Hi3516C V500/Hi3516D V300/Hi3516A V300 Secure Boot User Guide

Issue 01

Date 2019-09-15

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 (Shanghai) Technologies Co., Ltd.

Address: New R&D Center, 49 Wuhe Road, Bantian,
Longgang District,
Shenzhen 518129 P. R. China

Website: http://www.hisilicon.com/en/

Email: support@hisilicon.com

About This Document

Purpose

This document describes how to perform secure boot for Hi3516C V500, Hi3516A V300, and Hi3516D V300. It covers the following topics: secure boot, secure image generating, and OTP burning.

The following boot media are supported: SPI NOR flash, SPI NAND flash, and eMMC

NOTE
Unless otherwise specified, the description of Hi3516D V300 is the same as that of Hi3516C V500.

Related Versions

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

Product Name	Version
Hi3516C	V500
Hi3516D	V300
Hi3516A	V300

Intended Audience

This document is intended for:

- Technical support engineers
- Software development engineers

Change History

Changes between document issues are cumulative. The latest document issue contains all the changes made in earlier issues.

Issue 01 (2019-09-15)

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

Sections 1.4 and 2.1 are modified.

Issue 00B03 (2018-12-28)

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

Chapter 3 "OTP Burning" is modified.

Issue 00B02 (2018-11-20)

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

In section 1.1, figure 1-1 is modified.

In section 1.2, figure 1-2 is modified.

Issue 00B01 (2018-09-06)

This issue is the first draft release.

