海思智能机顶盒2级安全方案
使用指南

文档版本　　00B04

发布日期　　2016-03-14

商标声明

 、HISILICON、海思和其他海思商标均为深圳市海思半导体有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

深圳市海思半导体有限公司

地址：　深圳市龙岗区坂田华为基地华为总部　邮编：518129

网址：　http://www.hisilicon.com

客户服务邮箱：　support@hisilicon.com

前 言

概述

本文档主要介绍海思安全启动和校验方案的使用方法。

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
DANGER	表示有高度潜在危险，如果不能避免，会导致人员死亡或严重伤害。
WARNING	表示有中度或低度潜在危险，如果不能避免，可能导致人员轻微或中等伤害。
CAUTION	表示有潜在风险，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
TIP	表示能帮助您解决某个问题或节省您的时间。
NOTE	表示是正文的附加信息，是对正文的强调和补充。

作者信息

章节号	章节名称	作者信息
全文	全文	W00269678/ G00180855

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2014-12-25	00B01	第一次临时发布。
2015-01-17	00B02	2.3 system 业务保护功能开关名称修改为 CFG_HI_ANDROID_SECURITY_L2_SYSTEM_CHECK； 2.4 recovery、kernel 分区签名校验方式改为快速签名校验方式，去掉 bootargs 中的环境变量 signature_check。
2015-05-29	00B03	2.4 增加 recovery、kernel 分区签名方式的开关配置 HISILICON_SECURITY_L2_COMMON_MODE_SIGN。
2016-03-14	00B04	增加 Hi3798CV200 二级安全方案的说明。

目 录

插图目录

1 概述

海思安全启动和校验方案，基于海思安全芯片提供的硬件保护机制，能够在不对 Android 系统和单板做过多的限制，不影响其必要的调试手段的情况下，对 Flash 中的重要数据如：boot、bootargs、kernel、system、recovery 等提供更高级别的保护，防止这些分区数据被非法篡改和替换，达到防止黑客通过专业工具或其它手段刷机的目的。

海思安全启动和校验方案是一个链式校验的机制，如图 1-1 所示。

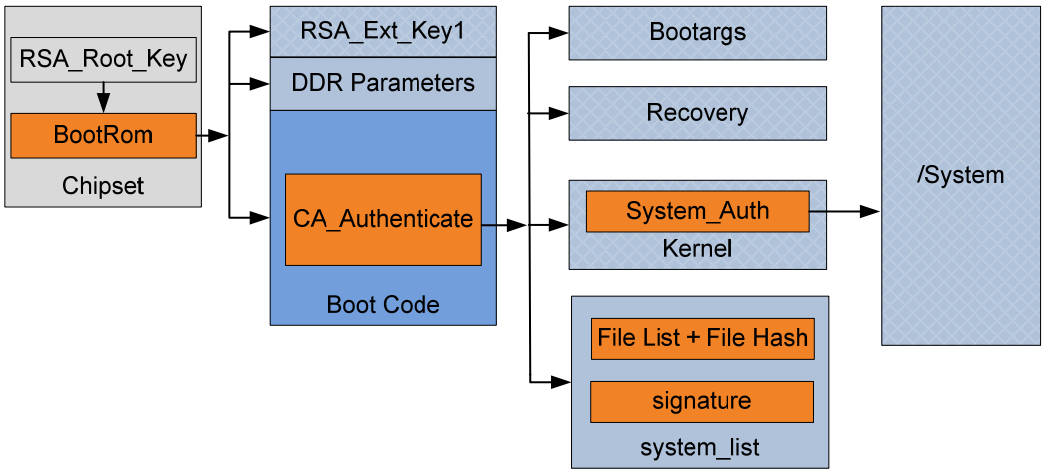
步骤 1 芯片校验 boot 镜像；

步骤 2 boot 校验 bootargs、recovery、kernel；

步骤 3 kernel 后台校验 system 分区数据。

----结束

图1-1 海思安全启动和校验方案流程图



本文档主要介绍了海思安全方案编译配置、量产烧写、HiTool 烧写的具体方法。

- 编译配置：主要介绍海思安全方案密钥（Key）生成和编译方法。
- 量产烧写：主要包括 USB 烧写和烧片器烧写。

- HiTool 烧写：主要用于研发人员开发调试。
- Hi3798CV200 二级安全方案：主要介绍针对 Hi3798CV200 芯片安全方案所做的改动。

该二级安全方案目前适用于海思 Hi3796MV100/Hi3798MV100/Hi3798CV200 芯片，Flash 类型支持：

- eMMC Flash
- SD Flash

说明

本文档前 4 章节适用于海思 Hi3796MV100 和 Hi3798MV100 芯片，第 5 章节针对 Hi3798CV200 芯片的安全方案的改动做了详细介绍。

2 编译配置

2.1 密钥生成

二级安全方案共需要用到如下 6 对密钥：

- 其中 4 对是 Android 系统密钥，用于对 APK、升级包等进行签名校验
- 另外 2 对是安全启动校验密钥，用于对各分区镜像进行签名校验

注意

安全方案的密钥不能丢失，不能泄露给他人，否则有可能引起系统被篡改或替换等安全问题，所以请妥善保管。

2.1.1 Android 系统密钥

Android 原生提供了 4 对密钥，用于 APK、升级包等的签名校验。海思安全方案使用 Android 原生机制进行 APK 和升级包的安全管理。编译海思安全方案前，需首先生成 Android 原生的 4 对密钥。

注意

- 编译用户在生成 Android 系统密钥时，必须为 root 用户。
- 安全方案量产时，必须要用 user 模式。
- 生成密钥及开发调试阶段，可以编译为 eng 模式（user 模式下串口无法使用）。
- 本文以 Hi3798MV100 芯片为例。

生成 Android 原生的 4 对密钥的步骤如下：

步骤 1 切换为 root 用户，进入代码的根目录

```
$ sudo -i
```

步骤 2 配置环境变量

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

步骤 3 执行脚本 mkkey.sh

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

该脚本会生成 4 对密钥，同时把密钥拷贝到/device/hisilicon/bigfish/upgrade/ota/security 下，以备制作 OTA 升级包用。执行过程中，会出现 4 次提示输入密码的信息，每次都请直接输入回车。

