算会话密钥：SAF_GenerateAgreementDataAndKeyWithECC
MM. 销毁对称算法对象：SAF_DestroySymmAlgoObj
NN. 销毁会话密钥句柄：SAF_DestroyKeyHandle
OO. 单块加密运算：SAF_SymmEncrypt
PP. 多块加密运算：SAF_SymmEncryptUpdate
QQ. 结束加密运算：SAF_SymmEncryptFinal
RR. 单块解密运算：SAF_SymmDecrypt
SS. 多块解密运算：SAF_SymmDecryptUpdate
TT. 结束解密运算：SAF_SymmDecryptFinal
UU. 单块 MAC 运算：SAF_Mac
VV. 多块 MAC 运算：SAF_MacUpdate
WW. MAC 运算结束：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[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[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[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[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[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[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]	随机数，长度为 randLen

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 *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
SGD_SM3	SM3 算法
SGD_SHA1	SHA1 算法
SGD_SHA256	SHA256 算法
pucInData[in]	输入数据
uiInDataLen[in]	输入数据长度
pucOutData[out]	HASH
puiOutDataLen[out]	HASH 长度

返回值：

0	成功
非 0	失败，返回错误代码

7.3.12 创建 HASH 对象

原型： int SAF_CreateHashObj(
    void **phHashObj,
    unsigned int uiAlgorithmType);
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
    SGD_SM3 SM3 算法
    SGD_SHA1 SHA1 算法
    SGD_SHA256 SHA256 算法
返回值： 0 成功
非 0 失败，返回错误代码

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,
            unsigned char *pucRsaKeyPairPin,
            unsigned int uiRsaKeyPairPinLen);
```

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
	pucRsaKeyPairPin[in]	密钥对保护口令
	uiRsaKeyPairPinLen[in]	密钥对保护口令长度
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
	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[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 公钥加密运算

```
原型：  int SAF_Pkcs1RsaPublicKeyEnc(
            unsigned char *pucPublicKey,
            unsigned int uiPublicKeyLen,
            unsigned char *pucInData,
            unsigned int uiInDataLen,
            unsigned char *pucOutData,
            unsigned int *puiOutDataLen);
```

描述： 按照 PKCS#1 的要求进行公钥加密运算。

参数：	pucPublicKey[in]	DER 编码的公钥
	uiPublicKeyLen[in]	公钥长度
	pucInData[in]	输入数据
	uiInDataLen[in]	输入数据长度
	pucOutData[out]	输出 DER 编码的密文数据
	puiOutDataLen[out]	输出数据长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.3.19 RSA 私钥解密运算

```
原型：  int SAF_Pkcs1RsaPrivateKeyDec(
            void *hAppHandle,
            unsigned char *pucContainerName,
            unsigned int uiContainerNameLen,
            unsigned int uiKeyUsage,
            unsigned char *pucRsaKeyPairPin,
            unsigned int uiRsaKeyPairPinLen,
            unsigned char *pucInData,
            unsigned int uiInDataLen,
            unsigned char *pucOutData,
            unsigned int *puiOutDataLen);
```

描述： 按照 PKCS#1 的要求进行私钥解密运算

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

	uiKeyUsage[in]	密钥用途
	SGD_KEYUSAGE_SIGN	签名
	SGD_KEYUSAGE_KEYEXCHANGE	加密
	pucRsaKeyPairPin[in]	设备保护 PIN
	uiRsaKeyPairPinLen[in]	设备保护 PIN 的长度
	pucInData[in]	输入数据
	uiInDataLen[in]	输入数据长度
	pucOutData[out]	输出数据
	puiOutDataLen[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：	如果已经登录成功，则忽略 rsaKeyPairPin	

7.3.20 RSA 签名运算

原型：
```
int SAF_RsaSign(
    void *hAppHandle,
    unsigned char *pucContainerName,
    unsigned int uiContainerNameLen,
    unsigned char *pucRsaKeyPairPin,
    unsigned int uiRsaKeyPairPinLen,
    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
	uiContainerNameLen[in]	密钥的容器名长度
	pucRsaKeyPairPin[in]	设备保护 PIN
	uiRsaKeyPairPinLen[in]	设备保护 PIN 的长度
	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[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

