HDCP Key

User Guide

Issue	**09**
Date	**2016-03-15**

Copyright © HiSilicon Technologies Co., Ltd. 2016. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of HiSilicon Technologies Co., Ltd.

Trademarks and Permissions

, HISILICON , and other HiSilicon icons are trademarks of HiSilicon Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

HiSilicon Technologies Co., Ltd.

Address:	Huawei Industrial Base Bantian, Longgang Shenzhen 518129 People's Republic of China
Website:	http://www.hisilicon.com
Email:	support@hisilicon.com

About This Document

Purpose

This document describes the usage of the HiSilicon high-bandwidth digital content protection (HDCP) key and the factory operation process for using the HDCP key solution.

Related Versions

The following table lists the product versions related to this document.

Product Name	Version
Hi3716C	V2XX
Hi3719C	V1XX
Hi3719M	V1XX
Hi3718C	V1XX
Hi3718M	V1XX
Hi3716C	V1XX
Hi3716H	V1XX
Hi3716M	V2XX
Hi3716M	V3XX
Hi3712	V1XX
Hi3716M	V4XX
Hi3798C	V1XX
Hi3798C	V2XX
Hi3796C	V1XX
Hi3798M	V1XX
Hi3796M	V1XX
Hi3110E	V5XX

HiSilicon Proprietary and Confidential Copyright ©
HiSilicon Technologies Co., Ltd

Intended Audience

This document is intended for:

- Technical support personnel
- Software development engineers

Change History

Changes between document issues are cumulative. Therefore, the latest document issue contains all changes made in previous issues.

Issue 09 (2016-03-15)

This issue is the ninth official release, which incorporates the following changes:

Hi3716M V330 is supported.

Issue 08 (2015-11-10)

This issue is the eighth official release, which incorporates the following changes:

Section 3.2.5 is modified.

Issue 07 (2015-04-30)

This issue is the seventh official release, which incorporates the following changes:

Hi3798C V200, Hi3716M V420, and Hi3716M V410 are supported.

Issue 06 (2015-03-10)

This issue is the sixth official release, which incorporates the following changes:

Hi3110E V500 is supported.

Issue 05 (2014-12-09)

This issue is the fifth official release, which incorporates the following changes:

Descriptions are added to support Hi3716M V310, Hi3719C V100, Hi3798M V100, and Hi3796M V100.

Chapter 2 is modified.

Issue 04 (2014-09-05)

This issue is the fourth official release, which incorporates the following changes:

The version, operation description, and figures for the HDCP key tool are modified.

Issue 03 (2014-05-20)

This issue is the third official release, which incorporates the following changes:

Chapter 3 Application Reference

Section 3.2 is modified.

Section 3.2.5 is added.

Descriptions are added to support Hi3798C V100 and Hi3796C V100.

Issue 02 (2014-01-20)

This issue is the second official release.

Some modifications are made to support Hi3716M V400.

Issue 01 (2013-11-30)

This issue is the first official release.

Chapter 3 Application Reference

Sections 3.2.4.1, 3.2.4.2, and 3.2.4.3 are added.

Issue 00B01 (2013-08-15)

This issue is the first draft release.

Contents

Figures

HiSilicon Proprietary and Confidential Copyright ©
HiSilicon Technologies Co., Ltd

1 Overview

1.1 Introduction

HDCP_key.exe is a tool that splits the HDCP key purchased from the Digital Content Protection Limited Liability Company. The tool runs on Windows.

Note the following before using **HDCP_key.exe**:

- HDCP is used for encrypted HDMI data transmission. The HDCP key must be purchased from the Digital Content Protection LLC.
- The HDCP key solution for the Hi3712/Hi3716C V200 and later versions is different from that for the Hi3716C V100/Hi3716M V200/Hi3716M V300.

NOTE

Unless otherwise specified, the Hi3712/Hi3716C V200 and later versions described in this document include the following chips: Hi3712, Hi3716C V200, Hi3719M V100, Hi3719C V100, Hi3798C V100, Hi3798C V200, Hi3798M V100, Hi3796M V100, Hi3796C V100, Hi3716M V310, Hi3716M V330,and Hi3716M V400.

- For the Hi3716C V100/Hi3716M V200/Hi3716M V300, the HDCP key can be directly burnt into the one-time programmable (OTP) of the chip. For the Hi3712/Hi3716C V200 and later versions, the chip OTP area stores the HDCP root key dedicated for HDCP key encryption and decryption.
- For the Hi3712/Hi3716C V200 and later versions, the HDCP key can be encrypted by using the HDCP root key, and the encrypted HDCP key can be stored in the specified area of the flash memory. If the HDCP key is used for encrypted HDMI data transmission, the HDMI module reads the encrypted HDCP key data from the flash memory, decrypts the HDCP key to the chip, and uses the decrypted HDCP key to transmit encrypted data. The decrypted HDCP key data cannot be read.

1.2 Environment Preparation

To use HDCP_key.exe, perform the following steps:

Step 1 Copy **HDCP_key.exe** to a local drive of a Windows PC.

Step 2 Double-click **HDCP_key.exe**.

----End

1.3 Quick Start

1.3.1 Installation

Installation is not required. Double-click **HDCP_key.exe** to run the tool.

