



ASR582X_Flash 加密及安全启动 使用指南

文档版本 1.1.0

发布日期 2023-12-22

版权所有 © 2022 翱捷科技

关于本文档

本文档旨在介绍 ASR IoT 芯片启用 Flash 加密及安全启动功能。

读者对象

本文档主要适用于以下工程师：

- 单板硬件开发工程师
- 软件工程师
- 技术支持工程师

产品型号

本文档适用于 ASR IoT 582X 系列 Wi-Fi Combo 芯片。

版权公告

版权归 © 2022 翱捷科技股份有限公司所有。保留一切权利。未经翱捷科技股份有限公司的书面许可，不得以任何形式或手段复制、传播、转录、存储或翻译本文档的部分或所有内容。

商标声明



ASR、翱捷和其他翱捷商标均为翱捷科技股份有限公司的商标。

本文档提及的其他所有商标名称、商标和注册商标均属其各自所有人的财产，特此声明。

免责声明

翱捷科技股份有限公司对本文档内容不做任何形式的保证，并会对本文档内容或本文中介绍的产品进行不定期更新。

本文档仅作为使用指导，本文的所有内容不构成任何形式的担保。本文档中的信息如有变更，恕不另行通知。

本文档不负任何责任，包括使用本文档中的信息所产生的侵犯任何专有权行为的责任。

防静电警告

静电放电（ESD）可能会损坏本产品。使用本产品进行操作时，须小心进行静电防护，避免静电损坏产品。

翱捷科技股份有限公司

地址：上海市浦东新区科苑路399号张江创新园10号楼9楼 邮编：201203

官网：<http://www.asrmicro.com/>

文档修订历史

| 日期 | 版本号 | 发布说明 |
|---------|--------|--------------------|
| 2023.9 | V1.0.0 | 首次发布。 |
| 2023.12 | V1.1.0 | 重构加签工具后更新安全启动部分内容。 |

目录

| | |
|--|----|
| 1. 概述..... | 1 |
| 2. Flash 加密 | 2 |
| 2.1 Flash 加密指令介绍 | 2 |
| 2.1.1 获取 Flash 加密状态..... | 2 |
| 2.1.2 写入 Flash 加密密钥..... | 2 |
| 2.1.3 使能 Flash 加密功能..... | 3 |
| 2.2 Flash 加密验证流程 | 3 |
| 2.2.1 硬件环境准备 | 3 |
| 2.2.2 进入 download 模式 | 4 |
| 2.2.3 写入 Flash 密钥 | 4 |
| 2.2.4 使能 Flash 加密 | 5 |
| 3. SecureBoot..... | 6 |
| 3.1 SecureBoot 指令介绍..... | 6 |
| 3.1.1 获取 SecureBoot 状态 | 6 |
| 3.1.2 写入 SecureBoot 密钥 | 6 |
| 3.1.3 读取 SecureBoot 密钥 | 7 |
| 3.1.4 使能 SecureBoot 功能 | 7 |
| 3.2 SecureBoot 固件准备..... | 7 |
| 3.2.1 SDK 版本及 RSA 加签工具在 SDK 中位置说明。 | 7 |
| 3.2.2 加签固件及密钥的生成及注意事项。 | 7 |
| 3.3 SecureBoot 验证流程..... | 8 |
| 3.3.1 硬件环境准备 | 8 |
| 3.3.2 进入 download 模式 | 8 |
| 3.3.3 写入 SecureBoot 密钥 | 9 |
| 3.3.4 使能 SecureBoot 功能 | 10 |
| 3.3.5 烧录加签固件 | 11 |

1.

概述

本文旨在介绍芯片启用 Flash 加密及 Secureboot 功能方法及步骤。

注意：两者同时启用时需先启用 Flash 加密功能再启用 Secureboot 功能、烧录加签固件。

ASR Confidential

2. Flash 加密

2.1 Flash 加密指令介绍

2.1.1 获取 Flash 加密状态

| | |
|----|---|
| 命令 | flashENC status |
| 功能 | 读取当前 Flash 加密状态及 Key 值反馈 |
| 响应 | 格式: <enable disable>=<efuse[0x80]>, <efuse[0x7C,,,0x7F]> 说明: efuse[0x80]中二进制中 1 的个数为奇数为 enable, 否则为 disable |
| 示例 | 例: 未加密 flashENC status disable=00, 0x00000000 >>> 例: 已加密 flashENC status enable=01, 0x12345678 >>> |

