**机房板卡接口**

# 数据结构定义

数据结构版本统一为0x10001001

结构体定义中主要包含4种ID：

①用户ID，用于唯一标识一个用户实体，此类ID名称写为XXXUserID

②密钥ID，用于唯一标识一个云端密钥，此类ID名称写为XXXKeyID

③许可ID，用于唯一标识一个许可，此类ID名称写为XXXLicID

④公钥指纹，用于唯一标识一个用户公钥，为公钥结构体的HASH值，此类ID名称写为XXXKeyFingerprint

## 创建密钥记录请求

structure **KEY\_REC\_REQ**  
{  
 uint32 Version; // 数据结构版本号  
 uint32 Reserved; // 保留字，用于结构体对齐  
 uint8 KeyID[16]; // 密钥ID  
 uint8 OwnerUserID[16]; // 密钥所有者用户ID  
 uint8 OwnerKeyFingerprint[32];// 密钥所有者生成此请求使用的公钥的用户公钥的指纹  
 uint8 DevlpID[8]; // APP开发商ID  
 uint8 APPID[8]; // APP ID  
 int64 timeStamp; // 密钥请求生成的时间，5分钟之后失效  
 uint32 Flag; // 密钥类型，算法  
 uint32 bits; // 密钥bit长度  
 int64 stTime; // 起始时间  
 int64 endTime; // 结束时间  
 uint8 Signaute[256]; // 用户签名，SS端生成，硬件验证  
}

Flag定义：

#define ALGID\_AES 0x00000001 // AES密钥

#define ALGID\_RSA\_PUB 0x00010100 // RSA公钥

#define ALGID\_RSA\_PRI 0x00020100 // RSA私钥

#define ALGID\_SM2\_PUB 0x00010200 // SM2公钥

#define ALGID\_SM2\_PRI 0x00020200 // SM2私钥

## 密钥记录

该结构体为加密卡内部生成，内部解析，外部不解析，，外部会把它当成字节流，该定义为参考不是强制要求

structure **KEY\_REC**  
{  
 uint32 Version; // 版本号  
 uint32 Reserved; // 保留字，用于结构体手动对齐  
 uint8 KeyID[16]; // 密钥ID  
 uint8 OwnerUserID[16]; // 密钥所有者的用户ID  
 uint8 OwnerKeyFingerprint[32]; // 密钥所有者生成此云端密钥使用的公钥的用户公钥的指纹  
 uint8 DevlpID[8]; // APP开发商ID  
 uint8 APPID[8]; // APP ID  
 int64 stTime; // 起始有效时间  
 int64 endTime; // 结束时间  
 uint32 Flag; // 密钥类型，算法  
 uint32 bits; // 密钥bit长度  
 uint8 Key\_C\_enc[40] // 卡内部密钥加密的云端密钥  
 uint8 MAC[32] // 密钥记录的校验码，由硬件计算，硬件验证  
}

Flag定义同上

## 用户公钥信息

structure **USER\_PUB\_KEY**  
{  
 uint32 Version; // 版本号  
 uint8 OwnerUserID[16]; // 密钥所有者的用户ID  
 uint32 Reserved; // 保留字，用于结构体手动对齐  
 int64 timeStamp; // 密钥生成时间  
 uint32 Flag // 密钥类型，算法  
 uint32 bits; // 密钥bit长度  
 uint32 KeyLen // keyValue中的有效密钥长度  
 uint8 KeyValue[300]; //   
 uint8 Mac[32]; // 用户公钥的校验码，由硬件计算，硬件验证  
}

Flag定义同上

## 许可条款

structure **LIC\_LIMITED**  
{  
 uint32 version; // 版本号  
 uint32 flag // 标志位，标识后续条款的有效性  
 int64 stTime; // 起始时间  
 int64 endTime; // 结束时间  
 int64 fsTime; // 第一次使用该授权的时间  
 int64 spanTime // 可用时间段  
 int64 times // 可用此次数  
 uint32 Policy // 策略，标识读，写，打印，继承  
 uint32 Reserved; // 保留字，用于结构体手动对齐  
}

flag定义：

#define FLAG\_START\_TIME 0x00000001

#define FLAG\_END\_TIME 0x00000002

#define FLAG\_SPAN\_TIME 0x00000004

#define FLAG\_TIMES 0x00000008

Policy定义：

#define POLICY\_INHERIT 0x00000001 // 允许继承

#define POLICY\_DECRYPT 0x00000002 //允许解密

#define POLICY\_ENCRYPT 0x00000004 //允许加密

#define POLICY\_PRINT 0x00001000 // 允许打印

#define POLICY\_EXPORT 0x00002000 // 允许导出明文

#define POLICY\_UNREVOKABLE 0x00004000 // 不可撤销的

## 许可信息

该结构体为加密卡内部生成，内部解析，外部不解析，，外部会把它当成字节流，该定义为参考不是强制要求

structure **LICENSE**  
{  
 uint32 Version; // 版本号  
 uint8 LicID[16]; // 许可ID  
 uint8 FartherLicID[16]; // 父许可ID  
 uint8 IssuerUserID[16]; // 许可签发者的用户ID  
 uint8 OwnerUserID[16]; // 许可所有者的用户ID  
 uint8 OwnerKeyFingerprint[32]; // 被授权允许使用云端密钥的许可所有者的一个用户公钥的指纹  
 uint8 KeyID[16]; // 被授权的云端密钥ID  
 uint32 Reserved; // 保留字，用于结构体手动对齐  
 LIC\_LIMITED licLimited; // 许可条款  
 uint8 Mac[32]; // 许可信息校验码，由硬件计算，硬件验证  
}

## 许可请求

structure **LIC\_REQ**  
{  
 uint32 Version; // 版本号  
 uint8 FartherLicID[16] // 父许可ID  
 uint8 OwnerUserID[16] // 许可所有者的用户ID  
 uint8 OwnerKeyFingerprint[32] // 被授权允许使用云端密钥的许可所有者的一个用户公钥的指纹  
 uint8 KeyID[16] // 被授权的云端密钥ID  
 uint32 Reserved; // 保留字，用于结构体手动对齐  
 int64 timeStamp; // 许可请求生成的时间生成时间，5分钟之后失效  
 LIC\_LIMITED licLimited; // 许可条款  
 uint8 Signature[256]; // 用户签名，SS端生成，硬件验证  
}

