密级状态：绝密( ) 秘密( ) 内部( ) 公开(√)

RK3399 SecurityBoot和AVB操作指南

文件状态:
[ ] 草稿
[√] 正式发布
[ ] 正在修改

文件标识:	RK-YH-YF-723
当前版本:	V1.0.0
作者:	吴良清
完成日期:	2022-09-07
审核:	金华君
审核日期:	2022-09-07

免责声明

本文档按"现状"提供，瑞芯微电子股份有限公司（"本公司"，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。
由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

"Rockchip"、"瑞芯微"、"瑞芯"均为本公司的注册商标，归本公司所有。
本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

版权所有 © 2022 瑞芯微电子股份有限公司

瑞芯微电子股份有限公司
Rockchip Electronics Co., Ltd.
地址: 福建省福州市铜盘路软件园A区18号
网址: www.rock-chips.com
客户服务电话: +86-4007-700-590
客户服务传真: +86-591-83951833
客户服务邮箱: fae@rock-chips.com

前言

概述

本文适用RK3399 RK3288等efuse的securityboot和AVB的操作说明。

产品版本

芯片名称	内核版本
RK3399	kernel4.4/kernel-4.19

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

修订记录

版本号	作者	修改日期	修改说明	备注
V1.0.0	吴良清	2022-09-07	初始版本	

文档问题反馈：wlq@rock-chips.com

代码环境

本文适用于RK3399/RK3288，其中security boot适用于Android7及之后的版本；AVB适用于Android11及之后版本。

安全启动SecurityBoot操作步骤

RK3399/RK3288使用efuse，做SecurityBoot时不需要修改源码，只需要对固件进行签名即可，具体操作步骤如下:

1. 工具：rk_sign_tool

SDK中的工具路径

- windows

```
RKTools\windows\rk_sign_tool_v1.42_win.tar.gz
```

- linux

```
RKTools/linux/Linux_SecureBoot/rk_sign_tool_v1.42_linux.tar.gz
```

解压后得到rk_sign_tool

2. 确认工具配置

- 单独securityBoot的方案(Android9.0之前的版本，不支持AVB)，SecurityBoot需要签名boot.img和recovery.img
  确认rk_sign_tool工具里面的setting.ing文件中`exclude_boot_sign =true`**不需要**配置，这个配置表示在签名update.img时**会将boot.img和recovery.img进行签名。**
- SecurityBoot+AVB的方案（Android9.0及之后版本，支持AVB），SecurityBoot不需要签名boot.img和recovery.img
  确认rk_sign_tool工具里面的setting.ing文件中`exclude_boot_sign =true` **需要**配置，这个配置表示在签名update.img时**不对boot.img和recovery.img进行签名。**

3. efuse key生成

```
rk_sign_tool cc --chip 3399  //配置芯片
rk_sign_tool kk --out .      //生成key到当前目录
```

- key每个产品只要生产一次，不要重复生成
- 不同芯片 --chip 后面更新参数不一样，支持
  3588|3566|3568|3308|3326|3399|3229|3228h|3368|3228|3288|px30|3328|1808|3228P|1109|1126|2206

4. 签名固件

- 单独SecurityBoot方案(Android9.0之前版本)
  编译update.img然后执行下面命令进行签名

```
rk_sign_tool sf --firmware update.img
```