2.1.2 写入 Flash 加密密钥

| | |
|----|---|
| 命令 | flashENC wrkey <val> |
| 功能 | 向 efuse 中写入 flash 加密密钥, 必须写入。 注: key 只能写入 1 次, 不能修改, 谨慎写入 |
| 参数 | 说明: (1) 将 16 进制 value 值写入 efuse 地址[0x7C...0x7F], 例如 0x12345678 (2) 写前检查 efuse[0x7c...0x7f]是否为空, 为空可写否则失败 (不显示值) (3) 写之后读 efuse 检查, 如果和写入值一致则提示成功, 否则提示失败 |
| 响应 | OK, FAIL:-1, 不为空 FAIL:-2, 写失败 |
| 示例 | 例: 第一次写入 flashENC wrkey 0x12345678 OK >>> 例: 重复写入 flashENC 0x12345678 >>> |

2.1.3 使能 Flash 加密功能

| | |
|----|---|
| 命令 | flashENC [enable/ENABLE] [<val> <~val>] |
| 功能 | <p>格式：</p> <p>(1) flashENC enable //直接在 0x80 位置写 0x01</p> <p>(2) flashENC ENABLE 0x3F 0xC0</p> <p>说明：</p> <p>(1) 使能 flash 加密的功能， [<val> <~val>]参数是可选的，~val 是 val 的取反，做检验用。</p> <p>(2) 0x80 位置的数值二进制 bit1 个数为奇数时表示开启，为偶数时表示关闭，可以多次使能和失能。</p> |
| 参数 | addr: 需要校验的起始地址 size: 需要校验的长度 |
| 响应 | <p>OK,</p> <p>FAIL: -1, key 为 null</p> <p>FAIL: -2, 命令校验错误</p> <p>FAIL: -3, 已开启功能</p> <p>FAIL: -4, 写失败</p> <p>FAIL: -5, 参数不合法</p> |
| 示例 | <p>例：第一次使能</p> <p>flashENC enable</p> <p>OK</p> <p>例：已使能</p> <p>flashENC enable</p> <p>FAIL: -3</p> <p>>>></p> |

