**No.18 Building, A District, No.89,
software Boulevard Fuzhou,Fujian, PRC
Tel: 0591-83991906-8006
Email:zyf@rock-chips.com**

Rockchip Secure Boot Application Note

Revision1.7

2017/05/19

Revision History

Revision	Date	Description	Author
1.0	2014-11-05	Original Document	ZYF
1.1	2015-12-21	Update Secure Boot Tool	YBC
1.2	2016-02-02	Update Secure Boot Tool	YHC
1.3	2016-09-29	RE-EDIT	ZYF
1.4	2016-11-15	add detail description of workflow	Joshua
1.5	2016-11-16	1. Add Terms and Definitions. 2. Add EFUSE layout.	Joshua
1.6	2017-02-15	Add RK3328 and RK3228H.	ZYF
1.7	2017-05-19	Add sequence chart and NOTE	ZZJ

Table of Contents

1.Terms and Definitions

sector: sector size is 512 bytes。

efuse: One-Time Programmable Memory IP in SOC

2. Basic Feature

Secure boot mechanism is for verifying firmware validity, which aims at preventing invalid firmware upgrade and booting.

The device which had programmed EFUSE will enable secure boot rom, and could not boot from the un-signed firmware. So try to upgrade un-signed firmware or un-match key signed firmware will fail.

NOTE: The valid signed firmware can boot smoothly on fake copies of device circuit board or same CPU platform hardware. Secure boot will verify the validity of software, but not hardware.

This document applies to RK3126, RK3128, RK3228, RK3229, RK3288, RK3368, RK3399, RK3228H and RK3328.

Secure boot feature:

1.1 Support Secure Boot Rom

1.2 Support SHA256

1.3 Support RSA2048

1.4 Support EFUSEHASH to verify public key

The relative tool and loader revision:

1、 MiniloaderV2.19 or the latest revision

2、 Uboot V2.17or the latest revision

3、 Efuse tool V1.35or the latest revision

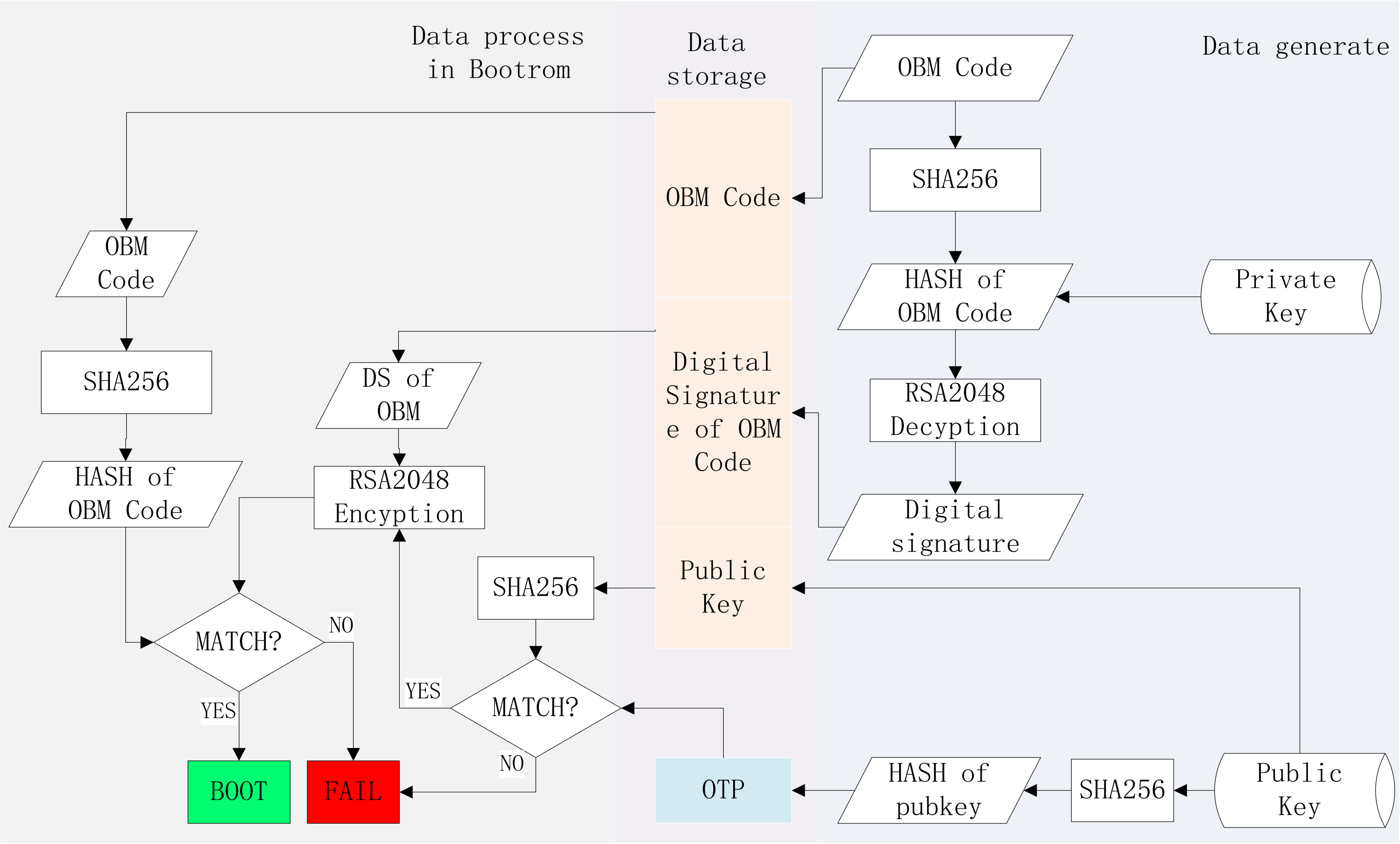
4、 SecureBootTool 1.79 or the latest revision

5、 RKBatchTool 1.8 or the latest revision

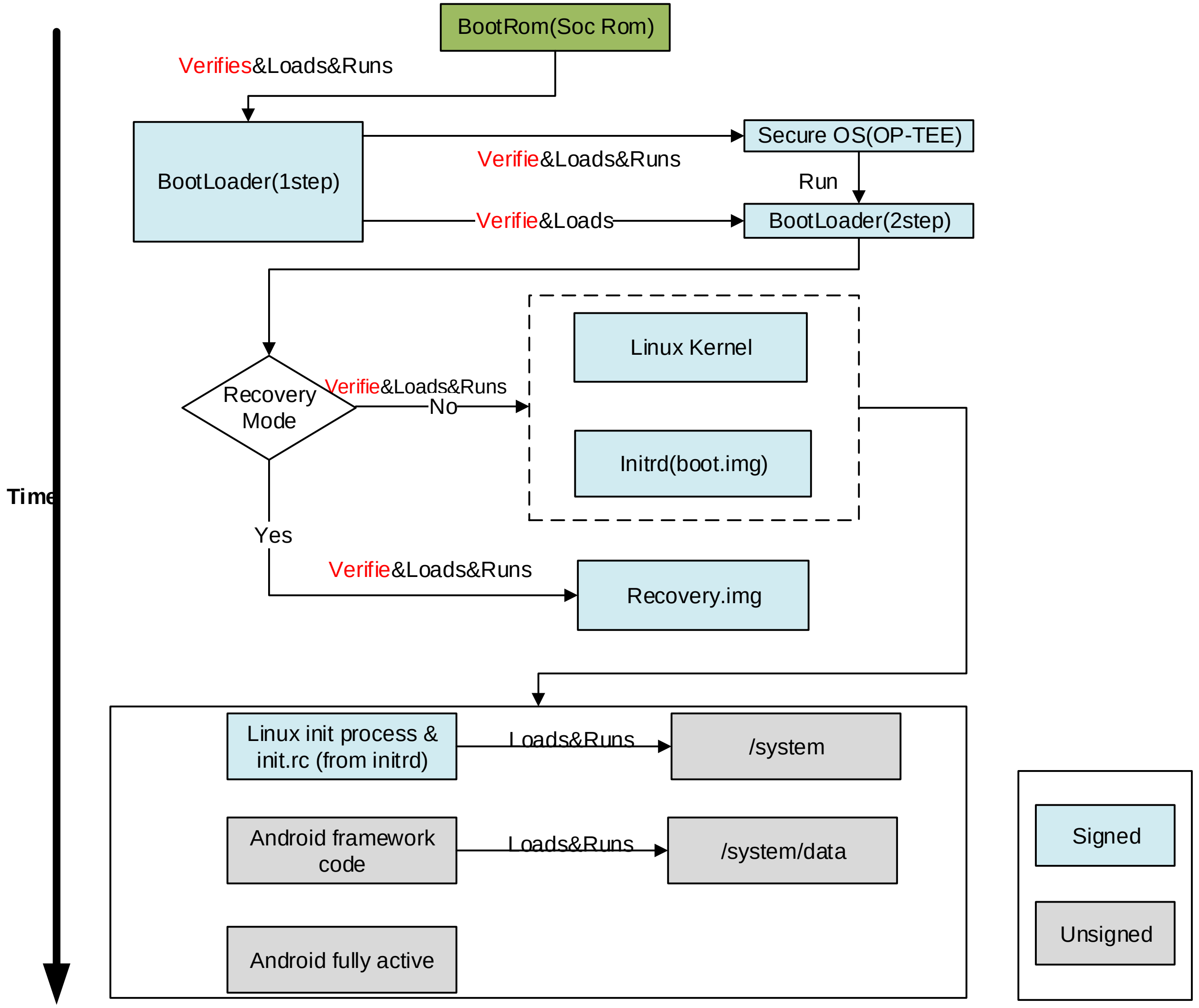
6、 FactoryTool 1.39 or the latest revision

