**PCIeSM 管理工具API手册**

**C语言版**

**V1.0**



北京江南天安科技有限公司

2021年09月4日

**版本：**

**当前版本：V1.0.0**

**历史版本：**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **日期** | **作者** | **描述** |
| 1.0.0 | 2021/09/04 | 李国亮 | 初始版本，接口函数说明 |

目录

[1 概述 5](#_Toc21754)

[2 全局说明 5](#_Toc3946)

[2.1.1 密码卡驱动 5](#_Toc11570)

[2.2 数据类型定义 6](#_Toc30978)

[2.2.1 常用数据类型 6](#_Toc16587)

[2.2.2 结构体数据类型 6](#_Toc22447)

[3 动态库文件 7](#_Toc14458)

[3.1 发布包目录 7](#_Toc5417)

[4 接口使用说明 8](#_Toc29595)

[4.1 编译 8](#_Toc26043)

[5 管理接口说明 8](#_Toc26152)

[5.1 初始设备 8](#_Toc8935)

[5.1.1 搜索设备 8](#_Toc11078)

[5.1.2 打开设备 9](#_Toc11330)

[5.1.3 关闭设备 9](#_Toc6449)

[5.2 管理员管理功能 9](#_Toc2093)

[5.2.1 获取用户列表 9](#_Toc18832)

[5.2.2 申请口令哈希加密公钥 10](#_Toc2898)

[5.2.3 哈希加密公钥加密口令hash值得到哈希值密文 10](#_Toc24959)

[5.2.4 登录设备 11](#_Toc17678)

[5.2.5 登出设备 11](#_Toc32462)

[5.2.6 修改口令 11](#_Toc8244)

[5.2.7 绑定UKey 12](#_Toc7235)

[5.2.8 添加密钥管理员 12](#_Toc25284)

[5.2.9 删除密钥管理员 13](#_Toc3299)

[5.3 设备管理功能 13](#_Toc19603)

[5.3.1 设备格式化 13](#_Toc28301)

[5.3.2 设备初始化 14](#_Toc21492)

[5.3.3 设备初始化恢复出厂 14](#_Toc13934)

[5.3.4 获取密码卡信息 15](#_Toc14559)

[5.3.5 设置密码卡信息 15](#_Toc19994)

[5.3.6 设备自检 16](#_Toc10803)

[5.3.7 恢复出厂设置 16](#_Toc11842)

[5.3.8 生成设备加密密钥对 16](#_Toc22004)

[5.3.9 生成设备本地保护密钥 16](#_Toc3180)

[5.3.10 导入设备加密密钥对 17](#_Toc17608)

[5.3.11 导入设备本地保护密钥 17](#_Toc6935)

[5.3.12 开机认证 18](#_Toc8695)

[5.3.13 导出设备加密密钥对 18](#_Toc6176)

[5.3.14 导出设备本地保护密钥 19](#_Toc23731)

[5.4 用户密钥管理 19](#_Toc895)

[5.4.1 获取密钥列表 19](#_Toc13441)

[5.4.2 生成密钥 20](#_Toc26749)

[5.4.3 导入明文密钥 21](#_Toc4591)

[5.4.4 导出非对称密钥 22](#_Toc13851)

[5.4.5 导入非对称密钥 22](#_Toc22016)

[5.4.6 删除密钥 23](#_Toc18249)

[5.4.7 修改私钥权限口令 24](#_Toc24053)

[5.4.8 备份设备信息 24](#_Toc526)

[5.4.9 恢复设备信息 25](#_Toc9818)

[5.4.10 校验备份数据信息的有效性 25](#_Toc27872)

# 概述

SJK1829 PCI-E 密码卡是江南天安科技有限公司自主研制，是支持国产密码算法（SM1、SM2、SM3、SM4、SM7）的密码设备，并且符合《密码设备应用接口规范》，可以提供多种国产密码算法，核心部件采用国家密码管理局批准的芯片，具备完善的密钥管理机制，可以提供高性能的数据加解密服务，又可以作为主机数据安全存储系统、身份认证系统以及对称、非对称密钥理

