ICS 35.030
CCS L 80

GM

中华人民共和国密码行业标准

GM/T 0118—2022

浏览器数字证书应用接口规范

Browser digital certificate application interface specification

2022-11-20 发布　　2023-06-01 实施

国家密码管理局　　发布

目　　次

前言 …… Ⅲ
1　范围 …… 1
2　规范性引用文件 …… 1
3　术语和定义 …… 1
4　缩略语 …… 2
5　总体技术框架 …… 2
6　算法标识与数据类型 …… 3
6.1　算法标识 …… 3
6.2　基本数据类型 …… 3
6.3　常量定义 …… 3
6.4　复合数据类型 …… 4
7　接口函数 …… 12
7.1　概述 …… 12
7.2　证书存储区管理接口 …… 13
7.3　UI 接口 …… 19
7.4　SKF 管理接口 …… 20
7.5　与其他接口规范的关系 …… 20
附录 A（规范性）　错误码定义和说明 …… 22
附录 B（资料性）　使用本规范接口的例程 …… 23
B.1　注册 SKF 以及证书存储 …… 23
B.2　SKF 函数指针 …… 25
B.3　加载释放 SKF 动态库 …… 25
B.4　证书使用 …… 26
参考文献 …… 28

前　言

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

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专利的责任。

本文件由密码行业标准化技术委员会提出并归口。

本文件起草单位：天津赢达信科技有限公司、北京信安世纪科技有限公司、北京数字认证股份有限公司、北京海泰方圆科技股份有限公司、中国民生银行股份有限公司、北京奇虎科技有限公司、亚数信息科技(上海)有限公司。

本文件主要起草人：张秋璞、曹伟、彭竹、李强强、张永强、张庆勇、蒋红宇、虞刚、刘书洪、袁丽欧、霍海涛、张志磊、翟新元。

浏览器数字证书应用接口规范

1 范围

本文件规定了浏览器SM2数字证书应用接口，描述了在支持国产密码算法应用的浏览器中，数字证书应用接口的函数、数据类型和参数的定义。

本文件适用于浏览器产品的开发、应用和检测，支持SM2数字证书的浏览器应用的开发，安全浏览器密码模块的检测，也可用于指导第三方应用调用不同终端设备中SM2数字证书的集成和开发。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；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。

GB/T 20518 信息安全技术 公钥基础设施 数字证书格式

GB/T 32918.2 信息安全技术 SM2椭圆曲线公钥密码算法 第2部分：数字签名算法

GB/T 33560 信息安全技术 密码应用标识规范

GB/T 35275 信息安全技术 SM2密码算法加密签名消息语法规范

GM/T 0016 智能密码钥匙密码应用接口规范

GM/T 0100 人工确权型数字签名密码应用技术要求

GM/Z 4001 密码术语

3 术语和定义

GM/Z 4001界定的以及下列术语和定义适用于本文件。

3.1

SM2密码算法 SM2 cryptographic algorithm

由GB/T 32918.5定义的公钥密码算法。

3.2

数字证书 digital certificate

也称公钥证书，由证书认证机构(CA)签名的包含公开密钥拥有者信息、公开密钥、签发者信息、有效期以及扩展信息的一种数据结构。按类型可分为个人证书、机构证书和设备证书，按用途可分为签名证书和加密证书。

3.3

数字签名 digital signature

签名者使用私钥对待签名数据做密码运算得到的结果，该结果只能用签名者的公钥进行验证，用于确认待签名数据的完整性、签名者身份的真实性和签名行为的抗抵赖性。

3.4

证书上下文 certificate context

一种数据结构，用于保存相关的证书信息，证书信息包括拥有者信息、公开密钥、签发者信息、有效期以及扩展信息等。

3.5

证书存储区　certificate storage

一种逻辑概念，用于集中保存数字证书，包括用户证书、根证书等不同类型的证书，具体的证书存储区的位置不做定义，由实现本接口的厂商自行实现。

3.6

证书存储上下文　certificate store context

一种数据结构，用于保存证书上下文和证书对应的 SKF 信息，SKF 信息包括 SKF 接口名称、设备名称、应用名称、容器名称。

注：SKF 为 GM/T 0016 定义的接口规范。

4　缩略语

下列缩略语适用于本文件。

AIA：授权信息访问（Authority Information Access）

API：应用编程接口（Application Programming Interface）

CRL：证书撤销列表（Certificate Revocation List）

DER：可辨别编码规则（Distinguished Encoding Rules）

LDAP：轻量目录访问协议（Lightweight Directory Access Protocol）

OCSP：在线证书状态协议（Online Certificate Status Protocol）

PEM：隐私增强文本（Privacy Enhancement Message）

PIN：个人身份识别码（Personal Identification Number）

5　总体技术框架

浏览器数字证书应用接口的总体技术框架如图 1 所示。

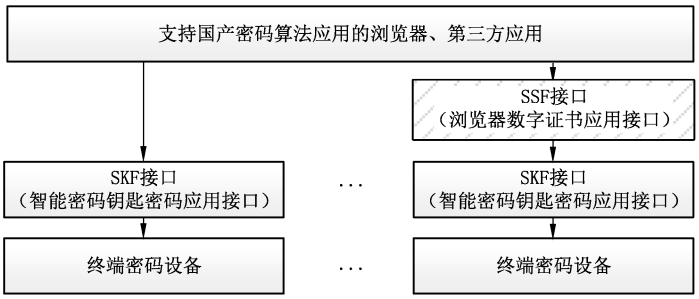

图 1　浏览器数字证书应用接口的总体技术框架

本文件规范的接口位于支持 SM 系列算法应用的浏览器、第三方应用和 SKF 接口之间，下层通过 SKF 接口访问终端密码设备；上层为 SM 系列算法浏览器、第三方应用提供接口，实现浏览器数字证书应用。其核心是定义终端操作系统上的 SM2 证书存储与调用的管理机制，如 SM2 数字证书在终端操作系统与浏览器中的存储、枚举、验证、与终端密码硬件的绑定关系等应用需求，使浏览器与第三方应用能快捷地在终端操作系统上找到 SM2 数字证书，并获取与 SM2 数字证书相关的 SKF 接口信息。本文件规范的接口以应用接口库方式和浏览器对接，应内置或者支持导入 SM2 根证书，并且为提升自身的工作效率，其实现与调用方式推荐采用 C 语言实现。

6 算法标识与数据类型

6.1 算法标识

本文件中使用的算法其标识定义按照 GB/T 33560 执行。

6.2 基本数据类型

本文件中的数据类型、字节数组均以高位字节在前(Big-Endian)方式存储和交换。基本数据类型定义如表 1 所示。

表 1 基本数据类型

类型名称	描述
INT8	有符号 8 位整数
INT16	有符号 16 位整数
INT32	有符号 32 位整数
INT64	有符号 64 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
UINT64	无符号 64 位整数
BOOL	布尔类型,取值为 TRUE 或 FALSE

6.3 常量定义

数据常量标识定义了在标准中用到的常量的取值。

数据常量标识的定义如表 2 所示。

表 2 常量定义

常量名	取值	描述
TRUE	0x00000001	布尔值为真
FALSE	0x00000000	布尔值为假
SSF_CERT_ALG_FLAG_RSA	0x00000001	RSA 证书
SSF_CERT_ALG_FLAG_SM2	0x00000002	SM2 证书
SSF_CERT_ALG_FLAG_OTHER	0x00000004	其他证书
SSF_CERT_TYPE_SIGN	0x00000001	签名证书
SSF_CERT_TYPE_CRYPT	0x00000002	加密证书
SSF_CERT_VERIFY_FLAG_TIME	0x00000001	验证时间有效期
SSF_CERT_VERIFY_FLAG_CHAIN	0x00000002	验证证书链以及签名
SSF_CERT_VERIFY_FLAG_CRL	0x00000004	验证吊销列表
SSFAPI	Windows 系统:_stdcall 其他系统:空值	Windows 系统:_stdcall 调用方式 其他系统:无

