视频密钥管理服务器

应用接口说明书

视频图像信息智能分析与共享应用技术国家工程实验室

北京中盾安全技术开发公司

2019 年 9 月

版权声明

免责声明

本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。若因本文档或其所提到的任何信息引起的直接或间接的资料流失、利益损失，视频国家实验室及其员工均不承担任何责任。

保密声明

相关授权人员自收到本文档之日起，需对文档内容进行保密，未经视频国家实验室许可，不得传播本文档的内容。

目 录

1 产品概述

1.1 接口开发环境说明

设计中本接口的开发及使用环境如下：

- 使用 C 语言进行开发
- 开发环境预计使用 RedHat 5 64 位环境进行开发编译
- 预期部署、运行环境为 Linux 类操作系统

1.2 接口主要功能说明

密钥管理服务器应用开发接口功能包括：

- 查询设备
- 注册 IPC 设备
- 注销 IPC 设备
- IPC 设备签到（为 IPC 设备生成 VKEK 并导出）
- NVR 设备签到（为 NVR 设备生成 VKEK 并导出）
- NVR 记录 IPC 的签到记录
- 根据时间区间查询历史 VKEK
- 根据版本号查询历史 VKEK
- 根据时间区间批量导出历史 VKEK
- 根据版本号导出历史 VKEK
- 视频数据解密
- 生成信令校验数据
- 校验信令校验数据
- 导入 SM2 公钥加密的 SIP 临时 VKEK

1.3 约定

1．设备编号——限定所有设备编号必须为长度为 20 位的数字。

2．时间信息——按照 GB 35114 规定。

3．VKEK 密钥版本号——按照 GB 35114 规定。

4．VKEK 的密钥版本号——整数形式。

5．数据加密时的数据填充——使用 PKCS#5 描述的填充方式（强制在数据末尾填充缺少的字节数值，即最少填充 1 个 0x01 最多填充 16 个 0x10）。

2 C 语言接口详细描述

2.1数据结构

2.1.1 宏定义

2.1.1.1 数据长度宏定义

宏描述	预定义值	说明
#define LENGTH_DEVICEID	32	设备编号缓冲区长度
#define LENGTH_TIMER	19	时间信息缓冲区长度
#define LENGTH_KCV	16	密钥校验值缓冲区长度
#define LENGTH_KVERSION	20	密钥版本号缓冲区长度
#define LENGTH_KCIPHER	128	密钥密文缓冲区长度

2.1.1.2 密码算法定义

宏描述	预定义值	说明
#define VGD_SM1_ECB	0x00000101	SM1 算法 ECB 运算模式
#define VGD_SM1_CBC	0x00000102	SM1 算法 CBC 运算模式
#define VGD_SM1_OFB	0x00000108	SM1 算法 OFB 运算模式
#define VGD_SM4_ECB	0x00000401	SM4 算法 ECB 运算模式
#define VGD_SM4_CBC	0x00000402	SM4 算法 CBC 运算模式
#define VGD_SM4_OFB	0x00000408	SM4 算法 OFB 运算模式

2.1.1.3 函数返回代码定义

宏描述	预定义值	说明
#define VKR_OK	0x00000000	执行成功，校验正确
#define VKR_VERIFY	0x00000001	校验失败
#define VKR_LIB_BASE	0x00010000	接口内返回代码基础值
#define VKR_EXTERN_BASE	0x00020000	外部功能错误（回调函数出错）
#define VKR_UNKNOWN	VKR_LIB_BASE+0x0001	未知错误
#define VKR_NOTINITIALIZE	VKR_LIB_BASE+0x0003	未调用初始化
#define VKR_INITIALIZED	VKR_LIB_BASE+0x0004	重复初始化
#define VKR_INITIALIZE_FAILED	VKR_LIB_BASE+0x0005	初始化失败
#define VKR_SESSION_OPEN	VKR_LIB_BASE+0x0006	打开会话失败
#define VKR_SESSION_CLOSED	VKR_LIB_BASE+0x0007	会话异常

