Security Class: Top-Secret( ) Secret( ) Internal( ) Public(√)

RK356X SecurityBoot And AVB Instruction

File Status:
[ ] Draft
[√] Released
[ ] Modifying

File ID:	RK-YH-YF-292
Current Version:	V1.0.0
Author:	Wu Liangqing
Finish Date:	2021-05-09
Auditor:	
Audit Date:	2021-05-09

Version No.	Author	Modified Date	Modification Description	Remark
V1.0	Wu Liangqing	2021-5-9	Initial Release	

If there is any question about the document, please email to: wlq@rock-chips.com

DISCLAIMER

Trademark Statement

All rights reserved. ©2020. Rockchip Electronics Co., Ltd.

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Code Environment

Only Android11 RKR7 and later versions support security boot and AVB.

SecurityBoot Operation Steps

Confirm Compiling Environment

Confirm whether fdtput version of the compiling server is 1.4.5 or not.

```
#fdtput --version
#Version: DTC 1.4.5
```

If fdtput version is older than 1.4.5, please execute the following command to upgrade.

```
sudo apt-get install device-tree-compiler
```

1. Enter U-Boot directory

```
cd u-boot
```

Execute all the following steps in u-boot directory.

2. Code Modification

Enter u-boot directory, open 'configs/rk3568_defconfig' corresponding to the platform, and then select the configuration as follows.

```
//Edit the document'configs/rk3568_defconfig', both RK3566 and RK3568 need to
modify thi file.
vim configs/rk3568_defconfig
// mandaroty.
CONFIG_FIT_SIGNATURE=y
CONFIG_SPL_FIT_SIGNATURE=y

//optional.
CONFIG_FIT_ROLLBACK_PROTECT=y        // boot.img anti-rollback
CONFIG_SPL_FIT_ROLLBACK_PROTECT=y    // uboot.img anti-rollback
```

3. Keys Generation

Execute the following operations in u-boot directory to create keys.

```
mkdir -p keys
../rkbin/tools/rk_sign_tool kk --bits 2048 --out .
cp privateKey.pem keys/dev.key && cp publicKey.pem keys/dev.pubkey
openssl req -batch -new -x509 -key keys/dev.key -out keys/dev.crt
```

Note: You only have to execute this step once, and then save these keys properly.

4. Compile Signature

Here we take RK3566 as an example, and you can change rk3566 to rk3568 if the chip is RK3568.

```
./make.sh rk3566 --spl-new --rollback-index-uboot 1 --burn-key-hash
```

Instruction:
--spl-new //re-package the signed spl
--rollback-index-uboot < version number> //set the version number, when config in step2 is configured as anti-rollback, you need to add this compiling option, otherwise it is not needed.
--burn-key-hash //If you add this compiling option, the chip will fuse during boot up after flashing the image.