表 2　常量定义（续）

常量名	取值	描述
SSF_OCSP_STATE_GOOD	0x00000000	证书查询 OCSP 状态正确
SSF_OCSP_STATE_REVOKED	0x00000001	证书被吊销
SSF_OCSP_STATE_UNKNOWN	0x00000002	证书未知
SSF_CERT_STATE_GOOD	0x00000000	证书正确
SSF_CERT_STATE_OUT_TIME	0x00000001	证书不在有效期
SSF_CERT_STATE_NO_CHIAN	0x00000002	证书未发现证书链
SSF_CERT_STATE_IN_CRL	0x00000003	证书被吊销
SSF_CERT_STATE_INVALID_SIGN	0x00000004	非法签名证书
SSF_CERT_STATE_FAIL	0x00000005	证书其他错误
SSF_CERT_CA_ROOT_STORE	0x00000001	证书存储区为根或中间 CA 存储区
SSF_CERT_USER_STORE	0x00000002	证书存储区为用户存储区
SSF_CERT_FIND_FLAG_ALG_TYPE	0x00000001	证书查找标识算法类型
SSF_CERT_FIND_FLAG_USAGE_TYPE	0x00000002	证书查找标识用途类型
SSF_CERT_FIND_FLAG_STORE_TYPE	0x00000004	证书查找标识存储类型
SSF_CERT_FIND_FLAG_SUBJECT	0x00000008	证书查找标识主题项
SSF_CERT_FIND_FLAG_ISSUER	0x00000010	证书查找标识颁发者
SSF_CERT_FIND_FLAG_PUBKEY	0x00000020	证书查找标识主题公钥
SSF_CERT_FIND_FLAG_SN	0x00000040	证书查找标识序列号
SSF_CERT_FIND_FLAG_SUBJECT_KEYID	0x00000080	证书查找标识使用者密钥 ID
SSF_CERT_FIND_FLAG_ISSUER_KEYID	0x00000100	证书查找标识颁发者密钥 ID
SSF_CERT_FIND_FLAG_VENDOR	0x00000200	证书查找标识用户自定义数据
SSF_CRL_FIND_FLAG_ISSUER	0x00000001	吊销列表查找标识颁发者
SSF_CRL_FIND_FLAG_EFFECT_TIME	0x00000002	吊销列表查找标识生效时间

6.4　复合数据类型

6.4.1　基本数据

6.4.1.1　类型定义

```
typedef struct _SSF_Data
{
    UINT32   uiLength;
    UINT8    * data;
}SSF_Data;
```

6.4.1.2　数据项描述

数据项描述如表 3 所示。

表 3　数据项描述

数据项	类型	意义	备注
uiLength	UINT32	长度	数据长度
data	UINT8 *	值	数据值

6.4.2　证书属性

6.4.2.1　类型定义

```
typedef struct _SSF_CertificateAttr
{
    SSF_Data stCommonName;
    SSF_Data stSubject;
    SSF_Data stIssuer;
    SSF_Data stPublicKey;
    SSF_Data stSerialNumber;
    SSF_Data stSubjectKeyID;
    SSF_Data stIssuerKeyID;
    SSF_Data stSubjectAltName;
    SSF_Data stVendorData;
    SSF_Data stExtKeyUsage;
    UINT32 uiCertAlgType;
    UINT32 uiCertUsageType;
    UINT32 uiVerify;
    UINT64 ulNotBefore;
    UINT64 ulNotAfter;
}SSF_CertificateAttr;
```

6.4.2.2　数据项描述

数据项描述如表 4 所示。

表 4　数据项描述

数据项	类型	意义	备注
stCommonName	SSF_Data	通用名	应符合 GB/T 20518
stSubject	SSF_Data	主题项	应符合 GB/T 20518
stIssuer	SSF_Data	颁发者	应符合 GB/T 20518
stPublicKey	SSF_Data	公钥	应符合 GB/T 20518
stSerialNumber	SSF_Data	序列号	应符合 GB/T 20518
stSubjectKeyID	SSF_Data	使用者密钥标识	应符合 GB/T 33560
stIssuerKeyID	SSF_Data	颁发者密钥标识	应符合 GB/T 33560
stSubjectAltName	SSF_Data	颁发者的其他名称	应符合 GB/T 20518

表 4　数据项描述（续）

数据项	类型	意义	备注
stVendorData	SSF_Data	用户自定义数据	应符合 GB/T 20518
stExtKeyUsage	SSF_Data	增强(扩展)私钥用法	应符合 GB/T 20518
uiCertAlgType	UINT32	证书算法类型	应符合 GB/T 20518
uiCertUsageType	UINT32	证书用途	应符合 GB/T 20518
uiVerify	UINT32	验证结果	应符合 GB/T 20518
ulNotBefore	UINT64	起始时间	应符合 GB/T 20518
ulNotAfter	UINT64	截止时间	应符合 GB/T 20518

6.4.3　证书查找属性

6.4.3.1　类型定义

```
typedef struct _SSF_CertificateFindAttr
{
    UINT32 uiFindFlag;
    UINT32 uiCertAlgType;
    UINT32 uiCertUsageType;
    UINT32 uiStoreType;
    SSF_Data stSubject;
    SSF_Data stIssuer;
    SSF_Data stPublicKey;
    SSF_Data stSerialNumber;
    SSF_Data stSubjectKeyID;
    SSF_Data stIssuerKeyID;
    SSF_Data stVendorData;
}SSF_CertificateFindAttr;
```

6.4.3.2　数据项描述

数据项描述如表 5 所示。

表 5　数据项描述

数据项	类型	意义	备注
uiFindFlag	UINT32	查找标识	取值为以下选项按位或，当多个标识位被设置时，与标识位对应的所有查找属性应同时满足： SSF_CERT_FIND_FLAG_ALG_TYPE； SSF_CERT_FIND_FLAG_USAGE_TYPE； SSF_CERT_FIND_FLAG_STORE_TYPE； SSF_CERT_FIND_FLAG_SUBJECT； SSF_CERT_FIND_FLAG_ISSUER； SSF_CERT_FIND_FLAG_PUBKEY； SSF_CERT_FIND_FLAG_SN； SSF_CERT_FIND_FLAG_SUBJECT_KEYID； SSF_CERT_FIND_FLAG_ISSUER_KEYID； SSF_CERT_FIND_FLAG_VENDOR；

表 5　数据项描述（续）

数据项	类型	意义	备注
uiCertAlgType	UINT32	证书算法类型	uiCertAlgType 取值：SSF_CERT_ALG_FLAG_RSA、SSF_CERT_ALG_FLAG_SM2、SSF_CERT_ALG_FLAG_OTHER； 在 uiFindFlag 设置了 SSF_CERT_FIND_FLAG_ALG_TYPE 时，本查找项有效
uiCertUsageType	UINT32	证书用途	uiCertUsageType 取值：SSF_CERT_TYPE_SIGN、SSF_CERT_TYPE_CRYPT； 在 uiFindFlag 设置了 SSF_CERT_FIND_FLAG_USAGE_TYPE 时，本查找项有效
uiStoreType	UINT32	存储类型	uiStoreType 取值 SSF_CERT_CA_ROOT_STORE、SSF_CERT_USER_STORE； 在 uiFindFlag 设置了 SSF_CERT_FIND_FLAG_STORE_TYPE 时，本查找项有效
stSubject	SSF_Data	主题项	在 uiFindFlag 设置了 SSF_CERT_FIND_FLAG_SUBJECT 时，本查找项有效
stIssuer	SSF_Data	颁发者	在 uiFindFlag 设置了 SSF_CERT_FIND_FLAG_ISSUER 时，本查找项有效
stPublicKey	SSF_Data	公钥	在 uiFindFlag 设置了 SSF_CERT_FIND_FLAG_PUBKEY 时，本查找项有效
stSerialNumber	SSF_Data	序列号	在 uiFindFlag 设置了 SSF_CERT_FIND_FLAG_SN 时，本查找项有效
stSubjectKeyID	SSF_Data	使用者密钥标识	在 uiFindFlag 设置了 SSF_CERT_FIND_FLAG_SUBJECT_KEYID 时，本查找项有效
stIssuerKeyID	SSF_Data	颁发者密钥标识	在 uiFindFlag 设置了 SSF_CERT_FIND_FLAG_ISSUER_KEYID 时，本查找项有效
stVendorData	SSF_Data	用户自定义数据	在 uiFindFlag 设置了 SSF_CERT_FIND_FLAG_VENDOR 时，本查找项有效