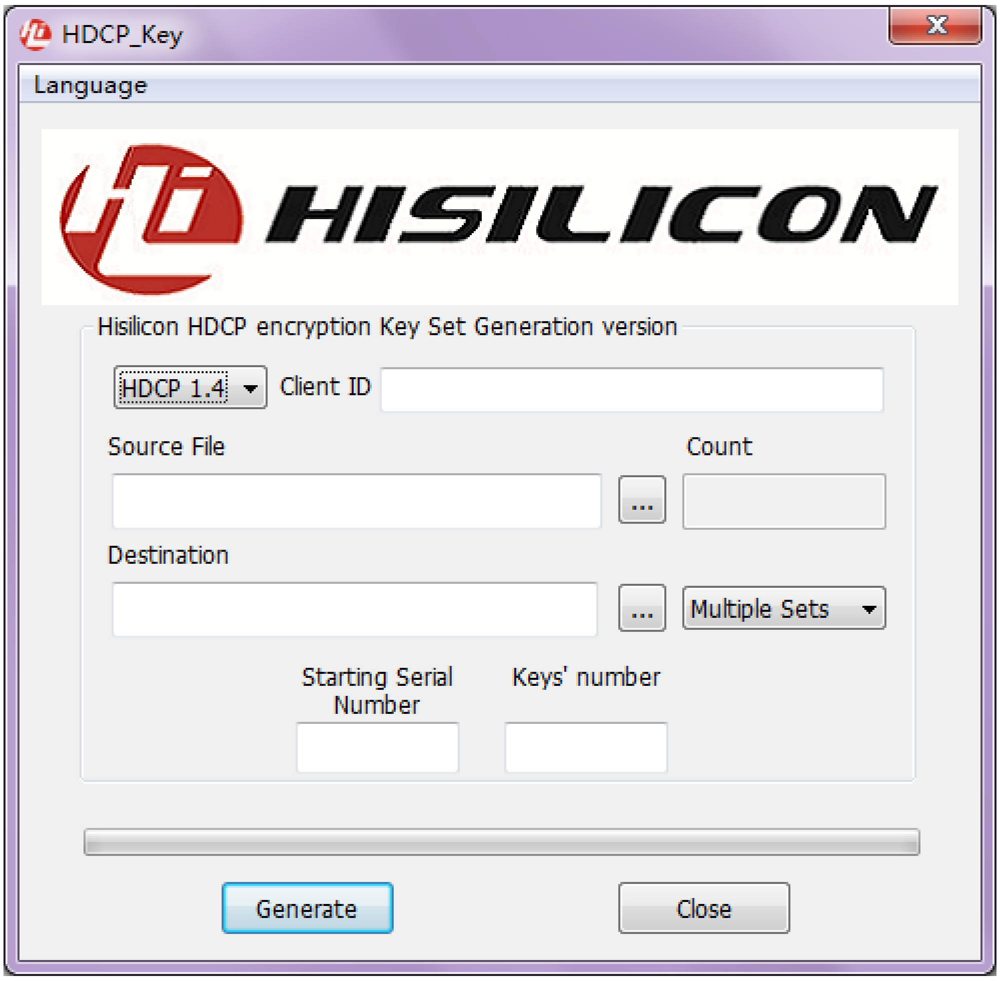
1.3.2 GUI

CAUTION

Descriptions of the tool in the following sections apply to HDCP_Key V3.0.0.0 (included) to HDCP_Key V3.1.0.0 (excluded).

Figure 1-1 shows the GUI of the tool.

Figure 1-1 GUI



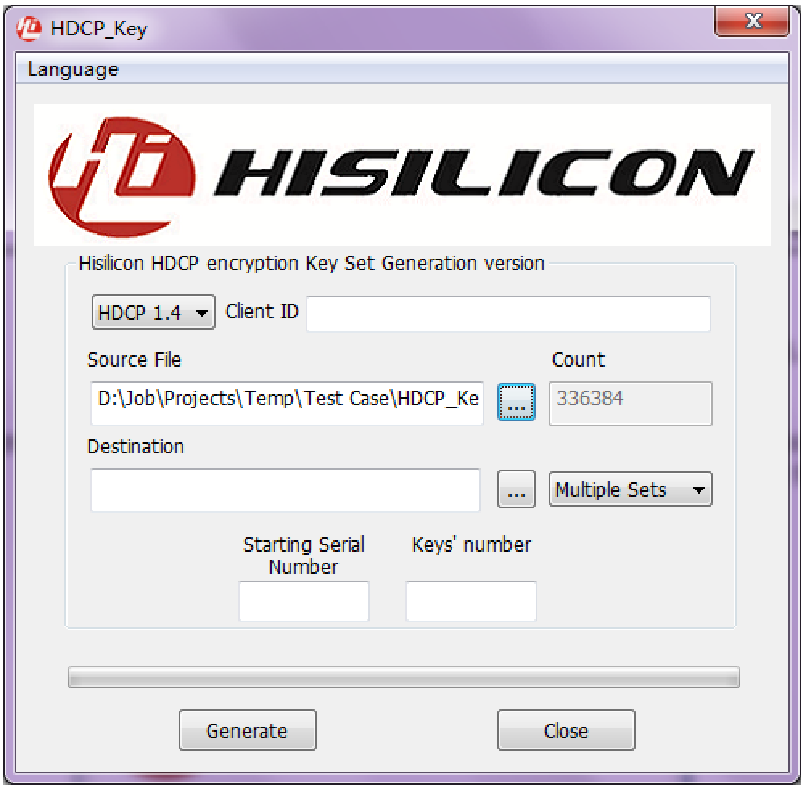
2 GUI and Function Description

HDCP_key.exe has only one GUI, on which the HDCP key can be split.

To split the HDCP key by using **HDCP_key.exe**, perform the following steps:

Step 1 Select the HDCP key package file (.bin). The number of HDCP keys is displayed, as shown in Figure 2-1. Typically the number is 10000, 100000, or 1000000. To split the HDCP 2.2 key, select **HDCP 2.2** from the drop-down list box. See Figure 2-1.

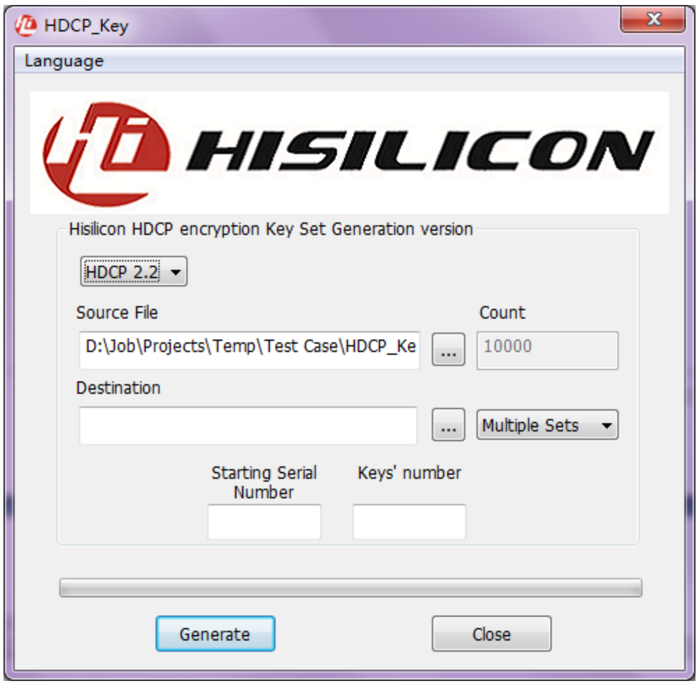
Figure 2-1 Selecting the HDCP key version (1)



HDCP 1.4
Client ID

Source File: D:\Job\Projects\Temp\Test Case\HDCP_Ke
Count: 336384
Destination
Multiple Sets

Figure 2-2 Selecting the HDCP key version (2)



Hisilicon HDCP encryption Key Set Generation version

HDCP 2.2

Source File: D:\Job\Projects\Temp\Test Case\HDCP_Ke

Count: 10000

Destination:

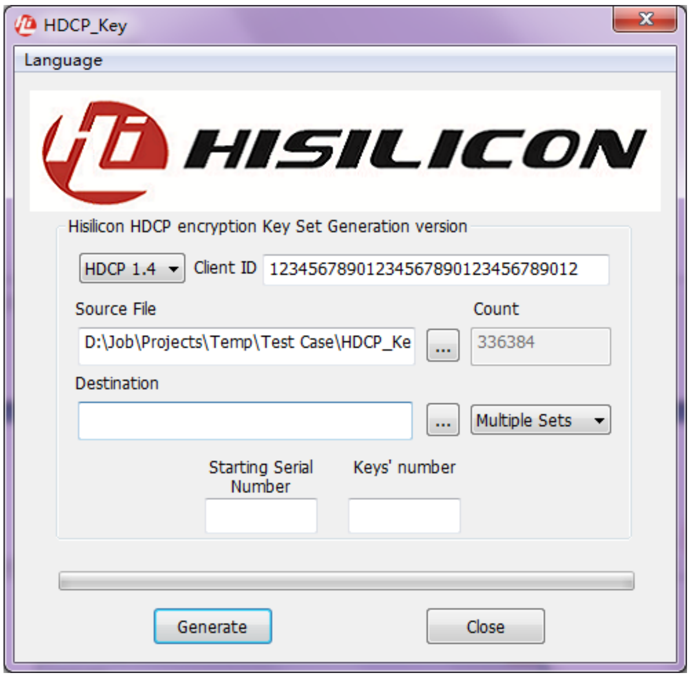
Multiple Sets

Starting Serial Number:

Keys' number:

Step 2 If **HDCP 1.4** is selected, set **Client ID** to 32-byte English characters or digits, for example, **12345678901234567890123456789012**. If **HDCP 2.2** is selected, **Client ID** does not need to be specified.

