ECx00U&EGx00U 系列 QuecOpen Secure Boot 用户指导

LTE Standard 模块系列

版本：1.0

日期：2021-08-19

状态：受控文件

上海移远通信技术股份有限公司始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市闵行区田林路 1016 号科技绿洲 3 期（B 区）5 号楼　　邮编：200233
电话：+86 21 51086236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：http://www.quectel.com/cn/support/sales.htm。

如需技术支持或反馈我司技术文档中的问题，可随时登陆如下网址：
http://www.quectel.com/cn/support/technical.htm 或发送邮件至：support@quectel.com。

前言

上海移远通信技术股份有限公司提供该文档内容用以支持其客户的产品设计。客户须按照文档中提供的规范、参数来设计其产品。因未能遵守有关操作或设计规范而造成的损害，上海移远通信技术股份有限公司不承担任何责任。在未声明前，上海移远通信技术股份有限公司有权对该文档进行更新。

免责声明

上海移远通信技术股份有限公司尽力确保开发中功能的完整性、准确性、及时性或效用，但不排除上述功能错误或遗漏的可能。除非其他有效协议另有规定，否则上海移远通信技术股份有限公司对开发中功能的使用不做任何暗示或明示的保证。在适用法律允许的最大范围内，上海移远通信技术股份有限公司不对任何因使用开发中功能而遭受的损失或损害承担责任，无论此类损失或损害是否可以预见。

保密义务

除非上海移远通信技术股份有限公司特别授权，否则我司所提供文档和信息的接收方须对接收的文档和信息保密，不得将其用于除本项目的实施与开展以外的任何其他目的。未经上海移远通信技术股份有限公司书面同意，不得获取、使用或向第三方泄露我司所提供的文档和信息。对于任何违反保密义务、未经授权使用或以其他非法形式恶意使用所述文档和信息的违法侵权行为，上海移远通信技术股份有限公司有权追究法律责任。

版权申明

本文档版权属于上海移远通信技术股份有限公司，任何人未经我司允许而复制转载该文档将承担法律责任。

文档历史

修订记录

版本	日期	作者	变更表述
-	2021-01-12	Neo KONG	文档创建
1.0	2021-08-19	Neo KONG	受控版本

目录

表格索引

图片索引

1 引言

移远通信 LTE Standard ECx00U 系列和 EGx00U 模块支持 QuecOpen®方案。QuecOpen®是基于 RTOS 的嵌入式开发平台，可简化 IoT 应用的软件设计和开发过程。有关 QuecOpen®的详细信息，请参考 ***文档 [1]***。

本文档主要介绍在 QuecOpen®方案下，移远通信 ECx00U 系列和 EGx00U 模块的 Secure Boot 功能，包括概述、方案介绍和使能步骤。

备注

移远通信提供的 SDK 分为 APP SDK 和 CSDK。APP SDK 中包含 Kernel、编译完成的固件以及示例源码等；CSDK 中包含包含 Kernel、编译完成的固件、示例源码和部分芯片供应商源码文件。

1.1. 适用模块

表 1：适用模块

模块系列	模块
ECx00U	EC200U 系列
	EC600U 系列
EGx00U	EG500U-CN
	EG700U-CN

2 Secure Boot 概述

2.1. 简介

Secure Boot 是建立在受信平台上的一种安全启动序列。为防止任何未经合法签名或恶意修改的软件在模块上运行，Secure Boot 在模块下载和启动过程的每个阶段均增加了签名验证环节。在一系列下载和启动过程中，须有一个基准受信阶段，移远通信 ECx00U 系列和 EGx00U QuecOpen 模块中的 Bootrom 作为固件固化在模块中，无法被修改，可作为基准受信阶段。

Secure Boot 功能作用：

1. 禁止烧录未经授权的官方固件
2. 禁止运行未经授权的官方固件
3. 禁止非法追踪和调试代码
4. 允许安全升级

2.2. 数字签名

模块采用的数字签名技术基于非对称加密算法。非对称加密算法将密钥分为公钥（Public key）和私钥（Private key）两种。采用私钥对某段信息进行加密，使用公钥对信息解密进行验证。公钥加密的内容，可使用私钥解开；且私钥加密的内容，公钥可以解开。然而，单独已知公钥或私钥，无法推算出另一份密钥。