If it occurs during compiling:

```
Can't load XXXXXX//.rnd into RNG
```

Execute:

```
touch ~/.rnd
```

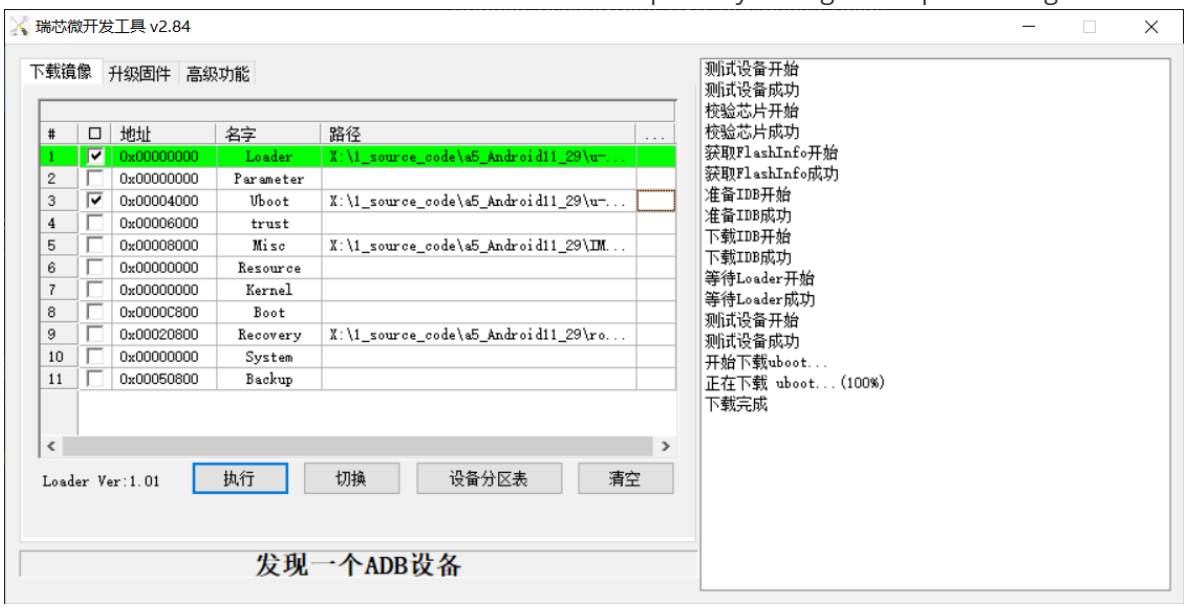
5. Compile Complete Image

Compile other images in normal way (uboot and loader have been already compiled, no need to compile again), for example:

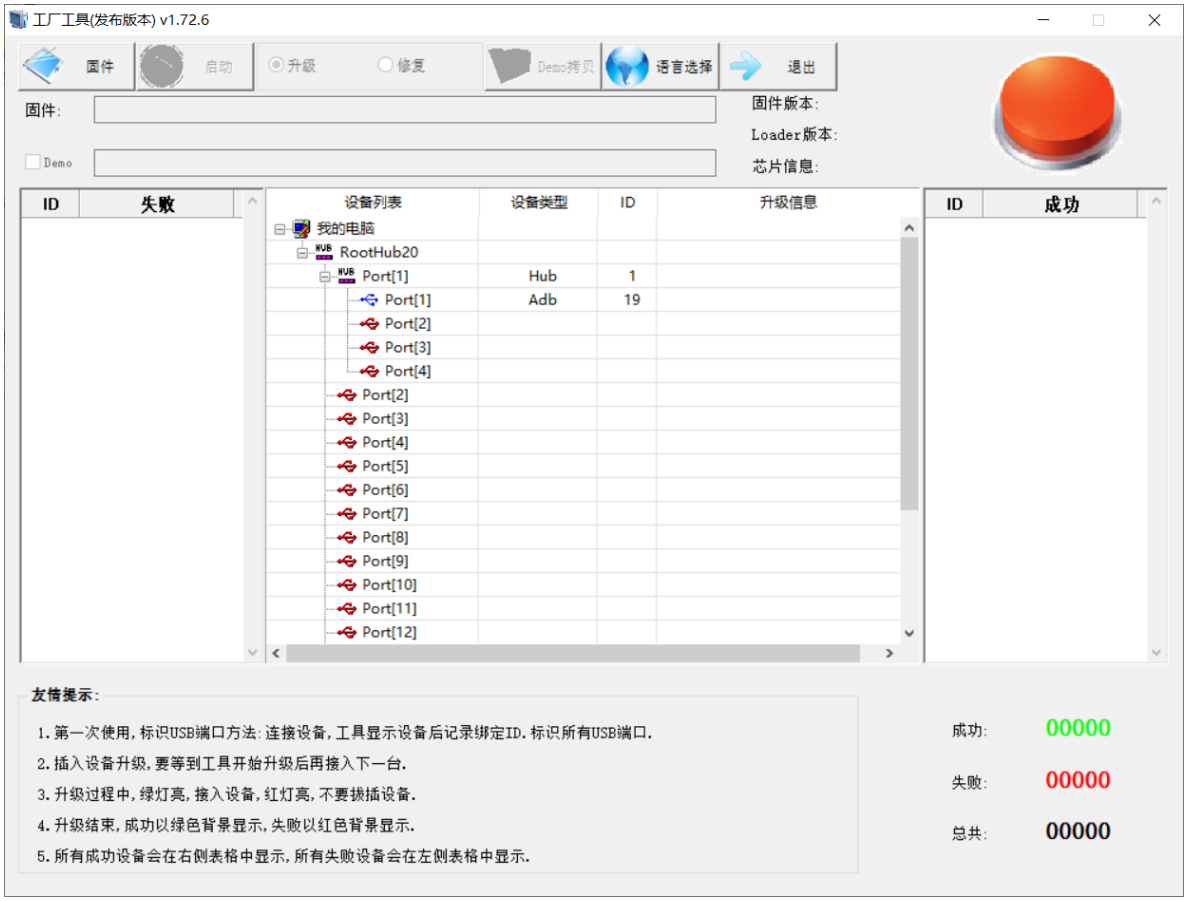
```
source build/envsetup.sh
lunch rk3566_r-userdebug
./build.sh -ACKup
```

