MT7986 Secure Boot Quick Start Guide

2022/1/19

Version History

Version	Date	Author (Optional)	Description
0.1	2021-8-13	Alvin Kuo	Initial draft
1.0	2021-10-28	Micheal Su	Official release
1.1	2022-1-19	Micheal Su	Update how to get mbedtls-mbedtls-2.24.0 source code

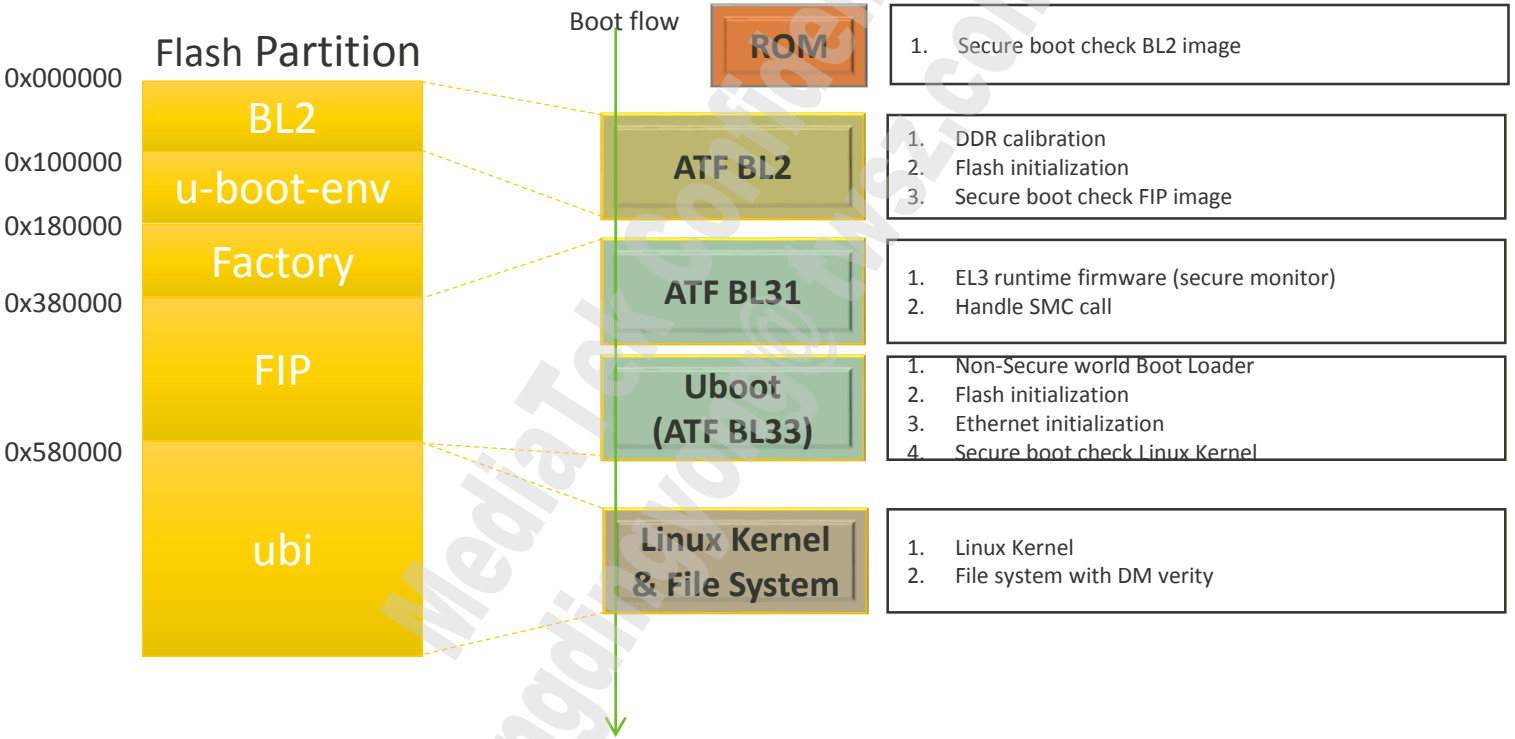
Glossary

- ATF: Arm Trusted Firmware, include below stage:
  - BL1: Boot Loader stage 1 (BL1) AP Trusted ROM
  - BL2: Boot Loader stage 2 (BL2) Trusted Boot Firmware (ex: preloader, SPL)
  - BL32: Boot Loader stage 3-1 (BL3-1) Secure Monitor
  - BL33: Boot Loader stage 3-3 (BL3-3) Non-trusted Firmware (ex: Uboot)
- BL: Boot Loader
- COT: Chain of Trust
- EL: Exception Level
- FIP: Firmware Image Package
- FIT: Flattened Image Tree
- ROT: Root of Trust
- SMC: Secure Monitor Call
- TEE: Trusted Execution Environment