系统的主要密码设备和核心构件。可广泛的应用于银行、保险、证券、交通、邮政、电子商务、移动通信等行业的安全业务应用系统。

# 全局说明

本章节对接口依赖的密码卡驱动信息，以及接口的基础数据对象加以说明。

### 密码卡驱动

在使用密码卡的管理接口库之前，需要先安装密码卡的驱动，安装成功后才可正常使用，windows和驱动和linux的有所不同：

1. Windows环境：需要安装windows环境下的密码卡的驱动，向江南天安技术人员索要相应的驱动文件和详细安装说明；

2、Linux环境：密码卡驱动名称为tass\_pci.ko，由于该驱动编译时依赖操作系统内核，不同内核版本的操作系统编译的驱动有所不同，所以在使用加密卡的SDK接口库之前，要向江南天安技术人员提供使用的操作系统内核版本，便于提供对应内核版本的密码卡驱动。在拿到linux的驱动后按照如下步骤操作：

使用root权限安装该驱动，在驱动的当前路径执行命令insmod tass\_pci.ko进行测试安装；

正式使用时按照如下步骤安装驱动：

1）将1829密码卡插到机器内的PCIE槽位上，给机器加电卡的红灯“灭”，绿灯“闪”，表示工作状态正常；

2）将tass\_pci.ko上传到机器的/root目录下，在root目录下执行以下命令

创建驱动目录：

mkdir /lib/modules/`uname -r`/kernel/drivers/tass/

拷贝驱动：

cp tass\_pci.ko /lib/modules/`uname -r`/kernel/drivers/tass

安装驱动：

depmod -a

modprobe tass\_pci

3） 检查驱动版本

modinfo tass\_pci

若version 字段跟预期版本一致，则认为驱动安装成功

## 数据类型定义

### 常用数据类型

|  |  |
| --- | --- |
| **数据类型** | **描述** |
| unsigned char | 字符串类型 |
| void \* | 无类型指针，这里主要用于设备句柄使用 |
| int | 整形 |
| unsigned int | 无符号整形 |
| unsigned long | 无符号长整型 |

### 结构体数据类型

#define TASS\_DEV\_ADMIN\_NAME "admin"

#define TASS\_DEV\_ID\_STR\_SIZE 32

#define TASS\_NAME\_SIZE\_MAX 8

#define TASS\_UKEY\_SN\_SIZE 5

#define TASS\_KEY\_LABEL\_SIZE\_MAX 128

#define TASS\_KEY\_ATTR\_SIZE\_MAX 1024

#define TASS\_SK\_SIZE\_MAX 4096

#define TASS\_PK\_SIZE\_MAX 1024

#define TASS\_KEY\_SIZE\_MAX 64

#define TASS\_KCV\_SIZE 16

#define TASS\_DEV\_ADMIN\_CNT\_MAX 1

#define TASS\_KEY\_ADMIN\_CNT\_MAX 4

#define TASS\_ECC\_INDEX\_MAX 64

#define TASS\_SYMM\_INDEX\_MAX 64

typedef enum {

TA\_DEV\_ADMIN = 0,

TA\_KEY\_ADMIN = 1,

}TassAdminType;

|  |  |
| --- | --- |
| **结构体成员** | **描述** |
| type | 管理员类型 |
| name | 管理员名称 |
| ukeySn | UKey序列号，制作UKey后有效 |

typedef struct {

TassAdminType type;

char name[TASS\_NAME\_SIZE\_MAX + 1];

char ukeySn[TASS\_UKEY\_SN\_SIZE + 1];

}TassAdminInfo;

|  |  |
| --- | --- |
| **结构体成员** | **描述** |
| alg | 算法 |
| index | 索引 |
| label | 标签 |
| sk\_key | 私钥/对称密钥密文 |
| sk\_keyLen | 私钥/对称密钥密文长度 |
| sk\_keyAttr | 私钥/对称密钥属性 |
| sk\_keyAttrLen | 私钥/对称密钥属性长度 |
| pk\_kcv | 公钥/对称密钥校验值 |
| pk\_kcvLen | 公钥/对称密钥校验值长度 |
| pkAttr | 公钥属性，非对称时有效 |
| pkAttrLen | 公钥属性长度 |

typedef struct {

TassAlg alg;

int index;

char label[TASS\_KEY\_LABEL\_SIZE\_MAX + 1];

unsigned char sk\_key[TASS\_SK\_SIZE\_MAX];

unsigned int sk\_keyLen;

unsigned char sk\_keyAttr[TASS\_KEY\_ATTR\_SIZE\_MAX];

unsigned int sk\_keyAttrLen;

unsigned char pk\_kcv[TASS\_PK\_SIZE\_MAX];

unsigned int pk\_kcvLen;

unsigned char pkAttr[TASS\_KEY\_ATTR\_SIZE\_MAX];

unsigned int pkAttrLen;

