CFCA 智能密码钥匙系列产品

应用接口（SKF）用户手册

（版本：7.5）

中国金融认证中心

2022 年 7 月 27 日

目　录

1 简介

本文档主要描述 CFCA 智能密码钥匙系列产品（Ulan、Utap 和 Uyee 等）SKF 接口。主要为上层应用集成开发提供参考。

本文表 7-1 中与 GM/T 0016-2012 描述一致的函数接口不在本文赘述。

2 遵循标准

项目产品开发所遵循的现有业界技术标准主要有：

（1）GB/T 19001-2016/ISO 9001:2015 质量管理体系；

（2）CMMI DEV v1.3 成熟度等级三级；

（3）GM/T 0027-2014 智能密码钥匙技术规范；

（4）GM/T 0016-2012 智能密码钥匙密码应用接口规范；

（5）GM/T 0017-2012 智能密码钥匙密码应用接口数据格式规范；

（6）GM/T 0028-2014 密码模块安全技术要求；

（7）CFCA 30007.01-2013 SM2 双证书申请及下载规范；

（8）CFCA 30008.01-2013 RSA 双证书申请及下载规范。

3 算法支持

（1）非对称密码算法

SM2，RSA2048 等。

（2）对称密码算法

SM1（ECB 和 CBC 模式）、SM4（ECB 和 CBC 模式）等。

（3）哈希算法

SM3、SHA-1、SHA256、SHA512 等。

4 运行环境

（1）Windows 平台

Win XP SP3 及以上版本操作系统（32 位和 64 位）。

（2）Linux 平台

支持基于 x64 处理器的各常用发行版本，如：Ubuntu、Fedora、CentOS 等；支持基于 x64（Intel/AMD/兆芯）、MIPS64（龙芯）和 aarch64（飞腾）处理器的国产操作系统，如：麒麟、统信等。

5 命名规则

（1）Utap 产品

Utap 产品 windows 平台企业标准版本 SKF 接口库文件名称为：“UtapSKF.dll”，客户定制版本默认在“UtapSKF”和“dll”之间插入客户英文简称，如:“UtapSKF.CFCA.dll”，或按照客户需求与客户协商确定。

Utap 产品 Linux 平台企业标准版本 SKF 接口库文件名称为：“libutapskf.so”，客户定制版本名称不变，默认安装到“utap-客户英文简称”命名的文件夹下，如:“utap-cfca/libutapskf.so”，并视系统版本在/usr/lib/或/usr/lib64/创建软链接，默认在“so”之前插入客户英文简称，如：“libutapskf-cfca.so”，或按照客户需求与客户协商确定。Linux 平台企业标准版本 SKF 接口库安装脚本文件名称为“skf_utap_enterprise.bin”，客户定制版本，默认在“skf_utap”之后添加客户英文简称，如“skf_utap_cfca.bin”。

（2）Uyee 产品

Uyee 产品 windows 平台企业标准版本 SKF 接口库文件名称为：“UyeeSKF.dll”，客户定制版本默认在“UyeeSKF”和“dll”之间插入客户英文简称，如:“UyeeSKF.CFCA.dll”，或按照客户需求与客户协商确定。

Uyee 产品 Linux 平台企业标准版本 SKF 接口库文件名称为：“libuyeeskf.so”，客户定制版本名称不变，默认安装到“uyee-客户英文简称”命名的文件夹下，如:“uyee-cfca/libuyeeskf.so”，并视系统版本在/usr/lib 或/usr/lib64/创建软链接，默认在“so”之前插入客户英文简

称，如：“libuyeeskf-cfca.so”，或按照客户需求与客户协商确定。Linux 平台企业标准版本 SKF 接口库安装文件名称为“skf_uyee_enterprise.bin”，客户定制版本，默认在“skf_uyee”后添加客户英文简称，如“skf_uyee_cfca.bin”。

6 安装使用

6.1 安装

（1）Windows 平台

根据实际应用需求提供两种安装方式：一、提供动态库文件，由客户直接集成；二、打包到设备服务、CSP 和 PKCS#11 安装包，运行安装文件进行安装，安装目录（以 32 位为例）位于“C:\Windows\System32”。

（2）Linux 平台

进入到安装目录，运行安装脚本，默认在”/usr/local/”目录创建产品文件夹。

6.2 卸载

（1）Windows 平台

使用安装包进行安装的情况，通过开始菜单或控制面板按照提示进行卸载。

（2）Linux 平台

删除安装文件夹内的 SKF 库文件。

6.3 使用

按照一般的 DLL 或者 SO 的加载方式进行调用。

7 接口函数列表

表 7-1 应用接口函数列表

分类	函数名称	功能描述	备注
设备管理	SKF_WaitForDevEvent	等待设备插拔事件	支持，与 GM/T 0016-2012 一致。
	SKF_CancelWaitForDevEvent	取消等待设备插拔事件	支持，与 GM/T 0016-2012 一致。
	SKF_EnumDev	枚举设备	支持，与 GM/T 0016-2012 一致。
	SKF_ConnectDev	连接设备	支持，与 GM/T 0016-2012 一致。
	SKF_DisconnectDev	断开连接	支持，与 GM/T 0016-2012 一致。
	SKF_GetDevState	获取设备状态	支持，与 GM/T 0016-2012 一致。
	SKF_SetLabel	设置设备标签	支持，与 GM/T 0016-2012 一致。
	SKF_GetDevInfo	获取设备信息	支持，与 GM/T 0016-2012 一致。
	SKF_LockDev	锁定设备	支持，与 GM/T 0016-2012 一致。
	SKF_UnlockDev	解锁设备	支持，与 GM/T 0016-2012 一致。
	SKF_Transmit	设备命令传输	应用维护接口，不对用户开放。
访问控制	SKF_ChangeDevAuthKey	修改设备认证密钥	应用维护接口，不对用户开放。
	SKF_DevAuth	设备认证	应用维护接口，不对用户开放。
	SKF_ChangePIN	修改 PIN	支持，见本文接口函数说明。
	SKF_GetPINInfo	获得 PIN 码信息	支持，与 GM/T 0016-2012 一致。
	SKF_VerifyPIN	校验 PIN	支持，见本文接口函数说明。
	SKF_UnblockPIN	解锁 PIN	标准企业版，不支持用户 PIN 解锁。
	SKF_ClearSecueState	清除应用安全状态	支持，与 GM/T 0016-2012 一致。
应用管理	SKF_CreateApplication	创建应用	应用维护接口，不对用户开放。
	SKF_EnumApplication	枚举应用	支持，与 GM/T 0016-2012 一致。

