Level 2 Security Solution for the HiSilicon Intelligent STB

User Guide

Issue	**00B04**
Date	**2016-04-08**

Trademarks and Permissions

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 how to use the HiSilicon secure boot and verification solution.

Intended Audience

This document is intended for:

- Technical support engineers
- Software development engineers

Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
⚠ DANGER	Alerts you to a high risk hazard that could, if not avoided, result in serious injury or death.
⚠ WARNING	Alerts you to a medium or low risk hazard that could, if not avoided, result in moderate or minor injury.
⚠ CAUTION	Alerts you to a potentially hazardous situation that could, if not avoided, result in equipment damage, data loss, performance deterioration, or unanticipated results.
🔑 TIP	Provides a tip that may help you solve a problem or save time.
📖 NOTE	Provides additional information to emphasize or supplement important points in the main text.

Change History

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

Issue 00B04 (2016-04-08)

This issue is the fourth draft release, which incorporates the following change:

The description of the level 2 security solution for Hi3798C V200 is added.

Issue 00B03 (2015-12-15)

This issue is the third draft release, which incorporates the following change:

Section 2.1.1 is modified.

Issue 00B02 (2015-06-02)

This issue is the second draft release, which incorporates the following change:

Section 2.4 is modified.

Issue 00B01 (2015-01-20)

This issue is the first draft release.

Contents

Figures

1 Overview

The HiSilicon secure boot and verification solution is developed based on the hardware protection mechanism provided by HiSilicon conditional access (CA) chips. This solution provides high-level protection on important data in the flash memory (such as the boot, bootargs, kernel, system, and recovery data), preventing the partition data from being tampered with or replaced by hackers using professional tools or other means. In addition, it imposes few restrictions on the Android system and board and does not affect the necessary debugging methods.

The HiSilicon secure boot and verification solution uses a linked verification mechanism, as shown in Figure 1-1.

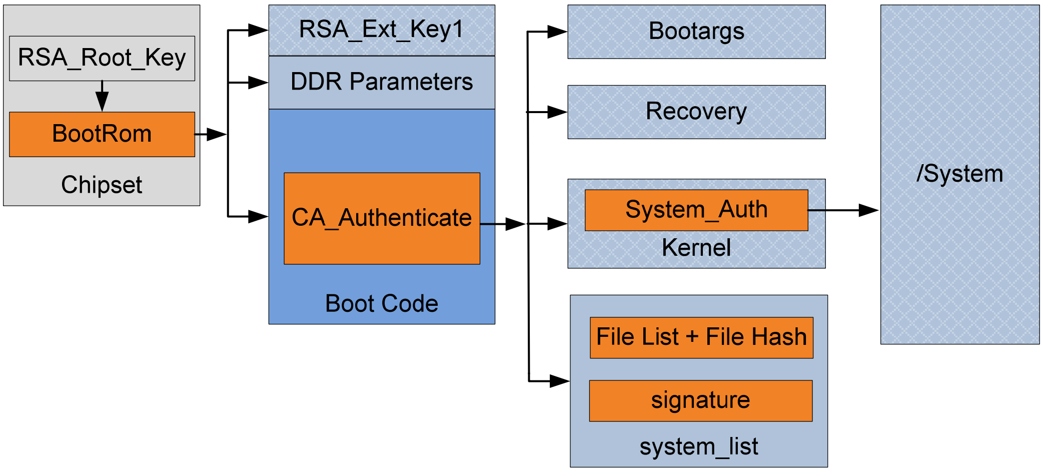
Step 1 The chip verifies the boot image.

Step 2 The boot verifies the bootargs, recovery, and kernel.

Step 3 The kernel verifies the system partition data at the background.

----End

Figure 1-1 Process of the HiSilicon secure boot and verification solution



This document describes the compilation configuration, how to burn images for mass production, and how to burn images by using the HiTool for the HiSilicon security solution.

- Compilation configuration: how to generate the keys and compile the HiSilicon security solution
- Burning images for mass production: how to burn images by using a USB flash drive or a burner
- Burning images by using the HiTool: used for development debugging of R&D engineers
- Level 2 security solution of Hi3798C V200: modifications on the chip security solution for Hi3798C V200

This solution applies only to Hi3796M V100, Hi3798M V100 and Hi3798C V200. The following flash memories are supported:

- eMMC flash
- SD flash

NOTE

Chapters 1–4 apply to Hi3796M V100 and Hi3798M V100, and chapter 5 describes the modification on the security solution for Hi3798C V200.

2 Compilation Configurations

2.1 Key Generation

The level 2 security solution requires six pairs of keys:

- Four pairs of Android system keys for verifying signatures of the APKs and upgrade packages
- Two pairs of secure boot verification keys for verifying the signature of each partition image

⚠ **CAUTION**

Keep the keys for the security solution properly. If they are lost or leaked, security issues may occur. For example, the system may be tampered with or replaced.

2.1.1 Android System Keys

The Android native system provides four pairs of keys for verifying signatures of APKs and upgrade packages. The HiSilicon security solution uses the Android native mechanism for security management of APKs and upgrade packages. Before compiling the HiSilicon security solution, you need to generate the four pairs of Android native keys.

⚠ **CAUTION**

- Ensure that the Android system keys are generated by the user **root**.
- Use the user mode for mass production of the security solution.
- Use the eng mode during key generation and development debugging (the serial port cannot be used in user mode).
- The following takes Hi3798M V100 as an example.

To generate the four pairs of Android native keys, perform the following steps:

Step 1 Switch the user to **root** and go to the root directory of the code.

```
$ sudo -i
```

Step 2 Configure the environment variables.

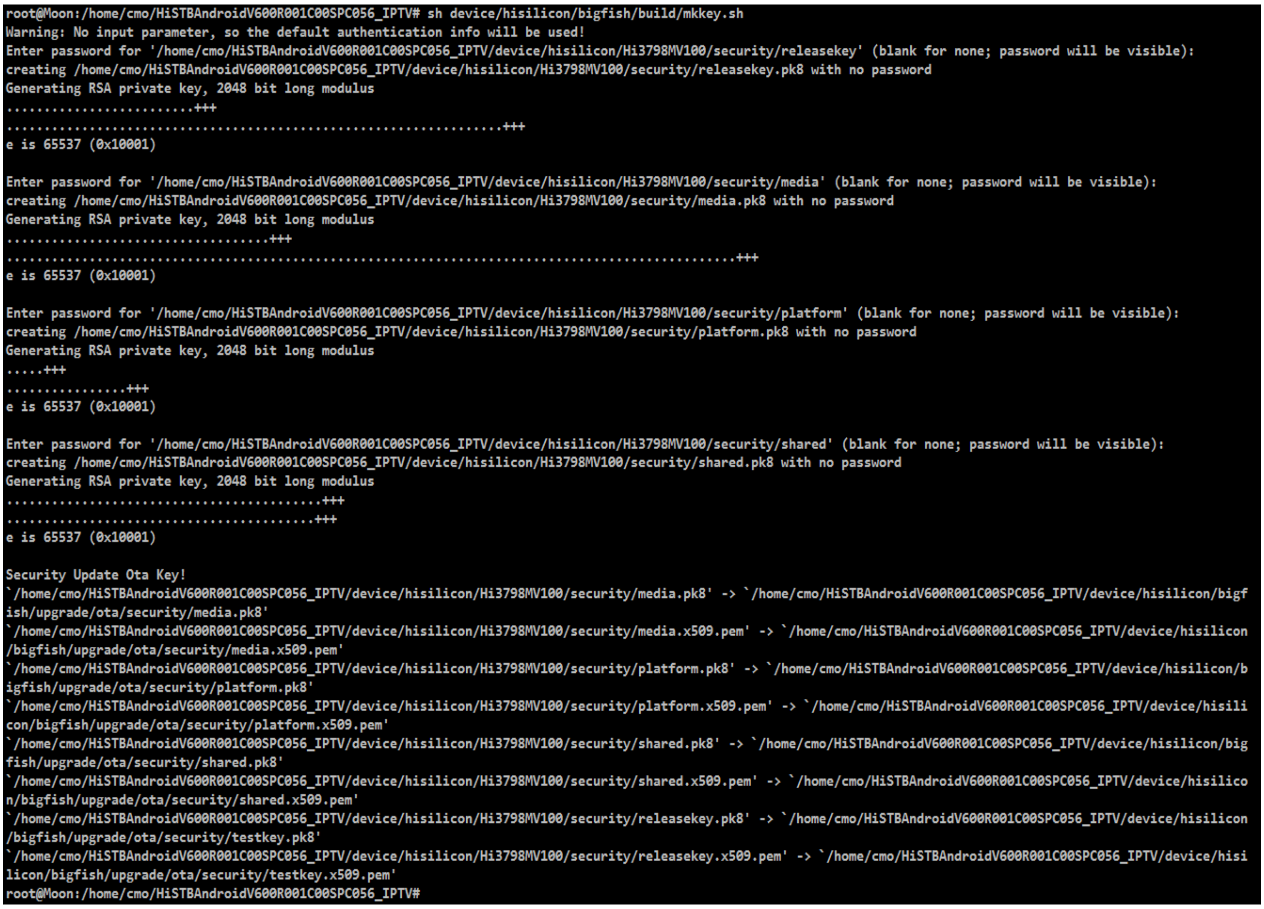
```
$ source build/envsetup.sh
$ lunch Hi3798MV100-user
```

Step 3 Run the **mkkey.sh** script.

```
$ sh device/hisilicon/bigfish/build/mkkey.sh
```

Running this script generates four pairs of keys. Copy the keys to **/device/hisilicon/bigfish/upgrade/ota/security** for creating the over-the-air (OTA) upgrade package. During the execution of this script, press **Enter** each time the information asking you to enter the password is displayed (for four times).