}TassKeyInfo;

# 动态库文件

SJK1829 PCI-E 密码卡是以动态库形式对外提供服务，目前接口（密码卡管控）支持的系统有Windows、Linux的32位和64位系统。

## 发布包目录

所有动态库文件均保存在“libs”目录中，位置和说明见下表：**libTassCtlAPI4PCIeSM.\***是管控接口动态库，其他为依赖库

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **目 录** | **文 件** | **说 明** |
| 2 | demo | PCIeCtl\_Test.cpp | 接口调用实例 |
| 3 | doc | PCIeSM 管里工具API手册 C语言版 | 接口文档 |
| 4 | include | TassCtlAPI4PCIeSM.h  TassType4PCIeSM.h | 接口头文件  结构体定义相关头文件 |
| 5 | libs/linux32 | 暂无 | linux 32位操作系统动态库文件  暂不支持 |
| 6 | libs/linux64 | libtass\_pcie\_api.so  libTassSDF4PCIeSM.so  libTassCtlAPI4PCIeSM.so | linux 64位操作系统动态库文件 |
| 7 | libs/windows32 | 暂无 | windows 32位操作系统动态库文件  暂不支持 |
| 8 | libs/windows64 | 暂无 | windows 64位操作系统动态库文件  暂不支持 |

# 接口使用说明

## 编译

* Windows环境VS开发工具静态加载：将libTassCtlAPI4PCIeSM.lib导入到工程中，在工程配置属性->连接器->输入->附加依赖项中输入libTassCtlAPI4PCIeSM.lib所在目录，如：“./lib/libTassCtlAPI4PCIeSM.lib”。
* Linux环境gcc编译加载：动态库链接参数-L. –TassCtlAPI4PCIeSM。
  + 编译Linux的C语言程序：

gcc test.c -o test -I. -L. -lTassCtlAPI4PCIeSM -Wl,-rpath,.

* + 编译提供的demo，Linux环境：

g++ -std=c++11 PCIeCtl\_Test.cpp -o test -I. -L. -lTassCtlAPI4PCIeSM -Wl,-rpath,.

# 管理接口说明

## 初始设备

全部接口的入参和出参均封装成对象输入和输出，本文档内请求参数和返回参数针对各个字段作出解释。

### 搜索设备

**功能介绍**

搜索设备。

* **接口定义**

int TassCtlScanDevice(char\* id, unsigned int\* idLen);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| id | char\* | 是 | 输入，设备ID缓冲区，传NULL时通过len返回需要的缓冲区大小。 |
| idLen | unsigned int\* | 是 | 输入时标识id缓冲区大小，输出时标识id实际长度，多个设备ID间以‘\0’分隔，最后以两个'\0'结尾 |

* **返回参数**

成功返回0，失败返回非0。

### 打开设备

* **功能介绍**

该接口实现打开设备。

* **接口定义**

int TassCtlOpenDevice(const char id[TASS\_DEV\_ID\_STR\_SIZE + 1], void\*\* phDevice);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| id | const char | 是 | 输入，要打开的设备ID |
| phDevice | void\*\* | 是 | 输出，设备句柄 |

* **返回参数**

成功返回0，失败返回非0。

### 关闭设备

* **功能介绍**

该接口实现关闭设备，并释放相关资源的功能。

* **接口定义**

int TassCtlCloseDevice(void\* hDevice);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |

* **返回参数**

成功返回0，失败返回非0。

## 管理员管理功能

### 获取用户列表

* **功能介绍**

该接口实现获取设备用户列表的功能。

* **接口定义**

int TassCtlListAdmin(void\* hDevice, TassAdminInfo\* info, unsigned int\* infoLen);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| info | TassAdminInfo\* | 是 | 输入，管理员信息，传NULL时通过len返回需要的缓冲区大小 |
| infoLen | unsigned int\* | 是 | 输入时标识info缓冲区大小  输出时标识info实际长度  用户数量可通过计算“len / sizeof(TassAdminInfo)”获取 |

* **返回参数**

成功返回0，失败返回非0。

### 申请口令哈希加密公钥

* **功能介绍**

该接口实现申请口令哈希加密公钥的功能。

* **接口定义**

int TassCtlRequestPwdHashEncPublicKey(void\* hDevice, unsigned char pk[64]);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| pk | unsigned char | 是 | 输出，用于加密口令哈希的公钥 |