分类	函数名称	功能描述	备注
	SKF_DeleteApplication	删除应用	应用维护接口，不对用户开放。
	SKF_OpenApplication	打开应用	支持，与 GM/T 0016-2012 一致。
	SKF_CloseApplication	关闭应用	支持，与 GM/T 0016-2012 一致。
文件管理	SKF_CreateFile	创建文件	支持，仅兼容 CFCA 电子签章功能。
	SKF_DeleteFile	删除文件	支持，仅兼容 CFCA 电子签章功能。
	SKF_EnumFiles	枚举文件	支持，仅兼容 CFCA 电子签章功能。
	SKF_GetFileInfo	获取文件信息	支持，仅兼容 CFCA 电子签章功能。
	SKF_ReadFile	读文件	支持，仅兼容 CFCA 电子签章功能。
	SKF_WriteFile	写文件	支持，仅兼容 CFCA 电子签章功能。
容器管理	SKF_CreateContainer	创建容器	支持，见本文接口函数说明。
	SKF_DeleteContainer	删除容器	支持，与 GM/T 0016-2012 一致。
	SKF_EnumContainer	枚举容器	支持，与 GM/T 0016-2012 一致。
	SKF_OpenContainer	打开容器	支持，与 GM/T 0016-2012 一致。
	SKF_CloseContainer	关闭容器	支持，与 GM/T 0016-2012 一致。
	SKF_GetContainerType	获得容器类型	支持，与 GM/T 0016-2012 一致。
	SKF_ImportCertificate	导入数字证书	支持，见本文接口函数说明。
	SKF_ExportCertificate	导出数字证书	支持，见本文接口函数说明。
密码服务	SKF_GenRandom	生成随机数	支持，与 GM/T 0016-2012 一致。
	SKF_GenRSAKeyPair	生成 RSA 签名密钥对	支持，与 GM/T 0016-2012 一致。
	SKF_ImportRSAKeyPair	导入 RSA 加密密钥对	支持，见本文接口函数说明。
	SKF_RSASignData	RSA 签名	支持，见本文接口函数说明。
	SKF_RSAVerify	RSA 验签	支持，与 GM/T 0016-2012 一致。
	SKF_RSAExportSessionKey	RSA 生成并导出会话密钥	不支持，暂无应用需求。
	SKF_ExtRSAPubKeyOperation	RSA 外来公钥运算	支持，非 GM/T 0016 定义接口
	SKF_GenECCKeyPair	生成 ECC 签名密钥对	支持，与 GM/T 0016-2012 一致。
	SKF_ImportECCKeyPair	导入 ECC 加密密钥对	支持，见本文接口函数说明。

分类	函数名称	功能描述	备注
	SKF_ECCSignData	ECC 签名	支持，见本文接口函数说明。
	SKF_ECCVerify	ECC 验签	支持，与 GM/T 0016-2012 一致。
	SKF_ECCExportSessionKey	ECC 生成并导出会话密钥	支持，与 GM/T 0016-2012 一致。
	SKF_ExtECCEncrypt	ECC 外来公钥加密	支持，与 GM/T 0016-2012 一致。
	SKF_ExtECCDecrypt	ECC 外来私钥解密	支持，非 GM/T 0016 定义接口。
	SKF_ExtECCSign	ECC 外来私钥签名	支持，非 GM/T 0016 定义接口。
	SKF_ExtECCVerify	ECC 外来公钥验签	支持，非 GM/T 0016 定义接口。
	SKF_GenerateAgreementDataWithECC	ECC 生成密钥协商参数并输出	支持，与 GM/T 0016-2012 一致。
	SKF_GenerateKeyWithECC	ECC 计算会话密钥	支持，与 GM/T 0016-2012 一致。
	SKF_GenerateAgreementDataAndKeyWithECC	ECC 产生协商数据并计算会话密钥	支持，与 GM/T 0016-2012 一致。
	SKF_ExportPublicKey	导出公钥	支持，与 GM/T 0016-2012 一致。
	SKF_ImportSessionKey	导入会话密钥	支持，见本文接口函数说明。
	SKF_SetSymmKey	明文导入会话密钥	支持，非 GM/T 0016 定义接口。
	SKF_EncryptInit	加密初始化	支持，与 GM/T 0016-2012 一致。
	SKF_Encrypt	单组数据加密	支持，与 GM/T 0016-2012 一致。
	SKF_EncryptUpdate	多组数据加密	支持，与 GM/T 0016-2012 一致。
	SKF_EncryptFinal	结束加密	支持，与 GM/T 0016-2012 一致。
	SKF_DecryptInit	解密初始化	支持，与 GM/T 0016-2012 一致。
	SKF_Decrypt	单组数据解密	支持，与 GM/T 0016-2012 一致。
	SKF_DecryptUpdate	多组数据解密	支持，与 GM/T 0016-2012 一致。
	SKF_DecryptFinal	结束解密	支持，与 GM/T 0016-2012 一致。
	SKF_DigestInit	密码杂凑初始化	支持，与 GM/T 0016-2012 一致。
	SKF_Digest	单组数据密码杂凑	支持，与 GM/T 0016-2012 一致。
	SKF_DigestUpdate	多组数据密码杂凑	支持，与 GM/T 0016-2012 一致。
	SKF_DigestFinal	结束密码杂凑	支持，与 GM/T 0016-2012 一致。
	SKF_MacInit	消息鉴别码运算初始化	支持，与 GM/T 0016-2012 一致。