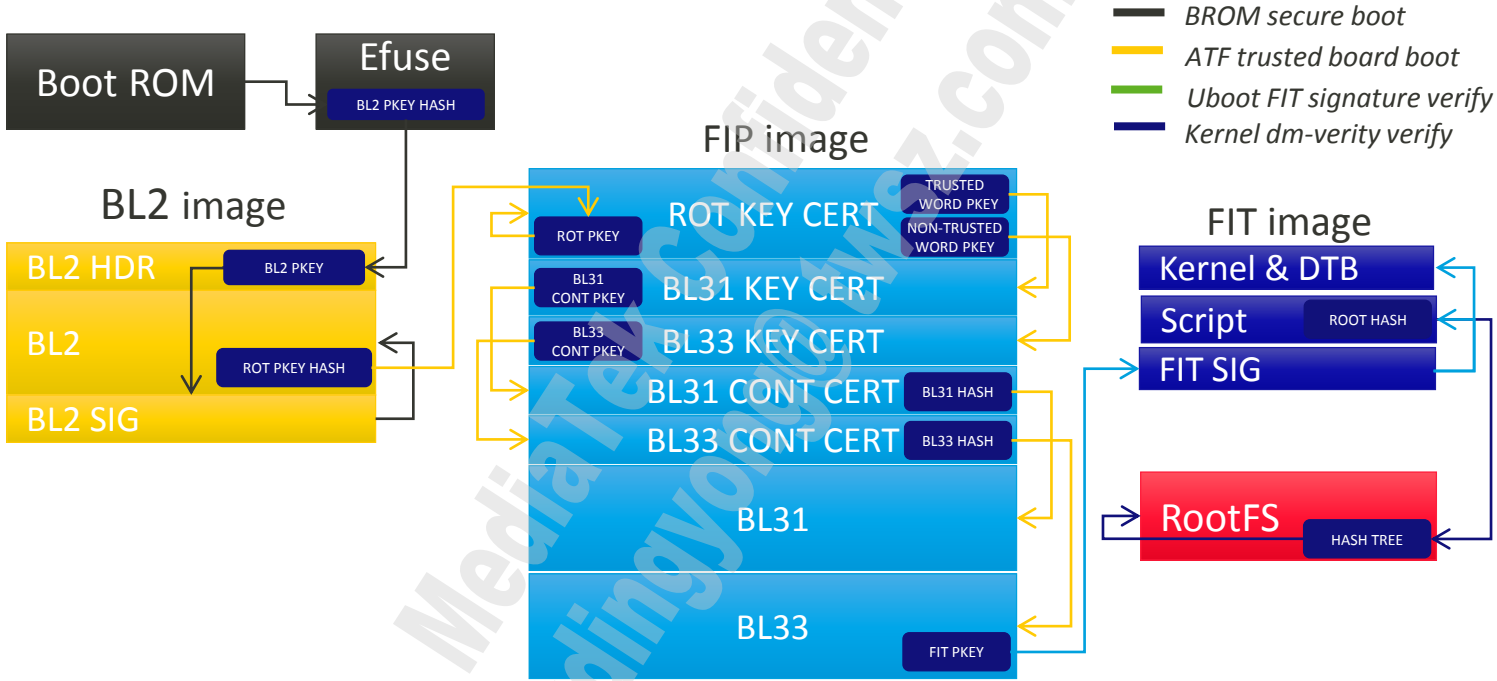
Secure Boot SDK Package

- **Secure boot SDK**
  - **ATF**
    - arf/
  - **Uboot-Upstream**
    - Uboot-upstream/
  - **OpenWRT**
    - openwrt/
  - **Tools**
    - tools/mbedtls-mbedtls-2.24.0/
  - **Document**
    - MT7986 Secure boot Quick Start Guide

Secure Boot Introduction



Chain of Trust Flow



To learn more about Authentication Framework & Chain of Trust, check out below link:
https://github.com/ARM-software/arm-trusted-firmware/blob/master/docs/design/auth-framework.rst

