EG06xK&Ex120K&EM06xK Series

Secure Boot Application Note

LTE-A Module Series

Version: 2.0

Date: 2024-12-30

Status: Released

At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local offices. For more information, please visit:
http://www.quectel.com/support/sales.htm.

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm.
Or email us at: support@quectel.com.

Legal Notices

We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an “as available” basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions

License Agreements

Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright

Our and third-party products hereunder may contain copyrighted material. Such copyrighted material shall not be copied, reproduced, distributed, merged, published, translated, or modified without prior written consent. We and the third party have exclusive rights over copyrighted material. No license shall be granted or conveyed under any patents, copyrights, trademarks, or service mark rights. To avoid ambiguities, purchasing in any form cannot be deemed as granting a license other than the normal non-exclusive, royalty-free license to use the material. We reserve the right to take legal action for noncompliance with abovementioned requirements, unauthorized use, or other illegal or malicious use of the material.

Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel's or third-party's servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.
b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.
c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.
d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

About the Document

Revision History

Version	Date	Author	Description
-	2022-05-09	Shaun DUAN	Creation of the document
1.0	2022-05-26	Shaun DUAN	First official release
1.1	2023-12-07	Shaun DUAN/ Jayde TONG	1. Added the applicable modules EG060K Series, EG120K Series, EM060K Series and EM120K-GL. 2. Updated the content of certificate chain (Chapter 2.3). 3. Added AT+QSECBOOT="roothash" (Chapter 3.3). 4. Updated development considerations (Chapter 4).
2.0	2024-12-30	Luck WANG	1. Added the applicable module EM061K-GL. 2. Due to the change of Secure Boot enablement scheme, numerous changes were made to this document. It should be read in its entirety.

Contents

Table Index

Figure Index

1 Introduction

Quectel LTE-A EG06xK, Ex120K and EM06xK series modules support Secure Boot function. This document describes how to enable the Secure Boot function on EG06xK, Ex120K and EM06xK series modules with Quectel_SDX12_SecBoot_Tools, including an overview of Secure Boot, procedures of enabling Secure Boot and FAQs.

1.1. Applicable Modules

Table 1: Applicable Modules

Module Family	Module
EG06xK	EG060K Series
	EG065K Series
Ex120K	EM120K-GL
	EG120K Series
EM06xK	EM060K Series
	EM061K-GL

2 Secure Boot Overview

Secure Boot is a security feature that establishes a trusted platform by using digital signatures to verify the legitimacy of each software component loaded during the boot sequence. It integrates signature verification throughout the majority of the module booting process and ensures that only authorized and unmodified software is executed on the module to prevent any unauthorized or maliciously modified software from compromising the module system.

3 Secure Boot Enabling Procedure

This chapter introduces how to enable Secure Boot function on the applicable modules.

3.1. Environment Preparation

Software environment:

- Ubuntu 16.04 or later versions
- Quectel_SDX12_SecBoot_Tools
- Python 2.7 version
- OpenSSL 1.0.x version

3.1.1. Tool Acquisition

Please contact Quectel Technical Support to obtain the *Quectel_SDX12_SecBoot_Tools.tar.gz* compressed signature tool package.

3.1.2. Dependency Installation

The Secure Boot related dependencies and installation commands are shown in the table below. Please strictly follow the sequence from 1 to 9 to install the dependencies, otherwise errors may occur during the installation process.

Table 2: Secure Boot Related Dependencies and Their Installation Commands

Installation Sequence	Dependency	Installation Command
1	mtd-utils	**sudo apt-get install mtd-utils**
2	squashfs-tools	**sudo apt-get install squashfs-tools**
3	libxml2-utils	**sudo apt-get install libxml2-utils**
4	cryptsetup-bin	**sudo apt-get install cryptsetup-bin**

5	crcmod	**sudo apt-get install python-crcmod**
6	liblzo2-dev	**sudo apt-get install liblzo2-dev**
7	pip	**sudo apt-get install python-pip**
8	python-lzo	**sudo pip install python-lzo**
9	future	**sudo pip install future**

Quectel offers pre-configured Docker or virtual machine images to simplify the setup process and avoid dependency issues. To acquire pre-configured Docker or virtual machine images, please contact Quectel Technical Support.

3.2. Key and Certificate Configuration

Extract the *Quectel_SDX12_SecBoot_Tools.tar.gz* compressed signature tool package to obtain the folder named *Quectel_SDX12_SecBoot_Tools*. The following files should be stored in the *certs* directory under the root directory of the tool:

- Root certificate for generating certificate chains: *oem_rootca.cer*
- Intermediate certificate for generating certificate chains: *oem_attestca.cer*
- Root key for issuing certificates: *oem_rootca.key*
- Intermediate key for issuing certificates: *oem_attestca.key*

Quectel provides two solutions for configuring keys and certificates:

- Configure through general keys and certificates
- Configure through script

NOTE

Before configuring keys and certificates, ensure that there are no existing keys and certificates in the *certs* directory.

