PERFECT WIRELESS EXPERIENCE

FG132

Secure Boot Application Guide

V1.0

Disclaimer

Any action you take in the course of using this document is at your own risk, and Fibocom shall not be liable for any damages or losses under any circumstances. Due to product version upgrade or other reasons, Fibocom reserves the right to modify any information in this document at any time without prior notice and any responsibility. Unless otherwise agreed, all statements, information and suggestions in this document do not constitute any express or implied guarantee.

This document may include the third-party information covering products, services, software, data, and so on. Fibocom does not control and assumes no responsibility for the third-party content, including but not limited to the accuracy, compatibility, reliability, availability, legitimacy, appropriateness, performance, non-infringement, and status update, unless otherwise specified in this document. Fibocom does not provide any guarantee or authorization for the third-party content mentioned or referenced in this document. If you need a third-party license, obtain it in an authorized or legal way, unless otherwise specified in this document.

Copyright Notice

Unless specially authorized by Fibocom, the recipient of the documents shall keep the documents and information received confidential, and shall not use them for any purpose other than the implementation and development of this project. Without the written permission of Fibocom, no unit or individual shall extract or copy part or all of the contents of this document without authorization, or transmit them in any form. Fibocom has the right to investigate legal liabilities for any offense and tort in connection with violation of confidentiality obligations, or unauthorized use or malicious use of the said documents and information in other illegal forms.

Trademark Statement

FIBOCOM The trademark is registered and owned by Fibocom Wireless Inc.

Other trademarks, product names, service names and company names appearing in this document are owned by their respective owners.

Contact Information

Website: https://www.fibocom.com

Address: 10/F-14/F, Block A, Building 6, Shenzhen International Innovation Valley, Dashi First Road, Xili Community, Xili Subdistrict, Nanshan District, Shenzhen

Tel: 0755-26733555

Contents

Change History

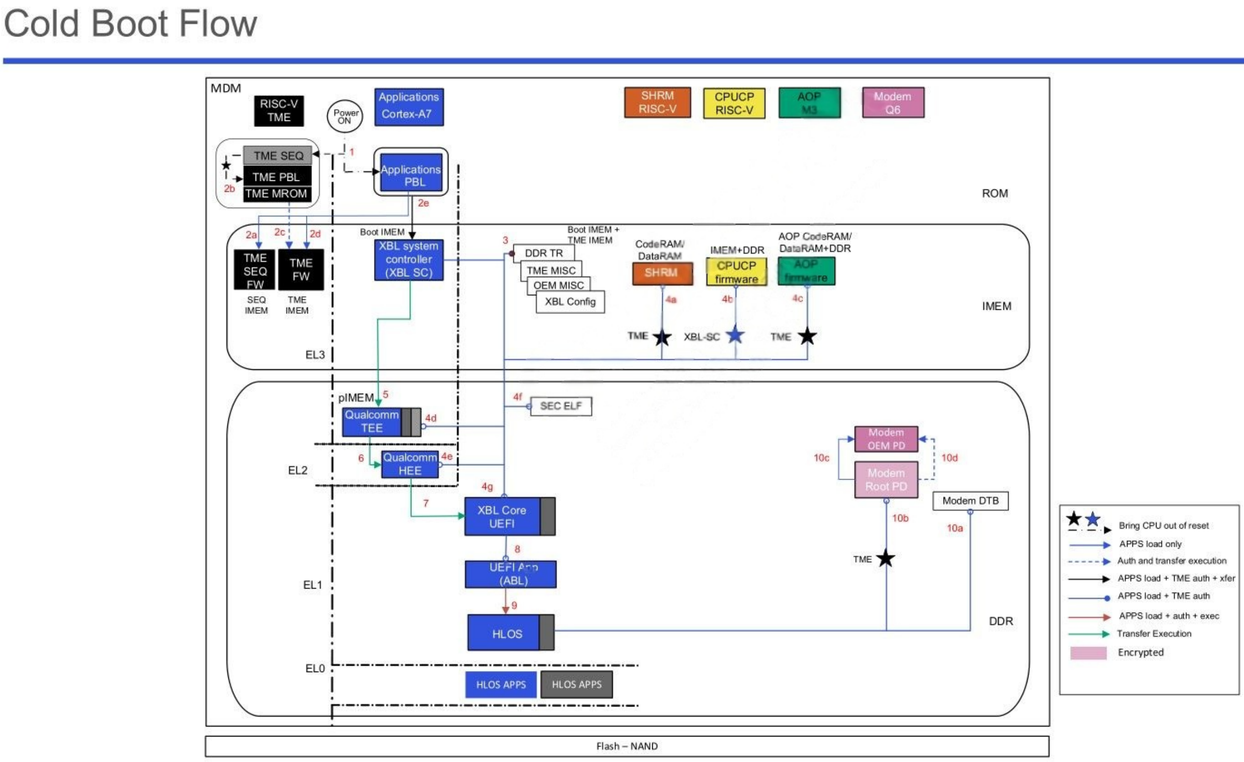
V1.0 (2024-08-01)	Initial version.