7.3.21 对文件进行 RSA 签名运算

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

```
    unsigned char *pucContainerName,
    unsigned int uiContainerNameLen,
    unsigned char *pucRsaKeyPairPin,
    unsigned int uiRsaKeyPairPinLen,
    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
	pucRsaKeyPairPin[in]	设备保护 PIN
	uiRsaKeyPairPinLen[in]	设备保护 PIN 的长度
	uiHashAlgorithmID[in]	HASH 算法
	pucFileName[in]	要签名的文件名
	pucSignData[out]	输出的 DER 格式的签名结果数据
	puiSignDataLen[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。

7.3.22 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

7.3.23 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 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
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。

7.3.24 基于证书的 RSA 公钥加密

原型： int SAF_ RsaPublicKeyEncByCert(
unsigned char *pucCertificate,
unsigned int uiCertificateLen,
unsigned char *pucInData,
unsigned int uiInDataLen,
unsigned char *pucOutData,
unsigned int *puiOutDataLen);

描述： 按照 PKCS#1 的要求，进行基于证书的 RSA 公钥加密

参数：

pucCertificate[in]	DER 编码的数字证书
uiCertificateLen[in]	DER 编码的数字证书长度
pucInData[in]	输入数据
uiInDataLen[in]	输入数据长度
pucOutData[out]	输出 DER 格式的密文数据
puiOutDataLen[out]	输出数据长度

返回值：

0	成功
非 0	失败，返回错误代码

7.3.25 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

7.3.26 生成 ECC 密钥对

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

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
	uiLeyUsage[in]	密钥用途
	SGD_SM2_	椭圆曲线签名
	SGD_SM2_	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
	pucEccKeyPairPin[in]	密钥对保护口令
	uiEccKeyPairPinLen[in]	密钥对保护口令长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.3.27 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_3	加密公钥
	pucPublicKey[out]	输出的 DER 编码的公钥数据
	puiPublicKeyLen[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

7.3.28 ECC 签名

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

原型：

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
	pucEccKeyPairPin[in]	设备保护 PIN
	uiEccKeyPairPinLen[in]	设备保护 PIN 的长度
	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[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： 对原文的杂凑运算在函数外部完成，得到待签名数据 inData 后，再调用本函数进行 ECC 签名。

7.3.29 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

备注： 对原文的杂凑运算在函数外部完成，得到待验证数据 inData 后，再调用本函数进行 ECC 签名。

7.3.30 ECC 公钥加密

原型： int SAF_EccPublicKeyEnc(
unsigned char *pucPublicKey,
unsigned int uiPublicKeyLen,
unsigned int uiAlgorithmID,
unsigned char *pucOnData,
unsigned int uiOnDataLen,
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
pucOnData[in]	输入数据
uiOnDataLen[in]	输入数据长度
pucOutData[out]	输出数据，DER 编码
puiOutDataLen[out]	输出数据长度

返回值：

0	成功
非 0	失败，返回错误代码

7.3.31 ECC 私钥解密

原型： int SAF_EccPrivateKeyDec(
    void *hAppHandle,
    unsigned char *pucContainerName,
    unsigned int uiContainerNameLen,
    unsigned int uiAlgorithmID,
    unsigned char *pucEccKeyPairPin,
    unsigned int ui#ccKeyPairPinLen,
    unsigned char *pucInData,
    unsigned int uiInDataLen,
    unsigned char *pucOutData,
    unsigned int *puiOutDataLen);

描述： ECC 私钥解密运算

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]	ECC 算法标识
	pucEccKeyPairPin[in]	设备保护 PIN
	ccKeyPairPinLen[in]	设备保护 PIN 的长度
	pucInData[in]	输入数据
	uiInDataLen[in]	输入数据长度
	pucOutData[out]	输出数据
	puiOutDataLen[out]	输出数据长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.3.32 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[out]	输出数据长度
返回值：	0	成功

	非 0	失败，返回错误代码

7.3.33 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 *pucOnData,
    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
pucOnData[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

备注：对原文的杂凑运算在函数外部完成，得到待验证数据 inData 后，再调用本函数进行 ECC 签名。

7.3.34 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

7.3.35 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

7.3.36 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[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

7.3.37 生成密钥协商参数并输出

```
原型：  int SAF_GenerateAgreementDataWithECC (
            void * hSymmKeyObj,
            unsigned int uiISKIndex,
            unsigned char *pucEccKeyPairPin,
            unsigned int uiEccKeyPairPinLen,
            unsigned int uiKeyBits,
```

```
        unsigned char *pucSponsorID,
        unsigned int uiSponsorIDLength,
        unsigned char *pucSponsorPublicKey,
        unsigned int *pucSponsorPublicKeyLen,
        unsigned char *pucSponsorTmpPublicKey,
        unsigned int *pucSponsorTmpPublicKeyLen,
        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
pucEccKeyPairPin[in]	设备保护 PIN
uiEccKeyPairPinLen[in]	设备保护 PIN 的长度
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
pucSponsorPublicKeyLen[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
pucSponsorTmpPublicKeyLen [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。
如果在具体的应用中，协商双方没有统一分配的 ID，可以将 ID 设定为常量。

