信雅达公司关于区块链应用的
加密机 API 接口设计
(V1.02)

编制：任秋安

杭州信雅达科技有限公司

2016 年 12 月

目　录

1. API 接口函数种类

信雅达公司 SJJ1316 型加密设备向用户提供以下四类应用程序接口（API）函数：

- 设备连接类 API
- 公钥基础类 API
- 会话密钥协商类 API
- 加密解密类 API

2. API 接口函数的提供形式

信雅达公司 SJJ1316 型加密设备的应用程序接口（API）将以 API 头文件+静态库（或动态链接库）的形式提供给应用开发。针对不同的操作系统平台（IBM AIX、linux）提供各自独立的库文件。

应用程序接口（API）头文件及链接库定义如下：

■ *头文件：*

sydapi.h　　描述加密及 PIN 转换、签名服务等应用程序接口(API)原型

■ *链接库：*

libsydapi.so　　提供加密及 PIN 转换、签名服务等应用接口听动态库

3. API 函数的详细描述

3.1. 设备连接类 API

3.1.1 SYD_Connect

/*

功能：

建立到指定地址和端口号的加密设备的 TCP/IP 连接，返回 TCP/IP 连接句柄。

如果采用短连接，交易结束，要调用 SYD_Disconnect 进行释放连接。

函数原型：

```
int SYD_Connect(
        char* sIp,
        int nPort,
        char* sPwdStr,//预留，目前不用
        int* pSocketFd
```

```
        );
```

返回值：0 表示成功；非 0 表示失败；

参数说明：

sIp：　　输入。　加密设备的 IP 地址。

nPort：　　输入。　加密设备的端口号。

sPwdStr：　输入。　加密设备访问口令,目录保留不用。

pSocketFd：输出。　与加密设备建立的连接句柄。

3.1.2 SYD_Disconnect

/

功能：

关闭与加密设备的 TCP/IP 连接。

函数原型：

```
int SYD_Disconnect(
        int nSocketFd
        );
```

返回值：0 表示成功；非 0 表示失败；

参数说明：

nSocketFd：输入。　与加密设备建立的连接句柄。

3.2. 公钥基础类 API

3.2.1 SYD_SM2_GenKeyPair

/

功能：

产生 SM2 的密钥对，密钥强度 256，私钥通过本地 LMK 加密

函数原型：

```
int   SYD_SM2_GenKeyPair(
    int nSocketFd,
    unsigned char *pPriKey,
    int *pPriKeyLen,
    unsigned char *pPubKey;
    int *pPubKeyLen
    );
```

返回值：0 表示成功；非 0 表示失败；

参数说明：

nSocketFd：　　输入　　与加密设备建立的连接句柄。

pPriKey：	输出。	私钥串，HEX 格式,
pPriKeyLen：	输出。	返回私钥串的长度指针。默认输入值为缓冲区的大小。
pPubKey：	输出。	公钥串，DER 编码，HEX 格式。
pPubKeyLen：	输出。	返回公钥串的长度指针。默认输入值为缓冲区的大小。

3.2.2 SYD_SM2_Sign

/*
功能：**

用指定的私钥对指定的原始数据进行数字签名。

函数原型：

```
int SYD_SM2_Sign(
                int nSocketFd,
                unsigned char *pPriKey,
                unsigned char nPriKeyLen,
                int nOrgDataType,
                unsigned char *pPubKey,
                int nPubKeyLen,
                unsigned char* pOrgData,
                int nOrgDataSize,
                unsigned char* pSignData,
                int* pSignDataSize
              );
```

返回值：0 表示成功；非 0 表示失败；

参数说明：

nSocketFd：	输入。	与加密设备建立的连接句柄。
pPriKey：	输入。	本地 LMK 加密的私钥串，HEX 格式。
nPriKeyLen：	输入。	本地 LMK 加密的私钥串的长度。
nOrgDataType:	输入。	输入的数据类型，0：HASH 值，1：原始数据
pPubKey：	输入。	公钥串，DER 编码，HEX 格式。仅 OrgDataType=1 有效
nPubKeyLen：	输入。	公钥串的长度。仅 OrgDataType=1 有效
pOrgData：	输入。	待签名的原始数据。
nOrgDataSize：	输入。	待签名的原始数据的长度。
pSignData：	输出。	签名数据。
pSignDataSize：	输出。	签名数据长度。默认输入值为缓冲区的大小。