Host machine setup -- toolchain

- Please reference below link to install toolchain
- https://ubuntu.pkgs.org/18.04/ubuntu-main-amd64/gcc-aarch64-linux-gnu_7.3.0-3ubuntu2_amd64.deb.html
- After install complete, check the toolchain version below:

```
user@server:~/SecureBoot/pre-release_for_eth/Uboot-upstream$ /usr/bin/aarch64-linux-gnu-gcc --version
aarch64-linux-gnu-gcc (Ubuntu/Linaro 7.4.0-1ubuntu1~18.04.1) 7.4.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Download

Type	URL
Mirror	archive.ubuntu.com
Binary Package	gcc-aarch64-linux-gnu_7.
Source Package	gcc-defaults

Install Howto

1. Update the package index:
   `# sudo apt-get update`
2. Install gcc-aarch64-linux-gnu deb package:
   `# sudo apt-get install gcc-aarch64-linux-gnu`

Files

Path
/usr/bin/aarch64-linux-gnu-gcc

Compile Environment Preparation

- **Setup environment in current working directory**
  - **ATF/Uboot-upstream**
    - tar -Jxvf atf.tar.xz
    - tar -Jxvf Uboot-upstream.tar.xz
- **Openwrt**
  - **git clone --branch openwrt-21.02 https://git.openwrt.org/openwrt/openwrt.git**
  - **tar -Jxvf mtk-wifi-mt7986.tar.xz**
  - **cp -rf mtk-wifi-mt7986/* openwrt/**
  - **cd openwrt/**
  - **echo "src-git mtk_openwrt_feed https://git01.mediatek.com/openwrt/feeds/mtk-openwrt-feeds" >> feeds.conf.default**

Compile Environment Preparation

- mbedtls-mbedtls-2.24.0
  - Get mbedtls-mbedtls-2.24.0 from https://github.com/ARMmbed/mbedtls/releases/tag/v2.24.0 (Note: get newest version by git clone https://github.com/ARMmbed/mbedtls.git)
  - cd ./../
  - mkdir tools
  - cd tools
  - tar -xvf mbedtls-mbedtls-2.24.0.tar.gz
- sign key (reference “Host Machine Generate Keys” page)
  - cd ./../
  - mkdir keys
  - cd keys
  - openssl genrsa -out bl2_private_key.pem 2048
  - openssl genrsa -out fip_private_key.pem 2048
  - openssl genrsa -F4 -out fit_key.key 2048
  - openssl req -batch -new -x509 -key fit_key.key -out fit_key.crt

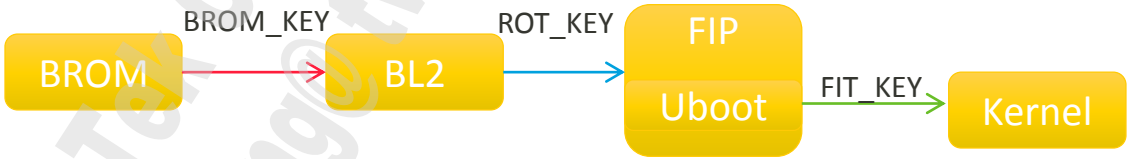
Compile Environment Preparation

- **The working directory should include below folders after setup environment**

```
atf  keys  openwrt  tools  Uboot-upstream
```

Host Machine Generate Keys

- **There are 3 keys for secure boot**
  - **BROM_KEY was used by BROM to verify BL2 image.**
  - **ROT_KEY was used by BL2 to verify FIP Image**
  - **FIT_KEY was used by Uboot to verify FIT Image**
- **How to generate keys (using openssl)**
  - mkdir keys
  - cd keys
  - **BROM_KEY (private)**
    - openssl genrsa -out bl2_private_key.pem 2048
  - **ROT_KEY (private)**
    - openssl genrsa -out fip_private_key.pem 2048
  - **FIT_KEY (private and public)**
    - openssl genrsa -F4 -out fit_key.key 2048
    - openssl req -batch -new -x509 -key fit_key.key -out fit_key.crt



How to Compile

- Compile Uboot-upstream
  - cd Uboot-upstream/
  - export CROSS_COMPILE=/usr/bin/aarch64-linux-gnu-
  - make mt7986_spim_nand_sb_rfb_defconfig
  - make V=s FIT_KEY=./../keys/fit_key.crt
  - # u-boot.bin will be created under Uboot-upstream folder.
- Compile FIP image
  - cd atf
  - make distclean
  - export CROSS_COMPILE=/usr/bin/aarch64-linux-gnu-
  - make PLAT=mt7986 BL33=../Uboot-upstream/u-boot.bin BOOT_DEVICE=spim-nand NAND_TYPE=spim:2k+64 NMBM=1 DRAM_USE_DDR4=0 MBEDTLS_DIR=../tools/mbedtls-mbedtls-2.24.0/ TRUSTED_BOARD_BOOT=1 GENERATE_COT=1 ROT_KEY=../keys/fip_private_key.pem BROM_SIGN_KEY=../keys/bl2_private_key.pem all fip
  - # The BL2 image is located in atf/build/mt7986/release/bl2.img
  - # The hash of BROM_KEY is located in atf/build/mt7986/release/bl2.img.signkeyhash
  - # The FIP image is located in atf/build/mt7986/release/fip.bin

How to Compile

- **Compile OpenWRT**
  - **./autobuild/mt7986-AX6000-sb/lede-branch-build-sanity.sh**
  - **# The openwrt image is located in openwrt/lede/bin/targets/mediatek/mt7986/openwrt-mediatek-mt7986-mt7986a-ax6000-snand-rfb-sb-squashfs-sysupgrade.bin**