Figure 2-1 Displayed information asking you to enter the password



```
root@Moon:/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV# sh device/hisilicon/bigfish/build/mkkey.sh
Warning: No input parameter, so the default authentication info will be used!
Enter password for '/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/releasekey' (blank for none; password will be visible):
creating /home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/releasekey.pk8 with no password
Generating RSA private key, 2048 bit long modulus
.........................+++
..................................................................+++
e is 65537 (0x10001)

Enter password for '/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/media' (blank for none; password will be visible):
creating /home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/media.pk8 with no password
Generating RSA private key, 2048 bit long modulus
...................................+++
...............................................................................................+++
e is 65537 (0x10001)

Enter password for '/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/platform' (blank for none; password will be visible):
creating /home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/platform.pk8 with no password
Generating RSA private key, 2048 bit long modulus
.....+++
................+++
e is 65537 (0x10001)

Enter password for '/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/shared' (blank for none; password will be visible):
creating /home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/shared.pk8 with no password
Generating RSA private key, 2048 bit long modulus
..........................................+++
.........................................+++
e is 65537 (0x10001)

Security Update Ota Key!
`/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/media.pk8' -> `/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/bigf
ish/upgrade/ota/security/media.pk8'
`/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/media.x509.pem' -> `/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon
/bigfish/upgrade/ota/security/media.x509.pem'
`/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/platform.pk8' -> `/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/b
igfish/upgrade/ota/security/platform.pk8'
`/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/platform.x509.pem' -> `/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisili
con/bigfish/upgrade/ota/security/platform.x509.pem'
`/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/shared.pk8' -> `/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/big
fish/upgrade/ota/security/shared.pk8'
`/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/shared.x509.pem' -> `/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilico
n/bigfish/upgrade/ota/security/shared.x509.pem'
`/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/releasekey.pk8' -> `/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon
/bigfish/upgrade/ota/security/testkey.pk8'
`/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/releasekey.x509.pem' -> `/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisi
licon/bigfish/upgrade/ota/security/testkey.x509.pem'
root@Moon:/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV#
```

If you are not the user **root**, the following error information is displayed, and the keys are not generated:

```
Warning: No input parameter, so the default authentication info will be
used!
Enter password for
```

```
'/home/XXX/worksapce/v5/iptv/device/hisilicon/Hi3798MV100/security/releas
ekey' (blank for none; password will be v
creating
/home/XXX/worksapce/v5/iptv/device/hisilicon/Hi3798MV100/security/release
key.pk8 with no password
/home/XXX/worksapce/v5/iptv/development/tools/make_key: line 52:
/tmp/tmp.ciDupItP64/two: Permission denied
Can't open input file /tmp/tmp.ciDupItP64/one
Generating RSA private key, 2048 bit long modulus
.Error opening Private Key /tmp/tmp.ciDupItP64/two
140117173147296:error:0200100D:system library:fopen:Permission
denied:bss_file.c:398:fopen('/tmp/tmp.ciDupItP64/two','r')
140117173147296:error:20074002:BIO routines:FILE_CTRL:system
lib:bss_file.c:400:
unable to load Private Key
........................+++
.....................................................................+++
e is 65537 (0x10001)
```

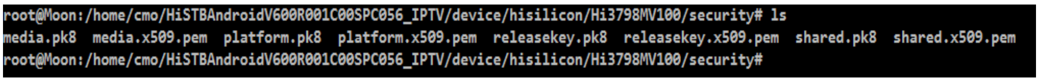
Step 4 Exit the **root** account.

```
$ exit
```

Step 5 Generate the keys.

The four pairs of keys (**releasekey**, **platform**, **shared**, and **media**) are generated in **device/hisilicon/Hi3798MV100/security/**, as shown in Figure 2-2.

Figure 2-2 Generated four pairs of keys



```
root@Moon:/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security# ls
media.pk8  media.x509.pem  platform.pk8  platform.x509.pem  releasekey.pk8  releasekey.x509.pem  shared.pk8  shared.x509.pem
root@Moon:/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security#
```

----End

The **mkkey.sh** script generates the four pairs of public or private keys: releasekey, platform, shared, and media, whose functions are described as follows:

- **platform**: This key is used to sign some APKs in the core platform. The system APK of HiSilicon STB is signed by using this platform key.
- **shared**: This key is used to sign the APKs of the shared parts in the home/contacts process.
- **media**: This key is used to sign some APKs in the media/download system.
- **releasekey**: This key is used to sign other APKs, and sign and verify the upgrading package (**update.zip**) in the system

Users may not replace the three pairs of keys (platform, shared, and media) when they copy the three pairs of Android native keys to the **device/hisilicon/Hi3798MV100/security/** directory by running the following commands under the root directory:

```
$ cp -rf build/target/product/security/platform.*
device/hisilicon/Hi3798MV100/security/
$ cp -rf build/target/product/security/shared.*
device/hisilicon/Hi3798MV100/security/
$ cp -rf build/target/product/security/media.*
device/hisilicon/Hi3798MV100/security/
```

⚠ **CAUTION**

- Typically the keys need to be generated only once. If you want to regenerate the keys, you must delete the corresponding files in **device/hisilicon/Hi3798MV100/security/** first.
- If the keys exist in **device/hisilicon/Hi3798MV100/security/**, you also need to run the **mkkey.sh** script. In this case, running this script does not generate the four pairs of keys but copies the keys to **/device/hisilicon/bigfish/upgrade/ota/security** for creating the OTA upgrade package.
- When all keys are replaced, the APKs with special permission and signed by using the native platform, shared, and media keys can be installed only after they are re-signed by using new platform, shared, and media keys. In this way, operation is more secure but complicated. If only the releasekey key is replaced, the APKs with special permission and signed by using the native platform, shared, and media keys can be installed directly. You can choose either way based on your application scenario.

2.1.2 Secure Boot Verification Keys

Images related to secure boot must be signed by using keys.

There are two pairs of secure boot verification keys:

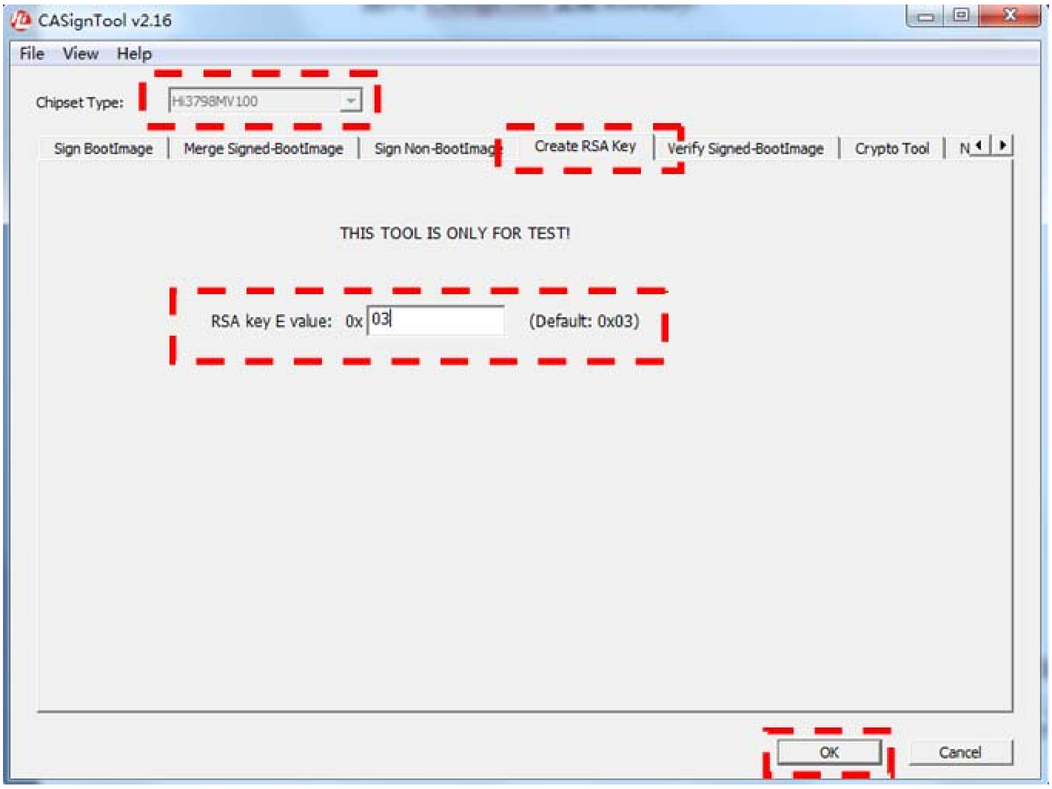
- One for verifying the fastboot signature
- One for verifying signatures of images except the fastboot image

These keys need to be generated by using the CASignTool provided by HiSilicon, which is stored in **device/hisilicon/bigfish/sdk/tools/windows/advca/CASignTool**.

To generate the two pairs of keys by using the CASignTool, perform the following steps:

Step 1 Start the CASignTool, and select a chipset from the **Chipset Type** drop-down list, for example, **Hi3798MV100**. Click the **Create RSA Key** tab, and set **RSA key E value** to the default value **03**, as shown in Figure 2-3.

Figure 2-3 Setting parameters for generating the RSA key



Chipset Type: Hi3798MV100

THIS TOOL IS ONLY FOR TEST!

