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

RK3399_Efuse_Operation_Instructions

(Technical Department, R & D Dept. II)

Status： [ ] Modifying [√] Released	Version：	V1.00
	Author：	Wu Liangqing, Wei Jianxing
	Date：	2019-2-14
	Auditor：	
	Date：	

Revision History

Version no.	Author	Revision Date	Revision description	Remark
V1.00	Wu Liangqing, Wei Jianxing	2019-2-14	Initial version release	

Content

Preface

Overview

This document mainly describes Rockchip RK3399 efuse burning method and other related notices.

Product version

Chip name	Kernel version	Android version
RK3399	Linux4.4	

Object

This document (guide) is mainly suitable for below engineers:

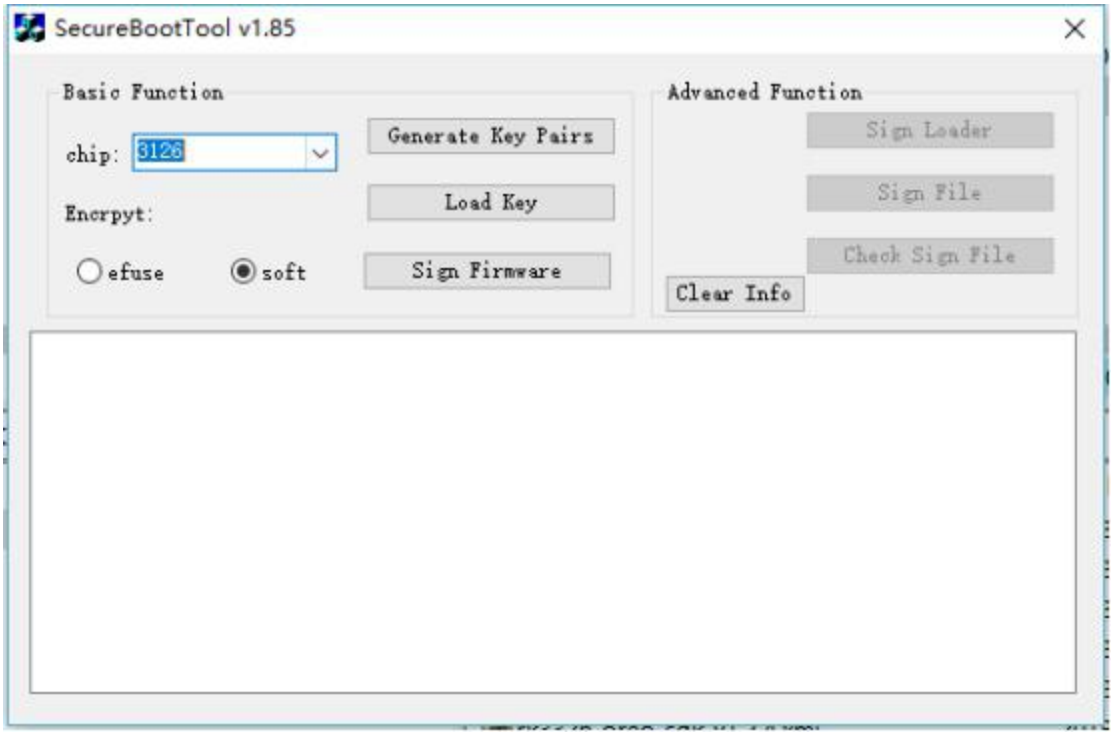
Field application engineers

Software development engineers

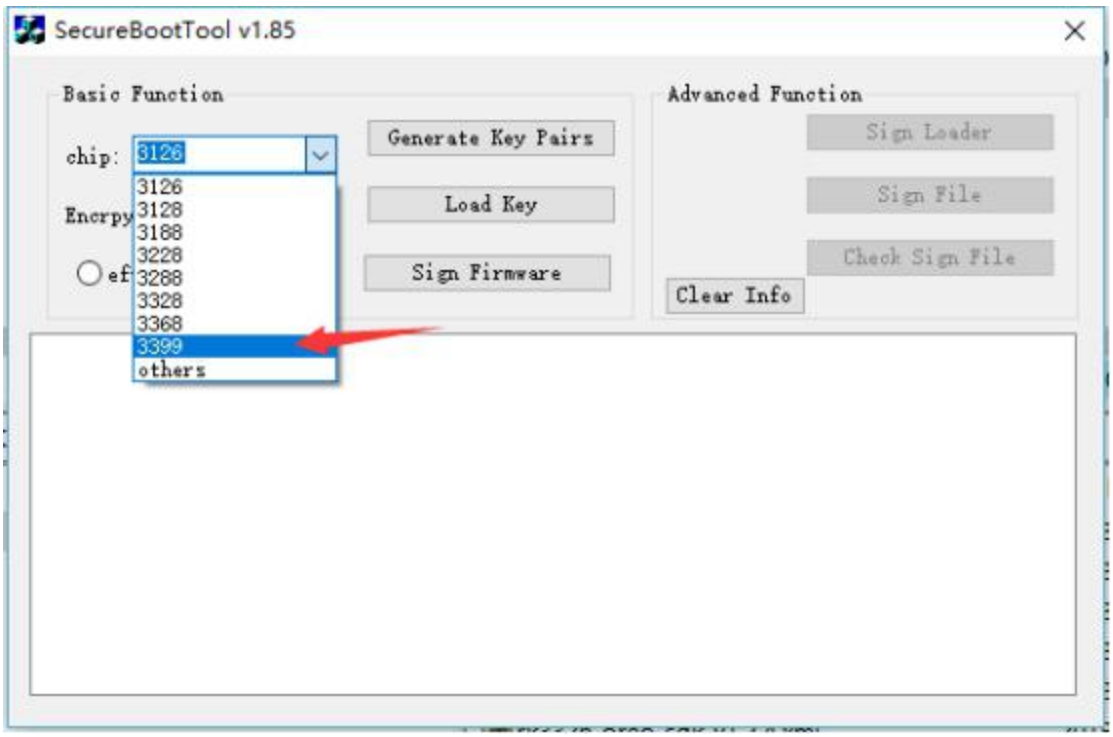
1 Windows tools signature step

1.1 Generate public keys and private keys

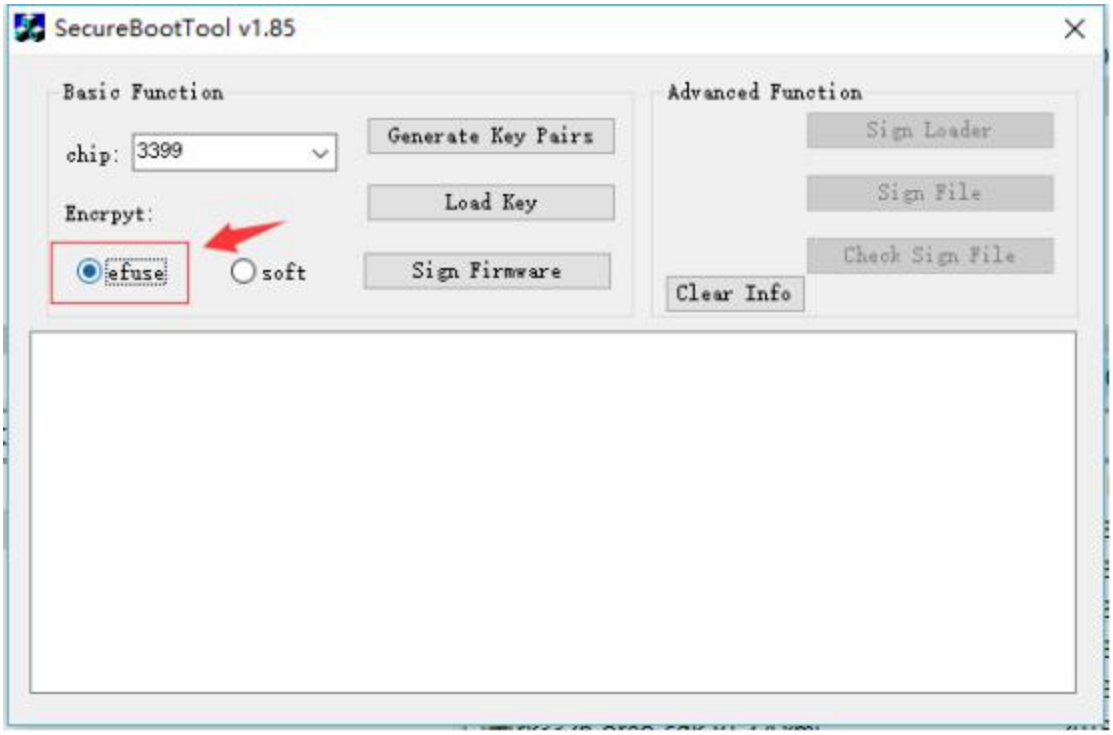
1) Open tools "SecureBootTool v.1.85"