3. Secure Boot Architecture

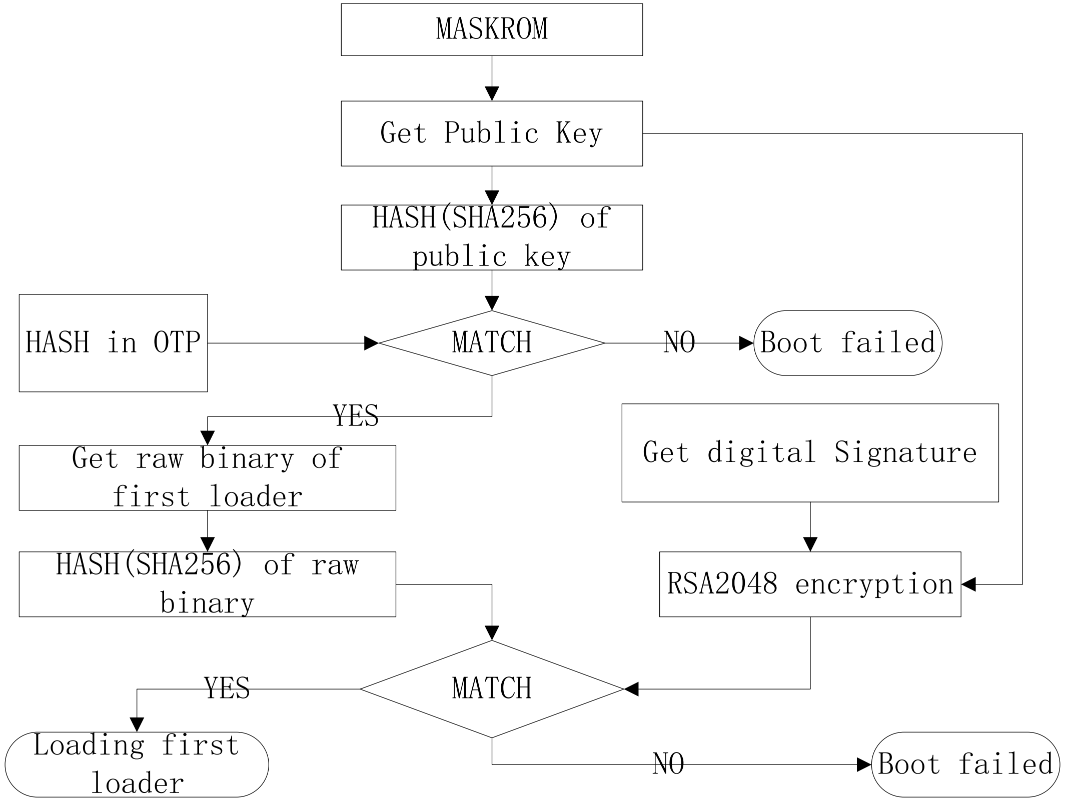
1. Secure Boot Process



2. Secure Boot Sequence



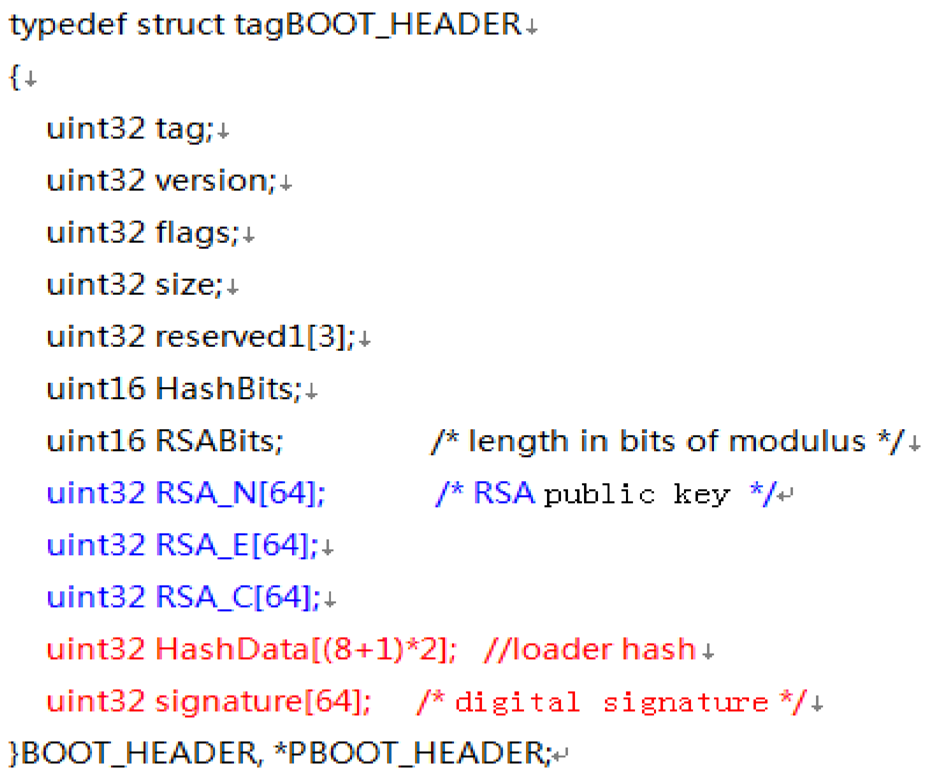
3.1 MASKROM boot to the first loader (RKminiLoader/uboot)



First loader layout in user partition of flash

0-63 sector	64 sector reverse	
first loader (8128 sector) (5 copys)	Boot loader partition	
	0-2047	loader header
	2048-4095	public key and digital signature
	4096 -	raw binary
	…	
	Boot loader copy(4) partition	
	0-2047	loader header
	2048-4095	public key and digital signature
	4096 -	raw binary

The structure of public key and digital signature layout at address 2048 to 4095:



```
typedef struct tagBOOT_HEADER↓
{↓
    uint32 tag;↓
    uint32 version;↓
    uint32 flags;↓
    uint32 size;↓
    uint32 reserved1[3];↓
    uint16 HashBits;↓
    uint16 RSABits;            /* length in bits of modulus */↓
    uint32 RSA_N[64];          /* RSA public key */↵
    uint32 RSA_E[64];↓
    uint32 RSA_C[64];↓
    uint32 HashData[(8+1)*2];  //loader hash↓
    uint32 signature[64];    /* digital signature */↓
}BOOT_HEADER, *PBOOT_HEADER;↵
```

Public key: uint32 RSA_E[64] uint32 RSA_E[64] uint32 RSA_C[64];

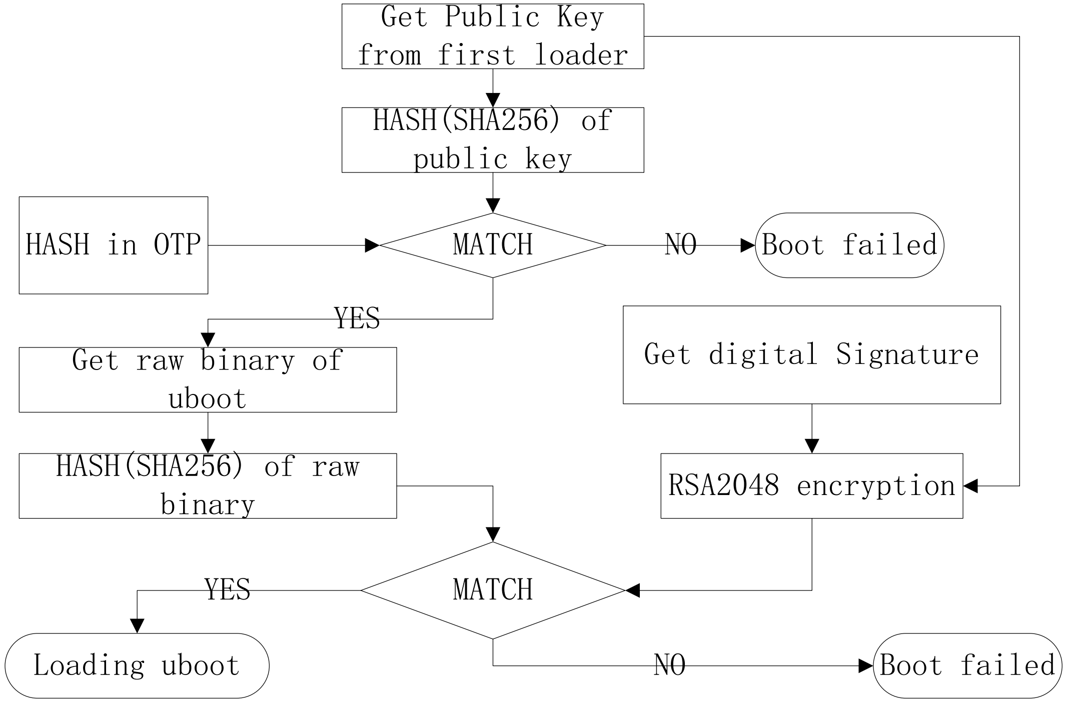
Digital signature: uint32 signature[64]

Step 1: Get public key from first loader partition.

Step 2: Calculate the hash(sha256) of public key and compare it with the hash stored in OTP, If matched, load the first loader successfully, otherwise booting failed.

step 3: Calculate the hash(SHA256) of raw binary and compare it with RSA2048 encryption(have been obtained in step 1) of digital signature. If matched, load the first loader successfully, otherwise booting failed.

3.2 First loader boot to u-boot(Secondary Boot Loader, option)

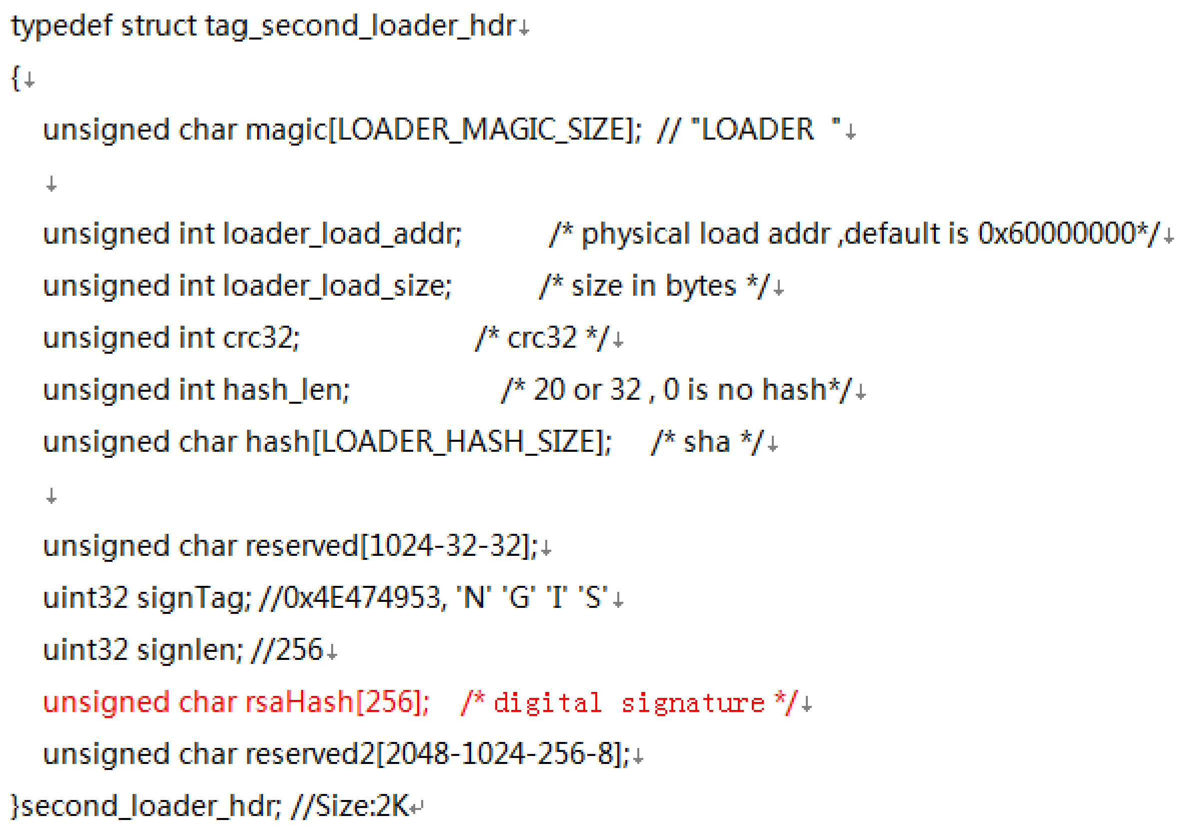


uboot layout in user partition of flash

uboot (4MB，4copys)	UBoot	
	0-2047	header，digital signature
	2048-	Raw binary
	…	
	UBoot copy(3)	
	0-2047	header，digital signature
	2048-	Raw binary

structure of　layout 0-2047 (header，digital signature)