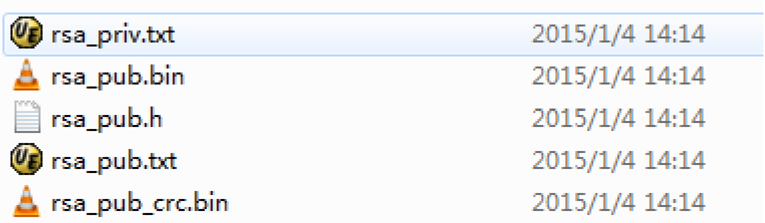
RSA key E value: 0x 03 (Default: 0x03)

CAUTION

The default value of **RSA key E value** is 0x03, and it is used for generating the RSA key pairs. You can also enter a positive hexadecimal integer (maximum 0xffffffff). However, the default value 0x03 is recommended because the required calculation increases significantly if another value is used.

Step 2 Click **OK**. A **RSA_*XXXXXXX*** directory is generated in the CASignTool directory for storing the generated keys, as shown in Figure 2-4.

Figure 2-4 Generated keys



rsa_priv.txt	2015/1/4 14:14
rsa_pub.bin	2015/1/4 14:14
rsa_pub.h	2015/1/4 14:14
rsa_pub.txt	2015/1/4 14:14
rsa_pub_crc.bin	2015/1/4 14:14

A pair of keys is generated for verifying the fastboot signature.

- Rename **rsa_pub.txt root_rsa_pub.txt** (.txt key file for verifying the fastboot signature, which is not used currently).

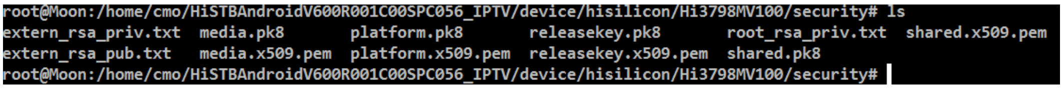
- Rename **rsa_priv.txt root_rsa_priv.txt** (.txt key file for verifying the fastboot signature, which needs to be copied to the specified directory).
- Rename **rsa_pub.bin root_rsa_pub.bin** (binary key file for verifying the fastboot signature, which is used when the key is burnt in the command-line interface (CLI). For details, see section 3.4 "Checking the CA Chip Status").
- Rename **rsa_pub_crc.bin root_rsa_pub_crc.bin** (binary key file generated after CRC signature on **root_rsa_pub.bin**, used for USB burning. For details, see section 3.1.1 "Burning Images to a Bare Chip by Using a USB Flash Drive" and 3.1.2 "Burning Images to a Non-Bare Chip by Using a USB Flash Drive").

Step 3 Click **OK** again to generate another pair of keys for verifying the signatures of the bootargs, recovery, and kernel images.

- Rename **rsa_pub.txt extern_rsa_pub.txt** (.txt key file for verifying image signatures, which needs to be copied to the specified directory).
- Rename **rsa_priv.txt extern_rsa_priv.txt** (.txt key file for verifying image signatures, which needs to be copied to the specified directory).

Step 4 Copy **extern_rsa_pub.txt**, **extern_rsa_priv.txt**, and **root_rsa_priv.txt** to **device/hisilicon/Hi3798MV00/security/**.

Figure 2-5 device/hisilicon/Hi3798MV00/security/ with all generated keys



```
root@Moon:/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security# ls
extern_rsa_priv.txt  media.pk8       platform.pk8       releasekey.pk8       root_rsa_priv.txt  shared.x509.pem
extern_rsa_pub.txt   media.x509.pem  platform.x509.pem  releasekey.x509.pem  shared.pk8
root@Moon:/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security#
```

----End

2.2 Hardware Configuration File

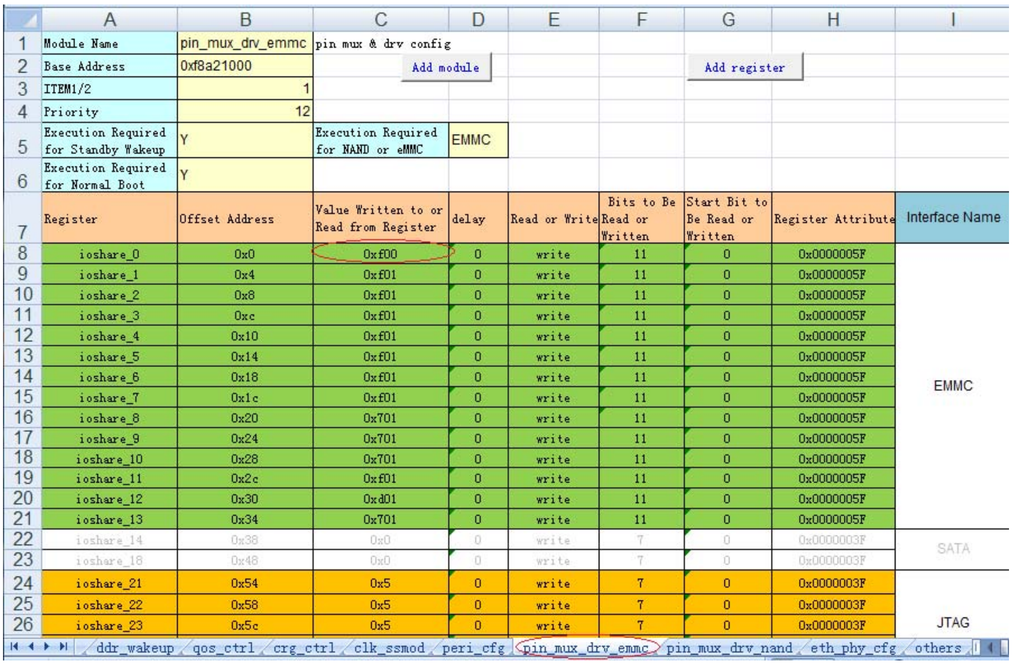
The hardware parameter configuration file is related to the board hardware. You need to modify it based on the board type.

Take the Hi3798M DMO1B board (Hi3798M V100) as an example. The corresponding hardware configuration table is **hi3798mdmo1b_hi3798mv100_ddr3_1gbyte_16bitx2_2layers.xlsm** in **device/hisilicon/bigfish/sdk/source/boot/sysreg/**.

Take the eMMC flash as an example. The corresponding hardware parameter configuration file is **hi3798mdmo1b_hi3798mv100_ddr3_1gbyte_16bitx2_2layers_emmc.cfg**.

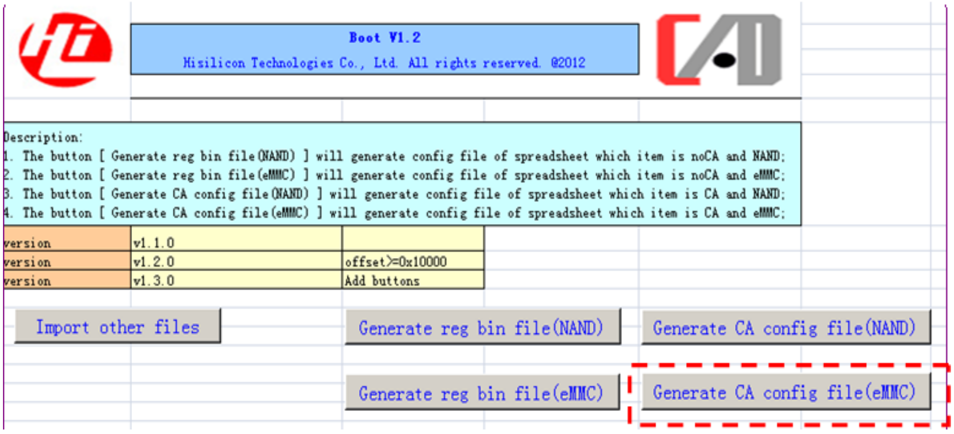
The default parameter configuration file supports only the board with the eMMC flash. To support the board with the SD flash, change **0xf01** in the red ellipse of the **pin_mux_drv_emmc** sheet in Figure 2-6 to **0xf00**, and then click **Generate CA config file(eMMC)** in the hardware configuration table shown in Figure 2-7 to generate the parameter configuration file in the current directory.

Figure 2-6 Hardware parameter configuration for supporting the SD flash



	A	B	C	D	E	F	G	H	I
1	Module Name	pin_mux_drv_emmc	pin mux & drv config						
2	Base Address	0xf8a21000							
3	ITEM1/2	1							
4	Priority	12							
5	Execution Required for Standby Wakeup	Y	Execution Required for NAND or eMMC	EMMC					
6	Execution Required for Normal Boot	Y							
7	Register	Offset Address	Value Written to or Read from Register	delay	Read or Write	Bits to Be Read or Written	Start Bit to Be Read or Written	Register Attribute	Interface Name
8	ioshare_0	0x0	0xf00	0	write	11	0	0x0000005F	EMMC
9	ioshare_1	0x4	0xf01	0	write	11	0	0x0000005F	
10	ioshare_2	0x8	0xf01	0	write	11	0	0x0000005F	
11	ioshare_3	0xc	0xf01	0	write	11	0	0x0000005F	
12	ioshare_4	0x10	0xf01	0	write	11	0	0x0000005F	
13	ioshare_5	0x14	0xf01	0	write	11	0	0x0000005F	
14	ioshare_6	0x18	0xf01	0	write	11	0	0x0000005F	
15	ioshare_7	0x1c	0xf01	0	write	11	0	0x0000005F	
16	ioshare_8	0x20	0x701	0	write	11	0	0x0000005F	
17	ioshare_9	0x24	0x701	0	write	11	0	0x0000005F	
18	ioshare_10	0x28	0x701	0	write	11	0	0x0000005F	
19	ioshare_11	0x2c	0xf01	0	write	11	0	0x0000005F	
20	ioshare_12	0x30	0xd01	0	write	11	0	0x0000005F	
21	ioshare_13	0x34	0x701	0	write	11	0	0x0000005F	
22	ioshare_14	0x38	0x0	0	write	7	0	0x0000003F	SATA
23	ioshare_18	0x48	0x0	0	write	7	0	0x0000003F	
24	ioshare_21	0x54	0x5	0	write	7	0	0x0000003F	JTAG
25	ioshare_22	0x58	0x5	0	write	7	0	0x0000003F	
26	ioshare_23	0x5c	0x5	0	write	7	0	0x0000003F	