```
define Device/mt7986a-ax6000-snand-rfb-sb
  DEVICE_VENDOR := MediaTek
  DEVICE_MODEL := mt7986a-ax6000-snand-rfb (SPI-NAND,UBI)
  DEVICE_DTS := mt7986a-snand-rfb
  DEVICE_DTS_DIR := $(DTS_DIR)/mediatek
  SUPPORTED_DEVICES := mediatek,mt7986a-snand-rfb
  UBINIZE_OPTS := -E 5
  BLOCKSIZE := 128k
  PAGESIZE := 2048
  IMAGE_SIZE := 65536k
  IMAGE_ROOTFS := $(KDIR)/root.squashfs-hashed-$$(DEVICE_NAME)
  KERNEL_IN_UBI := 1
  IMAGES := factory.bin
  FIT_KEY_DIR := $(TOPDIR)/../keys
  FIT_KEY_NAME := fit_key
  ANTI_ROLLBACK_TABLE := $(TOPDIR)/../../fw_ar_table.xml
  AUTO_AR_CONF := $(TOPDIR)/../../auto_ar_conf.mk
  HASHED_BOOT_DEVICE := /dev/ubiblock0_1
  BASIC_KERNEL_CMDLINE := console=ttyS0,115200n1 rootfstype=squashfs loglevel=8
  KERNEL = kernel-bin | lzma | squashfs-hashed | fw-ar-ver | \
        fit-sign lzma $$(KDIR)/image-$$(firstword $$(DEVICE_DTS)).dtb
  KERNEL_INITRAMFS =
  IMAGE/factory.bin := append-ubi | check-size $$$$(IMAGE_SIZE)
  IMAGE/sysupgrade.bin := sysupgrade-tar | append-metadata
endef
TARGET_DEVICES += mt7986a-ax6000-snand-rfb-sb
DEFAULT_DEVICE_VARS += IMAGE_ROOTFS FIT_KEY_DIR FIT_KEY_NAME ANTI_ROLLBACK_TABLE \
        AUTO_AR_CONF HASHED_BOOT_DEVICE BASIC_KERNEL_CMDLINE
```

Note: when compile openwrt, it will try to find FIT_KEY at "../keys/fit_key.key", if "../keys/fit_key.key" did not exist, then openwrt-mediatek-mt7986-mt7986a-ax6000-snand-rfb-sb-squashfs-sysupgrade.bin will not be created.
You can check related path on openwrt/target/linux/mediatek/image/mt7986.mk

How to Compile

- **You will have below files after all compile done.**
  - **# The BL2 in atf/build/mt7986/release/bl2.img**
  - **# The hash of BROM_KEY in atf/build/mt7986/release/bl2.img.signkeyhash**
  - **# The FIP in atf/build/mt7986/release/fip.bin**
  - **# The Firmware (Linux & File System) in openwrt/lede/bin/targets/mediatek/mt7986/openwrt-mediatek-mt7986-mt7986a-ax6000-snand-rfb-sb-squashfs-sysupgrade.bin**

Upgrade Image from Uboot Menu

- **Upgrade BL2 via U-boot menu**

```
*** U-Boot Boot Menu ***

   1. Startup system (Default)
   2. Upgrade firmware
   3. Upgrade ATF BL2
   4. Upgrade ATF FIP
   5. Upgrade single image
   6. Load image
   0. U-Boot console


Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

```
*** Upgrading ATF BL2 ***

Available load methods:
    0 - TFTP client (Default)
    1 - Xmodem
    2 - Ymodem
    3 - Kermit
    4 - S-Record

Select (enter for default):

Input U-Boot's IP address: 192.168.1.1
Input TFTP server's IP address: 192.168.1.2
Input IP netmask: 255.255.255.0
Input file name: bl2.img

Using ethernet@15100000 device
TFTP from server 192.168.1.2; our IP address is 192.168.1.1
Filename 'bl2.img'.
Load address: 0x46000000
Loading: #################
         801.8 KiB/s
done
Bytes transferred = 234024 (39228 hex)
Saving Environment to MTD... Erasing on MTD device 'nmbm0'... OK
Writing to MTD device 'nmbm0'... OK
OK

*** Loaded 234024 (0x39228) bytes at 0x46000000 ***

Erasing from 0x0 to 0x3ffff, size 0x40000 ... OK
Writing from 0x46000000 to 0x0, size 0x39228 ... OK
Verifying from 0x0 to 0x39227, size 0x39228 ... OK

*** ATF BL2 upgrade completed! ***
MT7986>
```

Upgrade Image from Uboot Menu

- **Upgrade FIP via U-boot menu**

```
*** U-Boot Boot Menu ***

   1. Startup system (Default)
   2. Upgrade firmware
   3. Upgrade ATF BL2
   4. Upgrade ATF FIP
   5. Upgrade single image
   6. Load image
   0. U-Boot console

Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

```
*** Upgrading ATF FIP ***

Available load methods:
    0 - TFTP client (Default)
    1 - Xmodem
    2 - Ymodem
    3 - Kermit
    4 - S-Record

Select (enter for default):

Input U-Boot's IP address: 192.168.1.1
Input TFTP server's IP address: 192.168.1.2
Input IP netmask: 255.255.255.0
Input file name: fip.bin

Using ethernet@15100000 device
TFTP from server 192.168.1.2; our IP address is 192.168.1.1
Filename 'fip.bin'.
Load address: 0x46000000
Loading: #################################################
         870.1 KiB/s
done
Bytes transferred = 723721 (b0b09 hex)
Saving Environment to MTD... Erasing on MTD device 'nmbm0'... OK
Writing to MTD device 'nmbm0'... OK
OK

*** Loaded 723721 (0xb0b09) bytes at 0x46000000 ***

Erasing from 0x0 to 0xbffff, size 0xc0000 ... OK
Writing from 0x46000000 to 0x0, size 0xb0b09 ... OK
Verifying from 0x0 to 0xb0b08, size 0xb0b09 ... OK

*** ATF FIP upgrade completed! ***

Erasing environment from 0x100000 to 0x11ffff, size 0x20000 ... OK
MT7986>
```

Upgrade Image from Uboot Menu

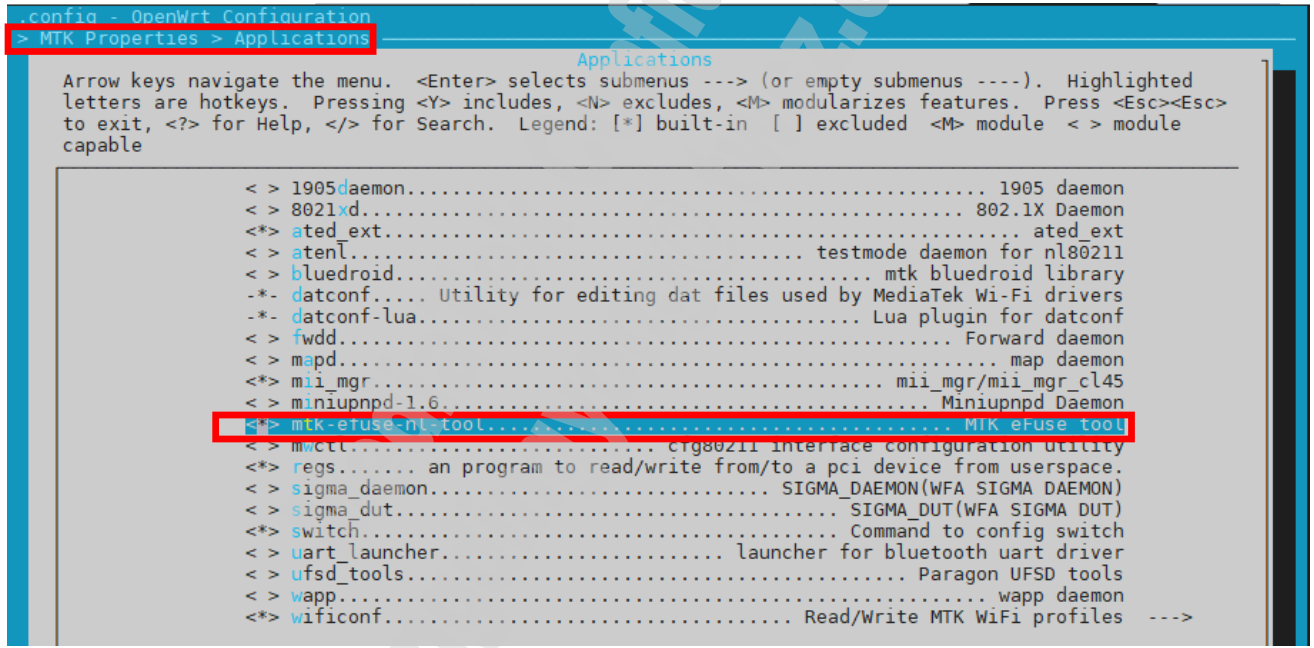
- **Upgrade FW via U-boot menu**