3.2.3 SYD_SM2_Verify

/*
功能：**

用指定的公钥对指定的原始数据进行数字签名验证。

函数原型：

```
int SYD_SM2_Verify(
```

```
int nSocketFd,
unsigned char *pPubKey,
int nPubKeyLen,
int nOrgDataType,
unsigned char* pOrgData,
int nOrgDataSize,
unsigned char* pSignData,
int nSignDataSize
);
```

返回值：0 表示验证正确；非 0 表示失败；

参数说明：

nSocketFd： 输入。 与加密设备建立的连接句柄。
pPubKey： 输入。公钥串，DER 编码，HEX 格式。
nPubKeyLen： 输入。公钥串的长度。
nOrgDataType: 输入。输入的数据类型，0：HASH 值，1：原始数据
pOrgData： 输入。 待签名的原始数据。仅 OrgDataType=1 有效
nOrgDataSize： 输入。 待签名的原始数据的长度。仅 OrgDataType=1 有效
pSignData： 输入。 签名数据。
pSignDataSize： 输入。 签名数据长度。

3.2.3 计算 SM3 散列值

/*

功能：**

输入对指定的原始数据和公钥，通过 SM3 算法计算数据的散列值。

函数原型：

```
int SYD_SM3_Hash(
                int nSocketFd,
                unsigned char *pPubKey,
                int nPubKeyLen,
                unsigned char* pOrgData,
                int nOrgDataSize,
                unsigned char* pHash
                );
```

返回值：0 表示成功；非 0 表示失败；

参数说明：

nSocketFd： 输入。 与加密设备建立的连接句柄。
pPubKey： 输入。公钥串，DER 编码，HEX 格式。
nPubKeyLen： 输入。公钥串的长度。
pOrgData： 输入。 原始数据。
nOrgDataSize： 输入。 原始数据的长度。
pHash： 输出。 HASH 值，32 个 Bytes.

3.3. 会话密钥协商类 API

3.3.1 SYD_SM2_GenSessionKey

/*

功能描述：**

此函数用于会话密钥的产生，用发起端。

加密机产生随机会话密钥，输出两类密文：

1、 公钥加密的密文，用于密钥的传递。

2、 本地 LMK 加密，用于本地应用数据的加解密。

函数原型：

```
int    SYD_SM2_GenSessionKey(
    int nSocketFd,
    unsigned char *pPubKey,
    int nPubKeyLen,
    unsigned char    *pCipherKey,
    int *pCipherKeyLen,
    unsigned char    *pSessionKey,
    int *pSessionKeyLen,
    unsigned char *pKCV
    );
```

返回值：0 表示成功；非 0 表示失败；

参数说明：

nSocketFd：	输入	与加密设备建立的连接句柄。
pPubKey：	输入。	公钥串，DER 编码，HEX 格式。
nPubKeyLen：	输入。	公钥串的长度。
pCipherKey：	输出。	公钥加密的会话密钥,DER 编码，HEX 格式
pCipherKeyLen：	输出。	返回公钥加密的密文长度指针。默认输入值为缓冲区的大小。
pSessionKey：	输出。	本地 LMK 加密的会话密钥，HEX 格式
pSessionKeyLen：	输出。	返回本地 LMK 加密的会话密钥长度，HEX 格式(不包括第 1 个字符)，一般为 33 个 Bytes。默认输入值为缓冲区的大小。
pKCV:	输出。	会话密钥的检验值，HEX 格式，长度为 32H 或 16H。当 pSessionKey 的第 1 个字节为’S’时，长度为 32H，否则为 16H。