6.4.4　证书内容

6.4.4.1　类型定义

```
typedef struct _SSF_CertificateData
{
    UINT32 uiLength;
    UINT8 * data;
}SSF_CertificateData;
```

6.4.4.2　数据项描述

数据项描述如表 6 所示。

表 6　数据项描述

数据项	类型	意义	备注
uiLength	UINT32	长度	data 的长度
data	UINT8 *	数据	数据值

6.4.5 证书上下文

6.4.5.1 类型定义

```
typedef struct _SSF_CertificateContext
{
    SSF_CertificateAttr stAttr;
    SSF_CertificateData stContent;
}SSF_CertificateContext;
```

6.4.5.2 数据项描述

数据项描述如表 7 所示。

表 7 数据项描述

数据项	类型	意义
stAttr	SSF_CertificateAttr	属性
stContent	SSF_CertificateData	内容

6.4.6 SKF 结构

6.4.6.1 类型定义

```
typedef struct _SSF_SKF
{
    SSF_Data    stName;
    SSF_Data    stPath;
    UINT32      uiSignType;
    UINT32      uiPinMode;
}SSF_SKF;
```

6.4.6.2 数据项描述

数据项描述如表 8 所示。

表 8 数据项描述

数据项	类型	意义	备注
stName	SSF_Data	SKF 接口库名称	
stPath	SSF_Data	SKF 接口库路径	根据接口是 32 位或 64 位，配置对应路径
uiSignType	UINT32	签名类型： 0：对待签名数据的杂凑值计算签名，数字签名接口符合 GM/T 0016； 1：对明文计算签名，数字签名接口符合 GM/T 0100，此时终端密码设备为人工确权型数字签名设备，例如可视按键型智能密码钥匙； 2：支持 SKF 扩展定义的数字签名接口，例如，由数字签名接口根据参数输入值的格式自动判断输入的参数是待签名数据还是待签名数据的杂凑值	在计算待签名数据的杂凑值时，待签名数据应执行 SM2 签名预处理，预处理过程应符合 GB/T 32918.2

表 8　数据项描述（续）

数据项	类型	意义	备注
uiPinMode	UINT32	PIN 码校验模式： 0：由上层应用弹出 PIN 框并实现校验用户 PIN 的功能； 1：上层应用不调用 SKF 校验 PIN 码函数，由终端密码设备厂商在需要时弹出校验 PIN 码框或通过自定义的安全通道进行用户鉴别功能，例如硬件自带键盘、自带指纹传感器等鉴别方式； 2：支持 SKF 扩展定义的鉴别方式或鉴别接口，由终端密码设备厂商通过自定义的鉴别方式或鉴别接口进行用户鉴别功能，例如硬件自带键盘、自带指纹传感器等鉴别方式	

6.4.7　SKF 链表

6.4.7.1　类型定义

```
typedef struct _SSF_SKF_NODE
{
    SSF_SKF  * ptr_data;
    struct _SSF_SKF_NODE  * ptr_next;
}SSF_SKF_NODE;
```

6.4.7.2　数据项描述

数据项描述如表 9 所示。

表 9　数据项描述

数据项	类型	意义
ptr_data	SSF_SKF	SKF 值
ptr_next	struct _SSF_SKF_NODE *	下一指针，当该指针的值为 NULL 时，表示 SKF 链表结束

6.4.8　SKF 证书对应信息

6.4.8.1　类型定义

```
typedef struct _SSF_SKFCertRef
{
    SSF_Data   stSKFName;
    SSF_Data   stDeviceName;
    SSF_Data   stApplicationName;
    SSF_Data   stContainerName;
} SSF_SKFCertRef;
```

6.4.8.2　数据项描述

数据项描述如表 10 所示。

表 10　数据项描述

数据项	类型	意义
stSKFName	SSF_Data	SKF 接口名称
stDeviceName	SSF_Data	设备名称
stApplicationName	SSF_Data	应用名称
stContainerName	SSF_Data	容器名称

6.4.9　证书存储上下文

6.4.9.1　类型定义

```
typedef struct _SSF_StoreContext
{
    SSF_CertificateContext stCertContext;
    SSF_SKFCertRef stSKFCertRef;
    UINT32 uiStoreType;
} SSF_StoreContext;
```

6.4.9.2　数据项描述

数据项描述如表 11 所示。

表 11　数据项描述

数据项	类型	意义	备注
stCertContext	SSF_CertificateContext	证书上下文	
stSKFCertRef	SSF_SKFCertRef	SKF 证书对应信息	
uiStoreType	UINT32	证书存储区存储类型	SSF_CERT_CA_ROOT_STORE： 证书存储区为根或中间 CA 存储区； SSF_CERT_USER_STORE： 证书存储区为用户存储区

6.4.10　证书存储上下文链表

6.4.10.1　类型定义

```
typedef struct _SSF_StoreContext_NODE
{
    SSF_StoreContext  * ptr_data;
    struct _SSF_StoreContext_NODE  * ptr_next;
}SSF_StoreContext_NODE;
```

6.4.10.2　数据项描述

数据项描述如表 12 所示。

表 12　数据项描述

数据项	类型	意义
ptr_data	SSF_StoreContext	证书存储上下文
ptr_next	struct _SSF_StoreContext_NODE *	下一证书存储上下文的指针，当该指针的值为NULL时，表示证书存储上下文链表结束

6.4.11　CRL 结构

6.4.11.1　类型定义

```
typedef struct _SSF_CRL
{
    UINT32 uiLength;
    UINT8 * data;
}SSF_CRL;
```

6.4.11.2　数据项描述

数据项描述如表 13 所示。

表 13　数据项描述

数据项	类型	意义	备注
uiLength	UINT32	长度	
data	UINT8 *	值	

6.4.12　CRL 链表

6.4.12.1　类型定义

```
typedef struct _SSF_CRL_NODE
{
    SSF_CRL * ptr_data;
    struct _SSF_CRL_NODE * ptr_next;
}SSF_CRL_NODE;
```

6.4.12.2　数据项描述

数据项描述如表 14 所示。

表 14　数据项描述

数据项	类型	意义	备注
ptr_data	SSF_CRL	CRL 值	
ptr_next	struct _SSF_CRL_NODE *	下一指针，当该指针的值为 NULL 时，表示 CRL 链表结束	

6.4.13 CRL 查找属性

6.4.13.1 类型定义

```
typedef struct _SSF_CRLFindAttr
{
    UINT32 uiFindFlag;
    SSF_Data stIssuer;
    SSF_Data stEffectTime;
}SSF_CRLFindAttr;
```