图2-1 4 次输入密码

```
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/bigfish/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/bigfish/upgrade/ota/security/platform.pk8'
`/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/platform.x509.pem' -> `/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/bigfish/upgrade/ota/security/platform.x509.pem'
`/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/shared.pk8' -> `/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/bigfish/upgrade/ota/security/shared.pk8'
`/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/Hi3798MV100/security/shared.x509.pem' -> `/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV/device/hisilicon/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/hisilicon/bigfish/upgrade/ota/security/testkey.x509.pem'
root@Moon:/home/cmo/HiSTBAndroidV600R001C00SPC056_IPTV#
```

非 root 权限用户会出现如下错误信息，不会生成密钥：

```
Warning: No input parameter, so the default authentication info will be used!
Enter password for '/home/XXX/worksapce/v5/iptv/device/hisilicon/Hi3798MV100/security/releasekey' (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)
```

步骤 4 退出 root 用户。

```
$ exit
```

步骤 5 生成密钥。

生成的 4 对密钥（releasekey、platform、shared、media）位于：device/hisilicon/Hi3798MV100/security/目录下。如图 2-2 所示。

图2-2 生成的 4 对密钥

```
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#
```

----结束

mkkey.sh 脚本生成的公私钥为 releasekey、platform、shared、media 4 对 key。这 4 对 key 的作用为：

- platform – 用于 core platform 中部分 apk 的签名，海思机顶盒的系统 apk 就是用 platform key 签名的；
- shared – 用于 home/contacts 进程中共享部分 apk 的签名；
- media – 用于 media/download 系统中部分 apk 的签名；
- releasekey – 用于系统中其他 apk 的签名、升级包（update.zip）的签名和校验。

此处如果用户不替换 platform, shared, media 这 3 对 Key 也是可以的，只需再拷贝 Android 原生的这 3 对 Key 到 device/hisilicon/Hi3798MV100/security/目录下即可。

在根目录下执行如下命令：

```
$ 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/
```

注意

- 通常密钥只需要生成一次。所以在未确定最终密钥前，如需更改，请删除 device/hisilicon/Hi3798MV100/security/目录下对应的文件后，再重新生成密钥，此删除文件操作不可缺省。
- 如果已经有密钥在 device/hisilicon/Hi3798MV100/security/目录下，也需要执行下 mkkey.sh。mkkey.sh 不会再生成新的密钥，而只会把生成的 4 对密钥拷贝到 /device/hisilicon/bigfish/upgrade/ota/security 下，以备制作 OTA 升级包用。
- 如果替换全部的 Key，后续对于用原生 platform, shared, media 签名的具有特殊权限的 APK 就要重新用新的 platform, shared, media 签名才能安装，更安全但是操作会更复杂；如果只替换 releasekey，后续对于用原生 platform, shared, media 签名的具有特殊权限的 APK 就可以直接安装，操作简单。用户可根据自己的使用场景自行选择。

2.1.2 安全启动校验密钥

安全启动涉及到的镜像都需要使用密钥签名。

- 一共有两对密钥：
  - 一对用于对 fastboot 签名校验；
  - 一对用于对除 fastboot 以外的其它镜像签名校验。

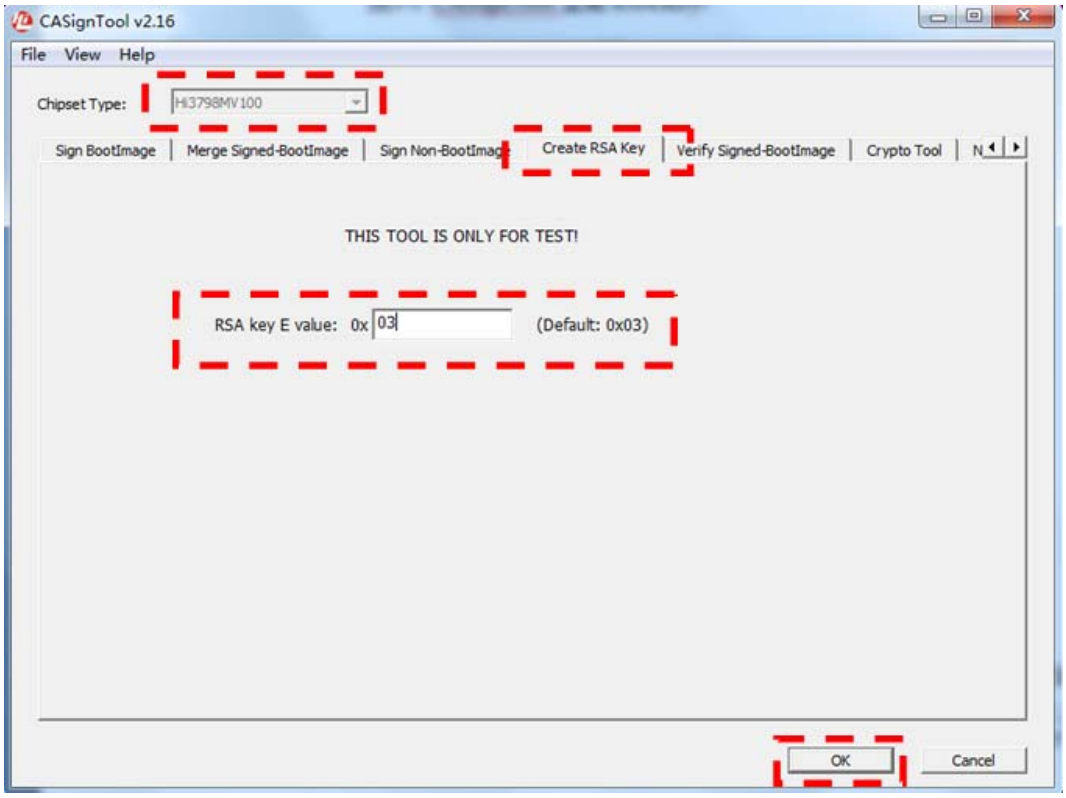
需要使用海思提供的工具生成，工具所在路径：

```
device/hisilicon/bigfish/sdk/tools/windows/advca/CASignTool
```

使用海思提供的工具生成两对密钥的步骤如下：

步骤 1 选择芯片类型：“Hi3798MV100”，切换至 Create RSA Key 页面，输入 RSA key E value 默认值：03，如图 2-3 所示。

