九鼎 E 系列加密锁

(JDKEY198)

开发手册

南通市九鼎软件科技有限公司

www.jdkey.cn

修订记录：

修订日期	版本	修订内容
2016 年 11 月	V1.0	第一版发布

一、产品概述

主要技术指标

• 32 位 ARM 智能卡安全芯片

• HID 界面的全速 USB 设备

• RSA1024+TDES128 的单向密钥协商机制，加密通信

• 可自定义编程下载到芯片内执行的多应用机制，程序空间为 64K

• 带有掉电恢复和磨损平衡机制的高可靠文件系统（最大支持 10 万次擦写），文件空间为 64K

• 基于匿名、用户、管理员的三级权限管理体系

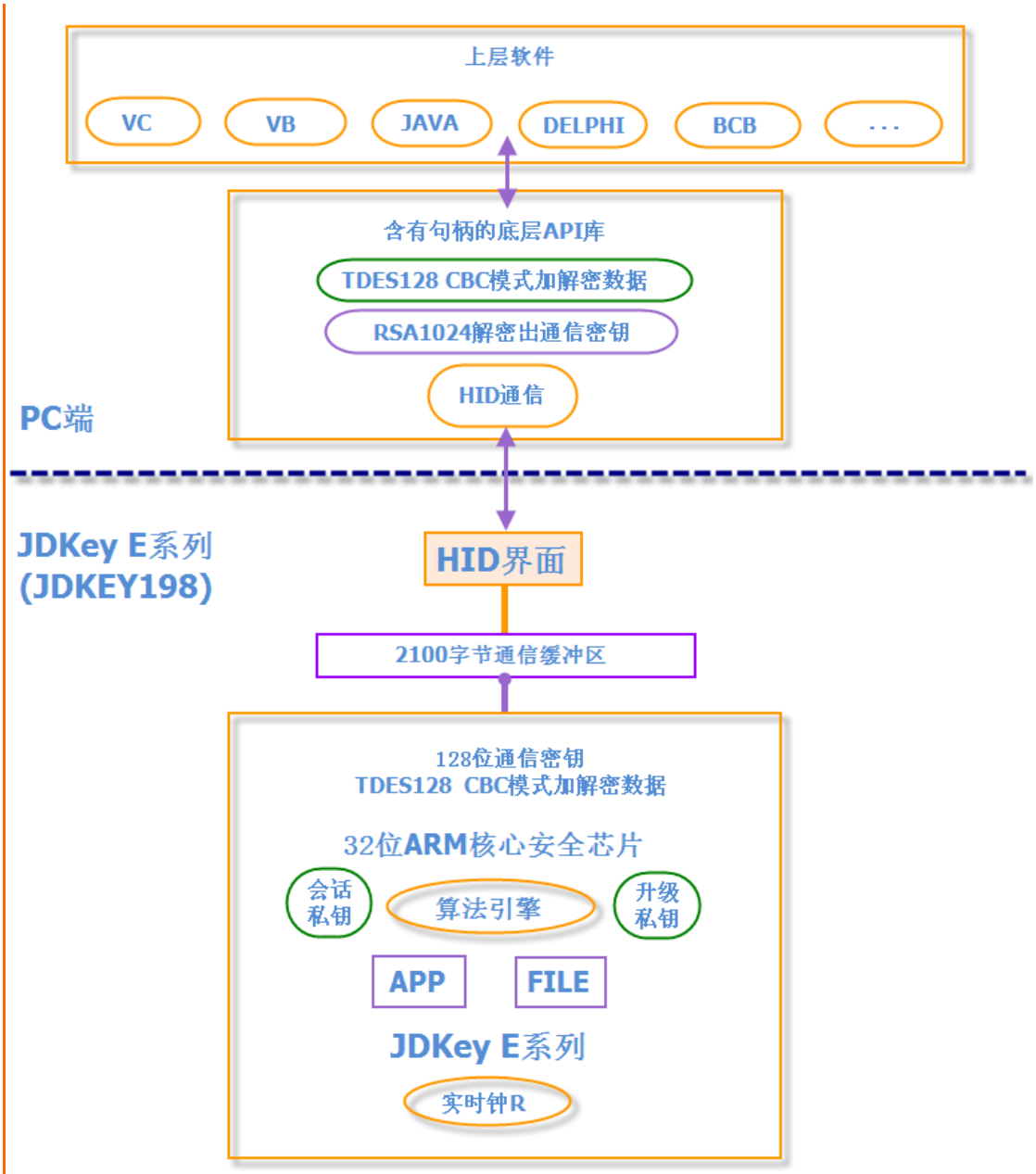
• 带有含硬件实时钟的型号，年走时误差小于 2 分钟

出厂默认设置

UserPIN：12345678

