实验手册： Matter 设备安全证书 DAC、PAI 及 CD 的生成和烧录

Matter 设备在 Commissioning 配网的过程中，有一个关键步骤是 Commissioner 需要对 Matter 设备进行设备认证过程，以验证 Matter 设备的合法性以及保证其是经过 Matter 测试认证。设备认证过程主要包括对 Matter 设备的 DAC/PAI/PAA 证书链的校验、对 CD 的签名校验和对 Matter 设备的 Product Id、Vendor Id 的验证等。因此，这些安全相关参数都需要设备厂商在工厂生产的时候烧录到设备里面。本次实验帮助开发者掌握如何使用 Silicon Labs 的解决方案对相应参数进行生成和烧录。

实验目的

- 理解 DAC、PAI、CD 的概念
- 掌握如何生成测试用的 DAC、PAI 证书及 CD 文件
- 掌握如何使用 Silicon Labs 的脚本文件将 DAC、PAI 证书及 CD 烧录到 Matter 设备

Table of Contents

1. 预备知识

本实验中涉及的专有名词及其解释：

- DAC(Device Attestation Certificate): 设备认证证书。由 PAI 机构颁发，用于在 Commissioning 的过程中验证 Matter 设备的合法性。工厂生产的时候需要将 DAC 烧录到 Matter 设备中。
- PAI(Product Attestation Intermediate): 产品认证中间机构。用于给 Matter 设备颁发 DAC 证书。PAI 证书由 PAA 机构颁发，在 Commissioning 的过程中会验证 DAC/PAI/PAA 证书链的合法性。工厂生产的时候需要将 PAI 证书烧录到 Matter 设备中。
- PAA(Product Attestation Authority): 产品认证机构。用于颁发 PAI 证书。PAA 证书由 PAA 机构自签名并颁发，其存储在 DCL（分布式合规数据库）中，Commissioner 需要定期从 DCL 中获取并更新 PAA 证书列表。
- CD(Certification Declaration): 认证声明。Matter 设备在测试实验室通过 Matter 测试认证之后，由 CSA 联盟向厂商颁发的证明该设备已经通过 Matter 认证的签名文件。

2. 实验目的

通过本次《Matter 设备安全证书 DAC、PAI 及 CD 的生成和烧录》实验，开发者可以理解并掌握以下内容：

- 理解 DAC、PAI、CD 的概念。
- 掌握如何生成测试用的 DAC、PAI 证书及 CD 文件。
- 掌握如何使用 Silicon Labs 的脚本文件将 DAC、PAI 证书及 CD 烧录到 Matter 设备。

3. 实验内容

Matter 设备在 Commissioning 配网的过程中，有一个关键步骤是 Commissioner 需要对 Matter 设备进行设备认证过程，以验证 Matter 设备的合法性以及保证其是经过 Matter 测试认证。设备认证过程主要包括对 Matter 设备的 DAC/PAI/PAA 证书链的校验、对 CD 的签名校验和对 Matter 设备的 Product Id、Vendor Id 的验证等。因此，这些安全相关参数都需要设备厂商在工厂生产的时候烧录到设备里面。本次实验帮助开发者掌握如何使用 Silicon Labs 的解决方案对相应参数进行烧录。

本次实验内容包括编译 chip-cert 工具，利用该工具生成测试用的 PAI 证书、PAI 私钥和 CD 文件，并使用 Silicon Labs 提供的 creds.py 脚本生成 DAC 证书，最后将 DAC、PAI、CD 等相关信息烧录到 Matter Breakout 开发板的 EFR32MG24 芯片内部 Flash 区域（最后一个 Flash page），以供 Matter 设备在 Commissioning 配网时使用对应的安全信息进行设备认证过程。

本次实验中使用的 DAC、PAI 证书、PAI 私钥以及 CD 都是通过 chip-cert 工具和 creds.py 脚本生成的测试版本，设备厂商在实际生产的时候需要经过相应途径获取正式版的 DAC、PAI 证书、PAI 私钥以及 CD。本次实验重点在于阐述设备安全认证相应的原理以及安全信息的烧录方法。

4. 实验准备

4.1 硬件准备

- 安装 Ubuntu 虚拟机的电脑一台
- MG24 Breakout 开发板一块
- Type-C 转 USB 线一条
- J-Link 烧录器一台
- 安装 Chip-Tool 的树莓派一台（可选）
- EFR32MG21 USB Dongle 一块（可选）

4.2 软件准备

- Virtual Box 虚拟机
- Ubuntu 20.04 镜像
- Silicon Labs Matter SDK
- Linux Commander 工具
- 安装 Linux JLink 驱动
- 安装 gcc-arm-none-eabi 交叉编译工具