由于数字签名算法耗时较长，为了提高效率，一般使用私钥把信息（镜像文件）的数字摘要（Hash 值）加密，将此密文称为签名，附着于信息后。接收方使用公钥解密，得到解密后的摘要（Hash 值），然后重新计算信息（镜像文件）获取另一个摘要（Hash 值）。若两个 Hash 值一致，则签名认证通过，烧录或启动程序继续下一个流程；若不一致，则说明签名认证不通过，退出固件烧录和模块启动过程。

3 Secure Boot 方案介绍

3.1. 秘钥生成

模块使用 rsakeygen.exe 工具生成密钥（公钥和私钥），该工具位于 SDK 包中 *tools\win32* 目录下。生成的密钥存放于当前目录下 *key.db* 文件中。客户需妥善保存私钥，不可泄露。

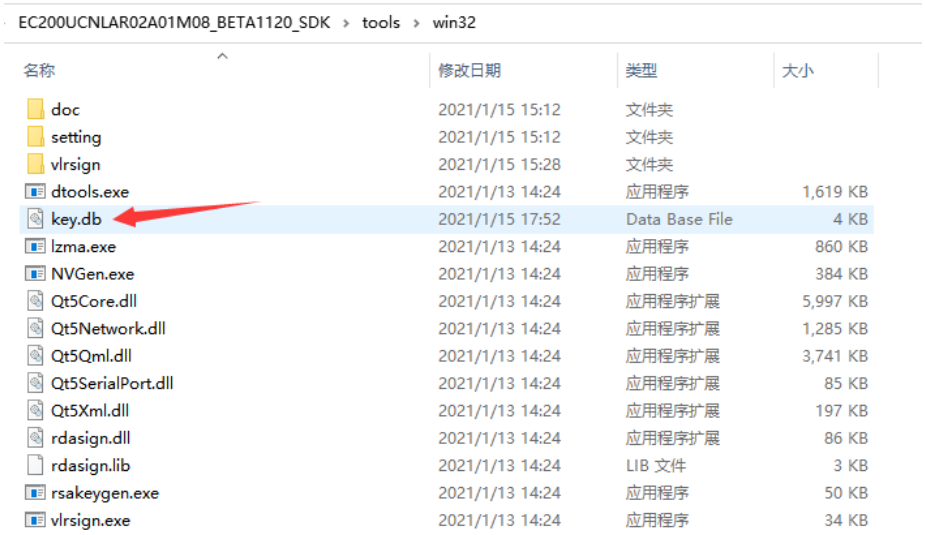

图 1：密钥位置

3.2. 签名流程

模块使用 vlrsign.exe 工具对镜像文件进行签名，该工具位于 SDK 包中 *tools\win32* 目录下，仅支持命令行模式，具体用法可参考*\tools\win32\doc* 目录下的 *vlrsign.md* 文件。签名流程如下：

1. vlrsign.exe 使用 blake2 hash 算法对待签名镜像文件（编译生成的镜像文件，需要签名的文件包括：*fdl1*、*fdl2*、*boot*、*ApSystemProgram* 和 *appimage*）进行计算并生成摘要（Hash 值）。
2. vlrsign.exe 使用 *key.db* 中保存的私钥对摘要进行签名（使用 ed25519 数字签名算法），然后将签名放在对应的镜像文件后面。
3. vlrsign.exe 将公钥放在 *boot.img* 后面，随固件一起烧录至模块的 flash 中。

3.3. 使能 Secure Boot

修改 SDK 中 *components\ql-application\ql_app_feature_config.cmake* 文件里字段 QL_APP_FEATURE_SECURE_BOOT为ON，随后重启模块，系统自动将*boot.img*后的公钥写入efuse中，从而使能Secure Boot功能。写入后不可更改，并且设置efuse中的Secure Boot使能标志位。详细使能步骤请参考***第4章***。

