# RK356X SecurityBoot和AVB操作指南

	文件标识:	RK-YH-YF-289
	当前版本:	V1.2.0
又件状念: []草稿	作者:	吴良清
[√]正式发布 [1正在修改	完成日期:	2022-04-23
	审核:	
	审核日期:	2022-04-23

版本号	作者	修改日期	修改说明	备注
V1.0.0	吴良清	2021-5-9	初始版本	
V1.1.0	吴良清	2022-2-11	增加AVB防回滚配置说明	
V1.1.0	吴良清	2023-4-21	增加AVB解锁说明	

文档问题反馈: <u>wlq@rock-chips.com</u>

# 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 © 2020 瑞芯微电子股份有限公司

超越合理使用范畴,非经本公司书面许可,任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部,并不得以任何形式传播。

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: <u>www.rock-chips.com</u>

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: <u>fae@rock-chips.com</u>

# 代码环境

安全启动和AVB需要在Android11 RKR7及之后版本才可以支持

# 安全启动SecurityBoot操作步骤

## 编译环境确认

确认编译服务器的fdtput版本是否是1.4.5版本

#fdtput --version
#Version: DTC 1.4.5

如果fdtput版本小于1.4.5,请执行如下命令升级

sudo apt-get install device-tree-compiler

## 1. 进入u-boot目录

cd u-boot

以下操作步骤都在u-boot目录下执行

## 2. 代码修改

进入u-boot目录,打开对应平台的configs/rk3568\_defconfig,选择如下配置:

```
//编辑configs/rk3568_defconfig文件,RK3566和RK3568都是修改这个文件
vim configs/rk3568_defconfig
// 必选。
CONFIG_FIT_SIGNATURE=y
CONFIG_SPL_FIT_SIGNATURE=y
CONFIG_AVB_VBMETA_PUBLIC_KEY_VALIDATE=y
// 可选。
CONFIG_FIT_ROLLBACK_PROTECT=y // boot.img防回滚
```