分类	函数名称	功能描述	备注
	SKF_Mac	单组数据消息鉴别码运算	支持，与 GM/T 0016-2012 一致。
	SKF_MacUpdate	多组数据消息鉴别码运算	支持，与 GM/T 0016-2012 一致。
	SKF_MacFinal	结束消息鉴别码运算	支持，与 GM/T 0016-2012 一致。
	SKF_CloseHandle	关闭密码对象句柄	支持，与 GM/T 0016-2012 一致。

8 数据类型和数据结构

8.1 基本数据类型

本节内容节选自 GM/T 0016-2012。本文中的字节数组均为高位字节在前（Big-Endian）方式存储和交换。基本数据类型定义如表 8-1 所示。

表 8-1 基本数据类型

类型名称	描述	定义
INT8	有符号 8 位整数	
INT16	有符号 16 位整数	
INT32	有符号 32 位整数	
UINT8	无符号 8 位整数	
UINT16	无符号 16 位整数	
UINT32	无符号 32 位整数	
BOOL	布尔类型，取值为 TRUE 或 FALSE	
BYTE	字节类型，无符号 8 位整数	typedef UINT8 BYTE
CHAR	字符类型，无符号 8 位整数	typedef UINT8 CHAR
SHORT	短整数，有符号 16 位	typedef INT16 SHORT
USHORT	无符号 16 位整数	typedef UINT16 USHORT
LONG	长整数，有符号 32 位整数	typedef INT32 LONG
ULONG	长整数，无符号 32 位整数	typedef UINT32 ULONG
UINT	无符号 32 位整数	typedef UINT32 UINT
WORD	字类型，无符号 16 位整数	typedef UINT16 WORD
DWORD	双字类型，无符号 32 位整数	typedef UINT32 DWORD
FLAGS	标志类型，无符号 32 位整数	typedef UINT32 FLAGS
LPSTR	8 位字符串指针，按照 UTF8 格式存储及交换	typedef CHAR * LPSTR
HANDLE	句柄，指向任意数据对象的起始地址	typedef void * HANDLE
DEVHANDLE	设备句柄	typedef HANDLE DEVHANDLE
HAPPLICATION	应用句柄	typedef HANDLE HAPPLICATION
HCONTAINER	容器句柄	typedef HANDLE HCONTAINER

8.2 ECC 加密密钥对保护结构

本节内容节选自 GM/T 0016-2012。

```
typedef struct SKF_ENVELOPEDKEYBLOB{
    ULONG Version;                  // 当前版本为 1
    ULONG ulSymmAlgID;              // 对称算法标识，限定 ECB 模式
    ULONG ulBits;                   // 加密密钥对的密钥位长度
    BYTE  cbEncryptedPriKey[64];    // 加密密钥对私钥的密文
    ECCPUBLICKEYBLOB PubKey;        // 加密密钥对的公钥
    ECCCIPHERBLOB ECCCipherBlob;    // 用保护公钥加密的对称密钥密文。
}ENVELOPEDKEYBLOB, *PENVELOPEDKEYBLOB;
```

私钥密文结构中 64 字节，有效密文分组从 0 偏移量字节开始，顺序解密密文分组后将明文连接得到加密密钥对的私钥的明文。

表 8-2 加密密钥对保护结构参数

数据项	类型	意义	备注
Version	ULONG	版本号，本版本为 1	
ulSymmAlgID	ULONG	对称算法标识	必须为 ECB 模式
ulBits	ULONG	加密密钥对的密钥位长	
cbEncryptedPrivKey	BYTE 数组	对称算法加密的加密私钥，加密私钥的原文为 ECCPRIVATEKEYBLOB 结构中的 PrivateKey。	其有效长度为原文的（ulBits + 7）/8
PubKey	ECCPUBLICKEYBLOB	加密密钥对的公钥	
ECCCipherBlob	ECCCIPHERBLOB	用保护公钥加密过的对称密钥密文	

8.3 CFCA ECC 加密密钥对保护结构

```
typedef struct _SM2PRIVATEKEYBLOB {
    ULONG AlgID;
    ULONG EncryptedPrivateKeyBitLen;
    BYTE *EncryptedPrivateKey;
}SM2PRIVATEKEYBLOB, *PSM2PRIVATEKEYBLOB;
```

加密私钥密文 EncryptedPrivateKey 存在两种格式：一种为 C1||C2||C3（国密老标准），另一种为 C1||C3||C2（国密新标准）。解密后的 SM2 密钥对为 x||y||d,其中 x，y 是 32 字节的公钥坐标点，d 是 32 字节的私钥。通过 CFCA 证书应用工具包导入 CFCA SM2 加密密钥对时，首先在容器中的加密密钥对位置生成临时密钥对，使用临时密钥对公钥对 SM2 加密密钥对进行加密保护。通过 SKF 接口导入 CFCA SM2 加密密钥对时，使用签名密钥对公钥对 SM2 加密密钥对进行加密保护。