3.4. 验签流程

3.4.1. 未使能 Secure Boot

若 Secure Boot 功能未使能，系统跳过验签步骤，直接下载或启动。

3.4.2. 已使能 Secure Boot

若 Secure Boot 功能已使能，模块重启后进入验签流程。

1. 系统使用 efuse 中公钥对附加在 *boot.img* 后的签名进行解密，获取摘要（Hash 值）；
2. 比较新获取的摘要（Hash 值）和原始摘要（Hash 值），若内容一致，表示验签通过，模块开始正常下载或启动；若内容不一致，表示验签失败，禁止下载或启动。

3.5. 信任链

3.5.1. 下载信任链（多级验证）

模块中最初的受信阶段为 Bootrom，是固化在芯片中无法修改的一段代码，因此是可信任的并可作为下一阶段的授信中心。Bootrom 通过 UART/USB 下载 *fdl1* 到 RAM 中对其进行验签，验签通过后可对 *fdl2* 进行下载并验签。随后由验签通过的 *fdl2* 下载 BootLoader、*ApSystemProgram* 和 APP 到模块 flash 并对其进行验签。

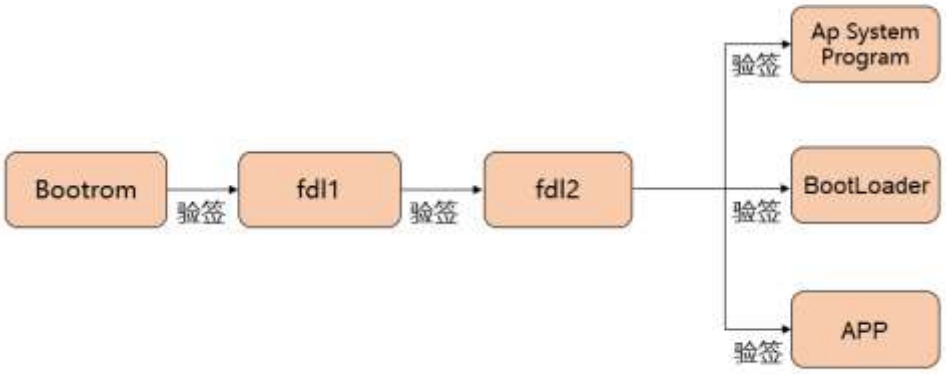

图 2：下载过程信任链

3.5.2. 启动信任链（多级验证）

模块的启动过程为多级加载过程，前一级代码加载后一级代码。使能 Secure Boot 后，每一级代码在加载前会都执行校验操作，如果校验失败，模块就会启动失败。

Bootrom 是可信任的（启动过程的最可靠实体）并可作为下一启动阶段的授信中心；下一启动阶段为 bootloader，经过验证签名后被执行；bootloader 会在加载 ApSystemProgram 时对其进行验签，验签通过后运行；ApSystemProgram 在加载 APP 时对其进行验签，验签通过后运行。此过程为整个固件启动过程信任链。

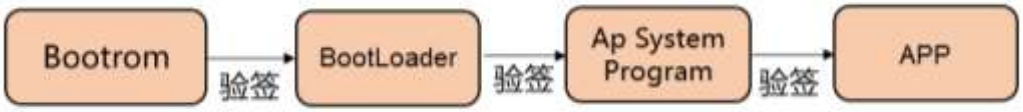

图 3：启动过程信任链

4 Secure Boot 使能步骤

移远通信 ECx00U 系列和 EGx00U QuecOpen 模块软件中已经封装了 Secure Boot 使能操作，Secure Boot 功能使能步骤如下（各模块的使能步骤一致）。下述章节中以 EC200U-CN QuecOpen 模块为例介绍如何开启 Secure Boot。

