

EG800Q&EG91xQ Series

Secure Boot Application Note

LTE Standard Module Series

Version: 1.0

Date: 2024-11-27

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.

Copyright © Quectel Wireless Solutions Co., Ltd. 2024. All rights reserved.

About the Document

Revision History

| Version | Date | Author | Description |
|---------|------------|------------|--------------------------|
| - | 2024-03-21 | Yong ZHENG | Creation of the document |
| 1.0 | 2024-11-27 | Yong ZHENG | First official release |

Contents

| | |
|--|-----------|
| About the Document..... | 3 |
| Contents | 4 |
| Table Index..... | 5 |
| Figure Index | 6 |
| 1 Introduction | 7 |
| 2 Secure Boot Overview | 8 |
| 2.1. Definition | 8 |
| 2.2. Secure Boot Components..... | 8 |
| 2.2.1. eFuse | 8 |
| 2.2.2. ECSecure | 9 |
| 2.2.3. Images..... | 9 |
| 2.2.4. Image Header Files..... | 9 |
| 2.2.5. BootROM..... | 9 |
| 2.2.6. BootLoader..... | 9 |
| 2.3. Digital Signature..... | 10 |
| 3 Secure Boot Implementation | 11 |
| 3.1. System Boot Process..... | 11 |
| 3.1.1. Two-stage Boot Process..... | 11 |
| 3.1.1.1. BootROM Process | 12 |
| 3.1.1.2. BootLoader Process..... | 13 |
| 3.1.2. Secure Boot Result Verification | 14 |
| 3.2. Secure DFOTA..... | 15 |
| 4 AT Command..... | 17 |
| 4.1. AT Command Introduction | 17 |
| 4.1.1. Definitions..... | 17 |
| 4.1.2. AT Command Syntax | 17 |
| 4.2. Declaration of AT Command Examples | 18 |
| 4.3. AT Command Description..... | 18 |
| 4.3.1. AT+QSECCFG Query Whether Secure Boot is Enabled | 18 |
| 5 Appendix References | 19 |

Table Index

| | |
|--|----|
| Table 1: Types of AT Commands | 17 |
| Table 2: Terms and Abbreviations | 19 |

Figure Index

| | |
|---------------------------------------|----|
| Figure 1: Two-stage Boot Process..... | 12 |
| Figure 2: BootROM Process | 13 |
| Figure 3: BootLoader Process | 14 |
| Figure 4: Secure Boot Log | 15 |
| Figure 5: Secure DFOTA Process | 16 |

1 Introduction

This document explains the Secure Boot function and how to implement Secure Boot function on Quectel EG800Q series and EG91xQ family (EG915Q series and EG916Q-GL) modules.

2 Secure Boot Overview

2.1. Definition

Secure Boot is a process that involves authentication of the BootLoader image and system (AP and CP) images before allowing them to execute. The boot code (i.e., BootROM image and BootLoader image code) runs on the device chip and launches additional code and applications. After the boot process, the device is ready for use. Secure Boot ensures that only the code securely signed by the device manufacturer is executed. Once the secure code (i.e., securely signed BootROM image and BootLoader image code) runs, other code and applications can run if their signatures match the public keys stored in hardware. In short, it ensures that the BootLoader image and system images are the intended manufacturer versions and have not been tampered with by malware or malicious third parties.

Secure Boot authentication involves two processes:

- **Verification of integrity (hash verification):** This involves calculating the hash values of the BootLoader image and system images and comparing them against the hash values in image header files.
- **Verification of authenticity (ECDSA signature verification):** This involves checking the signature against a public key to confirm that the BootLoader image and system images are from trusted sources.

2.2. Secure Boot Components

2.2.1. eFuse

The eFuse array is an on-chip One-Time Programmable (OTP) non-volatile memory (NVM) that stores the Secure Boot enable bit and the public key used by the BootROM image and BootLoader image in Secure Boot authentication.