## S1密文数据包

structure **S1\_CIPHER**  
{  
 uint32 Version; // 版本号  
 uint32 Len // 密文长度  
 uint8 Cipher[256] // 密文内容  
}

## 密钥记录有效期设置

structure **KEY\_PERIOD**  
{  
 uint32 Version; // 版本号  
 uint8 KeyID[16]; // 密钥ID  
 uint32 Reserved; // 保留字，用于结构体手动对齐  
 int64 timeStamp; // 有效期数据生成时间，5分钟之后失效  
 int64 stTime; // 起始时间  
 int64 endTime; // 结束时间  
 uint8 Signaute[256]; // 用户签名，SS端生成，硬件验证  
}

## 板卡初始化请求包

struct **CHIP\_INIT\_REQ** {

uint32 Version; //版本号

uint8 chipId[16]; //板卡ID

uint32 Flag; //用于签名的密钥类型、算法

uint32 bits; //用于签名的密钥bit长度

uint8 Signaute[256]; //使用板卡的设备私钥对以上数据的签名（SHA256withRSA算法）

};

Flag定义同[1.1节](#_创建密钥记录请求)

## 板卡初始化命令包

struct CHIP\_INIT\_CMD\_INNER {

uint8 chipId[16]; //板卡ID

uint8 Kseed[32]; //用于生成Kenc和Kmac的种子码

uint32 Flag; //用于签名的密钥类型、算法

uint32 bits; //用于签名的密钥bit长度

uint8 Signaute[256]; //使管理员锁的设备私钥对以上数据的签名（SHA256withRSA算法）

};

struct CHIP\_INIT\_CMD {

uint32 Version; //版本号

uint32 Flag; //用于加密会话密钥的密钥类型、算法

uint32 bits; //用于加密会话密钥的密钥bit长度

uint32 sessionKeyFlag; //会话密钥类型、算法

uint32 sessionKeyBits; //会话密钥bit长度

uint8 sessionKeyCipher[256]; //会话密钥密文，使用板卡的设备证书加密，此处会话密钥的明文为[IV+KEY],其中IV长度固定16字节,KEY长度为（sessionKeyBits/8）,下面是加密CHIP\_INIT\_CMD\_INNER的算法为AES256\_CBC

uint32 cmdCipherLen; // CHIP\_INIT\_CMD\_INNER的密文长度

uint8 cmdCipher[512]; //CHIP\_INIT\_CMD\_INNER的密文，使用会话密钥加密

};

Flag、sessionKeyFlag定义同[1.1节](#_创建密钥记录请求)

## 板卡认证管理员锁数据包

struct AUTH\_ADM\_KEY\_INNER {

uint8 rand[32]; //随机数

uint8 Mac[32]; //使用Kmac密钥对以上字段计算的MAC值

};

struct AUTH\_ADM\_KEY {

uint32 Version; //版本号

uint32 Flag; // 密钥类型，算法  
 uint32 bits; // 密钥bit长度

uint32 cipherLen; //密文长度

uint8 cipher[128]; //使用Kenc对AUTH\_ADM\_KEY\_INNER加密得到的密文（AES256\_ECB算法）

};

Flag定义同[1.1节](#_创建密钥记录请求)

## 管理员锁复制请求包

struct ADMIN\_KEY\_COPY\_REQ {

uint32 Version; //版本号

uint8 rand[32]; //随机数用于生成会话密钥

uint8 hash[32]; //对(rand | 固定种子1 | 固定种子2)的SHA256值

uint32 Flag; //用于签名的密钥类型、算法

uint32 bits; //用于签名的密钥bit长度

uint8 Signaute[256]; //使用空白管理员锁的设备私钥对以上数据的签名（SHA256withRSA算法）

};

## 管理锁复制命令包

struct ADMIN\_KEY\_COPY\_CMD\_INNER {

uint8 Kseed[32]; //用于生成Kenc和Kmac的种子码

uint32 Flag; //用于签名的密钥类型、算法

uint32 bits; //用于签名的密钥bit长度

uint8 Signaute[256]; //使就绪管理员锁的设备私钥对以上数据的签名（SHA256withRSA算法）

};

struct ADMIN\_KEY\_COPY\_CMD {

uint32 Version; //版本号

uint32 Flag; //用于加密会话密钥的密钥类型、算法

uint32 bits; //用于加密会话密钥的密钥bit长度

uint32 sessionKeyFlag; //会话密钥类型、算法

uint32 sessionKeyBits; //会话密钥bit长度

uint8 sessionKeyCipher[256]; //会话密钥密文，使用板卡的设备证书加密，此处会话密钥的明文为[IV+KEY],其中IV长度固定16字节,KEY长度为（sessionKeyBits/8）,下面是加密ADMIN\_KEY\_COPY\_CMD\_INNER的算法为AES256\_CBC

uint32 cmdCipherLen; //ADMIN\_KEY\_COPY\_CMD\_INNER的密文长度

uint8 cmdCipher[512]; //ADMIN\_KEY\_COPY\_CMD\_INNER的密文，使用会话密钥加密

};

# 业务接口定义

## 签发用户公钥

int SignUserPubKey((IN)**USER\_PUB\_KEY** userkey, (OUT)**USER\_PUB\_KEY** userkey)

功能：签发用户公钥

说明：输入的userkey结构体中，除Mac之外其他的数据均已经ready，本接口计算该userkey的Mac值，并填到该结构体的Mac成员位置，返回这个userkey结构体。

MAC计算方式对userkey除Mac部分，计算AES\_MAC

## 生成云端密钥

int GenerateKeyCloud((IN)**KEY\_REC\_REQ** req, **(IN)USER\_PUB\_KEY userkey, (OUT)KEY\_REC** key)

功能：生成云端密钥

说明：根据输入的请求创建一个云端密钥，返回云端密钥结构体。验证userkey的Mac值，验证req和userkey中的成员OwnerUserID的值相等，验证req中的OwnerKeyFingerprint成员的值等于userkey结构体的hash值（SHA256）