Figure 2-3 Setting the client ID



Hisilicon HDCP encryption Key Set Generation version

HDCP 1.4

Client ID: 12345678901234567890123456789012

Source File: D:\Job\Projects\Temp\Test Case\HDCP_Ke

Count: 336384

Destination:

Multiple Sets

Starting Serial Number:

Keys' number:

Step 3 Specify the directory for storing the keys, select a key split mode, and set the number of keys to be generated and the start serial number. The following three split modes are available:

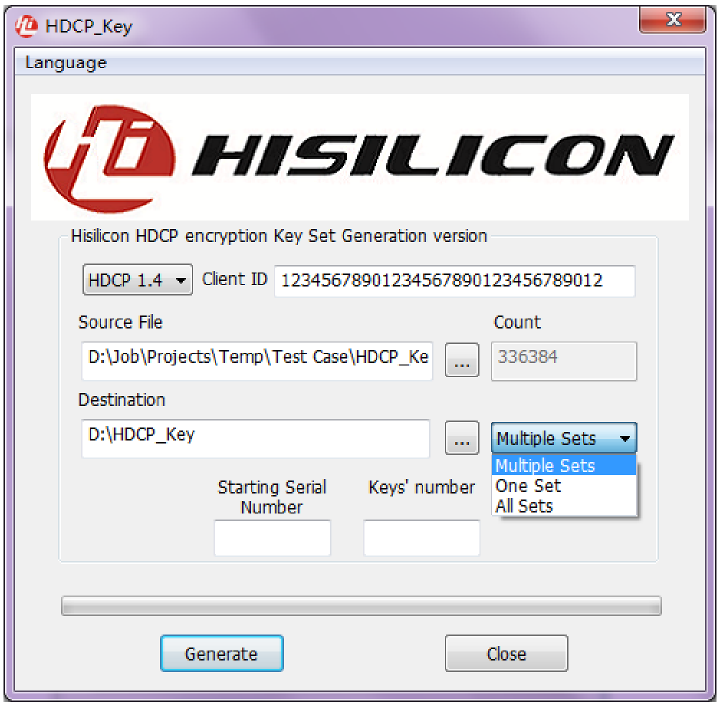
- **All Sets**: All keys are extracted.
- **Multiple Sets**: Consecutive keys within a specific range are extracted. The range must be within 1 to n (n is the number of HDCP keys). If this option is selected, the start serial number of keys and the number of keys to be generated must be specified.
- **One Set**: A specific key is extracted.

The directory for storing extracted keys can be specified.

NOTE

The following takes the HDCP 1.4 key as an example.

Figure 2-4 Setting the keys to be split (All Sets)



HDCP 1.4 Client ID 12345678901234567890123456789012

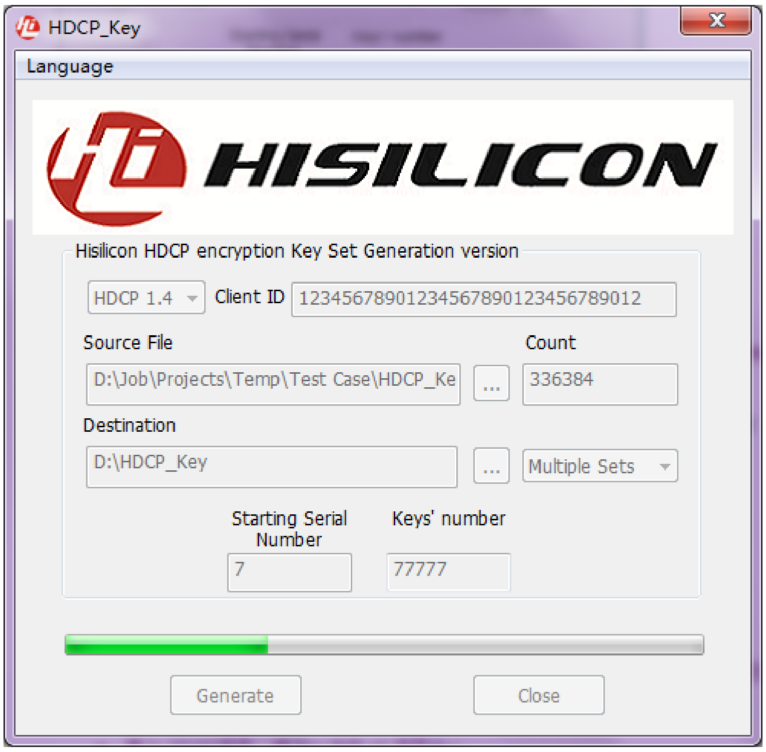
Source File: D:\Job\Projects\Temp\Test Case\HDCP_Ke

Count: 336384

Destination: D:\HDCP_Key

Multiple Sets

Figure 2-5 Setting the keys to be split (Multiple Sets)



Hisilicon HDCP encryption Key Set Generation version

HDCP 1.4

Client ID 12345678901234567890123456789012

Source File D:\Job\Projects\Temp\Test Case\HDCP_Ke

Count 336384

Destination D:\HDCP_Key

Multiple Sets

Starting Serial Number 7

Keys' number 77777

NOTE

The .bin file of a sub key of the HDCP 1.4 key has 384 bytes, which are described as follows:

- Bytes 0–7 (8 bytes): HiSilicon identifier bit (HISI_xxx)
- Bytes 8–15 (8 bytes): tool version number (V0000001)
- Bytes 16–47 (32 bytes): client ID
- Bytes 48–367 (320 bytes): encrypted HDCP key (16-byte-aligned)
- Bytes 368–384 (16 bytes): random digits, currently 0

The .bin file of a sub key of the HDCP 2.2 key has 902 bytes, which are described as follows:

- Bytes 0–39 (40 bytes): header of the source key
- Bytes 40–901(862 bytes): HDCP 2.2 keys

----End

3 Application Reference

3.1 HDCP Key Solution for Hi3716C V100/Hi3716H V100/Hi3716M V200/Hi3716M V300

3.1.1 APIs and Related Data Structures

- HI_UNF_HDCP_SetHDCPKeyEx is used to burn the HDCP key to the chip OTP area.
  ```
  HI_S32 HI_UNF_HDCP_SetHDCPKeyEx(HI_UNF_OTP_HDCPKEY_S stHdcpKey)
  ```
- HI_UNF_HDCP_LockHDCPKeyEx is used to lock the burnt HDCP key in the chip OTP area.
  ```
  HI_S32 HI_UNF_HDCP_LockHDCPKeyEx(HI_VOID)
  ```
- HI_UNF_HDCP_GetKeyBurnFlag is used to obtain the burning flag bit of the HDCP key, checking whether the HDCP key has been burnt.
  ```
  HI_S32 HI_UNF_HDCP_GetKeyBurnFlag(HI_BOOL *pbKeyBurnFlag)
  ```