2.2.2. ECSecure

ECSecure is a PC-based software designed to enhance security features through ECDSA and hash algorithms. ECTSecure generates the following:

- Generates ECDSA public/private key pairs for the BootLoader image and system images.
- Uses ECDSA algorithm to generate signatures for the BootLoader image and system images with corresponding private keys.
- Uses the SHA224 algorithm to generate hash values for BootLoader image and system images.
- Generates a BootLoader image header file and a system image header file containing hash values and signatures for BootLoader image and system image authentication.

2.2.3. Images

An image is a binary file. Typically, the BootLoader image is named *ap_bootloader.bin*, AP image is named *ap_at_command.bin*, CP image is named *CP-demo-flash.bin*, and DFOTA image is named *ap_updater.bin*.

2.2.4. Image Header Files

The image header file includes the hash values and signatures (Besides, BootLoader image header file includes extra public key information). There include three image header files: BootLoader image header file *bl_sec_header.bin*, system image header file *sys_sec_header.bin* and DFOTA image header file *updater_sec_header.bin*.

2.2.5. BootROM

The BootROM is a 64 KB mask-programmed boot Read-Only Memory that cannot be directly accessed or written to. BootROM memory contains the BootROM image that initializes essential clocks, reads eFuse information at reset or boot-up, and loads the BootLoader image.

2.2.6. BootLoader

The BootLoader image is programmed into on-chip XIP flash memory, and executed in XIP mode. The BootLoader image initializes essential clocks, drivers, reads eFuse information, and loads the system images.

2.3. Digital Signature

The digital signature technology used by the module is based on the ECDSA asymmetric encryption algorithm, which uses both public and private keys. The private key encrypts an image, while the public key verifies the decrypted image. Access to only one of these keys, whether private or public, does not provide access to the other key.

The whole digital signature process is as follows:

- Step 1:** ECSecure generates ECDSA public/private key pairs for the BootLoader image and system images.
- Step 2:** ECSecure calculates the hash values of BootLoader image and system images.
- Step 3:** ECSecure encrypts the hash values with corresponding private keys (named signatures) and writes the signatures into the image header files.
- Step 4:** ECSecure calculates the hash values of image header files.
- Step 5:** ECSecure generates the image header files including hash values for images and for image header files as well as signatures.
- Step 6:** Verify the integrity of image header files and images:
 - Secure code calculates the hash values of the image header files and compares them with the hash values stored in the header files to ensure the integrity of image header files.
 - Secure code calculates the hash values of the images and compares them with the hash values stored in the image header files to ensure the integrity of images.
- Step 7:** Decrypt the signatures with corresponding public keys to generate the decrypted hash values, and compare them with the previously calculated hash values. If they match, the images are secure, the signatures are verified and the boot process is continued; if they do not match, the signature verification fails, and the module fails to boot up.

3 Secure Boot Implementation

Secure Boot is implemented in two stages: the first stage involves setting Secure Boot enable bit and burning the public key, while the second stage is the regular boot process, which consists of two phases as detailed in **Chapters 3.1.1.1** and **3.1.1.2**.

3.1. System Boot Process

3.1.1. Two-stage Boot Process

During the first boot, if the Secure Boot enable bit in eFuse is not set, the BootLoader image will verify *bl_sec_header.bin* file, and then burn the public key read from the file into eFuse, followed by setting the Secure Boot enable bit in eFuse. Upon reboot, the second boot proceeds as the regular boot. BootROM image verifies the hash value and signature of the BootLoader image (See **Chapter 3.1.1.1** for details) and then loads BootLoader image to authenticate system images (See **Chapter 3.1.1.2** for details). The two-stage boot process is illustrated in the following figure.