使用userkey验证req中的用户签名。检查Req中的时间戳，确认req在有效期范围内。

根据输入的req中的成员值，设置结构体Key中的对应值，随机生成一个req成员flag指定的算法的密钥，并使用加密卡内预设的AES密钥加密，把密文保存在Key的成员Key\_C\_enc中，并计算Key中除Mac成员以外的所有成员的AES\_MAC值填入到key的Mac成员中。密钥记录的起始时间和结束时间设置成req中指定的起始时间和结束时间。

## 设置云端密钥有效期

int SetKeyCloudPeriod((IN) **KEY\_REC** key, (IN)**USER\_PUB\_KEY** userkey, (IN)**KEY\_PERIOD** KeyPeriod, (OUT)**KEY\_REC** key)

功能：设置云端密钥的有效期

说明：根据用户签发的密钥有效期设置数据，设置云端密钥的有效期。验证Key和userkey的Mac值，验证key和userkey中的成员OwnerUserID的值相等，验证key中的OwnerKeyFingerprint成员的值等于userkey结构体的hash值（SHA256），检查KeyPeriod中的timeStamp参数，确保KeyPeriod结构体是有效的。检查KeyPeriod中的KeyID和key中KeyID相等。

使用userkey验证KeyPeriod中的用户签名。

设置Key根据输入的period中的stTime和endTime成员，设置Key中的stTime和endTime成员，重新计算Key的Mac值。

## 生成密钥种子S1

int GenerateS1((IN)**KEY\_REC** key, (IN)**USER\_PUB\_KEY** userkey, (IN) **LICENSE** Lic, (OUT) S1**\_CIPHER** S1\_E\_Kc, (OUT) **S1\_CIPHER** S1\_E\_Ku, (OUT) **LICENSE** Lic\_new)

功能：生成S1

说明：验证Key和userkey的Mac值，检查Key中的stTime和endTime确保Key处于有效期内。**Lic不能为0。**验证Lic的Mac值，验证Lic中的KeyID和key中KeyID相等，验证Lic的成员OwnerUserID和userkey中的成员OwnerUserID的值相等，验证Lic中的OwnerKeyFingerprint成员的值等于userkey结构体的hash值（SHA256）。检查lic->licLimited看许可是否有效，是否有加密权限，如果许可过期或者失效，报错。如果有次数限制需要减掉一次，如果是时间段授权并且第一次使用该授权，需要设置授权的失效时间，**检查lic->licLimited中的Policy成员是否支持加密，如果不支持，报错。**重新结算lic的Mac值，随机生成S1，并使用输入的key和userkey分别对S1加密后生成S1\_E\_Ku和S1\_E\_Kc，返回S1\_E\_Kc，S1\_E\_Ku和新的许可结构体lic\_new。注：使用key对S1加密，必须先使用加密卡内预设的AES密钥对key中的成员Key\_C\_enc解密，得到密钥值，然后使用key结构体中的flag中指定的算法对S1加密

## 签发许可

int issueLicense((IN)**KEY\_REC** key, (IN)**USER\_PUB\_KEY** userkey, (IN)**BYTE[16]** LicID, (IN)**LICENSE** fartherLic, (IN)**LIC\_REQ** LicReq, (OUT) **LICENSE** Lic)

功能：签发许可

说明：验证Key和userkey的Mac值，

如果fartherLic为0，验证key和userkey中的成员OwnerUserID的值相等，验证key中的OwnerKeyFingerprint成员的值等于userkey结构体的hash值（SHA256），

如果fartherLic不为0，验证fartherLic的Mac值，验证fartherLic中的KeyID和key中KeyID相等，验证fartherLic的成员OwnerUserID和userkey中的成员OwnerUserID的值相等，验证fartherLic中的UserKeyFingerprint成员的值等于userkey结构体的hash值（SHA256），

验证使用userkey验证LicReq中的用户签名。验证LicReq中的KeyID和key中KeyID相等，验证LicReq中FartherLicID和fartherLic中LicID成员相等，检查LicReq中的timeStamp参数，确保LicReq结构体是有效的。

如果fartherLic为0，根据输入的licreq构建一个license，并计算License结构的Mac值返回新构建的license结构体。其中License中IssuerUserID设置为Key中的OwnerUserID，License中的FartherLicID设置为0

如果fartherLic不为0，检查fartherLic中的policy字段，看是否支持继承，如果不支持继承，则报错，如果支持继承，则检查licreq中的条款是否超出fartherLic中的条款限制，如果超出报错，构建license，并计算Mac返回新构建的license结构体，其中License中IssuerUserID设置为fartherLic中OwnerUserID。

## 转换密文

convertCipher(IN)**KEY\_REC** key, (IN)**USER\_PUB\_KEY** userkey, (IN) **LICENSE** Lic, (IN) **S1\_CIPHER** S1\_E\_Kc, (OUT) **S1\_CIPHER** S1\_E\_Ku, (OUT) **LICENSE** Lic\_new)

功能：转换密文，把被key加密的S1，转换成被userkey加密的S1

说明：验证Key和userkey的Mac值。

**Lic不能为0。**

验证Lic的Mac值，验证Lic中的KeyID和key中KeyID相等，验证Lic的成员OwnerUserID和userkey中的成员OwnerUserID的值相等，验证Lic中的UserKeyFingerprint成员的值等于userkey结构体的hash值（SHA256）。检查lic->licLimited看许可是否有效，是否有解密权限，如果许可过期或者失效，报错。**检查lic->licLimited中的Policy成员是否支持解密，如果不支持，报错。**检查Key中的stTime和endTime确保Key处于有效期内。使用key解密输入的S1\_E\_Kc得到S1，然后使用userkey加密S1得到S1\_E\_Ku。查看Lic的条款，如果有次数限制需要减掉一次，如果是时间段授权并且第一次使用该授权，需要设置授权的失效时间，重新结算lic的Mac值，返回S1\_E\_Ku和新的许可结构体lic\_new

# 管理功能流程概述

本章节主要通过流程通概述管理功能的几个主要流程，更多细节在【[管理接口定义](#_管理接口定义)】章节中描述

## 板卡初始化



## 管理员锁复制



# 管理接口定义

板卡在生产结束后，其内部应有以下数据：深思根证书、卡内生成的设备密钥对（其中私钥不可导出，只能内部使用）、DeviceCA证书、向DeviceCA申请的设备证书。

