

Rockchip Secure Boot

发布版本：1.0

作者邮箱：jason.zhu@rock-chips.com

日期：2019.01

文件密级：公开资料

前言

概述

本文档将详细介绍基于Rockchip U-boot next-dev的secure boot流程。

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

产品版本

修订记录

日期	版本	作者	修改说明
2019-01-14	V1.0	Jason Zhu	初始版本

Rockchip Secure Boot

1 引用参考

2 术语

3 简介

4 通信加密例子

5 AVB

5.1 AVB支持特性

5.2 key，固件签名及证书生成

5.3 AVB lock

5.4 AVB unlock

5.5 U-boot使能

5.6 kernel修改

5.7 Android SDK一些配置说明

AVB Enable

A/B system

5.8 CMDLINE新内容

6 分区参考

7 fastboot命令支持

- 7.1 fastboot支持命令速览
- 7.2 fastboot具体使用
- 8 固件烧写 (windows)
- 9 pre loader verified
- 10 U-boot verified
- 11 系统校验启动
- 12 基于linux环境的AVB操作及验证流程
 - 12.1 操作流程
 - 12.2 验证流程

1 引用参考

《Rockchip-Secure-Boot-Application-Note.md》

《Android Verified Boot 2.0》

2 术语

avb : Android Verified Boot

OTP&efuse : One Time Programmable

Product RootKey (PRK) : avb的root key由签名loader , uboot & trust的root key校验

ProductIntermediate Key (PIK) : 中间key , 中介作用

ProductSigning Key (PSK) : 用于签固件的key

ProductUnlock Key (PUK) : 用于解锁设备

3 简介

本文介绍Rockchip安全验证引导流程。所谓的安全验证引导流程分为安全性校验与完整性校验。安全性校验是加密公钥的校验，流程为从安全存储 (OTP&efuse) 中读取公钥hash，与计算的公钥hash对比，是否一致，然后解密固件hash。完整性校验为校验固件的完整性，流程为从存储里加载固件，计算固件的hash是否与解密出来的hash一致。

4 通信加密例子

设备的安全验证启动流程与通信中的数据加密校验流程类似，通过该例子可以加速对avb校验流程的理解。假如现在 Alice 向 Bob 传送数字信息，为了保证信息传送的保密性、真实性、完整性和不可否认性，需要对传送的信息进行数字加密和签名，其传送过程为：

- 1.Alice 准备好要传送的数字信息（明文）；
- 2.Alice 对数字信息进行哈希运算，得到一个信息摘要；
- 3.Alice 用自己的私钥对信息摘要进行加密得到 Alice 的数字签名，并将其附在数字信息上；
- 4.Alice 随机产生一个加密密钥，并用此密码对要发送的信息进行加密，形成密文；
- 5.Alice 用 Bob 的公钥对刚才随机产生的加密密钥进行加密，将加密后的 DES 密钥连同密文一起传送给Bob；

6. Bob 收到 Alice 传送来的密文和加密过的 DES 密钥，先用自己的私钥对加密的 DES 密钥进行解密，得到 Alice 随机产生的加密密钥；
 7. Bob 然后用随机密钥对收到的密文进行解密，得到明文的数字信息，然后将随机密钥抛弃；
 8. Bob 用 Alice 的公钥对 Alice 的数字签名进行解密，得到信息摘要；
 9. Bob 用相同的哈希算法对收到的明文再进行一次哈希运算，得到一个新的信息摘要；
 10. Bob 将收到的信息摘要和新产生的信息摘要进行比较，如果一致，说明收到的信息没有被修改过。
- 上面提及的DES算法可以更换其他算法，如RSA加密算法， 流程如下：



5 AVB

AVB为Android Verified Boot，谷歌设计的一套固件校验流程，主要用于校验boot system等固件。

5.1 AVB支持特性

1. 安全校验
2. 完整性校验
3. 防回滚保护
4. persistent partition支持
5. Chained partitions支持，可以与boot，system签名私钥一致，也可以由oem自己保存私钥，但必须由PRK签名

5.2 key，固件签名及证书生成

```
#!/bin/sh
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 -outform PEM -out
testkey_atx_prk.pem
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 -outform PEM -out
testkey_atx_psk.pem
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 -outform PEM -out
testkey_atx_pik.pem
python avbtool make_atx_certificate --output=atx_pik_certificate.bin --subject=temp.bin --
subject_key=testkey_atx_pik.pem --subject_is_intermediate_authority --subject_key_version
42 --authority_key=testkey_atx_prk.pem
python avbtool make_atx_certificate --output=atx_psk_certificate.bin --
subject=atx_product_id.bin --subject_key=testkey_atx_psk.pem --subject_key_version 42 --
authority_key=testkey_atx_pik.pem
python avbtool make_atx_metadata --output=atx_metadata.bin --
intermediate_key_certificate=atx_pik_certificate.bin --
product_key_certificate=atx_psk_certificate.bin
```

其中temp.bin需要自己创建的临时文件，新建temp.bin即可，无需填写数据。

atx_permanent_attributes.bin生成：

```
python avbtool make_atx_permanent_attributes --output=atx_permanent_attributes.bin --
product_id=atx_product_id.bin --root_authority_key=testkey_atx_prk.pem
```

其中atx_product_id.bin需要自己定义，占16字节，可作为产品ID定义。

boot.img签名示例：

```
avbtool add_hash_footer --image boot.img --partition_size 33554432 --partition_name boot --
key testkey_atx_psk.pem --algorithm SHA256_RSA4096
```

注意：partition size 要至少比原固件大64K，大小还要4K对齐，且小于parameter定义的大小。

system.img签名：

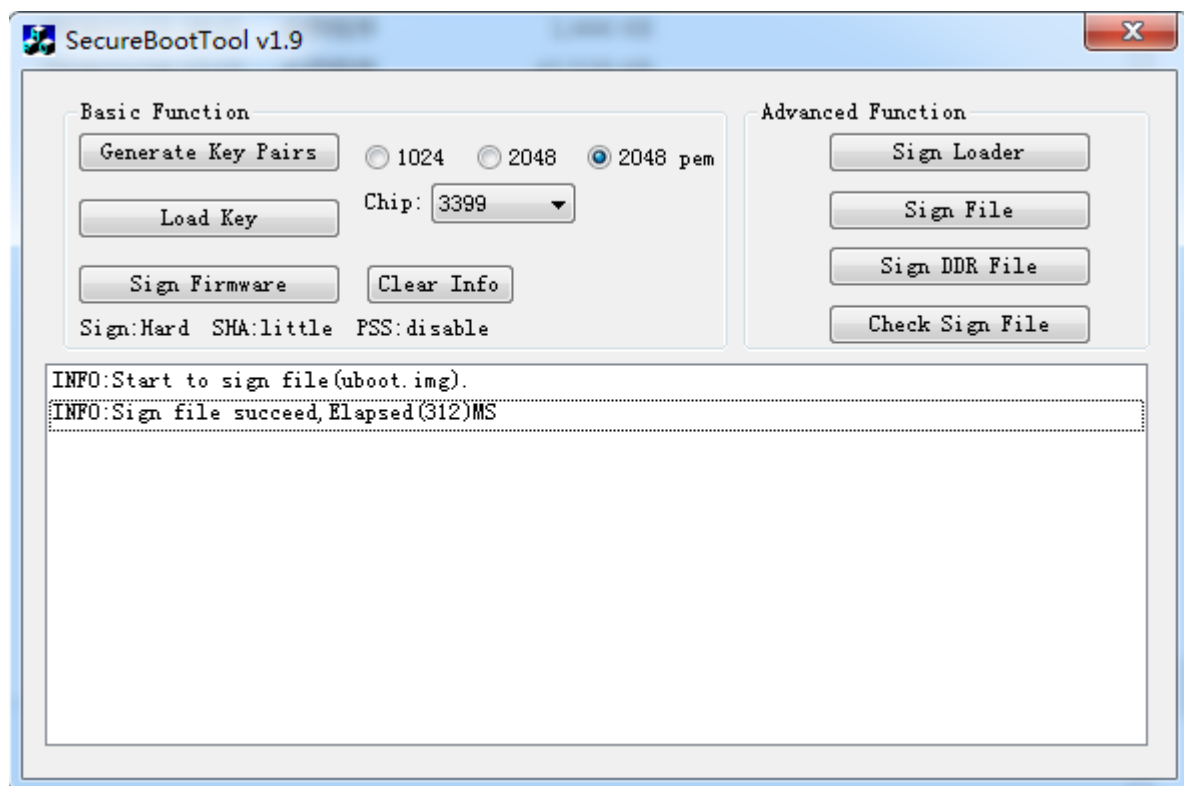
```
avbtool add_hashtree_footer --partition_size 536870912 --partition_name system --image
system.img --algorithm SHA256_RSA4096 --key testkey_atx_psk.pem
```