#### CONFIG\_SPL\_FIT\_ROLLBACK\_PROTECT=y // uboot.img防回滚

## 3. 生成keys

在u-boot目录执行如下操作生成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
```

注意: 该步骤执行一次即可, 然后妥善保存这些keys。

#### 4. 编译签名

以下以RK3566为例,如果是RK3568则将下面的rk3566改为rk3568即可

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

#### 说明:

--spl-new //重新打包签名后的spl --rollback-index-uboot <版本号> //设置版本号,当步骤2中的config配置了防回滚时,需要增加这个 编译选项,否则不需要 --burn-key-hash //加这个编译选项就会在烧写固件后开机的时候进行芯片熔断。

#### 如果编译出现:

Can't load XXXXXX//.rnd into RNG

执行:

touch ~/.rnd

## 5. 编译完整固件

按正常编译固件的方式编译其他固件(uboot和loader上面已经编译完,不需要重新编译),如用

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

### 6. 固件烧写

#### 开发验证阶段可以使用AndroidTool单独烧写uboot和loader即可

🔀 瑞芯微开发工具 v2.84

		Jub L.L.		B2 /7	
		nam	治子	) 商位 ····	10:3辺(心)/7 川4-9月 立ち町1121
	Ľ	0x00000000	Loader	X:\1_source_code\a5_Android11_29\u	統則FlashInfo开始
-		0x0000000	l'arameter		YAAT LANCHLONG Y
3	<u> </u>	0x00004000	Uboot	X:\1_source_code\a5_Android11_29\u	
1	<u> </u>	0x00006000	trust		下载IDB开始
5	느	0x00008000	Misc	X:\1_source_code\a5_Android11_29\DM	下载IDB成功
5	<u> </u>	0x00000000	Resource		等待Loader开始
7		0x00000000	Kernel		等待Loader成功
3		0x0000C800	Boot		测试设备开始
•		0x00020800	Recovery	X:\1_source_code\a5_Android11_29\ro	测试设备成功
0		0x00000000	System		开始下载uboot
1		0x00050800	Backup		正在下载 uboot(100%)
2				2	下販売地
oad	er V	er:1.01	执行	切换 设备分区表 清空	

量产阶段使用量产工具烧写步骤5中编译生成的update.img

Damo       达片信息:         ID       失敗       设备列表       设备类型       ID       升级信息       ID       成功         ●●● 我的电脑       ●●● 我的电脑       ●●● # 我的电脑       ●●● # 我的电脑       ●●● # # # RootHub20       ●● # # RootHub20       ●● # # RootHub20       ●● # # # RootHub20       ●● # RootHub20       <	ŧ:				Loader版本:		
D     失敗     送留列表     送留列表     送留关型     ID     升级信息     ID     成功	lemo				芯片信息:		
● 野田町       ●         ● 整 Port[1]       Hub       1         ● ● Port[1]       Adb       19         ● ● Port[2]       ●         ● ● Port[3]       ●         ● ● Port[4]       ●         ● ● Port[3]       ●         ● ● Port[4]       ●         ● ● Port[5]       ●         ● ● Port[6]       ●         ● ● Port[7]       ●         ● ● Port[9]       ●         ● ● Port[10]       ●         ● ● Port[11]       ●	D 失败	~ 设备列表	设备类型	ID	升级信息	ID	成功
Image: BootHub20       Hub       1         Image: BootHub20       Hub       1         Image: BootHub20       Hub       19         Image: BootHub20       Image: BootHub20       Image: BootHub20         Image: BootHub20		□ 🛃 我的电脑				^	
Port[1]       Hub       1         Image: Port[1]       Adb       19         Image: Port[2]       Image: Port[2]         Image: Port[3]       Image: Port[3]         Image: Port[3]       Image: Port[3]         Image: Port[4]       Image: Port[4]         Image: Port[3]       Image: Port[4]         Image: Port[4]       Image: Port[4]         Image: Port[4]       Image: Port[4]         Image: Port[5]       Image: Port[6]         Image: Port[6]       Image: Port[6]         Image: Port[8]       Image: Port[6]         Image: Port[1]       Image: Port[1]         Image: Port[1]       Image: Port[1]		e-₩ RootHub20				_	
		Port[1]	Hub	1		- 1	
••• Port[2]         ••• Port[4]         ••• Port[3]         ••• Port[3]         ••• Port[4]         ••• Port[5]         ••• Port[6]         ••• Port[8]         ••• Port[9]         ••• Port[1]         ••• Port[12]		Port[1]	Adb	19		- 11	
Image: Port[3]         Image: Port[2]         Image: Port[3]         Image: Port[3]         Image: Port[4]         Image: Port[5]         Image: Port[6]         Image: Port[8]         Image: Port[9]         Image: Port[10]         Image: Port[11]         Image: Port[12]		Port[2]				- 1	
		Port[3]				- 1	
••• Port[2]         ••• Port[3]         ••• Port[4]         ••• Port[5]         ••• Port[6]         ••• Port[7]         ••• Port[8]         ••• Port[9]         ••• Port[10]         ••• Port[11]         ••• Port[12]		Port[4]				- 1	
••• Port[4]         ••• Port[5]         ••• Port[6]         ••• Port[7]         ••• Port[8]         ••• Port[9]         ••• Port[10]         ••• Port[11]         ••• Port[12]		Port[3]				-	
		Port[4]					
		Port[5]				-	
						-	
← Port[8]     ← Port[9]     ← Port[10]     ← Port[11]     ← Port[12]		+++> Port[7]					
← Port[9]     ← Port[10]     ← Port[11]     ← Port[12]							
← Port[10]     ← Port[11]     ← Port[12]							
← Port[11]     ← Port[12]							
		Port[12]				~	
		v (				>	
						-10-1	00000
n	第一次使用,标识USB端口;	方法: 连接设备, 工具显示设备后i	记录绑定ID.标识所有	USB端口.		成功:	00000
第一次使用,标识USB端口方法:连接设备,工具显示设备后记录绑定ID.标识所有USB端口. 成功: 00000	插入设备升级,要等到工具	开始升级后再接入下一台.					00000
第一次使用,标识USB端口方法:连接设备,工具显示设备后记录绑定ID.标识所有USB端口. 成功: 000000 成功: 000000 成功: 000000 成功: 000000 成功: 000000 00000 0000000000	升级过程由 绿灯车 接入计	设备,红灯亮,不要拔插设备.				失败:	00000
第一次使用,标识USB端口方法: 连接设备, 工具显示设备后记录绑定ID.标识所有USB端口. 成功: 000000 插入设备升级, 要等到工具开始升级后再接入下一台. 升级过程中, 绿灯亮, 接入设备, 红灯亮, 不要拨插设备. 失败: 000000	71 2X 12 12 T 1 24 AL 261 10 7 1	and the second sec					

## 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 atf-3 0xfdcd0000 ... sha256+ OK
```