管理员锁在生产结束后，其内部应有以下数据：深思根证书、锁内生成的设备密钥对（其中私钥不可导出，只能内部使用）、DeviceCA证书、向DeviceCA申请的设备证书。

## 一组获取板卡基本信息的接口

* 板卡状态

#define CHIP\_STATUS\_BLANK 0x00000000

#define CHIP\_STATUS\_ALREADY 0x00000001

* 板卡ID

如有其它需要再补充

## 初始化管理员锁（只有第一把管理员锁需要本章节的流程）

### 初始化

int init();

功能：初始化管理员锁，主要就是生成种子和密钥

说明：

1. 检查锁的状态是否为空白状态，如果否则报错退出流程
2. 随机生成内部密钥1，并保存
3. 随机生成Kseed，并使用内部密钥1加密保存(AES256\_ECB)
4. 使用Kseed生成并保存Kenc和Kmac,生成方法如下：

Kenc = SHA512(Kseed | 固定种子1)

Kmac = SHA512(Kseed | 固定种子2)

根据所需密钥长度，取哈希结果前N个字节作为密钥

其中固定种子1和固定种子2，板卡和管理员锁中必须设置为一致

其中Kenc为用于加密的密钥；Kmac为用于计算MAC的密钥

1. 设置管理员锁为已就绪

## 板卡初始化

### 从板卡获取初始化请求包

int getInitReq(

(OUT)CHIP\_INIT\_REQ \*req,

(OUT)uint8 \*ca\_cert,

(OUT)uint32 \*ca\_cert\_len,

(OUT)uint8 \*cert,

(OUT)uint32 \*cert\_len);

功能：获取板卡初始化请求包，同时获取板卡的设备证书和DeviceCA证书

说明：

1. 板卡判断当前状态，如果当前已是就绪状态，则返回错误，退出流程
2. 按照数据结构定义组装板卡初始化请求包
3. 使用板卡设备私钥对请求包签名(SHA256withRSA)，并将签名填充到请求包的签名字段
4. 读取DeviceCA证书和板卡设备证书
5. 返回板卡初始化请求包、DeviceCA证书、板卡设备证书

### 就绪管理员锁校验证书

int checkCert(

(OUT)uint8 cert\_type,

(OUT)uint8 \*cert,

(OUT)uint32 \*cert\_len);

功能：校验证书

说明：

管理员锁依次校验从板卡中获得的DeviceCA证书、板卡设备证书

用预设的根证书校验DeviceCA证书通过后，将其缓存在内存中（ses库有现成的验证子CA证书的接口）

用内存缓存中的DeviceCA证书校验板卡设备证书通过后，将其缓存在内存中，用于后续校验初始化请求包中的签名（ses库有现成的验证terminal证书的接口）

### 从就绪管理员锁获取初始化命令包

int getInitCMD(

(IN)CHIP\_INIT\_REQ req,

(OUT)CHIP\_INIT\_CMD cmd);

功能：从管理员锁中获取板卡初始化命令包

说明：

1. 使用板卡设备证书验证请求包中的签名
2. 按照数据结构定义构造CHIP\_INIT\_CMD\_INNER数据包，使用管理员锁中的设备私钥签名并把签名填充到数据包的签名字段（调用ses库函数签名）
3. 随机生成一个会话密钥（IV+KEY），使用会话密钥加密CHIP\_INIT\_CMD\_INNER数据包(AES256\_CBC)
4. 使用板卡设备证书加密会话密钥(RSA2048)
5. 按照数据结构定义构造CHIP\_INIT\_CMD数据包
6. 返回命令包、管理员锁中的软件证书和SoftwareCA证书

### 读取管理员锁内的证书

int getCert(

(IN)uint8 cert\_type,

(OUT)uint8 \*cert,

(OUT)uint32 cert\_len);

功能：调用已有的H5API从锁中读取证书（可能与此处描述有差异）

说明：

可以从锁内读取DeviceCA证书、锁的设备证书

### 板卡执行初始化命令包

int init(

(IN)CHIP\_INIT\_CMD cmd,

(IN)uint8 \*ca\_cert,

(IN)uint32 ca\_cert\_len,

(IN)uint8 \*cert,

(IN)uint32 cert\_len);

功能：板卡执行初始化命令包

说明：

1. 使用板卡设备私钥解密命令包中的会话密钥(RSA2048),注意解密出来的密钥实际是IV+KEY,前16字节是IV
2. 使用会话密钥解密CHIP\_INIT\_CMD\_INNER密文包（AES256\_CBC）
3. 使用预设根证书验证传入的DeviceCA证书 --如何验证?
4. 使用DeviceCA证书验证传入的管理员锁设备证书 -如何验证？
5. 使用管理员锁设备证书验证CHIP\_INIT\_CMD\_INNER包中签名
6. 对比数据包中chipId是否与当前板卡的chipId一致
7. 使用数据包中的Kseed生成并保存Kenc和Kmac,生成方法如下：

Kenc = SHA256(Kseed | 固定种子1)

Kmac = SHA256(Kseed | 固定种子2)

根据所需密钥长度，取哈希结果前N个字节作为密钥

其中固定种子1和固定种子2，板卡和管理员锁中必须设置为一致

其中Kenc为业务接口中用于加密的密钥；Kmac为业务接口中用于计算MAC的密钥

1. 保存Kenc和Kmac（注意：不保存Kseed）
2. 将板卡状态设置为已就绪

## 管理员锁复制

### 从板卡获取认证管理员锁数据包

int getAuthPkg(

(OUT)AUTH\_ADM\_KEY \*pkg);

功能：从管理员锁获取板卡认证管理员锁数据包

说明：

1. 判断当前板卡是否已就绪，否则报错退出流程
2. 板卡内生成随机数
3. 使用Kmac对随机数计算MAC（AES128\_MAC算法），然后按数据结构定义组装AUTH\_ADM\_KEY\_INNER数据包
4. 使用Kenc对AUTH\_ADM\_KEY\_INNER数据包进行加密（AES256\_ECB算法）
5. 按照数据结构定义组装AUTH\_ADM\_KEY数据包，并返回

### 管理员锁认证认证数据包

int auth(

(IN)AUTH\_ADM\_KEY pkg);