AdminPIN：FFFFFFFFFFFFFFFF

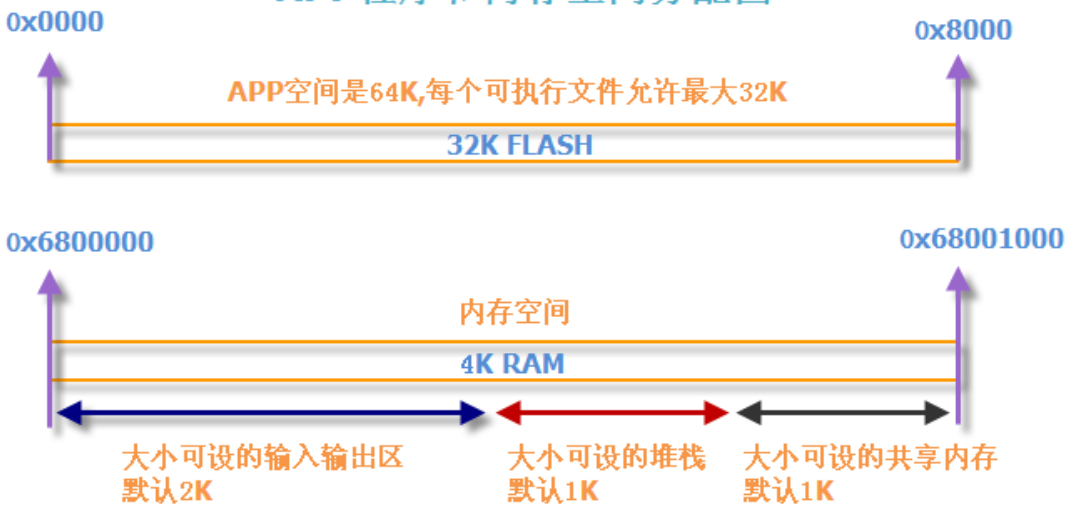
系统结构框图



型号	APP	FILE	实时钟
JDKEY-ES (JDKEY198)	64K	64K	
JDKEY-ER	64K	64K	支持

APP 使用指南

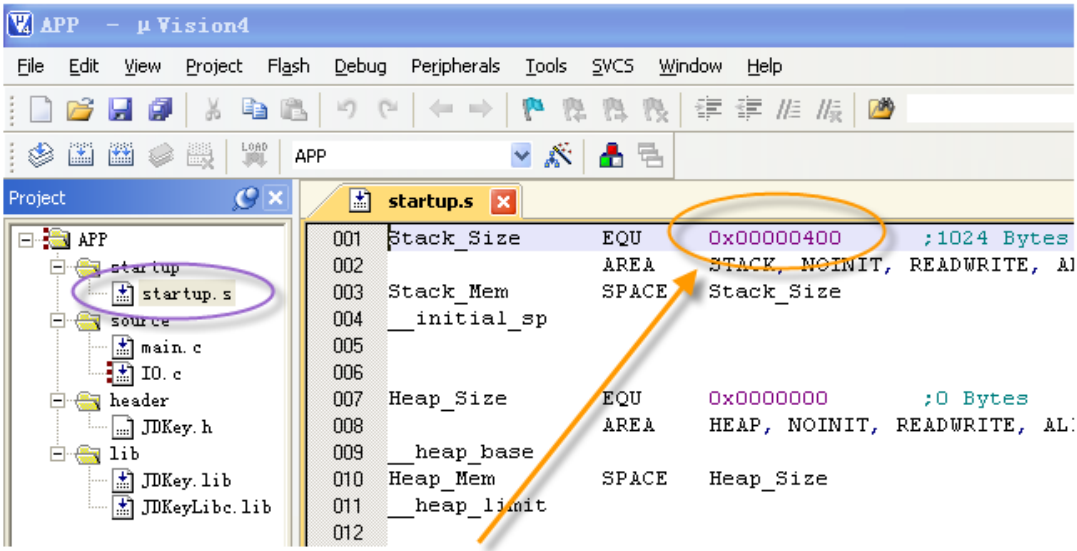
APP程序和内存空间分配图



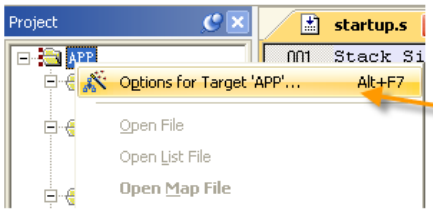
APP工程的设置与编译

打开一个最小的工程：

堆栈设置：

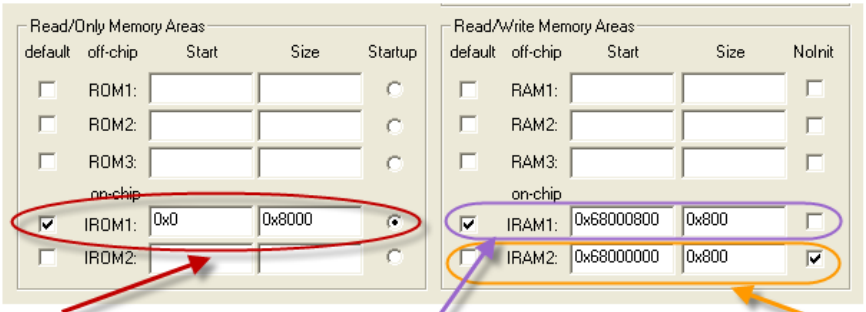


工程设置：



在工程的APP上点右键，在菜单中点"Options for Target 'APP'"