生成vbmeta包含atx_metadata.bin，命令示例如下：

```
python avbtool make_vbmeta_image --public_key_metadata atx_metadata.bin --
include_descriptors_from_image boot.img --include_descriptors_from_image system.img --
generate_dm_verity_cmdline_from_hashtree system.img --algorithm SHA256_RSA4096 --key
testkey_atx_psk.pem --output vbmeta.img
```

最终烧写生成的vbmeta烧写到对应的分区，如vbmeta分区。

通过SecureBootTool生成PrivateKey.pem和PublicKey.pem。



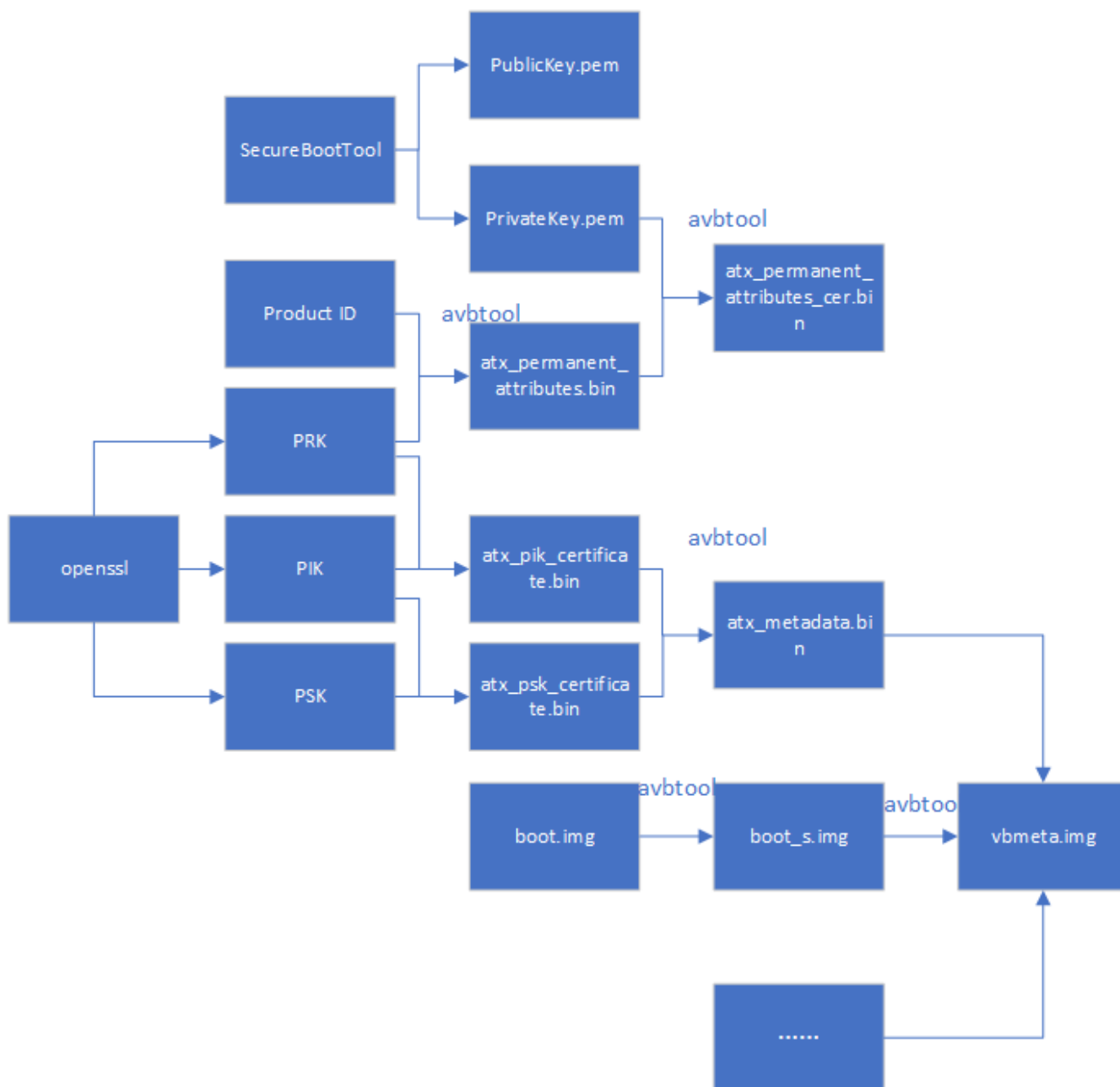
对atx_permanent_attributes.bin进行签名：

```
openssl dgst -sha256 -out atx_permanent_attributes_cer.bin -sign PrivateKey.pem  
atx_permanent_attributes.bin
```

pub_key烧写：

```
fastboot stage atx_permanent_attributes.bin  
fastboot oem fuse at-perm-attr  
fastboot stage atx_permanent_attributes_cer.bin  
fastboot oem fuse at-rsa-perm-attr  
fastboot reboot
```

整个签名流程：



5.3 AVB lock

```
fastboot oem at-lock-vboot
```

如何进入fastboot见fastboot命令支持章节。

5.4 AVB unlock

目前 Rockchip 采用严格安全校验，U-Boot下需要在相应的 `include/configs/rkxxxx_common.h` 开启 `CONFIG_RK_AVB_LIBAVB_ENABLE_ATH_UNLOCK`。否则输入 `fastboot oem at-unlock-vboot` 就可以解锁设备，启动校验vbmeta.img，boot.img失败也会成功启动。

首先，需要生成PUK：

```
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 -outform PEM -out  
testkey_atx_puk.pem
```

atx_unlock_credential.bin为需要下载到设备解锁的证书，其生成过程如下：

```
python avbtool make_atx_certificate --output=atx_puk_certificate.bin --  
subject=atx_product_id.bin --subject_key=testkey_atx_puk.pem --  
usage=com.google.android.things.vboot.unlock --subject_key_version 42 --  
authority_key=testkey_atx_pik.pem
```