The related data structures are as follows:

```
typedef struct hiUNF_DECRYPT_HDCP_S
{
    HI_U8 u8KSV[5]; /**<HDCP KSV:40bits, Original data */
    HI_U8 u8PrivateKey[280];
 }HI_UNF_DECRYPT_HDCP_S;
typedef struct hiUNF_ENCRYPT_HDCP_S
{
    HI_U8 u8EncryptKey[384];     /**<HDCP Encryption key */
}HI_UNF_ENCRYPT_HDCP_S;

typedef struct
{
    HI_BOOL EncryptionFlag;
    union
{
/**< Key = DecryptData,if EncryptionFlag == HI_FALSE.if EncryptionFlag ==
```

```
HI_TRUE,key = Encryptiondata */
      HI_UNF_DECRYPT_HDCP_S DecryptData;
      HI_UNF_ENCRYPT_HDCP_S EncryptData;
   }key;
   HI_U32 Reserved; /**<Reserved for future use */
}HI_UNF_OTP_HDCPKEY_S;
```

NOTE

When the **Reserved** member in the HI_UNF_OTP_HDCPKEY_S structure is **0xabcd1234**, the HDCP key is not burnt and a code indicating success is returned for debugging.

The HDCP key can be written for only once, and the second write operation will fail.

HI_UNF_HDCP_SetHDCPKey is implemented by using HI_UNF_HDCP_SetHDCPKeyEx and HI_UNF_HDCP_LockHDCPKeyEx.

3.1.2 Samples

Three samples for burning the HDCP key are stored in **SDK/sample/sethdcpkey**.

- **sethdcp_orgkey.c**: burning the original single HDCP key
- **sethdcp_orgpacketkey.c**: burning a specific key from the original HDCP key package
- **sethdcp_encrykey.c**: burning the encrypted HDCP keys generated by **HDCP_key.exe**

For details about the parameters, see **SDK/sample/sethdcp/Readme.txt**.

The following sections take **sethdcp_encrykey.c** (the executable file after compilation is sample_sethdcp_encrykey) as an example.

3.1.3 Burning the HDCP Key

To burn the HDCP key, perform the following steps:

Step 1 Verify that the required ko drivers (**hi_sethdcp.ko**, **hi_cipher.ko**, and **hi_otp.ko**) are loaded.

You can run the **insmod** command to check whether the preceding drivers are loaded. If not, load the drivers manually:

insmod /kmod/ hi_sethdcp.ko

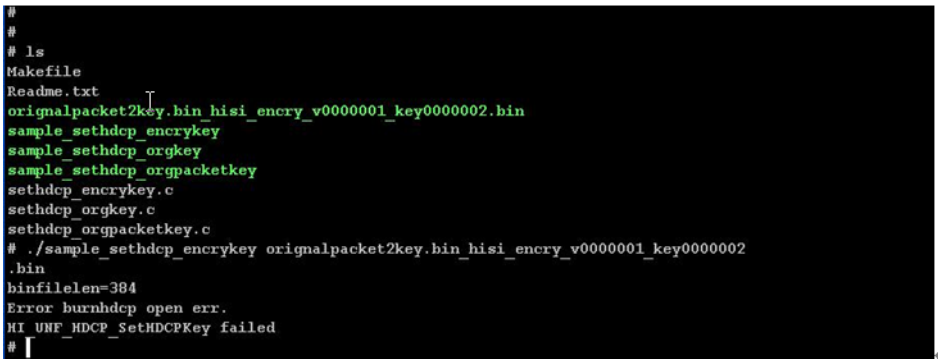
Step 2 Copy the encrypted sub keys (384-byte .bin files generated by the tool) to **SDK/sample/sethdcpkey**.

Step 3 Burn the HDCP key by running the following commands:

./sample_sethdcp_encrykey 10000_fake_key.bin_hisi_encry_v00000001_key0000001.bin

- If **hi_sethdcp.ko** is not loaded, the information similar to that shown in Figure 3-1 is displayed.

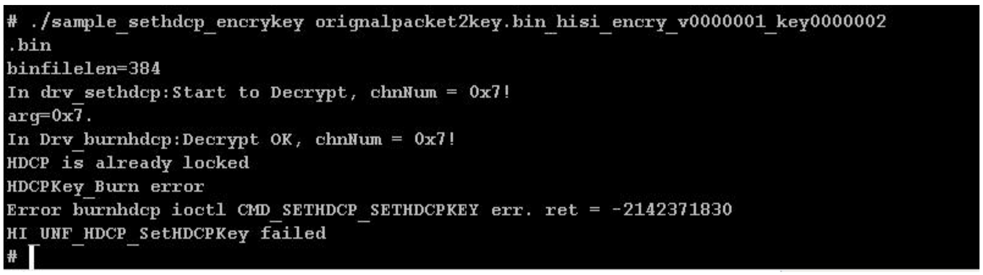
Figure 3-1 Information displayed after burning if hi_sethdcp.ko is not loaded



```
#
#
# ls
Makefile
Readme.txt
orignalpacket2key.bin_hisi_encry_v0000001_key0000002.bin
sample_sethdcp_encrykey
sample_sethdcp_orgkey
sample_sethdcp_orgpacketkey
sethdcp_encrykey.c
sethdcp_orgkey.c
sethdcp_orgpacketkey.c
# ./sample_sethdcp_encrykey orignalpacket2key.bin_hisi_encry_v0000001_key0000002
.bin
binfilelen=384
Error burnhdcp open err.
HI_UNF_HDCP_SetHDCPKey failed
#
```

- If the HDCP key has already been burnt or an error occurs during burning, the information similar to that shown in Figure 3-2 is displayed.

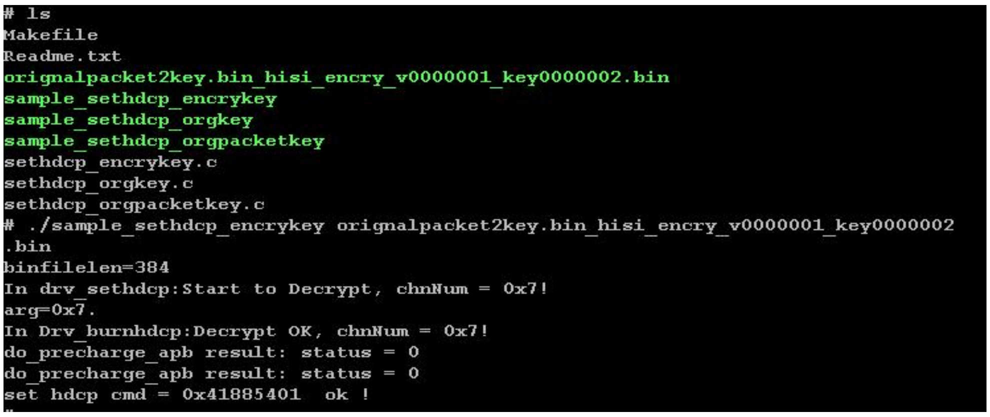
Figure 3-2 Information displayed after burning if the HDCP key has already been burnt or an error occurs during burning



```
# ./sample_sethdcp_encrykey orignalpacket2key.bin_hisi_encry_v0000001_key0000002
.bin
binfilelen=384
In drv_sethdcp:Start to Decrypt, chnNum = 0x7!
arg=0x7.
In Drv_burnhdcp:Decrypt OK, chnNum = 0x7!
HDCP is already locked
HDCPKey_Burn error
Error burnhdcp ioctl CMD_SETHDCP_SETHDCPKEY err. ret = -2142371830
HI_UNF_HDCP_SetHDCPKey failed
#
```

- If the HDCP key is burnt successfully, the information similar to that shown in Figure 3-3 is displayed.