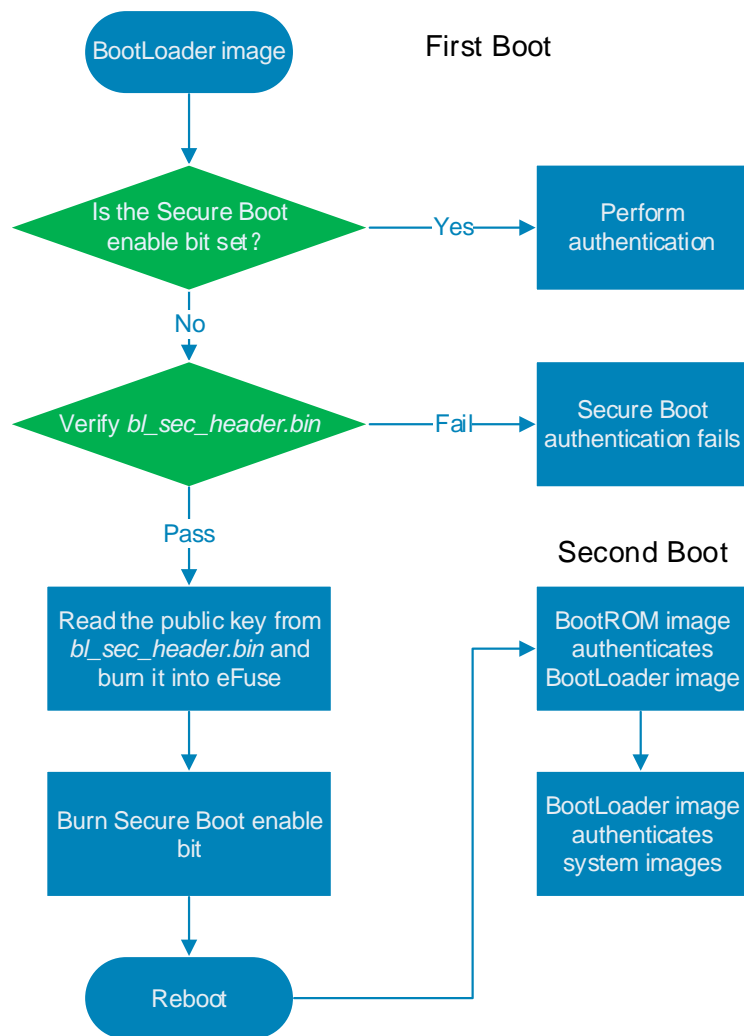


Figure 1: Two-stage Boot Process

NOTE

You can query whether Secure Boot is enabled through **AT+QSECCFG**. For more details, please refer to **Chapter 4.3.1**.

3.1.1.1. BootROM Process

The BootROM image reads the Secure Boot enable bit in eFuse to determine whether the Secure Boot is enabled. If it is enabled, the BootROM image verifies the integrity of the BootLoader image and image header file by calculating the hash values and comparing them with the hash values stored in the BootLoader image header file. If they do not match, the integrity verification fails, and the BootROM image will stop running. If integrity verification succeeds, the BootROM image reads the ECDSA public key from eFuse and verifies the digital signature read from *bl_sec_header.bin* using the ECDSA algorithm. If the signature verification fails, the BootROM image will stop running. If the BootLoader image is authenticated

successfully, the BootROM image will load the BootLoader image. The BootROM process is illustrated in the following figure.