2.2 Flash 加密验证流程

2.2.1 硬件环境准备

- (1)、芯片 UART1 接电脑
- (2)、芯片 SeL 引脚接 3.3V
- (3)、系统上电

2.2.2 进入 download 模式

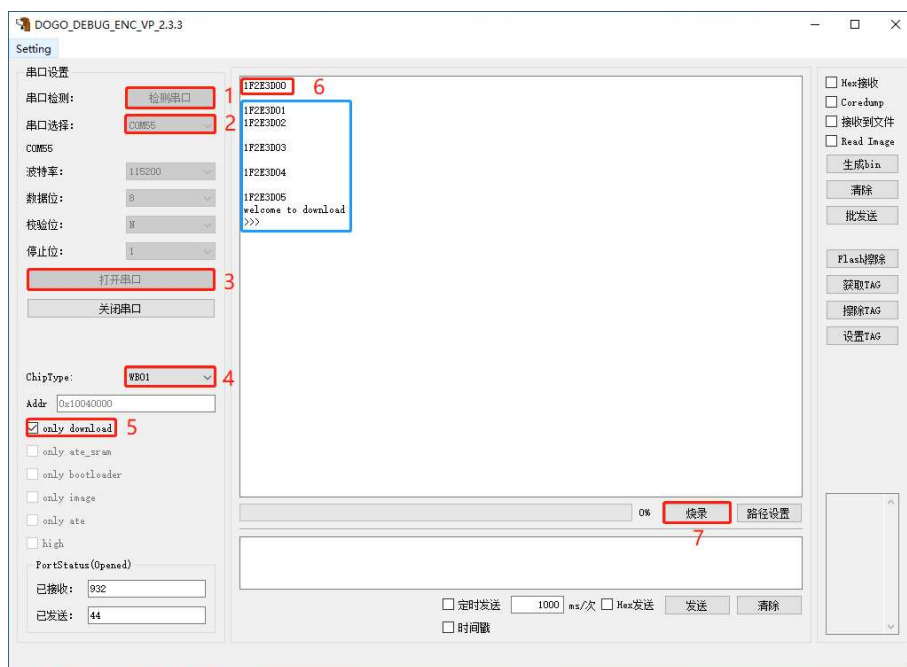


图 2.2.2-1 芯片进入 download 模式

- (1)、硬件接好后，打开 DOGO 工具。点击检测串口、选择对应串口号并打开串口，如图 1-3 步。
- (2)、ChipType 下拉选择对应芯片型号，勾选 only download 选项框，如图 4-5 步。
- (3)、芯片上电，待显示窗口打印“1F2D3E00”字样，点击烧录按钮，如图 6-7 步。
- (4)、显示窗口打印“welcome to download”字样表示已进入 download 模式。

2.2.3 写入 Flash 密钥

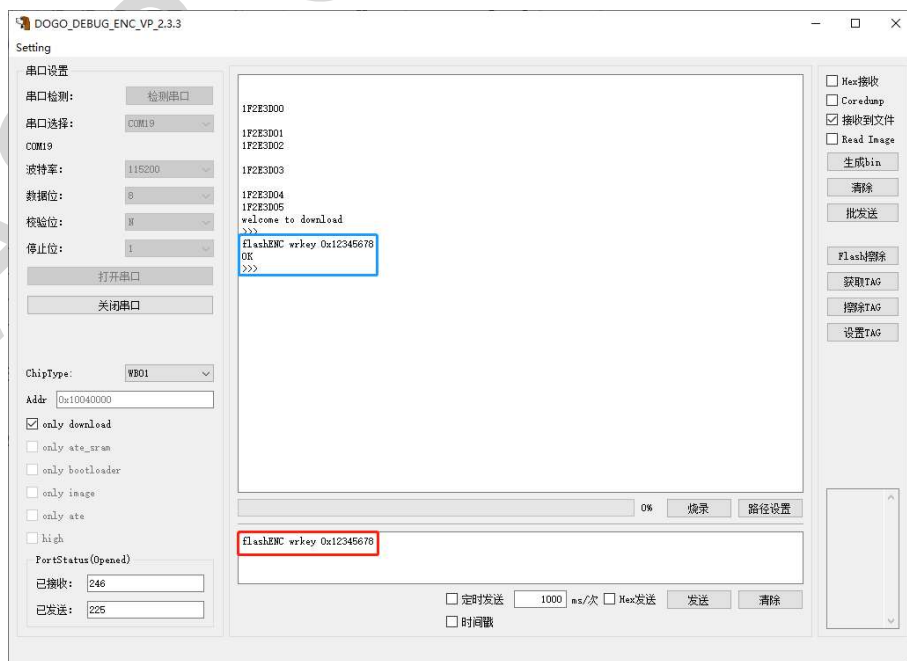


图 2.2.2-2 写入 Flash 加密密钥

- (1)、发送窗口写入发送密令及加密密钥值，例“flashENC wrkey 0x12345678”。
- (2)、写入密钥后显示窗口回显“OK”表示密钥写入成功。