Figure 3-3 Information displayed after burning if the HDCP key is burnt successfully



```
# ls
Makefile
Readme.txt
orignalpacket2key.bin_hisi_encry_v0000001_key0000002.bin
sample_sethdcp_encrykey
sample_sethdcp_orgkey
sample_sethdcp_orgpacketkey
sethdcp_encrykey.c
sethdcp_orgkey.c
sethdcp_orgpacketkey.c
# ./sample_sethdcp_encrykey orignalpacket2key.bin_hisi_encry_v0000001_key0000002
.bin
binfilelen=384
In drv_sethdcp:Start to Decrypt, chnNum = 0x7!
arg=0x7.
In Drv_burnhdcp:Decrypt OK, chnNum = 0x7!
do_precharge_apb result: status = 0
do_precharge_apb result: status = 0
set hdcp cmd = 0x41885401  ok !
```

----End

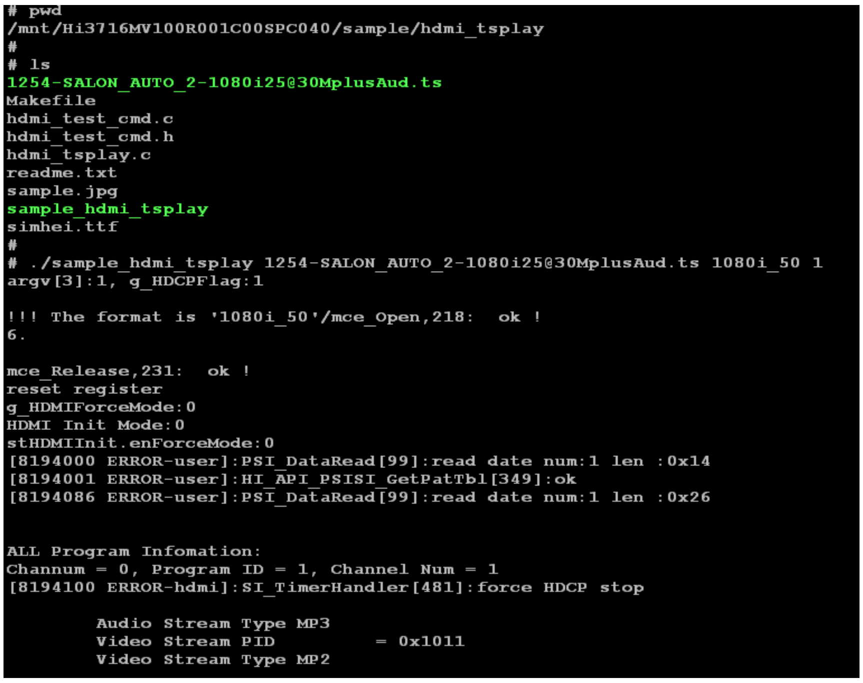
3.1.4 Verifying the HDCP Key

After burning the HDCP key, you can switch to the **SDK/sample/hdmi_tsplay** directory and play HD TS files to verify the burnt HDCP key. Perform the following steps:

Step 1 Save the TSs to be played to the **SDK/sample/hdmi_tsplay** directory.

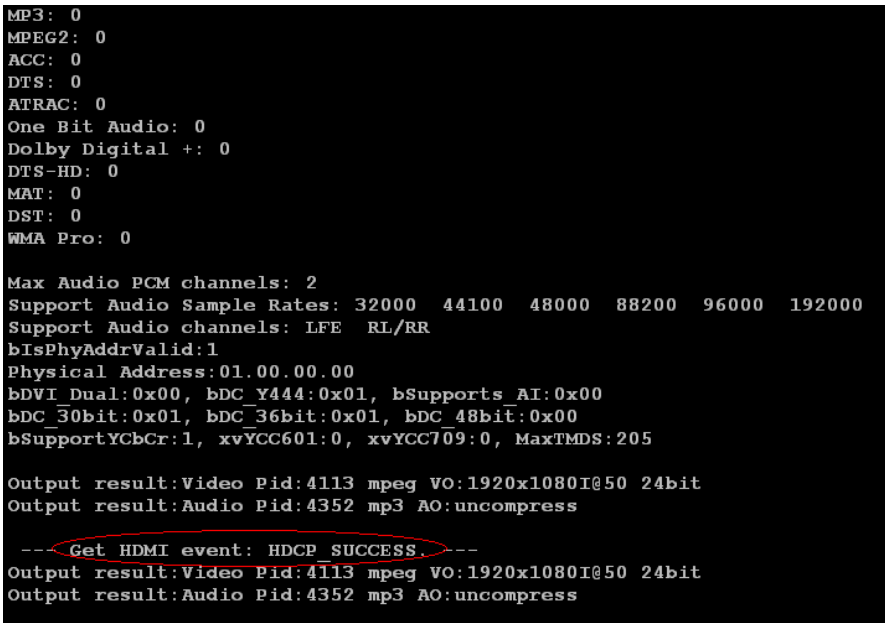
Step 2 Run **./sample_hdmi_tsplay 1254-SALON_AUTO_2-1080i25@30MplusAud.ts 1080i_50 1** (the last 1 is the HDCP playback flag), and check whether the HDMI event is successfully obtained.