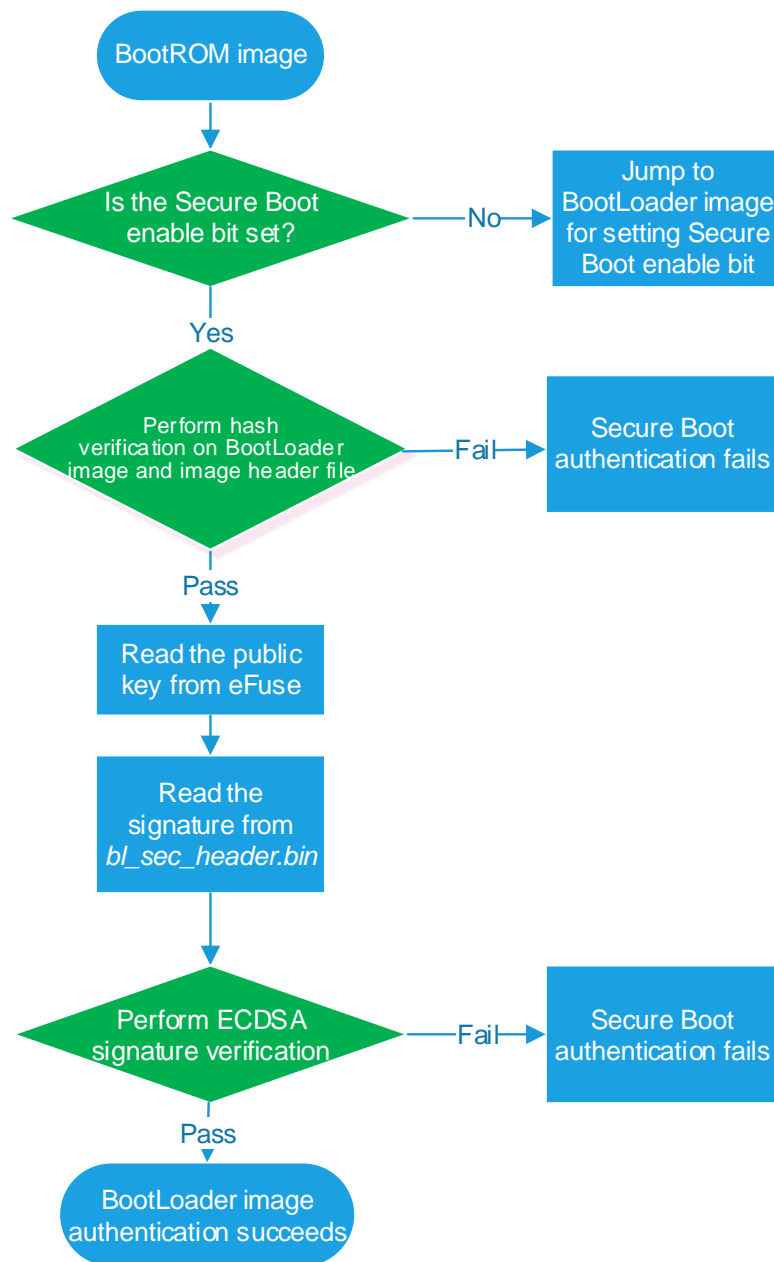


Figure 2: BootROM Process

3.1.1.2. BootLoader Process

The BootLoader image reads the Secure Boot enable bit in eFuse to determine if the Secure Boot is enabled. If it is enabled, the BootLoader image verifies the integrity of the system images and image header file by calculating their hash values and comparing them with the hash values stored in the system image header file. If the integrity verification of the system images fails, the BootLoader image will stop

running. If the integrity verification succeeds, the BootLoader image reads the public key from eFuse, and verifies the signature in the system image header file (*sys_sec_header.bin*) with the public key. If the signature verification fails, the BootLoader image will stop running. If the system images are authenticated successfully, the BootLoader image will load system images. The BootLoader process is illustrated in the following figure.