```
*** U-Boot Boot Menu ***

   1. Startup system (Default)
   2. Upgrade firmware
   3. Upgrade ATF BL2
   4. Upgrade ATF FIP
   5. Upgrade single image
   6. Load image
   0. U-Boot console


Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

```
*** Upgrading Firmware ***

Run image after upgrading? (Y/n): n

Available load methods:
   0 - TFTP client (Default)
   1 - Xmodem
   2 - Ymodem
   3 - Kermit
   4 - S-Record

Select (enter for default):

Input U-Boot's IP address: 192.168.1.1
Input TFTP server's IP address: 192.168.1.2
Input IP netmask: 255.255.255.0
Input file name: openwrt-mediatek-mt7986-mt7986a-ax6000-snand-rfb-sb-squashfs-sysupgrade.bin

Using ethernet@15100000 device
TFTP from server 192.168.1.2; our IP address is 192.168.1.1
Filename 'openwrt-mediatek-mt7986-mt7986a-ax6000-snand-rfb-sb-squashfs-sysupgrade.bin'.
Load address: 0x4007ff28
Loading: ###################################################################
         ###################################################################
         ###################################################################
         ###################################################################
         ###################################################################
         ###################################################################
         ###################################################################
         ###################################################################
         ###################################################################
         ###################################################################
         ###################################################################
         ###################################################################
         ###################################################################
         ###################################################################
         ###################################################################
         ###################################################################
         ###################################################################
         ###########################################
         751 KiB/s
done
Bytes transferred = 15862584 (f20b38 hex)
Saving Environment to nowhere... not possible

*** Loaded 15862584 (0xf20b38) bytes at 0x4007ff28 ***

spi-nand: spi_nand spi_nand@1: Macronix SPI NAND was found.
spi-nand: spi_nand spi_nand@1: 128 MiB, block size: 128 KiB, page size: 2048, OOB size: 64
ubi0: attaching mtd5
ubi0: scanning is finished
ubi0: attached mtd5 (name "ubi", size 64 MiB)
ubi0: PEB size: 131072 bytes (128 KiB), LEB size: 126976 bytes
ubi0: min./max. I/O unit sizes: 2048/2048, sub-page size 2048
```

How to Write Efuse

- **Build efuse tool in OpenWRT**
  - **make menuconfig --> MTK Properties --> Applications**



How to Write Efuse

- You should find mtk-efuse-tool at rootfs. (usr/sbin/mtk-efuse-tool)
- Put hash of BROM_KEY - bl2.img.signkeyhash in rootfs.
- Below are some example to write hash of BROM_KEY and enable secure boot
  - Note: Make sure hash of BROM_KEY is correct before enable secure boot

```
Example:
 @For secure boot:
   mtk-efuse-tool wh 0 bl2.img.signkeyhash
   mtk-efuse-tool rh 0
   mtk-efuse-tool lh 0
   mtk-efuse-tool es
   mtk-efuse-tool dj
   mtk-efuse-tool db
```

#1. Write hash of BROM_KEY into key hash index 0 and read it back to check its content if correct

```
tftp -g -r bl2.img.signkeyhash 192.168.1.3
root@LEDE:/# mtk-efuse-tool wh 0 bl2.img.signkeyhash

efuse operate (wh) success
root@LEDE:/# mtk-efuse-tool rh 0
PUBK0_HASH :
00000000  77 7d d3 21 21 f8 40 fb  70 0b bf 7b b7 b5 ff 49
00000010  ea 98 f6 ae 60 03 92 c2  71 22 85 8e 48 6e 99 95

efuse operate (rh) success
```

WARING!!