The structure of header with digital signature layout at address 0 to 2047:



```
typedef struct tag_second_loader_hdr
{
    unsigned char magic[LOADER_MAGIC_SIZE];  // "LOADER  "

    unsigned int loader_load_addr;          /* physical load addr ,default is 0x60000000*/
    unsigned int loader_load_size;          /* size in bytes */
    unsigned int crc32;                     /* crc32 */
    unsigned int hash_len;                  /* 20 or 32 , 0 is no hash*/
    unsigned char hash[LOADER_HASH_SIZE];    /* sha */

    unsigned char reserved[1024-32-32];
    uint32 signTag; //0x4E474953, 'N' 'G' 'I' 'S'
    uint32 signlen; //256
    unsigned char rsaHash[256];   /* digital  signature */
    unsigned char reserved2[2048-1024-256-8];
}second_loader_hdr; //Size:2K
```

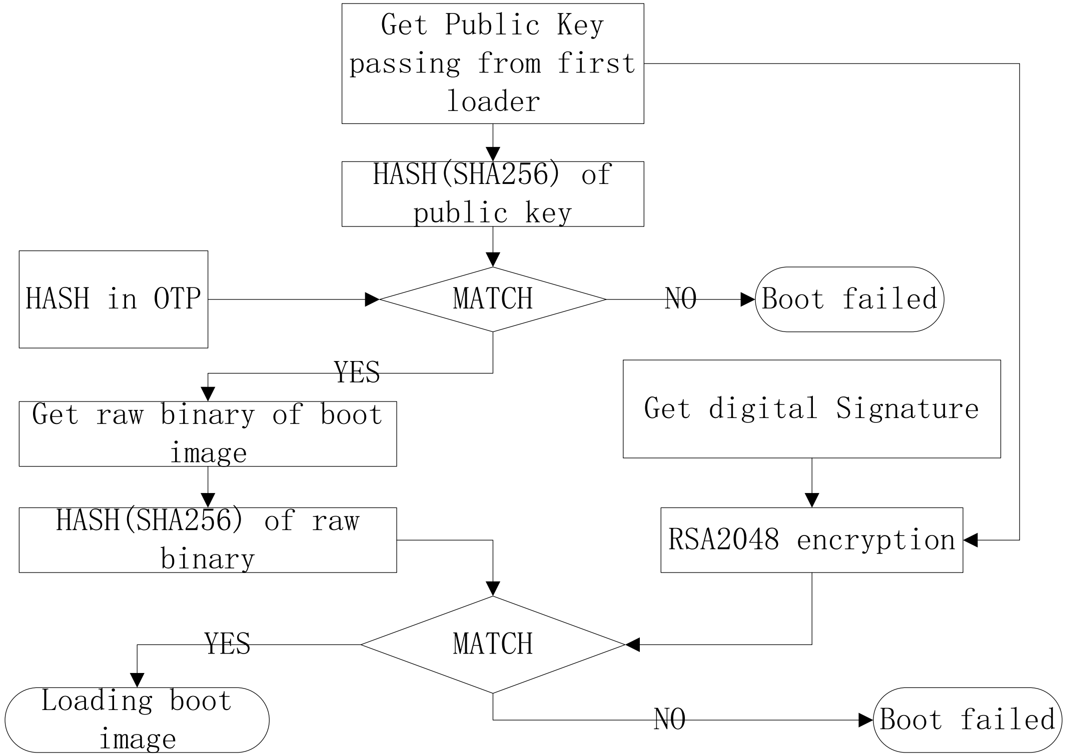
digital signature: unsigned char rsaHash[256];

Step 1: Get public key from first loader partiotion

Step 2: Calculate the hash (sha256) of public key and compare it with hash in OTP, if matched go to next step, otherwise booting failed.

Step 3: Calculate the hash(SHA256) of raw binary and compare it with RSA2048 encryption(have been obtained in step 1) of digital signature, if matched ,loading successfully and deliver the public key to U-Boot, otherwise booting failed.

3.3 u-boot boot to boot image(with linux kernel)


Get Public Key passing from first loader
HASH(SHA256) of public key
HASH in OTP
MATCH
NO
Boot failed
YES
Get raw binary of boot image
Get digital Signature
HASH(SHA256) of raw binary
RSA2048 encryption
MATCH
YES
Loading boot image
NO
Boot failed

boot.img	0-2047	header
	2048-4095	digital signature
	4096-	kernel,ramdisk,dtb···
	…	

The structure of layout 0-2047(header):

```
#define BOOT_MAGIC_SIZE 8
#define BOOT_NAME_SIZE 16
#define BOOT_ARGS_SIZE 512
typedef struct tag_boot_img_hdr{
    unsigned char magic[BOOT_MAGIC_SIZE]; /*"ANDROID!"*/
    unsigned int kernel_size;  /* size in bytes */
    unsigned int kernel_addr;  /* physical load addr */
    unsigned int ramdisk_size; /* size in bytes */
    unsigned int ramdisk_addr; /* physical load addr */
    unsigned int second_size;  /* size in bytes */
    unsigned int second_addr;  /* physical load addr */
    unsigned int tags_addr;    /* physical addr for kernel tags */
    unsigned int page_size;    /* flash page size we assume */
    unsigned int unused[2];    /* future expansion: should be 0 */
    unsigned char name[BOOT_NAME_SIZE]; /* asciiz product name */
    unsigned char cmdline[BOOT_ARGS_SIZE];
    unsigned int id[8]; /* timestamp / checksum / sha1 / etc */
    unsigned char reserved[0x400-0x260];
    uint32 signTag; //0x4E474953
    uint32 signlen; //128
    unsigned char rsaHash[128];   /* digital signature */
}boot_img_hdr;
```

digital signature: unsigned char rsaHash[128];

Step 1: u-boot get public key obtained from first loader.

Step 2: Calculate the hash (sha256) of public key and compare it with hash in OTP, if matched go to next step, otherwise booting failed.

Step 3: HASH(SHA256) of raw binary and compare it with RSA2048 encryption(using public key get in step 1) of digital signature, if matched, boot to linux kernel , otherwise booting failed.

3.4 u-boot boot to recovery

The same as boot to boot image, detail please refer to chapter 3.3

.

4. EFUSE layout

RK3368, RK3228, RK3229 and RK3228 used 1024 bits EFUSE for secure boot, data layout:

32-bit Word Addressing	Description
0x00	Security flag Bits [7:0] security enable flag Bits [31:8] Reserved
0x01-0x3	Reserved
0x04-0x07	Reserved
0x8-0xF	RSA public key hash
0x10-0x17	Reserved
0x18	Reserved
0x19-0x1A	Reserved
0x1B-0x1D	Reserved
0x1E	Reserved
0x1F	Efuse write Lock Bits

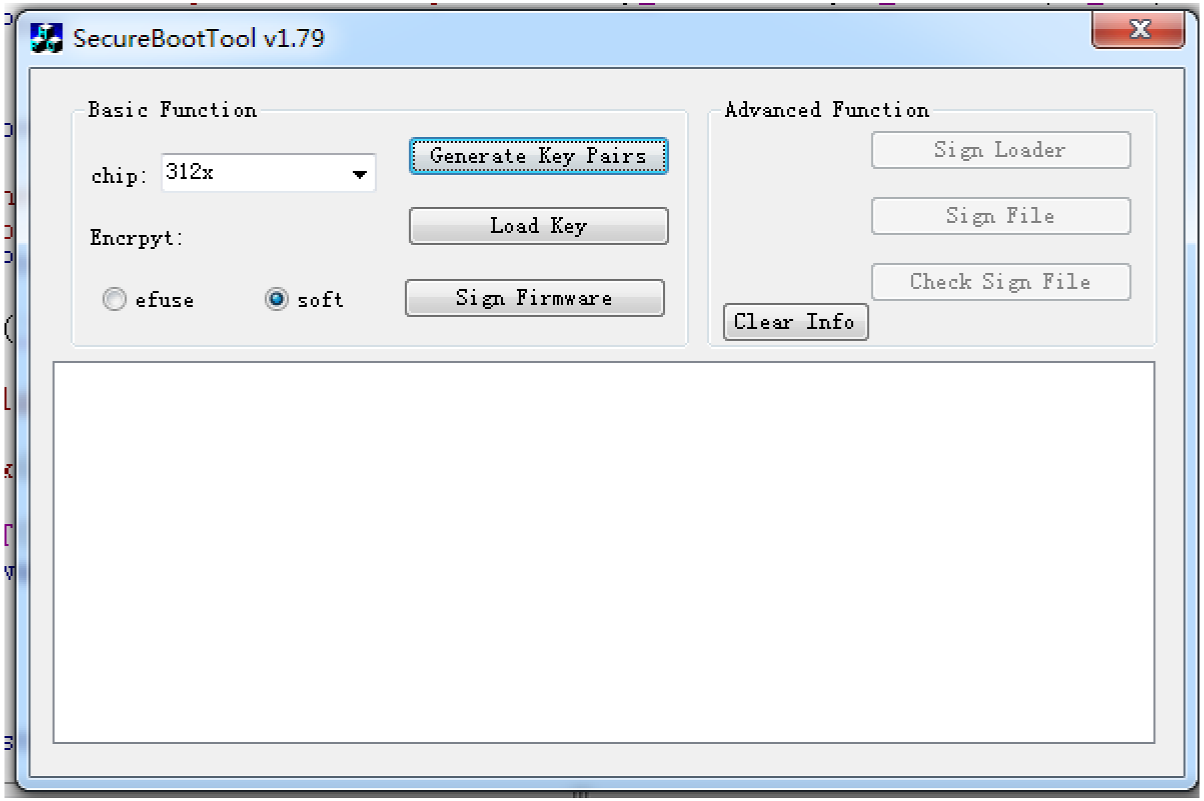
RK3228H and RK3328 used 7680 bits OTP for secure boot, data layout:

32-bit Word Addressing	Description
0-63	Public Key (N)
64-127	Public Key (E)
128	Security flag Bits [7:0] 0xff: security enable flag Bits [15:8] : RSA_E size (word uint) Bits [31:16] Reserved
129	Trusted Firmware revocation counter (ID #0)
130-131	Non-trusted Firmware revocation counter (ID #1)
132-239	Reserved

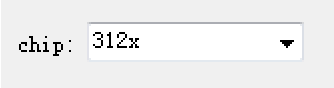
5. Firmware Sign Flow

This instruction is for Windows tools, while Linux have its own.