* **返回参数**

成功返回0，失败返回非0。

### 哈希加密公钥加密口令hash值得到哈希值密文

* **功能介绍**

该接口实现通过哈希加密公钥加密口令hash值得到哈希值密文的功能。

* **接口定义**

int TassCtlEncryptPwdHash(void\* hDevice, const unsigned char pk[64], const char\* pwd, unsigned char pwdHashCipher[128]);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| pk | unsigned char | 是 | 输入，用于加密口令哈希的公钥 |
| pwd | const char\* | 是 | 用户登录口令 |
| pwdHashCipher | unsigned char | 是 | 输出，管理员口令的哈希值密文 |

* **返回参数**

成功返回0，失败返回非0。

### 登录设备

* **功能介绍**

该接口实现登录设备的功能。

* **接口定义**

int TassCtlLogin(void\* hDevice, const char\* name, const unsigned char pwdHashCipher[128]);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| name | const char\* | 是 | 输入，管理员名称，设备管理员固定为“admin” |
| pwdHashCipher | unsigned char | 是 | 输入，管理员口令的哈希值密文 |

* **返回参数**

成功返回0，失败返回非0。

### 登出设备

* **功能介绍**

该接口实现登出设备的功能。

* **接口定义**

int TassCtlLogout(void\* hDevice);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |

* **返回参数**

成功返回0，失败返回非0。

### 修改口令

* **功能介绍**

该接口实现修改口令的功能。

* **接口定义**

int TassCtlUpdatePwdHash(void\* hDevice,

const char\* name,

const unsigned char oldPwdHashCipher[128],

const unsigned char newPwdHashCipher[128]);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| name | const char\* | 是 | 输入，已登录的用户名 |
| oldPwdHashCipher[128] | const unsigned char | 是 | 输入，旧管理员口令的哈希值密文 |
| newPwdHashCipher[128] | const unsigned char | 是 | 输入，新管理员口令的哈希值密文 |

* **返回参数**

成功返回0，失败返回非0。

### 绑定UKey

* **功能介绍**

该接口实现绑定UKey的功能。

* **接口定义**

int TassCtlBindUKey(void\* hDevice,

const char\* name,

const char pwdHashCipher[128],

const char\* ukeyId);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| name | const char\* | 是 | 输入，已登录的用户名 |
| pwdHashCipher[128] | const char | 是 | 输入，管理员口令的哈希值密文 |
| ukeyId | const char\* | 是 | 输入，UKey序号 |

* **返回参数**

成功返回0，失败返回非0。

### 添加密钥管理员

* **功能介绍**

该接口实现添加密钥管理员的功能。

* **接口定义**

int TassCtlAddKeyAdmin(void\* hDevice,

const unsigned char devPwdHashCipher[128],

const char\* name,

const unsigned char pwdHashCipher[128]);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| devPwdHashCipher[128] | const unsigned char | 是 | 输入，设备管理员口令的哈希值密文 |
| name | const char\* | 是 | 输入，增加的密钥管理员名称，不能与已存在管理员名称相同 |
| pwdHashCipher[128] | const unsigned char | 是 | 输入，增加的密钥管理员口令的哈希值密文 |

* **返回参数**

成功返回0，失败返回非0。

### 删除密钥管理员

* **功能介绍**

该接口实现删除密钥管理员的功能。

* **接口定义**

int TassCtlRemoveKeyAdmin(void\* hDevice,

const unsigned char devPwdHashCipher[128],

const char\* name);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| devPwdHashCipher[128] | unsigned int | 是 | 输入，设备管理员口令的哈希值密文 |
| name | const char\* | 是 | 输入，删除的密钥管理员名称 |

* **返回参数**

成功返回0，失败返回非0。

## 设备管理功能

### 设备格式化

* **功能介绍**

该接口实现清除设备中的全部数据，包括设备密钥FLASH数据等功。

* **接口定义**

int TassCtlDeviceFormat(void\* hDevice, const unsigned char pwdHashCipher[128]);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| pwdHashCipher[128] | const unsigned char | 是 | 输入，设备管理员口令的哈希值密文 |