在Target页中可设置内存的分配：

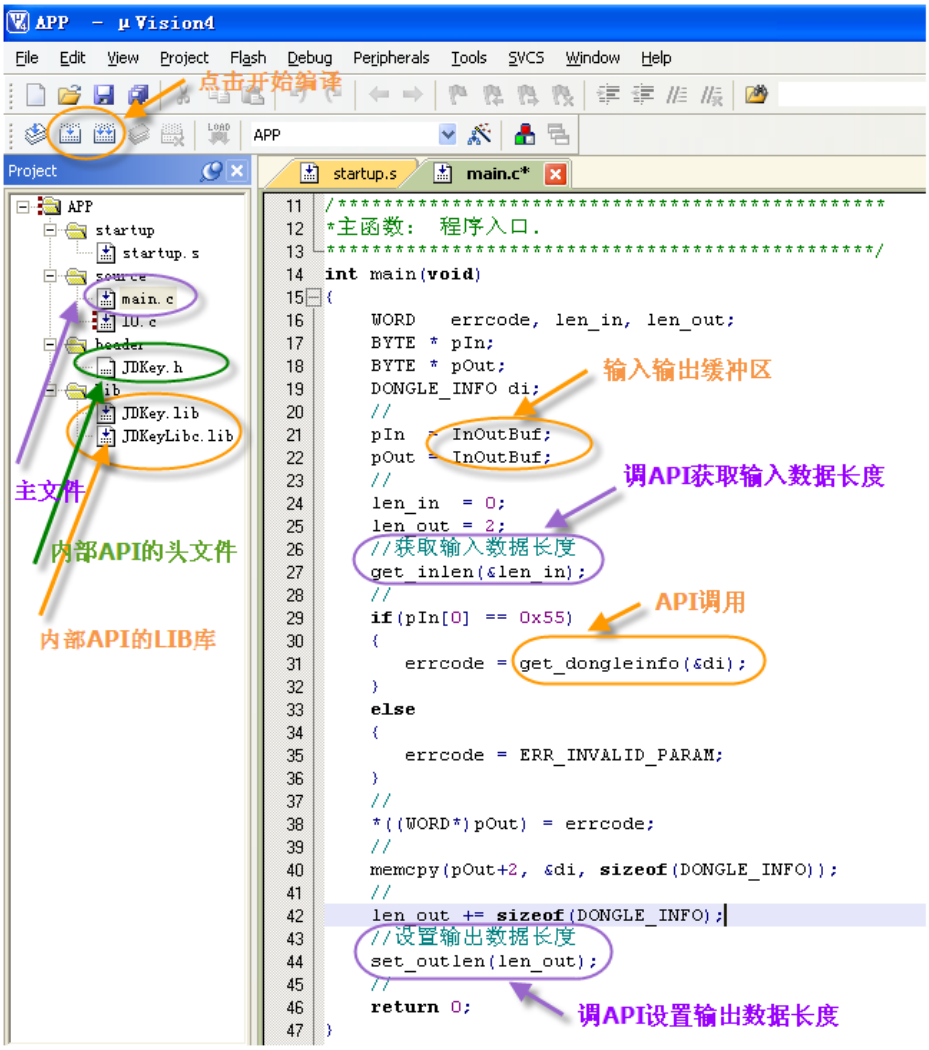


这是32K的Flash空间定义，不能更改

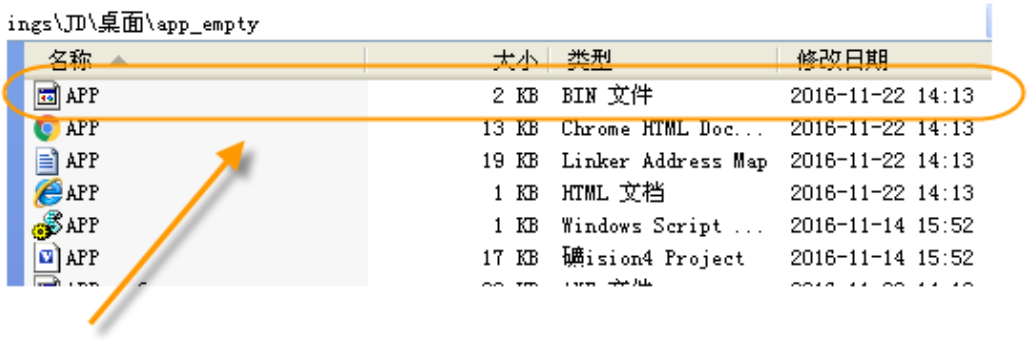
这是允许的最大堆栈空间，如果在startup.s中设的Stack_Size小于它，那多出来的高位内存可作为共享内存来用

这是输入输出缓冲区，默认是2K，如果用户的输入输出不需要这么大的吞吐量，可以按需设小些，多出的内存可用来增大堆栈或共享内存

编程使用：



编译的结果：



编译出的APP.bin就是APP了，通过文件接口下载后即可调用执行了

二、外部 API

1. JDKey_Enum 枚举加密锁

*DWORD WINAPI JDKey_Enum(OUT DONGLE_INFO * pDongleInfo, IN OUT int * pCount);*

说明：本函数会枚举出找到的设备（最多 64 个）

参数：

pDongleInfo [out]:设备信息的数组,当此参数为 NULL 时, pCount 返回找到的设备的数目

pCount [in][out]: 输入需要找的最大设备数目,0 表示不限制,返回实际找到的数目设备

返回值：

DONGLE_SUCCESS 执行成功。

2. JDKey_Open 打开加密锁

*DWORD WINAPI JDKey_Open(OUT DONGLE_HANDLE * phDongle, IN int nIndex);*

说明：本函数会查找并打开主机上的加密锁

参数：

phDongle [out]: 句柄指针,如果打开成功,会被填充句柄

index [in]: 基于 0 的索引值,指示打开找到的第几把加密锁

返回值：

DONGLE_SUCCESS 执行成功。

DONGLE_NOT_FOUND 未找到指定的加密锁

3.JDKey_Close 关闭加密锁

DWORD WINAPI JDKey_Close(IN DONGLE_HANDLE hDongle);

说明：本函数会关闭已打开的加密锁

参数：

hDongle [in]: 设备句柄

返回值：

DONGLE_SUCCESS 执行成功

4.JDKey_GetDongleInfo 获取硬件信息

*DWORD WINAPI JDKey_GetDongleInfo(IN DONGLE_HANDLE hDongle, OUT DONGLE_INFO * pDI);*

说明：本函数用于获取加密锁的基本信息，调用权限：匿名

参数：

hDongle [in]: 设备句柄

pDI [out]: 指向接收 Dongle 信息的缓冲区

返回值：

DONGLE_SUCCESS 执行成功

DONGLE_INFO 成员说明：

```
DWORD    m_Ver_Hardware;     //硬件版本 （Bit0 指示是否带硬时钟）
DWORD    m_Ver_Firmware;     //固件版本
DWORD    m_RealTime_UTC;     //UTC 实时钟 （格林威治时间）
DWORD    m_ProductID;        //产品 ID
BYTE     m_HardSN[16];       //16 字节的硬件 SN
BYTE     m_BirthDay[6];      //出厂时间
```

注意：m_RealTime_UTC 项仅对 ER 型号有效，在 ES 型号上此项值为 0，如果在 ER 型号上此项值为 0xFFFFFFFF，则说明时钟读取失败，原因可能是发生了电池失效、时钟停振等故障

m_ProductID 项（产品 ID）是由客户私有种子码计算生成的，具有唯一性，可用作产品的认证标识，出厂默认值为 FFFFFFFF

5.JDKey_VerifyPIN 校验 PIN 码

*DWORD WINAPI JDKey_VerifyPIN(IN DONGLE_HANDLE hDongle, IN int nFlag, IN char * pPIN, OUT int * pRemainCount);*

说明：本函数用于校验用户 PIN 码或管理员 PIN 码，共有 6 次重试机会

参数：

hDongle [in]: 设备句柄

nFlag [in]: FLAG_USERPIN、FLAG_ADMINPIN

pPIN [in]: 指向 8 字节长的用户 PIN 码字串，或 16 字节长的管理员 PIN 码字串

pRemainCount [out]: 返回的错误码是 DONGLE_INCORRECT_PIN 时，返回剩余的允许次数

返回值：

DONGLE_SUCCESS	执行成功
DONGLE_INCORRECT_PIN	PIN 码错误
DONGLE_PIN_BLOCKED	PIN 码已锁死

注意：

出厂时默认的用户 PIN 码为“12345678”

出厂时默认的管理员 PIN 码为“FFFFFFFFFFFFFFFF”

管理员 PIN 码锁死后继续输入，锁将恢复到出厂状态

6.JDKey_ChangePIN 修改 PIN 码

*DWORD WINAPI JDKey_ChangePIN(IN DONGLE_HANDLE hDongle, IN int nFlag, IN char * pOldPIN, IN char * pNewPIN, OUT int * pRemainCount);*

说明：本函数用于更改用户 PIN 码或管理员 PIN 码，共有 6 次重试机会

参数：

hDongle [in]: 设备句柄

nFlag [in]: FLAG_USERPIN、FLAG_ADMINPIN

pOldPIN [in]: 指向 8 字节长的旧用户 PIN 码字串，或 16 字节长的旧管理员 PIN 码字串

pNewPIN [in]: 指向 8 字节长的新用户 PIN 码字串，或 16 字节长的新管理员 PIN 码字串

pRemainCount [out]: 返回的错误码是 DONGLE_INCORRECT_PIN 时，返回剩余的允许次数

返回值：

DONGLE_SUCCESS	执行成功
DONGLE_INCORRECT_PIN	PIN 码错误
DONGLE_PIN_BLOCKED	PIN 码已锁死

7.JDKey_UnlockUserPIN 管理员解锁用户 PIN 码

*DWORD WINAPI JDKey_UnlockUserPIN(IN DONGLE_HANDLE hDongle, IN char * pAdminPIN, OUT int * pRemainCount);*