3.2.1. General Keys and Certificates

The *general_certs* directory under the root directory of the tool contains general keys and certificates that can be directly used for signing software images and generating certificate chains.

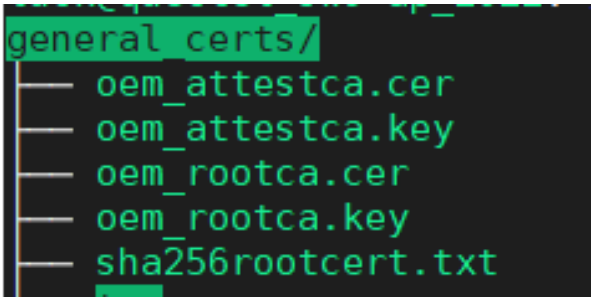

Figure 1: General Keys and Certificates

Execute the following command to copy the general keys and certificates to the *certs* directory to complete the configuration.

```
cp general_certs/* certs/ -rf
```

3.2.2. Script

The *gen_certs.sh* script in the root directory of the tool can be used to generate certificates and keys.

Execute the following command to run the *gen_certs.sh* script to generate keys and certificates with your project name (*<project_name>)* included in the OU field:

```
./gen_certs.sh -p <project_name>
```

The generated keys and certificates are automatically saved to the *certs* directory.

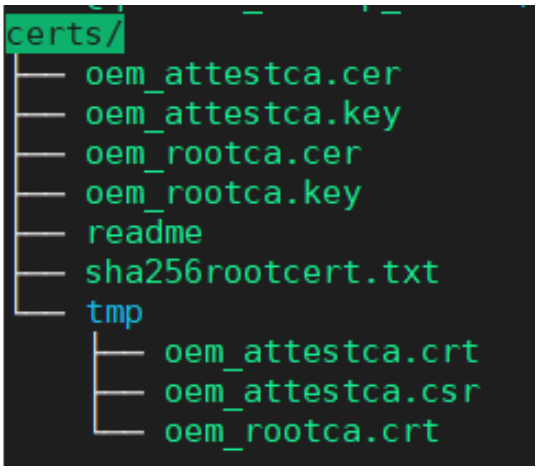

Figure 2: Keys and Certificates Generated Through Script

NOTE

Securely store the certificates and keys generated through the script. Losing them will prevent you from signing images and updating the firmware.

3.3. Image Signing

Step 1: Copy the following files from the firmware package to the *firmwares* directory in the root directory of the tool.

Table 3: Files to Be Copied

Directory	File
update	*appsboot.mbn*
	rpm.mbn
	sbl1.mbn
	tz.mbn
	sdxnightjar-sysfs.ubi
	devcfg.mbn
	NON-HLOS.ubi (if any)
update/firehose	*prog_nand_firehose_9x55.mbn*
upgrade	*targetfiles.zip*

NOTE

1. Depending on the actual firmware package, the *NON-HLOS.ubi* file might not be present in the *update* directory.
2. Copy *targetfiles.zip* from the *upgrade* directory only if you require FOTA.

Step 2: Sign the images.

Execute the following command to sign the images in *firmwares* directory and generate the *sec.dat* file for enabling Secure Boot:

```
./sign_firmwares.sh
```

The signed images and the generated *sec.dat* file are located in the *sign_firmwares* directory under the root directory of the tool.

Step 3: Replace the images listed in ***Table 3*** in the firmware package with the signed images, and copy *sec.dat* to the *update* directory of the firmware package.

3.4. Secure Boot Enablement

Secure Boot involves flashing the generated *sec.dat* file to the *sec* partition of the module using Quectel's QFlash tool. For instructions on how to use the QFlash tool, see ***document [1]***.

Step 1: Add the following contents of *sec* partition to the *rawprogram_nand_p4K_b256K_update.xml* configuration file in the *update/firehose/* directory within your original firmware package.

```
<program PAGES_PER_BLOCK="64" SECTOR_SIZE_IN_BYTES="4096" filename="..\sec.dat" num_partition_sectors=<value1> physical_partition_number="0" start_sector=<value2>/>
```