* **返回参数**

成功返回0，失败返回非0。

### 设备初始化

* **功能介绍**

该接口实现使用随机密钥初始化设备的功能。

* **接口定义**

int TassCtlDeviceInit(void\* hDevice,

const unsigned char newPwdHashCipher[128],

TassBool bootAuth,

const unsigned char devSn[4], unsigned int selfCheckCycle,

unsigned char kekCv[16]);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| newPwdHashCipher[128] | const unsigned char | 是 | 输入，新的设备管理员口令密文 |
| TassBool | bootAuth | 是 | 输入，是否开机认证 |
| devSn[4] | const unsigned char | 是 | 设备序列号，可以从设备表面标签或包装盒外部标签获取 |
| selfCheckCycle | unsigned int | 是 | 输入，设备自检周期 |
| kekCv[16] | unsigned char | 是 | 输出，设备本地保护密钥校验值，为NULL时不输出 |

* **返回参数**

成功返回0，失败返回非0。

### 设备初始化恢复出厂

* **功能介绍**

该接口实现使用出厂默认密钥初始化设备的功能。

* **接口定义**

int TassCtlDeviceInitRestoreFactory(void\* hDevice,

const unsigned char devSn[4],

unsigned int selfCheckCycle);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| devSn[4] | const unsigned char | 是 | 设备序列号，可以从设备表面标签或包装盒外部标签获取 |
| selfCheckCycle | unsigned int | 是 | 输入，设备自检周期 |

* **返回参数**

成功返回0，失败返回非0。

### 获取密码卡信息

* **功能介绍**

该接口实现获取密码卡信息的功能。

* **接口定义**

int TassCtlGetInfo(void\* hDevice,

TassDevInfo info,

void\* buf, unsigned int\* len);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| info | TassDevInfo | 是 | 信息类型，详见TassInfoType说明 |
| buf | void\* | 是 | 输出，具体信息，传NULL时通过len返回需要的缓冲区大小 |
| len | unsigned int\* | 是 | 输入时标识buf缓冲区大小  输出时标识buf实际长度 |

* **返回参数**

成功返回0，失败返回非0。

### 设置密码卡信息

* **功能介绍**

该接口实现设置密码卡信息的功能。

* **接口定义**

int TassCtlSetInfo(void\* hDevice,

TassDevInfo info,

void\* buf, unsigned int len,

const unsigned char sig[64]);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| info | TassDevInfo | 是 | 输入，信息类型，仅TA\_DEV\_SN和TA\_SELF\_CHECK\_CYCLE有效 |
| buf | void\* | 是 | 输入，具体信息，传NULL时通过len返回需要的缓冲区大小 |
| len | unsigned int | 是 | 输入，信息长度 |
| sig[64] | const unsigned char | 是 | 输入，管理员私钥对函数名和除hDevice及sig之外所有入参的签名 |

* **返回参数**

成功返回0，失败返回非0。

### 设备自检

* **功能介绍**

该接口实现使用随机密钥替换密码卡默认密钥的功能。

* **接口定义**

int TassCtlSelfCheck(void\* hDevice);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |

* **返回参数**

成功返回0，失败返回非0。

### 恢复出厂设置

* **功能介绍**

该接口实现恢复密码卡默认密钥和状态的功能。

* **接口定义**

int TassCtlRestoreFactory(void\* hDevice,

const unsigned char pwdHashCipher[128]);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| pwdHashCipher | const unsigned char | 是 | 输入，设备管理员口令的哈希值密文 |

* **返回参数**

成功返回0，失败返回非0。

### 生成设备加密密钥对

* **功能介绍**

该接口实现生成设备加密密钥对的功能。

* **接口定义**

int TassCtlGenDevEncKeyPair(void\* hDevice);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |

* **返回参数**

成功返回0，失败返回非0。

### 生成设备本地保护密钥

* **功能介绍**

该接口实现生成设备本地保护密钥的功能。

* **接口定义**

int TassCtlGenDevKEK(void\* hDevice, TassBool bootAuth);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| bootAuth | TassBool | 是 | 输入，是否开机认证 |

* **返回参数**

成功返回0，失败返回非0。

### 导入设备加密密钥对

* **功能介绍**

该接口实现导入设备加密密钥对的功能。

* **接口定义**

int TassCtlImportDevEncKeyPair(void\* hDevice,

const unsigned char pwdHashCipher[128],

const unsigned char pk[64],

const unsigned char skEnvelopByDevSignPk[144]);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| pwdHashCipher[128] | const unsigned char | 是 | 输入，设备管理员口令的哈希值密文 |
| pk[64] | const unsigned char | 是 | 输入，设备加密密钥对公钥 |
| skEnvelopByDevSignPk[144] | const unsigned char | 是 | 输入，设备签名密钥对封装的设备加密密钥对私钥信封 |