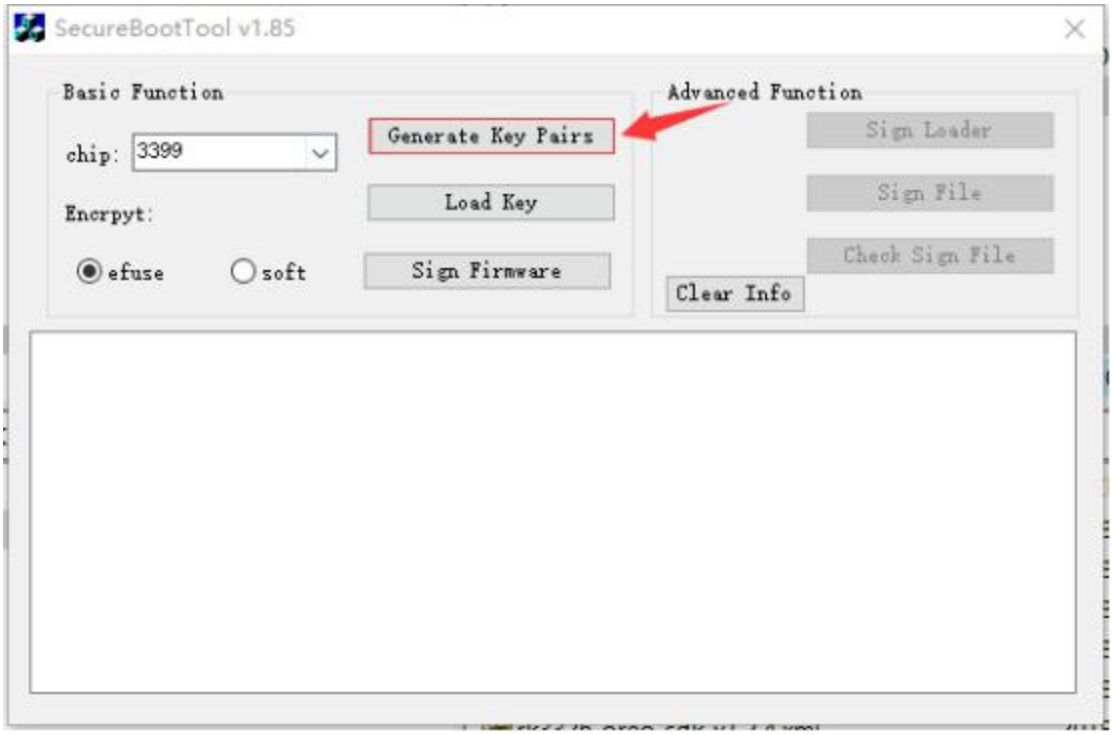
2) Select chip：3399



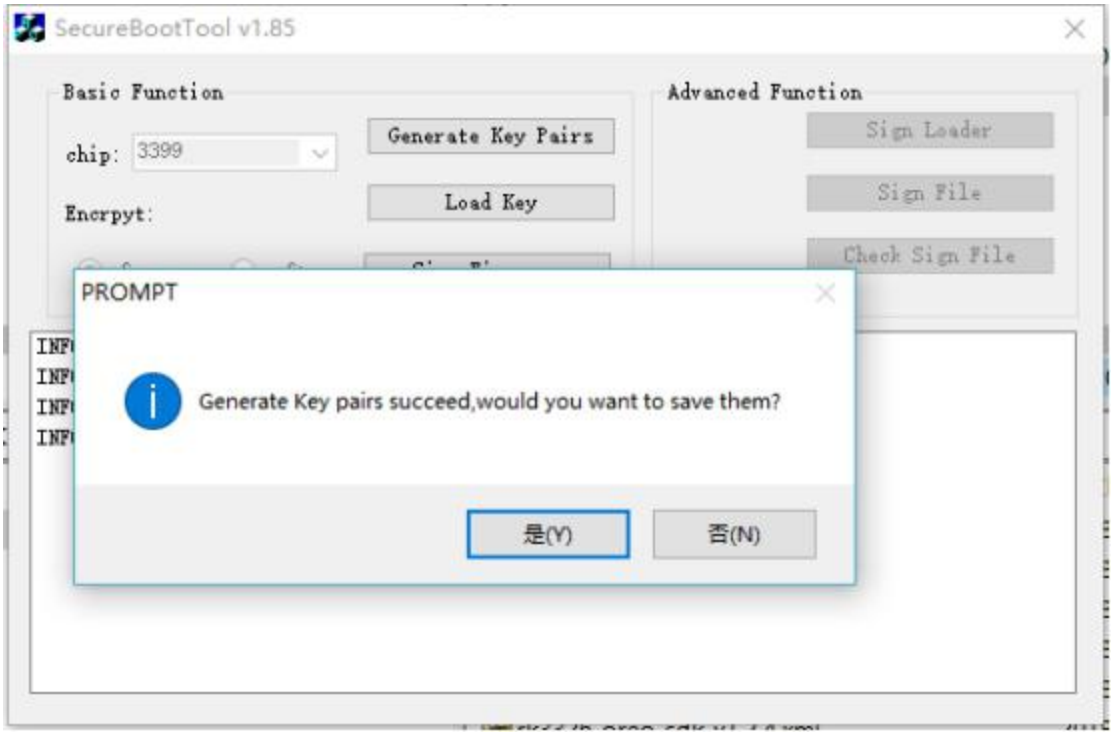
3) Select "Encrpyt" type: efuse



4) Select "Generate Key Pairs" to create key



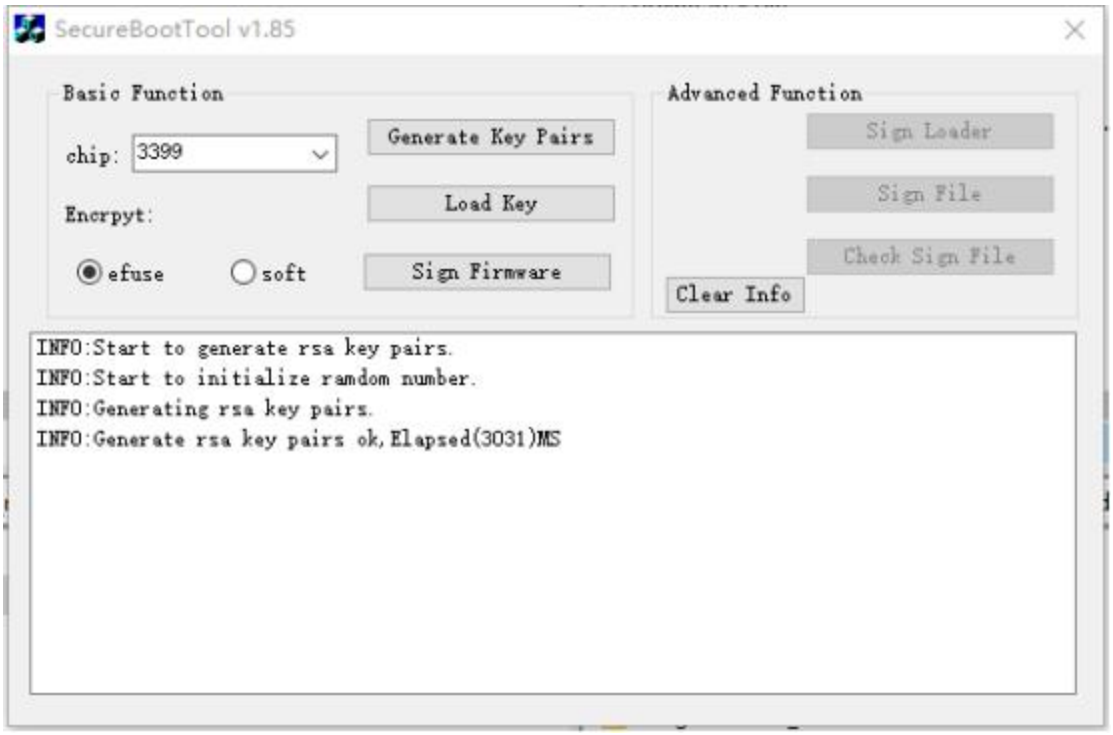