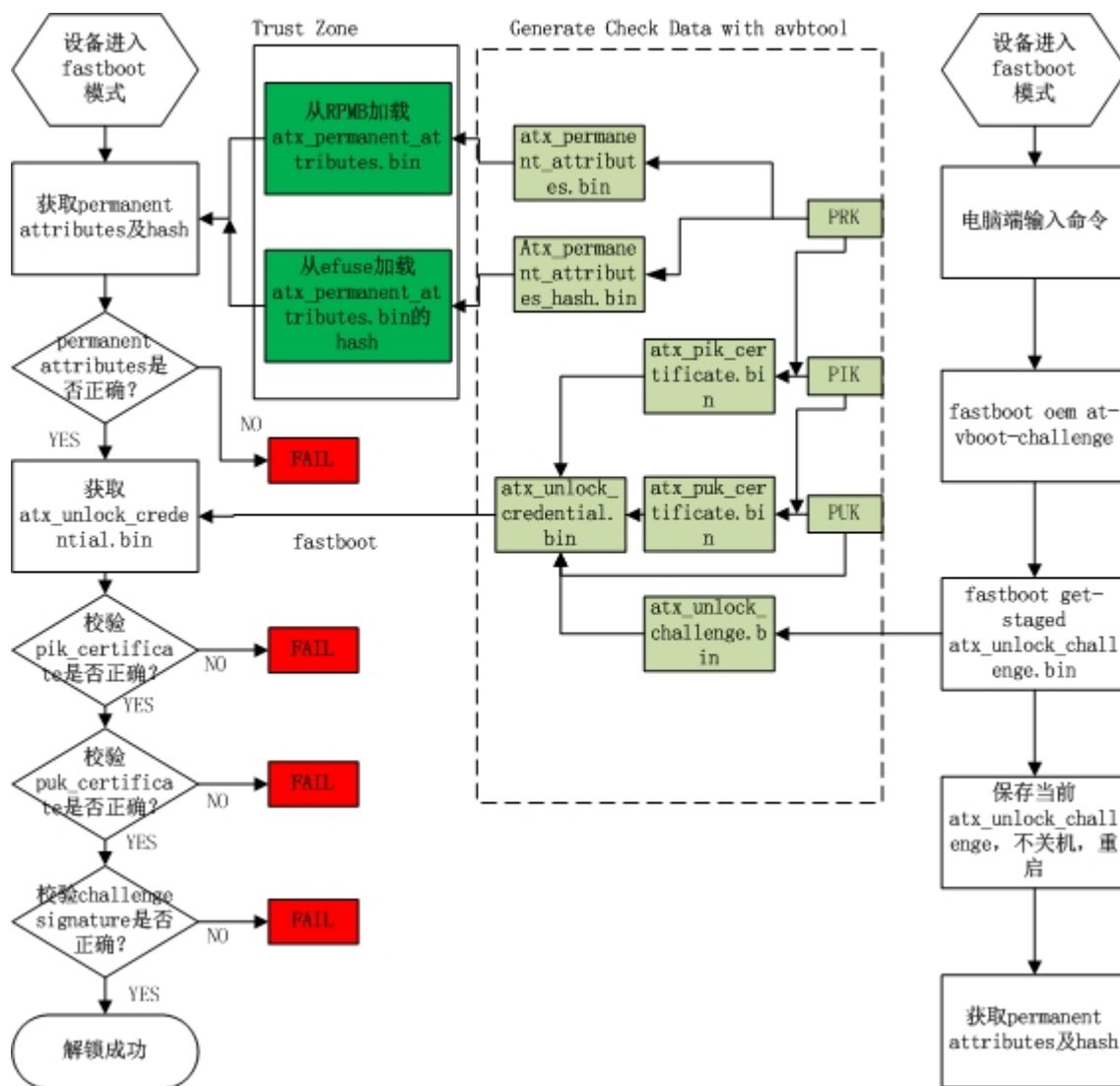
从设备获取atx_unlock_credential.bin，使用avb-challenge-verify.py脚本获取atx_unlock_credential.bin，执行下列命令获取atx_unlock_credential.bin：

```
python avbtool make_atx_unlock_credential --output=atx_unlock_credential.bin --  
intermediate_key_certificate=atx_pik_certificate.bin --  
unlock_key_certificate=atx_puk_certificate.bin --challenge=atx_unlock_challenge.bin --  
unlock_key=testkey_atx_puk.pem
```

最终可以把证书通过fastboot命令下载到设备，并解锁设备，fastboot命令如下：

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

最后设备解锁流程：



最后操作流程如下：

1. 设备进入fastboot模式，电脑端输入

```
fastboot oem at-get-vboot-unlock-challenge
fastboot get-staged raw_atx_unlock_challenge.bin
```

获得带版本、Product Id与16字节的随机数的数据，取出随机数作为atx_unlock_challenge.bin。

2. 使用avbtool生成atx_unlock_credential.bin
3. 电脑端输入

```
fastboot stage atx_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。

4. 设备进入解锁状态，开始解锁。

make_unlock.sh参考


```
#!/bin/sh
python avb-challenge-verify.py raw_atx_unlock_challenge.bin atx_product_id.bin
python avbtool make_atx_unlock_credential --output=atx_unlock_credential.bin --
intermediate_key_certificate=atx_pik_certificate.bin --
unlock_key_certificate=atx_puk_certificate.bin --challenge=atx_unlock_challenge.bin --
unlock_key=testkey_atx_puk.pem
```

avb-challenge-verify.py源码

```
#!/user/bin/env python
"this is a test module for getting unlock challenge"
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('atx_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 challenge 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()
```

5.5 U-boot使能

开启avb需要trust支持，需要U-Boot在defconfig文件中配置：

```
CONFIG_OPTEE_CLIENT=y
CONFIG_OPTEE_V1=y
CONFIG_OPTEE_ALWAYS_USE_SECURITY_PARTITION=y //安全数据存储到security分区
```

CONFIG_OPTEE_V1：适用平台有312x,322x,3288,3228H,3368,3399。 CONFIG_OPTEE_V2：适用平台有3326,3308。 CONFIG_OPTEE_ALWAYS_USE_SECURITY_PARTITION：当emmc的rpmb不能用，才开这个宏，默认不开。

avb开启需要在defconfig文件中配置：

```
CONFIG_AVB_LIBAVB=y
CONFIG_AVB_LIBAVB_AB=y
CONFIG_AVB_LIBAVB_ATX=y
CONFIG_AVB_LIBAVB_USER=y
CONFIG_RK_AVB_LIBAVB_USER=y
上面几个为必选，下面选择为支持AVB与AB特性，两个特性可以分开使用。
CONFIG_ANDROID_AB=y //这个支持a/b
CONFIG_ANDROID_AVB=y //这个支持avb
```

开启安全性校验需要打开宏CONFIG_AVB_VBMETA_PUBLIC_KEY_VALIDATE。

5.6 kernel修改

system，vendor，oem等分区的校验由kernel的dm-verify模块加载校验，所以需要使能该模块。

使能AVB需要在kernel dts上配置参数avb，参考如下：

```
&firmware_android {
    compatible = "android,firmware";
    boot_devices = "fe330000.sdhci";
    vbmeta {
        compatible = "android,vbmeta";
        parts = "vbmeta,boot,system,vendor,dtbo";
    };
    fstab {
        compatible = "android,fstab";
        vendor {
            compatible = "android,vendor";
            dev = "/dev/block/by-name/vendor";
            type = "ext4";
            mnt_flags = "ro,barrier=1,inode_readahead_blks=8";
            fsmgr_flags = "wait,avb";
        };
    };
};
```

使能A/B system需要配置slotsselect参数，参考如下：

```
firmware {
    android {
        compatible = "android,firmware";
```

```

fstab {
    compatible = "android,fstab";
    system {
        compatible = "android,system";
        dev = "/dev/block/by-name/system";
        type = "ext4";
        mnt_flags = "ro,barrier=1,inode_readahead_blks=8";
        fsmgr_flags = "wait,verify,slotselect";
    };
    vendor {
        compatible = "android,vendor";
        dev = "/dev/block/by-name/vendor";
        type = "ext4";
        mnt_flags = "ro,barrier=1,inode_readahead_blks=8";
        fsmgr_flags = "wait,verify,slotselect";
    };
};
};
};

```

5.7 Android SDK一些配置说明

AVB Enable

使能BOARD_AVB_ENABLE

A/B system

这些变量主要有三类：

- A/B 系统必须定义的变量
 - AB_OTA_UPDATER := true
 - AB_OTA_PARTITIONS := boot system vendor
 - BOARD_BUILD_SYSTEM_ROOT_IMAGE := true
 - TARGET_NO_RECOVERY := true
 - BOARD_USES_RECOVERY_AS_BOOT := true
 - PRODUCT_PACKAGES += update_engine update_verifier
- A/B 系统可选定义的变量
 - PRODUCT_PACKAGES_DEBUG += update_engine_client
- A/B 系统不能定义的变量
 - BOARD_RECOVERYIMAGE_PARTITION_SIZE
 - BOARD_CACHEIMAGE_PARTITION_SIZE
 - BOARD_CACHEIMAGE_FILE_SYSTEM_TYPE

5.8 CMDLINE新内容