步骤一：执行 rsakeygen.exe 程序，输入密码和产品名称，生成秘钥；
步骤二：更改 *chip_8910.cmake*（CSDK 开发包）或 *CMakeLists.txt*（APP SDK 开发包）中的 sign_password 和 sign_product 字段，分别为运行 RSAKeyGen.exe 程序时输入的密码和产品名称；
步骤三：修改 *ql_app_feature_config.cmake* 中的宏 QL_APP_FEATURE_SECURE_BOOT 为 ON；
步骤四：取消 *ql_init.c* 中的 *ql_dev_enable_secure_boot* 函数注释；
步骤五：使用 *buildxxx.bat* 编译生成固件，生成的固件即为带有新秘钥签名且开启 Secure Boot 的固件，烧录该固件后模块的 Secure Boot 功能即使能。

备注

移远通信提供的 SDK 分为 APP SDK 和 CSDK。APP SDK 中包含 Kernel、编译完成的固件以及示例源码等；CSDK 中包含包含 Kernel、编译完成的固件、示例源码和部分芯片供应商源码文件。

4.1. 生成密钥

运行用于生成密钥的工具 rsakeygen.exe（rsakeygen.exe 的具体用法可参考*\tools\win32\doc* 目录下的 *vlrsign.md* 文档）。该工具仅支持命令行模式，命令行选项说明如下：

-v, --version 显示版本信息
-pw 输入口令。8 位 ASCII 码。后续 vlrsign 根据读取的签名秘钥校验口令是否正确。
-pn 输入产品名称。不超过 49 位 ASCII 码。后续 vlrsign 根据输入的产品名称检索相应的签名秘钥。

即使输入参数相同，rsakeygen.exe 每次产生的公私钥对均不同，生成的 *key.db* 也不同。重复执行命令生成密钥时，不会覆盖上一次运行生成的公私钥对。rsakeygen.exe 生成的公私钥对通过加密的方式保存到同目录下的 *key.db* 文件中，该文件包含有敏感信息，需妥善保存，不可泄露或丢失。

若文件信息泄露，他人可通过泄露的 *key.db* 对固件进行签名。由于公钥和终端 efuse 中的公钥是相同的，可通过 Secure Boot 检查；若文件信息丢失，因工具无法再次产生公钥相同的公私钥对，则无法对后续的固件进行签名，也无法对已签名的固件升级。