5) Create successfully and save



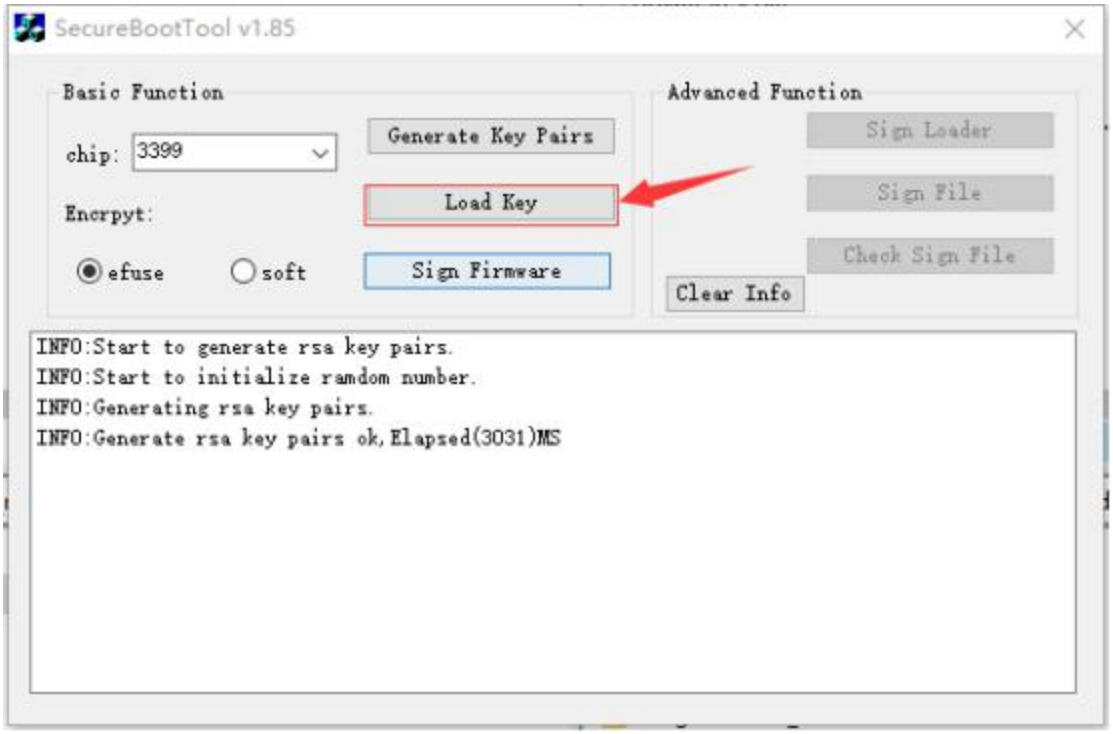
6) After saving the following key file

1.2 Signature firmware

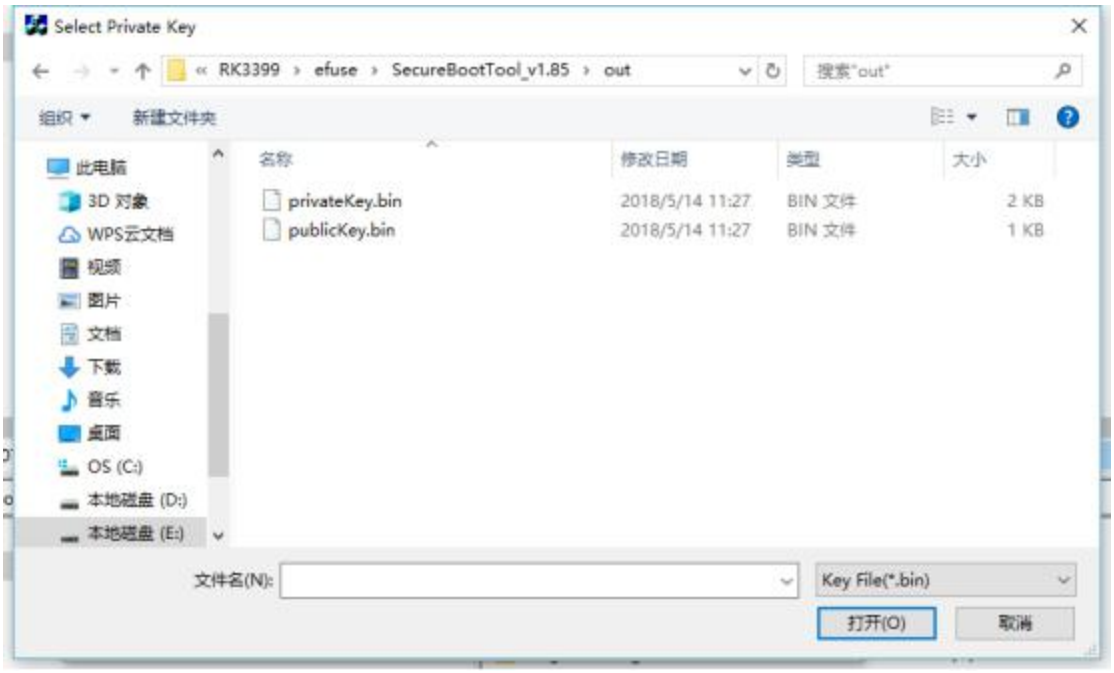
1) Open tool SecureBootTool v.1.85, configure as create key