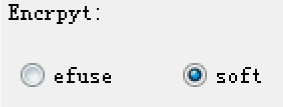
5.1 Sign tool UI



5.2 Configuration:


chip: 312x

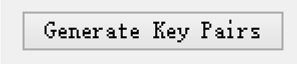
: Choose SOC platform


Encrpyt:

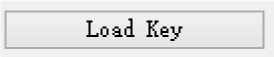
○ efuse ● soft

: Option ‘efuse’is means used EFUSE to store the hash of the RSA public key, and will enable secure boot rom (recommended).

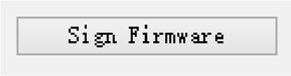
Option ‘soft’is for some special applications, will not enable secure boot rom, used RSA1024 and SHA160.



: Every product model will generate RSA KEY only once,please backup in case that cannot upgrade firmware or OTA again.

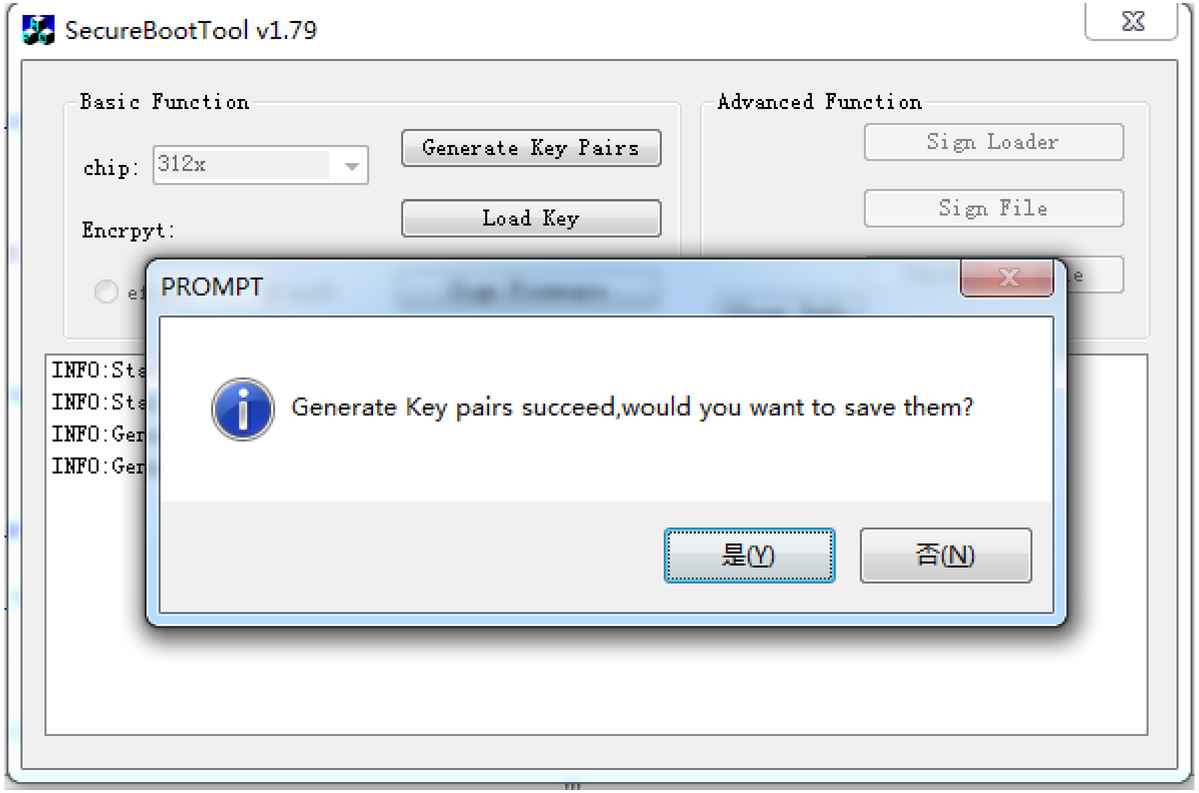


: Loading backup RSA key (support ‘.pem’file format generated byopenssl)



: Sign firmware

5.3Generating RSA key

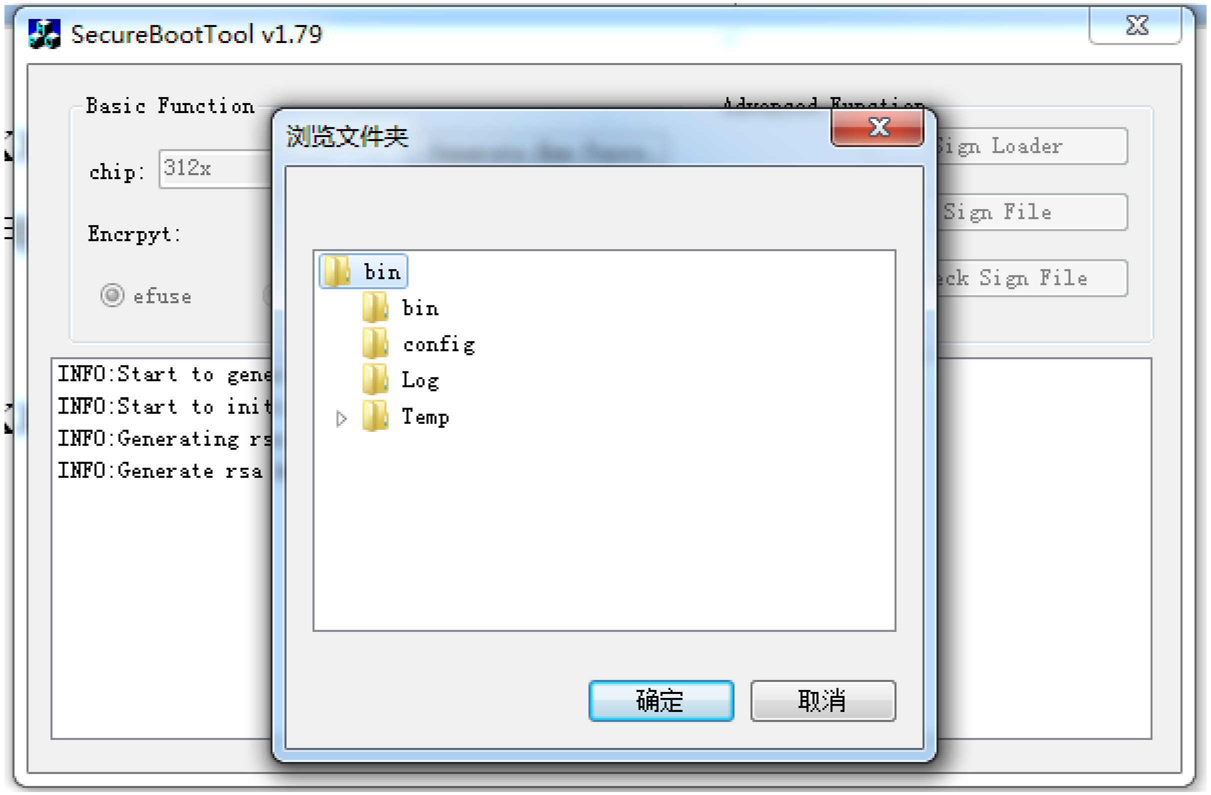


INFO:St
INFO:St
INFO:Ge
INFO:Ge

Generate Key pairs succeed,would you want to save them?

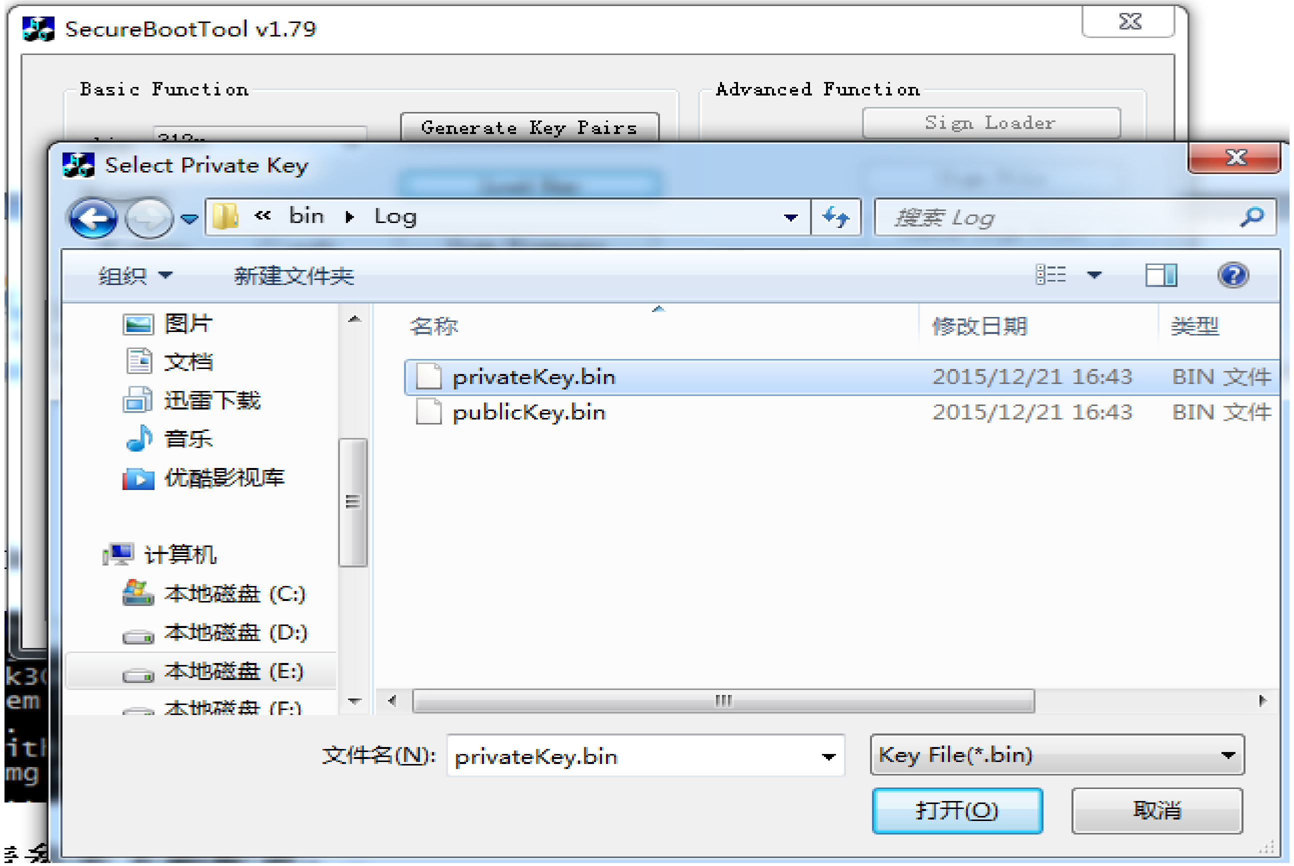
5.4 Save RSA key

This key will be used for signed firmware and for OTA,please back up to asecure storage



```
INFO:Start to gene
INFO:Start to init
INFO:Generating rs
INFO:Generate rsa
```