表 8-3 CFCA ECC 密钥对保护结构参数

字段名称	数据长度（字节）	含义
AlgID	4	CALG_SM2_SIGN/CALG_SM2_KEYX
EncryptedPrivateKeyBitLen	4	加密 SM2 私钥 EncryptedPrivateKey 的实际位(bit)长度
EncryptedPrivateKey	192	加密的 SM2 密钥对（公私钥）数据

8.4 RSA 加密密钥对保护结构

本节内容节选自 GM/T 0016-2012。

```
typedef struct EnvelopedECCKey_st{
    ULONG ulSymmAlgID;        // 对称算法标识，限定 ECB 模式
    ULONG ulBits;             // RSA 密钥对的模长
    BYTE  cbEncryptedPriKey[11][RSAref_MAX_PLEN]; // RSA 密钥对私钥的密文
    RSArefPublicKey PubKey;   // RSA 密钥对的公钥
    ULONG L;                  // 用 RSA 保护公钥加密的对称密钥密文长度
    BYTE  C[1];               // 用 RSA 保护公钥加密的对称密钥密文
}EnvelopedRSAKey;
```

私钥密文结构中有效密文分组从 0 偏移量字节开始，顺序解密密文分组后将明文连接得到 RSA 密钥对的私钥的明文依次为 m、e、d、prime、pexp、coef。

表 8-4 RSA 密钥对保护结构参数

字段名称	数据长度（字节）	含义
ulSymmAlgID	ULONG	对称算法标识（必须为 ECB 模式）
ulBits	ULONG	RSA 密钥对的密钥位长
cbEncryptedPrivKey	RSAref_MAX_PLEN*11	对称算法加密的 RSA 私钥,RSA 私钥的原文依次为 RSArefPrivateKey 结构中的 m、e、d、prime、pexp、coef，其有效长度为原文 RSArefPrivateKey 结构中的 m、e、d、prime、pexp、coef 长度之和。
PubKey	RSAref_MAX_LEN*2+4	RSA 密钥对的公钥
L	4	用保护公钥加密过的对称密钥密文长度
C	L	用保护公钥加密过的对称密钥密文

8.5 CFCA RSA 加密密钥对保护结构

CFCA RSA 密钥对保护结构与本文 8.4 节描述一致。

通过 CFCA 证书应用工具包导入 CFCA RSA 加密密钥对时，首先在容器中的加密密钥对位置生成临时密钥对，使用临时密钥对公钥作为保护公钥，对对称密钥进行加密保护。通过 SKF 接口导入 CFCA RSA 加密密钥对时，使用签名密钥对公钥作为保护公钥，对对称密钥进行加密保护。

9 主要接口函数说明

9.1 修改 PIN

CFCA 智能密码钥匙企业版默认用户 PIN 复杂度要求为：数字、字母（区分大小写）、符号中的两种或两种以上组合。SKF_ChangePIN 函数对新 PIN 值进行复杂度检查，不符合复杂度要求时返回错误码为“0x0A000026”。

客户定制项目用户 PIN 复杂度要求以项目需求文档为准。

9.2 校验 PIN

（1）强制修改用户默认 PIN

CFCA 智能密码钥匙企业版强制要求用户使用前对默认用户 PIN 进行修改。只有修改默认 PIN 值后才能使用容器内数字证书对应的密钥对进行数字签名操作，否则返回错误码为“0x0A000029”。

客户定制项目是否强制要求修改用户默认 PIN，以项目需求文档为准。

（2）用户 PIN 锁死

CFCA 智能密码钥匙企业版验证用户 PIN 连续错误次数超过 6 次后，用户 PIN 锁死，CFCA 智能密码钥匙企业版用户 PIN 锁死后，可使用 Windows 版管理员专用工具进行初始化。用户 PIN 锁死状态，调用 SKF_VerifyPIN、SKF_ChangePIN 等函数返回错误码为“0x0A000025”。

客户定制项目验证用户 PIN 连续错误次数，以及 PIN 锁死后的处理方式（初始化、解锁等）以项目需求文档为准。

（3）验证 PIN 码输入框

CFCA 智能密码钥匙企业版 SKF 接口支持字符串和 PIN 码输入框两种方式验证用户 PIN。调用 SKF_VerifyPIN 函数，输入以下 PIN 值触发 PIN 码输入框方式验证用户 PIN。

const char szPIN [11] = {0x01, 0x08, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x00};

图 9－1 PIN 码输入框

9.3 文件管理

CFCA 智能密码钥匙企业版 SKF 接口文件管理类函数默认用于兼容 CFCA 电子签章系统 PKCS#11 图章数据对象(object class:CKO_DATA)相关功能，支持 1 个数据对象（64KB）的创建、写、读和删除等功能。

客户定制项目如有其它应用需求，以项目需求文档和接口文档为准。

9.4 创建容器

表 9-1 创建容器函数说明

原型	ULONG DEVAPI SKF_CreateContainer (HAPPLICATION hApplication, LPSTR szContainerName, HCONTAINER *phContainer)	
功能描述	在应用下建立指定名称的容器并返回容器句柄。	
参数	hApplication	[IN]应用句柄。
	szContainerName	[IN] ASCII 字符串，表示所建立容器的名称，容器名称的最大长度不能超过 40 字节。
	phContainer	[OUT] 返回所建立容器的容器句柄。
返回值	SAR_OK：成功。 其他：错误码。	
备注	**在已知下载单证书的情况下，在容器名称末尾增加后缀“#2”，可将密钥对和对应数字证书分别存储在设备容器的加密密钥对和加密证书的位置，与 windows 端 CSP 和 PKCS#11 应用保持兼容，例如：“9125758C-60F3-45B7-8552-7BD1FDFA2B8F#2”。**	