Make sure enable secure boot is the last step, once it is enabled, it can not be disabled.

#2. Lock key hash index 0

```
root@LEDE:/# mtk-efuse-tool lh 0

efuse operate (lh) success
```

#3. Enable secure boot

```
root@LEDE:/# mtk-efuse-tool es

efuse operate (es) success
```

Tips:

1. You can also disable JTAG by efuse, enter "*mtk-efuse-tool*" without any parameter will show the help.

Application Note for eFuse Tool

- **mtk-efuse-nl-drv**
  - driver for receiving netlink commands and issue corresponding SMC to BL31 for R/W eFuse
- **mtk-efuse-nl-tool**
  - tool for interacting with user and issue netlink commands to mtk-efuse-nl-drv
  - eg : write key hash, lock key hash, enable secure boot, disable JTAG
- **For using Secure Boot customer, you will write key hash and enable secure boot in production line, so you need to build mtk-efuse-nl-drv + mtk-efuse-nl-tool in your production FW.**
- ***DO NOT build efuse tool and driver into your normal FW which you will release to end user, because the eFuse tool is very powerful. The eFuse tool is only for testing and producing !**

Enable Secure Boot Check

- If BL2 and FIP is signed properly, Uboot menu will show up
- If BL2 is signed with wrong key ; If BL2 is not signed

```
U-Boot> reset
resetting ...
#
F0: 102B 0000
F1: 5000 1006
F6: 0000 0000
V0: 706D 0000 [0001]
00: 1017 0000
F6: 0000 0000
V0: 706D 0000 [0001]
01: 102A 0001
02: 1017 0000
BP: 0000 02C0 [0001]
T0: 0000 027F [000F]
System halt!
```

```
resetting ...
#
F0: 102B 0000
F1: 5000 1006
F6: 0000 0000
V0: 100C 0000 [0001]
00: 1017 0000
F6: 0000 0000
V0: 100C 0000 [0001]
01: 102A 0001
02: 1017 0000
BP: 0000 02C0 [0001]
T0: 0000 0241 [000F]
System halt!
```

```
*** U-Boot Boot Menu ***

   1. Startup system (Default)
   2. Upgrade firmware
   3. Upgrade ATF BL2
   4. Upgrade ATF FIP
   5. Upgrade single image
   6. Load image
   0. U-Boot console
```

V0:100C, INVALID_SIG_TYPE
V0:706D, KEY_MISMATCH
00:1017, BL_VERIFY_FAILED

Enable Secure Boot Check

- If FW is signed properly, Linux can boot

```
  Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
Reading from 0x400000 to 0x42007f1c, size 0x800 ... OK
Reading from 0x400000 to 0x42007f1c, size 0x1b2e48 ... OK
## Loading kernel from FIT Image at 42007f1c ...
   Using 'config@1' configuration
   Verifying Hash Integrity ... sha1,rsa2048:fit_key+ OK
   Trying 'kernel@1' kernel subimage
     Description:  ARM OpenWrt Linux-4.4.241
     Type:         Kernel Image
```

- If using an FW with wrong key or not signed, Linux will boot fail

```
## Loading kernel from FIT Image at 42007f1c ...
   Using 'config@1' configuration
   Verifying Hash Integrity ...  error!
No 'signature' subnode found for '<NULL>' hash node in 'config@1' config node
Failed to verify required signature 'key-fit_key'
Bad Data Hash
ERROR: can't get kernel image!
U-Boot>
```

Secure Boot MP Notice – (1)

- **Disable JTAG and BROM CMD to avoid hacker using ICE or Flashtool to break COT**
  - **mtk-efuse-tool db**
  - **mtk-efuse-tool dj**

```
[Original]

F0: 102B 0000
FA: 1040 0000
FA: 1040 0000 [0200]
F9: 0000 0000
V0: 0000 0000 [0001]
00: 0000 0000
BP: 2400 0041 [0000]
G0: 1190 0000
EC: 0000 0000 [1000]
T0: 0000 09F7 [010F]
Jump to BL
```

```
[Disable BROM CMD]

F0: 102B 0000
FA: 1040 0000
FA: 1040 0000 [0200]
F9: 0000 0000
V0: 0000 0000 [0001]
00: 0007 8000
01: 0000 0000
BP: 2400 0001 [0000]
G0: 1190 0000
EC: 0000 0000 [1000]
T0: 0000 0C59 [010F]
Jump to BL
```

```
[Disable BROM CMD & JTAG]

F0: 102B 0000
FA: 1040 0000
FA: 1040 0000 [0200]
F9: 0000 0000
V0: 0000 0000 [0001]
00: 0007 8000
01: 0000 0000
BP: 2400 0209 [0000]
G0: 1190 0000
EC: 0000 0000 [1000]
T0: 0000 01EF [010F]
Jump to BL
```