6.4.13.2 数据项描述

数据项描述如表 15 所示。

表 15 数据项描述

数据项	类型	意义	备注
uiFindFlag	UINT32	查找标识	取值为以下选项按位或，当多个标识位被设置时，与标识位对应的所有查找属性应同时满足： SSF_CRL_FIND_FLAG_ISSUER； SSF_CRL_FIND_FLAG_EFFECT_TIME
stIssuer	SSF_Data	颁发者	stIssuer 取值：具体格式见 GB/T 20518； 在 uiFindFlag 设置了 SSF_CRL_FIND_FLAG_ISSUER 时，本查找项有效
stEffectTime	SSF_Data	生效时间	stEffectTime 取值：精确到天，具体格式见 GB/T 20518； 在 uiFindFlag 设置了 SSF_CRL_FIND_FLAG_EFFECT_TIME 时，本查找项有效

7 接口函数

7.1 概述

本文件中使用的算法其标识定义应符合 GB/T 33560，错误码应符合附录 A 的规定。

浏览器数字证书应用接口标准主要由以下几部分组成。

a) 证书存储区管理接口

证书存储区管理接口实现 SM2 证书及存储区的管理，包括证书的存储、枚举、查找、删除等功能，CRL 的存储、枚举、查找等功能以及通过 OCSP 和 LDAP 获取证书状态等功能。其中 SM2 数字证书应符合 GB/T 20518。

b) UI 接口

UI 接口包括显示证书和选择证书接口。

显示证书接口实现显示 SM2 证书详细信息。

选择证书接口实现显示多张证书选择框，让用户选择相应的证书。

c) SKF 管理接口

SKF 管理接口实现 SKF 在浏览器数字证书应用接口中的注册、注销与枚举。

证书存储区管理接口、UI 接口、SKF 管理接口三类接口函数定义简述如表 16 所示。

表 16　接口函数定义简述表

序号	分类	接口名称	简述
1	证书存储区管理接口	SSF_CreateCertCtx	创建证书上下文
2		SSF_FreeCertCtx	释放证书上下文
3		SSF_AddCert	添加证书到证书存储区
4		SSF_DelCert	从证书存储区中删除证书
5		SSF_ClrAllCert	清除证书存储区中指定类型的所有证书
6		SSF_FindStoreCert	在证书存储区中查找证书
7		SSF_EnumCert	在证书存储区中遍历证书
8		SSF_FreeStoreCertCtx	释放证书存储上下文
9		SSF_FreeStoreCertCtxLink	释放证书存储上下文链表
10		SSF_CreateCRL	创建 CRL 对象
11		SSF_FreeCRL	释放 CRL 对象
12		SSF_AddCRL	添加 CRL 到证书存储区
13		SSF_DelCRL	从证书存储区中删除 CRL
14		SSF_FindCRL	在证书存储区中查找 CRL
15		SSF_EnumCRL	在证书存储区中遍历 CRL
16		SSF_FreeCRLLink	释放 CRL 链表
17		SSF_GetCertStateByOCSP	通过 OCSP 获取证书状态
18		SSF_GetCertStateByOCSPResponse	通过 OCSP 响应获取证书状态
19		SSF_GetCertStateByCRL	通过 CRL 获取证书状态
20		SSF_GetCertByLDAP	通过 LDAP 获取证书
21		SSF_GetCRLByLDAP	通过 LDAP 获取 CRL
22		SSF_GetCertByHTTP	通过 HTTP 方式获取证书
23		SSF_UpdateChainByAIA	通过证书 AIA 更新证书链
24		SSF_VerifyCert	验证证书有效性
25	UI 接口	SSF_ViewCert	显示证书
26		SSF_SelectCert	选择证书
27	SKF 管理接口	SSF_RegisterSKF	注册 SKF 到本规范定义的系统环境中
28		SSF_UnregisterSKF	从本规范定义的系统环境中注销 SKF
29		SSF_EnumSKF	枚举本规范定义的系统环境支持的 SKF
30		SSF_FreeSKFLink	释放 SKF 链表

7.2　证书存储区管理接口

7.2.1　创建证书上下文

函数原型 UINT32 SSFAPI SSF_CreateCertCtx(const UINT8 * pCertificate,
UINT32 uiCertificateLen,SSF_CertificateContext * * ppCertCtx);

功能描述 创建证书上下文。

参数　pCertificate　[IN]证书内容,证书类型支持 X509 格式,支持 PEM 和 DER 编码,应符合 GB/T 20518。

	uiCertificateLen	[IN]证书长度。
	ppCertCtx	[OUT]指向证书上下文地址的指针。
返回值	ERR_SSF_OK：	成功。
	其他：	错误码。

7.2.2 释放证书上下文

函数原型 UINT32 SSFAPI SSF_FreeCertCtx(const SSF_CertificateContext * pCertCtx);

功能描述 释放证书上下文。

参数	pCertCtx	[IN]证书上下文地址。
返回值	ERR_SSF_OK：	成功。
	其他：	错误码。

7.2.3 添加证书到证书存储区

函数原型 UINT32 SSFAPI SSF_AddCert(const SSF_CertificateContext * pCertCtx,
const SSF_SKFCertRef * pSKFCertRef,UINT32 uiStoreType);

功能描述 添加证书到证书存储区，当 uiStoreType 为 SSF_CERT_CA_ROOT_STORE 时，导入根证书或 CA 中间证书到证书存储区，此时导入的根证书为本接口应用授信的根证书。

参数	pCertCtx	[IN]证书上下文。
	pSKFCertRef	[IN]SKF 证书对应信息。
	uiStoreType	[IN]证书存储区存储类型，为 SSF_CERT_CA_ROOT_STORE 或 SSF_CERT_USER_STORE
返回值	ERR_SSF_OK：	成功。
	其他：	错误码。

7.2.4 从证书存储区中删除证书

函数原型 UINT32 SSFAPI SSF_DelCert(const SSF_StoreContext * pStoreCertCtx);

功能描述 从证书存储区删除指定证书。

参数	pStoreCertCtx	[IN]证书存储上下文。
返回值	ERR_SSF_OK：	成功。
	其他：	错误码。

7.2.5 清除证书存储区中指定类型的所有证书

函数原型 UINT32 SSFAPI SSF_ClrAllCert(UINT32 uiStoreType);

功能描述 清除证书存储区中指定类型的所有证书。

参数	uiStoreType	[IN]证书存储区存储类型，为 SSF_CERT_CA_ROOT_STORE 或 SSF_CERT_USER_STORE
返回值	ERR_SSF_OK：	成功。
	其他：	错误码。

7.2.6 从证书存储区查找证书

函数原型 UINT32 SSFAPI SSF_FindCert(
const SSF_CertificateFindAttr * pCertificateFindAttr,
SSF_StoreContext_NODE * * ppStoreCtxNodeHeader);

功能描述 在证书存储区中查找证书。

参数	pCertificateFindAttr	[IN]查找属性。
	ppStoreCtxNodeHeader	[OUT]指向证书存储上下文链表的指针。

返回值　ERR_SSF_OK：　成功。
　　其他：　错误码。

7.2.7　遍历证书存储区中的证书

函数原型 UINT32 SSFAPI SSF_EnumCert(UINT32 uiStoreType,
　　SSF_StoreContext_NODE * * ppStoreCtxNodeHeader);
功能描述 在证书存储区中遍历证书，调用该函数，可获得证书存储上下文链表，并从中最终获得证书存储区中指定类型的所有证书。
参数　uiStoreType　[IN]证书存储区存储类型，为 SSF_CERT_CA_ROOT_STORE 或 SSF_CERT_USER_STORE
　　ppStoreCtxNodeHeader　[OUT]指向证书存储上下文链表的指针。
返回值　ERR_SSF_OK：　成功。
　　其他：　错误码。