#define VKR_SESSION_MUCH	VKR_LIB_BASE+0x0008	会话打开太多
#define VKR_SESSION_INVALID	VKR_LIB_BASE+0x0009	会话无效
#define VKR_MEMORY	VKR_LIB_BASE+0x0010	缓存区不足
#define VKR_ARGLENGTH	VKR_LIB_BASE+0x0011	输入参数长度无效
#define VKR_ARG_NULL	VKR_LIB_BASE+0x0012	参数为 NULL
#define VKR_ARG_INVALID	VKR_LIB_BASE+0x0013	参数取值无效
#define VKR_ALG_INVALID	VKR_LIB_BASE+0x0014	算法不支持
#define VKR_BASE64_ENCODE	VKR_LIB_BASE+0x0021	Base64 编码失败
#define VKR_BASE64_DECODE	VKR_LIB_BASE+0x0022	Base64 解码失败
#define VKR_BCD_ENCODE	VKR_LIB_BASE+0x0023	BCD 编码失败
#define VKR_BCD_DECODE	VKR_LIB_BASE+0x0024	BCD 解码失败
#define VKR_NVRID	VKR_LIB_BASE+0x0031	NVR 设备编号无效或与 IPC 设备不匹配

2.1.1.4 密钥管理服务器返回代码说明

十进制值	十六进制值	说明
0	0x00000000	执行成功，校验正确
1	0x00000001	校验失败
30	0x0000001E	密钥无效
61	0x0000003D	密钥不存在
99	0x00000063	设备运算失败
201	0x000000C9	数据长度无效
202	0x000000CA	数据不满足运算条件
203	0x000000CB	读数据库失败
204	0x000000CC	写数据库失败
205	0x000000CD	设备 ID 长度无效
206	0x000000CE	VKEK 版本号长度无效
210	0x000000D2	指定设备的上级设备不存在
211	0x000000D3	指定 ID 的设备已经存在
212	0x000000D4	指定 ID 的设备不存在
213	0x000000D5	指定 ID 的设备数量过多
214	0x000000D6	设备数量异常
215	0x000000D7	指定的 VKEK 已经存在
216	0x000000D8	存在多个相同的 VKEK
217	0x000000D9	指定的 VKEK 不存在
218	0x000000DA	指定的设备类型无效
219	0x000000DB	VEK 长度无效
220	0x000000DC	分散 VKEK 校验失败
221	0x000000DD	对称算法标识无效
231	0x000000E7	查找结果为空
232	0x000000E8	查找结果过多，超过预设最大值

241	0x000000F1	用于分散 VKEK 的版本号异常
242	0x000000F2	不存在对应的值

2.1.2 设备信息

2.1.2.1 数据定义

```
typedef struct DeviceInfo_st {
    unsigned int    uiStatus;
    unsigned int    uiType;
    unsigned int    uiSecurityLevel;
    unsigned int    uiVemkAlgorithm;
    char            strDeviceId[LENGTH_DEVICEID];
    char            strParentDevId[LENGTH_DEVICEID];
    char            strCreateTime[LENGTH_TIMER+2];
    char            strLastRegisterTime[LENGTH_TIMER+2];
    char            strDeleteTime[LENGTH_TIMER+2];
} DeviceInfo;
```

2.1.2.2 成员说明

成员名	说明	取值
uiStatus	设备状态	0,未登记 1,已登记 2,已停用
uiType	设备类型	0,IPC 1,NVR 2,NVR 下属 IPC
uiSecurityLevel	设备安全级别	
uiVemkAlgorithm	设备用于分散 VKEK 的算法标识	VGD_SM1_ECB、VGD_SM4_ECB
strDeviceId	设备 ID	20 位十进制字符（uiType 为 1 时此值为空）
strParentDevId	所属上级设备 ID	空或 20 位十进制字符
strCreateTime	设备注册时间	格式: YYYY-MM-DD HH:MM:SS 长度固定：19 字节
strLastRegisterTime	设备最后一次签到时间，即 VKEK 生成时间	格式: YYYY-MM-DD HH:MM:SS 长度固定：19 字节
strDeleteTime	设备注销时间	格式: YYYY-MM-DD HH:MM:SS 长度固定：19 字节

2.1.3 VKEK 信息

2.1.3.1 数据定义

```
typedef struct VKEKRecord_st {
    char            strDeviceId[LENGTH_DEVICEID];
    char            strRegisterTime[LENGTH_TIMER+2];
    char            strVkekVersion[LENGTH_KVERSION+2];
    unsigned int    uiAlgorithm;
    unsigned char   pucCipherVkey[LENGTH_KCIPHER];
    unsigned int    uiCipherVkeyLen;
    char            strKcv[LENGTH_KCV+2];
} VKEKRecord;
```