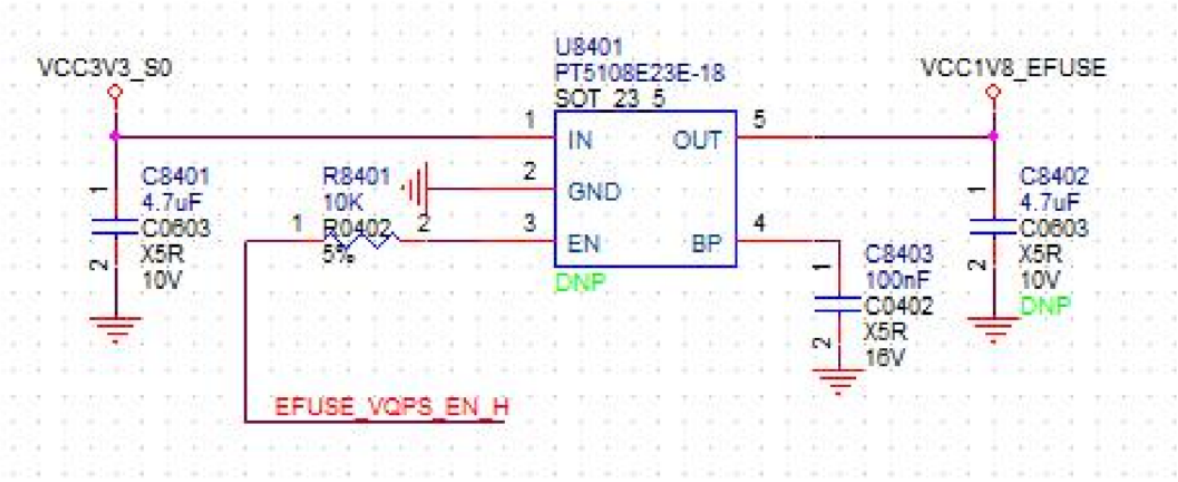
- SecurityBoot+AVB方案(Android9.0及之后版本)
  该方案需要先按下面的`Android Verified Boot(AVB)操作步骤`配置完AVB并编译出来的update.img进行签名

```
rk_sign_tool sf --firmware update.img
```

以上update.img签名后会直接覆盖源文件

5. efuse key烧写(熔断)

