

KeyConvertor工具的使用

KeyConvertor主要是用于切割或者提取HDCP1.x系列的原始Key，并转化为RK Hdcp烧写工具能识别的Key格式 (*.skf)

对于HDCP1.x系列，RX TX 原始Key大小都是308字节，原始文件包含了一个4字节的头，因此文件大小应该是 $308 * N + 4$ （或者大小是 $308 * N$ ）。如果大小不是这个值，那么RK Key转换工具就认为不是HDCP1.x原始Key了。KeyConvertor工具提示Key格式错误

工具分两部分，上面部分是提取，当原始Key文件超过1W个时，建议先拆分Key。此步骤不是一定要做。

提取Key

- 选择原始Key。
- 提取方式
 - 按输入提取范围，起始值从1开始，最大值不能超过Key总数
 - 按每个文件的Key个数提取，这样的话，会提取原始Key文件的全部Key，并且保存在输出文件夹里面。比如设置为1，那么会生产很多原始Key文件，每个文件里面包含了1个Key
- 点击提取。即可在指定目录生成指定个数的Key文件。

转换Key

将原始Key文件转换成RK自定义格式的Key文件

- 选择原始Key

- 点击转换。即可将原始Key转成自定义格式的skf文件；转换分为两种方式
 - 输出key：那么直接输出一个skf文件，里面包含了 原始Key 里面的全部key
 - 输出key(单独文件)：输出N个skf文件，N为原始Key的数量

注意：对于 HDMIRX 与 加密 选项，只有在key转化需要加密时候，才会有需要，一般不会去使用，如有需要，建议咨询RK相关工程师

附录一 关于Key的格式与解析

HDCEP Signing Facility User's Guide

Order Delivery Formats

The Device Key Set Package is delivered as a binary file of the Device Key Set Package. The file is of length $(4+308n)$ bytes, where n is the quantity of Device Key Sets in the Device Key Set Package (10,000, 100,000, or 1,000,000). The quantity, n , can be inferred by examining the length of the file.

File Format

The order file consists of a 4-byte Order Format, followed by an array of n Device Key Set Records. There is one Device Key Set Record representing each Device Key Set in the Device Key Set Package.

Size (bytes):	Description:
4	"Order Format." This value equals 1 for transmitter Device Key Sets and 2 for receiver Device Key Sets. All other values are undefined.
$308n$	This is an array of Device Key Set Records.

HDCEP Signing Facility User's Guide (HDCEP 2.x)

Transmitter Device Key Set

The file is of length 445 bytes. The quantity is always 1 and is the same for all transmitter devices. The Header value is always 1.

The Transmitter Device Key Set is made up of two parts:

- Licensed Constant (36 Bytes)
- Root Public Key (405 Bytes)

Position in Bytes	Name	Size in Bytes	Example Value	Description	Block & Size
0-3	Header	4	0000 0001		
4-19	Licensed Constant	16	F9A7 E002 676C 1E63 60D9 B03B 831D 9090	Licensed Constant (36 Bytes)	Transmitter Key Set (441 Bytes)
20-39	SHA-1(Licensed Constant)	20	A93C 5C7F 4EFA E892 067E 04EA 657A 38FF 9B6C B019		
40-423	Root Public Modulus N	384	B24E ... D321	Root Public Key (405 Bytes)	
424	Root Public Exponent E	1	11		
425-444	SHA-1(N,E)	20	E63F 9AC2 5238 6D0E DEC5 F784 9FB9 C3C0 3CC4 CC02		

Receiver Device Key Set

The file is of length $(4+36+862n)$ bytes, where n is the quantity of Receiver Device Key Sets in the Receiver Device Key Set Package (10,000, 100,000, or 1,000,000). The quantity, n , can be inferred by examining the length of the file. The header value is always 2.