3.3.2 SYD_SM2_ConfirmSessionKey

/*

功能描述：**

此函数用于会话密钥的接收，用于接收端。

加密机解密对方公钥加密密文，输出本地 LMK 加密的会话密钥，用于本地应用数据的加解密。

函数原型：

```
int    SYD_SM2_ConfirmSessionKey(
    int nSocketFd,
    unsigned char *pPriKey,
    int nPriKeyLen,
    unsigned char *pCipherKey,
    int nCipherKeyLen,
    unsigned char    *pSessionKey,
    int *pSessionKeyLen,
    unsigned char *pKCV
    );
```

返回值：0 表示成功；非 0 表示失败；

参数说明：

nSocketFd：	输入	与加密设备建立的连接句柄。
pPriKey：	输入。	本地 LMK 加密的私钥串，HEX 格式。
nPriKeyLen：	输入。	本地 LMK 加密的私钥串的长度。
pCipherKey：	输入。	公钥加密的会话密钥,DER 编码，HEX 格式
nCipherKeyLen：	输入。	公钥加密的密文长度
pSessionKey：	输出。	本地 LMK 加密的会话密钥，HEX 格式
pSessionKeyLen：	输出。	返回本地 LMK 加密的会话密钥长度，HEX 格式(不包括第 1 个字符)，一般为 33 个 Bytes。默认输入值为缓冲区的大小。
pKCV:	输出。	会话密钥的检验值，HEX 格式，长度为 32H 或 16H。当 pSessionKey 的第 1 个字节为’S’时，长度为 32H，否则为 16H。

3.4. 加密与解密类 API

3.4.1 SYD_SM4_Encrypt_Data

/*

功能：**

用输入的会话密钥，对指定报文数据进行加密处理。

报文长度以字节为单位，最长 2048 字节。

函数原型：

```
int    SYD_SM4_Encrypt_Data(
    int nSocketFd,
    unsigned char *pSessionKey,
```

```
    unsigned char* pInData,
    int nInDataSize,
    unsigned char* pOutData,
    int *pOutDataSize,
    );
```

返回值：0 表示成功；非 0 表示失败；

参数说明：

nSocketFd：	输入	与加密设备建立的连接句柄。
pSessionKey：	输入。	本地 LMK 加密的会话密钥，HEX 格式
pInData：	输入。	要加密的报文明文。
nInDataSize	输入。	报文输入长度，以字节为单位。
pOutData：	输出。	输出的报文密文。
pOutDataSize	输出。	输出的报文密文长度，8 或 16 的倍数。默认输入值为缓冲区的大小。

3.4.2 SYD_SM4_Decrypt_Data

/*

功能描述：

用输入的会话密钥，对指定报文数据进行解密处理。

报文长度以字节为单位，最大长度 2048 字节。

函数原型：

```
int   SYD_SM4_Decrypt_Data(
    int nSocketFd,
    unsigned char *pSessionKey,
    unsigned char* pInData,
    int nInDataSize,
    unsigned char* pOutData,
    int *pOutDataSize,
    );
```

返回值：0 表示成功；非 0 表示失败；

参数说明：

nSocketFd：	输入	与加密设备建立的连接句柄。
pSessionKey：	输入。	本地 LMK 加密的会话密钥，HEX 格式
pInData：	输入。	要解密的报文密文。
nInDataSize	输入。	报文输入长度，以字节为单位，必须是 8 或 16 的倍数。
pOutData：	输出。	输出的报文明文。
pOutDataSize	输出。	输出的报文明文长度。默认输入值为缓冲区的大小。

4. 其它说明

1. HEX 格式：可见字符，BCD 码的扩展。
2. 为了提高会话密钥的协商效率，目前采用由发起端产生会话密钥，接收端验证确认的方式。
3. 基于区块链去中心化的思想，也为了以后与加密卡模式兼容，目前私钥存于本地的区块链设备上，加密机上不存储任何应用密钥。
4. KCV 检验值，密钥校验值，对于两端会话密钥协商，用此值可以检验两端会话密钥的一致性，即同一个会话密钥，其 KCV 是相同的。
5. API 函数输出数据的长度值，在调用函数之前，必须进行初始化，其值必须为预留缓冲区的大小。