* **返回参数**

成功返回0，失败返回非0。

### 导入设备本地保护密钥

* **功能介绍**

该接口实现导入设备本地保护密钥的功能。

* **接口定义**

int TassCtlImportDevKEK(void\* hDevice,

const unsigned char pwdHashCipher[128],

TassBool bootAuth,

const unsigned char kekCipherByDevEncPk[112]);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| pwdHashCipher[128] | const unsigned char | 是 | 输入，设备管理员口令的哈希值密文 |
| bootAuth | TassBool | 是 | 输入，是否开机认证 |
| kekCipherByDevEncPk[112] | const unsigned char | 是 | 设备加密密钥对加密的设备本地保护密钥密文 |

* **返回参数**

成功返回0，失败返回非0。

### 开机认证

* **功能介绍**

该接口实现开机认证的功能。

* **接口定义**

int TassCtlBootAuth(void\* hDevice, const unsigned char pwdHashCipher[128]);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| pwdHashCipher[128] | const unsigned char | 是 | 输入，设备管理员口令的哈希值密文 |

* **返回参数**

成功返回0，失败返回非0。

### 导出设备加密密钥对

* **功能介绍**

该接口实现导出设备加密密钥对的功能。

* **接口定义**

int TassCtlExportDevEncKeyPair(void\* hDevice,

const unsigned char keyPwdHashCipher[128],

const unsigned char devPwdHashCipher[128],

const unsigned char pk[64],

unsigned char encPk[64],

unsigned char encSkEnvelopByPk[144]);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| keyPwdHashCipher[128] | const unsigned char | 是 | 输入，密钥管理员口令的哈希值密文 |
| devPwdHashCipher[128] | const unsigned char | 是 | 输入，设备管理员口令的哈希值密文 |
| pk[64] | const unsigned char | 是 | 输入，加密公钥，通常为另一个设备的签名公钥 |
| encPk[64] | unsigned char | 是 | 输出，设备加密密钥对公钥 |
| encSkEnvelopByPk[144] | unsigned char | 是 | 输出，加密公钥封装的设备加密密钥对私钥信封 |

* **返回参数**

成功返回0，失败返回非0。

### 导出设备本地保护密钥

* **功能介绍**

该接口实现导出设备本地保护密钥的功能。

* **接口定义**

int TassCtlExportDevKEK(void\* hDevice,

const unsigned char keyPwdHashCipher[128],

const unsigned char devPwdHashCipher[128],

const unsigned char pk[64],

const unsigned char encKek[112]);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| keyPwdHashCipher[128] | const unsigned char | 是 | 设备管理员口令的哈希值密文密钥管理员口令的哈希值密文 |
| devPwdHashCipher[128] | const unsigned char | 是 | 输入，设备管理员口令的哈希值密文 |
| pk[64] | const unsigned char | 是 | 输入，加密公钥，通常为另一个设备的加密公钥 |
| encKek[112] | unsigned char | 是 | 输出，加密公钥加密的设备本地保护密钥密文 |

* **返回参数**

成功返回0，失败返回非0。

## 用户密钥管理

### 获取密钥列表

* **功能介绍**

该接口实现密获取密钥列表的功能。

* **接口定义**

int TassCtlListKey(void\* hDevice,

TassAlg alg,

TassKeyInfo\* info, int\* len);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| alg | TassAlg | 是 | 输入，密钥列表的算法 |
| info | TassKeyInfo\* | 是 | 输入，密钥信息，传NULL时通过len返回需要的缓冲区大小 |
| len | int\* | 是 | 输入时标识info缓冲区大小  输出时标识info实际长度  密钥数量可通过计算“len / sizeof(TassKeyInfo)”获取 |