Contents

Figures

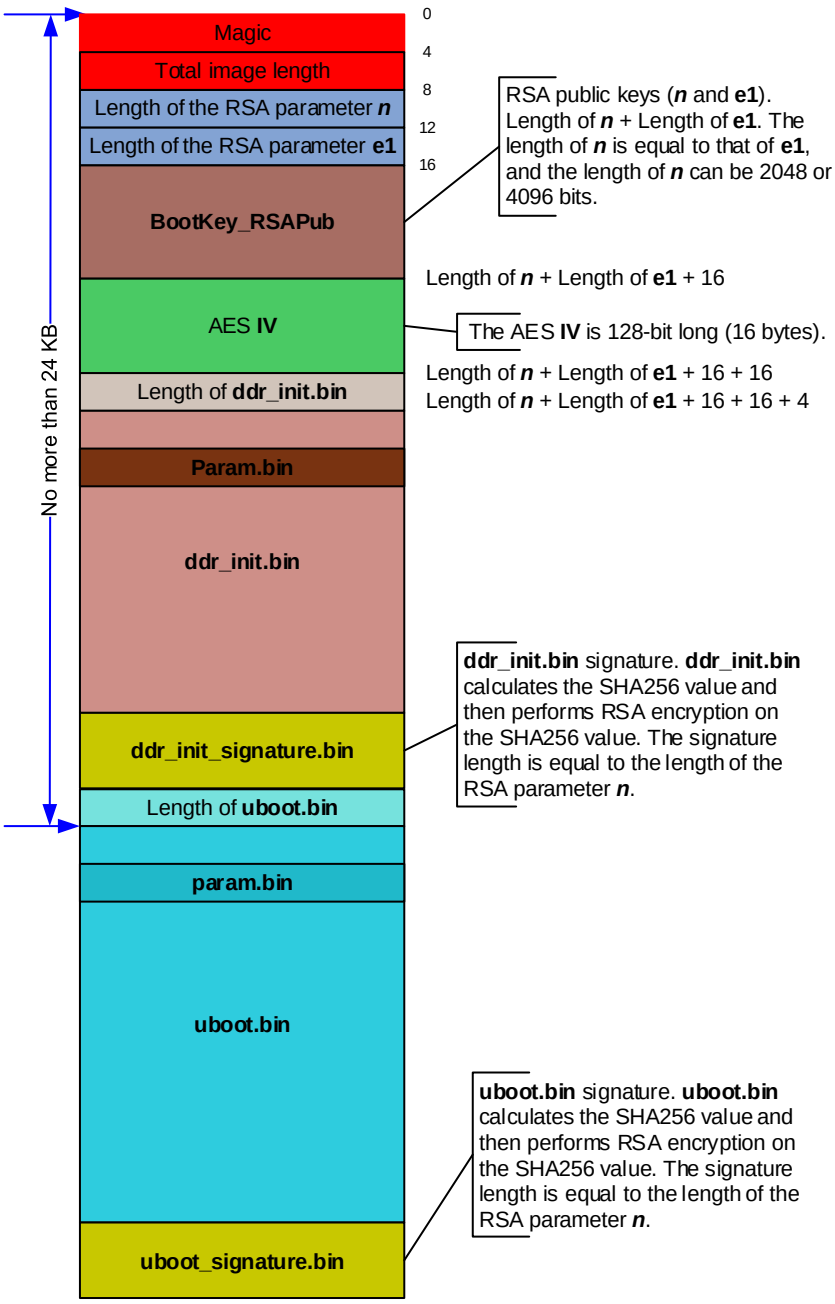
1 Secure Boot

Hi3516C V500 supports common secure boot and encrypted secure boot. The difference lies in that **u-boot.bin** and its signature file **uboot_signature.bin** are plaintexts in the common secure boot image but ciphertexts in the encrypted secure boot image.

1.1 Structure of the Common Secure Boot Image

Figure 1-1 shows the image structure of the common secure U-Boot of Hi3516C V500.