2.2.4 使能 Flash 加密

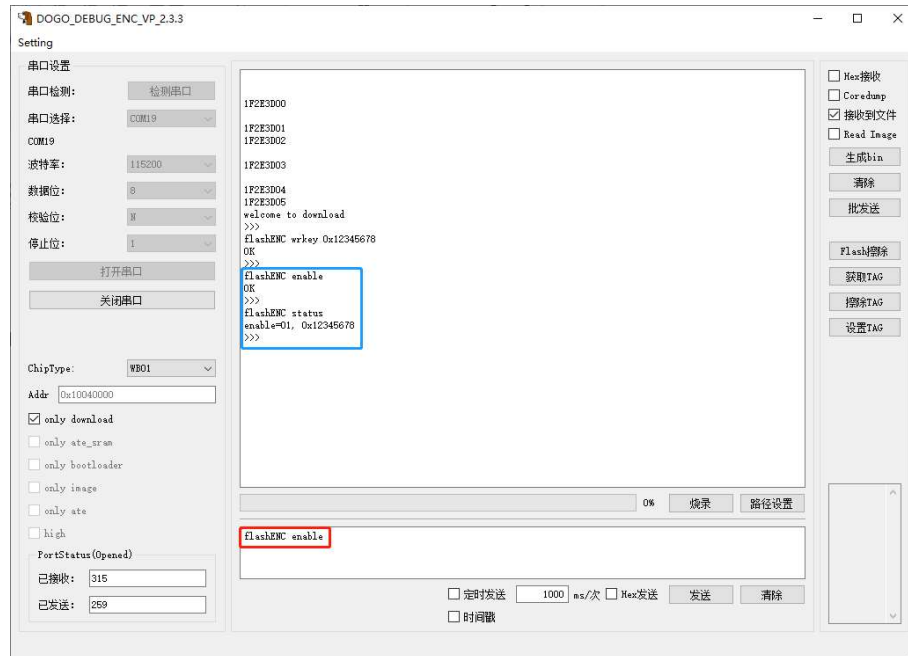


图 2.2.2-3 使能 Flash 加密

- (1)、发送窗口写入使能 Flash 加密指令“flashENC enable”。
- (2)、发送使能 Flash 加密指令后显示窗口回显“OK”表示使能成功。
- (3)、此时可通过状态查询指令“flashENC status”查询状态。

3. SecureBoot

3.1 SecureBoot 指令介绍

3.1.1 获取 SecureBoot 状态

| | |
|----|--|
| 命令 | secureBoot status |
| 功能 | 读取当前芯片 SecureBoot 状态 |
| 响应 | OK, ENABLE OK, DISABLE |
| 示例 | 例：未加密 flashENC status disable=00, 0x00000000 >>> 例：已加密 secureBoot status OK, ENABLE >>> |

3.1.2 写入 SecureBoot 密钥

| | |
|----|--|
| 命令 | secureBoot wrkey <val> |
| 功能 | 写入安全启动密钥, <val> 为 32 字节的 16 进制的字符串数据, efuse[0x3C...0x5C] 注: key 只能写入 1 次, 不能修改, 谨慎写入 |
| 参数 | 例: (1) secureBoot wrkey 46562ae46f1343a8fa3d1faf3cb0c549f8f8e63fec06e83ccbf7e314ce18a577 (2) key 数据来源于加签工具 out 目录 prim_key_hash.txt 文件中(0x46562ae4, 0x6f1343a8, 0xfa3d1faf, 0x3cb0c549, 0xf8f8e63f, 0xec06e83c, 0xcbf7e314, 0xce18a577) |
| 响应 | OK FAIL: -1, 不为空 FAIL: -2, 参数异常 (长度 32 字节, 64 个字符) FAIL: -3, 写入失败 (检测写进入和读出来是否一致) |
| 示例 | 例: 重复写入 secureBoot wrkey DB100D66632C25AB3A94BE4AF41E55B94D6C7606C1A59977376A3EFFF3CF6B83 FAIL: -1 >>> |

3.1.3 读取 SecureBoot 密钥

| | |
|----|---|
| 命令 | secureBoot rdkey |
| 功能 | 读安全启动密钥, efuse[0x3C~0x5C]的 32 字节值, 以 16 进制字符串形式打印 |
| 响应 | 返回 32 字节的数据, 16 进制形式字符串 (64 个字符) |
| 示例 | 例: secureBoot rdkey DB100D66632C25AB3A94BE4AF41E55B94D6C7606C1A59977376A3EFFD3CF6B83 >>> |

3.1.4 使能 SecureBoot 功能

| | |
|----|--------------------------------------|
| 命令 | secureBoot enable |
| 功能 | 使能 SecureBoot 功能 |
| 响应 | OK FAIL: -1 FAIL: -2 |
| 示例 | 例: secureBoot enable OK >>> |

3.2 SecureBoot 固件准备

3.2.1 SDK 版本及 RSA 加签工具在 SDK 中位置说明。

本文档后续对加签固件生成的说明均基于 1.9.0 及以上版本 SDK。

RSA 加签工具位于 sdk 目录 build/tools/gen_sign 下。

```
jixu@ZJLAB58 MINGW64 /e/project_codes/xuji_code/master_combo/freertos/build/tools/gen_sign (master)
$ ls
gen_sign  gen_sign.exe*  readme.md
```