说明：本函数用于解锁用户 PIN 码，共有 6 次重试机会

参数：

hDongle [in]: 设备句柄

pAdminPIN [in]: 指向 16 字节长的管理员 PIN 码字串

pRemainCount [out]: 返回的错误码是 DONGLE_INCORRECT_PIN 时，返回剩余的允许次数

返回值：

DONGLE_SUCCESS	执行成功
DONGLE_INCORRECT_PIN	PIN 码错误
DONGLE_PIN_BLOCKED	PIN 码已锁死

8. JDKey_ResetPIN 复位 PIN 码状态到匿名

DWORD WINAPI JDKey_ResetPIN(IN DONGLE_HANDLE hDongle);

说明：本函数用于复位加密锁内的当前 PIN 码状态为匿名

参数：

hDongle [in]: 设备句柄

返回值：

DONGLE_SUCCESS	执行成功

9.JDKey_File_Format 格式化文件系统

DWORD WINAPI JDKey_File_Format(IN DONGLE_HANDLE hDongle);

说明：本函数用于格式化加密锁中的文件系统和可执行文件区，调用权限：管理员

参数：

hDongle [in]: 设备句柄

返回值：

DONGLE_SUCCESS 执行成功

DONGLE_ADMINPIN_NOT_CHECK 管理员密码没有验证

说明：

格式化后文件系统中的原有文件都将不存在

10.JDKey_File_Create 创建文件

DWORD WINAPI JDKey_File_Create(IN DONGLE_HANDLE hDongle, IN WORD fileid, IN BYTE filetype, IN WORD filelen);

说明：本函数用于创建加密机中的文件，调用权限：管理员

参数：

hDongle [in]: 设备句柄

fileid [in]: 文件 ID

filetype [in]: 文件类型

filelen [in]: 文件长度

返回值：

DONGLE_SUCCESS 执行成功

DONGLE_ADMINPIN_NOT_CHECK 管理员密码没有验证

说明：

1. 文件 ID 不能为保留值：0X0000，0X3F00，0XFFFF

2. 文件类型定义为：

```
#define   FILE_TYPE_EXE                0xFF
#define   FILE_TYPE_DATA               0
#define   FILE_TYPE_KEY_RSA_PRI        1
#define   FILE_TYPE_KEY_RSA_PUB        2
#define   FILE_TYPE_KEY_ECC_PRI        3
#define   FILE_TYPE_KEY_ECC_PUB        4
#define   FILE_TYPE_KEY_SM2_PRI        5
#define   FILE_TYPE_KEY_SM2_PUB        6
#define   FILE_TYPE_KEY_SYM_16         8
```

3．文件长度：对于密钥文件，文件长度必须是密钥数据的长度，例：

对于 RSA 私钥文件，文件长度必须为 sizeof（RSA_PRIVATE_KEY）

对于 RSA 公钥文件，文件长度必须为 sizeof（RSA_PUBLIC_KEY）

对于 ECC、SM2 私钥文件，文件长度必须为 sizeof（ECC_PRIVATE_KEY）

对于 ECC、SM2 公钥文件，文件长度必须为 sizeof（ECC_PUBLIC_KEY）

对于 128 位的对称密钥文件，文件长度必须为 16

11.JDKey_File_Write 写文件

*DWORD WINAPI JDKey_File_Write(IN DONGLE_HANDLE hDongle, IN WORD fileid, IN WORD offset, IN BYTE * pData, IN WORD len_Data);*

说明：本函数用于写加密锁中的文件，调用权限：管理员

参数：

hDongle [in]:设备句柄

fileid [in]: 要写入文件的文件 ID

offset [in]: 要写入文件的偏移地址

pData [in]: 指向要写入的数据

len_Data [in]: 要写入的数据长度

返回值：

DONGLE_SUCCESS 执行成功

DONGLE_ADMINPIN_NOT_CHECK 管理员密码没有验证

说明：

如果写入的是密钥文件，参数 offset 必须为 0

12.JDKey_RunAPP 运行 APP

*DWORD WINAPI JDKey_RunAPP(IN DONGLE_HANDLE hDongle, IN WORD appid, IN BYTE * pInBuf, IN int Len_In, OUT BYTE * pOutBuf, IN OUT int * pLen_Out);*

说明：本函数用于运行 APP，调用权限：用户

参数：

hDongle [in]: 设备句柄

appid [in]: 可执行文件的文件 ID

pInBuf [in]: 指向运行时的输入数据

Len_In [in]: 指向输入数据的长度

pOutBuf [out]: 指向接收运行结果的缓冲区，缓冲区的大小需要 2048 字节

pLen_Out [in][out]: 输入接收缓冲区的大小，接收返回数据的长度

返回值：

DONGLE_SUCCESS 执行成功

DONGLE_USERPIN_NOT_CHECK 用户密码没有验证

DONGLE_RUN_APP_ERROR 运行 APP 错误

13.JDKey_Seed 种子码运算

*DWORD WINAPI JDKey_Seed(IN DONGLE_HANDLE hDongle, IN BYTE * pSeed, IN int len_Seed, OUT BYTE * pResult, IN int flag);*

说明：本函数用于进行种子码运算，调用权限：匿名或用户

参数：

hDongle [in]: 设备句柄

pSeed [in]: 指向输入的种子码数据，不应超过 1024 字节

len_Seed [in]: 种子码数据的长度，应大于 0 小于等于 1024

pResult [out]: 返回运算结果，长度固定为 16 字节

flag [in]: 指示运算权限，FLAG_SEED_GUEST 或 FLAG_SEED_USER

返回值：

DONGLE_SUCCESS 执行成功

DONGLE_USERPIN_NOT_CHECK 用户密码没有验证

DONGLE_FAILED 操作失败

注意：

1.种子码运算的结果和客户设置的产品 ID、输入的种子等都有关，因此运算结果对一个产品而言具有唯一性，可用作冲击响应认证

2.输入同样的种子，匿名运算权限和用户运算权限下算出的结果是不同的，用户运算权限时需要先验证用户密码后才能进行运算

3.种子码运算可通过设置来限制允许运算的次数和到期时间 ，授权到期后将拒绝运算，返回操作失败

14.JDKey_Seed_SetLimit 设置种子码运算的限制数据

DWORD WINAPI JDKey_Seed_SetLimit(IN DONGLE_HANDLE hDongle, IN int iCount, IN DWORD dwTime);