Figure 3-4 Checking whether the HDMI event is successfully obtained (1)



```
# pwd
/mnt/Hi3716MV100R001C00SPC040/sample/hdmi_tsplay
#
# ls
1254-SALON_AUTO_2-1080i25@30MplusAud.ts
Makefile
hdmi_test_cmd.c
hdmi_test_cmd.h
hdmi_tsplay.c
readme.txt
sample.jpg
sample_hdmi_tsplay
simhei.ttf
#
# ./sample_hdmi_tsplay 1254-SALON_AUTO_2-1080i25@30MplusAud.ts 1080i_50 1
argv[3]:1, g_HDCPFlag:1

!!! The format is '1080i_50'/mce_Open,218:  ok !
6.

mce_Release,231:  ok !
reset register
g_HDMIForceMode:0
HDMI Init Mode:0
stHDMIInit.enForceMode:0
[8194000 ERROR-user]:PSI_DataRead[99]:read date num:1 len :0x14
[8194001 ERROR-user]:HI_API_PSISI_GetPatTbl[349]:ok
[8194086 ERROR-user]:PSI_DataRead[99]:read date num:1 len :0x26


ALL Program Infomation:
Channum = 0, Program ID = 1, Channel Num = 1
[8194100 ERROR-hdmi]:SI_TimerHandler[481]:force HDCP stop

        Audio Stream Type MP3
        Video Stream PID         = 0x1011
        Video Stream Type MP2
```

Figure 3-5 Checking whether the HDMI event is successfully obtained (2)



```
MP3: 0
MPEG2: 0
ACC: 0
DTS: 0
ATRAC: 0
One Bit Audio: 0
Dolby Digital +: 0
DTS-HD: 0
MAT: 0
DST: 0
WMA Pro: 0

Max Audio PCM channels: 2
Support Audio Sample Rates: 32000  44100  48000  88200  96000  192000
Support Audio channels: LFE  RL/RR
bIsPhyAddrValid:1
Physical Address:01.00.00.00
bDVI_Dual:0x00, bDC_Y444:0x01, bSupports_AI:0x00
bDC_30bit:0x01, bDC_36bit:0x01, bDC_48bit:0x00
bSupportYCbCr:1, xvYCC601:0, xvYCC709:0, MaxTMDS:205

Output result:Video Pid:4113 mpeg VO:1920x1080I@50 24bit
Output result:Audio Pid:4352 mp3 AO:uncompress

 ---Get HDMI event: HDCP_SUCCESS.---
Output result:Video Pid:4113 mpeg VO:1920x1080I@50 24bit
Output result:Audio Pid:4352 mp3 AO:uncompress
```

If "Get HDMI event: HDCP_SUCCESS." is displayed, the HDCP key is successfully burnt.

----End

3.2 HDCP Key Solution for the Hi3712 /Hi3716C V200 and Later Versions

3.2.1 APIs and Related Data Structures

HI_UNF_HDCP_EncryptHDCPKey is used to determine whether to encrypt the HDCP key by using the HDCP root key stored in the chip or the private key defined by HiSilicon. Note that the HiSilicon-defined private key is used only for debugging. You need to use the customized HDCP root key to encrypt the HDCP key during mass production.

```
HI_S32  HI_UNF_HDCP_EncryptHDCPKey(HI_UNF_HDCP_HDCPKEY_S stHdcpKey,
HI_BOOL bIsUseHdcpRootKey, HI_U8 u8OutEncry
```

The related data structures are as follows:

```
typedef struct hiUNF_DECRYPT_HDCP_S
{
    HI_U8 u8KSV[5];          /**< HDCP KSV:40bits, Original data */
    HI_U8 u8PrivateKey[280]; /**< HDCP Device Private key:40*56bits, Original
data */
}HI_UNF_HDCP_DECRYPT_S;

/** Encrypted HDCP key */
/** CNcomment: encrypted HDCP key */
typedef struct hiUNF_ENCRYPT_HDCP_S
{
    HI_U8 u8EncryptKey[384];
}HI_UNF_HDCP_ENCRYPT_S;

typedef struct hiUNF_HDCPKEY_HDCP_S
{
   HI_BOOL EncryptionFlag;  /**< HI_TRUE:Encryption, HI_FALSE: Unencryption
*/
   union
{
/**< Key = DecryptData,if EncryptionFlag == HI_FALSE.if EncryptionFlag ==
HI_TRUE,key = Encryptiondata */
       HI_UNF_HDCP_DECRYPT_S DecryptData;
       HI_UNF_HDCP_ENCRYPT_S EncryptData;
}key;
   HI_U32 Reserved;   /**< Reserved for future use */
}HI_UNF_HDCP_HDCPKEY_S;
```

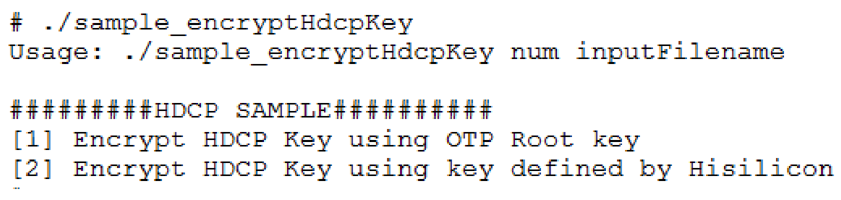
3.2.2 Samples

The HDCP root key is used to encrypt the HDCP key purchased from DCP LLC. It has 16 bytes, and its value is customized and maintained by the customer. To burn the HDCP root key to the chip OTP area, see **sample/otp/sample_otp_sethdcprootkey.c** in the SDK.

Modify the HDCP root key to be written in **sample_otp_sethdcprootkey.c**, and compile and run **sample_otp_sethdcprootkey.c** to burn the HDCP root key. After being burnt, the HDCP root key is locked and cannot be read.

To generate encrypted HDCP key data, see **sample/hdcpkey/sample_encryptHdcpKey.c** in the SDK.

Figure 3-6 Generating encrypted HDCP key data



```
# ./sample_encryptHdcpKey
Usage: ./sample_encryptHdcpKey num inputFilename

#########HDCP SAMPLE##########
[1] Encrypt HDCP Key using OTP Root key
[2] Encrypt HDCP Key using key defined by Hisilicon
```

After the HDCP root key is burnt, run ./sample_encryptHdcpKey to encrypt the HDCP key by using the HDCP root key.

- To encrypt the HDCP key by using the HDCP root key in the chip OTP area, run the following commands:

  ./sample_encryptHdcpKey 1 10000_fake_key.bin_hisi_encry_v00000001_key0000001.bin

- To encrypt the HDCP key by using the HiSilicon-defined key, run the following commands:

  ./sample_encryptHdcpKey 2 10000_fake_key.bin_hisi_encry_v00000001_key0000001.bin

NOTE

10000_fake_key.bin_hisi_encry_v00000001_key0000001.bin is a 384-byte sub key generated by using **HDCP_Key.exe**.

The HiSilicon-defined private key for all chips is the same. It is used only for debugging. You are advised not to use it during mass production. Therefore, set the first parameter to **1** when running sample_encryptHdcpKey.

3.2.3 Factory Manufacturing

3.2.3.1 Preparing Data

Prepare the following data before factory manufacturing:

- HDCP root key
- Encrypted HDCP key

The 16-byte HDCP root key can be customized. It is written into the OTP area of the chip during factory manufacturing and cannot be read after being locked.

To generate the encrypted HDCP key data, perform the following steps:

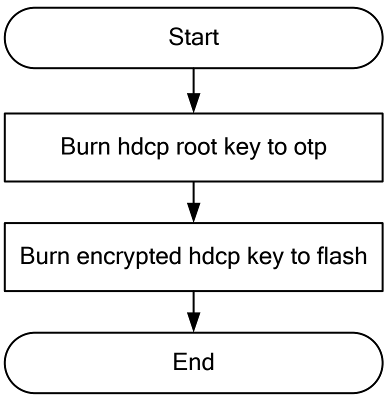
Step 1 Split the HDCP key by using **HDCP_Key.exe** to generate multiple sub keys (384 bytes each). For details, see section 3.1.2 "Samples."