9.5 导入数字证书

表 9-2 导入数字证书函数说明

原型	ULONG DEVAPI SKF_ImportCertificate(HCONTAINER hContainer, BOOL bSignFlag, BYTE* pbCert, ULONG ulCertLen)	
功能描述	向容器内导入数字证书。	
参数	hContainer	[IN]容器句柄。
	bSignFlag	[IN] TRUE 表示签名证书，FALSE 表示加密证书。
	pbCert	[IN] 指向证书内容缓冲区。
	ulCertLen	[IN] 证书长度。
返回值	SAR_OK：成功。 其他：错误码。	
备注	（1）导入数字证书前，容器内已有对应的密钥对。 **（2）容器内为双数字证书时：bSignFlag 的值为 TRUE，导入签名数字证书；bSignFlag 的值为 FLASE，导入加密数字证书。** **（3）容器内为单数字证书时，根据容器内签名密钥对所在位置导入对应数字证书，不判断 bSignFlag 的值。**	

9.6 导出数字证书

表 9-3 导出数字证书函数说明

原型	ULONG DEVAPI SKF_ExportCertificate(HCONTAINER hContainer, BOOL bSignFlag, BYTE* pbCert, ULONG *pulCertLen)	
功能描述	从容器内导出数字证书。	
参数	hContainer	[IN]容器句柄。
	bSignFlag	[IN]TRUE 表示签名证书，FALSE 表示加密证书。
	pbCert	[OUT]指向证书内容缓冲区，如果此参数为 NULL 时，pulCertLen 表示返回数据所需要缓冲区的长度，如果此参数不为 NULL 时，返回数字证书内容。
	pulCertLen	[IN/OUT] 输入时表示 pbCert 缓冲区的长度，输出时表示证书内容的长度。
返回值	SAR_OK：成功。 其他：错误码。	
备注	**（1）容器内为双数字证书时：bSignFlag 的值为 TRUE，导出签名数字证书；bSignFlag 的值为 FLASE，导出加密数字证书。** **（2）容器内为单数字证书时，根据容器内数字证书所在位置导出，不判断 bSignFlag 的值。**	

9.7 导入 RSA 加密密钥对

表 9-4 导入 RSA 加密密钥对函数说明

原型	ULONG DEVAPI SKF_ImportRSAKeyPair (HCONTAINER hContainer, ULONG ulSymAlgId, BYTE *pbWrappedKey, ULONG ulWrappedKeyLen, BYTE *pbEncryptedData, ULONG ulEncryptedDataLen)	
功能描述	导入 RSA 加密公私钥对。	
参数	hContainer	[IN] 容器句柄。
	ulSymAlgId	[IN] 对称算法密钥标识。
	pbWrappedKey	[IN] 使用该容器内签名公钥保护的对称算法密钥。
	ulWrappedKeyLen	[IN] 保护的对称算法密钥长度。
	pbEncryptedData	[IN] 对称算法密钥保护的 RSA 加密私钥。私钥的格式遵循 PKCS #1 v2.1: RSA Cryptography Standard 中的私钥格式定义。
	ulEncryptedDataLen	[IN] 对称算法密钥保护的 RSA 加密公私钥对长度。
返回值	SAR_OK：成功。 其他：错误码。	
备注	（1）权限要求：需要用户权限。 **（2）通过 SKF 接口导入 RSA 加密密钥对时，对称算法密钥由该容器内的签名公钥保护。** **（3）CFCA 智能密码钥匙支持容器内的签名公钥或在加密密钥对位置生成的临时公钥两种方式保护对称算法密钥，主要用于兼容 Windows 平台通过证书应用工具包导入 CFCA 加密证书的场景。**	

9.8 RSA 签名

表 9-5 RSA 签名函数说明

原型	ULONG DEVAPI SKF_RSASignData(HCONTAINER hContainer, BYTE *pbData, ULONG ulDataLen, BYTE *pbSignature, ULONG *pulSignLen)	
功能描述	使用 hContainer 指定容器的签名私钥，对指定数据 pbData 进行数字签名。签名后的结果存放到 pbSignature 缓冲区，设置 pulSignLen 为签名的长度。	
参数	hContainer	[IN] 用来签名的私钥所在容器句柄。
	pbData	[IN] 被签名的数据。
	ulDataLen	[IN] 签名数据长度，应不大于 RSA 密钥模长-11。
	pbSignature	[OUT] 存放签名结果的缓冲区指针，如果值为 NULL，用于取得签名结果长度。
	pulSignLen	[IN，OUT] 输入时表示签名结果缓冲区大小，输出时表示签名结果长度。
返回值	SAR_OK：成功。 其他：错误码。	
备注	（1）权限要求：需要用户权限。 **（2）当指定的容器内为单证书时，自动选择此证书对应的私钥进行签名，接口不限制证书在容器中的存储位置。** **（3）pbData 和 ulDataLen 分别应当为填充哈希头后的数据和长度。下一版本接口将会根据数据长度对应的摘要类型，自动填充哈希头。**	

9.9 RSA 外来公钥运算

原型	ULONG DEVAPI SKF_ExtRSAPubKeyOperation (DEVHANDLE hDev, RSAPUBLICKEYBLOB* pRSAPubKeyBlob,BYTE* pbInput, ULONG ulInputLen, BYTE* pbOutput, ULONG* pulOutputLen)	
功能描述	使用外部传入的 RSA 公钥对输入数据做公钥运算并输出结果。	
参数	hDev	[IN] 设备句柄。
	pRSAPubKeyBlob	[IN] RSA 公钥数据结构。
	pbInput	[IN] 指向待运算的原始数据缓冲区。
	ulInputLen	[IN] 待运算原始数据的长度，必须为公钥模长。
	pbOutput	[OUT] 指向 RSA 公钥运算结果缓冲区，如果该参数为 NULL，则由 pulOutputLen 返回运算结果的实际长度。
	pulOutputLen	[IN,OUT] 输入时表示 pbOutput 缓冲区的长度,输出时表示 RSA 公钥运算结果的实际长度。
返回值	SAR_OK: 成功。 其他: 错误码。	