```
Kernel command line: androidboot.verifiedbootstate=green androidboot.slot_suffix=_a dm="1
vroot none ro 1,0 1031864 verity 1 PARTUUID=b2110000-0000-455a-8000-44780000706f
PARTUUID=b2110000-0000-455a-8000-44780000706f 4096 4096 128983 128983 sha1
90d1d406caac04b7e3fbf48b9a4dcd6992cc628e 4172683f0d6b6085c09f6ce165cf152fe3523c89 10
restart_on_corruption ignore_zero_blocks use_fec_from_device PARTUUID=b2110000-0000-455a-
8000-44780000706f fec_roots 2 fec_blocks 130000 fec_start 130000" root=/dev/dm-0
androidboot.vbmeta.device=PARTUUID=f24f0000-0000-4e1b-8000-791700006a98
androidboot.vbmeta.avb_version=1.1 androidboot.vbmeta.device_state=unlocked
androidboot.vbmeta.hash_alg=sha512 androidboot.vbmeta.size=6528
androidboot.vbmeta.digest=41991c02c82ea1191545c645e2ac9cc7ca08b3da0a2e3115aff479d2df61feacc
dd35b6360cfa936f6f4381e4557ef18e381f4b236000e6ecc9ada401eda4cae
androidboot.vbmeta.invalidate_on_error=yes androidboot.veritymode=enforcing
```

这里说明几个参数：

1. 为什么传递vbmeta的PARTUUID？因为确保后续使用vbmeta hash-tree的合法性，需要kernel再校验一遍vbmeta，digest为androidboot.vbmeta.digest。
2. skip_initramfs：boot ramdisk有无打包到boot.img问题。
3. root=/dev/dm-0开启dm-verity。
4. androidboot.vbmeta.device_state：android verify 状态
5. androidboot.verifiedbootstate：校验结果。

green：If in LOCKED state and an the key used for verification was not set by the end user.

yellow：If in LOCKED state and an the key used for verification was set by the end user.

orange：If in the UNLOCKED state.

这里特别说明一下dm="1 vroot none ro...."参数生成

```
avbtool make_vbmeta_image --include_descriptors_from_image boot.img --
include_descriptors_from_image system.img --generate_dm_verity_cmdline_from_hashtree
system.img --include_descriptors_from_image vendor.img --algorithm SHA512_RSA4096 --key
testkey_atx_psk.pem --public_key_metadata atx_metadata.bin --output vbmeta.img
```

avbtool生成vbmeta时，对system固件加--generate_dm_verity_cmdline_from_hashtree即可。dm="1 vroot none ro...."这些信息会保存到vbmeta。这部分安卓专用，如果分区只校验到boot.img，无需增加该参数。

Android SDK开启BOARD_AVB_ENABLE会把这些信息加到vbmeta内。

6 分区参考

新增加vbmeta分区与security分区，vbmeta分区存储固件校验信息，security分区存储加密过的安全数据。

```
FIRMWARE_VER:8.0
MACHINE_MODEL:RK3326
MACHINE_ID:007
MANUFACTURER: RK3326
MAGIC: 0x5041524B
ATAG: 0x00200800
MACHINE: 3326
CHECK_MASK: 0x80
PWR_HLD: 0,0,A,0,1
TYPE: GPT
CMDLINE:mtddparts=rk29xxnand:0x00002000@0x00004000(uboot),0x00002000@0x00006000(trust),0x00002000@0x00008000(misc),0x00008000@0x0000a000(resource),0x00010000@0x00012000(kernel),0x00002000@0x00022000(dtb),0x00002000@0x00024000(dtbo),0x00000800@0x00026000(vbmeta),0x00010000@0x00026800(boot),0x00020000@0x00036800(recovery),0x00038000@0x00056800(backup),0x00002000@0x0008e800(security),0x000c0000@0x00090800(cache),0x00514000@0x00150800(system),0x00008000@0x00664800(metadata),0x000c0000@0x0066c800(vendor),0x00040000@0x0072c800(oem),0x00004000@0x0076c800(frp),-@0x0076cc00(userdata:grow)
uuid:system=af01642c-9b84-11e8-9b2a-234eb5e198a0
```

A/B System分区定义参考：

```
FIRMWARE_VER:8.1
MACHINE_MODEL:RK3326
MACHINE_ID:007
MANUFACTURER: RK3326
MAGIC: 0x5041524B
ATAG: 0x00200800
MACHINE: 3326
CHECK_MASK: 0x80
PWR_HLD: 0,0,A,0,1
TYPE: GPT
CMDLINE:
mtddparts=rk29xxnand:0x00002000@0x00004000(uboot_a),0x00002000@0x00006000(uboot_b),0x00002000@0x00008000(trust_a),0x00002000@0x0000a000(trust_b),0x00001000@0x0000c000(misc),0x00001000@0x0000d000(vbmeta_a),0x00001000@0x0000e000(vbmeta_b),0x00020000@0x0000e000(boot_a),0x00020000@0x0002e000(boot_b),0x00100000@0x0004e000(system_a),0x00300000@0x0032e000(system_b),0x00100000@0x0062e000(vendor_a),0x00100000@0x0072e000(vendor_b),0x00002000@0x0082e000(oem_a),0x00002000@0x00830000(oem_b),0x00100000@0x00832000(factory),0x00008000@0x842000(factory_bootloader),0x00080000@0x008ca000(oem),-@0x0094a000(userdata)
```

7 fastboot命令支持

U-Boot下可以通过输入命令进入fastboot：

```
fastboot usb 0
```

7.1 fastboot支持命令速览

```
fastboot flash < partition > [ < filename > ]
fastboot erase < partition >
```

```
fastboot getvar < variable > | all
fastboot set_active < slot >
fastboot reboot
fastboot reboot-bootloader
fastboot flashing unlock
fastboot flashing lock
fastboot stage [ < filename > ]
fastboot get_staged [ < filename > ]
fastboot oem fuse at-perm-attr-data
fastboot oem fuse at-perm-attr
fastboot oem fuse at-rsa-perm-attr
fastboot oem at-get-ca-request
fastboot oem at-set-ca-response
fastboot oem at-lock-vboot
fastboot oem at-unlock-vboot
fastboot oem at-disable-unlock-vboot
fastboot oem fuse at-bootloader-vboot-key
fastboot oem format
fastboot oem at-get-vboot-unlock-challenge
fastboot oem at-reset-rollback-index
```

7.2 fastboot具体使用

1. fastboot flash < partition > [< filename >]

功能：分区烧写。

例：fastboot flash boot boot.img

2. fastboot erase < partition >

功能：擦除分区。

举例：fastboot erase boot

3. fastboot getvar < variable > | all

功能：获取设备信息

举例：fastboot getvar all（获取设备所有信息）

variable 还可以带的参数：

version	/* fastboot 版本 */
version-bootloader	/* U-Boot 版本 */
version-baseband	
product	/* 产品信息 */
serialno	/* 序列号 */
secure	/* 是否开启安全校验 */
max-download-size	/* fastboot 支持单次传输最大字节数 */
logical-block-size	/* 逻辑块数 */
erase-block-size	/* 擦除块数 */
partition-type : < partition >	/* 分区类型 */
partition-size : < partition >	/* 分区大小 */
unlocked	/* 设备lock状态 */