说明：本函数用于设置种子码运算允许的次数和到期时间，调用权限：管理员

参数：

hDongle [in]: 设备句柄

iCount [in]: 指向允许运算的次数

dwTime [in]: 指向到期时间

返回值：

DONGLE_SUCCESS 执行成功

DONGLE_ADMINPIN_NOT_CHECK 管理员密码没有验证

注意：

1.iCount 值为-1 时表示取消运行次数限制

2.iCount 值不能为 0

3.dwTime 值为 0xFFFFFFFF 时表示取消到期时间限制

4.dwTime 值<=0xFFFF 时表示设置的是允许小时数，根据第一次进行种子码运算的时间计算出到期时间（仅支持硬时钟的型号有效）

5.dwTime 值为 UTC 值（不含时区的格林威治时间），当硬件时钟的时间超过此值时种子码运算将返回操作失败（仅支持硬时钟的型号有效）

6.硬件出厂时的状态是取消次数限制和到期时间限制

7.次数限制和时间限制可同时设置，硬件内部的判断顺序为先判断次数，再判断时间，其中一个过期时种子码运算即受限

15.JDKey_Seed_GetLimit 读取种子码运算的限制数据

*DWORD WINAPI JDKey_Seed_GetLimit(IN DONGLE_HANDLE hDongle, OUT int * piCount, OUT DWORD * pdwTime);*

说明：本函数用于获取种子码运算剩余的次数和到期时间，调用权限：匿名

参数：

hDongle [in]: 设备句柄

piCount [out]: 返回剩余的运算次数

pdwTime [out]: 返回到期时间

返回值：

DONGLE_SUCCESS 执行成功

注意：

1.返回的 iCount 值为-1 时表示无运行次数限制

2.返回的 iCount 值为 0 时表示运行次数已受限，授权已到期

3.返回的 dwTime 值为 0xFFFFFFFF 时表示无到期时间限制

4.返回的 dwTime 值<=0xFFFF 时表示设置的是允许小时数，但尚未进行种子码运算，（仅支持硬时钟的型号有效）

5.返回的 dwTime 值为 UTC 值（不含时区的格林威治时间）的到期时间，当硬件时钟的时间超过此值时种子码运算将返回操作失败（仅支持硬时钟的型号有效）

16.JDKey_Setup 设置参数

*DWORD WINAPI JDKey_Setup(IN DONGLE_HANDLE hDongle, IN RSA_PRIVATE_KEY * pUpdatePriKey, IN BYTE * pSeed, IN int Len_Seed, OUT DWORD * pProductID, OUT char * pAdminPIN);*

说明：本函数用于设置加密锁，进行私有化，调用权限：管理员

参数：

hDongle [in]: 设备句柄

pUpdatePriKey [in]: 指向 1024 位的 RSA 升级私钥，为 NULL 时表示不设置升级私钥

pSeed [in]: 指向不超过 256 字节的种子码

Len_Seed [in]: 种子码的长度 (1-256 字节)

pProductID [out]: 接收生成的产品 ID

pAdminPIN [out]: 接收生成的管理员密码

返回值：

DONGLE_SUCCESS 执行成功

DONGLE_ADMINPIN_NOT_CHECK 管理员密码没有验证

注意：

1. 产品 ID 与管理员密码是由种子码根据 HASH 和密码运算所得，具有单向唯一性
2. 客户输入的种子码仅用于实时运算，并不会存储在芯片中，以保证种子码的安全
3. 请务必记录并安全保存生成的管理员密码

17.JDKey_RFS 恢复出厂设置

DWORD WINAPI JDKey_RFS(IN DONGLE_HANDLE hDongle);

说明：本函数用于把加密锁恢复到出厂时的状态，调用权限：管理员

参数：

hDongle [in]: 设备句柄

返回值：

DONGLE_SUCCESS 执行成功

DONGLE_ADMINPIN_NOT_CHECK　管理员密码没有验证

注意：本函数调用后硬件会自动复位重启

18. JDKey_Request 发起升级请求

*DWORD WINAPI JDKey_Request(IN DONGLE_HANDLE hDongle, IN BYTE Request, OUT BYTE * pResponse, OUT int * pLen_Resp);*

说明：　本函数用于发起升级请求，调用权限：匿名

参数：

hDongle　　[in]: 设备句柄

Request　　[in]:　FLAG_REQUEST_UNLOCK_USERPIN

FLAG_REQUEST_UPDATE_CREATEFILE

FLAG_REQUEST_UPDATE_WRITEFILE

FLAG_REQUEST_UPDATE_SEED

pResponse　[out]: 指向接收缓冲区,返回加密机的升级请求,需要 128 字节

pLen_Resp　[out]:返回升级请求数据的长度

返回值：

DONGLE_SUCCESS　　执行成功

19.JDKey_Update 升级

*DWORD WINAPI JDKey_Update(IN DONGLE_HANDLE hDongle, IN BYTE * pUpdatePacket, IN int len_UpdatePacket);*

说明：　本函数用于远程升级，调用权限：匿名

参数：

hDongle　　[in]: 设备句柄

pUpdatePacket　[in]: 指向单包升级包

len_UpdatePacket　[in]: 指向单包升级包的长度

返回值：

DONGLE_SUCCESS 执行成功

DONGLE_COMM_ERROR 通信错误

20.JDKey_ParseRequest 解析用户的升级请求

*DWORD WINAPI JDKey_ParseRequest(IN BYTE * pResponse, IN int Len_Resp, IN RSA_PUBLIC_KEY * pUpdatePubKey, OUT BYTE * pKey, OUT BYTE * pSN, OUT BYTE * pRequest);*

说明：本函数用于解析用户端的升级请求

参数：

pResponse [in]: 用户发来的升级请求

Len_Resp [in]: 用户发来的升级请求长度

pUpdatePubKey [in]: 用来解析用户升级请求的 RSA1024 升级公钥

pKey [out]: 指向接收缓冲区,返回解析出的 16 字节升级密钥

pSN [out]: 指向接收缓冲区,返回解析出的 16 字节硬件序列号,用于确认用户身份

pRequest [out]: 指向接收缓冲区，返回解析出的 1 字节请求类型：

FLAG_REQUEST_UNLOCK_USERPIN

FLAG_REQUEST_UPDATE_CREATEFILE

FLAG_REQUEST_UPDATE_WRITEFILE

FLAG_REQUEST_UPDATE_SEED

返回值：

DONGLE_SUCCESS 执行成功

DONGLE_FAILED 操作失败

注意： 本函数是纯软件函数，不会访问加密锁硬件

21.JDKey_GenUpdatePacket 产生升级包

*DWORD WINAPI JDKey_GenUpdatePacket(IN BYTE Func, IN BYTE * pKey, IN BYTE * pIndata, IN int len_Indata, OUT BYTE * pUpdatePacket, OUT int * plen_UpdatePacket);*