Figure 2-7 Generating the parameter configuration file



Boot V1.2
Hisilicon Technologies Co., Ltd. All rights reserved. @2012

Description:
1. The button [Generate reg bin file(NAND)] will generate config file of spreadsheet which item is noCA and NAND;
2. The button [Generate reg bin file(eMMC)] will generate config file of spreadsheet which item is noCA and eMMC;
3. The button [Generate CA config file(NAND)] will generate config file of spreadsheet which item is CA and NAND;
4. The button [Generate CA config file(eMMC)] will generate config file of spreadsheet which item is CA and eMMC;

version	v1.1.0	
version	v1.2.0	offset>=0x10000
version	v1.3.0	Add buttons

Import other files
Generate reg bin file(NAND)
Generate CA config file(NAND)
Generate reg bin file(eMMC)
Generate CA config file(eMMC)

In **device/hisilicon/Hi3798MV100/BoardConfig.mk**, the configuration items are as follows (corresponding to the used files):

- HISI_SDK_ANDROID_CFG=hi3798mdmo_hi3798mv100_android_cfg.mak
- HISI_SDK_SECURE_CFG=hi3798mdmo_hi3798mv100_android_security_cfg.mak
- HISI_SDK_RECOVERY_CFG=hi3798mdmo_hi3798mv100_android_recovery_cfg.mak
- EMMC_BOOT_CFG_NAME=hi3798mdmo1b_hi3798mv100_ddr3_1gbyte_16bitx2_2layers_emmc.cfg
- EMMC_BOOT_REG_NAME=hi3798mdmo1b_hi3798mv100_ddr3_1gbyte_16bitx2_2layers_emmc.reg

- NAND_BOOT_CFG_NAME=hi3798mdmo1b_hi3798mv100_ddr3_1gbyte_16bitx2_2layers_nand.cfg
- NAND_BOOT_REG_NAME=hi3798mdmo1b_hi3798mv100_ddr3_1gbyte_16bitx2_2layers_nand.reg

2.3 System Service Protection

2.3.1 Function Overview

The system service protection function is used to protect the original files in the system partition, preventing the files from being tampered with. If the original files are modified, the board is restarted.

2.3.2 Function Configuration

The system service protection function is disabled by default. To enable this function, modify the corresponding configuration item of **hi3798mdmo_hi3798mv100_android_security_cfg.mak** in **device/hisilicon/bigfish/sdk/configs/hi3798mv100**.

Take the Hi3798M V100 DMO1B board as an example. Set the configuration item **CFG_HI_ANDROID_SECURITY_L2_SYSTEM_CHECK** in **hi3798mdmo_hi3798mv100_android_security_cfg.mak** to **y** and then compile the version.

2.4 Compilation

To compile the version, perform the following steps:

Step 1 Set **HISILICON_SECURITY_L2** to **true** in **device/hisilicon/Hi3798MV100/customer.mk**.

```
HISILICON_SECURITY_L2 := true
```

Step 2 Set **HISILICON_SECURITY_L2_COMMON_MODE_SIGN** in **device/hisilicon/Hi3798MV100/customer.mk** as required:

- If the recovery and kernel images need to be signed in special mode during compilation, set **HISILICON_SECURITY_L2_COMMON_MODE_SIGN** to **false** (default value).
- If the recovery and kernel images need to be signed in common mode, set it to **true**.
- You are advised to set it to **false** for later versions. The setting is as follows:

```
HISILICON_SECURITY_L2_COMMON_MODE_SIGN := false
```

Step 3 Go to the root directory of the code, and run the following command in the CLI:

```
$ source build/envsetup.sh
```

Step 4 Run **$ lunch Hi3798MV100-user**.

Step 5 Run **$ make bigfish –j 2>&1 | tee bigfish.log**.

After compilation, images for two different purposes are generated in **out/target/product/Hi3798MV100/Security_L2/**:

- **out/target/product/Hi3798MV100/Security_L2/PRODUCTION/**: production directory
- **out/target/product/Hi3798MV100/Security_L2/MAINTAIN/**: maintenance directory

fastboot.bin, **bootargs.bin**, **recovery.img**, and **kernel.img** in the **MAINTAIN** directory are the signed images for the corresponding images in the **PRODUCTION** directory. The other files in the two directories are the same.

- For details about how to use the images in the two directories, see chapter 3 "Burning Images for Mass Production."
- You are advised not to rename those images. If **fastboot.bin**, **bootargs.bin**, **recovery.img**, and **kernel.img** in the **PRODUCTION** directory are renamed by modifying the compilation script, you need to make some modifications before compilation. For example:
  - If **fastboot.bin** in the **PRODUCTION** directory is renamed **fastboot_product.bin**, modify **Signboot_config.cfg** in **device/hisilicon/bigfish/security/secureSignTool/etc** as follows:

    Change **BootFile=fastboot.bin** to **BootFile=fastboot_product.bin**.
  - If **bootargs.bin** in the **PRODUCTION** directory is renamed **bootargs_product.bin**, modify **common_bootargs_config.cfg** in **device/hisilicon/bigfish/security/secureSignTool/etc** as follows:

    Change **InputFile=bootargs.bin** to **InputFile=bootargs_product.bin**.
  - If **recovery.bin** in the **PRODUCTION** directory is changed to **recovery_product.img**, modify **special_recovery_config.cfg** in **device/hisilicon/bigfish/security/secureSignTool/etc** as follows:

    Change **Image1=recovery.img** to **Image1= recovery_product.img**.
  - If **kernel.bin** in the **PRODUCTION** directory is changed to **kernel_product.img**, modify **special_kernel_config.cfg** in **device/hisilicon/bigfish/security/secureSignTool/etc** as follows:

    Change **Image1= kernel.img** to **Image1= kernel_product.img**.

----End

3 Burning Images for Mass Production

The Hi3798M is not considered a CA chip or a non-CA chip upon delivery.

- If the HiSilicon level 2 security solution is used to compile the secure version, the chip is considered a CA chip.
- If the HiSilicon level 2 security solution is not used to compile the version, the chip is considered a non-CA chip.

⚠ **CAUTION**

Once a chip is considered a non-CA chip, it cannot be changed to a CA chip, and the CA chip version cannot be burnt; vice versa. Therefore, burn the chip version with caution. For details about how to distinguish between the CA chip and non-CA chip, see section 3.4 "Checking the CA Chip Status."

3.1 Burning Images by Using a USB Flash Drive

The HiSilicon chips support burning by using a USB device, which is easy to use and applies to mass production.

⚠ **CAUTION**

- Burning by using a USB device is strongly recommended for mass production.
- The USB device must use the USB 2.0 port and FAT32 file system. For details about the requirements, see the Mass Production Burning User Guide.

3.1.1 Burning Images to a Bare Chip by Using a USB Flash Drive

Prepare the following items before burning:

- **fastboot.bin**, **bootargs.bin**, **recovery.img**, and **update.zip** in the **PRODUCTION** directory
- **root_rsa_pub_crc.bin** generated by the CASignTool
- USB flash drive with the FAT32 file system

To burn images to a bare chip by using the USB flash drive, perform the following steps:

Step 1 Copy **fastboot.bin**, **bootargs.bin**, **recovery.img**, **update.zip**, and **root_rsa_pub_crc.bin** to the root directory of the USB flash drive.

Step 2 Insert the USB flash drive to the USB 2.0 port.

Step 3 Power on the board. The burning process automatically starts. The indicator blinks during the burning process and is steady on after the burning is complete.

----End

3.1.2 Burning Images to a Non-Bare Chip by Using a USB Flash Drive

Restart the board after images are burnt to a bare chip by using a USB flash drive. The chip is considered a CA chip.

To reburn images to a CA chip, prepare the following items:

- **fastboot.bin**, **bootargs.bin**, **recovery.img**, and **update.zip** in the **MAINTAIN** directory
- **root_rsa_pub_crc.bin** generated by the CASignTool
- USB flash drive with the FAT32 file system

Step 1 Copy **fastboot.bin**, **bootargs.bin**, **recovery.img**, **update.zip**, and **root_rsa_pub_crc.bin** to the root directory of the USB flash drive.

Step 2 Insert the USB flash drive to the USB 2.0 port.

Step 3 Power on the board while holding down the USB burning button to enter the burning process. The indicator blinks during the burning process and is steady on after the burning is complete.

----End

⚠ **CAUTION**

- If **root_rsa_pub_crc.bin** is not in the USB flash drive when images are burnt to the bare chip or non-bare chip by using a USB flash drive, the OTP key and security flag need to be burnt by using the factory test program. For details, see section 3.3 "Burning the Secure Boot Flag and OTP Key in the Command Line."
- When images are burnt to a chip with burnt images, you need to hold down the USB burning button while powering on the board; otherwise, the USB burning process does not start. You can also erase the fastboot by running **mw.b 1000000 ff 800** and **mmc write 0 1000000 0 100** under fastboot. In this case, the chip enters the bare chip operation mode, and the process of burning images to a bare chip by using the USB flash drive is started.

3.2 Burning Images by Using the Burner

To burn images by using the burner, prepare the following items:

- Images to be burnt by the burner
  - Unsigned **fastboot.bin** in the **PRODUCTION** directory
  - System images except **fastboot.bin** in the **MAINTAIN** directory
- Factory test program

  The factory test program is developed based on the production line process. The following operations need to be implemented:
  - Burn the signed **fastboot.bin** in the **MAINTAIN** directory.
  - Burn the OTP key (**root_rsa_pub.bin** generated by using the CASignTool).
  - Burn the secure boot flag.