7.2.8　释放证书存储上下文

函数原型 UINT32 SSFAPI SSF_FreeStoreCertCtx(const SSF_StoreContext * pStoreCertCtx);
功能描述 释放证书存储上下文。
参数　pStoreCertCtx　[IN]证书存储上下文，即指向证书存储结构的指针。
返回值　ERR_SSF_OK：　成功。
　　其他：　错误码。

7.2.9　释放证书存储上下文链表

函数原型 UINT32 SSFAPI SSF_FreeStoreCertCtxLink(
　　SSF_StoreContext_NODE * pStoreCtxNodeHeader);
功能描述 释放证书存储上下文链表。
参数　pStoreCtxNodeHeader　[IN]指向证书存储上下文链表的指针。
返回值　ERR_SSF_OK：　成功。
　　其他：　错误码。

7.2.10　创建 CRL 对象

函数原型 UINT32 SSFAPI SSF_CreateCRL(const UINT8 * pCRLData,UINT32 uiCRLDataLen,
　　SSF_CRL * * ppCRL);
功能描述 创建 CRL 对象。
参数　pCRLData　[IN]CRL 内容，支持 PEM 和 DER 编码。
　　uiCRLDataLen　[IN]CRL 长度。
　　ppCRL　[OUT]指向 CRL 对象的指针。
返回值　ERR_SSF_OK：　成功。
　　其他：　错误码。

7.2.11　释放 CRL 对象

函数原型 UINT32 SSFAPI SSF_FreeCRL(const SSF_CRL * pCRL);
功能描述 释放结构 CRL。
参数　pCRL　[IN]CRL 对象。
返回值　ERR_SSF_OK：　成功。
　　其他：　错误码。

7.2.12 添加 CRL 到证书存储区

函数原型 UINT32 SSFAPI SSF_AddCRL(const SSF_CRL * pCRL);
功能描述 添加 CRL 到证书存储区。
参数 pCRL [IN]CRL 对象。
返回值 ERR_SSF_OK: 成功。
其他: 错误码。

7.2.13 从证书存储区删除 CRL

函数原型 UINT32 SSFAPI SSF_DelCRL(const SSF_CRL * pCRL);
功能描述 从证书存储区删除 CRL。
参数 pCRL [IN]CRL 对象。
返回值 ERR_SSF_OK: 成功。
其他: 错误码。

7.2.14 在证书存储区中查找 CRL

函数原型 UINT32 SSFAPI SSF_FindCRL(const SSF_CRLFindAttr * pCRLFindAttr,
SSF_CRL_NODE * * ppNodeHeader);
功能描述 从证书存储区查找 CRL。
参数 pCRLFindAttr [IN]查找属性。
ppNodeHeader [OUT]指向 CRL 链表的指针。
返回值 ERR_SSF_OK: 成功。
其他: 错误码。

7.2.15 在证书存储区中遍历 CRL

函数原型 UINT32 SSFAPI SSF_EnumCRL(SSF_CRL_NODE * * ppNodeHeader);
功能描述 在证书存储区中遍历 CRL,调用该函数,可获得 CRL 链表,并最终获得所有的 CRL。
参数 ppNodeHeader [OUT]指向 CRL 链表的指针。
返回值 ERR_SSF_OK: 成功。
其他: 错误码。

7.2.16 释放 CRL 链表

函数原型 UINT32 SSFAPI SSF_FreeCRLLink(SSF_CRL_NODE * pNodeHeader);
功能描述 释放 CRL 链表。
参数 pNodeHeader [IN]指向 CRL 链表的指针。
返回值 ERR_SSF_OK: 成功。
其他: 错误码。

7.2.17 通过 OCSP 获取证书状态

函数原型 UINT32 SSFAPI SSF_GetCertStateByOCSP(
const UINT8 * pOCSPHostURL,UINT32 uiOCSPHostURLLen,
const UINT8 * pUsrCertificate,UINT32 uiUsrCertificateLen,
const UINT8 * pCACertificate,UINT32 uiCACertificateLen,UINT32 * puiState);
功能描述 通过 OCSP 获取证书状态。
参数 pOCSPHostURL [IN]OCSP 主机地址或主机 URL。

	uiOCSPHostURLLen	[IN]OCSP 主机地址或主机 URL 长度。
	pUsrCertificate	[IN]用户证书，支持 PEM 和 DER 编码。
	uiUsrCertificateLen	[IN]用户证书长度。
	pCACertificate	[IN]CA 证书或证书链内容，支持 PEM 和 DER 编码，当输入为证书链时，应符合 GB/T 35275。
	uiCACertificateLen	[IN]CA 证书或证书链长度。
	puiState	[OUT]证书状态，取值为下列之一：SSF_OCSP_STATE _GOOD、SSF_OCSP_STATE _REVOKED、SSF_OCSP_STATE _UNKNOWN。
返回值	ERR_SSF_OK：	成功。
	其他：	错误码。

7.2.18 通过 OCSP 响应获取证书状态

函数原型 UINT32 SSFAPI SSF_GetCertStateByOCSPResponse(const UINT8 * pUsrCertificate, UINT32 uiUsrCertificateLen, const UINT8 * pOCSPResponse, UINT32 uiOCSPResponseLen, UINT32 * puiState);

功能描述 通过输入的 OCSP 响应获取证书状态。例如从 OCSP 装订(OCSP Stapling)中获取的 OCSP 响应。

参数	pUsrCertificate	[IN]用户证书，支持 PEM 和 DER 编码。
	uiUsrCertificateLen	[IN]用户证书长度。
	pOCSPResponse	[IN]DER 编码 OCSP 响应内容。
	uiOCSPResponseLen	[IN]OCSP 响应长度。
	puiState	[OUT]证书状态，取值为下列之一：SSF_OCSP_STATE _GOOD、SSF_OCSP_STATE_REVOKED、SSF_OCSP_STATE_UNKNOWN。
返回值	ERR_SSF_OK：	成功。
	其他：	错误码。

7.2.19 通过 CRL 获取证书状态

函数原型 UINT32 SSFAPI SSF_GetCertStateByCRL(
const UINT8 * pCertificate, UINT32 uiCertificateLen,
const UINT8 * pCRLData, UINT32 uiCRLDataLen, UINT32 * puiState);

功能描述 通过 CRL 获取证书状态。

参数	pCertificate	[IN]用户证书，支持 PEM 和 DER 编码。
	uiCertificateLen	[IN]用户证书长度。
	pCRLData	[IN]CRL 内容，支持 PEM 和 DER 编码。
	uiCRLDataLen	[IN]CRL 长度。
	puiState	[OUT]证书状态，取值为下列之一：SSF_CERT_STATE _GOOD、SSF_CERT_STATE _OUT_TIME、SSF_CERT_STATE _NO_CHIAN、SSF_CERT_STATE _IN_CRL、SSF_CERT_STATE _INVALID_SIGN、SSF_CERT_STATE _FAIL。
返回值	ERR_SSF_OK：	成功。
	其他：	错误码。

7.2.20 通过 LDAP 获取证书

函数原型 UINT32 SSFAPI SSF_GetCertByLDAP(

const UINT8 * pLDAPHostURL,UINT32 uiLDAPHostURLLen,
const UINT8 * pQueryDN,UINT32 uiQueryDNLen,
const UINT8 * pOutCertificate,UINT32 * puiOutCertificateLen);

功能描述 通过 LDAP 方式获取证书。

参数

pLDAPHostURL	[IN]LDAP 主机地址内容。
uiLDAPHostURLLen	[IN]LDAP 主机地址长度。
pQueryDN	[IN]DER 编码证书 DN。
uiQueryDNLen	[IN]证书 DN 长度 。
pOutCertificate	[OUT]DER 编码证书。
puiOutCertificateLen	[IN/OUT]指向证书长度的指针,输入时为 pOutCertificate 缓冲区的长度,输出时为证书的实际长度。