2.1.3.2 成员说明

成员名	说明	取值
strDeviceId	设备编号	20 位十进制字符
strRegisterTime	设备签到时间	格式: YYYY-MM-DD HH:MM:SS
strVkekVersion	VKEK 版本	ASCII 字符串（签到时间），计算 VKEK 时时直接对 ASCII 数据的 SM3 摘要结果折半异或作为分散数据
uiAlgorithm	VKEK 加密 VEK 的算法	VGD_SM1_ECB、VGD_SM4_ECB
pucCipherVkey	密文 VKEK	C1||C3||C2 格式 BCD 码
uiCipherVkeyLen	密文 VKEK 有效字节长度	
strKcv	VKEK 密钥校验值	16 进制字符串（16 位）

2.2 查询设备

2.2.1 函数定义

```
int VKF_SearchDevice ( char* strDeviceID, DeviceInfo* pstOutDeviceInfo );
```

2.2.2 参数说明

- strDeviceID　　【输入】设备编号
- pstOutDeviceInfo　　【输出】设备信息
- 返回值：0 表示成功，其他值表示失败

2.2.3 应用场景

- 查询该设备是否注册过

2.3注册设备

2.3.1 函数原型

int VKF_GenerateVEMK (char* strDeviceId, char* strParentDevId,

unsigned int uiAlgorithmVemk, char* strTimer, unsigned int uiSecurityLevel);

2.3.2 参数说明

- strDeviceId 【输入】IPC 设备编号
- strParentDevd 【输入】NVR 设备 ID（添加通用设备时此值为空）
- uiAlgorithmVemk 【输入】分散生成 VKEK 时使用的算法（支持 SM1 和 SM4 算法，分别对应值 VGD_SM1_ECB 和 VGD_SM4_ECB）
- strTimer 【输入】传入当前的时间信息（格式：YYYY-MM-DD HH:MM:SS）
- uiSecurityLevel 【输入】设备的安全级别
- 返回值：0 表示成功，其他值表示失败

2.3.3 应用场景

- 向密钥管理服务器添加（登记）一个 IPC 或 NVR 设备

2.4注销设备

2.4.1 函数原型

int VKF_DeleteVEMK (char* strDeviceId, char* strTimer);

2.4.2 参数说明

- strDeviceId 【输入】IPC 设备编号
- strTimer 【输入】传入当前的时间信息
- 返回值：0 表示成功，其他值表示失败

2.4.3 应用场景

- 停用 IPC 或 NVR 设备时，通知密钥管理服务器删除（停用）该设备

2.5 通用设备签到（为 IPC 等通用设备生成 VKEK 并导出）

2.5.1 函数原型

```
int VKF_GenerateIpcVKEK ( char* strDeviceId,
    char* strVkekVersion,
    unsigned int uiAlgorithmVkek,
    unsigned char* pucPubKey,
    unsigned int uiPubKeyL,
    char* strRegisterTime,
    unsigned char* pucOutCipherVkek,
    unsigned int* puiOutVkekLen );
```

2.5.2 参数说明

- strDeviceId 【输入】设备编号
- strVkekVersion 【输入】VKEK 版本号
- uiAlgorithmVkek 【输入】VKEK 密钥的算法属性标识（表示 VKEK 加密 VEK 时使用的算法，支持 SM1 和 SM4 算法，分别对应值 VGD_SM1_ECB 和 VGD_SM4_ECB）
- pucPubKey 【输入】设备公钥，DER 编码数据（二进制数据）
- uiPubKeyL 【输入】设备公钥数据的长度
- strRegisterTime 【输入】传入的设备注册的时间
- pucOutCipherVkek 【输出】VKEK 密文数据
- puiOutVkekLen 【输出】VKEK 密文数据长度
- 返回值：0 表示成功，其他值表示失败

2.5.3 应用场景

- 设备签到时，请求该设备的最新 VKEK 并发给设备

2.6 NVR 设备签到（为 NVR 设备生成 VKEK 并导出）

2.6.1 函数原型

```
int VKF_GenerateNvrVKEK ( char* strNvrId,
```

```
char* strVkekVersion,
unsigned int uiAlgorithmVkek,
unsigned char* pucPubKey,
unsigned int uiPubKeyL,
char* strRegisterTime,
unsigned char* pucOutCipherVkek,
unsigned int* puiOutVkekLen,
unsigned int* puiVemkVer,
unsigned char* pucOutCipherVemk,
unsigned int* puiOutVemkLen);
```