功能：管理员锁认证从板卡获得的认证数据包

说明：

1. 使用Kenc解密密文字段（AES256\_ECB算法），得到AUTH\_ADM\_KEY\_INNER包
2. 使用Kmac对rand字段结算MAC（AES128\_MAC算法），并与数据包中的MAC字段对比，是否相同，如果不同返回认证失败，退出流程
3. 相同，则设置管理锁已认证标识为true，该标识保存在内存中，掉电即被重置

### 从空白管理员锁获取复制请求包

int getCopyReq(

(OUT) ADMIN\_KEY\_COPY\_REQ req);

功能：从空白管理员锁获取复制请求包

说明：

1. 检查当前管理员锁是否处于空白状态，否则报错退出流程
2. 随机生成rand
3. 按照数据结构定义计算（rand | 固定种子1 | 固定种子2）的SHA256哈希值
4. 使用管理员锁设备私钥对数据包签名（调用ses库函数签名）
5. 按照数据结构定义组装请求包并返回

### 从就绪管理员锁获取复制命令包

int getCopyCmd(

(IN) ADMIN\_KEY\_COPY\_REQ req,

(OUT) ADMIN\_KEY\_COPY\_CMD cmd);

功能：从就绪管理员锁获取复制命令包。在此之前要先校验空白管理员锁的DeviceCA证书、设备证书，并把设备证书缓存在内存中，参考“[就绪管理员锁校验证书](#_就绪管理员锁校验证书)”章节

说明：

1. 判断当前管理员锁是否是已就绪状态，否则报错退出流程
2. 判断当前管理员锁的认证标识是否为true，否则报错退出流程
3. 使用内存缓存中的空白管理员锁设备证书校验请求包中的签名
4. 按照数据结构定义验证请求包中的hash值
5. 使用内部密钥1解密Kseed（AES256\_ECB），并组装ADMIN\_KEY\_COPY\_CMD\_INNER包
6. 使用就绪管理员锁的设备私钥对ADMIN\_KEY\_COPY\_CMD\_INNER包签名（调用ses库函数签名）
7. 随机生成会话密钥（IV+KEY），并使用会话密钥加密ADMIN\_KEY\_COPY\_CMD\_INNER包（AES256\_CBC）
8. 使用空白管理员锁的设备证书加密会话密钥
9. 安装数据结构定义组装ADMIN\_KEY\_COPY\_CMD包并返回

### 空白管理员锁执行复制命令包

int execCopyCmd(

(IN) ADMIN\_KEY\_COPY\_CMD cmd);

功能：空白管理员锁执行复制命令包。在此之前要先校验就绪管理员锁的DeviceCA证书、设备证书，并把设备证书缓存在内存中，参考“[就绪管理员锁校验证书](#_就绪管理员锁校验证书)”章节

说明：

1. 使用空白管理员锁设备私钥解密会话密钥（IV+KEY）（调用ses库函数解密)
2. 使用会话密钥解密得到ADMIN\_KEY\_COPY\_CMD\_INNER数据包（AES256\_CBC）
3. 使用就绪管理员锁设备证书校验ADMIN\_KEY\_COPY\_CMD\_INNER数据包中的签名
4. 随机生成空白管理员锁的内部密钥1，并保存
5. 使用内部密钥1加密保存Kseed
6. 使用数据包中的Kseed生成并保存Kenc和Kmac,生成方法如下：

Kenc = SHA512(Kseed | 固定种子1)

Kmac = SHA512(Kseed | 固定种子2)

根据所需密钥长度，取哈希结果前N个字节作为密钥

其中固定种子1和固定种子2，板卡和管理员锁中必须设置为一致

其中Kenc为用于加密的密钥；Kmac为用于计算MAC的密钥

1. 设置管理员锁为已就绪

# 加密卡预设AES密钥加密云端密钥算法

**加密算法**

输入：云端密钥按64bit分组（P1，P2，……Pn）K\_c

加密卡内预设的AES密钥K

输出：加密后的K\_c（C0，C1，……Cn）注意输出比输入多一个分组

1、初始化

令A = IV(IV=\xA6\xA6\xA6\xA6\xA6\xA6\xA6\xA6)

for i= 1，…，n

Ri = Pi

2、运算

for j = 0，1，…，5

for i= 1，2， …，n

B = AESK(A||Ri)

A = MSB64(B) ^ (n\*j + i)

Ri = LSB64(B)

3、输出结果

C0 = A

for i = 1，…，n

Ci = Ri

**解密算法**

输入：加密后的K\_c（C0，C1，……Cn）

加密卡内预设的AES密钥K

输出：云端密钥按64bit分组（P1，P2，……Pn）K\_c, 注意输出比输入少一个分组

1、初始化

令A = C0

for i= 1，…，n

Ri = Ci

2、运算

for j = 5，…，0

for i= n，n-1， …，1

B = AES-1K((A^(n\*j + i))||Ri)

A = MSB64(B)

Ri = LSB64(B)

3、输出结果

if(A != IV)