```

off-mode-charge
battery-voltage
variant
battery-soc-ok
slot-count                /* slot 数目 */
has-slot: < partition >   /* 查看slot内是否有该分区名 */
current-slot              /* 当前启动的slot */
slot-suffixes             /* 当前设备具有的slot,打印出其name */
slot-successful: < _a | _b > /* 查看分区是否正确校验启动过 */
slot-unbootable: < _a | _b > /* 查看分区是否被设置为unbootable */
slot-retry-count: < _a | _b > /* 查看分区的retry-count次数 */
at-attest-dh
at-attest-uuid
at-vboot-state

```

fastboot getvar 举例：

```

PS E:\U-Boot-AVB\adb> .\fastboot.exe getvar all
(bootloader) version:0.4
(bootloader) version-bootloader:U-Boot 2017.09-gc277677
(bootloader) version-baseband:N/A
(bootloader) product:rk3229
(bootloader) serialno:7b2239270042f8b8
(bootloader) secure:yes
(bootloader) max-download-size:0x04000000
(bootloader) logical-block-size:0x512
(bootloader) erase-block-size:0x80000
(bootloader) partition-type:bootloader_a:U-Boot
(bootloader) partition-type:bootloader_b:U-Boot
(bootloader) partition-type:tos_a:U-Boot
(bootloader) partition-type:tos_b:U-Boot
(bootloader) partition-type:boot_a:U-Boot
(bootloader) partition-type:boot_b:U-Boot
(bootloader) partition-type:system_a:ext4
(bootloader) partition-type:system_b:ext4
(bootloader) partition-type:vbmeta_a:U-Boot
(bootloader) partition-type:vbmeta_b:U-Boot
(bootloader) partition-type:misc:U-Boot
(bootloader) partition-type:vendor_a:ext4
(bootloader) partition-type:vendor_b:ext4
(bootloader) partition-type:oem_bootloader_a:U-Boot
(bootloader) partition-type:oem_bootloader_b:U-Boot
(bootloader) partition-type:factory:U-Boot
(bootloader) partition-type:factory_bootloader:U-Boot
(bootloader) partition-type:oem_a:ext4
(bootloader) partition-type:oem_b:ext4
(bootloader) partition-type:userdata:ext4
(bootloader) partition-size:bootloader_a:0x400000
(bootloader) partition-size:bootloader_b:0x400000
(bootloader) partition-size:tos_a:0x400000
(bootloader) partition-size:tos_b:0x400000
(bootloader) partition-size:boot_a:0x2000000
(bootloader) partition-size:boot_b:0x2000000

```