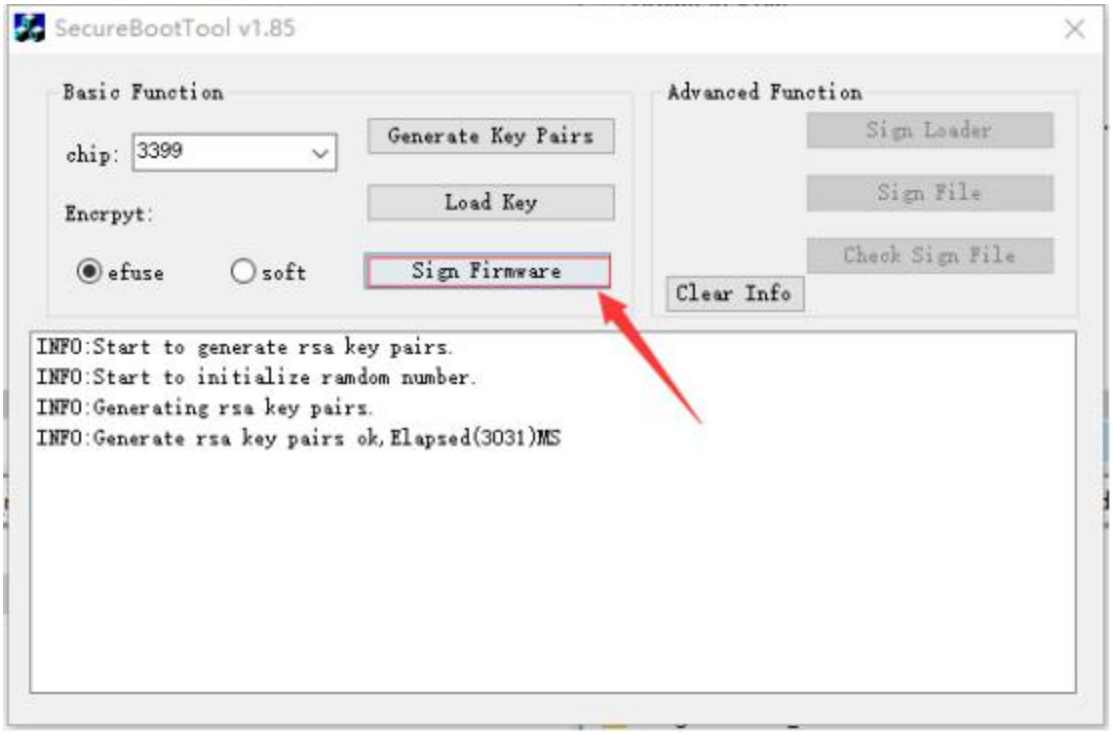
2) Click "Load Key" button to load key



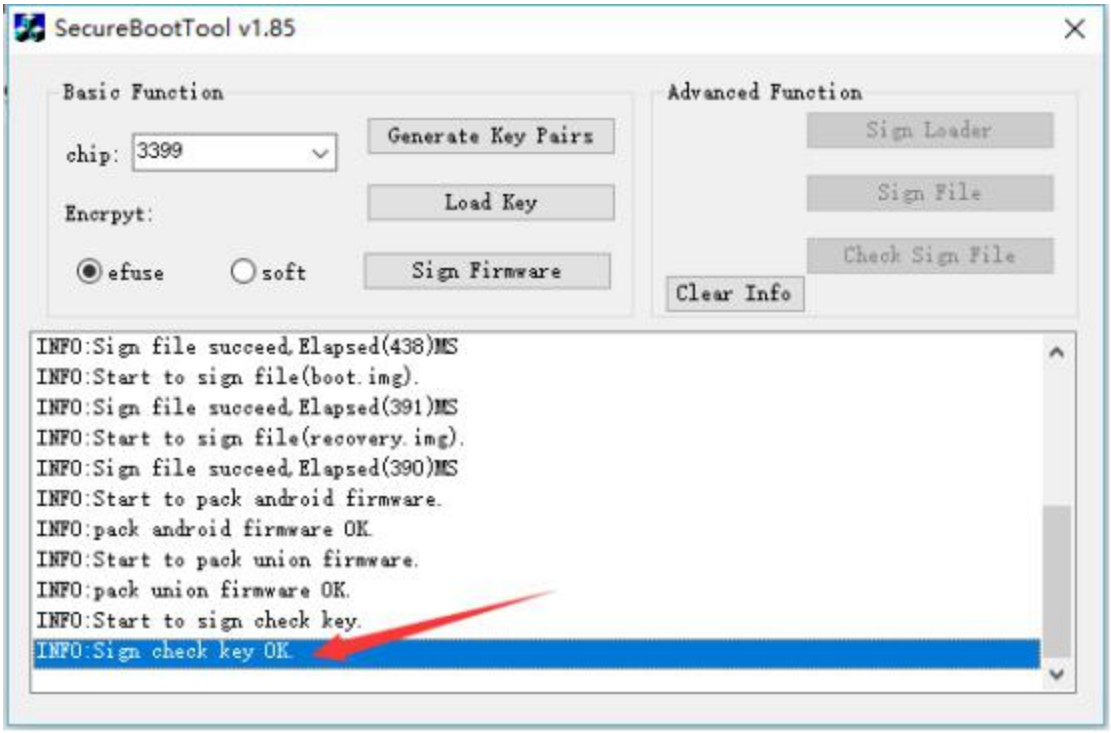
3) Select the key files generated in the previous step, and the public and private keys need to be loaded twice.