6. Image Flashing

You can use AndroidTool to flash uboot and loader separately during development stage.



You can use FactoryTool to flash update.img compiled and generated in step5 during mass production.



6. Judge Whether Fusing Is Successful

Judge by Serial Port Log of Boot-up

```
## Verified-boot: 0   //The image is signed but the chip isn't fused (1 means
being fused), the validity of hash is not verified , that is,'--burn-key-hash'
isn't added when compiling.
sha256,rsa2048:dev+
rollback index: 1 >= 0(min), OK   //rollback version, that is,'--rollback-index-
uboot' is added when compiling.
//The followings are validations of uboot integrity.
## Checking atf-1 0x00040000 ... sha256+ OK
## Checking uboot 0x00a00000 ... sha256+ OK
## Checking fdt 0x00b2a018 ... sha256+ OK
## Checking atf-2 0xfdcc9000 ... sha256+ OK
## Checking atf-3 0xfdcd0000 ... sha256+ OK
## Checking optee 0x00200000 ... sha256+ OK
```

Judge by Flashing Unsigned Loader And Uboot

Operation Steps of Android Verified Boot(AVB)

1. Compile avbtool Tool

```
mmma external/avb/ -j16
```

Generate after compiling finished:

```
out/host/linux-x86/bin/avbtool
```

2. Generate atx_permanent_attributes.bin

- **Modify Product ID**

```
cd external/avb/test
```

```
diff --git a/test/avb_atx_generate_test_data b/test/avb_atx_generate_test_data
index 1b8bb2b..2220688 100755
--- a/test/avb_atx_generate_test_data
+++ b/test/avb_atx_generate_test_data
@@ -48,7 +48,7 @@ AVBTOOL=$(dirname "$0")/../avbtool
 echo AVBTOOL = ${AVBTOOL}

 # Get a zero product ID.
-echo 00000000000000000000000000000000 | xxd -r -p - atx_product_id.bin
+echo 00000000000000000000000000000123 | xxd -r -p - atx_product_id.bin

 # Generate key pairs.
 if [ ! -f testkey_atx_prk.pem ]; then
```

```
cd -
```

Note:The digit of product ID is 16bit, and you can define the value by yourself.

- **Generate atx_permanent_attributes.bin**

```
cd external/avb/test/data
```

```
../avb_atx_generate_test_data
```

```
cd -
```

After executing the above operations, it will generate in external/avb/test/data:

- atx_permanent_attributes.bin
- atx_metadata.bin
- testkey_atx_pik.pem
- testkey_atx_prk.pem
- testkey_atx_psk.pem

Note:
-There is one pem document in the system by default, if you need to re-generate, delete the default document in the system and then execute the operations above to re-generate a pem document again. It's recommended for customers to re-generate by themselves.
-You just need to execute this step once for one product. Please keep carefully the documents generated above, it will be used in the following steps.

3. Code Modification

```
cd device/rockchip/rk356x
```

```
diff --git a/rk3566_r/BoardConfig.mk b/rk3566_r/BoardConfig.mk
index 24b415f..80fa60f 100644
--- a/rk3566_r/BoardConfig.mk
+++ b/rk3566_r/BoardConfig.mk
@@ -37,3 +37,7 @@ ifeq ($(strip $(BOARD_USES_AB_IMAGE)), true)
   include device/rockchip/common/BoardConfig_AB.mk
   TARGET_RECOVERY_FSTAB := device/rockchip/rk356x/rk3566_r/recovery.fstab_AB
 endif
+
+BOARD_AVB_ENABLE := true    //Enable AVB function
+BOARD_AVB_ALGORITHM := SHA256_RSA4096  //Configure encipher algorithm
+BOARD_AVB_KEY_PATH := external/avb/test/data/testkey_atx_psk.pem  //The path to save the keys
+BOARD_AVB_METADATA_BIN_PATH := external/avb/test/data/atx_metadata.bin //Specify metadata documents
+#BOARD_AVB_ROLLBACK_INDEX := 5   //Configure the version anti-rollback, which is disabled by default, and can be enabled according to the requirement.
~~~shell
cd -
cd u-boot
```