图 3.2.2-1 加签工具目录

3.2.2 加签固件及密钥的生成及注意事项。

加签固件及密钥的生成, 目前是伴随着编译过程自动完成, 无需用户手动去调用生成。

```
jixu@ZJLAB58 MINGW64 /e/project_codes/xuji_code/master_combo/freertos/build (master)
$ make TARGET=duet_demo ic_type=5822s SECUREBOOT=1
```

图 3.2.2-1 编译 secureboot 固件

如上图，在编译命令中加 SECUREBOOT=1 命令，固件编译完成后会自动生成加签固件、加签 ota 固件、加签 bootloader 固件及用于写入 otp 的 pub_key.hash.txt（公钥 hash 值）等文件。

```
jixu@ZJLAB58 MINGW64 /e/project_codes/xuji_code/master_combo/freertos/build/out/duet_demo/bin (master)
$ ls
ASRBOOTLOADER-58XX-MX-V1.1.1-2M-UART1-SECUREBOOT-202311021647.rsa.signed.bin
duet_demo.bin
duet_demo.elf
duet_demo.hex
duet_demo.map
duet_demo.rsa.signed.bin
duet_demo.rsa.signed_ota.bin
pub_key.hash.txt
```

图 3.2.2-2 加入 SECUREBOOT=1 后编译输出文件

同时在首次调用 SECUREBOOT=1 编译命令时，会在 build/build_rules/duet_demo 下生成名为 rsa 的目录，并将用于加签的密钥文件存放在这个目录下。后续用户再次编译时工具会自动检测该位置是否存在合法的密钥文件，如果有则不再重新生成，继续使用该目录下的密钥文件进行固件签名。

```
jixu@ZJLAB58 MINGW64 /e/project_codes/xuji_code/master_combo/freertos/build/build_rules/project/duet_demo/rsa (master)
$ ls
image_cert.bin      key_cert_Cert.txt  priv_key_1.pem     sb_content_cert.log
image_cert_Cert.txt passphrase1.txt     priv_key_2.pem     sb_key_cert.log
key_cert.bin        passphrase2.txt     pub_key.hash.txt
```

图 3.2.2-3 首次采用 SECUREBOOT=1 编译生成的密钥文件

注：

- 芯片的 SECUREBOOT 使能是一个不可逆过程。
- 请用户妥善备份及保护 build/build_rules/duet_demo/rsa 目录下的密钥文件，丢失或者被篡改可能导致已经使能 SECUREBOOT 的设备无法再次更新固件。
- 在有需要对加签密钥进行更新的情况下，建议将 build/build_rules/duet_demo/rsa 目录移动到其他地方备份，非必要不要轻易删除旧密钥文件夹。
- 当前 SECUREBOOT 开启的情况下只支持 remapping 一种升级方式。