2.6.2 参数说明

- strNvrId 【输入】NVR 设备编号
- strVkekVersion 【输入】VKEK 版本号
- uiAlgorithmVkek 【输入】VKEK 密钥的算法属性标识（表示 VKEK 加密 VEK 时使用的算法，支持 SM1 和 SM4 算法，分别对应值 VGD_SM1_ECB 和 VGD_SM4_ECB）
- pucPubKey 【输入】设备公钥，DER 编码数据（二进制数据）
- uiPubKeyL 【输入】设备公钥数据的长度
- strRegisterTime 【输入】传入的设备注册的时间
- pucOutCipherVkek 【输出】VKEK 密文数据
- puiOutVkekLen 【输出】VKEK 密钥字节长度
- puiVemkVer 【输出】用于分散 VKEK 的最新版本号（为 0 时表示无更新）
- pucOutCipherVemk 【输出】更新后的用于分散 VKEK 的值（puiVemkVer 版本号不为 0 时有效）
- puiOutVemkLen 【输出】分散 VKEK 的密钥的字节长度
- 返回值：0 表示成功，其他值表示失败

2.6.3 应用场景

- NVR 设备签到

2.7 NVR 记录 IPC 的签到记录

2.7.1 函数原型

int VKF_RegisterVKEKRecord (char* strIpcId,

char* strNvrId,

char* strVkekVersion,

unsigned int uiAlgorithmVkek,

unsigned int uiVemkVer,

char* strRegisterTime);

2.7.2 参数说明

- strIpcId 【输入】IPC 设备编号
- strNvrId 【输入】NVR 设备编号（若与 IPC 设备属性中的 NVR ID 不匹配，则报错）
- strVkekVersion 【输入】VKEK 版本号
- iAlgorithm 【输入】生成 VKEK 的分散算法标识（支持 SM1 和 SM4 算法，分别对应值 VGD_SM1_ECB 和 VGD_SM4_ECB）
- uiVemkVer 【输入】NVR 当前用于分散 VKEK 的版本号
- strRegisterTime 【输入】注册时间
- 返回值：0 表示成功，其他值表示失败

2.7.3 应用场景

- 摄像头向 NVR 注册时，NVR 把注册记录向密钥管理服务器注册。

2.8 根据时间区间查询历史 VKEK

2.8.1 函数原型

int VKF_SearchVKEK_Dates (char* strDeviceId,

char* strStartDate,

char* strEndDate,

unsigned int* puiOutCount,

VKEKRecord* pstOutVkeks);

2.8.2 参数说明

- strDeviceId 【输入】设备编号
- strStartDate 【输入】查询的起始日期
- strEndDate 【输入】查询的截止日期
- puiOutCount 【输入/输出】查询到的 VKEK 记录的条目数量，输入 pstOutVkeks 参数的缓冲区结构数量，输出实际查询到的记录数量
- pstOutVkeks 【输出】历史 VKEK 数据的指针地址（strCipherVkek 和 strKcv 值为空）
- 返回值：0 表示成功，其他值表示失败；若输入的 puiOutCount 参数小于查询结果的数量，则返回 VKR_ARGLENGTH，同时 puiOutCount 参数输出实际查询结果的数量，pstOutVkeks 不予返回。

注：strStartDate 和 strEndDate 都为空时，只输出最新 VKEK。

2.8.3 应用场景

- 向密钥管理服务器请求该视频时间段内所有的 VKEK 的版本号。

2.9 根据版本号查询历史 VKEK

2.9.1 函数原型

```
int VKF_SearchVKEK_Version ( char* strDeviceId,
    char* strVkekVersion,
    VKEKRecord* pstOutVkek);
```

2.9.2 参数说明

- strDeviceId 【输入】设备编号
- strVkekVersion 【输入】VKEK 版本号
- pstOutVkek 【输出】VKEK 信息（strCipherVKEK 值为空）
- 返回值：0 表示成功，其他值表示失败