7.3.38 计算会话密钥

```
int SAF_GenerateKeyWithECC (
        void *hAgreementHandle,
        unsigned char *pucResponseID,
        unsigned int uiResponseIDLength,
        unsigned char *pucResponsePublicKey,
        unsigned int pucResponsePublicKeyLen,
        unsigned char *pucResponseTmpPublicKey,
        unsigned int pucResponseTmpPublicKeyLen,
        void **phKeyHandle);
```

原型：

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
pucResponsePublicKeyLen[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
	pucResponseTmpPublicKeyLen[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。 如果在具体的应用中，协商双方没有统一分配的 ID，可以将 ID 设定为常量。	

7.3.39 产生协商数据并计算会话密钥

原型：

```
int SAF_GenerateAgreementDataAndKeyWithECC (
      void * hSymmKeyObj,
      unsigned char *pucEccKeyPairPin,
      unsigned int uiEccKeyPairPinLen,
      unsigned int uiISKIndex,
      unsigned int uiKeyBits,
      unsigned char *pucResponseID,
      unsigned int uiResponseIDLength,
      unsigned char *pucSponsorID,
      unsigned int uiSponsorIDLength,
      unsigned char *pucSponsorPublicKey,
      unsigned int  pucSponsorPublicKeyLen,
      unsigned char *pucSponsorTmpPublicKey,
      unsigned int  pucSponsorTmpPublicKeyLen,
      unsigned char *pucResponsePublicKey,
      unsigned int  *pucResponsePublicKeyLen,
      unsigned char *pucResponseTmpPublicKey
      unsigned int  *pucResponseTmpPublicKeyLen,
      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和密钥句柄。

参数：

hSymmKeyObj[in]	对称算法对象
pucEccKeyPairPin[in]	密钥对保护口令
uiEccKeyPairPinLen[in]	设备保护 PIN 的长度
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
pucSponsorTmpPublicKey[in]	外部输入的发起方临时 ECC 公钥
pucSponsorTmpPublicKeyLen[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
	pucResponsePublicKeyLen[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
	pucResponseTmpPublicKeyLen[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。 如果在具体的应用中，协商双方没有统一分配的 ID，可以将 ID 设定为常量	

7.3.40 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

7.3.41 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

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

7.3.43 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[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

7.3.44 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[out]	输出数据长度

返回值：

0	成功
非 0	失败，返回错误代码

7.3.45 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[out]	输出数据长度

返回值： 0 成功
非 0 失败，返回错误代码

7.3.46 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[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

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

7.3.48 单块 MAC 运算

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

7.3.49 多块 MAC 运算

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

描述：	继续 MAC 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	

7.3.50 MAC 运算结束

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

描述：	MAC 运算结束	
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 消息类函数概述

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 PKCS7 格式的带签名的数字信封数据：SAF_Pkcs7_EncodeData
B. 解码 PKCS7 格式的带签名的数字信封数据：SAF_Pkcs7_DecodeData
C. 编码 PKCS7 格式的签名数据：SAF_Pkcs7_EncodeSignedData
D. 解码 PKCS7 格式的签名数据：SAF_Pkcs7_DecodeSignedData
E. 编码 PKCS7 格式的数字信封数据：SAF_Pkcs7_EncodeEnvelopedData
F. 解码 PKCS7 格式的数字信封数据：SAF_Pkcs7_DecodeEnvelopedData
G. 编码 PKCS7 格式的摘要数据:SAF_Pkcs7_EncodeDigestedData
H. 解码 PKCS7 格式的摘要数据:SAF_Pkcs7_DecodeDigestedData

7.4.2 编码 PKCS7 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 uiSigestAlgorithms,
    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 PKCS7 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
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
pucDerP7Data[out]	编码后的数据
puiDerP7DataLen[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 PKCS7 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);
```

描述： 解码 PKCS7 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[out]	原始数据长度
pucSignerCertificate[out]	签名者证书
puiSignerCertificateLen[out]	签名者证书长度
puiDigestAlgorithms[out]	HASH 算法

返回值：

0	成功
非 0	失败，返回错误代码

7.4.4 编码 PKCS7 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 PKCS7 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
uiDigestAlgorithmsin]	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[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 PKCS7 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 PKCS7 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[out]	签名者证书长度
puiDigestAlgorithms[out]	HASH 算法
pucData[out]	被签名的数据
puiDataLen[out]	被签名的数据长度
pucSign[out]	签名值
puiSignLen[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 PKCS7 格式的数字信封数据