ERROR

for i = 1，…，n

Pi = Ri

**验证数据**

K: 000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F

K\_c: 00112233445566778899AABBCCDDEEFF000102030405060708090A0B0C0D0E0F

加密验证数据表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step t** |  | **A** | **R1** | **R2** | **R3** | **R4** |
| 1 | Input | A6A6A6A6A6A6A6A6 | 0011223344556677 | 8899AABBCCDDEEFF | 0001020304050607 | 08090A0B0C0D0E0F |
| AES | 794314D454E3FDE1 | F661BD9F31FBFA31 | 8899AABBCCDDEEFF | 0001020304050607 | 08090A0B0C0D0E0F |
| Add t | 794314D454E3FDE0 | F661BD9F31FBFA31 | 8899AABBCCDDEEFF | 0001020304050607 | 08090A0B0C0D0E0F |
| 2 | Input | 794314D454E3FDE0 | F661BD9F31FBFA31 | 8899AABBCCDDEEFF | 0001020304050607 | 08090A0B0C0D0E0F |
| AES | D450EA5C5BBCB561 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 0001020304050607 | 08090A0B0C0D0E0F |
| Add t | D450EA5C5BBCB563 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 0001020304050607 | 08090A0B0C0D0E0F |
| 3 | Input | D450EA5C5BBCB563 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 0001020304050607 | 08090A0B0C0D0E0F |
| AES | 9DF8F5405FBC00C1 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | 08090A0B0C0D0E0F |
| Add t | 9DF8F5405FBC00C2 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | 08090A0B0C0D0E0F |
| 4 | Input | 9DF8F5405FBC00C2 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | 08090A0B0C0D0E0F |
| AES | 564408FDD0DD2EA4 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| Add t | 564408FDD0DD2EA0 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| 5 | Input | 564408FDD0DD2EA0 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| AES | 4EF02EDD3146AFBB | E7D1194D853E53F8 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| Add t | 4EF02EDD3146AFBE | E7D1194D853E53F8 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| 6 | Input | 4EF02EDD3146AFBE | E7D1194D853E53F8 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| AES | 963AAFFD96B223EC | E7D1194D853E53F8 | EFD48BA304945576 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| Add t | 963AAFFD96B223EA | E7D1194D853E53F8 | EFD48BA304945576 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| 7 | Input | 963AAFFD96B223EA | E7D1194D853E53F8 | EFD48BA304945576 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| AES | 66D7A8ADD086B9DD | E7D1194D853E53F8 | EFD48BA304945576 | C365B66943E2D760 | E5923CB9FDB56FBC |
| Add t | 66D7A8ADD086B9DA | E7D1194D853E53F8 | EFD48BA304945576 | C365B66943E2D760 | E5923CB9FDB56FBC |
| 8 | Input | 66D7A8ADD086B9DA | E7D1194D853E53F8 | EFD48BA304945576 | C365B66943E2D760 | E5923CB9FDB56FBC |
| AES encrypt | C58B9D3AC6D5B94E | E7D1194D853E53F8 | EFD48BA304945576 | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| Add t | C58B9D3AC6D5B946 | E7D1194D853E53F8 | EFD48BA304945576 | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| 9 | Input | C58B9D3AC6D5B946 | E7D1194D853E53F8 | EFD48BA304945576 | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| AES | 1A681354E84C41F8 | D6AE29ECE7192D43 | EFD48BA304945576 | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| Add t | 1A681354E84C41F1 | D6AE29ECE7192D43 | EFD48BA304945576 | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| 10 | Input | 1A681354E84C41F1 | D6AE29ECE7192D43 | EFD48BA304945576 | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| AES | DBA417FB51F9E3CB | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| Add t | DBA417FB51F9E3C1 | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| 11 | Input | DBA417FB51F9E3C1 | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| AES | 0629EB29A42E4FD9 | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | F56701DAF0388216 | 73E3B6CBE5D05D74 |
| Add t | 0629EB29A42E4FD2 | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | F56701DAF0388216 | 73E3B6CBE5D05D74 |
| 12 | Input | 0629EB29A42E4FD2 | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | F56701DAF0388216 | 73E3B6CBE5D05D74 |
| AES | F9ED8A1429515665 | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | F56701DAF0388216 | 3CF149E90E8C04D9 |
| Add t | F9ED8A1429515669 | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | F56701DAF0388216 | 3CF149E90E8C04D9 |
| 13 | Input | F9ED8A1429515669 | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | F56701DAF0388216 | 3CF149E90E8C04D9 |
| AES | 2E8E2B6BB2016696 | 4745856AF333F01F | FBEC169FA5C0F6BA | F56701DAF0388216 | 3CF149E90E8C04D9 |
| Add t | 2E8E2B6BB201669B | 4745856AF333F01F | FBEC169FA5C0F6BA | F56701DAF0388216 | 3CF149E90E8C04D9 |
| 14 | Input | 2E8E2B6BB201669B | 4745856AF333F01F | FBEC169FA5C0F6BA | F56701DAF0388216 | 3CF149E90E8C04D9 |
| AES | 15342443CB95ADB1 | 4745856AF333F01F | BCA418BBF7DCE60B | F56701DAF0388216 | 3CF149E90E8C04D9 |
| Add t | 15342443CB95ADBF | 4745856AF333F01F | BCA418BBF7DCE60B | F56701DAF0388216 | 3CF149E90E8C04D9 |
| 15 | Input | 15342443CB95ADBF | 4745856AF333F01F | BCA418BBF7DCE60B | F56701DAF0388216 | 3CF149E90E8C04D9 |
| AES | 33FE29365885C4B7 | 4745856AF333F01F | BCA418BBF7DCE60B | C272E9466AAE98F9 | 3CF149E90E8C04D9 |
| Add t | 33FE29365885C4B8 | 4745856AF333F01F | BCA418BBF7DCE60B | C272E9466AAE98F9 | 3CF149E90E8C04D9 |
| 16 | Input | 33FE29365885C4B8 | 4745856AF333F01F | BCA418BBF7DCE60B | C272E9466AAE98F9 | 3CF149E90E8C04D9 |
| AES | 5075496800978B4A | 4745856AF333F01F | BCA418BBF7DCE60B | C272E9466AAE98F9 | 40F68C91DB49702C |
| Add t | 5075496800978B5A | 4745856AF333F01F | BCA418BBF7DCE60B | C272E9466AAE98F9 | 40F68C91DB49702C |
| 17 | Input | 5075496800978B5A | 4745856AF333F01F | BCA418BBF7DCE60B | C272E9466AAE98F9 | 40F68C91DB49702C |
| AES | A5382A26B47551F1 | 1BB8C765A84195E7 | BCA418BBF7DCE60B | C272E9466AAE98F9 | 40F68C91DB49702C |
| Add t | A5382A26B47551E0 | 1BB8C765A84195E7 | BCA418BBF7DCE60B | C272E9466AAE98F9 | 40F68C91DB49702C |
| 18 | Input | A5382A26B47551E0 | 1BB8C765A84195E7 | BCA418BBF7DCE60B | C272E9466AAE98F9 | 40F68C91DB49702C |
| AES | F19D80D437EFE8F9 | 1BB8C765A84195E7 | F7EDAD518C960D36 | C272E9466AAE98F9 | 40F68C91DB49702C |
| Add t | F19D80D437EFE8EB | 1BB8C765A84195E7 | F7EDAD518C960D36 | C272E9466AAE98F9 | 40F68C91DB49702C |
| 19 | Input | F19D80D437EFE8EB | 1BB8C765A84195E7 | F7EDAD518C960D36 | C272E9466AAE98F9 | 40F68C91DB49702C |
| AES encrypt | B422B444B87A190B | 1BB8C765A84195E7 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 40F68C91DB49702C |
| Add t | B422B444B87A1918 | 1BB8C765A84195E7 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 40F68C91DB49702C |
| 20 | Input | B422B444B87A1918 | 1BB8C765A84195E7 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 40F68C91DB49702C |
| AES | D058823360F88A37 | 1BB8C765A84195E7 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| Add t | D058823360F88A23 | 1BB8C765A84195E7 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| 21 | Input | D058823360F88A23 | 1BB8C765A84195E7 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| AES | C89A96CA7B163ECC | CBCCB35CFB87F826 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| Add t | C89A96CA7B163ED9 | CBCCB35CFB87F826 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| 22 | Input | C89A96CA7B163ED9 | CBCCB35CFB87F826 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| AES | 39D02FE7435870ED | CBCCB35CFB87F826 | 3F5786E2D80ED326 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| Add t | 39D02FE7435870FB | CBCCB35CFB87F826 | 3F5786E2D80ED326 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| 23 | Input | 39D02FE7435870FB | CBCCB35CFB87F826 | 3F5786E2D80ED326 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| AES | 0AEB82AE3146A91B | CBCCB35CFB87F826 | 3F5786E2D80ED326 | CBC7F0E71A99F43B | 07DFE775B9687E73 |
| Add t | 0AEB82AE3146A90C | CBCCB35CFB87F826 | 3F5786E2D80ED326 | CBC7F0E71A99F43B | 07DFE775B9687E73 |
| 24 | Input | 0AEB82AE3146A90C | CBCCB35CFB87F826 | 3F5786E2D80ED326 | CBC7F0E71A99F43B | 07DFE775B9687E73 |
| AES | 28C9F404C4B810EC | CBCCB35CFB87F826 | 3F5786E2D80ED326 | CBC7F0E71A99F43B | FB988B9B7A02DD21 |
| Add t | 28C9F404C4B810F4 | CBCCB35CFB87F826 | 3F5786E2D80ED326 | CBC7F0E71A99F43B | FB988B9B7A02DD21 |
|  | Ciphertext | 28C9F404C4B810F4 | CBCCB35CFB87F826 | 3F5786E2D80ED326 | CBC7F0E71A99F43B | FB988B9B7A02DD21 |