2.9.3 应用场景

- 请求特定版本的 VKEK。

2.10 根据时间区间批量导出历史 VKEK

2.10.1 函数原型

```
int VKF_ExportVKEK_Dates ( char* strDeviceId,
    unsigned char* pucPubKey,
    unsigned int uiPubKeyL,
    char* strStartDate,
    char* strEndDate,
    unsigned int* puiOutCount,
    VKEKRecord** ppstOutVkeks );
```

2.10.2 参数说明

- strDeviceId 【输入】设备编号
- pucPubKey 【输入】接收方公钥
- uiPubKeyL 【输入】接收方公钥的数据长度
- strStartDate 【输入】起始日期
- strEndDate 【输入】结束日期
- puiOutCount 【输出】查询到的 VKEK 记录的条目数量
- ppstOutVkeks 【输出】导出的 VKEK 缓冲区指针的地址
- 返回值：0 表示成功，其他值表示失败

注：若输入的 puiOutCount 参数小于查询结果的数量，则返回 VKR_ARGLENGTH，同时 puiOutCount 参数输出实际查询结果的数量，pstOutVkeks 不予返回。

2.10.3 应用场景

- 向密钥管理服务器请求该视频时间段内所有的 VKEK 的版本号。

2.11 VKEK 记录缓冲区释放

2.11.1 函数原型

```
int VKF_VkekRecorderFree(VKEKRecord * pstVkeks)
```

2.11.2 参数说明

- pstVkeks 【输入】待释放的缓冲区指针

2.11.3 应用场景

用于释放由接口分配的 VKEK 结构体缓冲区内存。

2.12 根据版本号导出历史 VKEK

2.12.1 函数原型

```
int VKF_ExportVKEK_Version ( char* strDeviceId,
    char* strVkekVersion,
    unsigned char* pucPubKey,
    unsigned int uiPubKeyL,
    unsigned int * puiOutCount,
    VKEKRecord** ppstOutVkek );
```

2.12.2 参数说明

- strDeviceId 【输入】设备编号
- strVkekVersion 【输入】VKEK 版本号
- pucPubKey 【输入】接收方公钥（XY 参数串联）
- uiPubKeyL 【输入】接收方公钥的数据长度
- puiOutCount 【输出】导出 VKEK 记录的数量缓冲区指针
- ppstOutVkek 【输出】导出的 VKEK 缓冲区指针的地址
- 返回值：0 表示成功，其他值表示失败

2.12.3 应用场景

- 请求特定版本的 VKEK。

2.13 视频数据解密

2.13.1 函数原型

```
int VKF_DecryptVideo ( char* strDeviceId,
```

char* strVkekVersion,

char* strCipherVek,

unsigned int uiAlgorithmVek,

unsigned char* pucIV,

unsigned char* pucCipherData,

unsigned int uiCipherDataL,

unsigned char* pucOutPlainData,

unsigned int* puiOutPlainDataL);

2.13.2 参数说明

- strIpcId 【输入】视频来源 IPC 设备编号
- strVkekVersion 【输入】视频包中的 VKEK 版本号
- strCipherVek 【输入】视频包中的密文 VEK（视频数据使用 VEK 加密，VEK 使用 VKEK 加密，Base64 编码数据）
- uiAlgorithmVek 【输入】VEK 加密视频数据的算法标识（VGD_SM1_OFB、VGD_SM1_OFB）
- pucIV 【输入/输出】数据解密的初始化向量，执行成功后输出下一段连续密文的解密初始化向量（固定长度为 16 字节）
- pucCipherData 【输入】密文视频数据
- uiCihperDataL 【输入】密文视频数据的长度
- pucOutPlainData 【输出】明文视频数据
- puiOutPlainDataL 【输出】明文视频数据长度
- 返回值：0 表示成功，其他值表示失败

2.13.3 应用场景

- 平台解密视频数据

2.14 生成信令校验数据

2.14.1 函数原型

int VKF_GenerateSignalCheckData (char* strDeviceId,

char* strSignalData,

char* strOutCheckData);

2.14.2 参数说明

- strDeviceId 【输入】设备编号
- strSignalData 【输入】信令数据
- strOutCheckData 【输出】信令校验数据（Base64 编码）
- 返回值：0 表示成功，其他值表示失败

2.14.3 应用场景

- 生成信令校验数据

2.15 校验信令校验数据

2.15.1 函数原型

int VKF_VeriySignalCheckData (char* strDeviceId,

char* strSignalData,

char* strCheckData);