⚠ **CAUTION**

- The NAND flash is not supported currently.
- The factory test program is implemented by the customer. For details about how to burn the OTP key and secure boot flag, see section 3.3 "Burning the Secure Boot Flag and OTP Key in the Command Line." Burn the signed **fastboot.bin** by calling the flash read/write interfaces provided by HiSilicon.

3.3 Burning the Secure Boot Flag and OTP Key in the Command Line

HiSilicon provides two samples for burning the secure boot flag and OTP key: **device/hisilicon/bigfish/sdk/sample/advca/sample_ca_writeRSAkey.c** and **device/hisilicon/bigfish/sdk/sample/advca/sample_ca_opensecboot.c**.

To burn the OTP key, perform the following steps:

- Perform step 1 to step 2 in section "When all keys are replaced, the APKs with special permission and signed by using the native platform, shared, and media keys can be installed only after they are re-signed by using new platform, shared, and media keys. In this way, operation is more secure but complicated. If only the releasekey key is replaced, the APKs with special permission and signed by using the native platform, shared, and media keys can be installed directly. You can choose either way based on your application scenario.

Step 1 " to generate **root_rsa_pub.bin** by using the CASignTool.

Step 2 Push **root_rsa_pub.bin** to the **sdcard** directory of the board by using the Android debug bridge (ADB).

Step 3 Run the following command in the command line of the board over the serial port:

```
sample_ca_writeRSAkey /sdcard/root_rsa_pub.bin
```

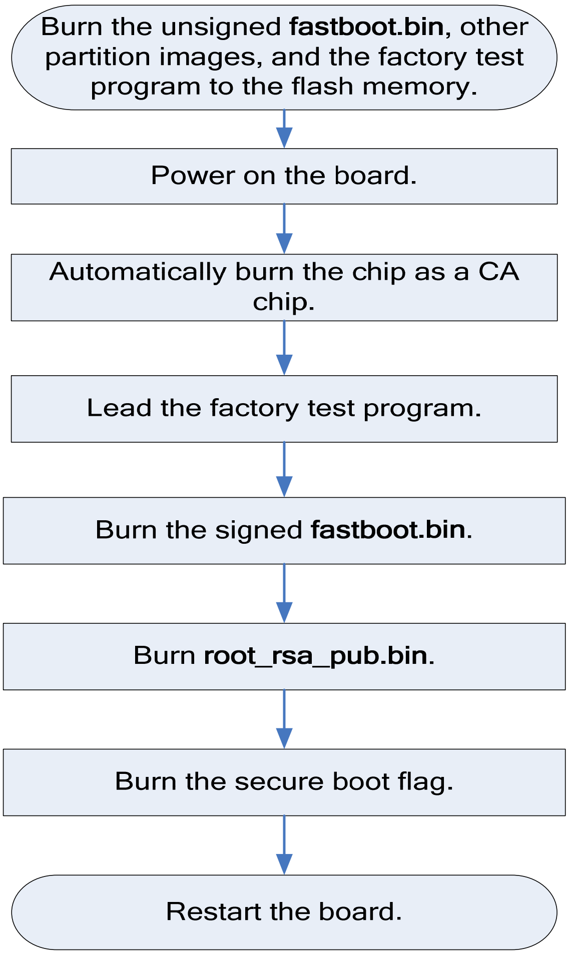
----End

To burn the secure boot flag:

- If the eMMC flash is used, run the following command on the board over the serial port:
```
sample_ca_opensecboot  emmc
```
- If the SD flash is used, run the following command on the board over the serial port:
```
sample_ca_opensecboot  sd
```

Figure 3-1 shows the process of burning images to a bare chip.

Figure 3-1 Burning images to a bare chip


Burn the unsigned **fastboot.bin**, other partition images, and the factory test program to the flash memory.
Power on the board.
Automatically burn the chip as a CA chip.
Lead the factory test program.
Burn the signed **fastboot.bin**.
Burn **root_rsa_pub.bin**.
Burn the secure boot flag.
Restart the board.

CAUTION

If the factory test program does not run successfully the first time, the system cannot be started. In this case, perform operations in section 3.1.2 "Burning Images to a Non-Bare Chip by Using a USB Flash Drive."

3.4 Checking the CA Chip Status

3.4.1 By Reading OTP Registers

To check the CA chip status by reading OTP registers, perform the following steps:

Step 1 Run **otpreadall** under the boot or run **cat /proc/msp/otp** after the system starts to obtain the register values.

```
0000 00000000 00800000 00000000 00000001
0010 00480400 00000000 00000000 00000000
0020 00000000 00000000 00000000 00000000
0030 00000000 00000000 00000000 00000000
0040 00000000 00000000 00000000 00000000
0050 00000000 00000000 00000000 00000000
0060 00000000 00000000 00000000 00000000
0070 00000000 00000000 00000000 00000000
0080 00000000 00000000 00000000 00000000
0090 00000000 00000000 00000000 00000000
00a0 00000000 00000000 2a13c812 00000000
00b0 00000000 00000000 00000000 00000000
00c0 00000000 00000000 00000000 00000000
00d0 00000000 00000000 00000000 00000000
00e0 00000000 00000000 00000000 00000000
00f0 92022090 0080e610 00000000 00000000
0100 00000000 00000000 00000000 00000000
0110 00000000 00000000 00000000 00000000
0120 00000000 00000000 00000000 00000000
0130 00000000 00000000 00000000 00000000
0140 00000000 00000000 00000000 00000000
0150 00000000 00000000 00000000 00000000
0160 00000000 00000000 00000000 00000000
0170 00000000 00000000 00000000 00000000
0180 00000000 00000000 00000000 00000000
0190 00000000 00000000 00000000 00000000
01a0 00000000 00000000 00000000 00000000
01b0 00000000 00000000 00000000 00000000
01c0 00000000 00000000 00000000 00000000
01d0 00000000 00000000 00000000 00000000
01e0 00000000 00000000 00000000 00000000
01f0 00000000 00000000 00000000 00000000
0200 00000000 00000000 00000000 00000000
```

Step 2 Determine whether the chip is a CA chip.

Check the value in address 00a8 to 00ab, that is, the four bytes in the third column of 00a0. It is the ID_WORD (ID for distinguishing between the CA chip and non-CA chip).

- If **ID_WOPD** is **0x2a13c812**, the chip is a non-CA chip.
- If **ID_WOPD** is **0x6edbe953**, the chip is a CA chip.

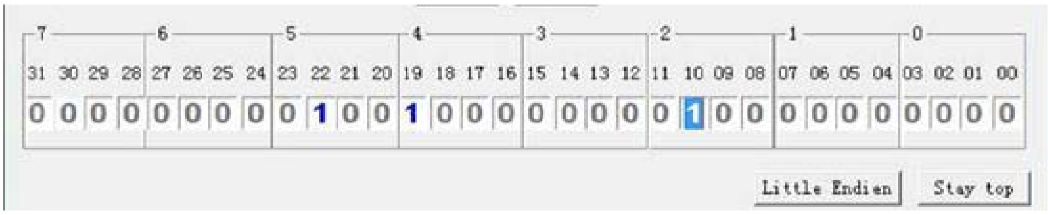
Step 3 Check whether ID_WORD is locked.

If bit[10] of 0010 (bit[2] of byte 0x11 in the OTP) is **0**, ID_WORD is not locked; if it is **1**, ID_WORD is locked.

For example, for 00480400, bit[10] is **1**, indicating that ID_WORD is locked and cannot be changed.

Figure 3-2 shows the binary format of 0x00480400.

Figure 3-2 Converting 0x00480400 into the binary format



7				6				5				4				3				2				1				0			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

CAUTION

After images are burnt to a chip and the chip starts for the first time, the ID_WORD of the chip is locked and cannot be changed.

Step 4 Check whether the OTP key (**root_rsa_pub.bin**) is locked.

Bit[0] of byte 0x14 in the OTP indicates whether the OTP key is locked. The value **0** indicates not locked, and the value **1** indicates locked.

The OTP key can be burnt only once. Therefore, ensure that the OTP key is correct before burning it.

Step 5 Check whether the secure boot flag is enabled.

Bit[0] of byte 0x18 in the OTP indicates whether the secure boot flag is enabled. The value **0** indicates disabled, and the value **1** indicates enabled.

Before enabling the secure boot flag, ensure that the OTP key is burnt properly; otherwise, the system cannot start.

----End

3.4.2 By Viewing the Log Information During System Startup

The following describes the information displayed over the serial port for a CA chip and a non-CA chip.