4.2.1 下载 Silicon Labs Matter SDK 源码

- 当 Ubuntu-64 位系统虚拟机安装成功后，需要更新或安装系统工具、下载 Matter SDK 源码、构建 Matter 编译环境。如果使用本次实验中 Silicon Labs 提供的 Ubuntu 镜像，以下步骤已经完成，无需再次搭建。
    - 更新或安装系统工具。

```
> sudo apt update

> sudo apt upgrade -y

> sudo apt-get install git gcc g++ pkg-config libssl-dev libdbus-1-dev libglib2.0-dev libavahi-client-dev ninja-build python3-venv python3-dev python3-pip unzip libgirepository1.0-dev libcairo2-dev libreadline-dev
```

```
> sudo reboot
```

- 下载 Matter SDK 源码，更新 submodule，构建 Matter 编译环境。

```
> git clone https://github.com/SiliconLabs/matter.git

> cd matter

> git checkout release_1.0.2-1.0

> git pull

> ./scripts/checkout_submodules.py --shallow --recursive --platform efr32

> git submodule update --init --recursive

# 以下步骤需要使用 VPN 网络

> source scripts/bootstrap.sh
```

- 若已下载 Matter SDK 源码，则更新到对应的 Branch 分支。

```
> git fetch

> git checkout release_1.0.2-1.0

> git submodule update --init --recursive

> source scripts/bootstrap.sh
```

- 可能存在代码同步更新失败的问题，导致固件或者工具编译不过，可先保存个人代码，执行如下命令强制同步。（下例强制同步 release_1.0.2-1.0 分支，仅供参考）

```
> git fetch --all

> git reset --hard release_1.0.2-1.0

> git pull
```

4.2.2 安装 Linux Commander 工具。

- 在 Ubuntu 中下载 Commander 安装包，并解压。Commander 操作手册请参考文档 UG162: Simplicity Commander Reference Guide。

```
> lesun@Ubuntu20:~/Downloads/SimplicityCommander-Linux$ tar jxvf
Commander_linux_x86_64_1v14p2b1232.tar.bz

# 可以在 commander/目录下看到 commander 执行文件
```

```
> cd ./commander

# 输入 commander 命令即可打开 Commander 工具的图形界面

> commander
```

- 将 Commander 路径添加到环境变量：

```
> sudo vim /etc/profile

# 将 Commander 路径添加添加到文件中的最后一行

> export PATH=$PATH:~/Downloads/SimplicityCommander-Linux/commander

# 生效添加的环境变量

> source /etc/profile

 # 在任何目录下，输入 commander 即可打开 Commander 工具
```

4.2.3 安装 JLink 驱动

- 当前是 Ubuntu 64 位系统，在 Ubuntu 中下载 JLink 驱动，选择 JLink_Linux_V784b_x86_64.deb 驱动包下载到~/Downloads 目录，并使用如下命令安装：

```
> cd ~/Downloads/

> sudo dpkg -i JLink_Linux_V784_x86_64.deb
```

4.2.4 安装 gcc-arm-none-eabi 交叉编译工具

- 当前是 Ubuntu-64 位系统，在 Ubuntu 中下载 gcc-arm-none-eabi 到~/Downloads 目录，并使用如下命令解压：

```
> cd ~/Downloads/

> tar -xvf gcc-arm-11.2-2022.02-x86_64-arm-none-eabi.tar.xz
```

5. 实验步骤

5.1 在 Ubuntu 虚拟机中编译生成 chip-cert 工具

- 执行如下命令：

```
> cd matter

> source ./scripts/activate.sh

> gn gen out/host

> ninja -C out/host chip-cert
```

- 若需了解命令参数的详细说明，可使用以下命令查看：

```
> ./out/host/chip-cert gen-att-cert --help

> ./out/host/chip-cert gen-cd –-help
```

5.2 利用 chip-cert 工具生成测试 PAI 证书、PAI 私钥和 CD

- 执行如下命令：

```
> cd matter

# 生成测试 PAI 证书和 PAI 私钥

> ./out/host/chip-cert gen-att-cert --type i --subject-cn "Matter Test PAI" --subject-vid 1049 --valid-from "2023-1-15 14:30:00" --lifetime 4294967295 --ca-key credentials/test/attestation/Chip-Test-PAA-NoVID-Key.pem --ca-cert credentials/test/attestation/Chip-Test-PAA-NoVID-Cert.pem --out-key ./Silabs-Test-PAI-Key.pem --out ./Silabs-Test-PAI-Cert.pem
```