```
diff --git a/configs/rk3568_defconfig b/configs/rk3568_defconfig
index 3017921487..84197eea1e 100644
--- a/configs/rk3568_defconfig
+++ b/configs/rk3568_defconfig
@@ -214,5 +214,9 @@ CONFIG_AVB_LIBAVB_AB=y
 CONFIG_AVB_LIBAVB_ATX=y
 CONFIG_AVB_LIBAVB_USER=y
 CONFIG_RK_AVB_LIBAVB_USER=y
+CONFIG_RK_AVB_LIBAVB_ENABLE_ATH_UNLOCK=y
+CONFIG_AVB_VBMETA_PUBLIC_KEY_VALIDATE=y
```

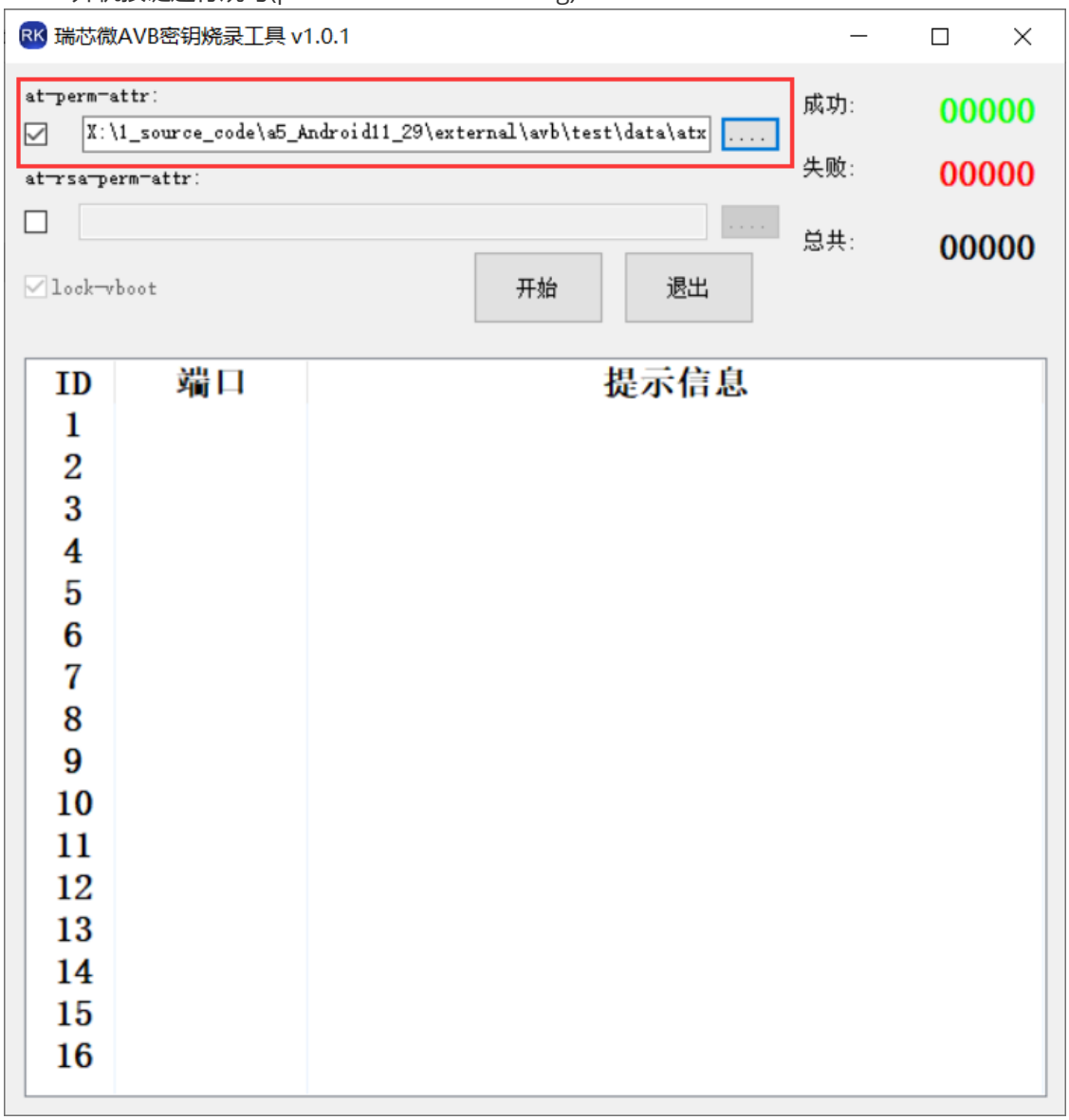
```
cd -
```

4. Flash AVB Key

Flashing tools: AvbKeyWriter（RKTools/windows/AvbKeyWriter-v1.0.1.7z)
Flashing the source file: atx_permanent_attributes.bin generated by external/avb/test/data which is generated in step1.
Flashing methods:

- Tick at-perm-attr