9.10 导入 ECC 加密密钥对

表 9-6 导入 ECC 加密密钥对函数说明

原型	ULONG DEVAPI SKF_ImportECCKeyPair (HCONTAINER hContainer, PENVELOPEDKEYBLOB pEnvelopedKeyBlob)	
功能描述	导入 ECC 公私钥对。	
参数	hContainer	[IN]密钥容器句柄。
	pEnvelopedKeyBlob	[IN] 受保护的加密密钥对。
返回值	SAR_OK：成功。 其他：错误码。	
备注	（1）权限要求：需要用户权限。 **（2）CFCA 智能密码钥匙支持国密标准方式（本文 8.2 节）和 CFCA 现有 CA 方式（本文 8.3 节）对加密密钥对进行保护。** **（3）当时有 CFCA 现有 CA 方式保护加密密钥对时，PENVELOPEDKEYBLOB 结构体 Version 值为 0xCFCA/0x00；ulSymmAlgID 值为 0；ulBits 值为 256；cbEncryptedPriKey 值为全零；PubKey 值为全零；ECCCipherBlob 值为 ECC 加密密钥对加密结果，ECCCipherBlob.CipherLen 的值为 96。**	

9.11 ECC 签名

表 9-7 ECC 签名函数说明

原型	ULONG DEVAPI SKF_ECCVerify (DEVHANDLE hDev, ECCPUBLICKEYBLOB* pECCPubKeyBlob, BYTE *pbData, ULONG ulDataLen, PECCSIGNATUREBLOB pSignature)	
功能描述	用 ECC 公钥对数据进行验签。	
参数	hDev	[IN] 设备句柄。
	pECCPubKeyBlob	[IN] ECC 公钥数据结构。
	pbData	[IN] 待验证签名的数据。
	ulDataLen	[IN] 数据长度。
	pSignature	[IN] 待验证签名值。
返回值	SAR_OK： 成功。 其他： 错误码。	
备注	（1）输入数据为待签数据的杂凑值。当使用 SM2 算法时，该输入数据为待签数据经过 SM2 签名预处理的结果，预处理过程遵循 GB/T AAAA。 **（2）当指定容器内为单数字证书时，接口自动选择容器内的签名密钥对进行数字签名。** **（3）容器内为单数字证书时，接口自动选择此密钥对进行数字签名，不限制密钥对所在的位置。**	

9.12 导出公钥

表 9-8 导出公钥函数说明

原型	ULONG DEVAPI SKF_ExportPublicKey (HCONTAINER hContainer, BOOL bSignFlag, BYTE* pbBlob, ULONG* pulBlobLen)	
功能描述	导出容器中的签名公钥或者加密公钥。	
参数	hContainer	[IN] 密钥容器句柄。
	bSignFlag	[IN] TRUE 表示导出签名公钥，FALSE 表示导出加密公钥。
	pbBlob	[OUT] 指向 RSA 公钥结构（RSAPUBLICKEYBLOB），或 ECC 公钥结构（ECCPUBLICKEYBLOB），或 SM9 用户标识（BYTE 数组），如果此参数为 NULL 时，由 pulBlobLen 返回 pbBlob 的长度。
	pulBlobLen	[IN，OUT] 输入时表示 pbBlob 缓冲区的长度，输出时表示导出公钥结构的大小。
返回值	SAR_OK：成功。 其他：错误码。	
备注	**（1）容器内为双数字证书时：bSignFlag 的值为 TRUE，导出签名密钥对公钥；bSignFlag 的值为 FLASE，导出加密密钥对公钥。** **（2）容器内为单数字证书时，根据容器内密钥对所在位置导出密钥对公钥，不判断 bSignFlag 的值。**	

9.13 导入会话密钥

表 9-9 导入会话密钥函数说明

原型	ULONG DEVAPI SKF_ImportSessionKey (HCONTAINER hContainer,ULONG ulAlgId,BYTE *pbWrapedData,ULONG ulWrapedLen, HANDLE *phKey)	
功能描述	导入会话密钥密文，使用容器中的加密私钥解密得到会话密钥。	
参数	hContainer	[IN] 容器句柄。
	ulAlgId	[IN] 会话密钥算法标识。
	pbWrapedData	[IN] 要导入的会话密钥密文。当容器为 ECC 类型时，此参数为 ECCCIPHERBLOB 密文数据，当容器为 RSA 类型时，此参数为 RSA 公钥加密后的数据。
	ulWrapedLen	[IN] 会话密钥密文长度。
	phKey	[OUT] 返回会话密钥句柄。
返回值	SAR_OK：成功。 其他：错误码。	
备注	（1）权限要求：需要用户权限。 **（2）导入的会话密钥在 CFCA 智能密码钥匙内支持 SM1 和 SM4 算法运算。**	