2.15.2 参数说明

- strDeviceId 【输入】设备编号
- strSignalData 【输入】信令数据
- strCheckData 【输入】校验数据（Base64 编码）
- 返回值，0 表示通过验证，1 表示验证错误，其他表示失败。

2.15.3 应用场景

- 校验信令完整性

2.16 导入 SM2 公钥加密的 SIP 临时 VKEK

2.16.1 函数原型

int VKF_ImportSipVKEK_WithSM2(char* strDeviceId,

char* strVkekVersion,

unsigned int uiVkekAlgorithm,

unsigned char* pucPubKey,

unsigned int puiPubKeyL,

char* strVkekGenerateTime,

unsigned char* pucVkekWithSM2,

unsigned int uiVkekWithSM2Len,

int (*pfCipherTranse)(unsigned char*, unsigned int, unsigned char*, unsigned int, unsigned char**, unsigned int*));

2.16.2 参数说明

- strDeviceId 【输入】对端 SIP 网关设备编号
- strVkekVersion 【输入】VKEK 密钥版本号
- uiVkekAlgorithm 【输入】VKEK 密钥的算法标识
- pucPubKey 【输入】SM2 公钥
- puiPubKeyL 【输入】SM2 公钥数据长度
- strVkekGenerateTime 【输入】VKEK 的产生时间
- pucVkekWithSM2 【输入】待导入的由 SM2 公钥加密的 VKEK（C1||C3||C2 格式）
- uiVkekWithSM2Len 【输入】SM2 公钥加密的数据长度
- pfCipherTranse 【输入】回调函数指针（使用 SM2 公钥加密的保护密钥加密 VKEK 并输出）
- 返回值：0 表示成功，其他值表示失败

2.16.3 回调函数说明

回调函数接收由同一把 SM2 公钥加密的保护密钥和 VKEK 密钥，使用保护密钥加密 VKEK 密钥后返回。参数如下：

- 【输入】SM2 公钥加密的 VKEK 密文
- 【输入】SM2 公钥加密的 VKEK 密文数据长度
- 【输入】SM2 公钥加密的保护密钥密文
- 【输入】SM2 公钥加密的保护密钥密文数据长度
- 【输出】保护密钥加密的 VKEK 密文（16 字节数据）
- 【输出】保护密钥加密 VKEK 的算法（支持 SM1_ECB 或 SM4_ECB）

- 【返回值】成功则返回 0，否则返回非 0

2.16.4 应用场景

实现平台间的级联时，用于本地平台获取从对端平台接收的 VKEK。

2.17 全局说明

2.17.1 获取错误代码

此接口运行过程中出错时抛出 cn.tass.exceptions.TAException 类的实例，可通过 getErrorCode 获取错误代码，若错误代码为 0，则表示该错误为接口执行中遇到无法支持的条件导致接口抛错。

2.18 查询设备

2.18.1 函数定义

public VEMKInfo VKF_SearchDevice(String deviceId) throws TAException;

2.18.2 参数说明

- deviceId 【输入】设备编号
- 返回值： 用于分散 VKEK 的值及设备信息

2.18.3 应用场景

- 查询该设备是否注册过

2.19 注册设备

2.19.1 函数定义

public void VKF_GenerateVEMK(String deviceId, String parentDeviceId, int algorithmVemk, String timer, int securityLevel) throws TAException;

2.19.2 参数说明

- deviceId 【输入】设备编号
- parentDeviceId 【输入】上级设备的编号
- algorithmVemk 【输入】分散生成 VKEK 时使用的算法（支持 SM1 和 SM4 算法，分别对应值 VGD_SM1_ECB 和 VGD_SM4_ECB）
- timer 【输入】传入当前的时间信息（格式：YYYY-MM-DD HH:MM:SS）
- securityLevel 【输入】设备的安全级别

2.19.3 应用场景

- 向平台的密钥管理服务器添加（登记）一个 IPC 或 NVR 设备

2.20 注销设备

2.20.1 函数原型

public void VKF_DeleteVEMK(String deviceId, String timer) throws TAException;

2.20.2 参数说明

- deviceId 【输入】IPC 设备编号
- timer 【输入】传入当前的时间信息

2.20.3 应用场景

停用 IPC 或 NVR 设备时，通知密钥管理服务器删除（停用）该设备