RK3399和RK3288烧写efuse时需要外面供电，板子上面没有加入下面的电路时，需要使用夹具进行供电

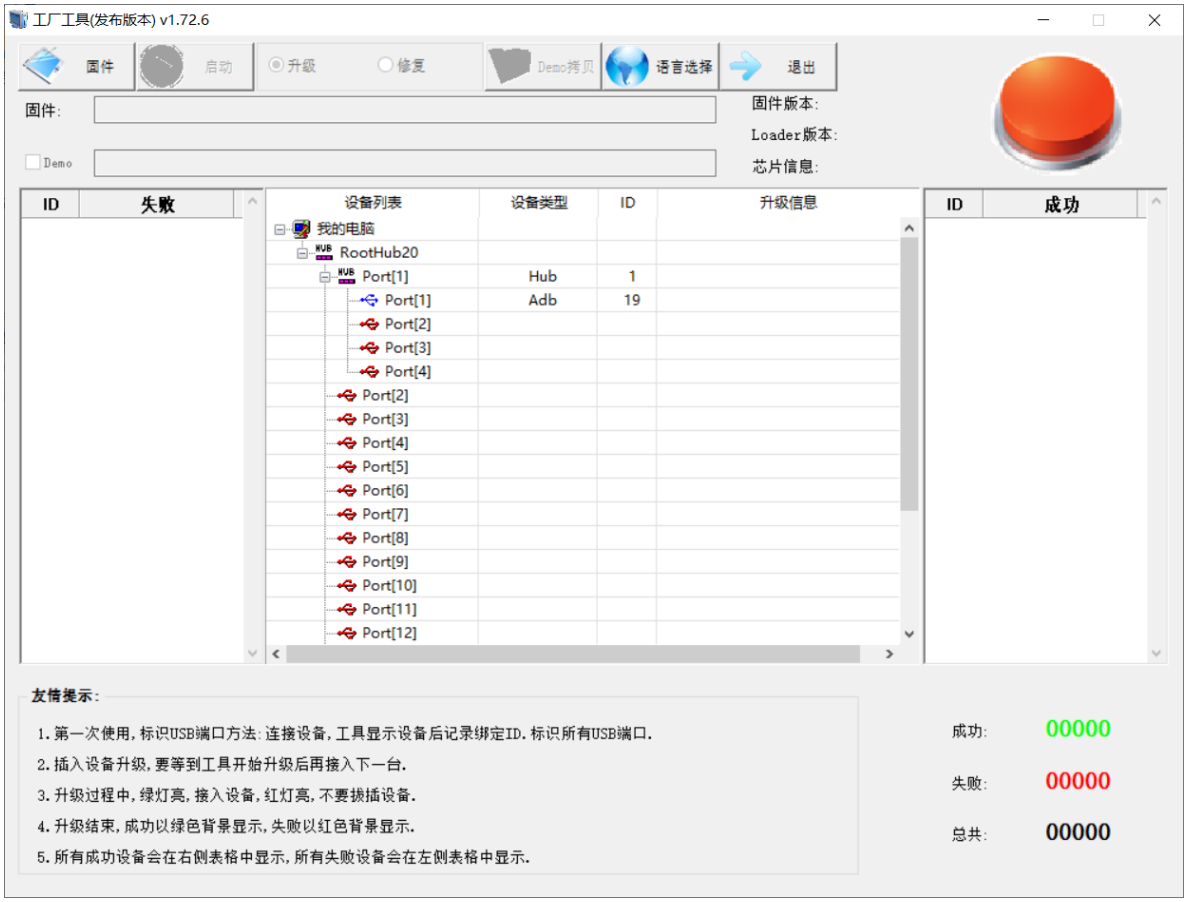


量产阶段使用量产工具烧写步骤4中签名完的update.img，通过修改配置文件可以同时做efuse烧写和固件烧写，配置文件修改如下：

量产工具FactoryTool目录下面config.ini文件中配置

```
#当设置FW_BURN_EFUSE=TRUE时,固件烧写时会去烧写efuse。注意：该选项和FW_CHCEK_EFUSE=TRUE互斥，二者只能选择一个。
FW_BURN_EFUSE=TRUE
```

配置完重新打开量产工具，导入签名完的update.img，然后识别到maskrom设备后点击开始即可,这一步会同时进行固件烧写



6. 判断是否熔断成功

- 通过开机串口log判断

```
## Verified-boot: 0   //固件签名但是芯片没有熔断（1表示有熔断）,没有进行hash有效性验证，即编译的时候没有加 --burn-key-hash
sha256,rsa2048:dev+
rollback index: 1 >= 0(min), OK   //回滚版本，即编译时加--rollback-index-uboot
//下面这些是uboot完整性的校验
## 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
```

- 量产工具中可以检测是否烧写了efuse，需要在config.ini文件中配置

```
#当设置FW_CHCEK_EFUSE=TRUE时,固件烧写时会校验efuse是否烧写。注意：该选项和FW_BURN_EFUSE=TRUE 互斥，二者只能选择一个。
FW_CHCEK_EFUSE=TRUE
```

Android Verified Boot(AVB)操作步骤

1. 编译avbtool工具

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

编译完成后生成：

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

2. 生成atx_permanent_attributes.bin

- **修改产品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 -
```

注意：产品ID的位数为16位，数值可以自己定义

- **生成atx_permanent_attributes.bin**

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

```
../avb_atx_generate_test_data
```

```
cd -
```

执行以上操作后在external/avb/test/data生成：

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

注意：

- pem文件系统中默认有一个，如果需要重新生成，需要删除系统默认的文件，然后再执行上面的操作重新生成pem文件，建议客户自己重新生成
- 这个步骤一个产品只要执行一次就可以，请妥善保管上面生产的文件，在下面的步骤中会使用

3. 代码修改

```
cd device/rockchip/rk3399
```

```
diff --git a/rk3399_Android11/BoardConfig.mk b/rk3399_Android11/BoardConfig.mk
index 24b415f..80fa60f 100644
--- a/rk3399_Android11/BoardConfig.mk
+++ b/rk3399_Android11/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/rk3399/rk3399_Android11/recovery.fstab_AB
 endif
+
+BOARD_AVB_ENABLE := true     //打开AVB功能
+BOARD_AVB_ALGORITHM := SHA256_RSA4096  //配置加密算法
+BOARD_AVB_KEY_PATH := external/avb/test/data/testkey_atx_psk.pem  //秘钥存放路径
+BOARD_AVB_METADATA_BIN_PATH := external/avb/test/data/atx_metadata.bin //指定
metadata文件
+#BOARD_AVB_ROLLBACK_INDEX := 5    //配置防版本回滚，默认不开，根据需求开关，需要配合
uboot修改
```