Figure 1-1 Structure diagram of the common secure boot image

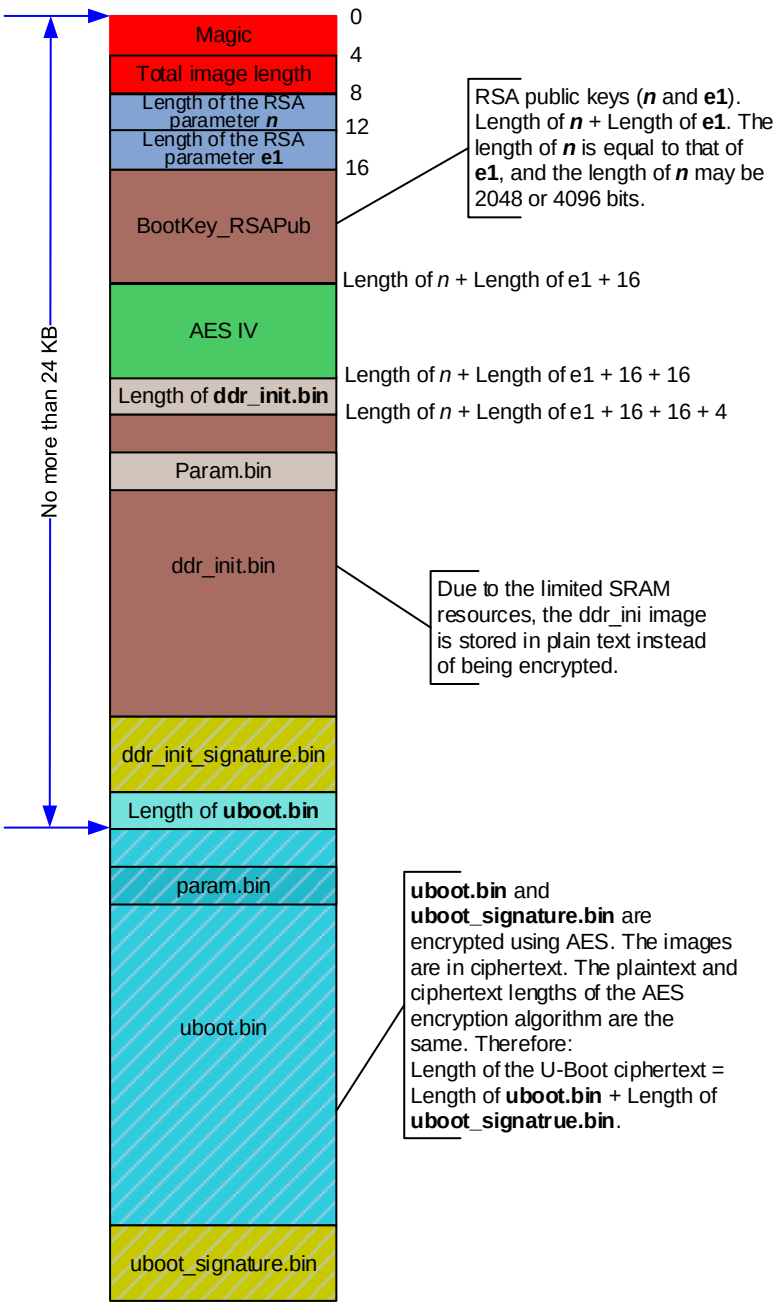


The U-Boot image for common secure boot consists of the public key image, **ddr_init.bin** image (including **param.bin** and **ddr_init.bin**), non-secure **uboot.bin** image, digital signature of **ddr_init.bin** (**ddr_init_signature.bin**), digital signature of the non-secure **uboot.bin** (**uboot_signature.bin**), as well as the length information of the preceding images.

RSA-2048 and RSA-4096 are supported. The value of the AES **IV** (that is, the initialization vector) is **0**.

1.2 Structure of the Encrypted Secure Boot Image

Figure 1-2 Structure diagram of the encrypted secure boot image

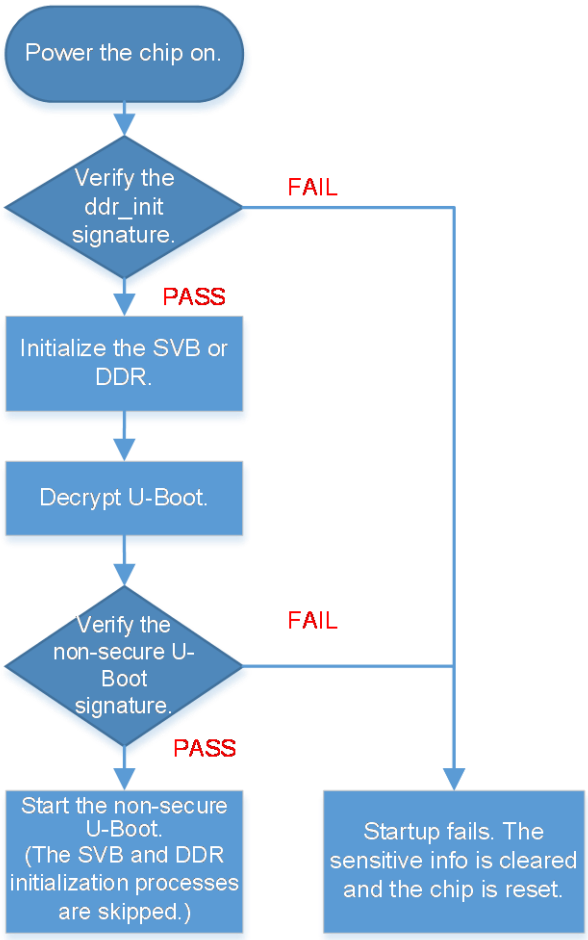


The U-Boot image for encrypted secure boot consists of the public key image, **ddr_init.bin** image (including **param.bin** and **ddr_init.bin**) and its digital signature, non-secure **uboot.bin** image and its ciphertext digital signature after AES encryption, as well as the length information of the preceding images.

RSA-2048 and RSA-4096 are supported. The value of AES **IV** is not **0**.