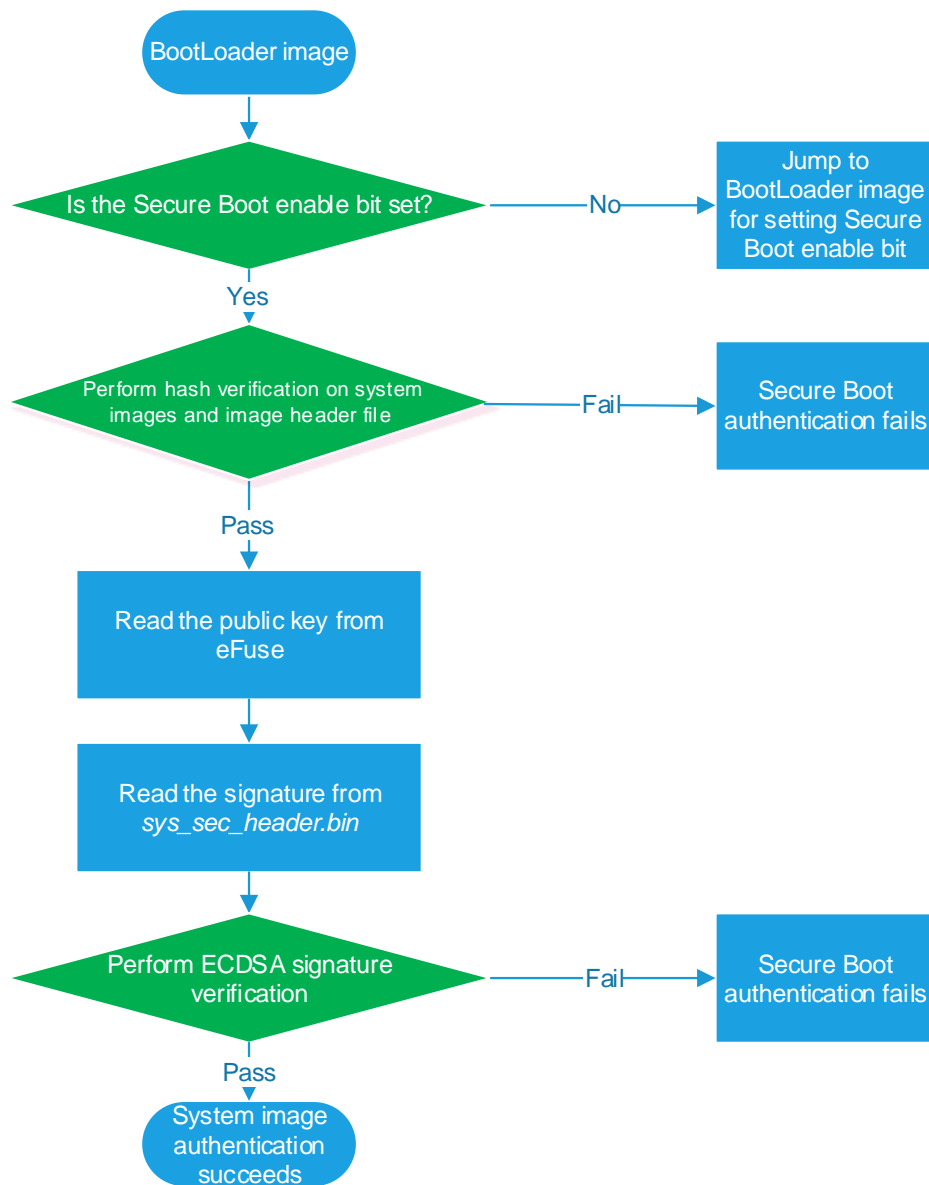


Figure 3: BootLoader Process

3.1.2. Secure Boot Result Verification

Boot the module and check if the BootLoader image and system images have been authenticated successfully by examining the debug log.

```

0xf8, 0x38, 0xec, 0xd6, 0xc5, 0x87, 0x2a, 0x4a, 0x34, 0x52, 0x6a, 0xca, 0x2f, 0x65, 0x50, 0xd6, 0x1
, 0xa5, 0x1f, 0xa5, 0xd5, 0x7, 0xd3, 0xab, 0xbc, 0x9b, 0x6c, 0xd,
[2024-01-15_19:42:19:592]
0x82, 0x1b, 0x5d, 0x53, 0xb0, 0xb7, 0x50, 0xf0, 0x8, 0x7e, 0x2b, 0x3d, 0x93, 0x89, 0x72, 0x89, 0xe4
, 0xf5, 0xbf, 0x66, 0x1b, 0x2d, 0x81, 0x7a, 0xed, 0x65, 0x6b, 0xe,
[2024-01-15_19:42:19:617]start secure boot process
[2024-01-15_19:42:19:617]LoadVerifyImageHead read(len=272), Time(ms)->1.
[2024-01-15_19:42:19:617]VerifyImageHead Output:
[2024-01-15_19:42:19:617]
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
[2024-01-15_19:42:19:659]sha256_data(len=272), Time(ms)->351.
[2024-01-15_19:42:19:659]verify image head success
[2024-01-15_19:42:19:659]VerifyImageBody size 0x2dc000
[2024-01-15_19:42:19:659]LoadVerifyImageBody ApImageSize=2975252
[2024-01-15_19:42:20:094]LoadVerifyImageBody CpImageSize=466180
[2024-01-15_19:42:20:169]sha224_data(aplen=2975252, cplen=466180), Time(ms)->12662.
[2024-01-15_19:42:20:169]secure bit enable to verify image
[2024-01-15_19:42:20:304]uECC_verify Time(ms)->2732.
[2024-01-15_19:42:20:304]VerifyImageBody successful.