5.5Loading RSA key



名称	修改日期	类型
privateKey.bin	2015/12/21 16:43	BIN 文件
publicKey.bin	2015/12/21 16:43	BIN 文件

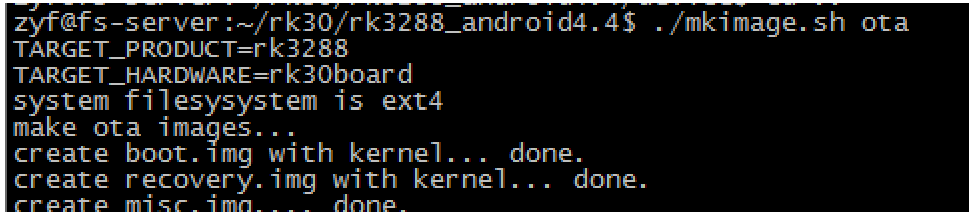
文件名(N): privateKey.bin

Key File(*.bin)

5.6 Sign Firmware

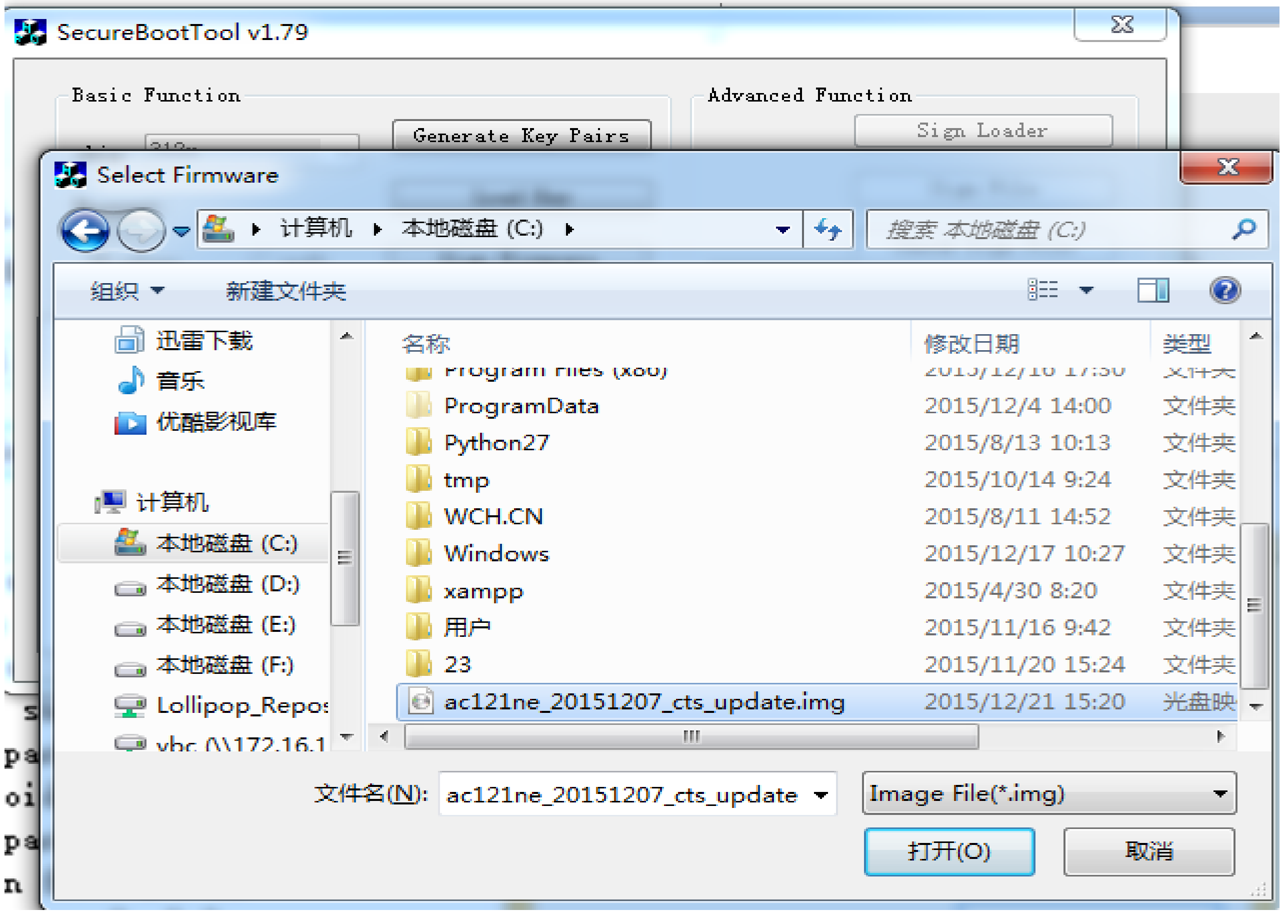
Make sure the ‘boot.img’ and the ‘recovery.img’ are included kernel image.

Refer to the pack command:



```
zyf@fs-server:~/rk30/rk3288_android4.4$ ./mkimage.sh ota
TARGET_PRODUCT=rk3288
TARGET_HARDWARE=rk30board
system filesysystem is ext4
make ota images...
create boot.img with kernel... done.
create recovery.img with kernel... done.
create misc.img.... done.
```

Open firmware image:

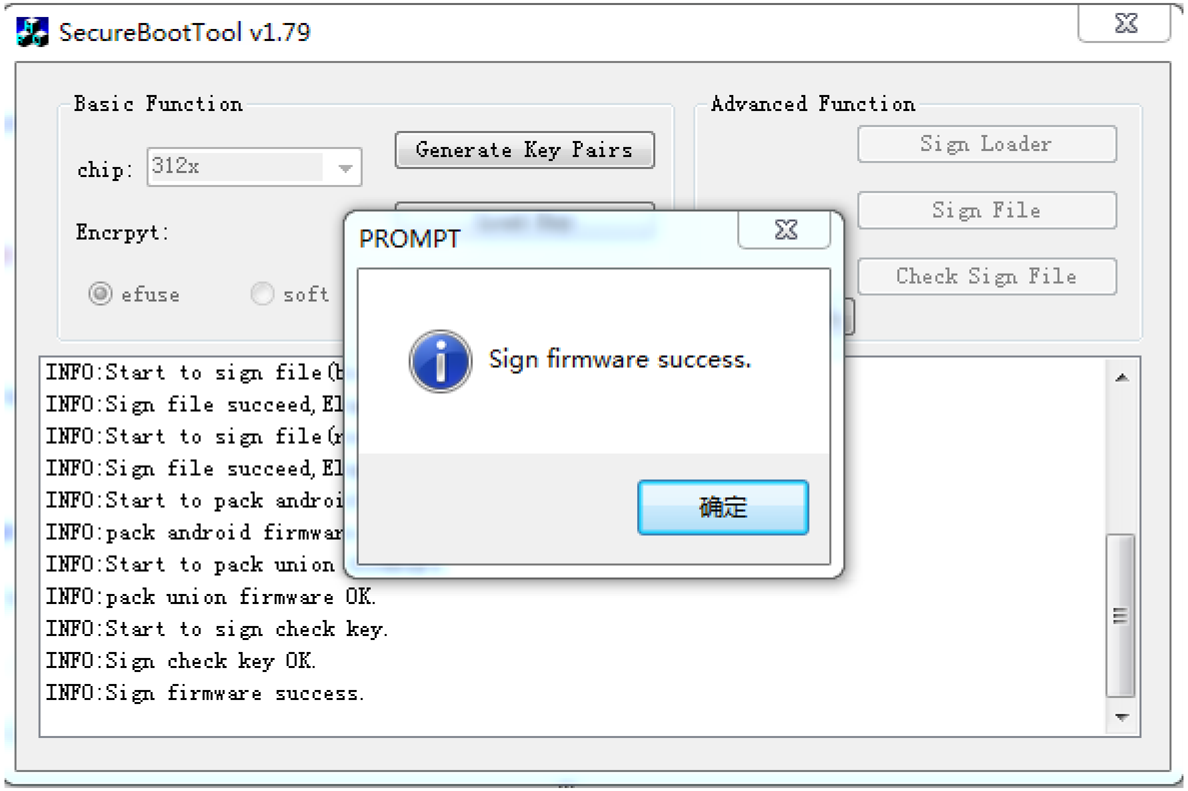


名称	修改日期	类型
Program Files (x86)	2015/12/16 17:30	文件夹
ProgramData	2015/12/4 14:00	文件夹
Python27	2015/8/13 10:13	文件夹
tmp	2015/10/14 9:24	文件夹
WCH.CN	2015/8/11 14:52	文件夹
Windows	2015/12/17 10:27	文件夹
xampp	2015/4/30 8:20	文件夹
用户	2015/11/16 9:42	文件夹
23	2015/11/20 15:24	文件夹
ac121ne_20151207_cts_update.img	2015/12/21 15:20	光盘映

文件名(N): ac121ne_20151207_cts_update

Image File(*.img)

Signed firmware:

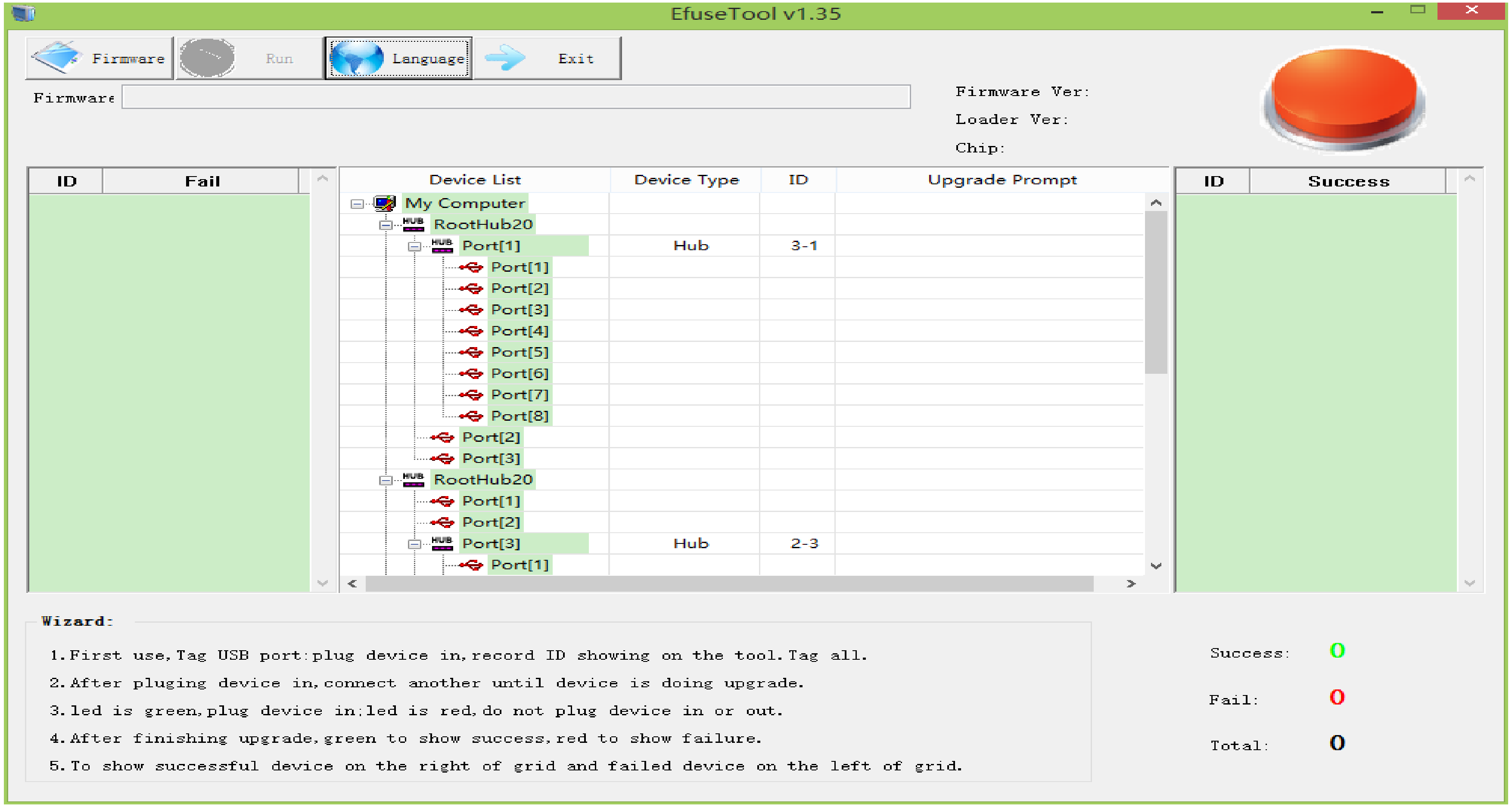


Sign firmware success.

```
INFO:Start to sign file(b
INFO:Sign file succeed,El
INFO:Start to sign file(r
INFO:Sign file succeed,El
INFO:Start to pack androi
INFO:pack android firmwar
INFO:Start to pack union
INFO:pack union firmware OK.
INFO:Start to sign check key.
INFO:Sign check key OK.
INFO:Sign firmware success.
```

6. Programming EFUSE

6.1 Tool UI



Device List	Device Type	ID	Upgrade Prompt
My Computer			
RootHub20			
Port[1]	Hub	3-1	
Port[1]			
Port[2]			
Port[3]			
Port[4]			
Port[5]			
Port[6]			
Port[7]			
Port[8]			
Port[2]			
Port[3]			
RootHub20			
Port[1]			
Port[2]			
Port[3]	Hub	2-3	
Port[1]			

Wizard:

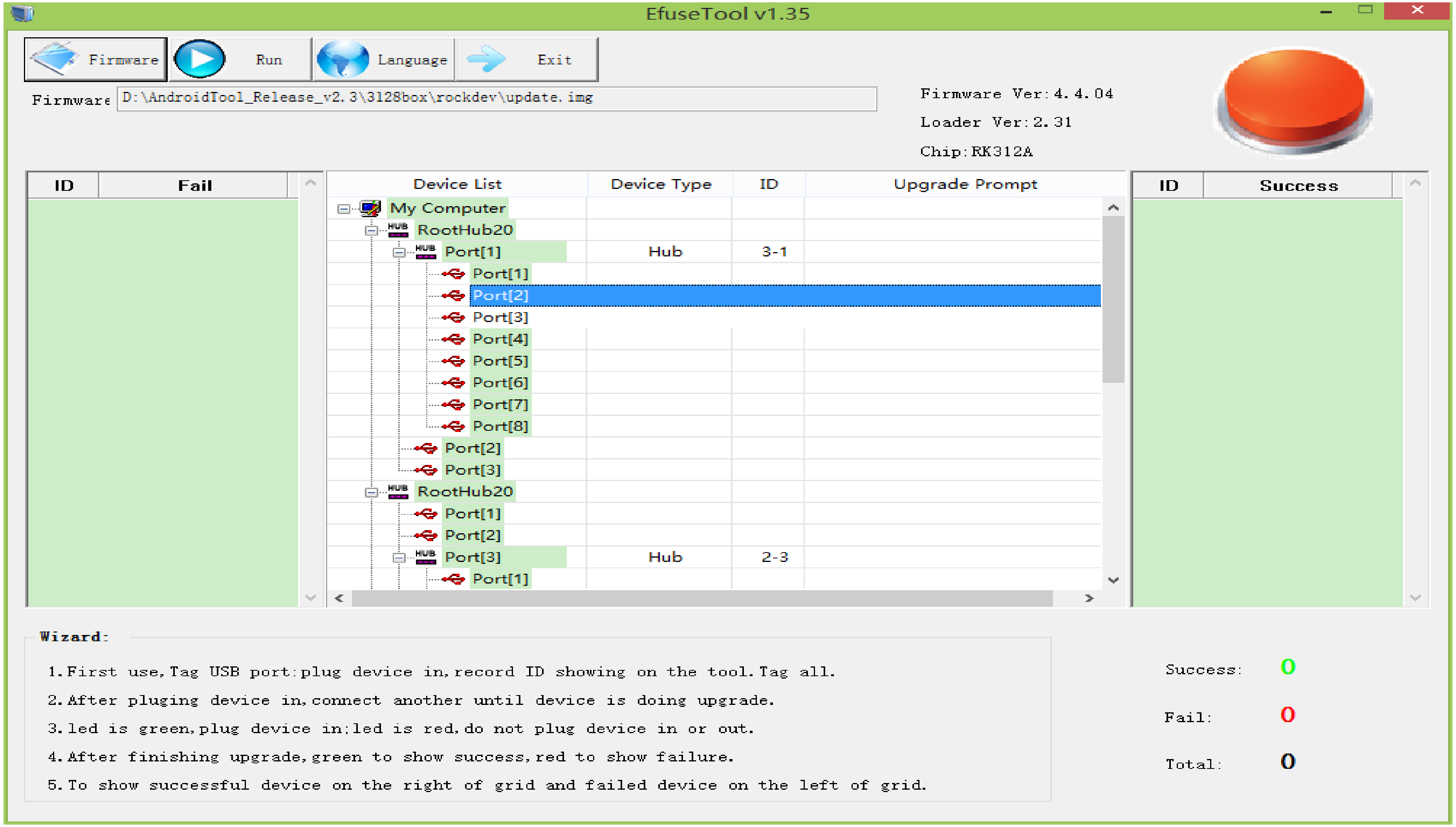
1. First use, Tag USB port: plug device in, record ID showing on the tool. Tag all.
2. After pluging device in, connect another until device is doing upgrade.
3. led is green, plug device in; led is red, do not plug device in or out.
4. After finishing upgrade, green to show success, red to show failure.
5. To show successful device on the right of grid and failed device on the left of grid.

Success: 0

Fail: 0

Total: 0

6.2 Load the signed firmware



Firmware D:\AndroidTool_Release_v2.3\3128box\rockdev\update.img

Firmware Ver:4.4.04

Loader Ver:2.31

Chip:RK312A

ID	Fail

Device List	Device Type	ID	Upgrade Prompt
My Computer			
RootHub20			
Port[1]	Hub	3-1	
Port[1]			
Port[2]			
Port[3]			
Port[4]			
Port[5]			
Port[6]			
Port[7]			
Port[8]			
Port[2]			
Port[3]			
RootHub20			
Port[1]			
Port[2]			
Port[3]	Hub	2-3	
Port[1]			

ID	Success

Wizard:

1.First use,Tag USB port:plug device in,record ID showing on the tool.Tag all.

2.After pluging device in,connect another until device is doing upgrade.

3.led is green,plug device in;led is red,do not plug device in or out.

4.After finishing upgrade,green to show success,red to show failure.