3.3 SecureBoot 验证流程

3.3.1 硬件环境准备

- (1)、芯片 UART1 接电脑
- (2)、芯片 SeL 引脚接 3.3V
- (3)、系统上电

3.3.2 进入 download 模式

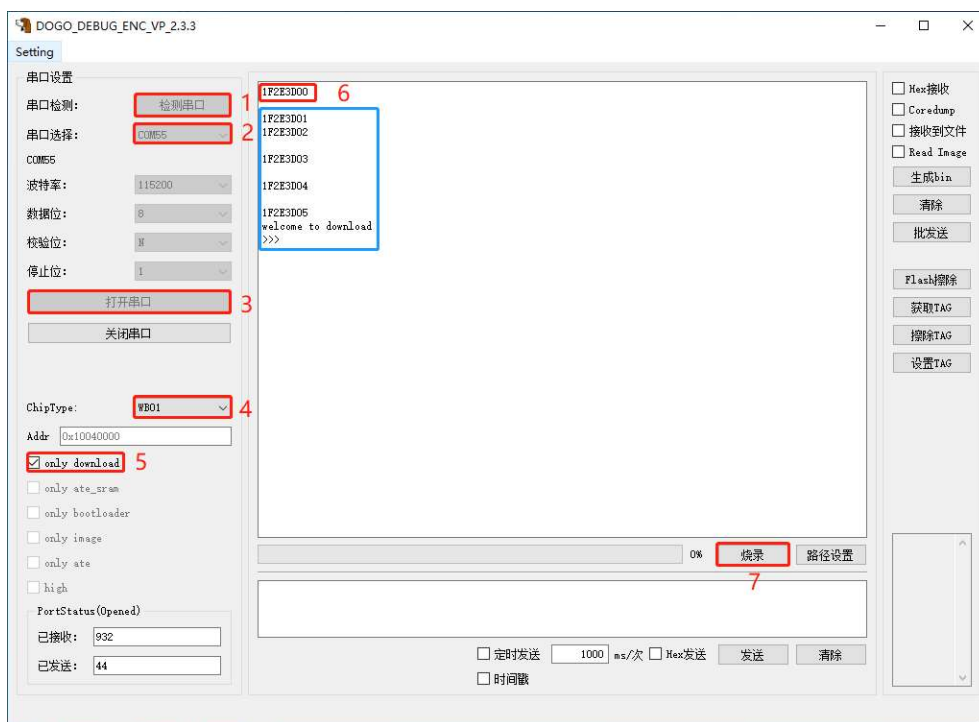


图 3.3.2-1 芯片进入 download 模式

- (1)、硬件接好后，打开 DOGO 工具。点击检测串口、选择对应串口号并打开串口，如图 1-3 步。
- (2)、ChipType 下拉选择对应芯片型号，勾选 only download 选项框，如图 4-5 步。
- (3)、芯片上电，待显示窗口打印“1F2D3E00”字样，点击烧录按钮，如图 6-7 步。
- (4)、显示窗口打印“welcome to download”字样表示已进入 download 模式。

3.3.3 写入 SecureBoot 密钥

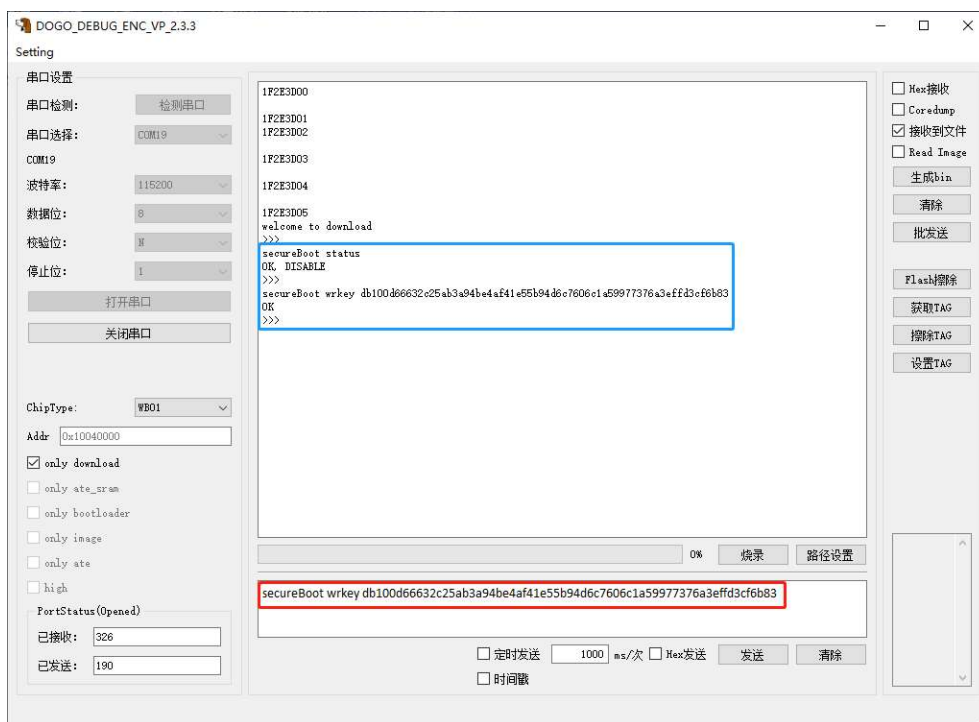


图 3.3.3-1 芯片写 SecureBoot 命令

(1)、发送窗口写入 secureboot 密钥，例“secureBoot wrkey db100d66632c25ab3a94be4af41e55b94d6c7606c1a59977376a3effd3cf6b83”。