图2-3 CASignTool 生成 RSA Key

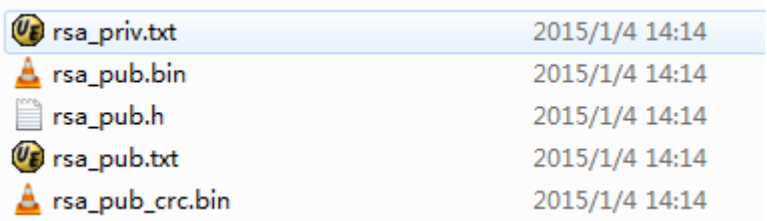


注意

输入的 RSA key E value 默认值为 0x03，会用来生成 RSA Key 对。也可以输入其他 16 进制正整数，最大为 0xffffffff。不过改为其他值会大大增加生成密钥的计算量，推荐使用默认值 0x03。

步骤 2 点击 OK 键，工具会在其所在目录生成一个“RSA_XXXXXXX”的目录用于存放生成的 Key 文件，如图 2-4 所示。

图2-4 工具生成的 key 文件



生成第一对密钥，用于对 fastboot 的签名校验。

- 将 rsa_pub.txt 重命名为 root_rsa_pub.txt（txt 格式存放对 fastboot 校验的密钥，暂时用不到）
- 将 rsa_priv.txt 重命名为 root_rsa_priv.txt（txt 格式存放对 fastboot 签名的密钥，需要拷贝到指定文件夹）
- 将 rsa_pub.bin 重命名为 root_rsa_pub.bin（二进制格式存放对 fastboot 校验的密钥，命令行方式烧 Key 时使用，参考“3.4 安全芯片状态判断方法”）
- 将 rsa_pub_crc.bin 重命名为 root_rsa_pub_crc.bin（对 root_rsa_pub.bin 做了 CRC 签名后的二进制文件，USB 烧写时使用，参考 3.1.1 和 3.1.2 ）

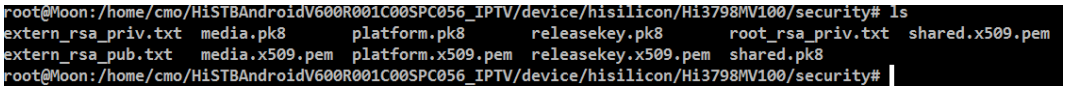
步骤 3 再次点击 OK，生成第二对密钥，用于 bootargs、recovery、kernel 等的签名校验。

- 将 rsa_pub.txt 重命名为 extern_rsa_pub.txt（txt 格式存放对镜像校验的密钥，需要拷贝到指定文件夹）
- 将 rsa_priv.txt 重命名为 extern_rsa_priv.txt（txt 格式存放对镜像签名的密钥，需要拷贝到指定文件夹）

步骤 4 把 extern_rsa_pub.txt、extern_rsa_priv.txt、root_rsa_priv.txt 拷贝至如下目录：

```
device/hisilicon/Hi3798MV00/security/
```

图2-5 全部密钥生成完成后的 device/hisilicon/Hi3798MV00/security/目录



----结束

2.2 硬件配置文件

硬件参数配置文件与硬件单板相关，请根据自己的单板类型进行修改。

以 Hi3798mdmo1b 单板，Hi3798MV100 芯片为例，对应的硬件配置表格在 device/hisilicon/bigfish/sdk/source/boot/sysreg/目录，名为：hi3798mdmo1b_hi3798mv100_ddr3_1gbyte_16bitx2_2layers.xlsm。

以 eMMC flash 为例，对应的硬件参数配置文件为 hi3798mdmo1b_hi3798mv100_ddr3_1gbyte_16bitx2_2layers_emmc.cfg。

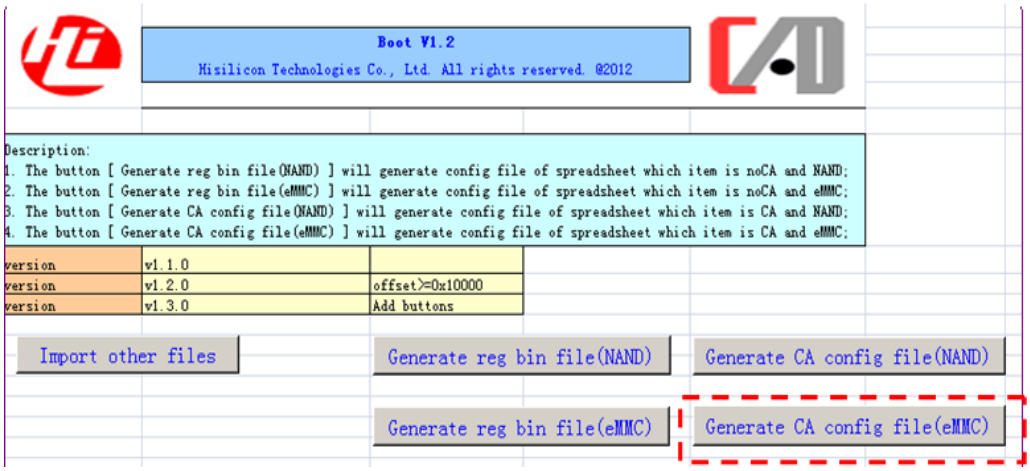
默认的参数配置文件只能支持 eMMC flash 的单板，如果要支持 SD flash 类型的单板，需要先按照图 2-6 红色标记的地方修改表格（0xf01 改成 0xf00），然后再点击硬件配置表格中的“Generate CA config file(eMMC)”按钮在当前目录生成参数配置文件，如图 2-7 所示。

图2-6 SD flash 的硬件参数配置表格

	A	B	C	D	E	F	G	H	I
1	Module Name	pin_mux_drv_emmc	pin mux & drv config						
2	Base Address	0xf8a21000	Add module				Add register		
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	
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	EMMC
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	
25	ioshare_22	0x58	0x5	0	write	7	0	0x0000003F	
26	ioshare_23	0x5c	0x5	0	write	7	0	0x0000003F	JTAG

ddr_wakeup | qos_ctrl | crg_ctrl | clk_ssmod | peri_cfg | pin_mux_drv_emmc | pin_mux_drv_nand | eth_phy_cfg | others