1.3 Secure Boot Process

Figure 1-3 Secure boot process



1.4 Directory of the Secure Boot Source Code

The directory of the secure boot source code is **secureboot_release**. The detailed directory structure is as follows:

```
├── CASignTool > CASignTool_Linux_BVT/CASignTool/bin/CASignTool_m64
├── CASignTool_Linux_BVT ---------- Directory of the CASignTool source code
│   ├── build.sh
│   ├── CASignTool
│   ├── libCASign
│   └── readme.txt
├── ddr_init ---------- Directory of the DDR initialization source code
```

```
│   ├── boot
│   ├── cfg.mk
│   ├── ddr_init_ reg_info.bin
│   ├── drv
│   ├── include
│   ├── linker.lds
│   ├── linker.lds.mk
│   ├── Makefile
│   └── mkddrinit.sh
│   ├── ddr_init_reg_info.bin ---------- Generated DDR initialization image
├── HASH ---------- Hash value parser for the RSA public key, generated by the hash_modify.c file
├── hash_modify.c
├── AES ---------- AES KEY value parsing tool, generated by the aeskey2reg.c file
├── aeskey2reg.c
├── Makefile ---------- General Makefile of the secure boot SDK
├── rsa2048pem ---------- Directory for storing 2048-bit key files
├── rsa2048pem.sh ---------- 2048-bit key script
├── rsa4096pem ---------- Directory for storing 4096-bit key files
├── rsa4096pem.sh ---------- 4096-bit key script
├── secure_boot.cfg ---------- File used to set the AES KEY and IV value, If the AES Key and IV value are not null, the u-boot.bin and uboot_signature.bin images can be encrypted using the AES algorithm. If the AES KEY and IV values are null, the u-boot.bin and uboot_signature.bin images are not encrypted.
├── sha256.cfg ---------- File used to set the algorithm to be executed. The algorithm has been set to SHA256 and does not need to be modified.
└── u-boot- original.bin ---------- Non-secure U-Boot image
```

NOTICE

In the secure boot process, the SVB and DDR initialization code in **secureboot_release** is executed. The SVB and DDR initialization processes in U-Boot are not executed.

Therefore, in secure boot scenarios, if you want to update the SVB or DDR initialization process, files in the following directory must be modified:
osdrv/opensoruce/uboot/secureboot_release/ddr_init/drv/

```
├── cmd_bin
├── cmd_ddr_training_v2.c
```

```
├── ddr_cmd_ctl.c
├── ddr_cmd_loc.S
├── ddr_ddrc_v500.h
├── ddr_ddrc_v510.h
├── ddr_ddrc_v520.h
├── ddr_ddrt_s40.h
├── ddr_ddrt_t12_v100.h
├── ddr_ddrt_t16.h
├── ddr_ddrt_t28.h
├── ddr_interface.h
├── ddr_phy_s40.h
├── ddr_phy_t12_v100.h
├── ddr_phy_t12_v101.h
├── ddr_phy_t16.h
├── ddr_phy_t28.h
├── ddr_training_boot.c
├── ddr_training_console.c
├── ddr_training_ctl.c
├── ddr_training_custom.c
├── ddr_training_custom.h
├── ddr_training_impl.c
├── ddr_training_impl.h
├── ddr_training_internal_config.h
└── Makefile
```

osdrv/opensoruce/uboot/secureboot_release/ddr_init/boot/lowlevel_init_v300.c

In the SDK, the SVB and DDR initialization processes for secure boot are the same as those for non-secure U-Boot.

2 Generating the Secure Image

2.1 Generating Secure U-Boot Image

Perform the following to generate the secure U-Boot image:

Step 1 Generate a non-secure U-Boot image:

For details, see chapter 2 "Porting the U-Boot" in *Hi3516C V500/Hi3516D V300/Hi3516A V300 U-Boot Porting Development Guide*.

Step 2 Decompress the secure U-Boot SDK by running the following command:

tar xvf secureboot_release.tgz

Change the name of the non-secure U-Boot image generated in Step 1 to **u-boot-original.bin** and copy it to the **secureboot_release** directory.

Step 3 Configure the **KEY** and **IV** in the **secure_boot.cfg** file:

To generate a U-Boot image for common secure boot (unencrypted U-Boot), set **KEY** and **IV** in the **secure_boot.cfg** file to null. To generate a U-Boot image for encrypted secure boot, set **KEY** and **IV** in the **secure_boot.cfg** file as required, for example:

- KEY = 67452301efcdab8998badcfe10325476332211007766554 4bbaa9988ffeeddcc
- IV = 00112233445566778899aabbccddeeff

Step 4 Compile the U-Boot image for secure boot by running the following command:

cd secureboot_release

Then, Run **make all**.