Applicable Model

No.	Applicable Model	Description
1	FG132	--

1 Introduction

1.1 Firmware Boot Process



1.2 Enable Secure Boot

1. Generate private key and certificate.
2. Sign image.
3. Make configuration file sec.elf.
4. Burn firmware.
5. Restart the module.

2 Make Certificate Chain

2.1 Environment Preparation

- **Environment**: Ubuntu 22.04, Python 2.7, OpenSSL 3.0.2 or later.
- **Obtain signature environment**: Refer to *Fibocom_FG132_OPENSDK Compilation Environment Building Guide* to obtain the SDK.

```
[redacted]@ubuntu:~/code/fg132/fibo_open_sdk$ ls
build_abl_image.sh     build_genCerts.sh           build_recovery_apps_proc.sh  owrt_workspace
build_apps_proc.sh     build_genSecelf.sh          build.sh                     projects
build_boot_images.sh   build_ota_target_files.sh   build_signImage.sh           tools
build_common.sh        build_partition.sh          common
[redacted]ubuntu:~/code/fg132/fibo_open_sdk$
```

Where:

- ◦ **build_genCerts.sh** generates the certificate chain.
- ◦ **build_signImage.sh** signs firmware.
- ◦ **build_genSecelf.sh** generates the **sec.elf** file.

- Obtain the firmware package of the corresponding version: Obtain the firmware package from Fibocom SPM or Fibocom technical support.

2.2 Generate Certificate

Execute the **./build_genCerts.sh** script, and the system will generate new certificates in the **common/secboot_keys/OEM-KEYS/** directory, as shown in the figure below.

```
[redacted]@ubuntu:~/code/fg132/fibo_open_sdk$ ./build_genCerts.sh
Certificate request self-signature ok
subject=C = US, ST = California, CN = Generated OEM Attestation CA, O = OEM, L = San Diego
============================================================
 ./build_genCerts.sh creat keys end!
============================================================
[redacted]@ubuntu:~/code/fg132/fibo_open_sdk$ ls  common/secboot_keys/OEM-KEYS/
attestca.CSR      qpsa_attestca.cer  qpsa_rootca.key      v3_attest.ext
attestca_pem.crt  qpsa_attestca.key  rootca_pem.crt       v3.ext
opensslroot.cfg   qpsa_rootca.cer    sha384rootcert.txt
[redacted]@ubuntu:~/code/fg132/fibo_open_sdk$
```

3 Sign Image

3.1 Obtain Firmware Package To Be Signed

Obtain the firmware package that needs to be signed from Fibocom SPM or Fibocom technical support.

```
[redacted]@ubuntu:~/code/fg132/fibo_open_sdk$ ls  ../Unsigned
abl.elf          fw_ipa_gsi_5.2_le.elf  qupv3fw.elf                   uefi.elf
aop_devcfg.mbn   img                    qwes.mbn                      Unsigned.rar
aop.mbn          km41_32.mbn            sequencer_ram.elf             Ver_Info.txt
build_flavor.txt modemr.jsn             shrm.elf                      xbl_config_devprg.elf
cmnlib.mbn       multi_oem.mbn          signed_firmware_soc_view.elf  xbl_config.elf
contents.xml     my_ubi.ini             smplap32.mbn                  xbl_ramdump.elf
cpucp.elf        prog_firehose_ddr.elf  tz.mbn                        xbl_sc.elf
devcfg.mbn       qdsp6sw.mbn            tzsc.mbn
[redacted]@ubuntu:~/code/fg132/fibo_open_sdk$
```

For details of each file, see the overview of the partition table in *Fibocom_FG132_OpenSDK Partition Adjustment Guide*.

3.2 Copy Customer-compiled Firmware Package To Be Signed

Before executing the signing command, copy the **abl.elf** file compiled by the customer into the firmware package that needs to be signed. Otherwise, the image will not be signed. You can skip this step if the **abl.elf** file has not changed.

3.3 Run Signing Command

Run the **./build_signImage.sh ../Unsigned 2048** command.