```
(bootloader) partition-size:system_a:0x20000000
(bootloader) partition-size:system_b:0x20000000
(bootloader) partition-size:vbmeta_a:0x10000
(bootloader) partition-size:vbmeta_b:0x10000
(bootloader) partition-size:misc:0x100000
(bootloader) partition-size:vendor_a:0x4000000
(bootloader) partition-size:vendor_b:0x4000000
(bootloader) partition-size:oem_bootloader_a:0x400000
(bootloader) partition-size:oem_bootloader_b:0x400000
(bootloader) partition-size:factory:0x2000000
(bootloader) partition-size:factory_bootloader:0x1000000
(bootloader) partition-size:oem_a:0x10000000
(bootloader) partition-size:oem_b:0x10000000
(bootloader) partition-size:userdata:0x7ad80000
(bootloader) unlocked:no
(bootloader) off-mode-charge:0
(bootloader) battery-voltage:0mv
(bootloader) variant:rk3229_evb
(bootloader) battery-soc-ok:no
(bootloader) slot-count:2
(bootloader) has-slot:bootloader:yes
(bootloader) has-slot:tos:yes
(bootloader) has-slot:boot:yes
(bootloader) has-slot:system:yes
(bootloader) has-slot:vbmeta:yes
(bootloader) has-slot:misc:no
(bootloader) has-slot:vendor:yes
(bootloader) has-slot:oem_bootloader:yes
(bootloader) has-slot:factory:no
(bootloader) has-slot:factory_bootloader:no
(bootloader) has-slot:oem:yes
(bootloader) has-slot:userdata:no
(bootloader) current-slot:a
(bootloader) slot-suffixes:a,b
(bootloader) slot-successful:a:yes
(bootloader) slot-successful:b:no
(bootloader) slot-unbootable:a:no
(bootloader) slot-unbootable:b:yes
(bootloader) slot-retry-count:a:0
(bootloader) slot-retry-count:b:0
(bootloader) at-attest-dh:1:P256
(bootloader) at-attest-uuid:
all: Done!
finished. total time: 0.636s
```

4. fastboot set_active < slot >

功能：设置重启的slot。

举例：fastboot set_active _a

5. fastboot reboot

功能：重启设备，正常启动

举例：fastboot reboot

6. fastboot reboot-bootloader

功能：重启设备，进入fastboot模式

举例：fastboot reboot-bootloader

7. fastboot flashing unlock

功能：解锁设备，允许烧写固件

举例：fastboot flashing unlock

8. fastboot flashing lock

功能：锁定设备，禁止烧写

举例：fastboot flashing lock

9. fastboot stage [< filename >]

功能：下载数据到设备端内存，内存起始地址为CONFIG_FASTBOOT_BUF_ADDR

举例：fastboot stage atx_permanent_attributes.bin

10. fastboot get_staged [< filename >]