The secure images **u-boot-rsa2048.bin** and **u-boot-rsa4096.bin** are generated in the **secureboot_release** directory.

----End

NOTICE

The public key file and private key file are generated during the first compilation by the release package script. These two files are to be used for the secure images during the subsequent compilation. To update the public key and private key, you need to manually delete the files in the **rsa2048pem** or **rsa4096pem** directory or run the **make distclean** command.

2.2 Key Files

```
├── rsa2048pem
│   ├── rsa2048_pem_hash_val.txt    //Public key hash and register configuration commands in .txt format
│   ├── rsa_priv_2048.pem           //Private key in .pem format
│   ├── rsa_pub_2048.bin            //Public key in .bin format
│   ├── rsa_pub_2048.pem            //Public key in .pem format
│   └── rsa_pub_2048_sha256.txt     //Hash value of the public key in .txt format
├── rsa4096pem
│   ├── rsa4096_pem_hash_val.txt    //Public key hash and register configuration commands in .txt format
│   ├── rsa_priv_4096.pem           //Private key in .pem format
│   ├── rsa_pub_4096.bin            //Public key in .bin format
│   ├── rsa_pub_4096.pem            //Public key in .pem format
│   └── rsa_pub_4096_sha256.txt     //Hash value of the public key in .txt format
├aes_otp_cfg.txt                   //AES KEY register configuration commands
```

3 OTP Burning

The One Time Programmable (OTP) burning procedure is as follows:

NOTICE

Be very careful while performing the following steps. Otherwise, the chip may be damaged.

To generate a common secure U-boot image, set **KEY** and **IV** in the **secure_boot.cfg** file to null.

Step 1 Burn the non-secure U-Boot and start U-Boot to the command lines.

Step 2 (Mandatory) Burn the public key hash by running the following commands:

mw 0x100b0008 0x6

mw 0x100b000c 0Xxxxxxxxx

mw 0x100b0010 0Xxxxxxxxx

mw 0x100b0014 0Xxxxxxxxx

mw 0x100b0018 0Xxxxxxxxx

mw 0x100b001c 0Xxxxxxxxx

mw 0x100b0020 0Xxxxxxxxx

mw 0x100b0024 0Xxxxxxxxx

mw 0x100b0028 0Xxxxxxxxx

NOTE

The preceding hash configuration commands can be directly copied from **rsa2048_pem_hash_val.txt** or **rsa4096_pem_hash_val.txt**.

mw 0x100b0000 0x2

mw 0x100b0004 0x1acce551

Step 3 (Mandatory) Burn the secure boot bits by running the following commands:

mw 0x100b0034 0x0

mw 0x100b0030 0x1

mw 0x100b0000 0x4

mw 0x100b0004 0x1acce551

Step 4 (Optional) Burn the DDR scrambling bits by running the following commands:

mw 0x100b0034 0x1

mw 0x100b0030 0x2

mw 0x100b0000 0x4

mw 0x100b0004 0x1acce551

Step 5 (Optional) Burn the AES keys:

mw 0x100b0008 0x0

mw 0x100b000c 0Xxxxxxxxx

mw 0x100b0010 0Xxxxxxxxx

mw 0x100b0014 0Xxxxxxxxx

mw 0x100b0018 0Xxxxxxxxx

mw 0x100b001c 0Xxxxxxxxx

mw 0x100b0020 0Xxxxxxxxx

mw 0x100b0024 0Xxxxxxxxx

mw 0x100b0028 0Xxxxxxxxx

NOTE

The preceding AES KEY configuration commands can be directly copied from **aes_otp_cfg.txt**. This step needs to be performed only when the encrypted secure boot image is burnt.

mw 0x100b0000 0x2

mw 0x100b0004 0x1acce551

Step 6 Burn the secure image to the boot medium by running the U-Boot command, or burn the secure image to the boot medium by using HiTool.

----End