- The first parameter is the unsigned directory path.
- The second parameter is the page size of the flash. Generally, the page size of 2+2 MCP is 2048, and that of 4+2 MCP is 4096. It changes with customer project configuration.

```
[redacted]@ubuntu:~/code/fg132/fibo_open_sdk$ ./build_signImage.sh ../Unsigned 2048
 unsigned_dir:../Unsigned
 signed_dir:../Unsigned/Out/Signed
 page_size:2048
============================================
 ./build_signImage.sh sign all image
============================================
signing file is ../Unsigned/abl.elf, id is ABL
signing file is ../Unsigned/aop.mbn, id is AOP
```

```
ubinize: sub-page size:             2048
ubinize: VID offset:                2048
ubinize: data offset:               4096
ubinize: UBI image sequence number: 1581245895
ubinize: loaded the ini-file "../Unsigned/my_ubi.ini"
ubinize: count of sections: 1

ubinize: parsing section "ubifs"
ubinize: mode=ubi, keep parsing
ubinize: volume type: dynamic
ubinize: volume ID: 0
ubinize: volume size was not specified in section "ubifs", assume minimum to fit image "N
.ubifs"40759296 bytes (38.8 MiB)
ubinize: volume name: modem
ubinize: volume alignment: 1
ubinize: autoresize flags found
ubinize: adding volume 0
ubinize: writing volume 0
ubinize: image file: NON-HLOS.ubifs

ubinize: writing layout volume
ubinize: done
============================================================
 package modem end!
============================================================
============================================================
 sign image success
============================================================
============================================================
 ./build_signImage.sh signed images end!
============================================================
[redacted]@ubuntu:~/code/fg132/fibo_open_sdk$
```

3.4 Generate Signature File

```
[redacted]@ubuntu:~/code/fg132/fibo_open_sdk$ ls  ../Unsigned/Out/Signed/
abl.elf                  prog_firehose_ddr.elf       xbl_config.elf
aop_devcfg.mbn           qdsp6sw.mbn                 xbl_ramdump.elf
aop.mbn                  qupv3fw.elf                 xbl_sc.elf
cmnlib.mbn               qwes.mbn                    xbl_s_devprg_ns.melf
cpucp.elf                shrm.elf                    xbl_s.hash
devcfg.mbn               signed_firmware_soc_view.elf xbl_s_hash.hd
fw_ipa_gsi_5.2_le.elf    smplap32.mbn                xbl_s.melf
img                      tz.mbn                      xbl_s_nand.melf
km41_32.mbn              tzsc.mbn                    xbl_s_phdr.pbn
multi_oem.mbn            uefi.elf                    xbl_s_preamble.mbn
NON-HLOS.ubi             xbl_config_devprg.elf
[redacted]@ubuntu:~/code/fg132/fibo_open_sdk$
```

4 Make Configuration File

Execute the **./build_genSecelf.sh** script, and the system will generate the configuration file **sec.elf** in the **Out/Signed/** directory, as shown in the figure below. This file is used to enable Secure Boot.

```
@ubuntu:~/code/fg132/fibo_open_sdk$ ls Out/Signed/
sec.elf
@ubuntu:~/code/fg132/fibo_open_sdk$ ./build_genSecelf.sh
==========================================================
 ./build_genSecelf.sh signed images end!
==========================================================
@ubuntu:~/code/fg132/fibo_open_sdk$ ls Out/Signed/
sec.elf
@ubuntu:~/code/fg132/fibo_open_sdk$
```

5 Secure Burn

5.1 Obtain Complete Firmware Package

Obtain the complete firmware package from Fibocom SPM or Fibocom technical support. The following figure shows the files in this package.

```
@ubuntu:~/code/fg132/fibo_open_sdk$ ls ../19003.1000.00.02.01.39_20240801_80000/Maincode/
abl.elf            cpucp.elf           fw_ipa_gsi_5.2_le.elf  shrm.elf
aop_devcfg.mbn     devcfg_low_ddr.mbn  km41_32.mbn            sysfs-2k.ubi
aop.mbn            devcfg.mbn          multi_oem.mbn          tz.mbn
apdp.mbn           efs2gld.bin         multi_qti.mbn          uefi.elf
boot.img           f_cust_devcfg.bin   NON-HLOS.ubi           xbl_config.elf
boot-recovery.img  f_cust_devcfg.xml   qupv3fw.elf            xbl_ramdump.elf
cefs.mbn           f_def_devcfg.bin    recoveryfs-2k.ubi      xbl_s_nand.melf
cmnlib.mbn         f_def_devcfg.xml    sec.elf
@ubuntu:~/code/fg132/fibo_open_sdk$
```

5.2 Assemble New Full Firmware Package

1. Use the signed firmware package files to replace files in the complete firmware package.

```
@ubuntu:~/code/fg132/fibo_open_sdk$ cp -r ../Unsigned/Out/Signed/. ../19003.1000.00.02.01
.39_20240801_80000/Maincode/
ubuntu:~/code/fg132/fibo_open_sdk$
```

2. Use the generated configuration file **sec.elf** to replace that in the complete firmware package. If you need to update the **sec.elf** file, you can overwrite the original one. If you do not need to update the file, copy the original configuration file to the complete firmware package.

```
g@ubuntu:~/code/fg132/fibo_open_sdk$ cp -r Out/Signed/sec.elf ../19003.1000.00.02.01.39_20
240801_80000/Maincode/
ubuntu:~/code/fg132/fibo_open_sdk$
```

3. Generate a new complete firmware package and then burn the new firmware package.

5.3 Burn Firmware Package

- Use the multi upgrade tool for burning

  Refer to *Fibocom_Qualcomm_SDX35_Multi Upgrade Tool User Guide_Windows* to complete the burning.