* **返回参数**

成功返回0，失败返回非0。

### 生成密钥

* **功能介绍**

该接口实现生成密钥的功能。

* **接口定义**

int TassCtlGenerateKey(void\* hDevice,

TassAlg alg,

unsigned int keyBits,

unsigned int index,

const char\* label,

const char\* pwd,

unsigned char\* sk\_key, unsigned int\* sk\_keyLen,

unsigned char\* pk\_kcv, unsigned int\* pk\_kcvLen);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| keyBits | unsigned int |  | 模长密钥类型为 2-SM2、3-256R1、8-256K1时，模长只支持256  密钥类型为 4-RSA时，模长只支持2048  密钥类型为 9-HMAC时，支持模长是128、256、384、512bit,  类型为0-SM4、1-SM1、5-AES、6-DES、7-SM7时，仅模长仅支持128 |
| alg | TassAlg | 是 | 输入，密钥的算法 |
| index | unsigned int | 是 | 输入，密钥索引，为0时根据标签存储密钥，为-1时不存储 |
| label | const char\* | 是 | 输入，密钥标签 |
| pwd | const char\* | 否 | 输出，私钥口令，type为非对称密钥时有效 |
| sk\_key | unsigned char\* | 是 | 输出，私钥密文或对称密钥密文，传NULL时通过sk\_keyLen返回需要的缓冲区大小 |
| sk\_keyLen | unsigned int\* | 是 | 输入时标识sk\_key缓冲区大小  输出时标识sk\_key实际长度 |
| pk\_kcv | unsigned char\* | 是 | 输出，公钥，传NULL时通过pk\_kcvLen返回需要的缓冲区大小 |
| pk\_kcvLen | unsigned int\* | 是 | 输入时标识pk\_kcv缓冲区大小  输出时标识pk\_kcv实际长度 |

* **返回参数**

成功返回0，失败返回非0。

### 导入明文密钥

* **功能介绍**

该接口实现导入明文密钥的功能。

* **接口定义**

int TassCtlImportPlainKey(void\* hDevice,

TassAlg alg,

unsigned int index,

const char\* label,

const char\* pwd,

const unsigned char\* sk\_key, unsigned int sk\_keyLen,

const unsigned char\* pk\_kcv, unsigned int pk\_kcvLen);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| alg | TassAlg | 是 | 输入，导入的密钥算法 |
| index | unsigned int | 是 | 输入，导入的密钥索引，为0时根据标签存储密钥，为-1时不存储 |
| label | const char\* | 是 | 输入，导入的密钥标签 |
| pwd | const char\* | 否 | 输入，私钥口令，type为非对称密钥时有效 |
| sk\_key | unsigned char\* | 是 | 输入，私钥明文或对称密钥明文 |
| sk\_keyLen | unsigned int\* | 是 | 输入，sk\_key长度 |
| pk\_kcv | unsigned char\* | 是 | 输入，公钥 |
| pk\_kcvLen | unsigned int\* | 是 | 输入，公钥长度 |

* **返回参数**

成功返回0，失败返回非0。

### 导出非对称密钥

* **功能介绍**

该接口实现通过sm2密钥保护导出非对称密钥的功能。

* **接口定义**

int TassCtlExportKey(void\* hDevice,

const unsigned char sm2Pk[64],

TassAlg exportedKeyAlg,

unsigned int exportedKeyIndex,

const unsigned char devPwdHashCipher[128],

unsigned char\* symmKeyCipher,

unsigned int\* symmKeyCipherLen,

unsigned char\* exportedPk,

unsigned int\* exportedPkLen,

unsigned char\* exportedKeyCipherByPk,

unsigned int\* exportedKeyCipherByPkLen);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| sm2Pk | const unsigned char | 是 | 输入，SM2保护公钥 |
| exportedKeyAlg | TassAlg | 是 | 输入，待导出的密钥算法 |
| exportedKeyIndex | unsigned int | 是 | 输入，待导出的密钥索引 |
| devPwdHashCipher | const unsigned char | 否 | 输入，设备管理员口令的哈希值密文 |
| symmKeyCipher | unsigned char\* | 是 | 输出，随机对称密钥密文 |
| symmKeyCipherLen | unsigned int\* | 是 | 输出，随机对称密钥密文长度 |
| exportedPk | unsigned char\* | 是 | 输出，待导出的公钥 |
| exportedPkLen | unsigned int\* | 是 | 输出，待导出的公钥长度 |
| exportedKeyCipherByPk | unsigned char\* | 是 | 输出，SM2公钥加密的待导出私钥密钥密文 |
| exportedKeyCipherByPkLen | unsigned int\* | 是 | 输出，SM2公钥加密的待导出私钥密钥密文长度 |