返回值

ERR_SSF_OK:	成功。
其他:	错误码。

7.2.21 通过 LDAP 获取证书对应的 CRL

函数原型 UINT32 SSFAPI SSF_GetCRLByLDAP(
const UINT8 * pLDAPHostURL,UINT32 uiLDAPHostURLLen,
const UINT8 * pCertificate,UINT32 uiCertificateLen,
const UINT8 * pCRLData,UINT32 * puiCRLDataLen);

功能描述 通过 LDAP 方式获取证书对应的 CRL。

参数

pLDAPHostURL	[IN]LDAP 主机地址内容。
uiLDAPHostURLLen	[IN]LDAP 主机地址长度。
pCertificate	[IN]证书,支持 PEM 和 DER 编码。
uiCertificateLen	[IN]证书长度。
pCRLData	[OUT]DER 编码 CRL 内容。
puiCRLDataLen	[IN/OUT]指向 CRL 长度的指针,输入时为 pCRLData 缓冲区的长度,输出时为 CRL 内容的实际长度。

返回值

ERR_SSF_OK:	成功。
其他:	错误码。

7.2.22 通过 HTTP 获取证书

函数原型 UINT32 SSFAPI SSF_GetCertByHTTP(const UINT8 * pHTTPURL,UINT32 uiHTTPURLLen,const UINT8 * pOutCertificate,UINT32 * puiOutCertificateLen);

功能描述 通过 HTTP 方式获取证书。

参数

pHTTPURL	[IN]HTTP 地址内容。
uiHTTPURLLen	[IN]HTTP 地址长度。
pOutCertificate	[OUT]DER 编码证书。
puiOutCertificateLen	[IN/OUT]指向证书长度的指针,输入时为 pOutCertificate 缓冲区的长度,输出时为证书的实际长度。

返回值

ERR_SSF_OK:	成功。
其他:	错误码。

7.2.23 通过证书 AIA 更新证书链

函数原型 UINT32 SSFAPI SSF_UpdateChainByAIA(
const UINT8 * pCertificate,UINT32 uiCertificateLen);

功能描述 通过输入证书的 AIA 信息进行证书链更新。对于依据 AIA 信息获取的证书需要验证,

确认是已信任证书所签发。

参数　pCertificate　[IN]证书，支持 PEM 和 DER 编码。

uiCertificateLen　[IN]证书长度。

返回值　ERR_SSF_OK：　成功。

其他：　错误码。

7.2.24　证书有效性验证

函数原型 UINT32 SSFAPI SSF_VerifyCert(
UINT32 uiFlag,const UINT8 * pCertificate,UINT32 uiCertificateLen)；

功能描述 通过证书链、有效期等证书验证策略，验证证书有效性。

参数　uiFlag　[IN]验证标识，指明证书验证策略，
取值见表 2 常量定义 SSF_CERT_VERIFY_FLAG_XXX

pCertificate　[IN]证书内容，支持 PEM 和 DER 编码。

uiCertificateLen　[IN]证书长度。

返回值　ERR_SSF_OK：　成功。

其他：　错误码。

7.3　UI 接口

7.3.1　显示证书

函数原型 UINT32 SSFAPI SSF_ViewCert(const UINT8 * pCertificate,UINT32 uiCertificateLen)；

功能描述 显示证书。

参数　pCertificate　[IN]输入证书内容，支持 PEM 和 DER 编码。

uiCertificateLen　[IN]输入证书内容长度。

返回值　ERR_SSF_OK：　成功。

其他：　错误码。

备注　采用非阻塞模式函数调用，证书信息借鉴 Windows 证书显示，支持多级证书链显示，并显示证书链的验证，包括证书链有效性验证、证书有效期验证等，并添加目前规范已定义的 OID 描述，便于用户理解，显示界面示例如图 2 所示。其中 UI 风格可根据操作系统的不同做相应适配。

图 2　证书显示界面

7.3.2 选择证书

函数原型 UINT32 SSFAPI SSF_SelectCert(
const SSF_CertificateFindAttr * pCertificateFindAttr,
SSF_StoreContext * * ppStoreCertCtx);
功能描述 从证书存储区查找证书。
参数 pCertificateFindAttr [IN]查找属性。
ppStoreCertCtx [OUT]指向用户选择的证书存储上下文的指针。
返回值 ERR_SSF_OK: 成功。
其他: 错误码。
备注 采用阻塞模式函数调用,根据用户操作获取用户选择的数字证书,以证书存储上下文的形式返回。

7.4 SKF 管理接口

7.4.1 注册 SKF 到本规范定义的系统环境中

函数原型 UINT32 SSFAPI SSF_RegisterSKF(const SSF_SKF * pSKF);
功能描述 将 SKF 注册到本规范定义的系统环境中。
参数 pSKF [IN]SKF 对象。
返回值 ERR_SSF_OK: 成功。
其他: 错误码。
备注 应用调用该函数,实现将 SKF 注册到本规范定义的系统环境中,实现例程见附录 B。

7.4.2 从本规范定义的系统环境中注销 SKF

函数原型 UINT32 SSFAPI SSF_UnregisterSKF(const SSF_SKF * pSKF);
功能描述 将 SKF 从本规范定义的系统环境中注销。
参数 pSKF [IN]SKF 对象。
返回值 ERR_SSF_OK: 成功。
其他: 错误码。

7.4.3 枚举本规范定义的系统环境支持的 SKF

函数原型 UINT32 SSFAPI SSF_EnumSKF(SSF_SKF_NODE * * ppNodeHeader);
功能描述 枚举本规范定义的系统环境支持的 SKF。
参数 ppNodeHeader SKF [OUT]返回指向 SKF 链表的指针。
返回值 ERR_SSF_OK: 成功。
其他: 错误码。

7.4.4 释放 SKF 链表

函数原型 UINT32 SSFAPI SSF_FreeSKFLink(SSF_SKF_NODE * pNodeHeader);
功能描述 释放 SKF 链表。
参数 pNodeHeader [IN]指向 SKF 链表的指针。
返回值 ERR_SSF_OK: 成功。
其他: 错误码。

7.5 与其他接口规范的关系

对于 GM/T 0019,其主要目的是通过统一密码服务接口,向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。

对于 GM/T 0020，其定位于应用系统和典型密码服务接口之间，向应用层直接提供证书信息解析、基于数字证书身份认证和信息的机密性、完整性和不可否认性等高级密码服务，将应用系统的密码服务请求转向通用密码服务接口，通过通用密码服务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。

对于本规范，不关心具体的密码运算和密钥操作，仅定位于使浏览器与第三方应用能快捷地在终端操作系统上找到 SM2 数字证书，并获取与 SM2 数字证书相关的 SKF 接口信息。

在上述函数定义中，部分函数借鉴了 GM/T 0019。其主要区别在于：在 GM/T 0019 中接口的参数需指定 hAppHandle 句柄，在本文件中无需指定句柄；对于本文件中 SSF_GetCertStateByOCSP 和 SSF_GetCertStateByCRL 接口使用 puiState 参数返回获取的状态，GM/T 0019 中通过函数返回值获取状态。借鉴 GM/T 0019 的函数列表如表 17 所示。