功能：从设备端获取数据

举例：fastboot get_staged raw_atx_unlock_challenge.bin

11. fastboot oem fuse at-perm-attr

功能：烧写ATX及hash

举例：fastboot stage atx_permanent_attributes.bin

fastboot oem fuse at-perm-attr

12. fastboot oem fuse at-perm-attr-data

功能：只烧写ATX到安全存储区域 (RPMB)

举例：fastboot stage atx_permanent_attributes.bin

fastboot oem fuse at-perm-attr-data

13. fastboot oem at-get-ca-request

14. fastboot oem at-set-ca-response

15. fastboot oem at-lock-vboot

功能：锁定设备

举例：fastboot oem at-lock-vboot

16. fastboot oem at-unlock-vboot

功能：解锁设备，现支持authenticated unlock

举例：fastboot oem at-get-vboot-unlock-challenge fastboot get_staged raw_atx_unlock_challenge.bin

./make_unlock.sh (见make_unlock.sh参考) fastboot stage atx_unlock_credential.bin fastboot oem at-unlock-vboot

17. fastboot oem fuse at-bootloader-vboot-key

功能：烧写bootloader key hash

举例：fastboot stage bootloader-pub-key.bin

fastboot oem fuse at-bootloader-vboot-key

18. fastboot oem format

功能：重新格式化分区，分区信息依赖于\$partitions

举例：fastboot oem format

19. fastboot oem at-get-vboot-unlock-challenge

功能：authenticated unlock，需要获得unlock challenge 数据

举例：参见16. fastboot oem at-unlock-vboot

20. fastboot oem at-reset-rollback-index

功能：复位设备的rollback数据

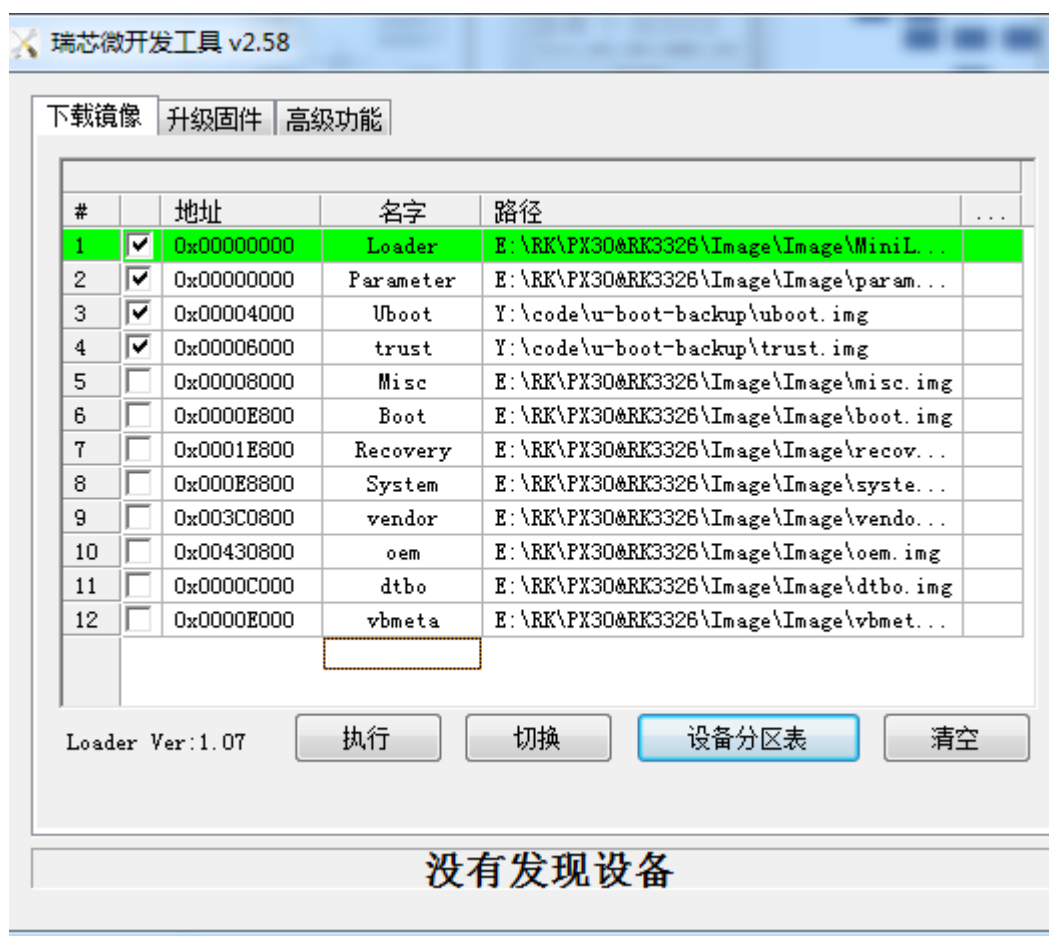
举例：fastboot oem at-reset-rollback-index

21. fastboot oem at-disable-unlock-vboot

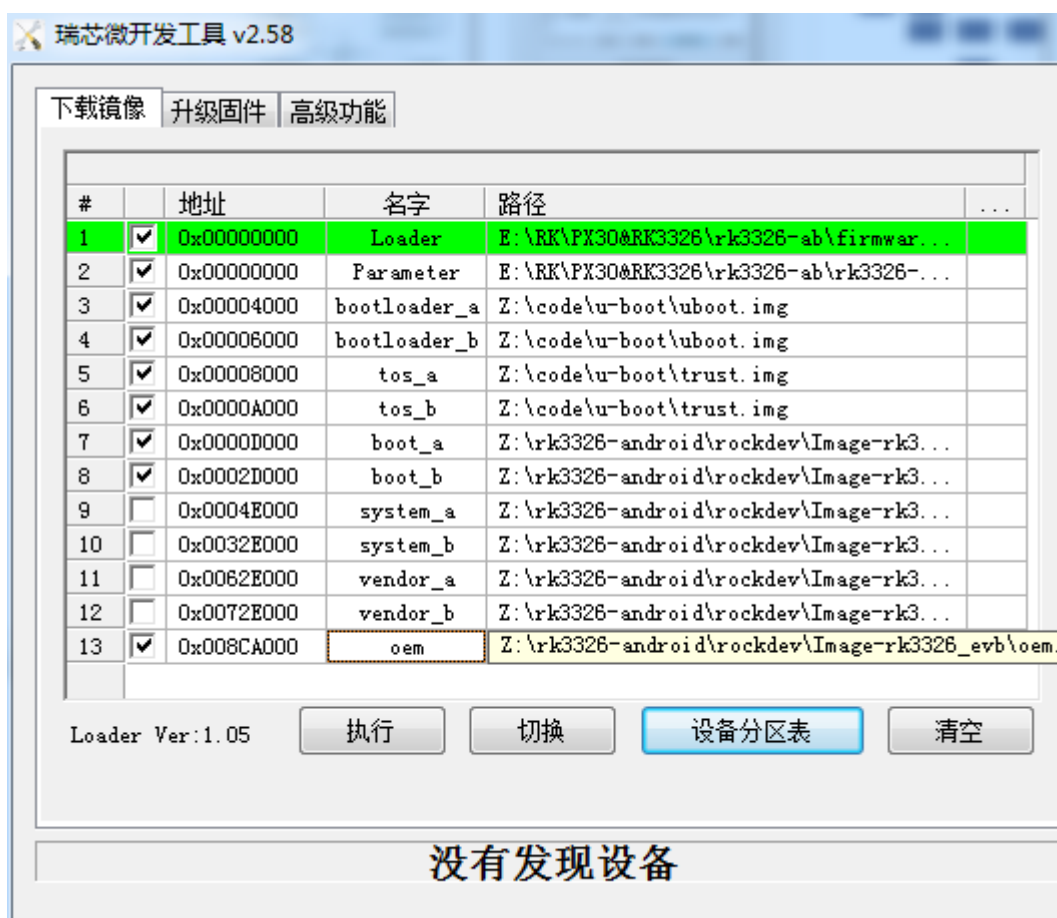
功能：使fastboot oem at-unlock-vboot命令失效

举例：fastboot oem at-disable-unlock-vboot

8 固件烧写 (windows)

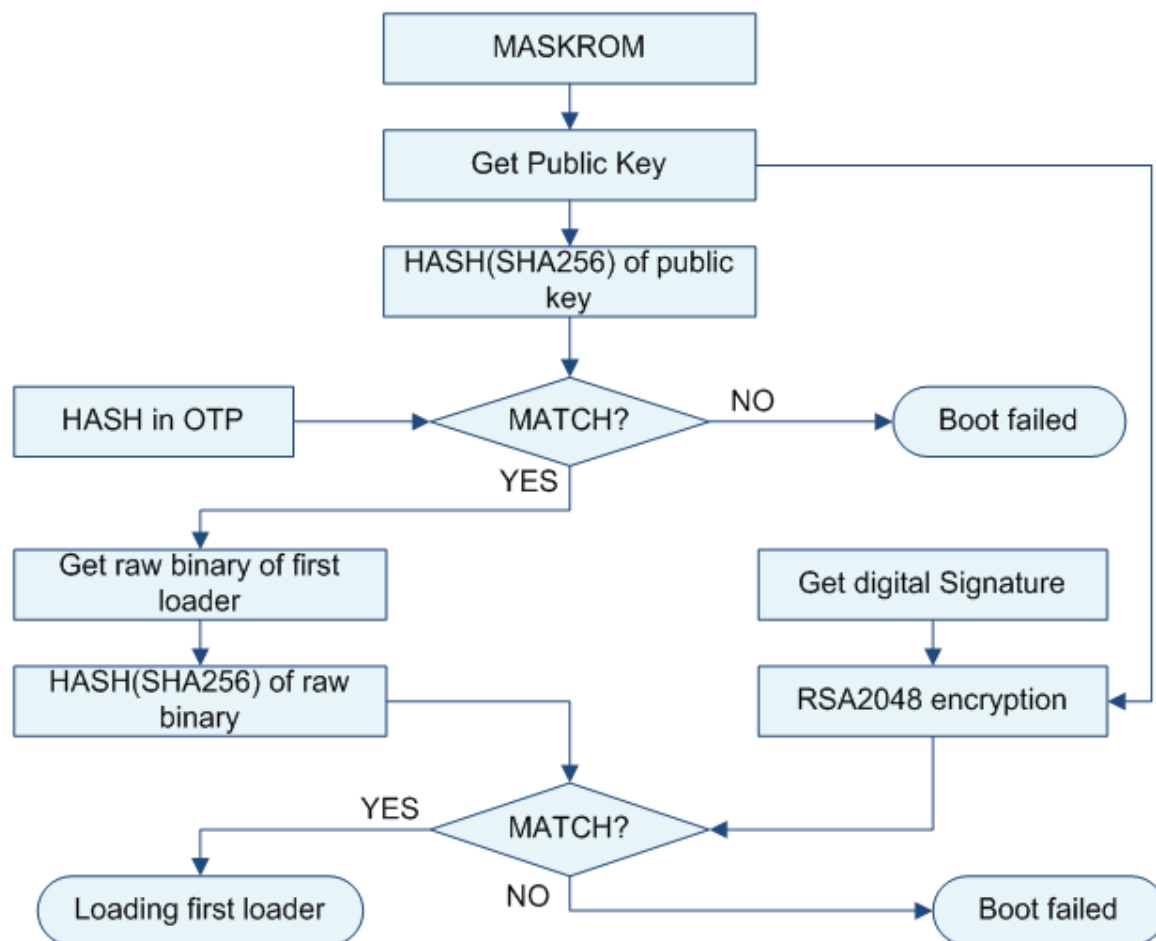


A/B System烧写

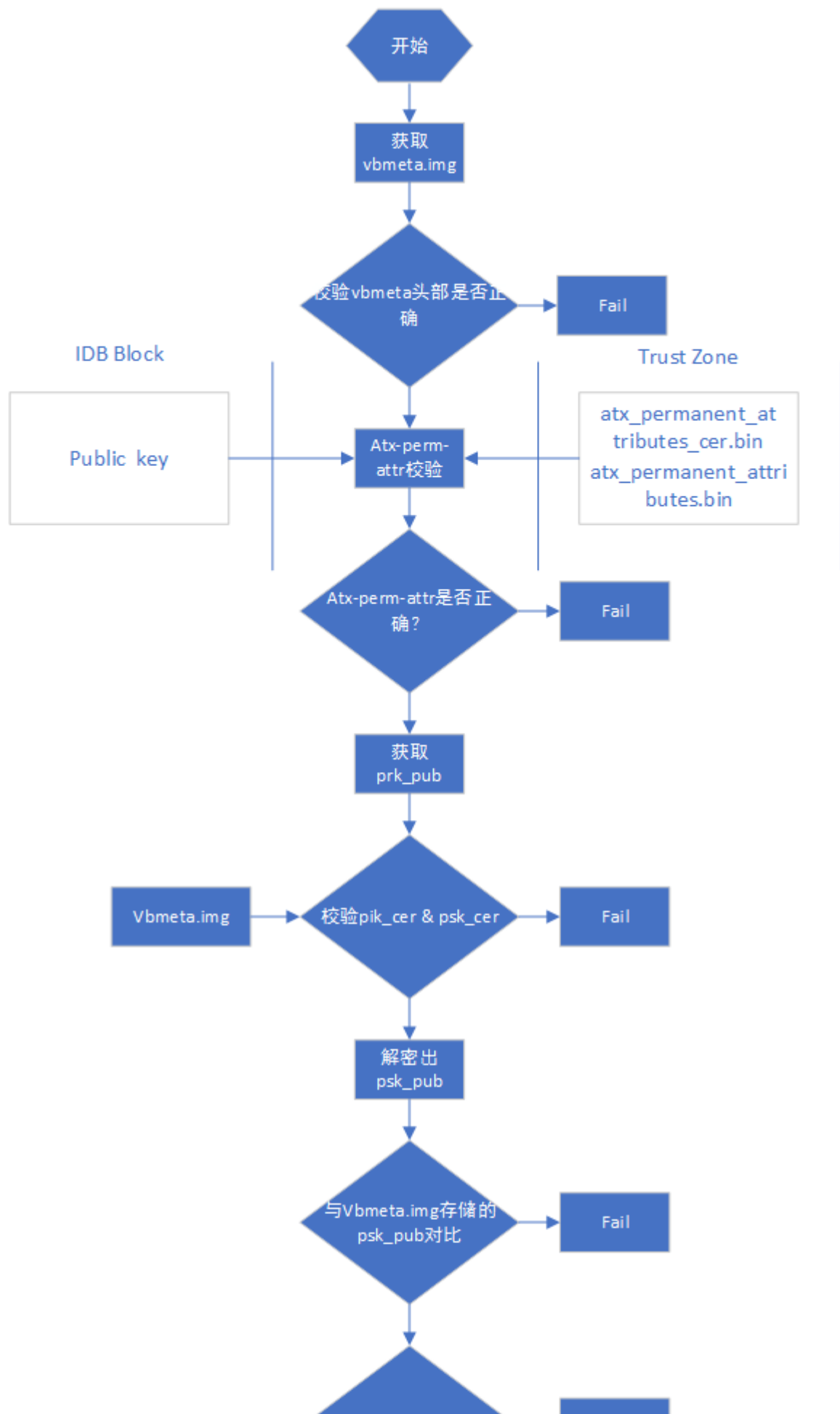


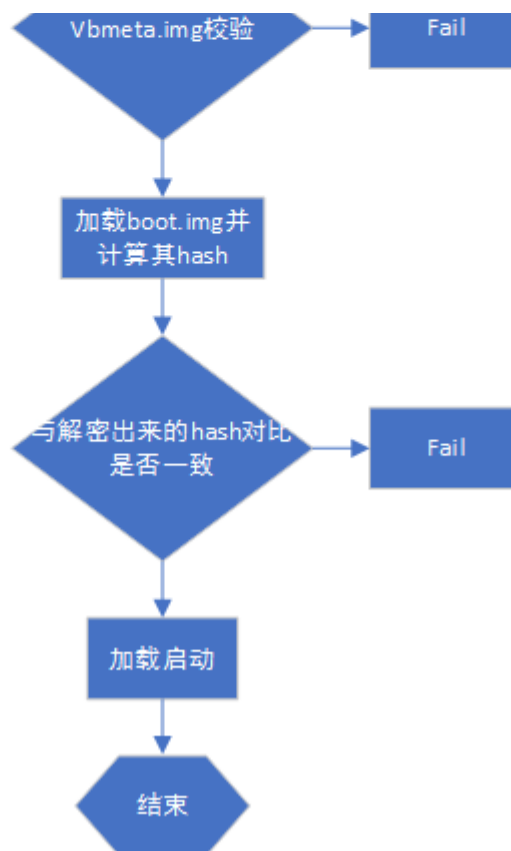
9 pre loader verified

参见《Rockchip-Secure-Boot-Application-Note.md》

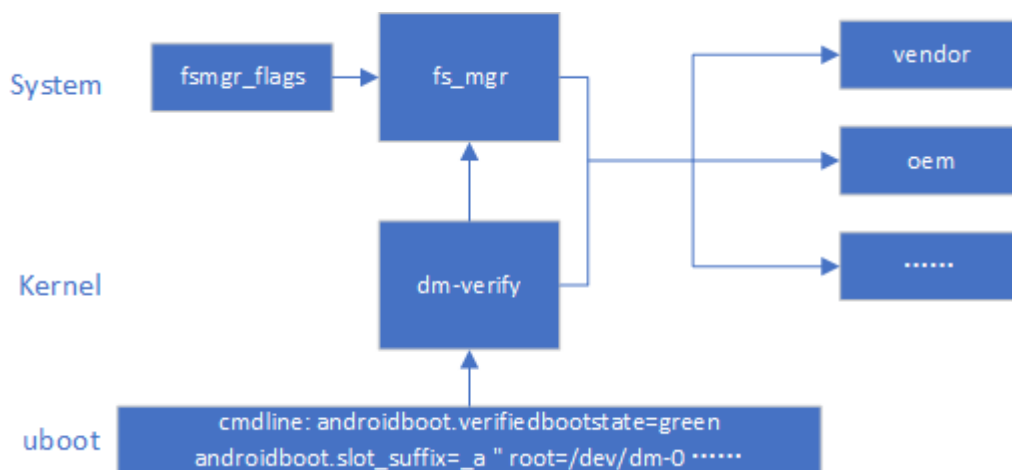


10 U-boot verified





11 系统校验启动



系统启动到kernel，kernel首先解析U-Boot传递的cmdline参数，确认系统启动是否使用dm-verify。然后加载启用system的fs_mgr服务。fs_mgr依据fsmgr_flags的参数来校验加载固件，固件hash & hash tree存放于vbmeta.img。主要有如下参数：

avb：使用avb的方式加载校验分区

slotselect：该分区分A/B，加载时会使用到cmdline的"androidboot.slot_suffix=_a"这个参数。

12 基于linux环境的AVB操作及验证流程

12.1 操作流程

1. 生成整套固件

2. 使用SecureBootConsole生成PrivateKey.pem与PublicKey.pem，工具为rk_sign_tool，命令如下：