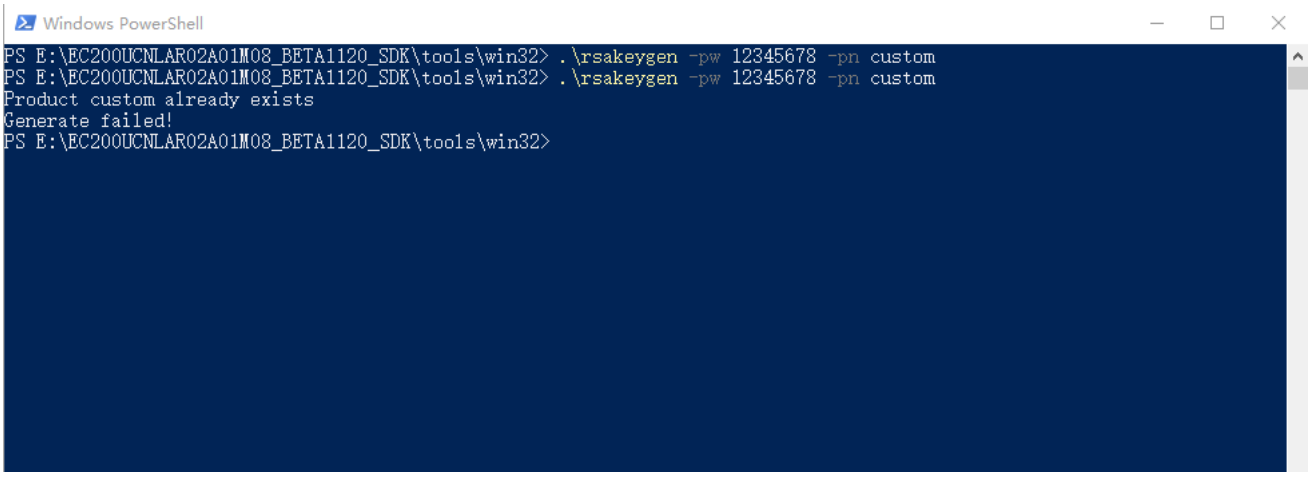

图 4：生成新密钥

4.2. 更新密钥

移远通信 ECx00U 系列和 EGx00U QuecOpen 模块的签名流程已集成到 Windows 和 Linux 平台集成编译中，编译完成后自动对 fdl1、fdl2、boot、ApSystemProgram 以及 appimage 的镜像文件进行签名，并组成后缀为.pac 的包。默认使用 test 秘钥，即 Product Name：test 以及 Password：12345678。在实际操作中需将默认使用的 test 密钥修改为新生成的密钥。

- **CSDK**

生成公钥和私钥后，分别修改 CSDK 中 *components\chip\chip_8910.cmake* 文件里 sign_password 和 sign_product 字段为运行 rsakeygen.exe 时输入的密码和产品名称。示例如下：

```
set(unittest_ldscript ${SOURCE_TOP_DIR}/components/hal/ldscripts/flashrun.ld)
set(pac_fdl_files ${out_hex_dir}/fdl1.sign.img ${out_hex_dir}/fdl2.sign.img)
set(sign_password 12345678)      # customer product shall replace with customer's key
set(sign_product test)           # customer product shall replace with customer's product name

function(sign_image src dst)
    add_custom_command(OUTPUT ${dst}
        COMMAND vlrsign --pw ${sign_password} --pn ${sign_product} --ha Blake2
            --img ${src} --out ${dst}
        DEPENDS ${src}
    )
endfunction()

function(pacvariant_gen)
    # Sign fdl1/fdl2/boot/ap. They are common to all variants
    set(sign_boot_padlen 0xbce0)     # can't be changed
    add_custom_command(OUTPUT ${pac_fdl_files} ${out_hex_dir}/boot.sign.img ${out_hex_dir}/${BUILD_TARGET}.sign.img  ${out_
        COMMAND vlrsign --pw ${sign_password} --pn ${sign_product} --ha Blake2
            --img ${out_hex_dir}/boot.img --out ${out_hex_dir}/boot.sign.img --plen ${sign_boot_padlen}
        COMMAND vlrsign --pw ${sign_password} --pn ${sign_product} --ha Blake2
            --img ${out_hex_dir}/fdl1.img --out ${out_hex_dir}/fdl1.sign.img
```

图 5：CSDK 重新配置秘钥

- **APP SDK**

生成公钥和私钥后，修改 APP SDK 根目录下 *CMakeLists.txt* 配置文件中的 sign_password 和 sign_product 字段，分别为 rsakeygen.exe 运行时输入的值。示例如下：

```
#if(CONFIG_QUEC_PROJECT_FEATURE_SECURE_BOOT)
if(QL_APP_FEATURE_SECURE_BOOT)
# Sign fdl1/fdl2/boot/ap. They are common to all variants
set(img_files ${SOURCE_TOP_DIR}(components/lib)
set(sign_password 12345678)      # customer product shall replace with customer's key
set(sign_product test)           # customer product shall replace with customer's product name
set(sign_boot_padlen 0xbce0)     # can't be changed
execute_process(
    COMMAND ${SOURCE_TOP_DIR}/tools/win32/vlrsign/VLRSign --pw ${sign_password} --pn ${sign_product} --ha Blake2
        --img ${img_files}/8915DM_cat1_open.img --out ${img_files}/8915DM_cat1_open.sign.img
    COMMAND ${SOURCE_TOP_DIR}/tools/win32/vlrsign/VLRSign --pw ${sign_password} --pn ${sign_product} --ha Blake2
        --img ${img_files}/boot.img --out ${img_files}/boot.sign.img --plen ${sign_boot_padlen}
    COMMAND ${SOURCE_TOP_DIR}/tools/win32/vlrsign/VLRSign --pw ${sign_password} --pn ${sign_product} --ha Blake2
        --img ${img_files}/fdl1.img --out ${img_files}/fdl1.sign.img
    COMMAND ${SOURCE_TOP_DIR}/tools/win32/vlrsign/VLRSign --pw ${sign_password} --pn ${sign_product} --ha Blake2
        --img ${img_files}/fdl2.img --out ${img_files}/fdl2.sign.img
    DEPENDS ${img_files}/8915DM_cat1_open.img ${img_files}/boot.img ${img_files}/fdl1.img #${out_hex_dir}/appimage.img
        ${img_files}/fdl2.img
)
endif()
```