```
cd -
cd u-boot
```

```
diff --git a/configs/rk3399_defconfig b/configs/rk3399_defconfig
index 3017921487..84197eea1e 100644
--- a/configs/rk3399_defconfig
+++ b/configs/rk3399_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
+CONFIG_ROCKCHIP_PRELOADER_PUB_KEY=y
//选配
+CONFIG_ANDROID_AVB_ROLLBACK_INDEX=y //防回滚功能，需要改功能的才配置，需要配合device下面的BOARD_AVB_ROLLBACK_INDEX一起配置
```

```
cd -
```

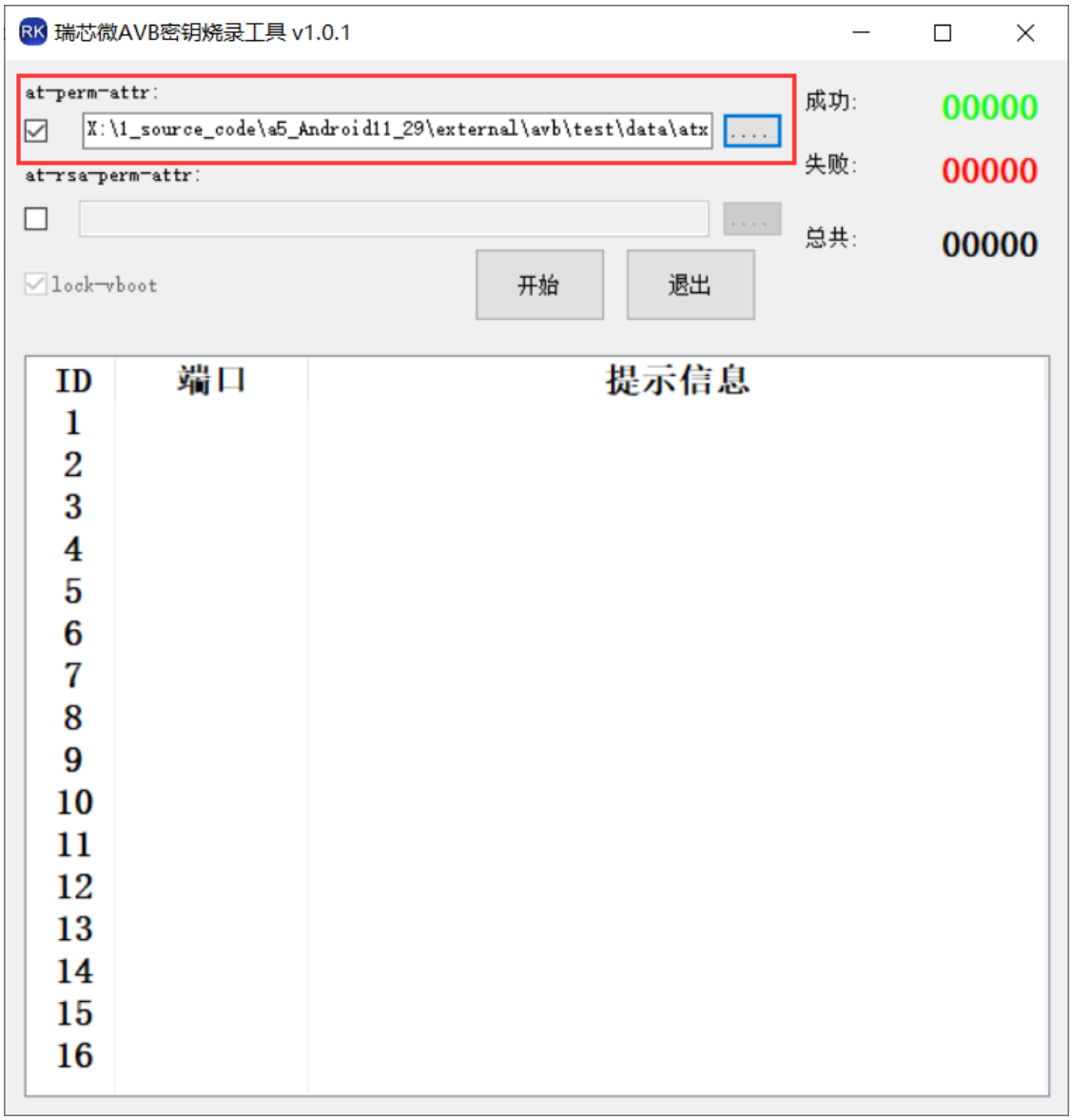
4. AVB key烧写

方式一：AVB key烧写工具烧写

烧写工具：AvbKeyWriter（RKTools/windows/AvbKeyWriter-v1.0.1.7z)
烧写源文件：步骤1中生产的external/avb/test/data生成atx_permanent_attributes.bin
烧写方式：

- 勾选at-perm-attr
- 导入步骤3中生产的external/avb/test/data生成atx_permanent_attributes.bin
- 待烧写设备进入loader模式

- 点击“开机按键进行烧写”



方式二： AVB key集成到uboot代码内，跟固件一起烧写到机器内，不需要再用AVB key工具烧写AVB key

该方式SDK默认没有支持，需要在uboot下打上补丁，

补丁路径：RKDocs/common/security/patch/u-boot/0001-avb-add-embedded-key.patch