图2-7 参数配置文件生成工具



device/hisilicon/Hi3798MV100/BoardConfig.mk 文件中，以下配置要与实际所用到文件名一一对应：

- 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 业务保护

2.3.1 功能概述

System 业务保护功能是对原始 system 分区下的文件进行保护，防止被篡改。如果发现原始文件被修改，单板会重启。

2.3.2 功能开关配置

System 业务保护功能默认关闭，如果需要打开，编译前请按照下面的方法修改 device/hisilicon/bigfish/sdk/configs/hi3798mv100 下的对应的 hi3798mdmo_hi3798mv100_android_security_cfg.mak 文件中的配置项。

以 Hi3798MV100 1b 的单板为例，需要修改 hi3798mdmo_hi3798mv100_android_security_cfg.mak 中的 CFG_HI_ANDROID_SECURITY_L2_SYSTEM_CHECK 配置项为：

CFG_HI_ANDROID_SECURITY_L2_SYSTEM_CHECK=y，然后再执行编译。

2.4 编译

请按照以下步骤编译：

步骤 1 在 device/hisilicon/Hi3798MV100/customer.mk 中，将 HISILICON_SECURITY_L2 的值配置为 true：

```
HISILICON_SECURITY_L2 := true
```

步骤 2 在 device/hisilicon/Hi3798MV100/customer.mk 中，根据需要配置 HISILICON_SECURITY_L2_COMMON_MODE_SIGN：

HISILICON_SECURITY_L2_COMMON_MODE_SIGN 默认为 false，作用是在编译时对 recovery 和 kernel 用 special 方式进行签名；如果配置为 true，则会对 recovery 和 kernel 用 common 方式进行签名。后续版本推荐配置为 false：

```
HISILICON_SECURITY_L2_COMMON_MODE_SIGN := false
```

步骤 3 进入代码的根目录，在命令行下执行：

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

步骤 4 然后执行：

```
$ lunch Hi3798MV100-user
```

步骤 5 最后执行：

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

编译完成后，将在 out/target/product/Hi3798MV100/Security_L2/目录下生成两种用途的镜像，其中：

- out/target/product/Hi3798MV100/Security_L2/PRODUCTION/：生产目录。
- out/target/product/Hi3798MV100/Security_L2/MAINTAIN/：维修目录。

两种镜像区别为：MAINTAIN 下的 fastboot.bin、bootargs.bin、recovery.img、kernel.img 是分别对 PRODUCTION 下的这几个文件签过名的，其他文件都相同。

- 生产目录和维修目录的镜像使用，请参考“3 量产烧写”。
- 建议用户不要随意修改镜像名字。如果用户通过修改编译脚本使得 PRODUCTION 下的 fastboot.bin、bootargs.bin、recovery.img、 kernel.img 的名字改变了，编译前需要做如下改动。例如：
  - 如果 PRODUCTION 下生成的 fastboot.bin 被改成了 fastboot_product.bin，需要同时修改 device/hisilicon/bigfish/security/secureSignTool/etc 下的 Signboot_config.cfg 文件如下：

    BootFile=fastboot.bin ➔BootFile=fastboot_product.bin
  - 如果 PRODUCTION 下生成的 bootargs.bin 被改成了 bootargs_product.bin，需要同时修改 device/hisilicon/bigfish/security/secureSignTool/etc 下的 common_bootargs_config.cfg 文件如下：

    InputFile=bootargs.bin➔InputFile=bootargs_product.bin
  - 如果 PRODUCTION 下生成的 recovery.bin 被改成了 recovery_product.img，需要同时修改 device/hisilicon/bigfish/security/secureSignTool/etc 下的 special_recovery_config.cfg 文件如下：

    Image1=recovery.img➔Image1= recovery_product.img
  - 如果 PRODUCTION 下生成的 kernel.bin 被改成了 kernel_product.img，需要同时修改 device/hisilicon/bigfish/security/secureSignTool/etc 下的 special_kernel_config.cfg 文件如下：

    Image1= kernel.img➔Image1= kernel_product.img

----结束

3 量产烧写

Hi3798M 出厂时不再区分安全芯片与非安全芯片。

- 用户使用海思二级安全方案，编译出的安全版本，会把芯片锁定为安全芯片。
- 不使用海思二级安全方案，编译出的非安全版本，会把芯片锁定为非安全芯片。

注意

一旦芯片被锁定为非安全芯片，就不能再更改为安全芯片，也无法烧写安全芯片的版本，反之亦然。所以请务必小心烧写。关于如何区分安全芯片或者非安全芯片，请参考“3.4 安全芯片状态判断方法”。

3.1 USB 烧写

海思芯片支持 USB 烧写，可在量产中使用，操作简单。

注意

- **产线强烈推荐 U 盘量产方式。**
- **USB 烧写基本要求是在 USB2.0 端口下，U 盘格式为 fat32，详细的约束条件请参考文档《量产烧写 使用指南》。**

3.1.1 USB 裸片烧写

烧写前需要准备如下：

- PRODUCTION 目录下的 fastboot.bin、bootargs.bin、recovery.img、update.zip
- CAsigntool 工具生成的 key：root_rsa_pub_crc.bin
- FAT32 格式的 U 盘

USB 裸片烧写的步骤如下：

步骤 1 将 fastboot.bin、bootargs.bin、recovery.img、update.zip、root_rsa_pub_crc.bin 拷贝至 U 盘根目录。

步骤 2 将 U 盘插入 USB2.0 接口。

步骤 3 将单板上电，将自动进入烧写流程。烧写过程中指示灯会不断闪烁，烧写完成后，指示灯将常亮。

----结束

3.1.2 USB 非裸片烧写

经过 USB 裸片烧写，启动完成的单板，已经锁定为安全芯片。

安全芯片再烧写，需使用 USB 非裸片烧写方式。

需要准备：

- MAINTAIN 目录下的 fastboot.bin、bootargs.bin、recovery.img、update.zip
- CAsigntool 工具生成的 key：root_rsa_pub_crc.bin
- FAT32 格式的 U 盘

步骤 1 将 fastboot.bin、bootargs.bin、recovery.img、 update.zip、root_rsa_pub_crc.bin 拷贝至 U 盘根目录。

步骤 2 将 U 盘插入 USB2.0 接口。

步骤 3 按住单板上的 USB 烧写按键上电，进入烧写流程。烧写过程中指示灯会不断闪烁，烧写完成后，指示灯将常亮。

----结束

注意

- **如在 USB 裸片或非裸片烧写时，不将 root_rsa_pub_crc.bin 放入到 U 盘，则需要厂测程序烧写 OTP key 和安全标志位。具体方法请参考“3.3 命令行烧写安全启动标志和 OTP key”。**
- **U 盘多次烧写时，需要强制按住 USB 烧写按键上电，否则不会走 U 盘烧写流程。也可以通过擦除 fastboot 的方法（在 fastboot 下，执行 `mw.b 1000000 ff 800` 和 `mmc write 0 1000000 0 100` 两条命令），此时芯片为裸片操作模式，U 盘烧写会走 USB 裸片烧写的流程。**