Step 2 Generate the encrypted HDCP keys by calling the HiSilicon UNF interface, using the HDCP root key stored on the chip or private key defined by HiSilicon.

----End

3.2.3.2 Factory Manufacture Process

Figure 3-7 Factory manufacturing process of the HDCP key solution for the Hi3712/Hi3716C V200 and later versions

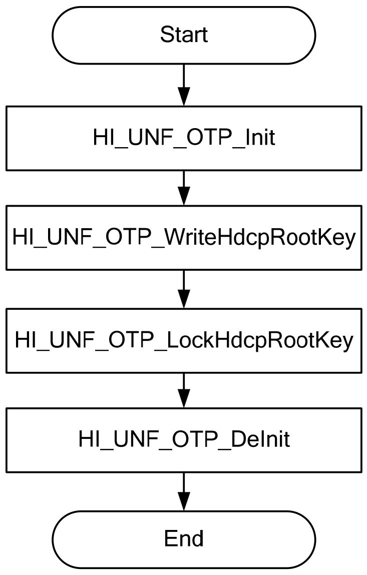


NOTE

If you use the HDCP root key to encrypt the HDCP key, the HDCP root key must match the encrypted HDCP key. Otherwise, the verification fails and the HDMI data transmission cannot be protected.

Figure 3-8 shows the process for burning the HDCP root key to the OTP area of the chip and locking the burnt HDCP root key.

Figure 3-8 Burning and locking the HDCP root key



CAUTION

The HDCP root key can be burnt only once, and it cannot be read after being locked.

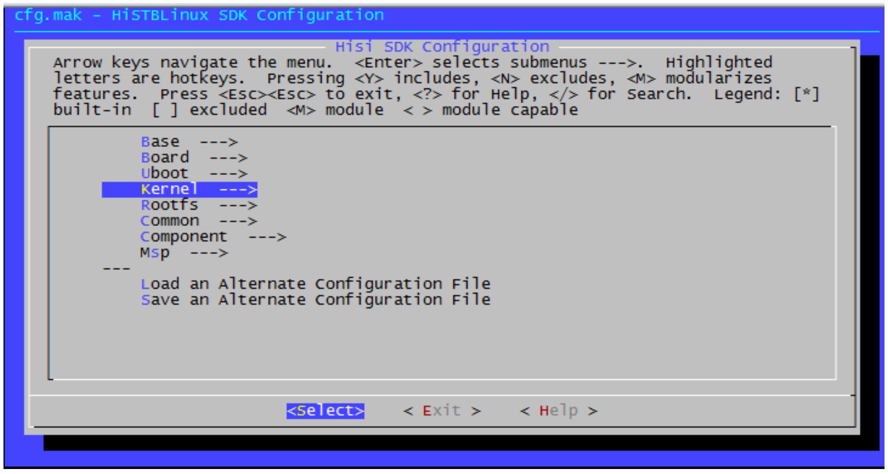
3.2.4 Verifying the HDCP Key

3.2.4.1 Enabling HDMI Support for HDCP

To enable the HDMI support for HDCP, perform the following steps:

Step 1 Run **make menuconfig** in the SDK root directory, as shown in Figure 3-9.

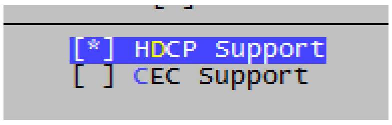
Figure 3-9 menuconfig



```
cfg.mak - HiSTBLinux SDK Configuration
                     Hisi SDK Configuration
 Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted
 letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
 features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*]
 built-in  [ ] excluded  <M> module  < > module capable

          Base  --->
          Board  --->
          Uboot  --->
          Kernel  --->
          Rootfs  --->
          Common  --->
          Component  --->
          Msp  --->
      ---
          Load an Alternate Configuration File
          Save an Alternate Configuration File

              <Select>    < Exit >    < Help >
```

Step 2 Choose **Msp** > **HDMI config**, and select **HDCP support**, as shown in Figure 3-10.

Figure 3-10 HDCP option

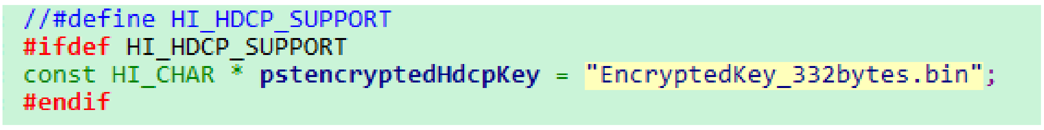


```
[*] HDCP Support
[ ] CEC Support
```

Step 3 Enable the HDCP macro in **SDK/sample/common/hi_adp_hdmi.c**.

```
#define HI_HDCP_SUPPORT
```

Figure 3-11 Sample default configuration



```
//#define HI_HDCP_SUPPORT
#ifdef HI_HDCP_SUPPORT
const HI_CHAR * pstencryptedHdcpKey = "EncryptedKey_332bytes.bin";
#endif
```

Step 4 Compile the SDK.

----End

3.2.4.2 HDMI-Related APIs

HI_UNF_HDMI_LoadHDCPKey is used to import the encrypted key.

```
HI_S32 HI_UNF_HDMI_LoadHDCPKey(HI_UNF_HDMI_ID_E enHdmi,
HI_UNF_HDMI_LOAD_KEY_S *pstLoadKey);
```

The related data structures are as follows:

```
/**HDMI HDCP key struct*/
typedef struct hiUNF_HDMI_LOAD_KEY_S
```

```
{
HI_U8 *pu8InputEncryptedKey;   /**<Encrypted key pointer */
HI_U32 u32KeyLength;  /**<Encrypted key length*/
}HI_UNF_HDMI_LOAD_KEY_S;
```

For details about the usage, see **/sample/common/hi_adp_hdmi.c** in the SDK.

3.2.4.3 Verifying the HDCP Key by Running the Reference Sample

To verify the HDCP key, perform the following steps:

Step 1 Generate an HDCP key (**org_xxxxx.bin**) by using **HDCP_Key.exe** of HiSilicon.

Step 2 Encrypt **org_xxxxx.bin** by using the HDCP sample:

./sample_encryptHdcpKey 1 org_xxxx.bin

The encrypted HDCP key file **EncryptedKey_332bytes.bin** is generated.

Step 3 Assign read and write properties to **EncryptedKey_332bytes.bin**:

chmod 777 EncryptedKey_332bytes.bin

Step 4 Save **EncryptedKey_332bytes.bin** to **SDK/sample/hdmi_tsplay**.

Step 5 Save the TSs to be played to **SDK/sample/hdmi_tsplay**.

Step 6 Run **./sample_hdmi_tsplay stream.ts 1080i_50**.

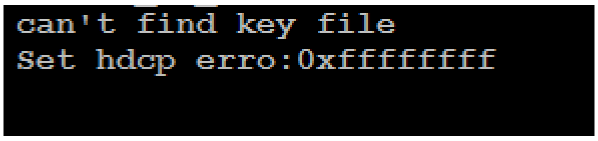
Step 7 Enter **hdmi_hdcp 1** to initiate the HDCP handshake process.