4) Click the "Sign Firmware" button to select the target firmware. For more details about the compilation and packaging of Firmware, please refer to section 2 of rockchip_secure_boot_application_note_v1.2.1 _20171128.pdf.



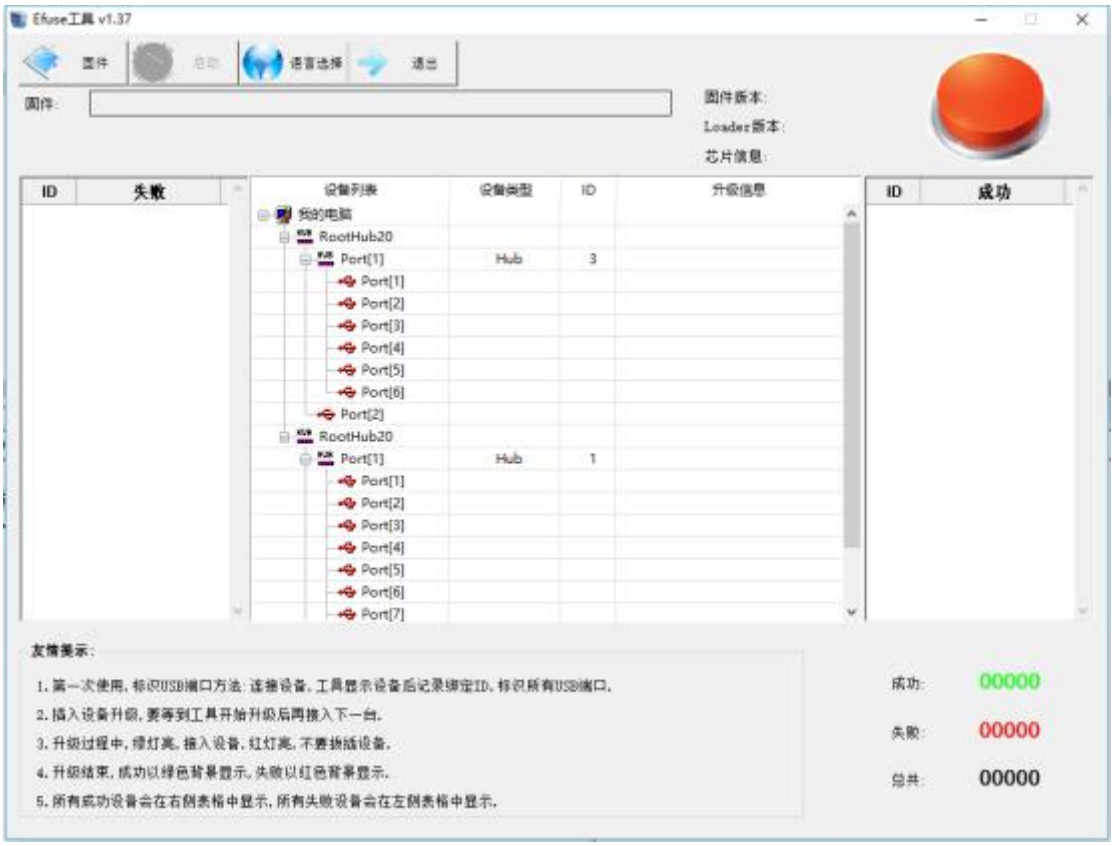
5) The firmware signing process takes about 10 minutes, and the signed firmware is under the out directory in the path where the tool is located, and its name is update_signal.img