Each Receiver Device Key Set is broken up into three parts:

- Licensed Constant (36 Bytes only available once at the top of the file)
- Device Public Certificate (522 Bytes)
- Device Private Key (340 Bytes)

Position in Bytes	Name	Size in Bytes	Example Value	Description	Block & Size
0-3	Header	4	0000 0002		
4-19	Licensed Constant	16	F9A7 ... 9090	Used for each Receiver Device	Licensed Constant (36 bytes)
20-39	SHA-1(Licensed Constant)	20	A93C ... B019		
40-44	Receiver ID	5	745B B8BD 04	Device Public Certificate #0 (522 bytes)	Key Set #0 (862 bytes)
45-172	Device Public Modulus N	128	E5E2 ... B3F1		
173-175	Device Public Exponent E	3	0100 01		
176-177	Reserved	2	0000		
178-561	Root RSAPKCSv1.5_Signature (SHA-256(Receiver ID, N, E, Reserved))	384	713F... FE3A		
562-625	Device Private CRT P	64	FC49 ... B29B	Device Private Key #0 (340 bytes)	
626-689	Device Private CRT Q	64	E944 ... 0663		
690-753	Device Private CRT D mod (P-1)	64	0BDC ... 7F6B		
754-817	Device Private CRT D mod (Q-1)	64	C041 ... BF9F		
818-881	Device Private CRT $q^{-1} \text{ mod } p$	64	001D ... 5729		
882-901	SHA-1(RECEIVER ID, N, E, Reserved, Signature, P, Q, D mod (P-1), D mod (Q-1), $q^{-1} \text{ mod } p$)	20	A78B ... E757		

解析伪代码

```
srcFile.Open(strSrcKey, CFile::modeRead|CFile::typeBinary);
```

```
if(uiHeader==0x02000000) { //Receiver
    /* Receiver */
    bTransmit = FALSE;
    if(((uiFileSize - 40) % 862) != 0) {
        /* 错误的格式 */
    }
    uikeySize = 862;
```

```

    uiKeyCount = (uiFileSize - 40)/uiKeySize;
    isHdcp2x = true;
} else if(uiHeader==0x01000000){
    /* Transmitter */
    bTransmit = TRUE;
    if(((uiFileSize - 40)%405) != 0) {
        if(g_pLog) {
            g_pLog->Record(_T("Error:ConvertKeyProc check source key file
failed,is not multiply of 405"));
        }
        strError = GetLocalString(_T("ERROR_CHECK_SOURCE_KEY"));
        goto Exit_Convert;
    }
    uiKeySize = 405;
    uiKeyCount = (uiFileSize-40)/uiKeySize;
    isHdcp2x = true;
} else if((((uiFileSize - 4) % 308 == 0) || uiFileSize % 308 == 0)) {
    uiKeySize = 308;
    uiKeyCount = uiFileSize/308;
    isHdcp2x = false;
}
}

```

RK key格式

对于RK的SKF格式的key，是在原始Key的基础上，头部加了一个20字节的Header，尾部加了一个CRC32校验（其中尾部CRC32校验是可选的）

```

typedef struct
{
    char szTag[4]; /* " SKF" */
    UINT uiKeySize;
    UINT uiKeyAllocSize;
    UINT uiKeyCount;
} STRUCT_SKF_HEADER, *PSTRUCT_SKF_HEADER;

```

RK烧写工具，会去校验头部与尾部CRC，如果不对的话，会提示格式错误。对于只有一个Key的HDCP1.x skf文件大小为：20 + 308 + 4 = 332（或者 20 + 308 = 328）

附录二 关于Key的烧写

目前HDCP1.x部分烧写，客户问题比较多，这里解释下

原始的Key是308字节，在RK工具烧写过程中的话，会生成一个2字节随机数（seed）做一个IP vendor的一个算法加密，这个加密过程是不会改变key大小的，因此Key还是308字节。在烧写key的时候，seed会随文件一起传输到设备，并在文件尾部加了一个CRC32校验，烧写到设备的实际上是314字节。因为驱动从设备读取的key实际上是314字节