- Import external/avb/test/data generated in step3 to generate atx_permanent_attributes.bin.
- Waiting for the flashing device entering the loader mode.

- Click “开机按键进行烧写(power button for flashing)”.



5. Image Compiling

You can compile the complete image after all of steps above are finished, we take RK3566_r product as an example to compile as follows:

-The solution with secure boot

Uboot of the solution with secure boot, should be compiled alone first, referring to the operation steps of secure boot above.

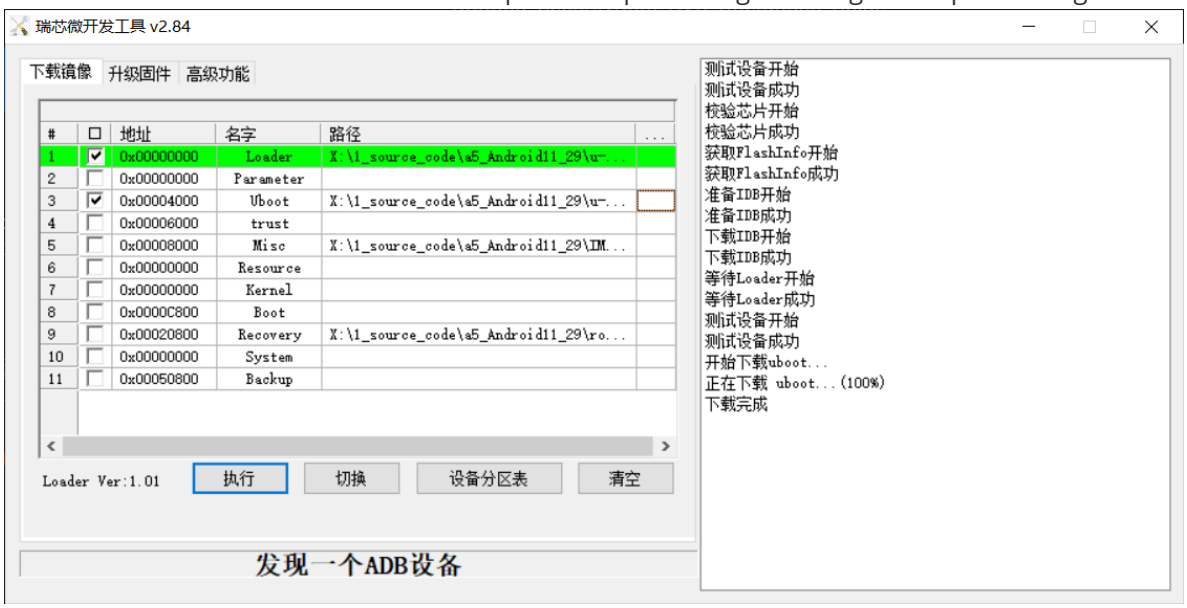
```
source build/envsetup.sh
lunch rk3566_r-userdebug
./build.sh -ACKup
```

- **The solution without secure boot**