烧写未签名的loader和uboot来判断

# 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
```

#### 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/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
                           //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/rk3568_defconfig b/configs/rk3568_defconfig
index 3017921487..84197eeale 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_RK_AVB_LIBAVB_ENABLE_ATH_UNLOCK=y
+CONFIG_AVB_VBMETA_PUBLIC_KEY_VALIDATE=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=z
```

```
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"			
RK 瑞芯微	☆AVB密钥烧录工具 v	1.0.1	_	
at-perm-s	attr:	成功:	00000	
☑ X:	\1_source_code\a5_#	ndroid11_29\external\avb\test\data	a\atx	
at-rsa-p	erm-attr:	关规:	00000	
		总共:	00000	
∠lock-v	boot	开始	退出	
ID	端口	提示	示信息	
1				
2				
3				
4				
5				
0				
ן 8				
Q				
10				
11				
12				
13				
14				
15				
16				

### 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

这种方式是将key内置在uboot.img的固件里面,没有烧到安全分区中,当boot.img被替换后key就会失效,所以需要配合security boot来保证uboot分区的固件不被篡改或者替换。 注意:这种方式不支持严格AVB校验的解锁功能,所以需要解锁请使用方式一。

#### 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 uboot/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,
Oxad, 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RK3566\_r的产品为例进行编译:

#### - 带安全启动的方案

带安全启动的方案uboot要单独先编译,参考上面安全启动的操作步骤

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

#### • 不带安全启动的方案

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

#### 开发阶段用AndroidTool工具烧写编译出来的完整固件

🕻 瑞芯(	故开发	亡人 v2.84			- 🗆 X
下载镜	懅	升级固件 高级	吸功能	测试设备开始 	
#		地址	名字	按验芯片开始 按验芯片成功 按和Flash.fsf.df	ħía
2 3 4 5 6 7 8 9 10 11		0x0000000 0x00004000 0x00006000 0x00008000 0x00000000 0x00000000 0x000020800 0x000020800 0x000050800	Parameter Uboot trust Misc Resource Kernel Boot Recovery System Backup	X:\1_source_code\a5_Android11_29\u X:\1_source_code\a5_Android11_29\u X:\1_source_code\a5_Android11_29\UIM X:\1_source_code\a5_Android11_29\UIM X:\1_source_code\a5_Android11_29\ro X:\1_source_code\a5_Android11_29\ro TraTst uboct TraTst uboct	。 . (100%)
< Load	ler V	er:1.01	<sup>执行</sup> 发现	切換     设备分区表     清空       一个ADB设备	

量产阶段使用量产工具烧写步骤4中编译生成的update.img

STLT 📲	【发布版本	) v1.72.6								-	$\times$
<	面件	自动	③升編	及 〇修复	Demo拷贝	- 	择 🤿 退出				
固件:							固件版本:				
							Loader版本:				7
Demo							芯片信息:			-	
ID	4	<del>、</del> 敗		设备列表 我的由脑	设备类型	ID	升级信息			成功	^
				WB RootHub20							
				Port[1]	Hub	1					
					Adb	19					
				- Port[3]							
				Arrow Port[4]							
				Port[2]							
				Port[3]							
				Port[4]							
				Port[6]				_			
								_			
								_			
				Port[9]				_			
								_			
								× 1			
1			~ <					>			~
- 友情提才	<b>N</b> :										
								_		00000	
1.第一	次使用, 标	识USB端口开	方法: 连接设	:备,工具显示设备后记	录绑定ID.标识所有	USB端口.		h	4功:	00000	
2.插入	设备升级,	要等到工具	开始升级后	再接入下一台.						00000	
3.升级	过程中,绿	灯亮,接入说	设备,红灯亮	,不要拔插设备.				失	: 败:	00000	
4.升级	结束,成功	以绿色背景	显示,失败以	以红色背景显示.				~	5 <del>++</del> .	00000	
5 所右	成市设备。	全在右側まだ	2. 山泉云 所	有生阶设备全在左侧主	格由昆云			5	(共:	00000	
0. //11	网络贝皮菌	5114003761	а <del>т</del> <u>че</u> лъ М	日天观以田本住在関本	216 T YEAN.						

## 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

## 8. AVB严格校验的解锁步骤

• 设备进入 fastboot 模式, 电脑端输入

```
adb reboot bootloader
fastboot oem at-get-vboot-unlock-challenge
fastboot get_staged raw_unlock_challenge.bin
```