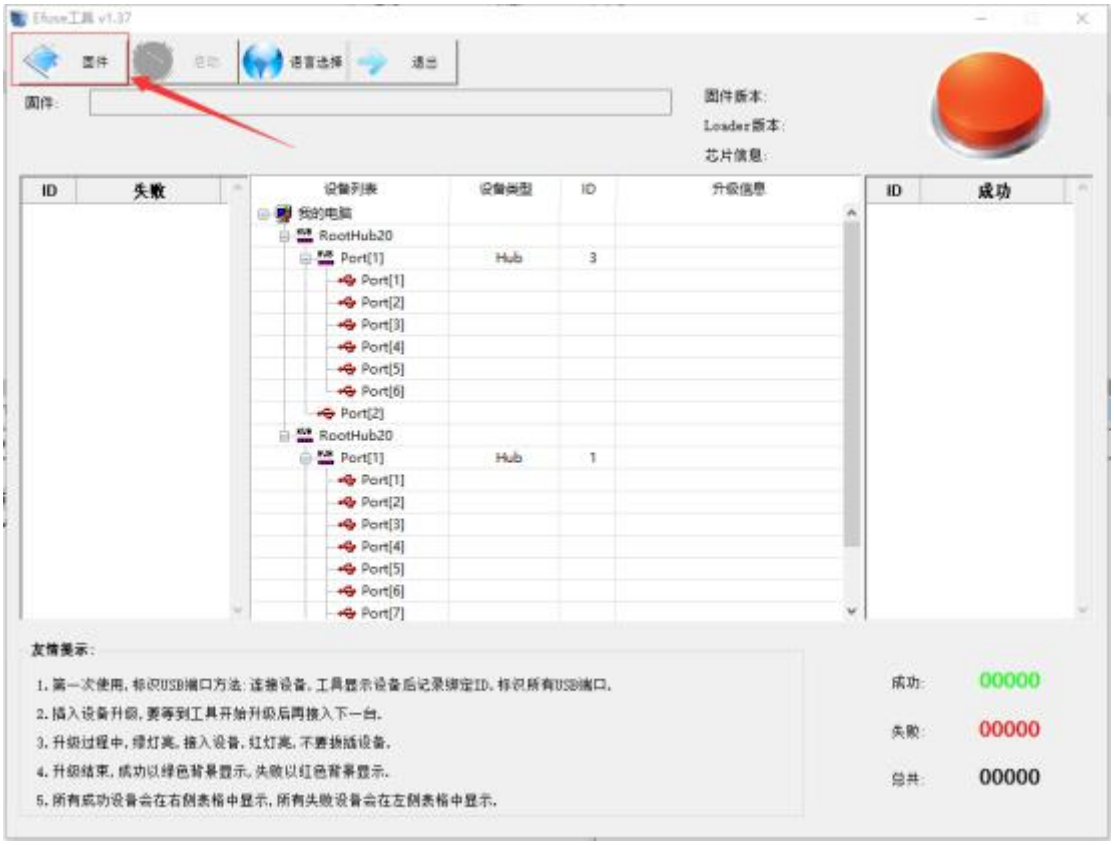
1.3 Programme efuse

RK3399 board configuration: efuse power on, enter “masrom” mode

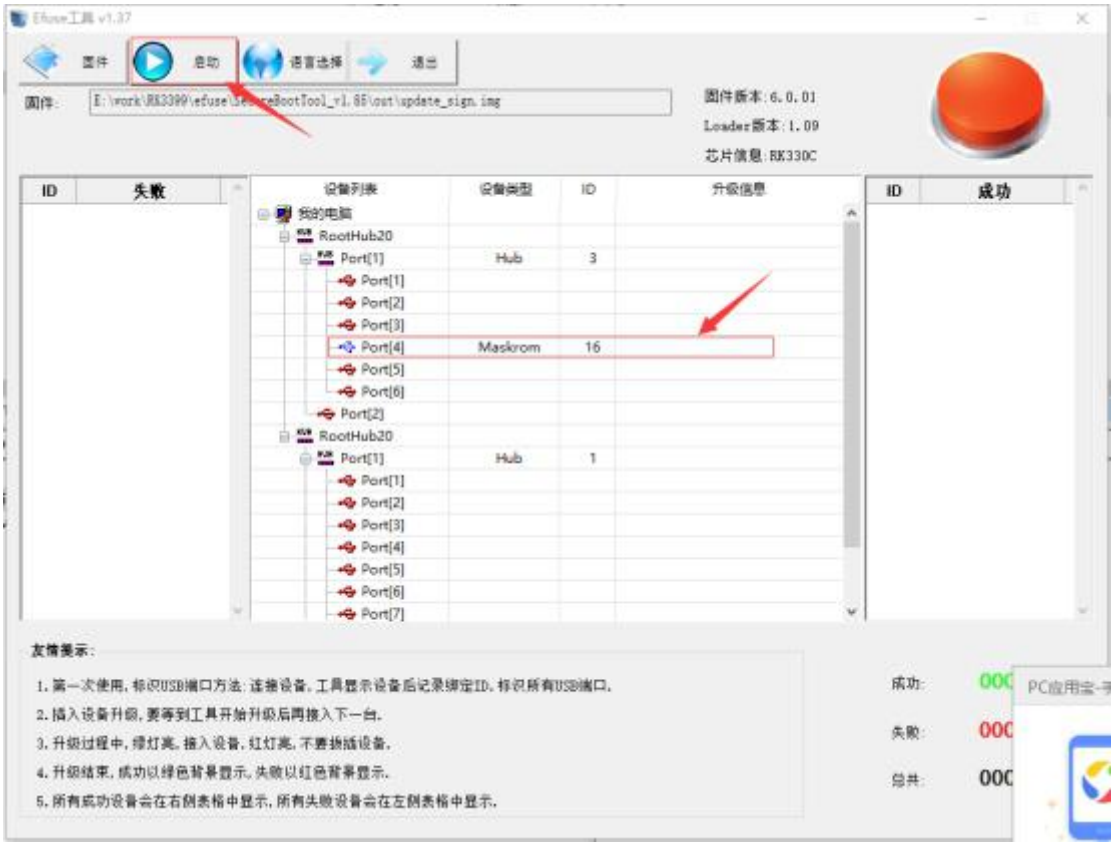
1) Open tool: ”Efuse 工具 v1.37”



2) Select the firmware signed in previous step



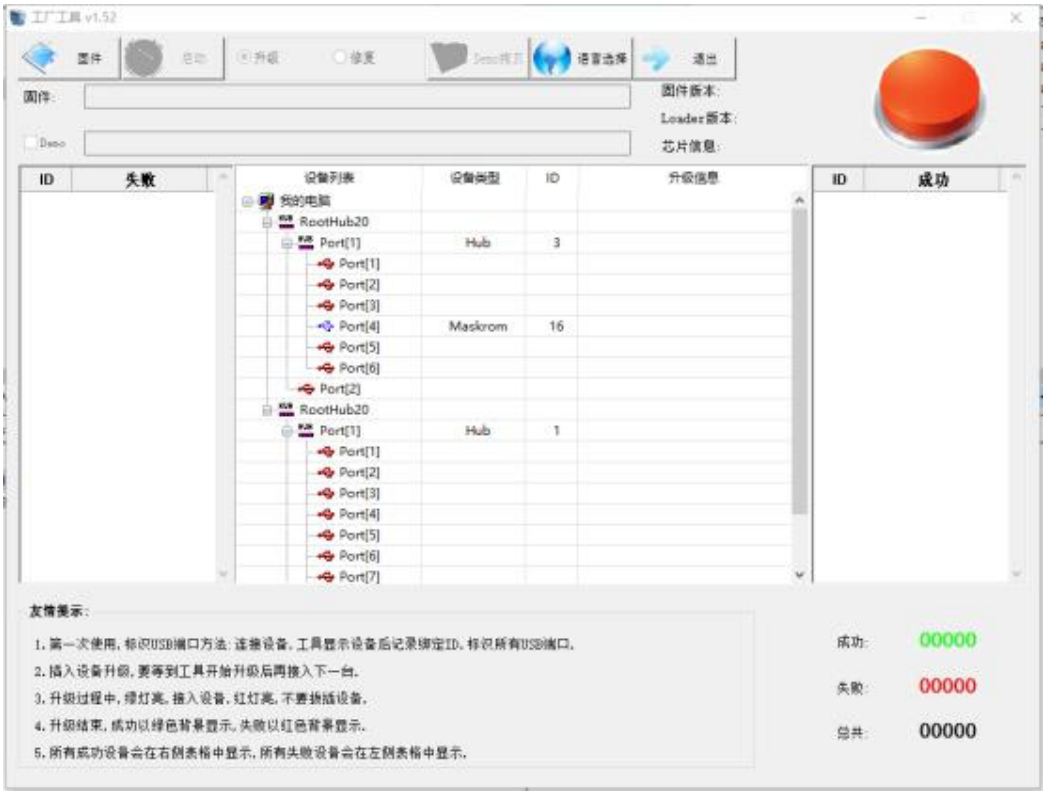
3) After identifying the maskrom device, click start to write