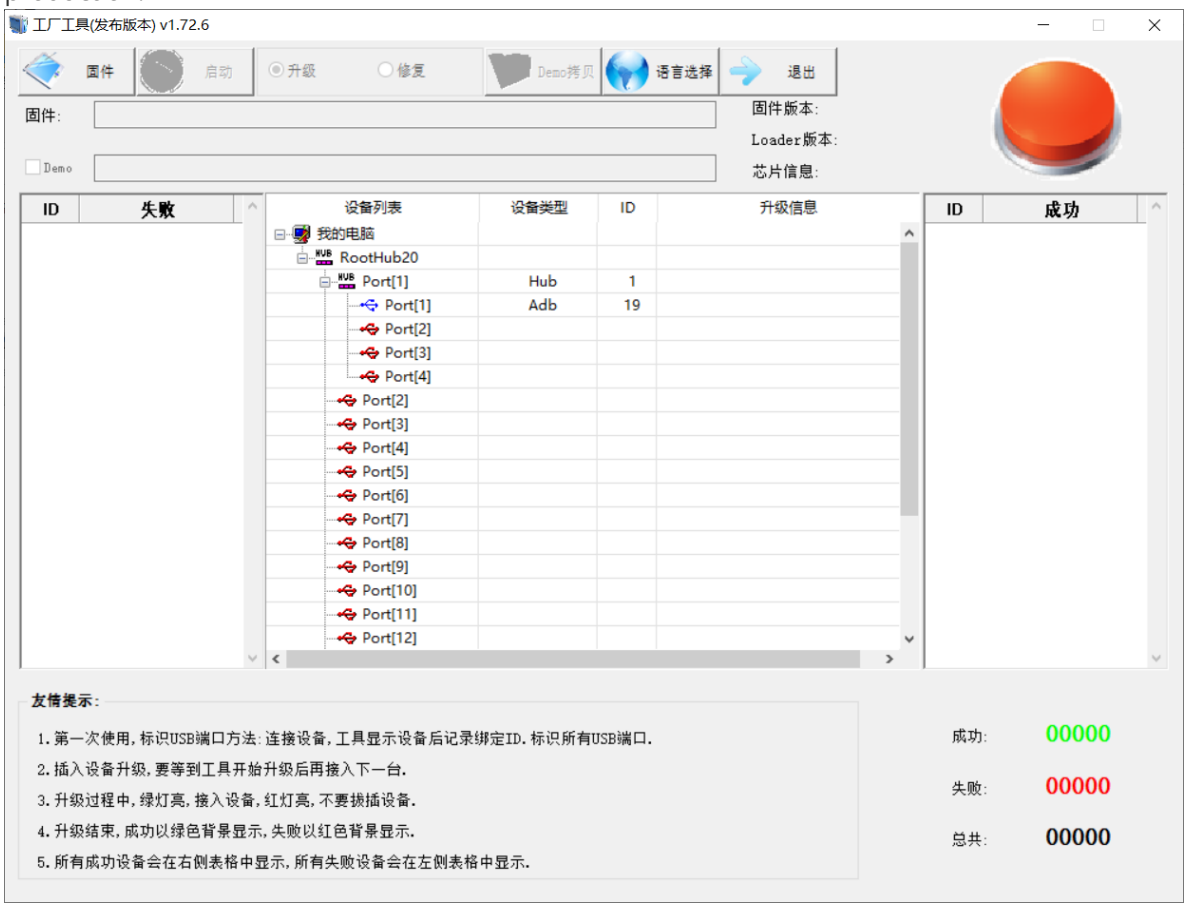
```
source build/envsetup.sh
lunch rk3566_r-userdebug
./build.sh -ACKUup
```

6. Image flashing

You can use AndroidTool to flash the compiled complete images during development stage.



You can use FactoryTool to flash update.img compiled and generated in step4 during mass production.



7. Boot-up Verification

- **Confirm uboot boot-up log**
  After the above steps, the serial port will print the following log in u-boot stage when the system is powered on.

```
Vboot=0, AVB images, AVB verify
read_is_device_unlocked() ops returned that device is LOCKED
ANDROID: Hash OK
```

- It will fail to boot-up if flashing non-AVB image or other images