- The log information displayed when a CA chip starts is as follows (pay attention to the characters in red):

```
System startup
```

```
Reg Version:  v1.1.0
Reg Time:    2014/12/5  11:09:01
Reg Name:
hi3798mdmo1c_hi3798mv100_ddr3_1gbyte_16bitx2_2layers_emmc.reg

Relocate Boot to DDR

Jump to DDR

Compressed-boot v1.0.0
Uncompress......................Ok


System startup

Reg Version:  v1.1.0
Reg Time:    2014/12/5  11:09:01
Reg Name:
hi3798mdmo1c_hi3798mv100_ddr3_1gbyte_16bitx2_2layers_emmc.reg

Relocate Boot to DDR

Jump to DDR

Fastboot 3.3.0 (wangmengfu@hidolphin154) (Dec 16 2014 - 15:20:50)

Fastboot:     Version 3.3.0
Build Date:   Dec 16 2014, 15:20:58
CPU:         Hi3798Mv100 (CA)  /*CA chip*/
Boot Media:   eMMC
DDR Size:     1GB

Check nand flash controller v610. found
Special NAND id table Version 1.36
Nand ID: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
No NAND device found!!!

MMC/SD controller initialization.
MMC/SD Card:
    MID:        0x45
    Read Block:  512 Bytes
    Write Block: 512 Bytes
    Chip Size:   3776M Bytes (High Capacity)
    Name:       "SEM04"
    Chip Type:   MMC
    Version:    4.5
    Speed:      52000000Hz
    Bus Width:   8bit
    Boot Addr:   0 Bytes

Boot Env on eMMC
    Env Offset:         0x00100000
    Env Size:           0x00010000
    Env Range:          0x00010000
HI_OTP_LockIdWord,313: ID_WORD have already been locked
/*ID_WORD is locked.*/

SDK Version: HiSTBLinuxV100R002

Security Begin Read RSA Key!
Secure boot is enabled  /*The OTP key is burnt and the secure boot
flag is enabled.*/
press the key!!
get key 0 2
get chipid =137980100
get chipType (HI3798MV100)
count=2
mac:32:31:74:B6:26:7B

MMC read: dev # 0, block # 1, count 1 ... 1 blocks read: OK
```

- The log information displayed when a non-CA chip starts is as follows (pay attention to the characters in red):

```
System startup

Reg Version:  v1.1.0
Reg Time:    2014/12/5  11:09:27
Reg Name:
hi3798mdmo1b_hi3798mv100_ddr3_1gbyte_16bitx2_2layers_emmc.reg

Relocate Boot to DDR

Jump to DDR

Fastboot 3.3.0 (wangmengfu@hidolphin154) (Dec 12 2014 - 19:16:27)

Fastboot:     Version 3.3.0
```

```
Build Date:    Dec 12 2014, 19:16:36
CPU:          Hi3798Mv100   /*Non-CA chip*/
Boot Media:    eMMC
DDR Size:      1GB
Check nand flash controller v610. found
Special NAND id table Version 1.36
Nand ID: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
No NAND device found!!!

MMC/SD controller initialization.
MMC/SD Card:
    MID:         0x45
    Read Block:  512 Bytes
    Write Block: 512 Bytes
    Chip Size:   3776M Bytes (High Capacity)
    Name:        "SEM04"
    Chip Type:   MMC
    Version:     4.5
    Speed:       52000000Hz
    Bus Width:   8bit
    Boot Addr:   0 Bytes

Boot Env on eMMC
    Env Offset:        0x00100000
    Env Size:          0x00010000
    Env Range:         0x00010000
HI_OTP_LockIdWord,313: ID_WORD have already been locked
/*The chip ID_WORD is locked.*/

SDK Version: HiSTBLinuxV100R002
/*No information related to secure boot is displayed.*/
press the key!!
get key 0 2
get chipid =137980100
get chipType (HI3798MV100)
count=2
mac:3E:77:2F:81:C1:DA
```

4 Burning Images by Using the HiTool

The HiTool burning functions are used for development debugging of R&D engineers.

Images can be burnt to a bare chip or a non-bare chip by using the HiTool.

After images are burnt by using the HiTool, burn the secure boot flag and OTP key by following instructions in section 3.3 "Burning the Secure Boot Flag and OTP Key in the Command Line."

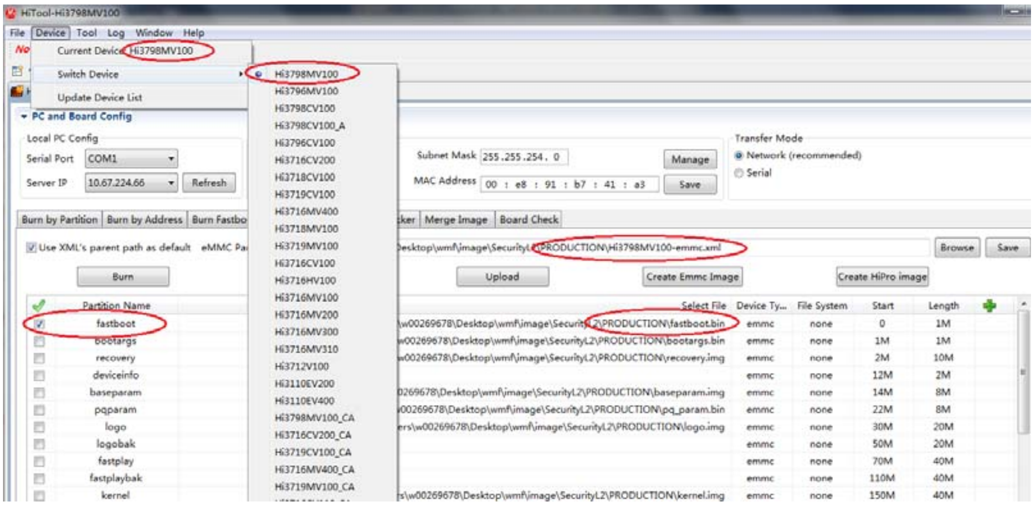
4.1 Burning Images to a Bare Chip by Using the HiTool

To burn images to a bare chip by using the HiTool, perform the following steps:

Step 1 Burn the unsigned **fastboot.bin** in the **PRODUCTION** directory.

- Choose **Device** > **Switch Device**, and select **Hi3798MV100**.
- Burn only **fastboot.bin** in the **PRODUCTION** directory.
- Start burning the image. After the burning is complete, power on the board. The chip is burnt as a CA chip.

Figure 4-1 Configuring the HiTool (1)



Serial Port COM1

Server IP 10.67.224.66

Subnet Mask 255.255.254. 0

MAC Address 00 : e8 : 91 : b7 : 41 : a3

● Network (recommended)
○ Serial

☑ Use XML's parent path as default   eMMC Pa

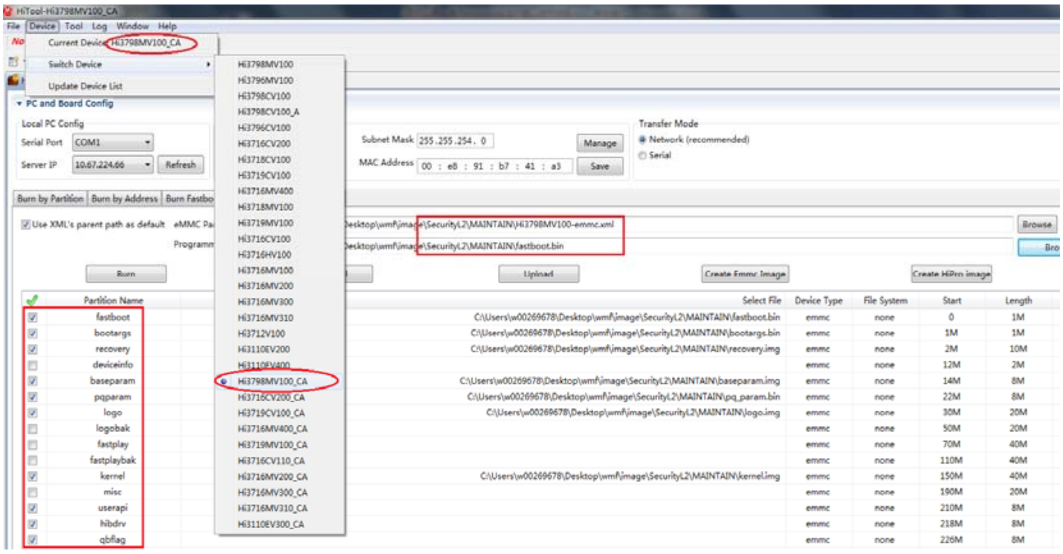
esktop\wmf\image\SecurityL2\PRODUCTION\Hi3798MV100-emmc.xml

✓	Partition Name	Select File	Device Ty...	File System	Start	Length
☑	fastboot	\w00269678\Desktop\wmf\image\SecurityL2\PRODUCTION\fastboot.bin	emmc	none	0	1M
□	bootargs	w00269678\Desktop\wmf\image\SecurityL2\PRODUCTION\bootargs.bin	emmc	none	1M	1M
□	recovery	w00269678\Desktop\wmf\image\SecurityL2\PRODUCTION\recovery.img	emmc	none	2M	10M
□	deviceinfo		emmc	none	12M	2M
□	baseparam	0269678\Desktop\wmf\image\SecurityL2\PRODUCTION\baseparam.img	emmc	none	14M	8M
□	pqparam	00269678\Desktop\wmf\image\SecurityL2\PRODUCTION\pq_param.bin	emmc	none	22M	8M
□	logo	ers\w00269678\Desktop\wmf\image\SecurityL2\PRODUCTION\logo.img	emmc	none	30M	20M
□	logobak		emmc	none	50M	20M
□	fastplay		emmc	none	70M	40M
□	fastplaybak		emmc	none	110M	40M
□	kernel	s\w00269678\Desktop\wmf\image\SecurityL2\PRODUCTION\kernel.img	emmc	none	150M	40M

Step 2 Burn other images.

- Choose **Device** > **Switch Device**, and select **Hi3798MV100_CA**.
- Select the signed **fastboot.bin** in the **MAINTAIN** directory as the programmer file.
- Specify the partition table and images. Use the images in the **MAINTAIN** directory.
- Select all partitions and corresponding images.
- Start burning the images.