3.2 烧片器烧写

使用烧片器烧写，需要做以下准备：

- 烧片器烧写镜像：
  - PRODUCTION 目录内未签名的 fastboot.bin
  - MAINTAIN 目录除 fastboot.bin 以外的其他的系统镜像
- 厂测程序：

  厂测程序由客户根据产线流程开发，需要完成以下工作：
  - 烧写 MAINTAIN 目录下签名后的 fastboot.bin
  - 烧写 OTP key，即使用 CAsigntool 工具生成的 root_rsa_pub.bin
  - 烧写安全启动标志

注意

- 目前暂不支持 flash 器件类型为 NAND 器件。
- 厂测程序由客户完成，烧写 OTP Key 及安全标志位请参考“3.3 命令行烧写安全启动标志和 OTP key”，烧写签名后的 fastboot.bin，请使用海思提供的 flash 读写接口完成。

3.3 命令行烧写安全启动标志和 OTP key

对于烧写安全启动标志和 OTP key，海思提供了 Sample：

device/hisilicon/bigfish/sdk/sample/advca/sample_ca_writeRSAkey.c 和 device/hisilicon/bigfish/sdk/sample/advca/sample_ca_opensecboot.c

烧写 OTP key 的步骤如下：

步骤 1 准备到 2.1.2 步骤 2 中，使用 CAsigntool 工具生成的 root_rsa_pub.bin。

步骤 2 将 root_rsa_pub.bin 用 adb 工具推送到单板的 sdcard 目录下。

步骤 3 在单板端串口命令行执行：

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

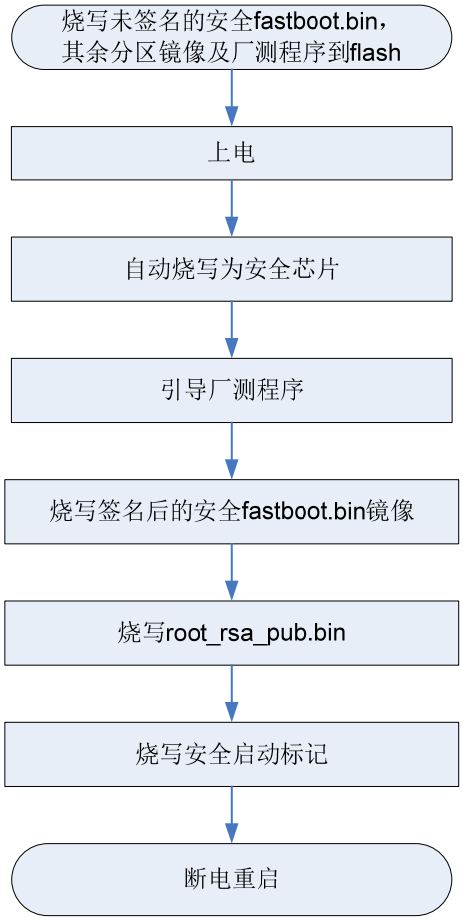
----结束

烧写安全启动标志：

- 若 flash 器件类型为 eMMC 器件，则需要在单板端串口命令行执行：
  ```
  sample_ca_opensecboot  emmc
  ```
- 若 flash 器件类型为 SD 器件，则需要在单板端串口命令行执行：
  ```
  sample_ca_opensecboot  sd
  ```

裸片烧写流程如图 3-1 所示。

图3-1 裸片烧写流程



注意

如第一次启动厂测程序没有执行成功，此时系统将无法启动，请参考 3.1.2 中，USB 非裸片烧写。

3.4 安全芯片状态判断方法

3.4.1 通过读 otp 寄存器

通过读 otp 寄存器判断安全芯片状态的方法如下：

步骤 1 在 boot 下通过 otpreadall 命令或者系统起来后查看 cat /proc/msp/otp 的方法获得寄存器值：

```
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
```

步骤 2 判断芯片是否是安全芯片。

看 00a8~00ab，00a0 第三列的四个字节，即为芯片的 ID_WORD（区分安全芯片和非安全芯片的 ID）。

- 如果 ID_WORD 为 0x2a13c812，那芯片就是非安全芯片
- 如果 ID_WORD 为 0x6edbe953，那芯片就是安全芯片

步骤 3 确认芯片 ID_WORD 是否已被锁定。

通过 0010 第 10bit 位（OTP 中 0x11 字节的 bit[2]）来确认 ID_WORD 是否是锁住了，bit[10]如果为 0，ID_WORD 就没有锁住，如果为 1 就锁住了。

例如对于 00480400，第 10 位为 1，说明已经锁住了，无法再改变。

0x00480400 转化为二进制如图 3-2 所示。

图3-2 二进制转化图

注意

芯片经过烧写流程，首次启动后，芯片的 ID_WORD 就锁定完成，不能再改变。

步骤 4 判断芯片 OTP Key（root_rsa_pub.bin）是否被锁定。

方法同上，OTP 中 0x14 字节的 bit[0]标志的是 OTP Key 是否被锁定。

0 表示没锁定，1 表示已锁定；

OTP Key 只能被烧写一次，所以烧写前请确保 Key 的正确性。

步骤 5 判断芯片安全启动标志是否使能。

方法同上，OTP 中 0x18 字节的 bit[0]标志的是安全启动标志。

0 表示未使能，1 表示已使能；

使能安全启动标志前请确保 OTP Key 已被正确烧写，两者顺序不能颠倒，否则会导致系统起不来。

----结束

3.4.2 通过开机打印日志区分

安全和非安全芯片的串口打印：

- 安全芯片启动串口日志打印如下（注意红色字体部分）：

```
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

Fastboot:      Version 3.3.0
Build Date:    Dec 16 2014, 15:20:58
CPU:           Hi3798Mv100 (CA)   /*安全芯片*/
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
/*表示芯片ID_WORD已锁定*/

SDK Version: HiSTBLinuxV100R002

Security Begin Read RSA Key!
Secure boot is enabled  /*OTP Key已烧写并且安全启动已使能*/
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
```