说明：本函数用于生成升级包

参数：

Func:	[in]:	UPDATE_FUNC_REBOOT
		UPDATE_FUNC_UNLOCK_USERPIN
		UPDATE_FUNC_CREATE_FILE
		UPDATE_FUNC_WRITE_FILE
		UPDATE_FUNC_SEED_LIMIT
pKey	[in]:	指向 16 字节的解析出的升级密钥
pIndata	[in]:	指向要升级的数据
len_Indata	[in]:	指向要升级数据的长度，必须小于等于 2048 字节，且是 8 的整数倍
pUpdatePacket	[out]:	指向接收缓冲区,返回生成的升级包数据，缓冲区的大小需要 2056 字节
plen_UpdatePacket	[out]:	返回生成的升级包长度

返回值：

DONGLE_SUCCESS	执行成功

注意：

1.本函数是纯软件函数，不会访问加密锁硬件

2.当 Func 是 UNLOCK_USERPIN 时， pIndata 应指向 16 字节的用户硬件序列号

3.当 Func 是 CREATE_FILE 时,pIndata 应指向 UPDATE_CREATE_FILE 结构体

4. 当 Func 是 WRITE_FILE 时,pIndata 应指向 UPDATE_WRITE_FILE 结构体

5. 当 Func 是 SEED_LIMIT 时,pIndata 应指向 16 字节的用户硬件序列号 + 限次值(int) + 到期时间值(UTC)

附结构体定义：

```
typedef struct
{
    WORD    m_FILE_ID;          //文件 ID
    WORD    m_FILE_LEN;         //文件长度
    BYTE    m_FILE_TYPE;        //文件类型
    BYTE    m_Reserved[11];
} UDPATE_CREATE_FILE;


typedef struct
{
    WORD     m_FileID;          //文件 ID
    WORD     m_Offset;          //要写入的文件偏移地址
    WORD     m_Len;             //要写入的数据长度
    WORD     m_Reserved;        //保留
    BYTE     m_Buf[2040];     //要写入的数据
} UPDATE_WRITE_FILE;
```

三、内部 API

1.get_inlen 取输入数据长度

*WORD get_inlen(WORD * plen_in);*

说明：取输入数据长度

参数：

plen_in [out]: 输出接收到的数据长度 （1-2048）

2.set_outlen 设置输出数据长度

WORD set_outlen(WORD len_out);

说明：设置要返回的数据长度

参数：

len_out [in]: 输入要返回的数据长度 （1-2048）

3.create_file 创建文件

WORD create_file(WORD fileid, WORD filelen, BYTE filetype);

说明：创建文件

参数：

fileid [in]: 文件 ID，不能为 0x0000，0xFFFF，0x3F00 这三个值

filelen [in]: 文件长度，不能大于 0xFFFF

filetype [in]: 文件类型，共有以下几种文件类型

FILE_TYPE_DATA

FILE_TYPE_KEY_RSA_PRI

FILE_TYPE_KEY_RSA_PUB

FILE_TYPE_KEY_ECC_PRI

FILE_TYPE_KEY_ECC_PUB

FILE_TYPE_KEY_SM2_PRI

FILE_TYPE_KEY_SM2_PUB

FILE_TYPE_KEY_SYM_16

4.read_file 读文件

WORD read_file(WORD fileid, WORD offset, WORD len, BYTE pbuf);*

说明： 读文件

参数：

fileid [in]: 文件 ID

offset [in]: 偏移地址

len [in]: 长度

pbuf [out]: 读取缓冲区

注意： 只有数据文件才可读，密钥文件不可读出

5.write_file 写文件

WORD write_file(WORD fileid, WORD offset, WORD len, BYTE pbuf);*

说明： 写文件

参数：

fileid [in]: 文件 ID

offset [in]: 偏移地址，

注：写入密钥文件时 offset 必须为 0

len [in]: 长度，必须小于等于 2048 字节

注：写入密钥文件时 len 必须是相对应的密钥长度

pbuf [in]: 写入缓冲区

6. delete_file 删除文件

WORD delete_file(WORD fileid);

说明： 删除文件

参数：

fileid [in]: 文件 ID

7.get_freespace 取剩余空间大小

*WORD get_freespace(DWORD * pSpace);*

说明： 取文件系统剩余空间大小

参数：

pSpace [out]: 接收空间大小

8.get_dongleinfo 取硬件信息

*WORD get_dongleinfo(DONGLE_INFO * pDI);*

说明： 取 Dongle 信息

参数：

pDI [out]: 接收 Dongle 信息

9.led_control LED 控制

WORD led_control(BYTE flag);

说明： 控制 LED

参数：

flag [in]: LED_OFF 灭

LED_ON 亮

LED_BLINK_MODE_1	1HZ 闪
LED_BLINK_MODE_2	2HZ 闪
LED_BLINK_MODE_3	双闪
LED_BREATH	呼吸

10.get_random　取硬件随机数

WORD　get_random(BYTE pbuf, WORD len);*

说明：　取硬件随机数

参数：

pbuf　[in]:　指向接收缓冲区

len　[in]:　要取的随机数长度，应不大于 2048 字节

11. get_tickcount　取 Dongle 上电时间

*WORD　get_tickcount(DWORD * pCount);*

说明：　获取加密机的上电时间，单位是毫秒

参数：

pCount　[out]: 接收上电时间

12.get_RTC 读取实时钟

*WORD　get_RTC(DWORD * pUTC);*

说明：　获取硬件时钟的格林威治 UTC 时间，本函数仅带 R 的型号才支持

参数：

pUTC　[out]: 接收实时钟时间

返回值：ERR_SUCCESS 或 ERR_NOT_SUPPORT 或 ERR_FAILED

13. convert_RTC_TIME 转换 UTC 时间格式为 RTC_TIME 格式

*WORD convert_RTC_TIME(DWORD UTC, RTC_TIME * pRT);*

说明： 转换 UTC 时间为年月日时分秒周格式

参数：

UTC [in]: 输入要转换的 UTC 时间

pRT [out]: 返回转换的结果，成员说明如下：

```
typedef struct {
wYear;      年
bMonth;     月
bDay;       日
bHour;      时
bMinute;    分
bSecond;    秒
bWeek;      星期（0-6， 0 表示星期天）
} RTC_TIME;
```

14. rsa_genkey 生成 rsa 公私钥对

*WORD rsa_genkey(WORD bits, RSA_PRIVATE_KEY * pRPK);*

说明： 生成 RSA 公私钥对

参数：

bits [in]: 1024、2048

pRPK [out]: 接收生成的 RSA 私钥

注意：因为 RSA_PUBLIC_KEY 的结构就是 RSA_PRIVATE_KEY 结构的前面部分，所以公钥也就同时得到了

15.rsa_pri_f 基于私钥文件的 rsa 私钥运算

WORD rsa_pri_f(WORD fileid, BYTE pIn, WORD len, BYTE* pOut, WORD* plen_Out, BYTE flag);*