9.14 明文导入会话密钥

表 9-10 明文导入会话密钥函数说明

原型	ULONG DEVAPI SKF_SetSymmKey (DEVHANDLE hDev, BYTE* pbKey, ULONG ulAlgID, HANDLE* phKey)	
功能描述	设置明文对称密钥，返回密钥句柄。	
参数	hDev	[IN] 设备句柄。
	pbKey	[IN] 指向会话密钥值的缓冲区。
	ulAlgID	[IN] 会话密钥算法标识。
	phKey	[OUT] 返回会话密钥句柄。
返回值	SAR_OK：成功。 其他：错误码。	
备注	**（1）本函数仅用于测试和调试，不建议用于实际的密码服务。** **（2）导入的会话密钥在 CFCA 智能密码钥匙内支持 SM1 和 SM4 算法运算。**	

10 错误代码定义和说明

表 10-1 错误代码定义和说明

序号	宏描述	预定义值	说明
1	SAR_OK	0x00000000	成功
2	SAR_FAIL	0x0A000001	失败
3	SAR_UNKNOWNERR	0x0A000002	异常错误
4	SAR_NOTSUPPORTYETERR	0x0A000003	不支持的服务
5	SAR_FILEERR	0x0A000004	文件操作错误
6	SAR_INVALIDHANDLEERR	0x0A000005	无效的句柄
7	SAR_INVALIDPARAMERR	0x0A000006	无效的参数
8	SAR_READFILEERR	0x0A000007	读文件错误
9	SAR_WRITEFILEERR	0x0A000008	写文件错误
10	SAR_NAMELENERR	0x0A000009	名称长度错误
11	SAR_KEYUSAGEERR	0x0A00000A	密钥用途错误
12	SAR_MODULUSLENERR	0x0A00000B	模的长度错误
13	SAR_NOTINITIALIZEERR	0x0A00000C	未初始化
14	SAR_OBJERR	0x0A00000D	对象错误
15	SAR_MEMORYERR	0x0A00000E	内存错误
16	SAR_TIMEOUTERR	0x0A00000F	超时
17	SAR_INDATALENERR	0x0A000010	输入数据长度错误
18	SAR_INDATAERR	0x0A000011	输入数据错误
19	SAR_GENRANDERR	0x0A000012	生成随机数错误
20	SAR_HASHOBJERR	0x0A000013	HASH 对象错
21	SAR_HASHERR	0x0A000014	HASH 运算错误
22	SAR_GENRSAKEYERR	0x0A000015	产生 RSA 密钥错
23	SAR_RSAMODULUSLENERR	0x0A000016	RSA 密钥模长错误
24	SAR_CSPIMPRTPUBKEYERR	0x0A000017	CSP 服务导入公钥错误
25	SAR_RSAENCERR	0x0A000018	RSA 加密错误
26	SAR_RSADECERR	0x0A000019	RSA 解密错误
27	SAR_HASHNOTEQUALERR	0x0A00001A	HASH 值不相等
28	SAR_KEYNOTFOUNTERR	0x0A00001B	密钥未发现
29	SAR_CERTNOTFOUNTERR	0x0A00001C	证书未发现
30	SAR_NOTEXPORTERR	0x0A00001D	对象未导出
31	SAR_DECRYPTPADERR	0x0A00001E	解密时做补丁错误
32	SAR_MACLENERR	0x0A00001F	MAC 长度错误
33	SAR_BUFFER_TOO_SMALL	0x0A000020	缓冲区不足
34	SAR_KEYINFOTYPEERR	0x0A000021	密钥类型错误
35	SAR_NOT_EVENTERR	0x0A000022	无事件错误
36	SAR_DEVICE_REMOVED	0x0A000023	设备已移除

37	SAR_PIN_INCORRECT	0x0A000024	PIN 不正确
38	SAR_PIN_LOCKED	0x0A000025	PIN 被锁死
39	SAR_PIN_INVALID	0x0A000026	PIN 无效
40	SAR_PIN_LEN_RANGE	0x0A000027	PIN 长度错误
41	SAR_USER_ALREADY_LOGGED_IN	0x0A000028	用户已经登录
42	SAR_USER_PIN_NOT_INITIALIZED	0x0A000029	没有初始化用户口令
43	SAR_USER_TYPE_INVALID	0x0A00002A	PIN 类型错误
44	SAR_APPLICATION_NAME_INVALID	0x0A00002B	应用名称无效
45	SAR_APPLICATION_EXISTS	0x0A00002C	应用已经存在
46	SAR_USER_NOT_LOGGED_IN	0x0A00002D	用户没有登录
47	SAR_APPLICATION_NOT_EXISTS	0x0A00002E	应用不存在
48	SAR_FILE_ALREADY_EXIST	0x0A00002F	文件已经存在
49	SAR_NO_ROOM	0x0A000030	空间不足
50	SAR_FILE_NOT_EXIST	0x0A000031	文件不存在
51	SAR_REACH_MAX_CONTAINER_COUNT	0x0A000032	已达到最大可管理容器数
扩展错误码			
52	SAR_AUTH_BLOCKED	0x0A000033	密钥已被锁住
53	SAR_INVALIDCONTAINERERR	0x0A000035	无效容器
54	SAR_CONTAINER_NOT_EXISTS	0x0A000036	容器不存在
55	SAR_CONTAINER_EXISTS	0x0A000037	容器已存在
56	SAR_KEYNOUSAGEERR	0x0A000039	密钥未被使用
57	SAR_FILEATTRIBUTEERR	0x0A00003A	文件操作权限错误
58	SAR_DEVNOAUTH	0x0A00003B	设备未认证

11 编程参考

11.1 枚举并打开设备

```
ULONG32 nResult = SAR_UNKNOWNERR;
ULONG32 nDevSize = 0;
char *pDevList = NULL;
char *pName = NULL;
DEVHANDLE hDev = NULL;
nResult = SKF_EnumDev(TRUE, NULL, &nDevSize);
pDevList = new char[nDevSize];
memset(pDevList, 0x00, nDevSize);
nResult = SKF_EnumDev(TRUE, pDevList, &nDevSize);
pName = pDevList;
while (0 != *pName)
{
    nResult = SKF_ConnectDev(pName, &hDev);
    pName += strlen(pName) + 1;
}
delete [] pDevList;
pDevList = NULL;
```