- 非安全芯片启动串口日志打印如下（注意红色字体部分）：

```
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   /*非安全芯片*/
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
/*芯片ID_WORD已锁定*/

SDK Version: HiSTBLinuxV100R002
/*没有安全启动相关打印*/
press the key!!
get key 0 2
get chipid =137980100
get chipType (HI3798MV100)
count=2
mac:3E:77:2F:81:C1:DA
```

4 HiTool 烧写

HiTool 烧写主要用于研发人员开发调试。

HiTool 烧写分为裸片烧写和非裸片烧写两种情况。

通过 HiTool 烧写完镜像后，烧写安全启动标志和 OTP Key 请参考“3.3 命令行烧写安全启动标志和 OTP key”。

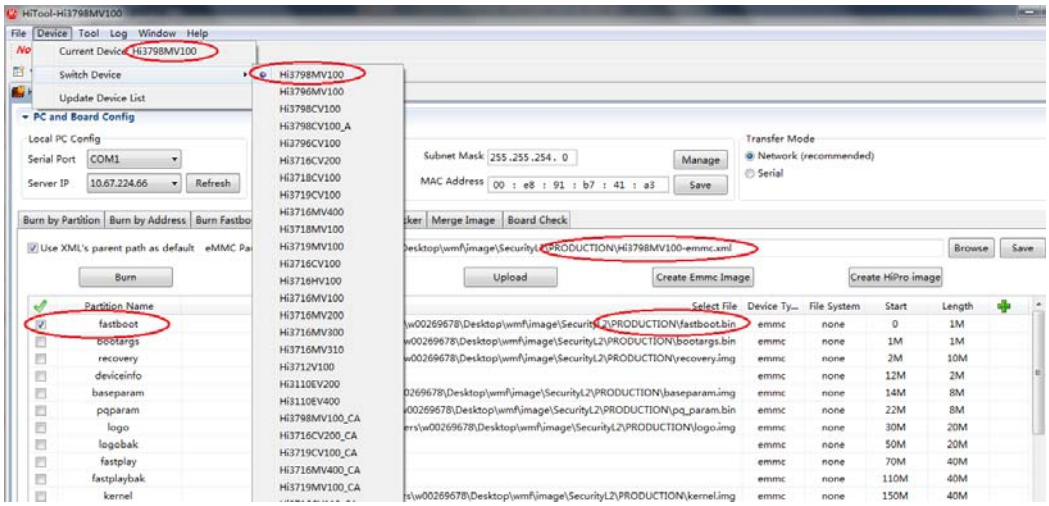
4.1 HiTool 裸片烧写

HiTool 裸片烧写步骤如下：

步骤 1 烧写 PRODUCTION 下未签名 fastboot.bin。

- 选择芯片类型：Hi3798MV100
- 只烧写 PRODUCTION 下的 fastboot.bin
- 启动烧写，烧写完毕后上电，单板将被烧写成安全芯片

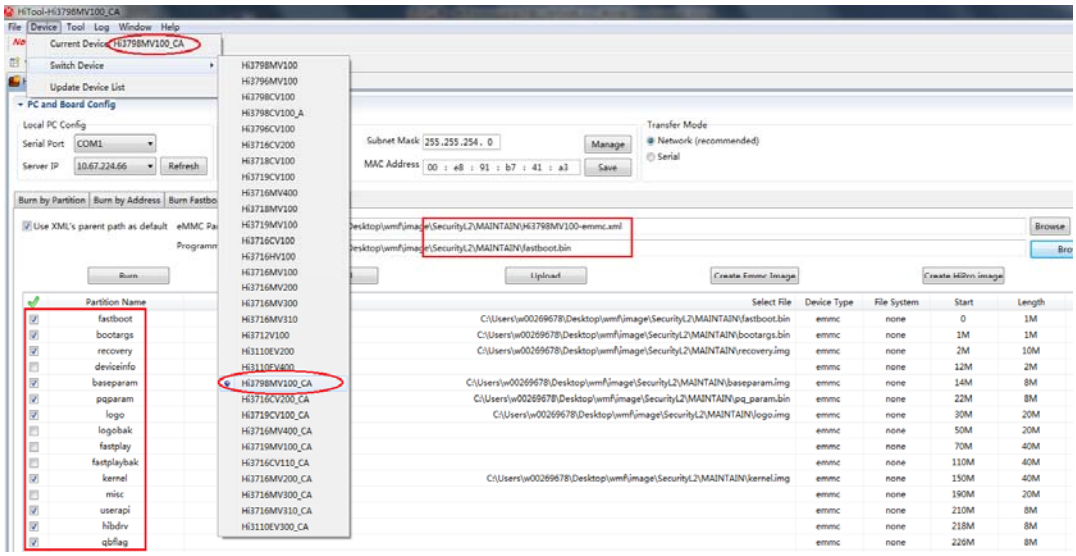
图4-1 步骤 1 烧写工具配置



步骤 2 烧写其他镜像。

- 选择芯片类型：Hi3798MV100_CA
- Programmer 文件选择：MAINTAIN 目录下的签名的 fastboot.bin
- 分区表和镜像使用 MAINTAIN 下的完整镜像
- 选择全部分区及配套的镜像
- 启动烧写

图4-2 烧写工具配置



4.2 HiTool 非裸片烧写

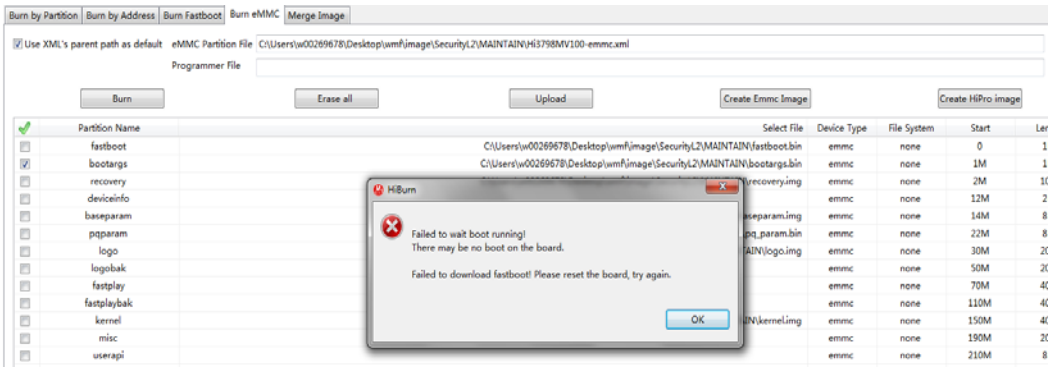
HiTool 非裸片烧写与 4.1 步骤 2 方法相同：

- 选择芯片类型：Hi3798MV100_CA
- Programmer 文件选择：MAINTAIN 目录下的签名的 fastboot.bin
- 分区表和镜像使用 MAINTAIN 下的完整镜像
- 选择全部分区及配套的镜像
- 启动烧写