图 6：APP SDK 重新配置秘钥

4.3. 启用 Secure Boot 宏

打开 *components\ql-application* 目录下的 *ql_app_feature_config.cmake* 文件，将 QL_APP_FEATURE_SECURE_BOOT 开关 OFF 改为 ON。

```
if(CONFIG_QUEC_PROJECT_FEATURE_SECURE_BOOT)
  option(QL_APP_FEATURE_SECURE_BOOT  "Enable SECURE BOOT" OFF)
  else()
  message(STATUS "FEATURE SECURE BOOT is disabled at core!")
  option(QL_APP_FEATURE_SECURE_BOOT  "Enable SECURE BOOT" OFF)
  endif()
  message(STATUS "QL_APP_FEATURE_SECURE_BOOT ${QL_APP_FEATURE_SECURE_BOOT}")
```

图 7：启用 Secure Boot 宏

4.4. 启用 Secure Boot 使能函数

将 *components\ql-application\init* 目录下 *ql_init.c* 文件中的 Secure Boot 使能函数启用（取消注释）。

```
static void ql_init_demo_thread(void *param)
{
    QL_INIT_LOG("init demo thread enter, param 0x%x", param);
#ifdef QL_APP_FEATURE_SECURE_BOOT
    //ql_dev_enable_secure_boot();
#endif

#if 0
    ql_gpio_app_init();
    ql_gpioint_app_init();
#endif

#ifdef QL_APP_FEATURE_LEDCFG
    ql_ledcfg_app_init();
#endif

#ifdef QL_APP_FEATURE_AUDIO
    //ql_audio_app_init();
#endif
```

图 8：取消使能函数的注释

4.5. 编译固件包

使用根目录下的 *build_all.bat*（CSDK 开发包）或 *build_app.bat*（APP SDK 开发包）编译生成固件。该固件带有新秘钥签名且开启 Secure Boot 功能。编译完成后在 SDK 根目录下生成 *target* 文件夹，用于存放固件包。

CSDK 编译输出文件如下，其中 *8915DM_cat1_open_EC200UCNLAR01A01M08.pac* 为仅含 Kernel 的固件包，*8915DM_cat1_open_EC200UCNLAR01A01M08 _merge.pac* 为含 Kernel 和 appimage 的固件包。

名称	修改日期	类型	大小
app	2020/12/31 16:14	文件夹	
prepack	2020/12/31 16:14	文件夹	
8915DM_cat1_open.elf	2020/12/31 16:14	ELF 文件	16,988 KB
8915DM_cat1_open.map	2020/12/31 16:14	MAP 文件	3,834 KB
8915DM_cat1_open_EC200UCNLAR01A01M08.pac	2020/12/31 16:14	PAC 文件	5,787 KB
8915DM_cat1_open_EC200UCNLAR01A01M08_merge.pac	2020/12/31 16:14	PAC 文件	5,854 KB

图 9：CSDK 编译输出文件

APP SDK 编译输出文件如下，其中 *appimage.pac* 为仅含 appimage 的固件包，*8915DM_cat1_open_core.pac* 为仅含 Kernel 的固件包，*appimage_merge.pac* 为含 Kernel 和 appimage 的固件包。