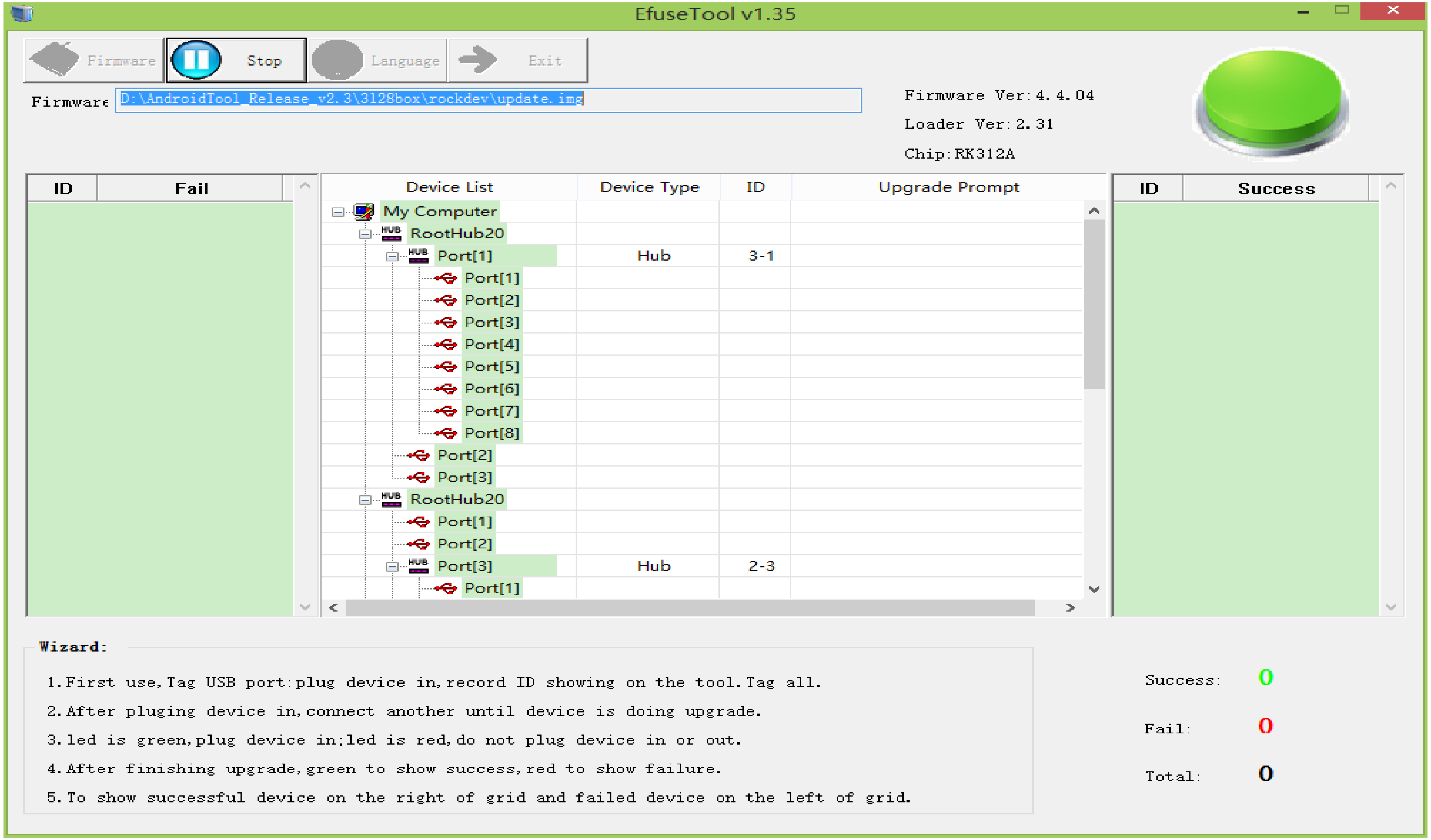
5.To show successful device on the right of grid and failed device on the left of grid.

Success: 0

Fail: 0

Total: 0

6.3 Click'run' Button to start



Firmware D:\AndroidTool_Release_v2.3\3128box\rockdev\update.img

Firmware Ver:4.4.04

Loader Ver:2.31

Chip:RK312A

ID	Fail

Device List	Device Type	ID	Upgrade Prompt
My Computer			
RootHub20			
Port[1]	Hub	3-1	
Port[1]			
Port[2]			
Port[3]			
Port[4]			
Port[5]			
Port[6]			
Port[7]			
Port[8]			
Port[2]			
Port[3]			
RootHub20			
Port[1]			
Port[2]			
Port[3]	Hub	2-3	
Port[1]			

ID	Success

Wizard:

1.First use,Tag USB port:plug device in,record ID showing on the tool.Tag all.

2.After pluging device in,connect another until device is doing upgrade.

3.led is green,plug device in;led is red,do not plug device in or out.

4.After finishing upgrade,green to show success,red to show failure.

5.To show successful device on the right of grid and failed device on the left of grid.

Success: 0

Fail: 0

Total: 0

6.4 Programming EFUSE

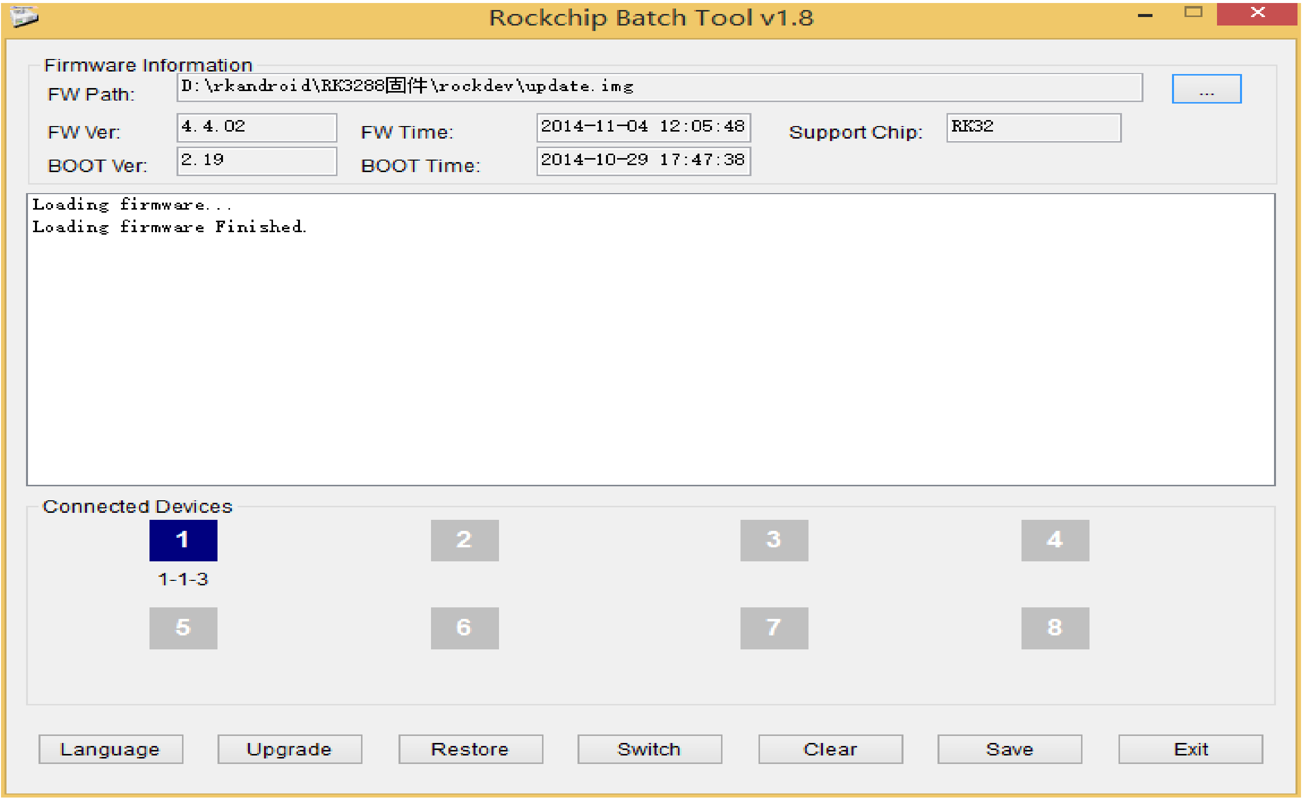
Connect the device to the PC by USB cable; the tool will program the hash of RSA public key to EFUSE automatically.

ProgrammingEFUSE need an external power supply, the detail information please reference to SOC's DATASHEET

7 Firmware Upgrade and Test

7.1 Firmware Upgrade

Open the signed firmware and connect the device which had programmed EFUSE to the PC by USB cable:



FW Path: D:\rkandroid\RK3288固件\rockdev\update.img

FW Ver: 4.4.02

FW Time: 2014-11-04 12:05:48

Support Chip: RK32

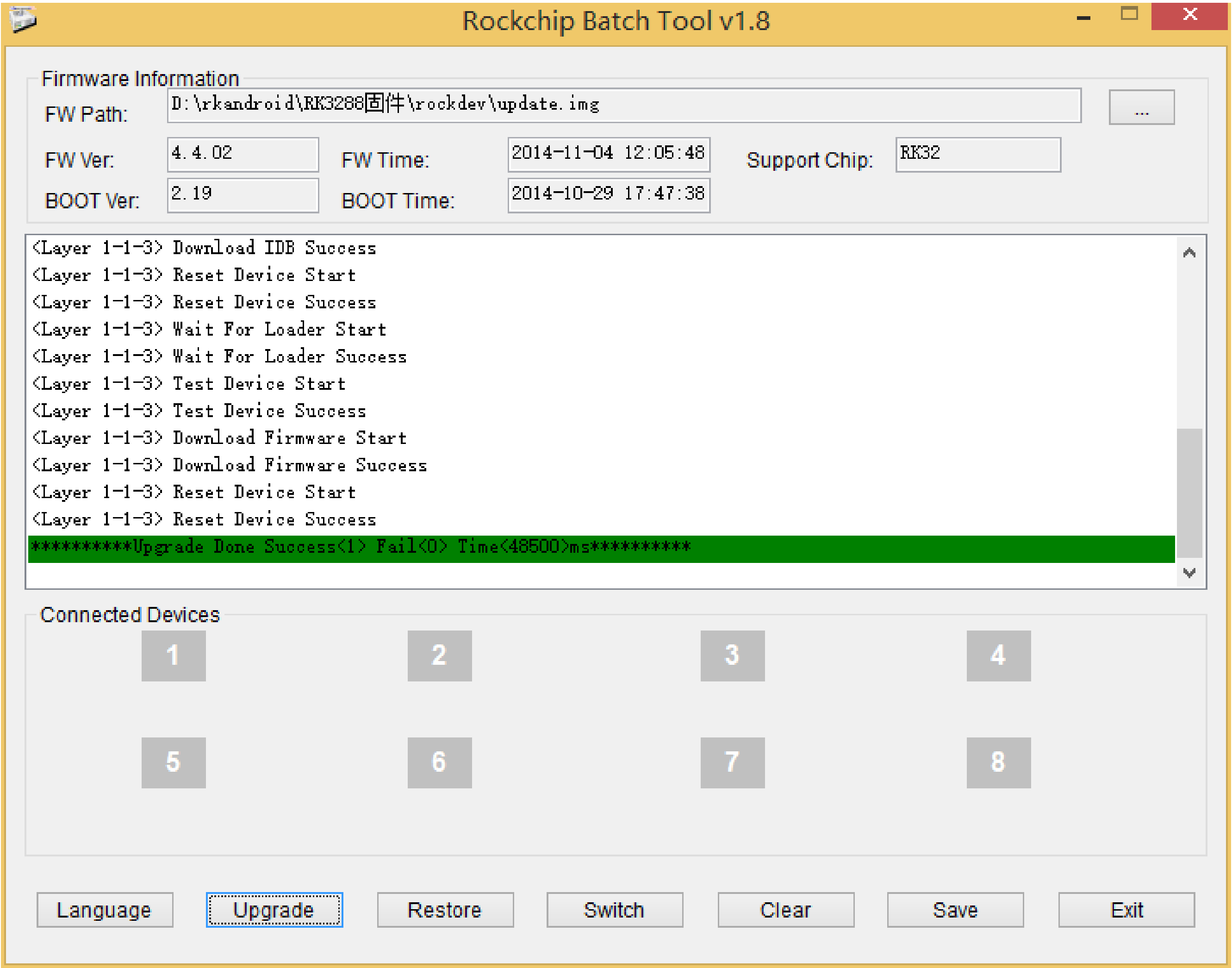
BOOT Ver: 2.19

BOOT Time: 2014-10-29 17:47:38

```
Loading firmware...
Loading firmware Finished.
```