注意

HiTool 非裸片烧写时，如需要单独烧写除 fastboot 分区以外的其他分区，请在 HiTool 的 Programmer 文件那一栏选择文件夹 MAINTAIN 下 fastboot.bin，否则会报错误“There may be no boot on the board.”如图 4-3 中，单独烧写 bootargs 时，Programmer 文件未选择，则 HiTool 工具报错。

图4-3 单独烧写 bootargs 分区

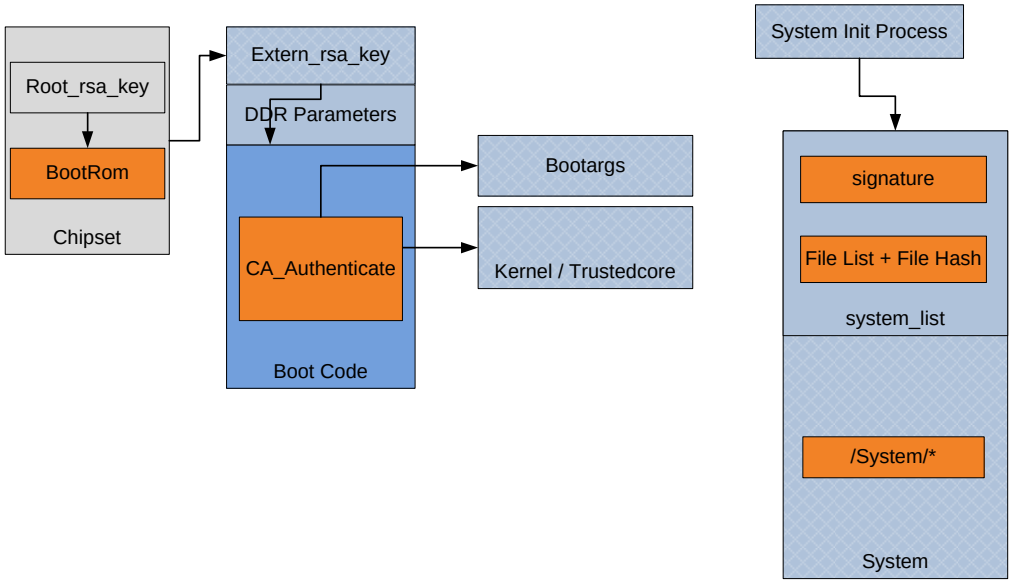


5 Hi3798CV200 二级安全方案

Hi3798CV200 芯片二级安全方案涉及到的改动包括校验流程，编译，量产烧写，HiTool 烧写，system 业务保护。

5.1 方案改动

图5-1 Hi3798CV200 安全启动和校验方案流程图



和之前的安全方案相比，主要有以下改动：

- 正常启动时不校验 recovery 分区，只有触发升级后需要引导进入 recovery 时才会对 recovery 分区进行校验。
- 对 system_list 和 system 文件的校验放到 system init 进程中。
- ID_WORD 的烧写不在 boot 中烧写，放到工厂生产阶段的厂测程序中进行。

5.2 编译

5.2.1 Android 系统密钥

对于 Android 系统密钥，修改了 mkkey.sh，只替换 releasekey，其他 3 对 key（platform, shared, media）直接从 Android 原生的 key 拷贝过来。对于用 Android 原生 platform, shared, media 签名的具有特殊权限的 APK 可以直接安装。

5.2.2 硬件配置文件

Hi3798CV200 芯片对 fastboot 的签名不需要硬件配置文件。

5.2.3 编译步骤

对安全 fastboot 做了修改，只需要编译出一套签名后的镜像即可，编译步骤如下：

步骤 1 在 device/hisilicon/Hi3798CV200/customer.mk 中，将 HISILICON_SECURITY_L2 的值配置为 true：

```
HISILICON_SECURITY_L2 := true
```

步骤 2 进入代码的根目录，在命令行下执行：

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

步骤 3 然后执行：

```
$ lunch Hi3798CV200-user
```

步骤 4 最后执行：

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

编译完成后，将在 out/target/product/Emmc/目录下生成签过名的镜像。

其中 bootargs 是通过 common 方式签名，recovery、kernel 和 trustedcore 是通过 special 方式签名的。

----结束

注意

当单板类型变更时，需要修改 device/hisilicon/Hi3798CV200/BoardConfig.mk **里的** BOOT_REG_NAME **匹配单板和芯片类型，具体名字可以在** device/hisilicon/bigfish/sdk/source/boot/sysreg/**下找到。**

5.3 量产烧写

5.3.1 ID_WORD 烧写并锁定

ID_WORD 烧写并锁定的时机，原先是在 boot 下烧写，后续修改为在厂测程序中烧写的方案。海思提供如下 sample，客户可根据此 sample 在厂测程序中实现。

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

烧写 ID_WORD：

- 系统起来后，在单板端串口下执行如下命令烧写安全 ID_WORD 并锁定：
  `sample_otp_lockidword 1`
- 查看 ID_WORD 烧写状态，执行如下命令：
  `sample_otp_lockidword 2`

5.3.2 USB 烧写

烧写前需要准备如下：

- Emmc 目录下的 fastboot.bin、bootargs.bin、recovery.img、update.zip
- CAsigntool 工具生成的 key：root_rsa_pub_crc.bin
- FAT32 格式的 U 盘
- 厂测程序：
  厂测程序由客户根据产线流程开发，需要完成烧写 ID_WORD 并锁定为安全芯片。

USB 烧写的步骤如下：

步骤 1 将 fastboot.bin、bootargs.bin、recovery.img、update.zip、root_rsa_pub_crc.bin 拷贝至 U 盘根目录。