Find the name of the previous partition in *update/partition_nand.xml*. For example, the previous partition of the *sec* partition is *recoveryfs*, and the image of *recoveryfs* partition is *sdxnightjar-recoveryfs.ubi*.

```
    <partition>
        <name length="16" type="string">0:recoveryfs</name>
        <size_kb length="4">13312</size_kb>
        <pad_kb length="4">512</pad_kb>
        <which_flash>0</which_flash>
        <attr>0xFF</attr>
        <attr>0x01</attr>
        <attr>0x00</attr>
        <attr>0xFF</attr>
        <img_name type="string">sdxnightjar-recoveryfs.ubi</img_name>
    </partition>
    <partition>
        <name length="16" type="string">0:sec</name>
        <size_kb length="4">256</size_kb>
        <pad_kb length="4">256</pad_kb>
        <which_flash>0</which_flash>
        <attr>0xFF</attr>
        <attr>0x01</attr>
        <attr>0x00</attr>
        <attr>0xFF</attr>
    </partition>
    <partition>
        <name length="16" type="string">0:system</name>
        <size_kb length="4">91136</size_kb>
        <pad_kb length="4">1024</pad_kb>
        <which_flash>0</which_flash>
        <attr>0xFF</attr>
        <attr>0x01</attr>
        <attr>0x00</attr>
        <attr>0xFF</attr>
        <img_name type="string">sdxnightjar-sysfs.ubi</img_name>
    </partition>
```

1) *num_partition_sectors* represents the number of sectors in the partition, calculated as follows: *num_partition_sectors = (size_kb + pad_kb) / PageSize*.

- The values of *size_kb* (partition size) and *pad_kb* (additional padding space) can be found in the *update/partition_nand.xml* file. Here is an example:

  recoveryfs partition:

```
<partition>
    <name length="16" type="string">0:recoveryfs</name>
    <size_kb length="4">13312</size_kb>
    <pad_kb length="4">512</pad_kb>
    <which_flash>0</which_flash>
    <attr>0xFF</attr>
    <attr>0x01</attr>
    <attr>0x00</attr>
    <attr>0xFF</attr>
    <img_name type="string">sdxnightjar-recoveryfs.ubi</img_name>
</partition>
```

sec partition:

```
<partition>
    <name length="16" type="string">0:sec</name>
    <size_kb length="4">256</size_kb>
    <pad_kb length="4">256</pad_kb>
    <which_flash>0</which_flash>
    <attr>0xFF</attr>
    <attr>0x01</attr>
    <attr>0x00</attr>
    <attr>0xFF</attr>
</partition>
```

- The value of *PageSize* (page size of the partition) can be found in the *update/firehose/rawprogram_nand_p4K_b256K_update.xml file*, as shown below:

```
<?xml version="1.0" ?>
<data>
  <!--NOTE: This is an ** Autogenerated file **-->
  <!--BlockSize = 256 KB-->
  <!--PageSize  = 4 KB-->
  <!--NUM_PARTITION_SECTORS  = 131072-->
```

In this example:
$num\_partition\_sectors_{(recoveryfs)}$ = (13312 + 512) / 4 = 3456
$num\_partition\_sectors_{(sec)}$ = (256 + 256) / 4 = 128

2) *start_sector* represents the start sector of the partition, calculated as follows: *start_sector* = $start\_sector_{(previous\ partition)}$ + $num\_partition\_sectors_{(previous\ partition)}$.

The previous partition of the *sec* partition is *recoveryfs*. According to *update/firehose/rawprogram_nand_p4K_b256K_update.xml* file, the value of $start\_sector_{(recovery\_fs)}$ is 32576, as shown below:

```
filename="..\sdxnightjar-recoveryfs.ubi" num_partition_sectors="3456" physical_partition_number="0" start_sector="32576"/>
```

In this example:
$start\_sector_{(sec)}$ = $start\_sector_{(recoveryfs)}$ + $num\_partition\_sectors_{(recoveryfs)}$ = 32576 + 3456 = 36032

Therefore, the values you should add to *update/firehose/rawprogram_nand_p4K_b256K_update.xml* are as follows:

```
<program PAGES_PER_BLOCK="64" SECTOR_SIZE_IN_BYTES="4096" filename="..\sec.dat" num_partition_sectors="128" physical_partition_number="0" start_sector="36032"/>
```

NOTE

1. To avoid the failure of enabling Secure Boot due to miscalculations in the start sector of the *sec* partition, a simple validation can be performed.

   Taking the partition order of *recoveryfs*, *sec* and *system* as examples from the aforementioned sequence, validate the start sector of the *system* partition. If the calculated *start_sector* of the *system* partition matches the value obtained from *update/firehose/rawprogram_nand_p4K_b256K_update.xml*, the calculation is correct. The specific steps are as follows:

   1) According to *update/partition_nand.xml*, in this example, the partition following the *sec* partition is the *system* partition.
   2) According to *update/firehose/rawprogram_nand_p4K_b256K_update.xml*, the *start_sector* value for the *system* partition is 36160, as shown below:

```
"..\sdxnightjar-sysfs.ubi" num_partition_sectors="23040" physical_partition_number="0" start_sector="36160
```

   3) According to the above calculation, the value of *num_partition_sectors* for the *sec* partition is 128, and the value of *start_sector* is 36032, so *start_sector*$_{(system)}$ = *start_sector*$_{(sec)}$ + *num_partition_sectors*$_{(sec)}$ = 36032 + 128 = 36160, which is the value of *start_sector* for the *system* partition.