解密验证数据表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step t** |  | **A** | **R1** | **R2** | **R3** | **R4** |
| 24 | Input | 28C9F404C4B810F4 | CBCCB35CFB87F826 | 3F5786E2D80ED326 | CBC7F0E71A99F43B | FB988B9B7A02DD21 |
| Add t | 28C9F404C4B810EC | CBCCB35CFB87F826 | 3F5786E2D80ED326 | CBC7F0E71A99F43B | FB988B9B7A02DD21 |
| AES | 0AEB82AE3146A90C | CBCCB35CFB87F826 | 3F5786E2D80ED326 | CBC7F0E71A99F43B | 07DFE775B9687E73 |
| 23 | Input | 0AEB82AE3146A90C | CBCCB35CFB87F826 | 3F5786E2D80ED326 | CBC7F0E71A99F43B | 07DFE775B9687E73 |
| Add t | 0AEB82AE3146A91B | CBCCB35CFB87F826 | 3F5786E2D80ED326 | CBC7F0E71A99F43B | 07DFE775B9687E73 |
| AES | 39D02FE7435870FB | CBCCB35CFB87F826 | 3F5786E2D80ED326 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| 22 | Input | 39D02FE7435870FB | CBCCB35CFB87F826 | 3F5786E2D80ED326 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| Add t | 39D02FE7435870ED | CBCCB35CFB87F826 | 3F5786E2D80ED326 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| AES | C89A96CA7B163ED9 | CBCCB35CFB87F826 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| 21 | Input | C89A96CA7B163ED9 | CBCCB35CFB87F826 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| Add t | C89A96CA7B163ECC | CBCCB35CFB87F826 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| AES | D058823360F88A23 | 1BB8C765A84195E7 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| 20 | Input | D058823360F88A23 | 1BB8C765A84195E7 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| Add t | D058823360F88A37 | 1BB8C765A84195E7 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 07DFE775B9687E73 |
| AES | B422B444B87A1918 | 1BB8C765A84195E7 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 40F68C91DB49702C |
| 19 | Input | B422B444B87A1918 | 1BB8C765A84195E7 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 40F68C91DB49702C |
| Add t | B422B444B87A190B | 1BB8C765A84195E7 | F7EDAD518C960D36 | 1CFBF6B4C24CB982 | 40F68C91DB49702C |
| AES | F19D80D437EFE8EB | 1BB8C765A84195E7 | F7EDAD518C960D36 | C272E9466AAE98F9 | 40F68C91DB49702C |
| 18 | Input | F19D80D437EFE8EB | 1BB8C765A84195E7 | F7EDAD518C960D36 | C272E9466AAE98F9 | 40F68C91DB49702C |
| Add t | F19D80D437EFE8F9 | 1BB8C765A84195E7 | F7EDAD518C960D36 | C272E9466AAE98F9 | 40F68C91DB49702C |
| AES | A5382A26B47551E0 | 1BB8C765A84195E7 | BCA418BBF7DCE60B | C272E9466AAE98F9 | 40F68C91DB49702C |
| 17 | Input | A5382A26B47551E0 | 1BB8C765A84195E7 | BCA418BBF7DCE60B | C272E9466AAE98F9 | 40F68C91DB49702C |
| Add t | A5382A26B47551F1 | 1BB8C765A84195E7 | BCA418BBF7DCE60B | C272E9466AAE98F9 | 40F68C91DB49702C |
| AES | 5075496800978B5A | 4745856AF333F01F | BCA418BBF7DCE60B | C272E9466AAE98F9 | 40F68C91DB49702C |
| 16 | Input | 5075496800978B5A | 4745856AF333F01F | BCA418BBF7DCE60B | C272E9466AAE98F9 | 40F68C91DB49702C |
| Add t | 5075496800978B4A | 4745856AF333F01F | BCA418BBF7DCE60B | C272E9466AAE98F9 | 40F68C91DB49702C |
| AES | 33FE29365885C4B8 | 4745856AF333F01F | BCA418BBF7DCE60B | C272E9466AAE98F9 | 3CF149E90E8C04D9 |
| 15 | Input | 33FE29365885C4B8 | 4745856AF333F01F | BCA418BBF7DCE60B | C272E9466AAE98F9 | 3CF149E90E8C04D9 |
| Add t | 33FE29365885C4B7 | 4745856AF333F01F | BCA418BBF7DCE60B | C272E9466AAE98F9 | 3CF149E90E8C04D9 |
| AES | 15342443CB95ADBF | 4745856AF333F01F | BCA418BBF7DCE60B | F56701DAF0388216 | 3CF149E90E8C04D9 |
| 14 | Input | 15342443CB95ADBF | 4745856AF333F01F | BCA418BBF7DCE60B | F56701DAF0388216 | 3CF149E90E8C04D9 |
| Add t | 15342443CB95ADB1 | 4745856AF333F01F | BCA418BBF7DCE60B | F56701DAF0388216 | 3CF149E90E8C04D9 |
| AES | 2E8E2B6BB201669B | 4745856AF333F01F | FBEC169FA5C0F6BA | F56701DAF0388216 | 3CF149E90E8C04D9 |
| 13 | Input | 2E8E2B6BB201669B | 4745856AF333F01F | FBEC169FA5C0F6BA | F56701DAF0388216 | 3CF149E90E8C04D9 |
| Add t | 2E8E2B6BB2016696 | 4745856AF333F01F | FBEC169FA5C0F6BA | F56701DAF0388216 | 3CF149E90E8C04D9 |
| AES | F9ED8A1429515669 | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | F56701DAF0388216 | 3CF149E90E8C04D9 |