步骤 2 将 U 盘插入 USB2.0 接口。

步骤 3 如果是裸片，单板上电，将自动进入烧写流程；如果是非裸片，需要按住单板上的 USB 烧写按键，单板上电会进入烧写流程。烧写过程中指示灯会不断闪烁，烧写完成后，指示灯将常亮。

步骤 4 启动单板进入厂测程序烧写 ID_WORD 并锁定为安全芯片。

----结束

注意

USB 烧写时，客户也可以选择把烧写 otp key 和安全启动标志放在厂测程序中烧写，这样就不需要在 U 盘根目录放 root_rsa_pub_crc.bin 。

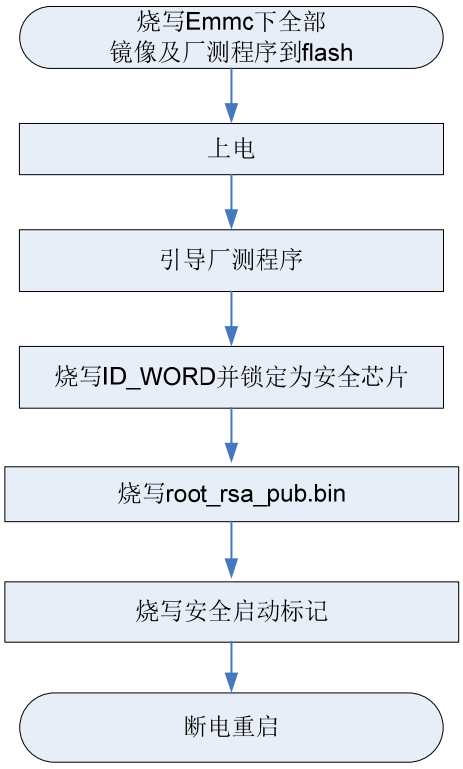
5.3.3 烧片器烧写

使用烧片器烧写，需要做以下准备：

- 烧片器烧写镜像：
  - Emmc 目录下的全部镜像
- 厂测程序：

  厂测程序由客户根据产线流程开发，需要完成以下工作：
  - 烧写 ID_WORD 并锁定为安全芯片
  - 烧写 OTP key，即使用 CAsigntool 工具生成的 root_rsa_pub.bin
  - 烧写安全启动标志

图5-2 Hi3798CV200 裸片烧写流程



5.3.4 HiTool 烧写

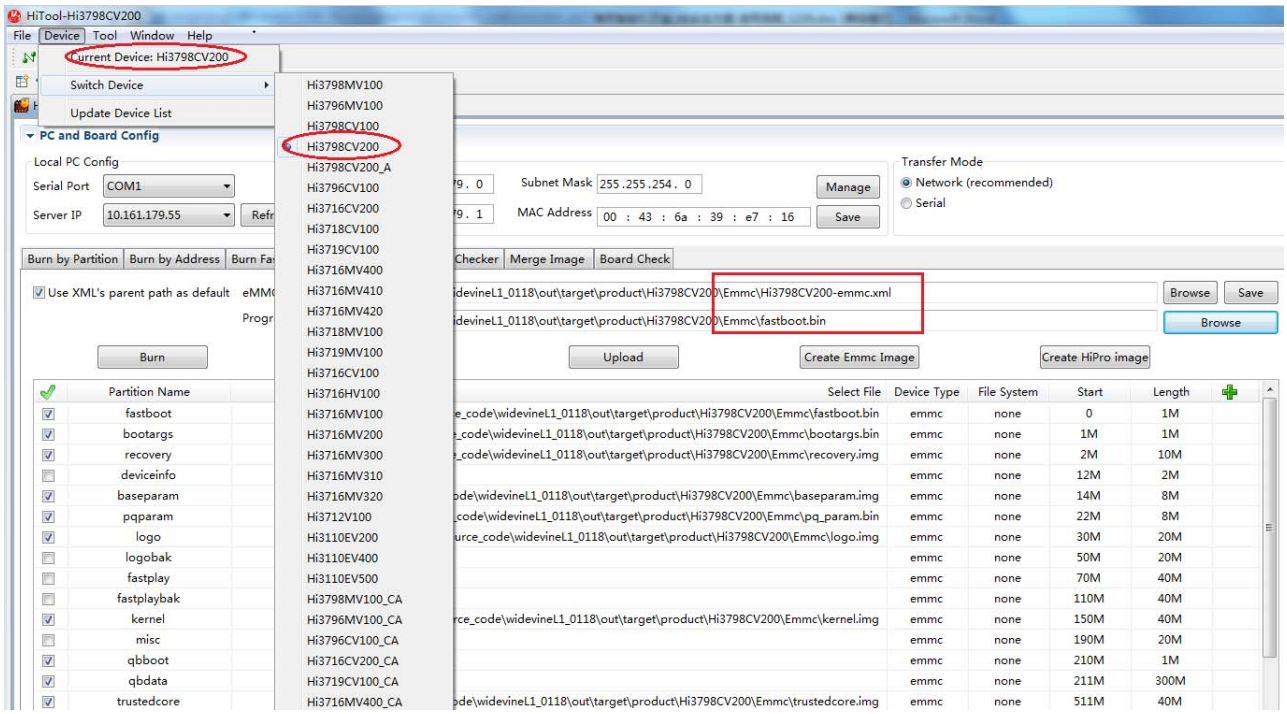
HiTool 烧写主要用于研发人员开发调试。

Hi3798CV200 芯片通过 HiTool 烧写不再区分裸片烧写和非裸片，配置如下：

- 选择芯片类型：Hi3798CV200
- Programmer 文件选择：Emmc 目录下的签名的 fastboot.bin
- 分区表和镜像使用 Emmc 下的完整镜像

- 选择全部分区及配套的镜像
- 启动烧写

图5-3 Hi3798CV200 烧写工具配置



烧写完之后待单板启动后，执行 sample 烧写 ID_WORD 并锁定为安全芯片。

5.4 System 业务保护

Hi3798CV200 芯片现在还不支持 system 业务保护功能，CV200 之前的方案是通过 kernel 的后台线程去校验 system 系统的文件，CV200 以后会改为在 init 进程中进行校验。原因如下：

- init 进程是特殊进程，它不接收也不处理信号，不会被杀；
- init 进程的可执行文件是放在 kernel 镜像中，在 fastboot 校验 kernel 步骤可以保证 init 文件不被篡改；
- 代码放到用户态，方便维护和新增功能；