1-1-3

Click the ‘Upgrade’ button to start firmware upgrade and wait completed:



Firmware Information

FW Path: D:\rkandroid\RK3288固件\rockdev\update.img

FW Ver: 4.4.02

FW Time: 2014-11-04 12:05:48

Support Chip: RK32

BOOT Ver: 2.19

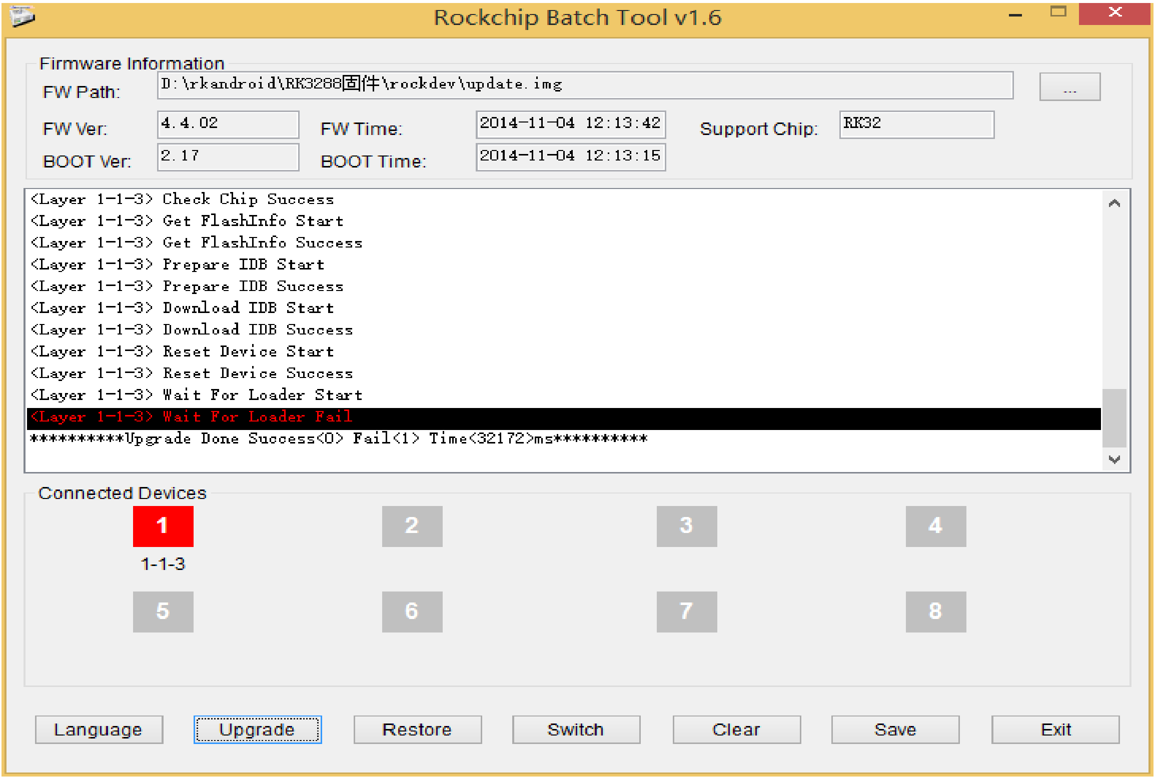
BOOT Time: 2014-10-29 17:47:38

```
<Layer 1-1-3> Download IDB Success
<Layer 1-1-3> Reset Device Start
<Layer 1-1-3> Reset Device Success
<Layer 1-1-3> Wait For Loader Start
<Layer 1-1-3> Wait For Loader Success
<Layer 1-1-3> Test Device Start
<Layer 1-1-3> Test Device Success
<Layer 1-1-3> Download Firmware Start
<Layer 1-1-3> Download Firmware Success
<Layer 1-1-3> Reset Device Start
<Layer 1-1-3> Reset Device Success
**********Upgrade Done Success<1> Fail<0> Time<48500>ms**********
```

7.2 Secure Boot test

The device which had programmed EFUSE will enable secure boot rom, and could not boot from the un-signed firmware.
So try to upgrade un-signed firmware or un-match key signed firmware will fail;
And upgrade match signed firmware will boot success.

SOC 3128 and 3126 will fail at‘wait for loader’:



FW Path: D:\rkandroid\RK3288固件\rockdev\update.img

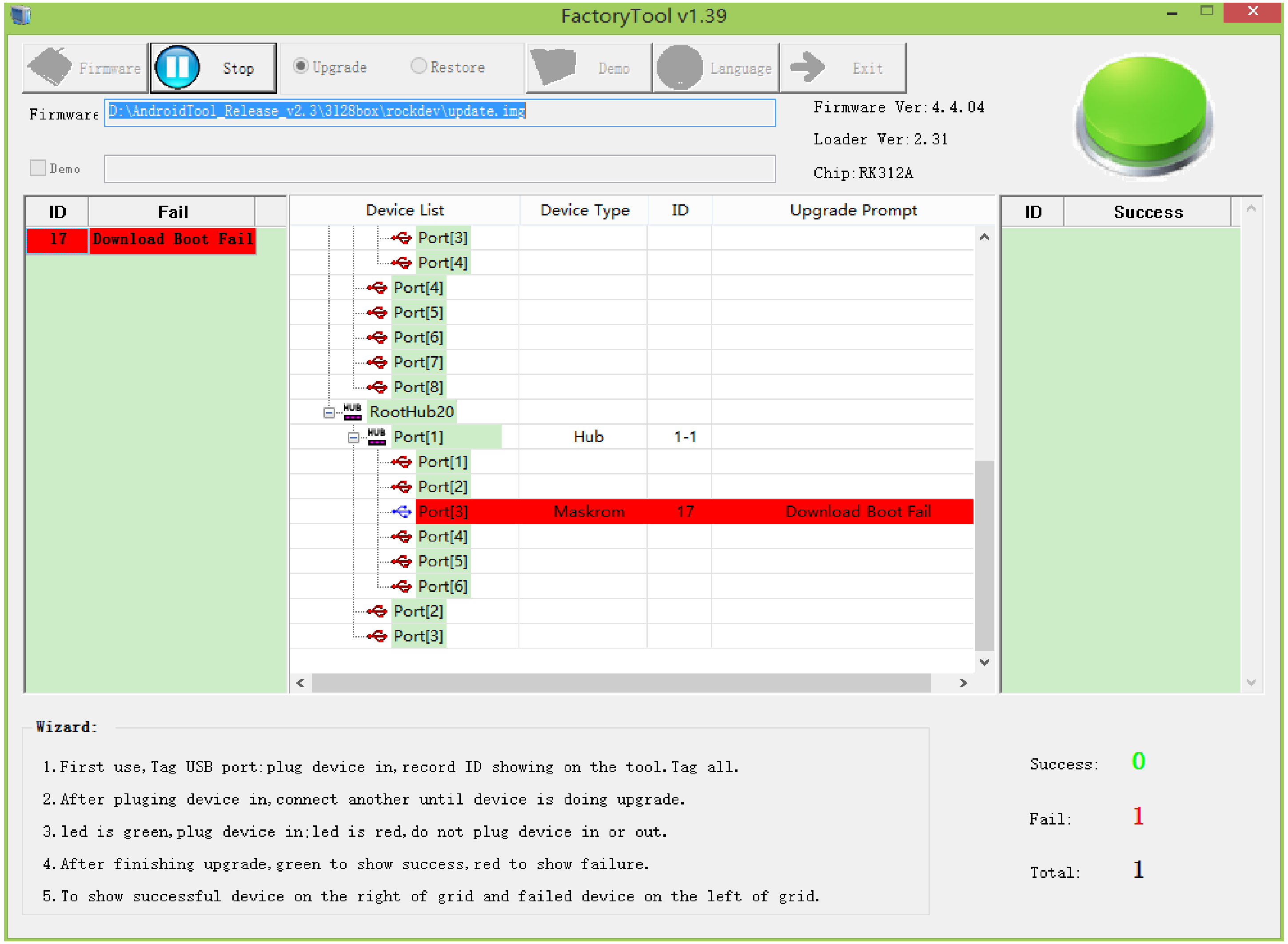
FW Ver: 4.4.02 FW Time: 2014-11-04 12:13:42 Support Chip: RK32

BOOT Ver: 2.17 BOOT Time: 2014-11-04 12:13:15

```
<Layer 1-1-3> Check Chip Success
<Layer 1-1-3> Get FlashInfo Start
<Layer 1-1-3> Get FlashInfo Success
<Layer 1-1-3> Prepare IDB Start
<Layer 1-1-3> Prepare IDB Success
<Layer 1-1-3> Download IDB Start
<Layer 1-1-3> Download IDB Success
<Layer 1-1-3> Reset Device Start
<Layer 1-1-3> Reset Device Success
<Layer 1-1-3> Wait For Loader Start
<Layer 1-1-3> Wait For Loader Fail
**********Upgrade Done Success<0> Fail<1> Time<32172>ms**********
```

1-1-3

Other SOC will fail at‘Download Boot’:



Firmware D:\AndroidTool_Release_v2.3\3128box\rockdev\update.img

☐ Demo

Firmware Ver:4.4.04

Loader Ver:2.31

Chip:RK312A

ID	Fail
17	Download Boot Fail

Device List	Device Type	ID	Upgrade Prompt
Port[3]			
Port[4]			
Port[4]			
Port[5]			
Port[6]			
Port[7]			
Port[8]			
RootHub20			
Port[1]	Hub	1-1	
Port[1]			
Port[2]			
Port[3]	Maskrom	17	Download Boot Fail
Port[4]			
Port[5]			
Port[6]			
Port[2]			
Port[3]			

ID	Success

Wizard:

1.First use,Tag USB port:plug device in,record ID showing on the tool.Tag all.

2.After pluging device in,connect another until device is doing upgrade.

3.led is green,plug device in;led is red,do not plug device in or out.

4.After finishing upgrade,green to show success,red to show failure.

5.To show successful device on the right of grid and failed device on the left of grid.

Success: 0

Fail: 1

Total: 1