| 12 | Input | F9ED8A1429515669 | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | F56701DAF0388216 | 3CF149E90E8C04D9 |
| Add t | F9ED8A1429515665 | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | F56701DAF0388216 | 3CF149E90E8C04D9 |
| AES | 0629EB29A42E4FD2 | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | F56701DAF0388216 | 73E3B6CBE5D05D74 |
| 11 | Input | 0629EB29A42E4FD2 | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | F56701DAF0388216 | 73E3B6CBE5D05D74 |
| Add t | 0629EB29A42E4FD9 | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | F56701DAF0388216 | 73E3B6CBE5D05D74 |
| AES | DBA417FB51F9E3C1 | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| 10 | Input | DBA417FB51F9E3C1 | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| Add t | DBA417FB51F9E3CB | D6AE29ECE7192D43 | FBEC169FA5C0F6BA | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| AES | 1A681354E84C41F1 | D6AE29ECE7192D43 | EFD48BA304945576 | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| 9 | Input | 1A681354E84C41F1 | D6AE29ECE7192D43 | EFD48BA304945576 | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| Add t | 1A681354E84C41F8 | D6AE29ECE7192D43 | EFD48BA304945576 | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| AES | C58B9D3AC6D5B946 | E7D1194D853E53F8 | EFD48BA304945576 | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| 8 | Input | C58B9D3AC6D5B946 | E7D1194D853E53F8 | EFD48BA304945576 | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| Add t | C58B9D3AC6D5B94E | E7D1194D853E53F8 | EFD48BA304945576 | C365B66943E2D760 | 73E3B6CBE5D05D74 |
| AES | 66D7A8ADD086B9DA | E7D1194D853E53F8 | EFD48BA304945576 | C365B66943E2D760 | E5923CB9FDB56FBC |
| 7 | Input | 66D7A8ADD086B9DA | E7D1194D853E53F8 | EFD48BA304945576 | C365B66943E2D760 | E5923CB9FDB56FBC |
| Add t | 66D7A8ADD086B9DD | E7D1194D853E53F8 | EFD48BA304945576 | C365B66943E2D760 | E5923CB9FDB56FBC |
| AES | 963AAFFD96B223EA | E7D1194D853E53F8 | EFD48BA304945576 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| 6 | Input | 963AAFFD96B223EA | E7D1194D853E53F8 | EFD48BA304945576 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| Add t | 963AAFFD96B223EC | E7D1194D853E53F8 | EFD48BA304945576 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| AES | 4EF02EDD3146AFBE | E7D1194D853E53F8 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| 5 | Input | 4EF02EDD3146AFBE | E7D1194D853E53F8 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| Add t | 4EF02EDD3146AFBB | E7D1194D853E53F8 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| AES | 564408FDD0DD2EA0 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| 4 | Input | 564408FDD0DD2EA0 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| Add t | 564408FDD0DD2EA4 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | E5923CB9FDB56FBC |
| AES | 9DF8F5405FBC00C2 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | 08090A0B0C0D0E0F |
| 3 | Input | 9DF8F5405FBC00C2 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | 08090A0B0C0D0E0F |
| Add t | 9DF8F5405FBC00C1 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 6CA405593A3B5154 | 08090A0B0C0D0E0F |
| AES | D450EA5C5BBCB563 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 0001020304050607 | 08090A0B0C0D0E0F |
| 2 | Input | D450EA5C5BBCB563 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 0001020304050607 | 08090A0B0C0D0E0F |
| Add t | D450EA5C5BBCB561 | F661BD9F31FBFA31 | F60E0CDB7F429FE8 | 0001020304050607 | 08090A0B0C0D0E0F |
| AES | 794314D454E3FDE0 | F661BD9F31FBFA31 | 8899AABBCCDDEEFF | 0001020304050607 | 08090A0B0C0D0E0F |
| 1 | Input | 794314D454E3FDE0 | F661BD9F31FBFA31 | 8899AABBCCDDEEFF | 0001020304050607 | 08090A0B0C0D0E0F |
| Add t | 794314D454E3FDE1 | F661BD9F31FBFA31 | 8899AABBCCDDEEFF | 0001020304050607 | 08090A0B0C0D0E0F |
| AES | A6A6A6A6A6A6A6A6 | 0011223344556677 | 8899AABBCCDDEEFF | 0001020304050607 | 08090A0B0C0D0E0F |
|  | Plaintext | A6A6A6A6A6A6A6A6 | 0011223344556677 | 8899AABBCCDDEEFF | 0001020304050607 | 08090A0B0C0D0E0F |