```
cd u-boot
```

```
git am RKDocs/common/security/patch/u-boot/0001-avb-add-embedded-key.patch
```

```
cd -
```

抽取公钥

```
avbtool extract_public_key --key external/avb/test/data/testkey_atx_psk.pem --
output avb_root_pub.bin
xxd -i avb_root_pub.bin > external/avb/test/data/avb_root_pub.h
```

替换公钥

抽取的公钥(external/avb/test/data/avb_root_pub.h)替换 u-boot/lib/avb/libavb_user/avb_ops_user.c中的avb_root_pub 数组

```
cd u-boot
```

```
vim lib/avb/libavb_user/avb_ops_user.c
```

```
/**
* Internal builds use testkey_rsa4096.pem
* OEM should replace this Array with public key used to sign vbmeta.img
* *
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 \
* -outform PEM -out avb_rsa4096.pem
* avbtool extract_public_key --key avb_rsa4096.pem --output avb_root_pub.bin
* xxd -i avb_root_pub.bin > avb_root_pub.h
*/
static const char avb_root_pub [] = {
0x00, 0x00, 0x10, 0x00, 0x55, 0xd9, 0x04, 0xad, 0xd8, 0x04, 0xaf, 0xe3,
0xd3, 0x84, 0x6c, 0x7e, 0x0d, 0x89, 0x3d, 0xc2, 0x8c, 0xd3, 0x12, 0x55,
0xe9, 0x62, 0xc9, 0xf1, 0x0f, 0x5e, 0xcc, 0x16, 0x72, 0xab, 0x44, 0x7c,
0x2c, 0x65, 0x4a, 0x94, 0xb5, 0x16, 0x2b, 0x00, 0xbb, 0x06, 0xef, 0x13,
0x07, 0x53, 0x4c, 0xf9, 0x64, 0xb9, 0x28, 0x7a, 0x1b, 0x84, 0x98, 0x88,
0xd8, 0x67, 0xa4, 0x23, 0xf9, 0xa7, 0x4b, 0xdc, 0x4a, 0x0f, 0xf7, 0x3a,
0x18, 0xae, 0x54, 0xa8, 0x15, 0xfe, 0xb0, 0xad, 0xac, 0x35, 0xda, 0x3b,
0xad, 0x27, 0xbc, 0xaf, 0xe8, 0xd3, 0x2f, 0x37, 0x34, 0xd6, 0x51, 0x2b,
… … …
```

```
cd -
```

5. 固件编译

以上步骤执行完后可以进行完整的固件编译，以rk3399_Android11的产品为例进行编译：

```
source build/envsetup.sh
lunch rk3399_Android11-userdebug
./build.sh -AUCKup
```

6. 固件烧写

这里跳回到上面的ScurityBoot的步骤5

7. 启动验证

- **uboot开机log确认**
  通过以上步骤后系统开机时在串口的log中u-boot阶段会有如下打印：

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

- 烧写非AVB固件或者其它的固件会无法开机