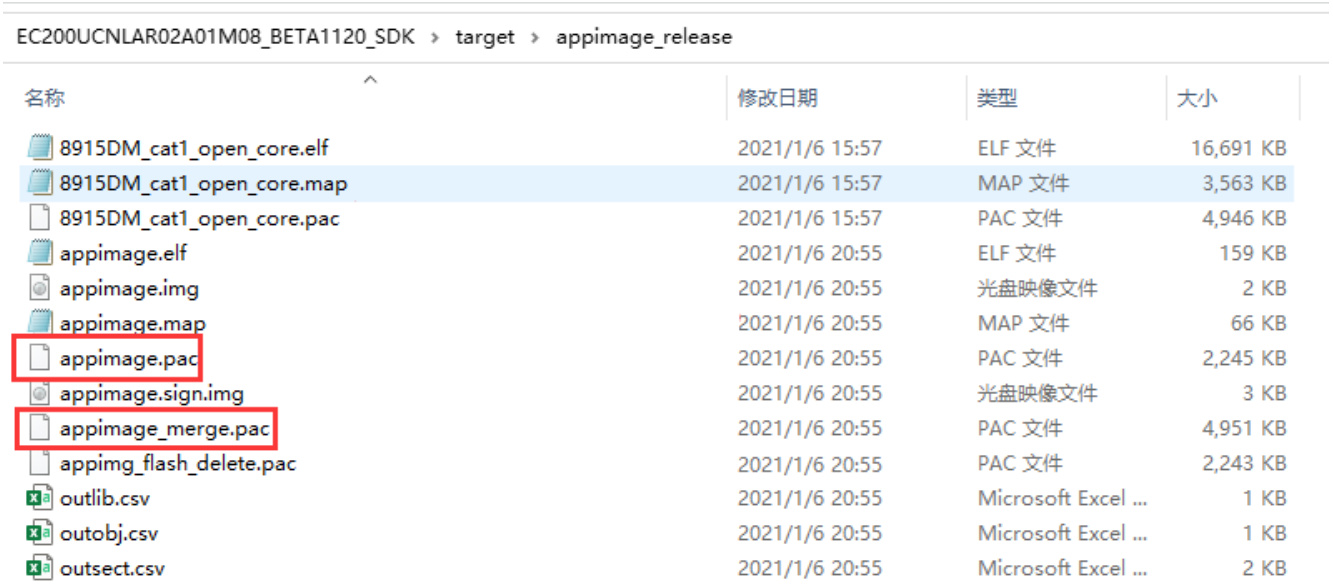

图 10：APP SDK 编译输出文件

4.6. 烧录固件包

根据使用的 SDK 以及实际需要烧录编译生成的固件包至模块，重启模块后 Secure Boot 功能使能。秘钥会被写入 efuse 中，不可更改，后续该模块下载和启动均会执行验签操作，如果验签成功，可正常下载或启动；如果验签失败，则无法下载或启动。以此确保模块中运行的是用户的代码，而不是未经合法签名或恶意修改的软件。