表 17　借鉴 GM/T 0019 的函数列表

序号	本文件	GM/T 0019
1	UINT32 SSFAPI SSF_AddCRL(const SSF_CRL * pCRL);	int SAF_AddCRL(void * hAppHandle, unsignedchar * pucDerCrl, UINT32 uiDerCrlLen);
2	UINT32 SSFAPI SSF_GetCertStateByOCSP(const UINT8 * pOCSPHostURL, UINT32 uiOCSPHostURLLen, const UINT8 * pUsrCertificate, UINT32 uiUsrCertificateLen, const UINT8 * pCACertificate, UINT32 uiCACertificateLen, UINT32 * puiState);	int SAF_GetCertificateStateByOCSP(void * hAppHandle, BYTE * pcOcspHostURL, UINT32 uiOcspHostURLLen, BYTE * pucUsrCertificate, UINT32 uiUsrCertificateLen, BYTE * pucCACertificate, UINT32 uiCACertificateLen);
3	UINT32 SSFAPI SSF_GetCertStateByCRL(const UINT8 * pCert, UINT32 uiCertLen, const UINT8 * pCRLData, UINT32 uiCRLDataLen, UINT32 * puiState);	int SAF_VerifyCertificateByCrl(void * hAppHandle, BYTE * pucUsrCertificate, UINT32 uiUsrCertificateLen, unsignedchar * pucDerCrl, UINT32 uiDerCrlLen);
4	UINT32 SSFAPI SSF_GetCertByLDAP(const UINT8 * pLDAPHostURL, UINT32 uiLDAPHostURLLen, const UINT8 * pQueryDN, UINT32 uiQueryDNLen, const UINT8 * pOutCert, UINT32 * puiOutCertLen);	int SAF_GetCertFromLdap(void * hAppHandle, char * pcLdapHostURL, UINT32 uiLdapHostURLLen, BYTE * pucQueryDN, UINT32 uiQueryDNLen, BYTE * pucOutCert, UINT32 * puiOutCertLen);
5	UINT32 SSFAPI SSF_GetCRLByLDAP(const UINT8 * pLDAPHostURL, UINT32 uiLDAPHostURLLen, const UINT8 * pCertificate, UINT32 uiCertificateLen, const UINT8 * pCRLData, UINT32 * puiCRLDataLen);	int SAF_GetCrlFromLdap (void * hAppHandle, char * pcLdapHostURL, UINT32 uiLdapHostURLLen, BYTE * pucCertificate, UINT32 uiCertificateLen, BYTE * pucCrlData, UINT32 * puiCrlDataLen);

附 录 A
（规范性）
错误码定义和说明

错误码定义和说明如表 A.1 所示。

表 A.1 错误码定义和说明

宏描述	预定义值	说明
ERR_SSF_OK	0	成功
ERR_SSF_BASE	0xF0010000	错误码基础值
ERR_SSF_DLL_PATH	0xF0010001	获取函数地址失败
ERR_SSF_NO_APP	0xF0010002	没有应用
ERR_SSF_CREATE_STORE	0xF0010003	创建存储区失败
ERR_SSF_OPEN_STORE	0xF0010004	打开存储区失败
ERR_SSF_NO_CERT_CHAIN	0xF0010005	没有证书链
ERR_SSF_EXPORT_PUK	0xF0010006	导出公钥失败
ERR_SSF_VERIFY_CERT	0xF0010007	验证证书签名失败
ERR_SSF_VERIFY_TIME	0xF0010008	验证证书有效期失败
ERR_SSF_INVAILD_CERT	0xF0010009	非法证书，如证书被吊销等
ERR_SSF_CREATE_CERT_CONTEXT	0xF001000A	创建证书上下文失败
ERR_SSF_ADD_CERT	0xF001000B	添加证书上下文失败
ERR_SSF_DELETE_CERT	0xF001000C	删除证书上下文失败
ERR_SSF_ADD_CERT_TO_STORE	0xF001000D	添加证书到证书存储区失败
ERR_SSF_DEL_CERT_FROM_STORE	0xF001000E	从证书存储区删除证书失败
ERR_SSF_NO_RIGHT	0xF001000F	没有权限
ERR_SSF_SET_CERT_CONTEXT_PROPERTY	0xF0010010	设置属性出错
ERR_SSF_NO_MEM	0xF0010011	内存不足
ERR_SSF_BUFFER_TOO_SMALL	0xF0010012	输出缓冲区不足
ERR_SSF_INVALID_ARG	0xF0010013	参数错误
ERR_SSF_NO_CERT	0xF0010014	未找到证书
ERR_SSF_PARSE_CRL	0xF0010015	CRL 解析失败
ERR_SSF_ADD_CRL	0xF0010016	添加 CRL 失败
ERR_SSF_DELETE_CRL	0xF0010017	删除 CRL 失败
ERR_SSF_REGISTERSKF	0xF0010018	注册 SKF 失败
ERR_SSF_UNREGISTERSKF	0xF0010019	注销 SKF 失败
ERR_SSF_USERCANCEL	0xF0010020	用户选择了“取消”按钮
ERR_SSF_GENERAL_FAIL	0xF001FFFF	通用错误

附 录 B
（资料性）
使用本规范接口的例程

B.1 注册 SKF 以及证书存储

本例程实现证书注册，本例程用到的 SSF 接口有：SSF_RegisterSKF、SSF_AddCert。

本例程实现流程为：

a) 注册 SKF 库；

b) 读取 SKF 以及 SKF 接口读取到的证书；

c) 证书存储。

```
#define STORE_TYPE_USER 2
int testdemo()
{
    UINT32  uiRet = 0;
    SSF_Data  stName = {strlen("SKFName"), (UINT8*)"SKFName"};  // 名称
    SSF_Data stPath = {strlen("c:\\SKFName.dll"), (UINT8*)"c:\\SKFName.dll" }; // 路径
    //Linux 和国产操作系统 SKF 路径为 usr/lib/libSKFName.so
    SSF_SKF pSKF = { stName, stPath, 0, 0};
    SSF_SKFCertRef      *pSKFCertRef =NULL;
    SSF_StoreContext_NODE   *header = NULL;
    SSF_CertificateContext    *pCertCtx =NULL;
    SSF_CertificateFindAttr *pCertificateFindAttr = NULL;

    //----------------------1.将设备中证书注册到系统中--------------------------
    //1.1 注册 SKF 接口
    uiRet = SSF_RegisterSKF(&pSKF);
    if (ERR_SSF_OK != uiRet)
    {
        return uiRet;
}  //1.2 根据 SKF 信息，动态加载 SKF 接口库 LoadLibrary 或 dlopen
    HINSTANCE ghInst = DEV_LoadLibrary("c:\\SKFName.dll");
    if (! ghInst)
    {
        uiRet = ERR_SSF_DLL_PATH;
        return uiRet;
    }
    while (1)
    {
        //1.3 枚举 SKF 对应的证书
        //通过 SKF 接口从设备中获取证书，获取 stCertCtx，stSKFCertRef 结构
        //stCertCtx 为结构体 SSF_CertificateContext，
        //typedef struct _SSF_CertificateContext
        //{
        //SSF_CertificateAttr stAttr;
```

```
        //SSF_CertificateData stContent;
        //}SSF_CertificateContext;
        //其中 stAttr、stContent 在调用 SKF 获取证书后,可以获取对应内容。
        //stSKFCertRef 为结构体 SSF_SKFCertRef,结构体中
        // typedef struct _SSF_SKFCertRef
        //{
        //   SSF_Data   stSKFName;
        //   SSF_Data   stDeviceName;
        //   SSF_Data   stApplicationName;
        //   SSF_Data   stContainerName;
        //}SSF_SKFCertRef,都可以通过 SKF 接口获取,并保存。

        //1.4 将枚举到的证书存储到证书存储区
        uiRet = SSF_AddCert(pCertCtx,pSKFCertRef, STORE_TYPE_USER);
        if (ERR_SSF_OK ! = uiRet)
        {
            return uiRet;
        }
            //其他证书处理,没有证书退出循环...
    }

    //------------------------------------------------------------------------
    //其他资源释放...
    //1.5 根据 SKF 信息,释放 SKF 接口库
    Dev_FreeLibrary(ghInst);
    return uiRet;
}
```