注：这里写入的密钥即为，3.2.2-2 图中所列的 pub_key.hash.txt 文件里的内容。

```
jixu@ZJLAB58 MINGW64 /e/project_codes/xuji_code/master_combo/freertos/build/out/duet_demo/bin (master)
$ ls
ASRBOOTLOADER-58XX-MX-V1.1.1-2M-UART1-SECUREBOOT-202311021647.rsa.signed.bin
duet_demo.bin
duet_demo.elf
duet_demo.hex
duet_demo.map
duet_demo.rsa.signed.bin
duet_demo.rsa.signed_ota.bin
pub_key.hash.txt
```

图 3.3.3-2 secureBoot wrkey 写入秘钥文件

(2)、写入密钥后显示窗口回显“OK”表示密钥写入成功。

注：写入前可以使用“secureBoot status”指令获取下当前状态。

3.3.4 使能 SecureBoot 功能

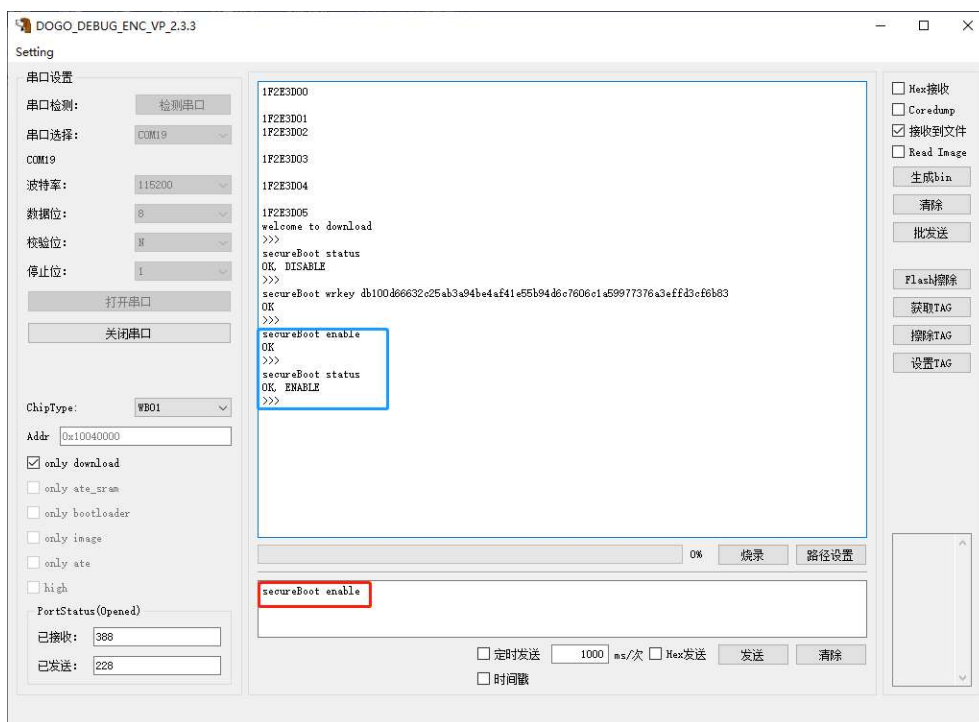


图 3.3.4-3 使能 SecureBoot 功能

- (1)、发送窗口写入使能 SECUREBOOT 指令“secureBoot enable”。
- (2)、发送使能 SECUREBOOT 指令后显示窗口回显“OK”表示使能成功。
- (3)、此时可通过状态查询指令“secureBoot status”查询状态。

3.3.5 烧录加签固件

芯片使能 SECUREBOOT 后需要烧录经过加签的 bootloader 及应用固件，否则芯片将启动失败。烧录地址和非加签固件一致。

```
jixu@ZJLAB58 MINGW64 /e/project_codes/xuji_code/master_combo/Freertos/build/out/duet_demo/bin (master)
$ ls
ASRBOOTLOADER-58XX-MX-V1.1.1-2M-UART1-SECUREBOOT-202311021647.rsa.signed.bin
duet_demo.bin
duet_demo.elf
duet_demo.hex
duet_demo.map
duet_demo.rsa.signed.bin
duet_demo.rsa.signed_ota.bin
pub_key.hash.txt
```

图 3.3.5-1 烧录时选择编译输出目录的加签固件