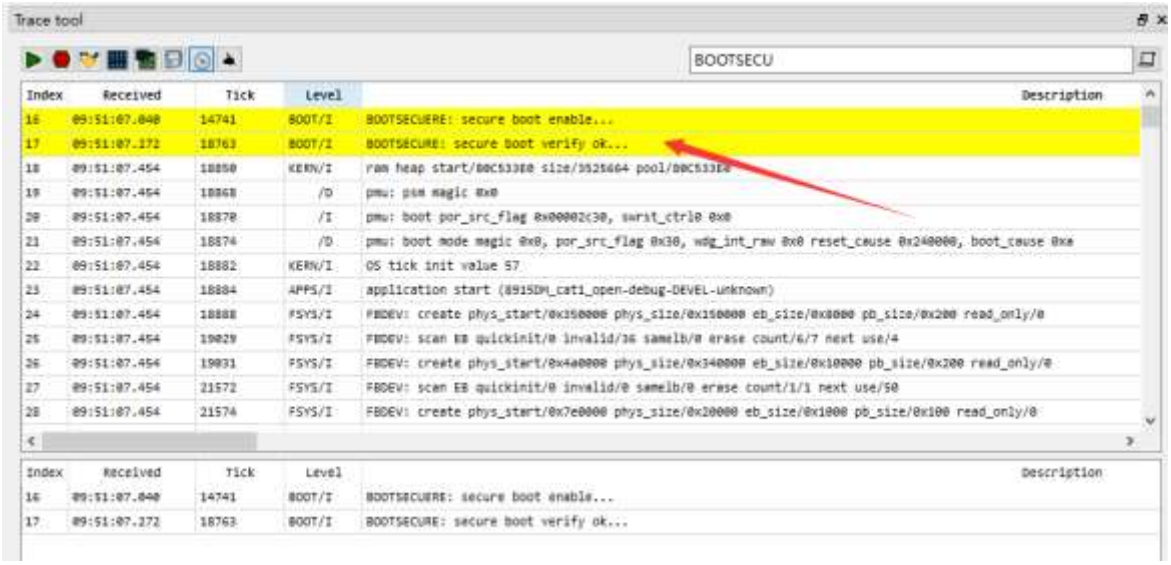

图 11：验签成功

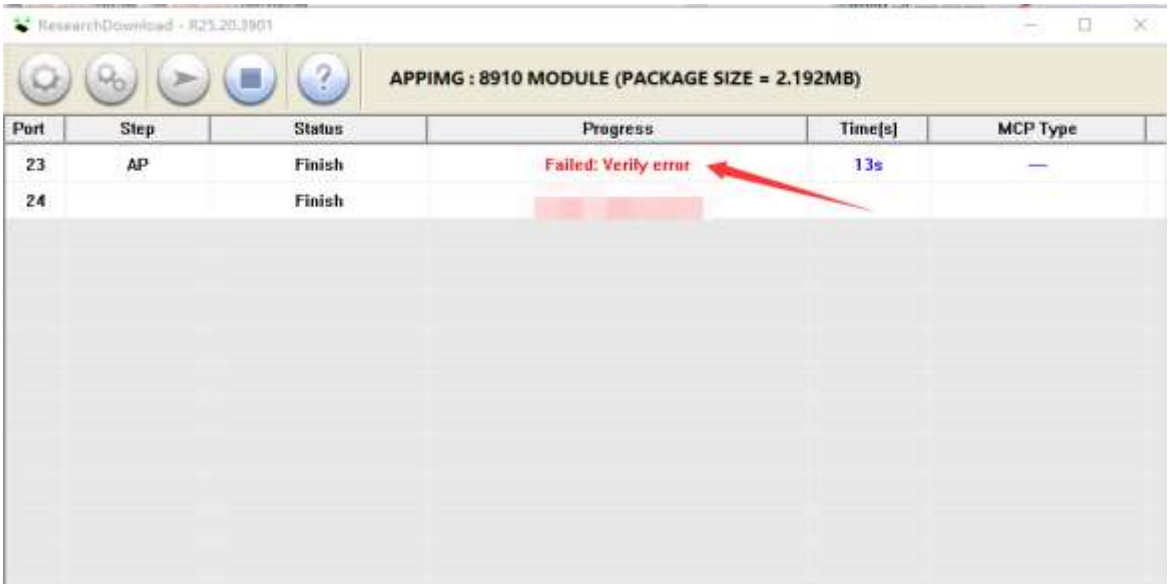

图 12：验签失败

5 附录 参考文档及术语缩写

表 2：参考文档

文档名称
[1] Quectel_ECx00U&EGx00U 系列_QuecOpen_CSDK_快速开发指导

表 3：术语缩写

缩写	英文全称	中文全称
ASCII	American Standard Code for Information Interchange	美国信息交换标准码
IoT	Internet of Things	物联网
RAM	Random Access Memory	随机存储器
RTOS	Real-Time Operating System	实时操作系统
SDK	Software Development Kit	软件开发工具包
UART	Universal Asynchronous Receiver/Transmitter	通用异步收发传输器
USB	Universal Serial Bus	通用串行总线