2. ECC 签名

函数原型	ULONG DEVAPI SKF_ECCSignData (HCONTAINER hContainer, BYTE *pbData, ULONG ulDataLen, PECCSIGNATUREBLOB pSignature)	
功能描述	ECC 数字签名。采用 ECC 算法和指定私钥 hKey，对指定数据 pbData 进行数字签名。签名后的结果存放到 pSignature 中	
参数	hContainer	[IN] 密钥容器句柄
	pbData	[IN] 待签名的数据
	ulDataLen	[IN] 待签名数据长度，必须小于密钥模长
	pSignature	[OUT] 签名值
返回值	SAR_OK：　成功 其他：　错误码	
备注	权限要求：需要用户权限 输入数据为待签数据的哈希值	

3. 密码哈希初始化

函数原型	ULONG DEVAPI SKF_DigestInit(DEVHANDLE hDev, ULONG ulAlgID, ECCPUBLICKEYBLOB *pPubKey, unsigned char *pucID, ULONG ulIDLen, HANDLE *phHash)	
功能描述	初始化密码哈希计算操作，指定计算密码哈希值的算法	
参数	hDev	[IN] 连接设备时返回的设备句柄
	ulAlgID	[IN] 密码哈希算法标识
	pPubKey	[IN] 签名者公钥。当 ulAlgID 为 SGD_SM3 时有效
	pucID	[IN] 签名者的 ID 值，当 ulAlgID 为 SGD_SM3 时有效
	ulIDLen	[IN] 签名者 ID 的长度，当 ulAlgID 为 SGD_SM3 时有效
	phHash	[OUT] 密码哈希对象句柄
返回值	SAR_OK：　成功 其他：　错误码	
备注	在 ulAlgID 为 SGD_SM3 且 ulIDLen 不为 0 的情况下 pPubKey、pucID 有效，执行 SM2 算法签名预处理 1 操作（具体参见“8.2 SM2”节内容）	

4. 多组数据密码哈希

函数原型	ULONG DEVAPI SKF_DigestUpdate (HANDLE hHash, BYTE *pbData, ULONG ulDataLen)	
功能描述	对多个分组的消息进行密码哈希计算。调用 SKF_DigestUpdate 之前，必须调用 SKF_DigestInit 初始化密码哈希计算操作；调用 SKF_DigestUpdate 之后，必须调用 SKF_DigestFinal 结束密码哈希计算操作	
参数	hHash	[IN] 密码哈希对象句柄
	pbData	[IN] 指向消息数据的缓冲区
	ulDataLen	[IN] 消息数据的长度
返回值	SAR_OK：　成功 其他：　错误码	

5．结束密码哈希

函数原型	ULONG DEVAPI SKF_DigestFinal (HANDLE hHash, BYTE *pHashData, ULONG　*pulHashLen)	
功能描述	结束多个分组消息的密码哈希计算操作，将密码哈希结果保存到指定的缓冲区	
参数	hHash	[IN] 密码哈希对象句柄
	pHashData	[OUT] 返回的密码哈希结果缓冲区指针，如果此参数为 NULL，则由 pulHashLen 返回哈希结果的长度
	pulHashLen	[IN，OUT] 输入时表示哈希结果缓冲区的长度，输出时表示密码哈希结果的长度
返回值	SAR_OK：　成功 其他：　错误码	
备注	SKF_DigestFinal 必须用于 SKF_DigestUpdate 之后	

6．数据结构

（1）公钥数据结构 ECCPUBLICKEYBLOB

```
typedef struct Struct_ECCPUBLICKEYBLOB {
    ULONG    BitLen;
    BYTE     XCoordinate[ECC_MAX_XCOORDINATE_BITS_LEN/8];
    BYTE     YCoordinate[ECC_MAX_YCOORDINATE_BITS_LEN/8];
} ECCPUBLICKEYBLOB, *PECCPUBLICKEYBLOB;
```

其中，BitLen：模数的实际位长度，必须是 8 的倍数。XCoordinate：曲线上点的 X 坐标，有限域上的整数。YCoordinate：曲线上点的 Y 坐标，有限域上的整数。

```
#define ECC_MAX_XCOORDINATE_BITS_LEN 512
#define ECC_MAX_YCOORDINATE_BITS_LEN 512
```

ECC_MAX_XCOORDINATE_LEN 与 ECC_MAX_YCOORDINATE_LEN 分别为 ECC 算法 X 坐标和 Y 坐标的最大长度。

（2）签名数据结构 ECCSIGNATUREBLOB

```
typedef struct Struct_ECCSIGNATUREBLOB {
    BYTE r[ECC_MAX_XCOORDINATE_BITS_LEN/8];
    BYTE s[ECC_MAX_XCOORDINATE_BITS_LEN/8];
} ECCSIGNATUREBLOB, *PECCSIGNATUREBLOB;
```

其中，r：签名结果的 r 部分；s：签名结果的 s 部分。

```
#define ECC_MAX_MODULUS_BITS_LEN 512
```

ECC_MAX_MODULUS_BITS_LEN 为 ECC 算法模数的最大长度。

12.6.3.2　示例程序

与访问证书相比，使用私钥时，必须进行身份认证，即在打开应用句柄后，必须进行 PIN 码认证。其步骤包括：①连接设备；②获取设备信息（可选）；③打开应用句柄；④进行 PIN 码认证；⑤打开容器句柄；⑥进行签名。