```

Figure 4: Secure Boot Log

3.2. Secure DFOTA

Secure DFOTA is mainly performed in the *ap_updater.bin* image and it includes two stages: BootLoader image stage and *ap_updater.bin* image stage.

- BootLoader image stage
 - 1) After BootLoader image detects DFOTA upgrade flag, it executes a hash verification on *ap_updater.bin* to ensure the security of *ap_updater.bin*. After a successful hash verification, proceed to the step 2).
 - 2) BootLoader image reads the signature from *updater_sec_header.bin* and perform ECDSA signature verification. If the signature verification succeeds, loads the *ap_updater.bin* image.
- *ap_updater.bin* image stage
 - 1) *ap_updater.bin* image calculates the hash value on delta upgrade package and compares it with the hash value in the delta upgrade package to ensure the integrity of the delta upgrade package.
 - 2) Execute hash verification on AP image and CP image respectively to ensure the old images in delta upgrade package are consistent with the images of the module.
 - 3) After a successful authentication, the upgrade proceeds. Upon completion of the upgrade, reboot the system and authenticate the new images.

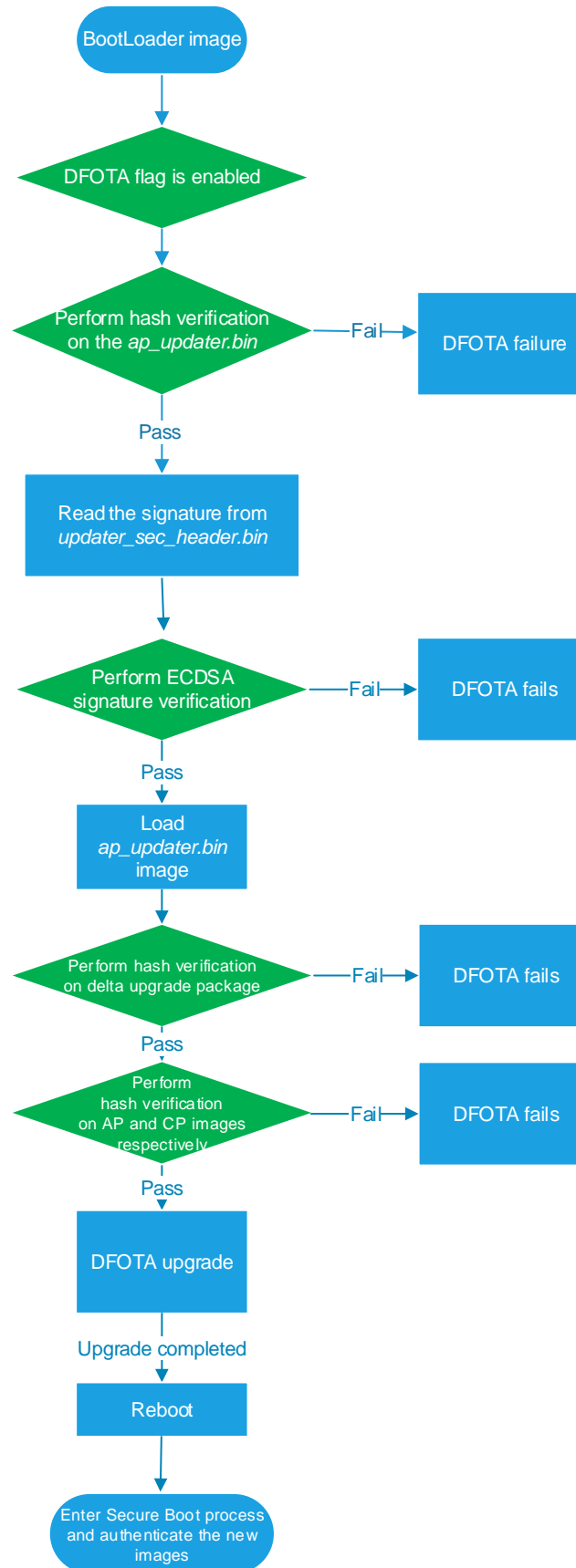


Figure 5: Secure DFOTA Process

4 AT Command

4.1. AT Command Introduction

4.1.1. Definitions

- **<CR>** Carriage return character.
- **<LF>** Line feed character.
- **<...>** Parameter name. Angle brackets do not appear on the command line.
- **[...]** Optional parameter of a command or an optional part of TA information response. Square brackets do not appear on the command line. When an optional parameter is not given in a command, the new value equals its previous value or the default settings, unless otherwise specified.
- **Underline** Default setting of a parameter.

4.1.2. AT Command Syntax

All command lines must start with **AT** or **at** and end with **<CR>**. Information responses and result codes always start and end with a carriage return character and a line feed character: **<CR><LF><response><CR><LF>**. In tables presenting commands and responses throughout this document, only the commands and responses are presented, and **<CR>** and **<LF>** are deliberately omitted.