说明：使用文件系统中的 rsa 私钥文件进行 rsa 私钥运算

参数：

fileid [in]: 已创建并写入好的 RSA 私钥文件 ID

pIn [in]: 指向输入数据

len [in]: 输入数据长度

pOut [out]: 指向输出缓冲区

plen_Out [out]: 返回输出数据长度

flag [in]: FLAG_ENCODE、FLAG_DECODE

注意：这个函数内部已进行了标准 PKCS#1 填充，因此在进行加密操作时，len 应小于密钥位数 bits/8-11

16.rsa_pub_f 基于公钥文件的 rsa 公钥运算

WORD rsa_pub_f(WORD fileid, BYTE pIn, WORD len, BYTE* pOut, WORD* plen_Out, BYTE flag);*

说明：使用文件系统中的 rsa 公钥文件进行 rsa 公钥运算

参数：

fileid [in]: 已创建并写入好的 RSA 公钥文件 ID

pIn [in]: 指向输入数据

len [in]: 输入数据长度

pOut [out]: 指向输出缓冲区

plen_Out [out]: 返回输出数据长度

flag [in]: FLAG_ENCODE、FLAG_DECODE

注意：这个函数内部已进行了标准 PKCS#1 填充，因此在进行加密操作时，len 应小于密钥位数 bits/8-11

17.rsa_pri 基于外部传入私钥的 rsa 私钥运算

WORD rsa_pri(RSA_PRIVATE_KEY pPri, BYTE* pIn, WORD len, BYTE* pOut, WORD* plen_Out, BYTE flag);*

说明：使用外部传入的 rsa 私钥进行 rsa 私钥运算

参数：

pPri [in]: 传入的 RSA 私钥数据

pIn [in]: 指向输入数据

len [in]: 输入数据长度

pOut [out]: 指向输出缓冲区

plen_Out [out]: 返回输出数据长度

flag [in]: FLAG_ENCODE、FLAG_DECODE

注意：这个函数内部已进行了标准 PKCS#1 填充，因此在进行加密操作时，len 应小于密钥位数 bits/8-11

18.rsa_pub 基于外部传入公钥的 rsa 公钥运算

WORD rsa_pub(RSA_PUBLIC_KEY pPub, BYTE* pIn, WORD len, BYTE* pOut, WORD* plen_Out, BYTE flag);*

说明：使用外部传入的的 rsa 公钥进行 rsa 公钥运算

参数：

pPub [in]: 传入的 RSA 公钥数据

pIn [in]: 指向输入数据

len [in]: 输入数据长度

pOut [out]: 指向输出缓冲区

plen_Out [out]: 返回输出数据长度

flag [in]: FLAG_ENCODE、FLAG_DECODE

注意：这个函数内部已进行了标准 PKCS#1 填充，因此在进行加密操作时，len 应小于密钥位数 bits/8-11

19.ecc_genkey 生成 ecc 公私钥对

*WORD ecc_genkey(WORD bits, ECC_KEY_PAIR * pEKP);*

说明：生成 ecc 公私钥对

参数：

bits [in]: 192、256

pEKP [in]: 接收生成的 ecc 公私钥对

20.ecc_sign_f 基于私钥文件的 ECC 签名

WORD ecc_sign_f(WORD fileid, BYTE pIn, WORD len, BYTE* pOut, WORD* plen_Out);*

说明：使用 ECC 私钥文件进行签名

参数：

fileid [in]: ecc 私钥文件的 ID

pIn [in]: 待签名的 hash 输入数据

len [in]: hash 输入数据的长度，注意：应小于等于私钥长度

pOut [out]: 接收签名数据，需要 64 字节

plen_Out [out]: 接收签名数据长度，正常应返回 64 字节

21.ecc_verify_f 基于公钥文件的 ECC 验签

WORD ecc_verify_f(WORD fileid, BYTE pHash, WORD len_Hash, BYTE* pSign);*

说明：使用 ECC 公钥文件进行验签

参数：

fileid [in]: ecc 公钥文件的 ID

pHash [in]: 待验签的 hash 输入数据

len_Hash [in]: hash 数据的长度

pSign [in]: 签名数据，长度为 64 字节

22.ecc_sign 基于外部输入私钥数据的 ECC 签名

WORD ecc_sign(ECC_PRIVATE_KEY pPri, BYTE* pIn, WORD len, BYTE* pOut, WORD* plen_Out);*

说明：使用外部输入的 ECC 私钥进行签名

参数：

pPri [in]: ecc 私钥数据

pIn [in]: 待签名的 hash 输入数据

len [in]: hash 输入数据的长度，注意：应小于等于私钥长度

pOut [out]: 接收签名数据，需要 64 字节

plen_Out [out]: 接收签名数据长度，正常应返回 64 字节

23.ecc_verify 基于外部输入公钥数据的 ECC 验签

WORD ecc_verify(ECC_PUBLIC_KEY pPub, BYTE* pHash, WORD len_Hash, BYTE* pSign);*

说明：使用外部输入的 ECC 公钥进行验签

参数：

pPub [in]: ecc 公钥数据

pHash [in]: 待验签的 hash 输入数据

len_Hash [in]: hash 数据的长度

pSign　　[in]:　签名数据，长度为 64 字节

24.生成 SM2 公私钥对

*WORD　sm2_genkey(ECC_KEY_PAIR * pEKP);*

说明：生成 SM2 公私钥对

参数：

pEKP　[in]:　接收生成的 SM2 公私钥对

25.sm2_sign_f　基于私钥文件的 SM2 签名

WORD　sm2_sign_f(WORD fileid, BYTE pIn, WORD len, BYTE* pOut, WORD* plen_Out);*

说明：使用 SM2 私钥文件进行签名

参数：

fileid　　[in]:　sm2 私钥文件的 ID

pIn　　[in]:　待签名的 hash 输入数据

len　　[in]:　hash 输入数据的长度，注意：应小于等于 32 字节

pOut　　[out]:　接收签名数据，需要 64 字节

plen_Out [out]:　接收签名数据长度，正常应返回 64 字节

26.sm2_verify_f　基于公钥文件的 SM2 验签

WORD　sm2_verify_f(WORD fileid, BYTE pHash, WORD len_Hash, BYTE* pSign);*

说明：使用 SM2 公钥文件进行验签

参数：

fileid [in]: sm2 公钥文件的 ID

pHash [in]: 待验签的 hash 输入数据

len_Hash [in]: hash 数据的长度

pSign [in]: 签名数据，长度为 64 字节

27. sm2_sign 基于外部 sm2 私钥的 SM2 签名

WORD sm2_sign(ECC_PRIVATE_KEY pPri, BYTE* pIn, WORD len, BYTE* pOut, WORD* plen_Out);*