本例程的工作流程图如图 B.1 所示。

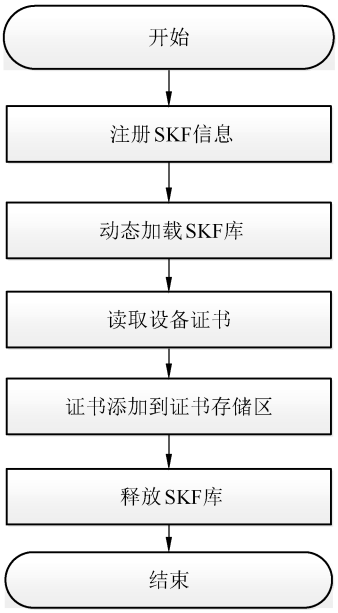

图 B.1　工作流程图

B.2 SKF 函数指针

本例程定义了 SKF 接口的函数指针类型，以及函数指针变量的读取。示例代码如下：

```
typedef ULONG(DEVAPI * pSKF_RSASignData)(HCONTAINER hContainer, BYTE * pbData, ULONGulDataLen, BYTE * pbSignature, ULONG * pulSignLen);
//声明和获取 SKF 函数指针宏定义
#if defined(WINDOWS) || defined(WIN32)
#else
    #include <dlfcn.h>
    typedef void * HINSTANCE;
    #define GetProcAddress dlsym
#endif
    #define FUNC_NAME_DECLARE(FUNC_NAME_PREFIX,FUNC_NAME,FUNC_NAME_SUFFIX) \
        pSKF_##FUNC_NAME FUNC_NAME_PREFIX##FUNC_NAME##FUNC_NAME_SUFFIX=NULL
    #define FUNC_NAME_INIT(FUNC_NAME_PREFIX, FUNC_NAME, FUNC_NAME_SUFFIX) (FUNC_NAME_PREFIX##FUNC_NAME##FUNC_NAME_SUFFIX) = (pSKF_##FUNC_NAME)GetProcAddress(ghInst,"SKF_"#FUNC_NAME); \
        if(NULL == FUNC_NAME_PREFIX##FUNC_NAME##FUNC_NAME_SUFFIX) \
    { \
        ulRet = ERR_SSF_DLL_PATH; \
        goto err; \
    }\
        else \
    { \
    }
```

B.3 加载释放 SKF 动态库

本例程定义了 SKF 动态库在不同系统下的加载和释放。示例代码如下：

```
HINSTANCE DEV_LoadLibrary(char * pszDllPath)
{
    HINSTANCE ghInst = NULL;
    if (NULL == ghInst)
    {
#if defined(WINDOWS) || defined(WIN32)
      ghInst = LoadLibraryA(pszDllPath);
#else
      ghInst = dlopen(pszDllPath, RTLD_LAZY);
#endif
    }
    return ghInst;
}
void DEV_FreeLibrary(HINSTANCE ghInst)
{
#if defined(WINDOWS) || defined(WIN32)
    FreeLibrary(ghInst);
```

```
#else
    dlclose(ghInst)
#endif
}
```

B.4 证书使用

本例程模拟了查找在证书存储区中已存储证书的使用流程，本例程使用到的 SSF 函数有：SSF_EnumCert、SSF_FindCert、SSF_FreeStoreCertCtxLink。

本例程使用到的 SKF 接口有：SKF_VerifyPIN、SKF_GetContainerType、SKF_ExportPublicKey、SKF_DigestInit 、SKF_Digest、SKF_ECCSignData、SKF_RSASignData。

本例程实现流程为：

a) 枚举(查找)证书；

b) 证书使用；

c) 释放证书存储上下文链表。

示例代码如下：

```
int testdemo2(char *pszPIN, ULONG *puiRetryCount)
{
    UINT32  uiRet = 0;
    SSF_StoreContext_NODE   *header = NULL;
    SSF_CertificateFindAttr *pCertificateFindAttr = NULL;
    UINT32 ulContainerType; //容器类型
    HAPPLICATION hAPP = NULL; //应用句柄
    HCONTAINER hCon = NULL; //容器句柄
    int uiSignType; //签名类型
    //------------------------------------------------------------------------
    //-------------------------2.使用证书-----------------------------------
    //两种方式使用其一获取证书：1.枚举所有证书，2.根据条件查找证书
    //2.1 枚举所有证书
    uiRet = SSF_EnumCert(STORE_TYPE_USER, &header);
    if (ERR_SSF_OK != uiRet)
    {
        return uiRet;
    }
    //2.2 根据条件查找证书
    uiRet = SSF_FindCert(pCertificateFindAttr, &header);
    if (ERR_SSF_OK != uiRet)
    {
        return uiRet;
    }
    if (NULL != header)
    {
        //2.3 找到用户选择的证书，根据证书获取证书和对应 SKF 等信息
        //header->ptr_data->stCertCtx;        //证书信息
        //header->ptr_data->stSKFCertRef;   //SKF 信息
        //2.4 根据 SKF 信息，动态加载 SKF 接口库
        //2.5 调用 SKF 接口，打开应用(SKF_OpenApplication)和容器(SKF_OpenContainer)
        //证书使用，以下为签名部分代码
```

```
        // 调用 SKF_VerifyPIN
        // 0,标准 PIN 有效使用
        // 1,无需调用校验 PIN 接口
        // 2,PIN 无效,但需调用校验 PIN 接口
        // 调用 SKF_GetContainerType 获取容器类型,代码省略
        if (2 == ulContainerType)    // SM2 签名
        {
            if (0 == uiSignType)  // 对 Hash 签名
            {
                // 调用 SKF_ECCSignData 签名,代码省略
            }
            else if (1 == uiSignType)  // 对原文签名
            {
                // 调用 GM/T 0100 定义的 SKF_SignData 进行签名,代码省略
            }
        }
        else  // RSA 签名
        {
            // 调用 SKF_RSASignData 签名,代码省略
        }
    }
    //2.6 调用 SKF 接口,实现签名(SKF_ECCSignData)
    SSF_FreeStoreCertCtxLink(&header);
    //其他资源释放...
    //2.7 根据 SKF 信息,释放 SKF 接口库
    return uiRet;
}
```

本例程的工作流程图如图 B.2 所示。

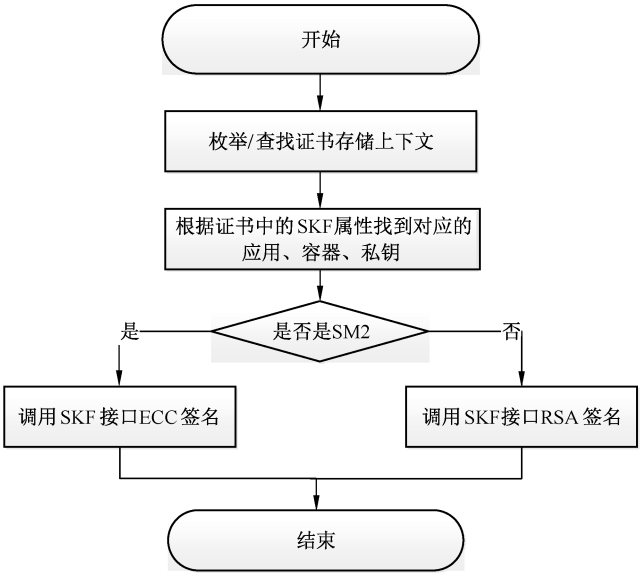

图 B.2　工作流程图

参 考 文 献

[1] GB/T 32918.5 信息安全技术 SM2 椭圆曲线公钥密码算法 第5部分:参数定义
[2] GM/T 0019 通用密码服务接口规范
[3] GM/T 0020 证书应用综合服务接口规范