Verify the following items:

1. When the HDMI module is being initialized, the encrypted HDCP key data in the flash memory is decrypted and loaded to the chip. If the loading fails, the error message "Load HDCP key error!" is displayed.
2. After being loaded to the chip, the HDCP key is verified. If the verification fails, the error message "HDCP crc check is error!" is displayed.
3. If the HDCP key loading or verification fails, it is most likely that the encrypted HDCP key is inconsistent with the key in the chip. You can solve this problem by regenerating the HDCP key in the flash memory.

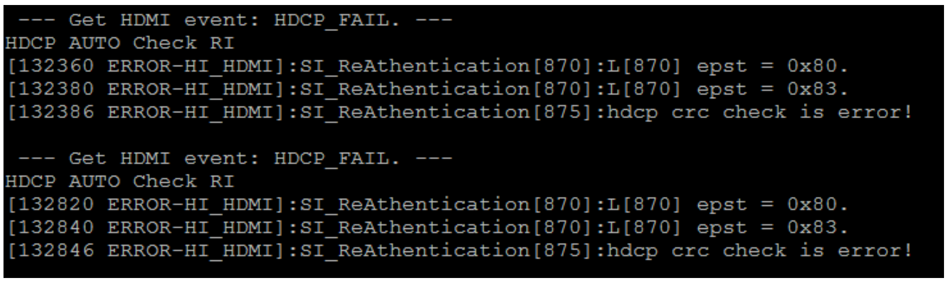
Figure 3-12 to Figure 3-14 shows the debugging information when the HDCP key is being loaded.

Figure 3-12 Error information indicating that the encrypted HDCP key is not prepared when the HDMI sample runs



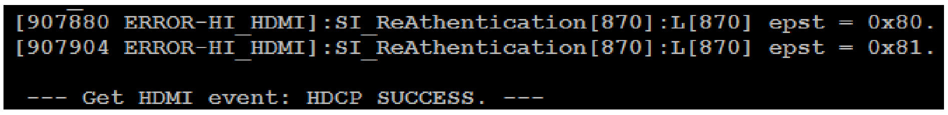
```
can't find key file
Set hdcp erro:0xffffffff
```

Figure 3-13 Error information indicating that the HDMI module loads the incorrect encrypted HDCP key



```
 --- Get HDMI event: HDCP_FAIL. ---
HDCP AUTO Check RI
[132360 ERROR-HI_HDMI]:SI_ReAthentication[870]:L[870] epst = 0x80.
[132380 ERROR-HI_HDMI]:SI_ReAthentication[870]:L[870] epst = 0x83.
[132386 ERROR-HI_HDMI]:SI_ReAthentication[875]:hdcp crc check is error!

 --- Get HDMI event: HDCP_FAIL. ---
HDCP AUTO Check RI
[132820 ERROR-HI_HDMI]:SI_ReAthentication[870]:L[870] epst = 0x80.
[132840 ERROR-HI_HDMI]:SI_ReAthentication[870]:L[870] epst = 0x83.
[132846 ERROR-HI_HDMI]:SI_ReAthentication[875]:hdcp crc check is error!
```

Figure 3-14 Information indicating that the HDCP key is successfully loaded



```
[907880 ERROR-HI_HDMI]:SI_ReAthentication[870]:L[870] epst = 0x80.
[907904 ERROR-HI_HDMI]:SI_ReAthentication[870]:L[870] epst = 0x81.

 --- Get HDMI event: HDCP SUCCESS. ---
```

----End

3.2.5 Enabling the HDCP 1.x Feature on the Android Platform

Perform the following steps:

Step 1 Split the HDCP 1.*x* key package by using the tool provided by HiSilicon to obtain a separate encrypted key in a name similar to **org_*xxxxx*.bin**. The size of the key is 384 bytes.

CAUTION

org_*xxxxx*.bin is a single HDCP 1.x key generated by using **HDCP_Key.exe** of HiSilicon.

For details about how to use **HDCP_Key.exe**, see the *HDCP Key User Guide* in the SDK.

Step 2 Store the 384-byte key obtained in Step 1 to a directory with the read and write properties, such as **/mnt/sdcard/**. Assign the read and write properties to this key, such as **chmod 777 /mnt/sdcard/org_*xxxxx*.bin**.

Decrypt the encrypted **org_*xxxxx*.bin** by calling the setHdmiHDCPKey interface in HiSysManager of the HiSilicon. After the decryption is complete, encrypt the key once again by using the HDCP root key in the OTP, and store the key in the **deviceinfo** partition. The size of the key after encryption is 332 bytes.

The prototype of the setHdmiHDCPKey interface is as follows:
```
public int setHdmiHDCPKey(String OrgKeyPath)
```

OrgKeyPath is the character string of the **org_*xxxxx*.bin** file directory, such as **/mnt/sdcard/org_*xxxxx*.bin**.

After the calling of setHdmiHDCPKey is complete, **org_*xxxxx*.bin** is deleted automatically.

CAUTION

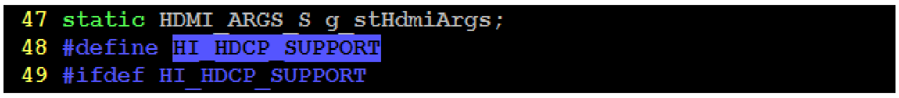
The HDMI HDCP key is stored at the tail of the **deviceinfo** partition.

The 128 KB space at the tail of the **deviceinfo** partition is used for storing the DRM key information. The 128K+4K offset at the tail of the **deviceinfo** partition is used for storing the HDMI HDCP key by default for this version.

The storage position of HDMI HDCP key can be configured automatically by writing the offset to persist.sys.hdmi.hdcp.offset. Note that the offset is in the unit of bytes and is relative to the tail of the **deviceinfo**. It must be greater than storage offset for the HDMI HDCP key plus that for the DRM key (128K + 4K: 128 x 1024 + 4 x 1024) and be less than the total size of the **deviceinfo** partition deducted by the storage size for the HDMI HDCP key (2M – 4K: 2 x 1024 x 1024 – 4 x 1024).

If the position for storing the HDMI HDCP key is not configured, then the 128K+8K offset at the tail of the **deviceinfo** partition of HiSilicon is used by default.

Step 3 Modify **device/hisilicon/bigfish/frameworks/hidisplaymanager/hal/hi_adp_hdmi.c** to enable HI_HDCP_SUPPRT (**#define HI_HDCP_SUPPORT**). For example:



```
static HDMI_ARGS_S g_stHdmiArgs;
#define HI_HDCP_SUPPORT
#ifdef HI_HDCP_SUPPORT
```

Step 4 Recompile **hidisplay.bigfish.so**, and push it to the **/system/lib/hw** directory of the board.

cd device/hisilicon/bigfish/frameworks/hidisplaymanager/hal

m -B

Push the generated file **out/target/product/Hi3719CV100/system/lib/hw/hidisplay.bigfish.so** to the **/system/lib/hw** directory.

Restart the board, and insert the HDMI cable. The HDCP 1.x feature is enabled by default after the board starts.

----End

4 Precautions

Before using **HDCP_key.exe**, check the chip type and the corresponding HDCP key solution, and perform related operations by following the instructions in chapter 3 "Application Reference."