* **返回参数**

成功返回0，失败返回非0。

### 导入非对称密钥

* **功能介绍**

该接口实现通过sm2密钥保护导入非对称密钥的功能。

* **接口定义**

int TassCtlImportKey(void\* hDevice,

const unsigned char sm2Sk[32],

TassAlg importedKeyAlg,

unsigned int importedKeyIndex,

const unsigned char\* importedKeyLabel,

unsigned int importedKeyLabelLen,

unsigned char\* symmKeyCipher,

unsigned int symmKeyCipherLen,

const unsigned char\* importedPk,

unsigned int importedPkLen,

const unsigned char\* importedKeyCipherByPk,

unsigned int importedKeyCipherByPkLen);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| sm2Sk | const unsigned char | 是 | 输入，SM2保护私钥 |
| importedKeyAlg | TassAlg | 是 | 输入，待导入的密钥算法 |
| importedKeyIndex | unsigned int | 是 | 输入，待导入的密钥索引 |
| importedKeyLabel | const unsigned char | 否 | 输入，待导入的密钥标签文 |
| importedKeyLabelLen | unsigned char\* | 是 | 输入，待导入的密钥标签长度 |
| symmKeyCipher | unsigned char\* | 是 | 输入，随机对称密钥密文 |
| symmKeyCipherLen | unsigned int\* | 是 | 输入，随机对称密钥密文长度 |
| importedPk | unsigned char\* | 是 | 输入，待导入的公钥 |
| importedPkLen | unsigned int\* | 是 | 输入，待导入的公钥长度 |
| importedKeyCipherByPk | unsigned char\* | 是 | 输入，SM2公钥加密的待导入私钥密钥密文 |
| importedKeyCipherByPkLen | unsigned int\* | 是 | 输入，SM2公钥加密的待导入私钥密钥密文长度 |

* **返回参数**

成功返回0，失败返回非0。

### 删除密钥

* **功能介绍**

该接口实现删除密钥的功能。

* **接口定义**

int TassCtlDestroyKey(void\* hDevice,

TassAlg alg, unsigned int index);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| alg | TassAlg | 是 | 输入，密钥算法 |
| index | unsigned int | 是 | 输入，密钥索引 |

* **返回参数**

成功返回0，失败返回非0。

### 修改私钥权限口令

* **功能介绍**

该接口实现修改私钥权限口令的功能。

* **接口定义**

int TassCtlChangePrivateKeyAccessRight(void\* hDevice,

TassAlg alg, unsigned int index,

const unsigned char\* privateKeyPwd,

unsigned int privateKeyPwdLen);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| alg | TassAlg | 是 | 输入，密钥算法 |
| index | unsigned int | 是 | 输入，密钥索引 |
| privateKeyPwd | unsigned char\* | 是 | 私钥口令 |
| privateKeyPwdLen | unsigned int | 是 | 私钥口令长度 |

* **返回参数**

### 备份设备信息

* **功能介绍**

该接口实现备份，包括设备信息、密钥和FLASH索引信息的功能。

* **接口定义**

int TassCtlBackup(void\* hDevice,

const unsigned char keyPwdHashCipher[128],

const unsigned char devPwdHashCipher[128],

unsigned char\* info, unsigned int\* infoLen);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| keyPwdHashCipher[128] | const unsigned char | 是 | 输入，密钥管理员口令的哈希值密文 |
| devPwdHashCipher[128] | const unsigned char | 是 | 输入，设备管理员口令的哈希值密文 |
| info | unsigned char\* | 是 | 输出，备份信息，传NULL时通过infoLen返回需要的缓冲区大小 |
| infoLen | unsigned int\* | 是 | 输入时标识info缓冲区大小  输出时标识info实际长度 |

* **返回参数**

成功返回0，失败返回非0。

### 恢复设备信息

* **功能介绍**

该接口实现恢复设备信息的功能。

* **接口定义**

int TassCtlRecover(void\* hDevice,

const unsigned char pwdHashCipher[128],

const unsigned char\* info, unsigned int infoLen);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| pwdHashCipher[128] | const unsigned char | 是 | 输入，设备管理员口令的哈希值密文 |
| info | const unsigned char\* | 是 | 输入，备份的设备信息 |
| infoLen | unsigned int | 是 | 输入，info长度 |

* **返回参数**

成功返回0，失败返回非0。

### 校验备份数据信息的有效性

* **功能介绍**

该接口实现恢复设备信息的功能。

* **接口定义**

int TassCtlCheckBackupInfoValid(void\* hDevice,

const unsigned char pwdHashCipher[128],

const unsigned char\* info, unsigned int infoLen);

* **参数描述**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **是否必需** | **描述** |
| hDevice | void\* | 是 | 输入，已打开的设备句柄 |
| pwdHashCipher[128] | const unsigned char | 是 | 输入，设备管理员口令的哈希值密文 |
| info | const unsigned char\* | 是 | 输入，备份的设备信息 |
| infoLen | unsigned int | 是 | 输入，info长度 |

* **返回参数**

成功返回0，失败返回非0。