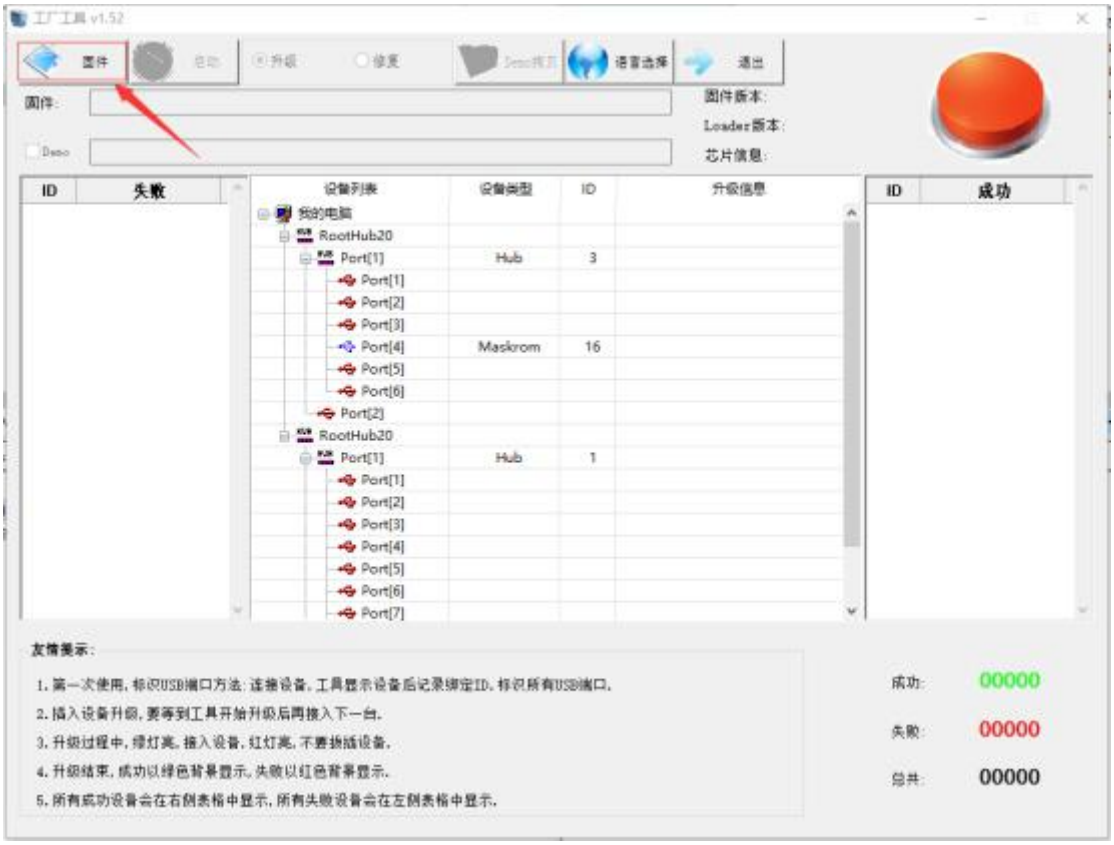
4) Burning successfully

1.4 Burning signature firmware

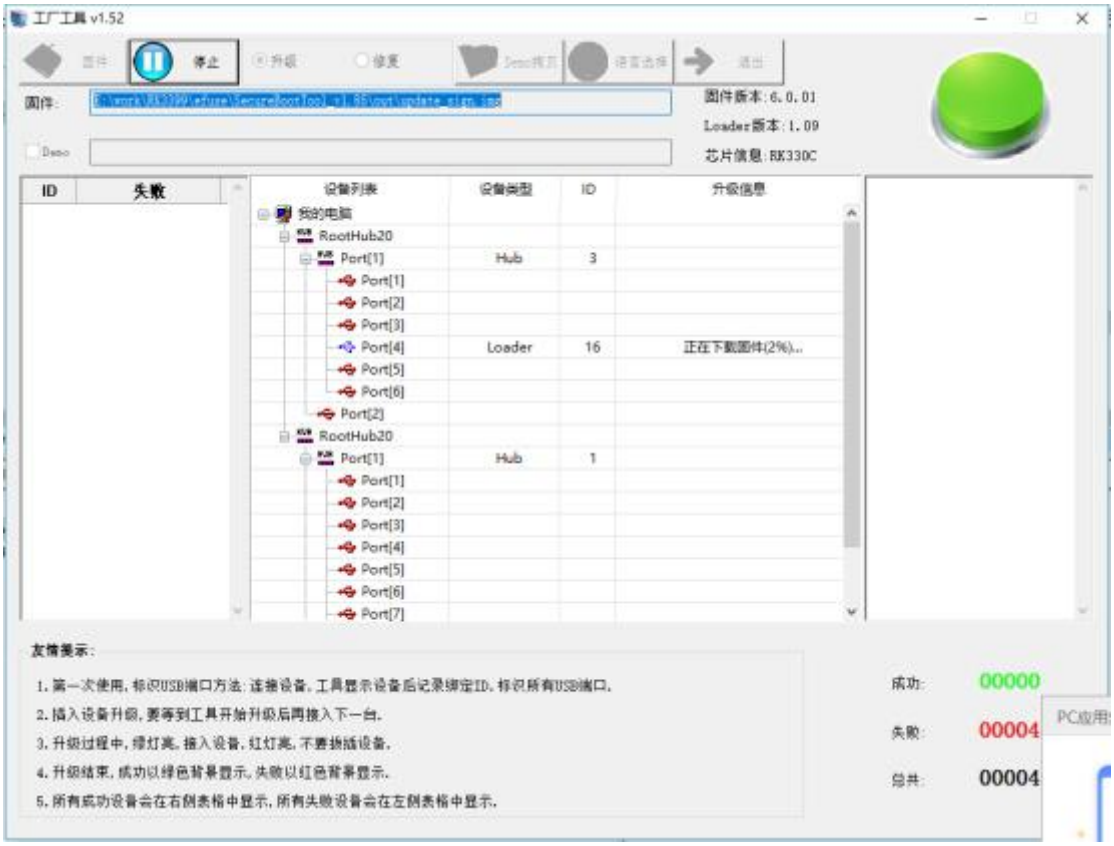
1) Open tool: ”FactoryTool_v1.63”



2) Select the firmware signed in previous step



3) Start to download



2 Linux tools signature step

2.1 Generate Key Pairs

SecureBootConsole -k|-kk SaveKeyDir //generate key -k(1024) -kk(2048)

2.2 Sign firmware

trust.img uboot.img recovery.img boot.img

SecureBootConsole -si privatekey _path image_path

2.3 Sign loader

//SignEx Loader （efuse 2048）

SecureBootConsole -slx privatekey_path publickey _pathl oader_path

2.4 Package update.img

Package the signed images such as Loader, trust.img, uboot.img, recovery.img and boot.img into update.img.