```
-pc:~/GMS-Suite/tools$ adb reboot bootloader
-pc:~/GMS-Suite/tools$
-pc:~/GMS-Suite/tools$
-pc:~/GMS-Suite/tools$ fastboot devices
         Android Fastboot
J7U
-pc:~/GMS-Suite/tools$
pc:~/GMS-Suite/tools$
pc:~/GMS-Suite/tools$ fastboot oem at-get-vboot-unlock-challenge
0.015s]
d. Total time: 0.016s
-pc:~/GMS-Suite/tools$
pc:~/GMS-Suite/tools$
pc:~/GMS-Suite/tools$ fastboot get_staged raw_unlock_challenge.bin
ng 'raw_unlock_challenge.bin'
                                                OKAY [ 0.002s]
. Total time: 0.002s
pc:~/GMS-Suite/tools$
pc:~/GMS-Suite/tools$
pc:~/GMS-Suite/tools$
pc:~/GMS-Suite/tools$
pc:~/GMS-Suite/tools$
pc:~/GMS-Suite/tools$ ls -l
160
--- 1 hcq hcq 2876596 4月
--- 1 hcq hcq 52 4月
--- 1 hcq hcq 19692 2月
--- 1 hcq hcq 3703111 2月
                            14 19:57 OverlayTools.apk
                             27 14:21 raw_unlock_challenge.bin
                             17 09:27 rockchip_selinux_tools
                             16 20:41 SSR.apk
                   580 2月
-x 1 hcg hcg
                             16 20:41 ssr.txt
-pc:~/GMS-Suite/tools$
```

获得带版本、Product Id 与 16 字节的随机数的数据,取出随机数生成 raw\_unlock\_challenge.bin

• 使用 avbtool 生成 unlock\_credential.bin

根据raw\_unlock\_challenge.bin和 atx\_product\_id.bin,其中atx\_product\_id.bin是前面生成 atx\_permanent\_attributes.bin时生成的

python avb-challenge-verify.py raw\_unlock\_challenge.bin atx\_product\_id.bin

```
python avbtool make_unlock_credential --output=unlock_credential.bin --
intermediate_key_certificate=pik_certificate.bin --
unlock_key_certificate=puk_certificate.bin --challenge=unlock_challenge.bin --
unlock_key=testkey_puk.pem
```

其中avb-challenge-verify.py的实现代码如下,请复制代码另存为avb-challenge-verify.py

```
#/user/bin/env python
"This is a test module for getting unlock_challenge.bin"
import sys
import os
```

```
from hashlib import sha256
def challenge_verify():
    if (len(sys.argv) != 3) :
        print "Usage: rkpublickey.py [challenge_file] [product_id_file]"
        return
    if ((sys.argv[1] == "-h") or (sys.argv[1] == "--h")):
        print "Usage: rkpublickey.py [challenge_file] [product_id_file]"
        return
    try:
        challenge_file = open(sys.argv[1], 'rb')
        product_id_file = open(sys.argv[2], 'rb')
        challenge_random_file = open('unlock_challenge.bin', 'wb')
        challenge_data = challenge_file.read(52)
        product_id_data = product_id_file.read(16)
        product_id_hash = sha256(product_id_data).digest()
        print("The challege version is %d" %ord(challenge_data[0]))
        if (product_id_hash != challenge_data[4:36]) :
            print("Product id verify error!")
            return
        challenge_random_file.write(challenge_data[36:52])
        print("Success!")
    finally:
        if challenge_file:
            challenge_file.close()
        if product_id_file:
            product_id_file.close()
        if challenge_random_file:
            challenge_random_file.close()
if ___name___ == '___main___':
    challenge_verify()
```

• 烧写unlock\_credential.bin, 电脑端输入

```
fastboot stage unlock_credential.bin
fastboot oem at-unlock-vboot
```

注意:此时设备状态一直处于第一次进入 fastboot 模式状态,在此期间不能断电、关机、重启。因为步骤 1.做完后,设备存储着生成的随机数,如果断电、关机、重启,会导致随机数丢失,后续校验 challenge signature 会因为随机数不匹配失败。

#### 如果开启:

```
CONFIG_MISC=y
CONFIG_ROCKCHIP_EFUSE=y
CONFIG_ROCKCHIP_OTP=y
```

就会使用 CPUID 作为 challenge number,而 CPUID 是与机器匹配的,数据不会因为关机而丢失,生成的unlock\_credential.bin 可以重复使用。省去重复生成unlock\_challenge.bin,制作unlock\_credential.bin的步骤。再次解锁步骤变为:

fastboot oem at-get-vboot-unlock-challenge
fastboot stage unlock\_credential.bin
fastboot oem at-unlock-vboot