Figure 4-2 Configuring the HiTool (2)



Serial Port: COM1

Server IP: 10.67.224.66

Subnet Mask: 255 .255 .254 . 0

MAC Address: 00 : e8 : 91 : b7 : 41 : a3

Transfer Mode: ● Network (recommended) ○ Serial

☑ Use XML's parent path as default

eMMC Pa… : …esktop\wmf\image\SecurityL2\MAINTAIN\Hi3798MV100-emmc.xml

Programm… : …esktop\wmf\image\SecurityL2\MAINTAIN\fastboot.bin

✓	Partition Name	Select File	Device Type	File System	Start	Length
☑	fastboot	C:\Users\w00269678\Desktop\wmf\image\SecurityL2\MAINTAIN\fastboot.bin	emmc	none	0	1M
☑	bootargs	C:\Users\w00269678\Desktop\wmf\image\SecurityL2\MAINTAIN\bootargs.bin	emmc	none	1M	1M
☑	recovery	C:\Users\w00269678\Desktop\wmf\image\SecurityL2\MAINTAIN\recovery.img	emmc	none	2M	10M
□	deviceinfo		emmc	none	12M	2M
☑	baseparam	C:\Users\w00269678\Desktop\wmf\image\SecurityL2\MAINTAIN\baseparam.img	emmc	none	14M	8M
☑	pqparam	C:\Users\w00269678\Desktop\wmf\image\SecurityL2\MAINTAIN\pq_param.bin	emmc	none	22M	8M
☑	logo	C:\Users\w00269678\Desktop\wmf\image\SecurityL2\MAINTAIN\logo.img	emmc	none	30M	20M
□	logobak		emmc	none	50M	20M
□	fastplay		emmc	none	70M	40M
□	fastplaybak		emmc	none	110M	40M
☑	kernel	C:\Users\w00269678\Desktop\wmf\image\SecurityL2\MAINTAIN\kernel.img	emmc	none	150M	40M
□	misc		emmc	none	190M	20M
☑	userapi		emmc	none	210M	8M
☑	hibdrv		emmc	none	218M	8M
☑	qbflag		emmc	none	226M	8M

----End

4.2 Burning Images to a Non-Bare Chip by Using the HiTool

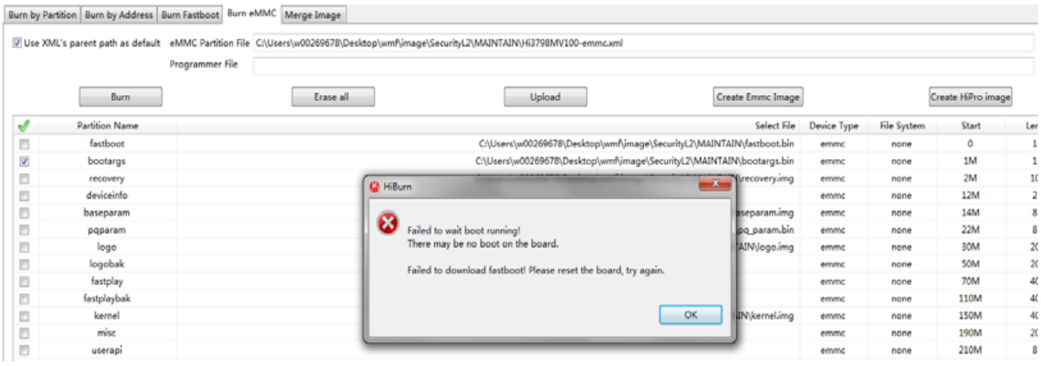
To burn images to a non-bare chip by using the HiTool, perform step 2 in section 4.1 "Burning Images to a Bare Chip by Using the HiTool."

- Choose **Device** > **Switch Device**, and select **Hi3798MV100_CA**.
- Select the signed **fastboot.bin** in the **MAINTAIN** directory as the programmer file.
- Specify the partition table and images. Use the images in the **MAINTAIN** directory.
- Select all partitions and corresponding images.
- Start burning the images.

CAUTION

When burning images to a non-bare chip by using the HiTool, if you need to separately burn a partition except the fastboot partition, select **fastboot.bin** in the **MAINTAIN** directory as the programmer file. Otherwise, the error information "There may be no boot on the board." is displayed. As shown in Figure 4-3, when the bootargs is separately burnt and the programmer file is not specified, the HiTool displays an error message.

Figure 4-3 Separately burning the bootargs partition



☑ Use XML's parent path as default

eMMC Partition File: C:\Users\w00269678\Desktop\wmf\image\SecurityL2\MAINTAIN\Hi3798MV100-emmc.xml

Programmer File:

✓	Partition Name	Select File	Device Type	File System	Start	Le
□	fastboot	C:\Users\w00269678\Desktop\wmf\image\SecurityL2\MAINTAIN\fastboot.bin	emmc	none	0	1
☑	bootargs	C:\Users\w00269678\Desktop\wmf\image\SecurityL2\MAINTAIN\bootargs.bin	emmc	none	1M	1
□	recovery	\recovery.img	emmc	none	2M	1
□	deviceinfo		emmc	none	12M	2
□	baseparam	aseparam.img	emmc	none	14M	8
□	pqparam	pq_param.bin	emmc	none	22M	8
□	logo	AIN\logo.img	emmc	none	30M	2
□	logobak		emmc	none	50M	2
□	fastplay		emmc	none	70M	4
□	fastplaybak		emmc	none	110M	4
□	kernel	IN\kernel.img	emmc	none	150M	4
□	misc		emmc	none	190M	2
□	userapi		emmc	none	210M	8

Failed to wait boot running!
There may be no boot on the board.

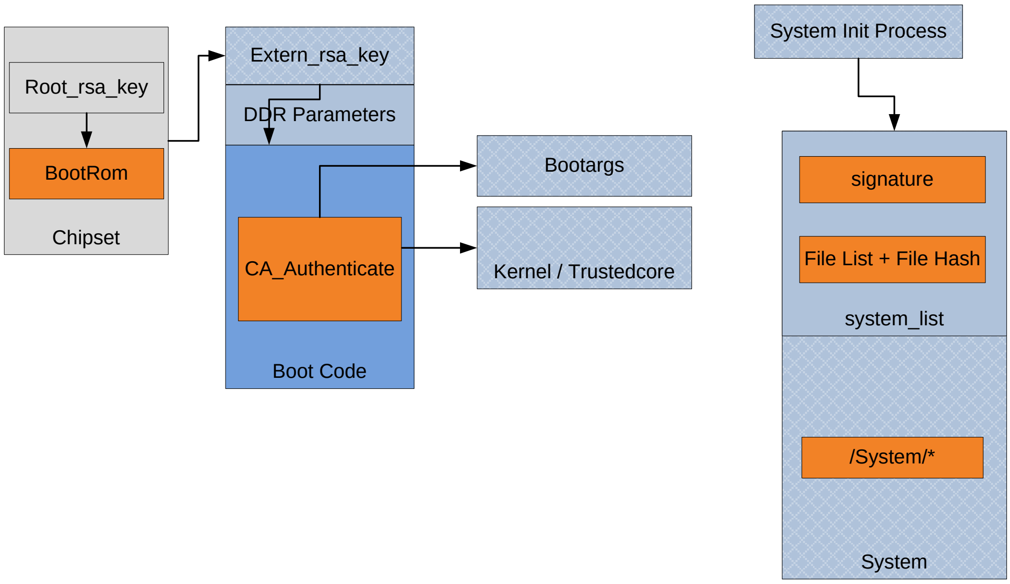
Failed to download fastboot! Please reset the board, try again.

5 Level 2 Security Solution for Hi3798C V200

The level 2 security solution for Hi3798C V200 involves modifications on the verification process, compilation, burning images for mass production, burning images by using the HiTool, and system service protection.

5.1 Solution Modification

Figure 5-1 Process of the secure boot and verification solution for Hi3798C V200



Compared with the previous security solution, the modifications are as follows:

- The recovery partition is not verified when the board is started normally, and is verified only when upgraded is triggered and the recovery partition needs to be entered.
- The system_list and system file are verified in the system init process.

- The ID_WORD is not burnt in boot, and is placed in the factory test program for burning at the factory production phase.

5.2 Compilation

5.2.1 Android System Keys

For Android system keys, **mkkey.sh** is modified, only **releasekey** is replaced, and the other three key pairs (platform, shared, and media) are directly copied from Android native keys. The APK that is signed by using Android native platform, shared, or media keys and has special permissions can be directly installed.

5.2.2 Hardware Configuration File

The hardware configuration file is not required for signing fastboot of Hi3798C V200.

5.2.3 Compilation Procedure

The secure fastboot is modified and only one set of signed images needs to be compiled. To compile the images, perform the following steps:

Step 1 Set **HISILICON_SECURITY_L2** to **true** in **device/hisilicon/Hi3798CV200/customer.mk**.

```
HISILICON_SECURITY_L2 := true
```

Step 2 Go to the root directory of the code, and run the following command in the CLI:

```
$ source build/envsetup.sh
```

Step 3 Run the following command：

```
$ lunch Hi3798CV200-user
```

Step 4 Run the following command：

```
$ make bigfish –j 2>&1 | tee bigfish.log
```

After compilation is complete, the signed images are generated in **out/target/product/Emmc/**.

The bootargs image is signed in common mode, whereas the recovery, kernel, and trustedcore images are signed in special mode.

----End

CAUTION

When the board type is changed, **BOOT_REG_NAME** in **device/hisilicon/Hi3798CV200/BoardConfig.mk** needs to be modified to match the board and chip types. The specific name can be found in **device/hisilicon/bigfish/sdk/source/boot/sysreg/**.