使用私钥进行签名的示例代码如下：

```
ULONG        rv = 0;
DEVHANDLE hDev = NULL;
DEVINFO      DevInfo;
HAPPLICATION hApplication = NULL;
LPSTR        szAppName="My-Application";
LPSTR        containerName = "test";
HANDLE       hCon = NULL;
BYTE         bufCert[4096];
ULONG        ulCertLen = sizeof(bufCert);

LPSTR        szUserPin = "111111";
LPSTR        szUserId = "1234567812345678";
DWORD        dwUserPinRetryCount=10;
ECCSIGNATUREBLOB pSignature;
ECCPUBLICKEYBLOB eccSignPubKey;
HANDLE       hDigest = NULL;
BYTE         pbData[3072] = "12345678901234567890", pbDigest[64] = {0};
ULONG        ulDataLen = 33, ulSig = 0, ulDigest, pukLen = 0;

do {
    // 打开设备句柄
    rv = SKF_ConnectDev(“xxx 设备名称”, &hDev);
    if (rv) {
        printf("call SKF_ConnectDev error.\n");
        break;
    }
    // 获得设备的一些信息，可以进行展示
    rv = SKF_GetDevInfo(hDev, &DevInfo);
    if (rv) {
        printf("call SKF_GetDevInfo error.\n");
        break;
    }
    // 根据名称打开应用句柄，应用名称为 szAppName 中存储的“My-Application”
    rv = SKF_OpenApplication(hDev, szAppName, &hApplication);
    if (rv) {
        printf("call SKF_OpenApplication error.\n");
        break;
    }
    // 验证用户 PIN 码
    rv = SKF_VerifyPIN(hApplication, USER_TYPE, szUserPin, &dwUserPinRetryCount);
        printf("call SKF_VerifyPIN error.\n");
        break;
    }
```

```
        // 打开应用中容器，容器名称为 test
        rv = SKF_OpenContainer(hApplication, "test", &hCon);
        if (rv) {
            printf("call SKF_OpenContainer error\n");
            break;
        }
        // 导出容器中的签名公钥，用于计算签名哈希值
        //  TRUE 导出签名密钥，FALSE 导出加密密钥
        pukLen = sizeof(eccSignPubKey);
        rv = SKF_ExportPublicKey(hCon, TRUE, (unsigned char*)&eccSignPubKey, &pukLen);
        if (rv) {
            printf("SKF_ExportPublicKey(%0x) failed\n", rv);
            break;
        }
        // 计算签名哈希值
        rv = SKF_DigestInit(hDev, SGD_SM3, &eccSignPubKey, szUserId , 16, &hDigest);
        if (rv) {
            printf("call SKF_DigestInit error(%0x)\n", rv);
            break;
        }
        SKF_DigestUpdate(hDigest, pbData, totalLen);
        ulDigest = sizeof(pbDigest);
        rv = SKF_DigestFinal(hDigest, pbDigest, &ulDigest);
        if (rv) {
            printf("call SKF_Digest error(%0x)\n", rv);
            break;
        }
        SKF_CloseHandle(hDigest);
        hDigest = NULL;
        // 进行签名
        ulSig = sizeof(pSignature);
        rv = skf_ECCSignData(hCon, pbDigest, ulDigest, &pSignature);
        if (rv) {
            printf("Error: Sign Len(%ld) - result(%0x)\n", ulDataLen, rv);
            break;
        }
    } while(0);
    // 关闭打开句柄，以防止资源泄露
    if (hDigest) {   SKF_CloseHandle(hDigest);   }
    if (hCon) {      SKF_CloseContainer(hCon);    }
    if (hApplication) {    rv = SKF_CloseApplication(hApplication);   }
    if (hDev) {      SKF_DisconnectDev(hDev);    }
```

在计算 SM2 签名值前，需要对数据进行哈希运算，此哈希运算需要签名者公钥和用户 ID 参数。

第13章 实 验 二

13.1 RSA 公钥格式编码示例

13.1.1 ASN.1 描述与实例

1. ASN.1 描述

RSA 公钥格式用 ASN.1 描述如下：

```
RSAPublicKey ::= SEQUENCE {
    modulus           INTEGER,   -- n
    publicExponent    INTEGER    -- e
}
```

2. RSA 公钥实例

假设 RSA 公钥 PK={e，n}，e 为 65537，n 为 128 字节的大整数，用十六进制表示为：

```
n=B4 F6 CF 18 3D 5E 8E 1D 46 7A 90 7D 8E 41 D2 E3
  C8 F1 A3 AE F3 6D 8A 24 FF 55 23 25 BD EB 0C D0
  7B 87 36 5D 1F 73 98 65 3E 57 97 F6 65 7D 13 E0
  E1 B5 FC BC 38 6F 56 3E 57 4E D6 51 1D 13 12 7C
  33 B3 60 31 79 32 07 97 F3 3C 8B 29 0D B5 78 38
  93 CE 84 E4 A3 DD FB F9 25 47 1C 72 A6 5E 78 02
  CF F3 48 9D CA D9 00 73 DE 4B 16 07 52 48 20 06
  F3 4F CA A5 2D 66 88 95 C6 6C D6 3F 61 34 F7 E3
  （简写为：B4 F6…F7 F3）
e=01 00 01
```

13.1.2 DER 编码过程

1. 对 e 和 n 进行 DER 编码

e 和 n 为 INTEGER 结构类型，编码规则采用基本类型定长模式。

对于标识串，采用低标识编码方式，只需 1 个字节。INTEGER 的 tag 为 0x02；class 选择 universal，则位 8 和位 7 为 0，INTEGER 为基本类型，则位 6 为 0。因此，标识串=0x02。

对于长度串，e 采用短型编码方式，只需 1 个字节，n 采用长型编码方式，需要 2 个字节。

对于内容串，由 e 和 n 的十六进制值组成。由于 INTEGER 类型 DER 编码后第 1 字节位 8 表示正负整数，因此如果正整数第 1 字节位 8 为 1 时，在前填充 1 个字节 0x00。