说明：使用外部输入的 SM2 私钥进行签名

参数：

pPri [in]: sm2 私钥数据

pIn [in]: 待签名的 hash 输入数据

len [in]: hash 输入数据的长度，注意：应小于等于 32 字节

pOut [out]: 接收签名数据，需要 64 字节

plen_Out [out]: 接收签名数据长度，正常应返回 64 字节

28.sm2_verify 基于外部 sm2 公钥的 SM2 验签

WORD sm2_verify(ECC_PUBLIC_KEY pPub, BYTE* pHash, WORD len_Hash, BYTE* pSign);*

说明：使用 SM2 公钥文件进行验签

参数：

pPub [in]: ecc 公钥数据

pHash [in]: 待验签的 hash 输入数据

len_Hash [in]: hash 数据的长度

pSign [in]: 签名数据，长度为 64 字节

29.tdes_f　基于对称密钥文件的 tdes 运算

WORD　tdes_f(WORD fileid, BYTE mode, BYTE pIv, BYTE* pdata ,WORD len, BYTE flag);*

说明：使用对称密钥文件进行 TDES 运算

参数：

fileid　[in]:　16 字节对称密钥文件的 ID

mode　[in]:　MODE_ECB、MODE_CBC

pIv　[in]:　指向 8 字节的初始向量

pdata　[in]:　输入数据

len　[in]:　输入数据长度，应不大于 2048 字节，且必须为 8 的整数倍

flag　[in]:　FLAG_ENCODE、FLAG_DECODE

30.tdes　基于外部对称密钥的 tdes 运算

*WORD　tdes(BYTE * pkey, BYTE mode, BYTE* pIv, BYTE* pdata ,int len, BYTE flag);*

说明：使用外部传入的对称密钥进行 TDES 运算

参数：

pkey　[in]:　16 字节的对称密钥

mode　[in]:　MODE_ECB、MODE_CBC

pIv　[in]:　指向 8 字节的初始向量

pdata　[in]:　输入数据

len　[in]:　输入数据长度，应不大于 2048 字节，且必须为 8 的整数倍

flag　[in]:　FLAG_ENCODE、FLAG_DECODE

31.aes_f　基于对称密钥文件的 aes 运算

WORD　aes_f(WORD fileid, BYTE mode, BYTE pIv, BYTE* pdata ,WORD*

len, BYTE flag);

说明：使用对称密钥文件进行 AES 运算

参数：

fileid [in]: 16 字节对称密钥文件的 ID

mode [in]: MODE_ECB、MODE_CBC

pIv [in]: 指向 16 字节的初始向量

pdata [in]: 输入数据

len [in]: 输入数据长度，应不大于 2048 字节，且必须为 16 的整数倍

flag [in]: FLAG_ENCODE、FLAG_DECODE

32.aes 基于外部对称密钥的 aes 运算

*WORD aes(BYTE * pkey, BYTE mode, BYTE* pIv, BYTE* pdata ,int len, BYTE flag);*

说明：使用外部传入的对称密钥进行 AES 运算

参数：

pkey [in]: 16 字节的对称密钥

mode [in]: MODE_ECB、MODE_CBC

pIv [in]: 指向 16 字节的初始向量

pdata [in]: 输入数据

len [in]: 输入数据长度，应不大于 2048 字节，且必须为 16 的整数倍

flag [in]: FLAG_ENCODE、FLAG_DECODE

33.sm4_f 基于对称密钥文件的 sm4 运算

WORD sm4_f(WORD fileid, BYTE mode, BYTE pIv, BYTE* pdata ,int len, BYTE flag);*

说明：使用对称密钥文件进行 SM4 运算

参数：

fileid [in]: 16 字节对称密钥文件的 ID

mode [in]: MODE_ECB、MODE_CBC

pIv [in]: 指向 16 字节的初始向量

pdata [in]: 输入数据

len [in]: 输入数据长度，应不大于 1024 字节，且必须为 16 的整数倍

flag [in]: FLAG_ENCODE、FLAG_DECODE

34.sm4 基于外部对称密钥的 sm4 运算

*WORD sm4(BYTE * pkey, BYTE mode, BYTE* pIv, BYTE* pdata ,int len, BYTE flag);*

说明：使用外部传入的对称密钥进行 SM4 运算

参数：

pkey [in]: 16 字节的对称密钥

mode [in]: MODE_ECB、MODE_CBC

pIv [in]: 指向 16 字节的初始向量

pdata [in]: 输入数据

len [in]: 输入数据长度，应不大于 1024 字节，且必须为 16 的整数倍

flag [in]: FLAG_ENCODE、FLAG_DECODE

35.md5 基于 md5 算法的 hash 运算

WORD md5(BYTE pdata ,int len, BYTE* phash);*

说明：使用 md5 算法进行 hash 运算

参数：

pdata [in]: 输入数据

len [in]: 输入数据长度，应不大于 2048 字节

phash [out]: 输出 16 字节的 hash 结果

36.sha1　基于 sha1 算法的 hash 运算

WORD　sha1(BYTE pdata ,int len, BYTE* phash);*

说明：使用 sha1 算法进行 hash 运算

参数：

pdata　[in]:　输入数据

len　[in]:　输入数据长度，应不大于 2048 字节

phash　[out]:　输出 20 字节的 hash 结果

37.sha256　基于 sha256 算法的 hash 运算

WORD　sha256(BYTE pdata ,int len, BYTE* phash);*

说明：使用 sha256 算法进行 hash 运算

参数：

pdata　[in]:　输入数据

len　[in]:　输入数据长度，应不大于 2048 字节

phash　[out]:　输出 32 字节的 hash 结果

38.sm3　基于 sm3 算法的 hash 运算

WORD　sm3(BYTE pdata ,int len, BYTE* phash);*

说明：使用 sm3 算法进行 hash 运算

参数：

pdata　[in]:　输入数据

len　[in]:　输入数据长度，应不大于 2048 字节

phash　[out]:　输出 32 字节的 hash 结果

39.seed　基于不公开种子码算法的 hash 运算

WORD　seed(BYTE pseed ,int len, BYTE* presult);*

说明：使用种子码算法进行不公开运算

参数：

pseed　[in]:　输入数据

len　[in]:　输入数据长度，应不大于 256 字节

presult　[out]:　输出 16 字节的运算结果

注意：此种子码运算中，客户设置的产品 ID 参与了运算，因此本算法的结果可用于硬件可信性识别