Secure Boot MP Notice – (1)

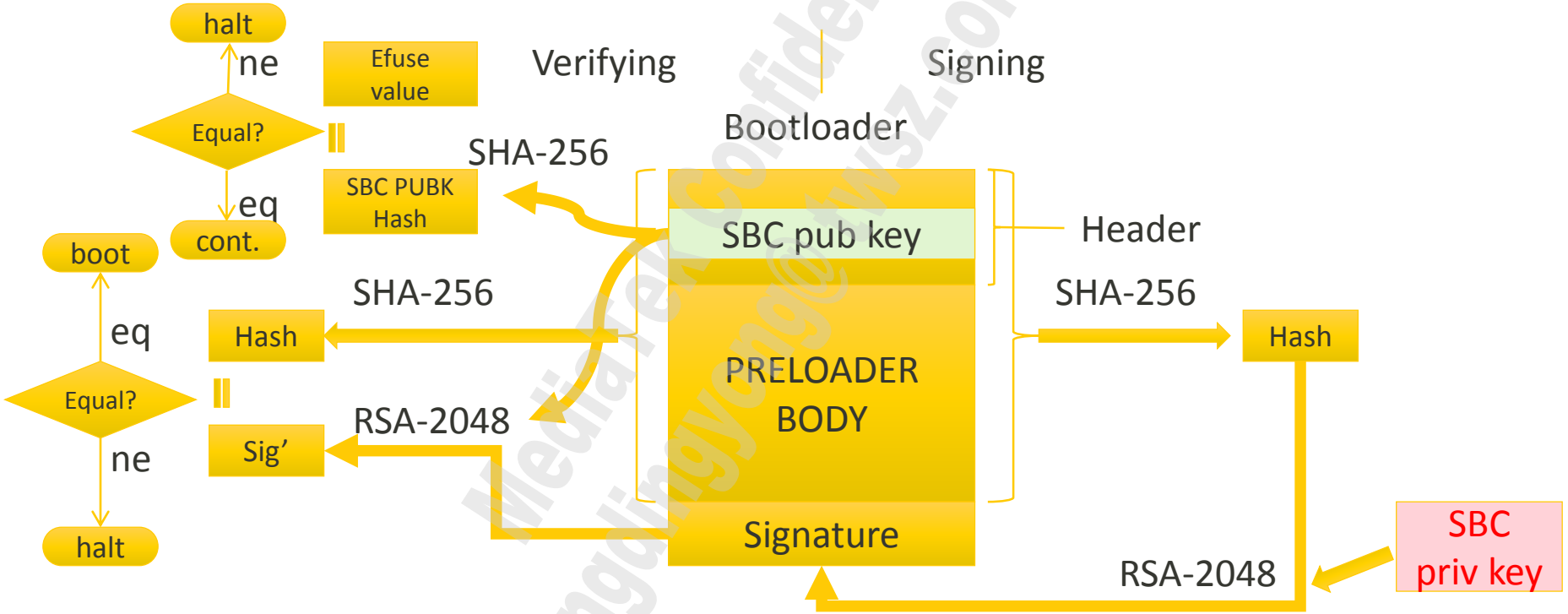
- **Disable CMD and Ctrl+C in U-Boot to avoid hacker using uboot cmd to break COT you can do it by your self, or enable bootsecure**

```
bootsecure will directly run bootcmd, disable Ctrl+C and console. If any thing wrong, the board will hang.
@ configs/mt7986_spim_nand_sb_rfb_defconfig
+ CONFIG_AUTOBOOT_MENU_SHOW is not set

@ arch/arm/dts/mt7986a-rfb.dts
/ {
+               config {
+                               bootcmd = "mtkboardboot";
+                               bootsecure = <1>;
+               };
};
```

Secure Boot MP Notice – (2)

- Don’t build-in mtk-eFuse-nl-tool and mtk-efuse-nl-drv in your MP FW, or hacker may use this powerful tool to attack the device
- U-boot env was disabled default, DO NOT ENABLE it or fit signature verifying may bypass via setup verify=0

Appendix - Secure Boot Sign/Verify Flow (1)



Appendix - Secure Boot Sign/Verify Flow (2)