      The two values match, indicating that the calculation of the *start_sector* value for the partition is correct.

2. The partitions may vary for different modules. The above example is for reference only. Please refer to the partition table in the firmware package for the actual configuration.
3. If the *start_sector* corresponding to the previous partition of the *sec* partition is not found in the *update/firehose/rawprogram_nand_p4K_b256K_update.xml* file, then locate the partition before the previous one for the *sec* partition. And the *start_sector* calculation method remains the same as shown above.

Step 2: After modifying the configuration file, flash the updated firmware into the module through QFlash. For detailed flashing instructions, see ***document [1]***.

3.5. Secure Boot Enabling Verification

After successful flashing of the updated firmware, the module automatically reboots twice. During the first rebooting, the SBL detects the content of the *sec* partition and sends a command to TrustZone, instructing TrustZone to burn the root hash (extracted from *sec.dat*) to the hardware fuse. Following the root hash burning, the module reboots for the second time and starts normally. Once the module has completed its booting sequence, you can verify if Secure Boot is enabled using two methods:

1) AT command
2) devmem command

3.5.1. AT Command

To query the Secure Boot enablement status, send **AT+QSECCFG="secboot"** through the QCOM tool. If the response is:

- **+QSECCFG: "status",1**, it indicates that Secure Boot is enabled;
- **+QSECCFG: "status",0**, it indicates that Secure Boot is disabled. Please check whether the above steps have been performed correctly.

NOTE

1. For details about **AT+QSECCFG**, please refer to ***document [2]***.
2. For usage of QCOM tool, please refer to ***document [3]***.

3.5.2. devmem Command

Execute the following command through **adb shell** or UART after logging into the module:

```
devmem 0x000A01D0
```

If 0x00303030 is returned, it indicates that Secure Boot is enabled. If any other value is returned, it indicates that Secure Boot is disabled. Please check whether the above steps have been performed correctly.

3.6. Memory Dump Information Capture

After Secure Boot is enabled on the module, debugging features such as memory dump will be disabled for security reasons. To enable these debugging features, execute the following command:

```
./sign_firmwares.sh -s <serial_num>
```

The generated *apdp.mbn* is located in the *sign_firmwares* directory under the root directory of the tool. Copy *apdp.mbn* to the firmware package and then flash the firmware package to the module.

NOTE

1. The module serial number is a unique identifier for each module. You can retrieve it by executing **AT+QSECCFG="secboot"**. For details about **AT+QSECCFG**, see ***document [2]***.

```
AT+QSECCFG="secboot"
…
+QSECCFG: "serial_num","0xda491416"
…

OK
```

4 FAQs

4.1. Can Secure Boot Be Disabled?

Secure Boot can only be enabled with the hardware fuse and cannot be disabled after being enabled.

4.2. Do I Need to Generate *sec.dat* Repeatedly

If the module model and root certificate remain unchanged, there is no need to regenerate *sec.dat*.

4.3. Can I Run Unsigned Firmware on Module

Before Secure Boot is enabled, unsigned firmware can run on the module. However, once Secure Boot is enabled, unsigned firmware cannot run on the module.

4.4. Can I Flash Unsigned Firmware After Enabling Secure Boot

You cannot flash unsigned firmware with QFlash tool after enabling Secure Boot.

4.5. Does Secure Boot Still Work After *sec* Partition Is Erased

Erasing the data of the *sec* partition does not affect the functionality of Secure Boot.

4.6. When Should I Enable Secure Boot

It is only recommended that users enable Secure Boot when flashing the firmware package during the module factory settings phase.

4.7. How to Determine the Certificates Used on Secure Boot-enabled Devices

You can query the root certificate (*oem_rootca.cer*) used by Secure Boot-enabled devices by **AT+QSECCFG="secboot"**. The value of **<roothash>** in the command response **+QSECCFG: "root_hash",<roothash>** indicates the SHA256 hash of the root certificate (*oem_rootca.cer)*.

5 Appendix References

Table 4: Related Documents

Document Name
[1] Quectel_QFlash_User_Guide
[2] Quectel_EG06xK&Ex120K&EM06xK_Series_Security_Feature_Application_Note
[3] Quectel_QCOM_User_Guide

Table 5: Terms and Abbreviations

Abbreviation	Description
FOTA	Firmware Over-The-Air
PBL	Primary Boot Loader
SBL	Secondary Boot Loader
SRoT	Secure Root of Trust
UART	Universal Asynchronous Receiver/Transmitter