5.3 Burning Images for Mass Production

5.3.1 Burning and Locking ID_WORD

The ID_WORD is burnt in boot previously. However, subsequently it is changed to be burnt in the factory test program. You can burn the ID_WORD in the factory test program based on the following sample provided by HiSilicon:

device/hisilicon/bigfish/sdk/sample/otp/sample_otp_lockidword.c

The commands related to ID_WORD are as follows:

- After the system starts, run the following command over the serial port of the board to burn the secure ID_WORD and lock it:

  ```
  sample_otp_lockidword  1
  ```

- Run the following command to view the burning status of ID_WORD:

  ```
  sample_otp_lockidword  2
  ```

5.3.2 Burning Images by Using a USB Flash Drive

Prepare the following items before burning:

- **fastboot.bin**, **bootargs.bin**, **recovery.img**, and **update.zip** in the **Emmc** directory
- **root_rsa_pub_crc.bin** generated by the CASignTool
- USB flash drive with the FAT32 file system
- Factory test program

  The factory test program is developed by customers based on the production line process, and is used to burn the ID_WORD and lock it to secure chip.

To burn images by using the USB flash drive, perform the following steps:

Step 1 Copy **fastboot.bin**, **bootargs.bin**, **recovery.img**, **update.zip**, and **root_rsa_pub_crc.bin** to the root directory of the USB flash drive.

Step 2 Insert the USB flash drive to the USB 2.0 port.

Step 3 For the bare chip, power on the board, and then the board automatically enters the burning process. For the non-bare chip, press the USB burning key on the board so that the board enters the burning process after being powered on. The indicator blinks during the burning process and is steady on after the burning is complete.

Step 4 Start the board to enter the factory test program, burn the ID_WORD, and lock it to secure chip.

----End

⚠ **CAUTION**

When burning images by using the USB flash drive, you can burn the OTP key and secure boot flag in the factory test program so that **root_rsa_pub_crc.bin** does not need to be placed in the root directory of the USB flash drive.

5.3.3 Burning Images by Using a Burner

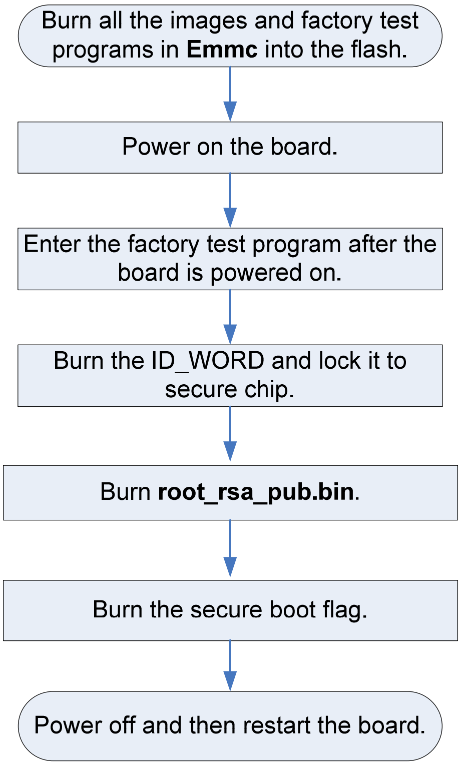
To burn images by using the burner, prepare the following items:

- Images to be burnt by the burner

  All images in the **Emmc** directory
- Factory test program

  The factory test program is developed based on the production line process. The following operations need to be implemented:
  - Burn ID_WORD and lock it to secure chip.
  - Burn the OTP key (**root_rsa_pub.bin** generated by using the CASignTool).
  - Burn the secure boot flag.

Figure 5-2 Burning images to a bare Hi3798C V200 chip


Burn all the images and factory test programs in Emmc into the flash.
Power on the board.
Enter the factory test program after the board is powered on.
Burn the ID_WORD and lock it to secure chip.
Burn root_rsa_pub.bin.
Burn the secure boot flag.
Power off and then restart the board.

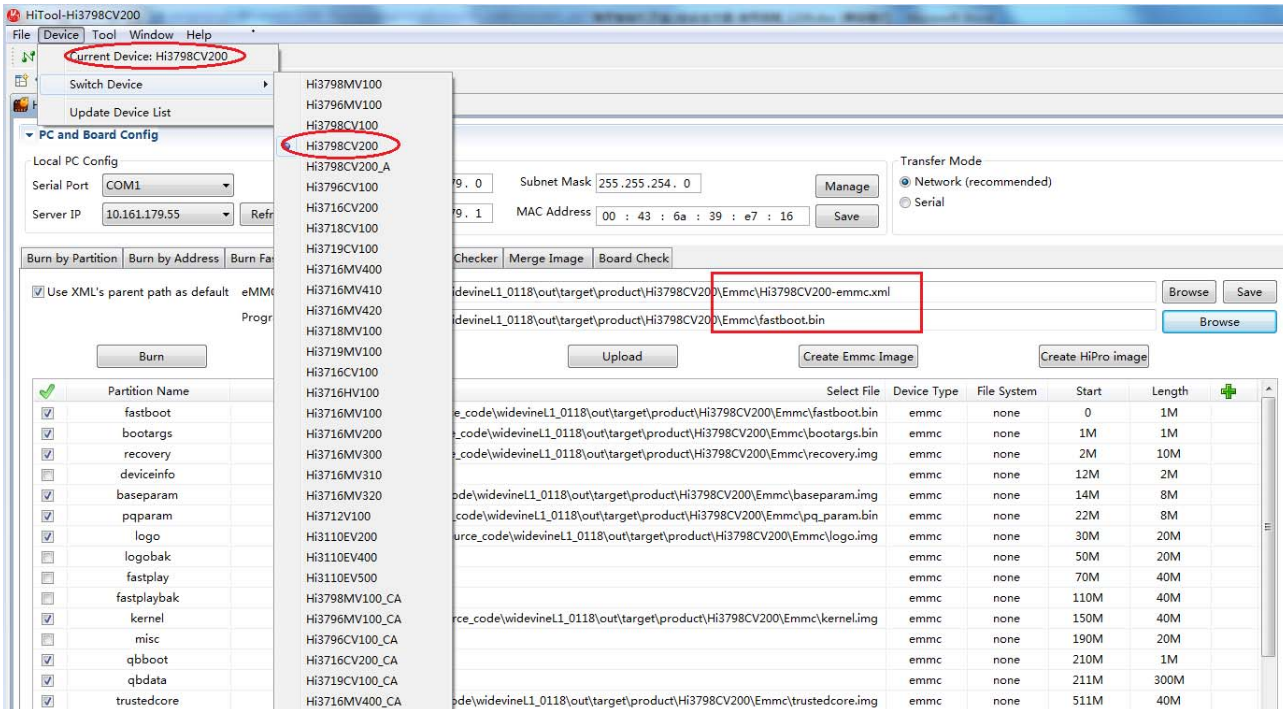
5.3.4 Burning Images by Using the HiTool

The function of burning images by using the HiTool is used for development and debugging of R&D engineers.

For Hi3798C V200, whether the bare chips and non-bare chips are not distinguished when images are burnt by using the HiTool. The configuration is as follows:

- Set the chipset type to **Hi3798CV200**.
- Select the signed **fastboot.bin** in the **Emmc** directory as the programmer file.
- Specify the partition table and images. Use the images in the **Emmc** directory.
- Select all partitions and corresponding images.
- Start burning the images.

Figure 5-3 Configuring the HiTool for Hi3798C V200



Local PC Config
Serial Port: COM1
Server IP: 10.161.179.55
79 . 0
79 . 1
Subnet Mask: 255 .255 .254 . 0
MAC Address: 00 : 43 : 6a : 39 : e7 : 16

Transfer Mode
● Network (recommended)
○ Serial

☑ Use XML's parent path as default

eMM(idevineL1_0118\out\target\product\Hi3798CV200\Emmc\Hi3798CV200-emmc.xml
Progr idevineL1_0118\out\target\product\Hi3798CV200\Emmc\fastboot.bin

✓	Partition Name	Select File	Device Type	File System	Start	Length
☑	fastboot	e_code\widevineL1_0118\out\target\product\Hi3798CV200\Emmc\fastboot.bin	emmc	none	0	1M
☑	bootargs	_code\widevineL1_0118\out\target\product\Hi3798CV200\Emmc\bootargs.bin	emmc	none	1M	1M
☑	recovery	_code\widevineL1_0118\out\target\product\Hi3798CV200\Emmc\recovery.img	emmc	none	2M	10M
☐	deviceinfo		emmc	none	12M	2M
☑	baseparam	ode\widevineL1_0118\out\target\product\Hi3798CV200\Emmc\baseparam.img	emmc	none	14M	8M
☑	pqparam	code\widevineL1_0118\out\target\product\Hi3798CV200\Emmc\pq_param.bin	emmc	none	22M	8M
☑	logo	urce_code\widevineL1_0118\out\target\product\Hi3798CV200\Emmc\logo.img	emmc	none	30M	20M
☐	logobak		emmc	none	50M	20M
☐	fastplay		emmc	none	70M	40M
☐	fastplaybak		emmc	none	110M	40M
☑	kernel	rce_code\widevineL1_0118\out\target\product\Hi3798CV200\Emmc\kernel.img	emmc	none	150M	40M
☐	misc		emmc	none	190M	20M
☑	qbboot		emmc	none	210M	1M
☑	qbdata		emmc	none	211M	300M
☑	trustedcore	ode\widevineL1_0118\out\target\product\Hi3798CV200\Emmc\trustedcore.img	emmc	none	511M	40M

After the burning is complete and the board is started, run the sample to burn the ID_WORD and lock it to the secure chip.

5.4 System Service Protection

Hi3798C V200 does not support system service protection currently. The solution before Hi3798C V200 uses the background thread of the kernel to verify the system file, whereas the solution after Hi3798C V200 will verify the system file in the init process. The reasons are as follows:

- The init process is a special process which does not receive or process signals and will not be killed.

- The executable file of the init process is in the kernel image. The procedure of verifying the kernel image in fastboot ensures that the init file is not tampered with.
- The code is placed in user mode, which facilitates maintaining and adding functions.