```
rk_sign_tool cc --chip 3399
rk_sign_tool kk --out .
```

3. load key

```
rk_sign_tool lk --key privateKey.pem --pubkey publicKey.pem
```

4. 签名loader

```
rk_sign_tool sl --loader loader.bin
```

5. 签名uboot.img & trust.img

```
rk_sign_tool si --img uboot.img
rk_sign_tool si --img trust.img
```

6. avb签名固件准备：准备空的temp.bin，16字节的atx_product_id.bin，待签名的boot.img，运行下列代码

```
#!/bin/bash
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 -outform PEM -out
testkey_atx_prk.pem
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 -outform PEM -out
testkey_atx_psk.pem
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:4096 -outform PEM -out
testkey_atx_pik.pem
python avbtool make_atx_certificate --output=atx_pik_certificate.bin --subject=temp.bin --
subject_key=testkey_atx_pik.pem --subject_is_intermediate_authority --subject_key_version 42
--authority_key=testkey_atx_prk.pem
python avbtool make_atx_certificate --output=atx_psk_certificate.bin --
subject=atx_product_id.bin --subject_key=testkey_atx_psk.pem --subject_key_version 42 --
authority_key=testkey_atx_pik.pem
python avbtool make_atx_metadata --output=atx_metadata.bin --
intermediate_key_certificate=atx_pik_certificate.bin --
product_key_certificate=atx_psk_certificate.bin
python avbtool make_atx_permanent_attributes --output=atx_permanent_attributes.bin --
product_id=atx_product_id.bin --root_authority_key=testkey_atx_prk.pem
python avbtool add_hash_footer --image boot.img --partition_size 33554432 --partition_name
boot --key testkey_atx_psk.pem --algorithm SHA256_RSA4096
python avbtool make_vbmeta_image --public_key_metadata atx_metadata.bin --
include_descriptors_from_image boot.img --algorithm SHA256_RSA4096 --key testkey_atx_psk.pem
--output vbmeta.img
openssl dgst -sha256 -out atx_permanent_attributes_cer.bin -sign PrivateKey.pem
atx_permanent_attributes.bin
```

生成vbmeta.img，atx_permanent_attributes_cer.bin，atx_permanent_attributes.bin。

该步骤就签名了boot.img.....

7.固件烧写

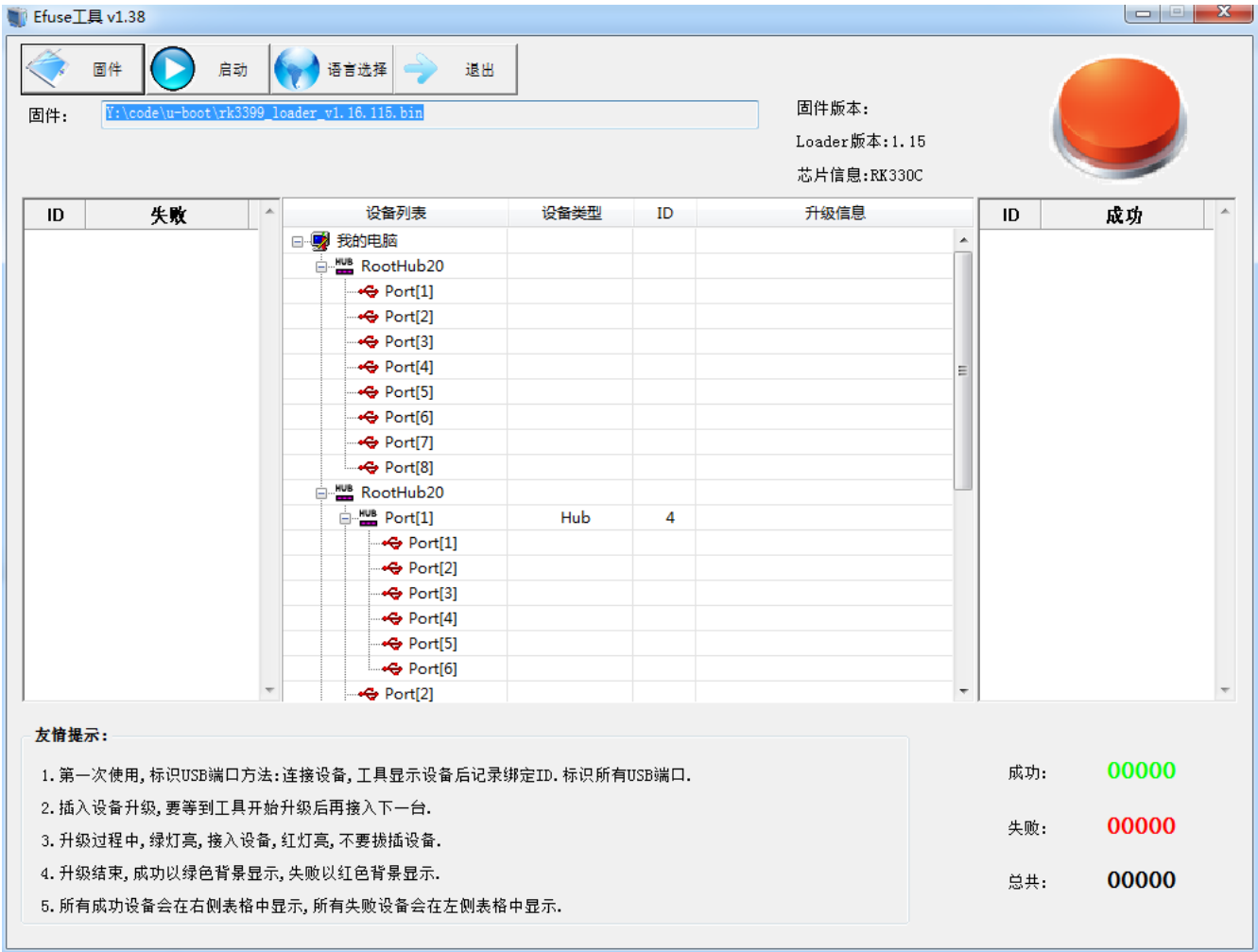
```
rkdeveloptool db loader.bin
rkdeveloptool u1 loader.bin
rkdeveloptool gpt parameter.txt
rkdeveloptool wlx uboot uboot.img
rkdeveloptool wlx trust trust.img
rkdeveloptool wlx boot boot.img
rkdeveloptool wlx system system.img
```

rkdeveloptool可以参考<https://github.com/rockchip-linux/rkdeveloptool>

8. 烧写atx_permanent_attributes_cer.bin , atx_permanent_attributes.bin

```
fastboot stage atx_permanent_attributes.bin
fastboot oem fuse at-perm-attr
fastboot stage atx_permanent_attributes_cer.bin
fastboot oem fuse at-rsa-perm-attr
```

9. efuse烧写（efuse工具目前只有windows版本），选择特定的loader，选择对应的设备，点击启动烧写。



12.2 验证流程