- Use the Linux full package tool for burning.

  Refer to Fibocom_Linux_Firmware_Upgrade.

- FOAT tool upgrade

  Refer to *Fibocom_Linux_FOAT_Upgrade*.

- FOTA upgrade using AT commands

  Refer to *Fibocom_MTC_Application Guide_FOTA*.

- FOTA upgrade using API

  Refer to *Fibocom_FG132_Open_API User Manual*.

6 FAQ

1. What is the enable status of the Secure Boot function in Fibocom?

By default, the Secure Boot function is turned off in Fibocom. Customers need to use their own certificate chain to sign the related files and then follow this document to enabled this function.

2. Does the certificate chain need to be executed every time?

No, it is not necessary if the private key and certificate are not changed. You only need to generate the certificate chain once. For subsequent firmware upgrade, you can follow Chapter 3 to re-sign the related files.

3. Does the sec.elf file need to be regenerated every time?

No, it is not necessary if the module model and root certificate are not changed. You can continue to use the **sec.elf** file of the previously correctly signed file for burning. Other signed **sec.elf** files will cause the system to fail to start.

4. Can unsigned firmware run on the module?

No.

5. After Secure Boot is enabled, can the original firmware be downloaded normally?

The correctly signed **prog_firehose_ddr.elf** file can be downloaded using the multi upgrade tool, Linux full package download tool, FOAT, or FOTA (subsequent downloads are not allowed).

It can also be downloaded using Fastboot. The open-source tool mtd-utils can also be used for writing.

Other methods are not supported.

6. After Secure Boot is enabled, how do I verify it?

The original incorrectly signed firmware package cannot be downloaded.

The module cannot start if some firmware is not signed.

7. What should I pay attention to when using Secure Boot?

Keep your private key and certificate safe to prevent any leakage.

Secure Boot has secure download features and you must keep the **prog_firehose_ddr.elf** file safe. This file is used and may be leaked when you trigger FOAT download, trigger FOTA download, turn off the fastboot function, or turn off the mtd-utils tool. You must prevent malicious damage of the firmware.

8. After Secure Boot is enabled and the sec partition data is erased, will Secure Boot still work normally?

Yes. After Secure Boot is enabled for the first time, the data of the **sec** partition will be written to the eFuse on the CPU side.

9. How do I perform a quick simple verification?

Refer to *Fibocom_Qualcomm_SDX35_Multi Upgrade Tool User Guide_Windows* to burn the signed full package.

10. After Secure Boot is enabled, how do I confirm whether it is enabled?

Check the debug uart log and you can see **Secure Boot: on**.

```
B - 238214Format: Log Type - Time(microsec) - Message - Optional Info
Log Type: B - Since Boot(Power On Reset),  D - Delta,  S - Statistic
S - QC_IMAGE_VERSION_STRING=BOOT.MXF.2.3-00501-KUNO_E-1
S - IMAGE_VARIANT_STRING=SocKunoLE
S - OEM_IMAGE_VERSION_STRING=ip-10-195-205-237
S - Boot Interface: NAND
S - Secure Boot: On
S - Boot Config @ 0x221c8600 = 0x00000041
S - JTAG ID @ 0x221c8744 = 0x002210e1
S - OEM ID @ 0x221c8700 = 0x00000167
S - Serial Number @ 0x221c8610 = 0x0000044cef9567dc
S - Feature Config Row 0 @ 0x221c2128 = 0x000aa90000006000
S - Feature Config Row 1 @ 0x221c2130 = 0x006dcf000365a900
S - Core 0 Frequency, 1248 MHz
S - PBL Patch Ver: 7
S - PBL freq: 600 MHZ
S - -------------------------
PBL Timestamps (in usecs)
-------------------------
```

7 Reference Documents

- Fibocom_FG132_OPENSDK Compilation Environment Building Guide
- Fibocom_Qualcomm_SDX35_Multi Upgrade Tool User Guide_Windows
- Fibocom_FG132_Upgrade Package Making Tool User Guide
- Fibocom_Linux_FOAT_Upgrade
- Fibocom_Linux_Firmware_Upgrade
- Fibocom_MTC_Application Guide_FOTA
- Fibocom_FG132_Open_API User Manual
- Fibocom_FG132_OpenSDK_Partition Adjustment Guide