Table 1: Types of AT Commands

| Command Type | Syntax | Description |
|-------------------|---|--|
| Test Command | AT+<cmd>=? | Test the existence of the corresponding command and return information about the type, value, or range of its parameter. |
| Read Command | AT+<cmd>? | Check the current parameter value of the corresponding command. |
| Write Command | AT+<cmd>=<p1>[,<p2>[,<p3>[...]]] | Set user-definable parameter value. |
| Execution Command | AT+<cmd> | Return a specific information parameter or perform a specific action. |

4.2. Declaration of AT Command Examples

The AT command examples in this document are provided to help you learn about the use of the AT commands introduced herein. The examples, however, should not be taken as Quectel's recommendations or suggestions about how to design a program flow or what status to set the module into. Sometimes multiple examples may be provided for one AT command. However, this does not mean that there is a correlation among these examples, or that they should be executed in a given sequence.

4.3. AT Command Description

4.3.1. AT+QSECCFG Query Whether Secure Boot is Enabled

This command queries whether Secure Boot is enabled.

AT+QSECCFG Query Whether Secure Boot is Enabled

| | |
|---|---|
| Write Command AT+QSECCFG="secboot_status" | Response +QSECCFG: "secboot_status",<status> OK Or ERROR |
| Maximum Response Time | 300 ms |
| Characteristics | The command takes effect immediately. The configuration is not saved. |

Parameter

| | |
|-----------------------|-----------------------------------|
| <status> | Integer type. Secure Boot status. |
| 0 | Disabled |
| 1 | Enabled |

Example

```

AT+QSECCFG="secboot_status"           //Query Secure Boot status.
+QSECCFG: "secboot_status",1          //Secure Boot is enabled.

OK

```

5 Appendix References

Table 2: Terms and Abbreviations

| Abbreviation | Description |
|--------------|--|
| AP | Application Processor |
| CP | Control Processor |
| DFOTA | Delta Firmware Upgrade Over-The-Air |
| ECDSA | Elliptic Curve Digital Signature Algorithm |
| NVM | Non-Volatile Memory |
| OTP | One Time Programmable |
| PC | Personal Computer |
| ROM | Read Only Memory |
| XIP | Execute in Place |