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 PKCS7 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]	签名后的数字信封数据
	puiDerP7EnvelopedDataLen[out]	签名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 PKCS7 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 PKCS7 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]	签名后的数字信封数据
	uiDerP7EnvelopedDataLen[in]	签名后的数字信封数据长度
	pucData[out]	需要签名的数据
	puiDataLen [out]	需要签名的数据长度
返回值：	0	成功
	非 0	失败，返回错误代码

7.4.8 编码 PKCS7 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[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 PKCS7 格式的摘要数据

原型：
```
int SAF_Pkcs7_DecodeDigestedData(
    void *hAppHandle,
    unsigned int uiDigestAlgorithm,
    unsigned char *pucData,
    unsigned int puiDataLen,
    unsigned char *pucDerP7DigestedData,
    unsigned int puiDerP7DigestedDataLen,
    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
	pucData[in]	原文数据
	puiDataLen [in]	原文数据长度
	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
	pucDigest[out]	摘要值
	puiDigestLen[out]	摘要值长度
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

错误代码标识		
宏描述	预定义值	说明
#define SAR_OK	0	成功
#define SAR_UnknownErr	0X02000001	异常错误
#define SAR_NotSupportYetErr	0X02000002	不支持的服务
#define SAR_FileErr	0X02000003	文件操作错误
#define SAR_ProviderTypeErr	0X02000004	服务提供者参数类型错误
#define SAR_LoadProviderErr	0X02000005	导入服务提供者接口错误
#define SAR_LoadDevMngApiErr	0X02000006	导入设备管理接口错误
#define SAR_AlgoTypeErr	0X02000007	算法类型错误
#define SAR_NameLenErr	0X02000008	名称长度错误
#define SAR_KeyUsageErr	0X02000009	密钥用途错误
#define SAR_ModulusLenErr	0X02000010	模的长度错误
#define SAR_NotInitializeErr	0X02000011	未初始化
#define SAR_ObjErr	0X02000012	对象错误
#define SAR_MemoryErr	0X02000100	内存错误
#define SAR_TimeoutErr	0X02000101	超时
#define SAR_IndataLenErr	0X02000200	输入数据长度错误
#define SAR_IndataErr	0X02000201	输入数据错误
#define SAR_GenRandErr	0X02000300	生成随机数错误
#define SAR_HashObjErr	0X02000301	HASH 对象错
#define SAR_HashErr	0X02000302	HASH 运算错误
#define SAR_GenRsaKeyErr	0X02000303	产生 RSA 密钥错
#define SAR_RsaModulusLenErr	0X02000304	RSA 密钥模长错误
#define SAR_CspImprtPubKeyErr	0X02000305	CSP 服务导入公钥错误
#define SAR_RsaEncErr	0X02000306	RSA 加密错误
#define SAR_RSGDecErr	0X02000307	RSA 解密错误
#define SAR_HashNotEqualErr	0X02000308	HASH 值不相等
#define SAR_KeyNotFountErr	0X02000309	密钥未发现
#define SAR_CertNotFountErr	0X02000310	证书未发现
#define SAR_NotExportErr	0X02000311	对象未导出
#define SAR_DecryptPadErr	0X02000400	解密时做补丁错误
#define SAR_MacLenErr	0X02000401	MAC 长度错误
#define SAR_KeyInfoTypeErr	0X02000402	密钥类型错误

参考文献

GB/T 17964-2000 信息技术　安全技术　加密算法　第 1 部分：概述

GB/T 17964-2000 信息技术　安全技术　加密算法　第 2 部分：非对称加密

GB/T 17964-2000 信息技术　安全技术　加密算法　第 3 部分：对称加密

GB/T 17903.1-1999 信息技术 安全技术 抗抵赖 第 1 部分:概述

GB/T 17903.2-1999 信息技术 安全技术 抗抵赖 第 2 部分:使用对称技术的机制

GB/T 17903.3-1999 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

GB/T 18238.1-2000 信息技术 安全技术 散列函数 第 1 部分:概述

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

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

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

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

GB 15851 信息技术　安全技术　带消息恢复的数字签名方案

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

RFC 2459　　X.509 互联网公开密钥基础设施证书和 CRL 轮廓

RSA Security: Public-Key Cryptography Standards (PKCS)。

Pkcs#11 Cryptographic Token Interface Standard

IETF Rfc2459，Internet X.509 Public Key Infrastructure Certificate and CRL Profile。

IETF Rfc2560，X.509 Internet Public Key Infrastructure Online Certificate Status Protocol。

IETF Rfc1777, Lightweight Directory Access Protocol。

IETF Rfc2587 Internet X.509 Public Key Infrastructure LDAPv2 Schema。

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

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）