11.2 捕获设备插入事件并打开设备

```
ULONG32 nResult = SAR_UNKNOWNERR;
ULONG32 ulNameSize = 256;
ULONG32 ulEvent = 0;
char szDevName[MAX_PATH_LENGTH] = {0};
DEVHANDLE hDev = NULL;
nResult = SKF_ConnectDev(szDevName, &hDev);
if (1 == ulEvent)
{
    nResult = SKF_ConnectDev(szDevName, &hDev);
```

```
}
```

11.3 枚举并打开应用

```
ULONG32 nResult = SAR_UNKNOWNERR;
ULONG32 nAppSize = 0;
char *pAppList = NULL;
char *pName = NULL;
HAPPLICATION hApp = NULL;
nResult = SKF_EnumApplication(hDev, NULL, &nAppSize);
pAppList = new char[nAppSize];
memset(pAppList, 0x00, nAppSize);
nResult = SKF_EnumApplication(hDev, pAppList, &nAppSize);
pName = pAppList;
while (0 != *pName)
{
    nResult = SKF_OpenApplication(hDev, pName, &hApp);
    pName += strlen(pName) + 1;
}
delete [] pAppList;
pAppList = NULL;
```

11.4 枚举并打开容器

```
ULONG32 nResult = SAR_UNKNOWNERR;
ULONG32 nContainerSize = 0;
char *pContainerList = NULL;
char *pName = NULL;
HCONTAINER hContainer = NULL;
nResult = SKF_EnumContainer (hDev, NULL, &nContainerSize);
pContainerList = new char[nContainerSize];
memset(pContainerList, 0x00, nContainerSize);
nResult = SKF_EnumContainer(hDev, pContainerList, &nContainerSize);
```

```
pName = pContainerList;
while (0 != *pName)
{
    nResult = SKF_OpenContainer(hDev, pName, &hContainer);
    pName += strlen(pName) + 1;
}
delete [] pContainerList;
pContainerList = NULL;
```

11.5 校验 PIN

```
ULONG32 nResult = SAR_UNKNOWNERR;
CHAR pin[] = {'1', '2', '3', '4', '5', '6'};//应用集成开发时，PIN由用户输入
ULONG32 count = 0;
hResult = SKF_VerifyPIN(pApp, USER_TYPE, pin, &count);
```

11.6 SM2 签名

```
ULONG32 nResult = SAR_UNKNOWNERR;
ECCPUBLICKEYBLOB stPublicKey = {0};
UINT32 nPublicKeySize = sizeof(ECCPUBLICKEYBLOB);
BYTE szDigestBuf[512] = {0};
UINT32 nDigestSize = 0, nFinalSize = 0;
ECCSIGNATUREBLOB stEccSign = {0};
nResult = SKF_ExportPublicKey(pCtnr, TRUE, (BYTE*)&stPublicKey, &nPublicKeySize);
SM2_GetZVal(&stPublicKey, szDigestBuf);
nDigestSize = 32;
//应用集成开发时，原文为实际数据
Hash_Digest(SGD_SM3, “12345678”, 8, szDigestBuf + nDigestSize, &nFinalSize);
nDigestSize += nFinalSize;
nResult = SKF_ECCSignData(pContainer, szDigestBuf, nDigestSize, &stEccSign);
```

11.7 导入 SM1/SM4 会话密钥并加密

```
ULONG32 nResult = SAR_UNKNOWNERR;
ECCPUBLICKEYBLOB stPublicKey = {0};
UINT32 nPublicKeySize = sizeof(ECCPUBLICKEYBLOB);
BYTE szEncSymmKey[512] = {0};
UINT32 nEncSymmKeySize = 0;
BYTE szCipherData[256] = {0};
```

```
    UINT32 nCipherSize = 0, nFinalSize = 0;
    PECCCIPHERBLOB pECCCipherBlob = NULL;
    HANDLE pSessionHandle = NULL;
    //应用集成开发时，密钥不可设置为确定值，每次都要随机产生
    BYTE szSessionKey[] =
{0x40,0xbb,0x12,0xdd,0x6a,0x82,0x73,0x86,0x7f,0x35,0x29,0xd3,0x54,0xb4,0xa0,0x26};
    BLOCKCIPHERPARAM ucBlockCiphier =
{{0xe8,0x3d,0x17,0x15,0xac,0xf3,0x48,0x63,0xac,0xeb,0x93,0xe0,0xe5,0xab,0x8b,0x90},16,
PKCS5_PADDING,0};
    nResult = SKF_ExportPublicKey(pCtnr, FALSE, (BYTE*)&stPublicKey, &nPublicKeySize);
    SM2_Encrypt(&stPublicKey, szSessionKey, 16, szEncSymmKey, &nEncSymmKeySize);
    pECCCipherBlob = szEncSymmKey;
    nResult = SKF_ImportSessionKey(pContainer, SGD_SMS4_CBC, (BYTE*)& pECCCipherBlob,
nEncSymmKeySize, &pSessionHandle);
    nResult = SKF_EncryptInit(pSessionHandle, ucBlockCiphier);
    nResult = SKF_EncryptUpdate(pSessionHandle, (BYTE*)”12345678”, 8,
szCipherData,(ULONG32*)&nCipherSize);
    nResult = SKF_EncryptFinal(pSessionHandle, szCipherData + nCipherSize,
(ULONG32*)&nFinalSize);
    nCipherSize += nFinalSize;
```