2.5 Sign update.img md5

SecureBootConsole -sh privateKey _path firmware_path

2.6 RK3399 commands

SecureBootConsole -kk SaveKeyDir

sudo ./SecureBootConsole -slx privateKey.bin publicKey.bin Image/MiniLoaderAll.bin

sudo ./SecureBootConsole -si privateKey.bin Image/recovery.img

sudo ./SecureBootConsole -si privateKey.bin Image/boot.img

sudo ./SecureBootConsole -si privateKey.bin Image/trust.img

sudo ./SecureBootConsole -si privateKey.bin Image/uboot.img

./mkupdate.sh

sudo ./SecureBootConsole -sh privateKey.bin update.img

3 Make efuse ota update.zip

1) Make sure build/tools/ has "drmsigntool" sign tool, if it is already existing, no need to add manually.

```
diff --git a/build/tools/drmsigntool/Android.mk b/build/tools/drmsigntool/Android.mk
new file mode 100755
index 0000000..a6c245c
--- /dev/null
+++ b/build/tools/drmsigntool/Android.mk
@@ -0,0 +1,11 @@
+#=========================================================
+# Android.mk for drmsigntool
+#
+# sign the boot.img and recovery.img for drm secure boot
+# ========================================================
+
+LOCAL_PATH:= $(call my-dir)
+include $(CLEAR_VARS)
+LOCAL_MODULE_TAGS := optional
+LOCAL_PREBUILT_EXECUTABLES := drmsigntool
+include $(BUILD_HOST_PREBUILT)
diff --git a/build/tools/drmsigntool/drmsigntool b/build/tools/drmsigntool/drmsigntool
new file mode 100755
index 0000000..1cf781a
Binary files /dev/null and b/build/tools/drmsigntool/drmsigntool differ
diff --git a/build/tools/releasetools/common.py b/build/tools/releasetools/common.py
index 26b0007..af4db33 100755
--- a/build/tools/releasetools/common.py
+++ b/build/tools/releasetools/common.py
@@ -498,6 +498,11 @@ def _BuildBootableImage(sourcedir, fs_config_file, info_dict=None,
   assert p.returncode == 0, "mkbootimg of %s image failed" % (
       os.path.basename(sourcedir),)

+  sign_cmd = ["drmsigntool", img.name, "build/target/product/security/privateKey.bin"]
+  p4 = Run(sign_cmd)
+  p4.communicate()
+  assert p4.returncode == 0, "mkbootimg of %s image failed" % (os.path.basename(sourcedir),)
+
   if (info_dict.get("boot_signer", None) == "true" and
       info_dict.get("verity_key", None)):
     # Hard-code the path as "/boot" for two-step special recovery image (which
diff --git a/device/rockchip/common/device.mk b/device/rockchip/common/device.mk
index 71d8f94..6291207 100755
--- a/device/rockchip/common/device.mk
+++ b/device/rockchip/common/device.mk
@@ -14,6 +14,12 @@
 # limitations under the License.
 #

+# lpz@neostra add 20180526
+# drmsigntool for SecureBoot
+PRODUCT_PACKAGES += \
+    drmsigntool
+# lpz@neostra end
+
 # Prebuild apps
```

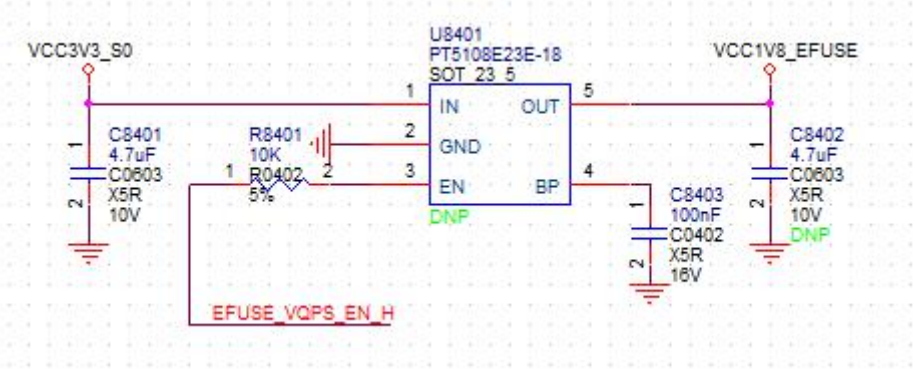
2) Copy key to build/target/product/security/

Need to sign loader, uboot and trust before making ota package and then put it into SDK project to replace the original file.

3) Make ota package as normal

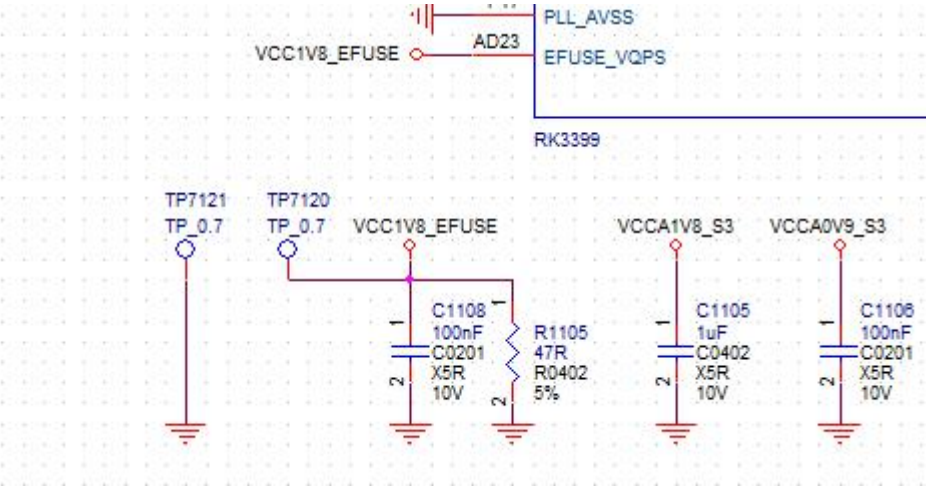
4 Efuse power up

1） Use the power supply method of reference design, as shown in below picture:



The above circuit is designed in the test fixture (recommend to use the LDO with voltage adjustable, then you can increase the LDO voltage in case there is voltage reduction on probe) to save cost. When flashing EFUSE, RK3399 （GPIO4_D3） EFUSE_VQPS_EN_H will output high to enable LDO to supply power for EFUSE_VQPS.

2） Directly use the DC regulated power supply to supply power for EFUSE_VQPS:



The two test points shown in above picture are designed in the main board. Directly

use the DC regulated power supply to supply 1.8V for the test fixture. Make sure the main board enters masrom, then supply 1.8V for EFUSE_VQPS, and finally flash EFUSE.