- 参数说明：
  - --type i 是指生成 PAI 证书类型。
  - --subject-cn 是指证书主题的 DN 名。
  - --subject-vid 是指证书主题的 Verdor ID。
  - --valid-from 是指证书有效期的开始时间。
  - --lifetime 是指证书的有效期，以天为单位，4294967295 代表证书永远有效，不会过期。
  - --ca-key 是指给颁发证书进行签名的 CA 机构的私钥。

- --ca-cert 是指给颁发证书进行签名的 CA 机构的证书。
- --spake2_iteration 和--spake2_salt，它们是生成 spake2_verifier 的参数，可使用命令：./out/host/spake2p gen-verifier --help 查看其详细说明。
- --out-key 是指生成的证书里面包含的公钥对应的私钥。
- --out 是指生成的证书。

- 在当前目录查看生成的 PAI 证书和 PAI 私钥，在此打印如下信息只为实验演示目的。实际生产中，请务必保存好 PAI 私钥，避免泄露。

```
lesun@Ubuntu20:~/matter$ ls Silabs-Test-PAI-*

Silabs-Test-PAI-Cert.pem  Silabs-Test-PAI-Key.pem


lesun@lesun-Ubuntu20:~/matter$ cat Silabs-Test-PAI-Cert.pem

-----BEGIN CERTIFICATE-----

MIIBpzCCAU6gAwIBAgIIcaR95W6tiMEwCgYIKoZIzj0EAwIwGjEYMBYGA1UEAwwP

TWF0dGVyIFRlc3QgUEFBMCAXDTIzMDExNTE0MzAwMFoYDzk5OTkxMjMxMjM1OTU5

WjAwMRgwFgYDVQQDDA9NYXR0ZXIgVGVzdCBQQUkxFDASBgorBgEEAYKifAIBDAQx

MDQ5MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEDOyIMTyEcJUpl3TM4X61berp

hqVWuNksa09/3A1a1GUwQl2IXssqPzl7okOU7GvoZ9XoSa49Ia6UiE5i4vQ38aNm

MGQwEgYDVR0TAQH/BAgwBgEB/wIBADAOBgNVHQ8BAf8EBAMCAQYwHQYDVR0OBBYE

FHPQHecQC6RDOaWcS43TvPLkg4HGMB8GA1UdIwQYMBaAFHhc5wW4a49Ob8eTqmDL

Q+ppaILVMAoGCCqGSM49BAMCA0cAMEQCIE/vcokjEKNi3TCDB8kc1O7ec8aZC2FG

D/JBjabt26acAiBprik/jPrzu94JZHGQxLi+/gai7kGWltz5HEPTf4LYzA==

-----END CERTIFICATE-----

lesun@20:~/matter$ cat Silabs-Test-PAI-Key.pem

-----BEGIN EC PRIVATE KEY-----

MHcCAQEEIK26M1/zO8VvZvYlkZxmmh3A9+KTEVdeCb/R9zBHBNFnoAoGCCqGSM49

AwEHoUQDQgAEDOyIMTyEcJUpl3TM4X61berphqVWuNksa09/3A1a1GUwQl2IXssq

Pzl7okOU7GvoZ9XoSa49Ia6UiE5i4vQ38Q==

-----END EC PRIVATE KEY-----
```

- 再执行如下命令，在当前目录生成 Silabs-Test-CD.bin 文件。

```
# 生成测试 CD

> ./out/host/chip-cert gen-cd --cert ./credentials/test/certification-declaration/Chip-Test-CD-Signing-Cert.pem --key ./credentials/test/certification-declaration/Chip-Test-CD-Signing-Key.pem --out ./Silabs-Test-CD.bin --format-version 1 --vendor-id 1049 --product-id 5555 --device-type-id 0016 --certificate-id "ZIG0000000000000001" --security-level 0 --security-info 0 --version-number 0005 --certification-type 0
```

- 参数说明：
  - --cert 是指签名私钥对应的证书，里面包含公钥信息。
  - --key 是指私钥，用来对 CD 进行签名。
  - --out 是指生成的 CD 文件。
  - --format-version 是指 CD 格式版本号。
  - --vendor-id 是指 CD 对应的 Vendor ID。
  - --product-id 是指 CD 对应的 Product ID。
  - --device-type-id 是指 CD 对应的设备类型。
  - --certificate-id 是指 CD 对应的证书 ID。
  - --security-level 是指 CD 对应的安全等级。
  - --security-info 是指 CD 对应的安全信息。
  - --version-number 是指 CD 对应的版本号。
  - --certification-type 是指 CD 对应的证书类型，0 表示用于开发和测试的 CD。

5.3 将生成的测试 PAI 证书、PAI 私钥和 CD 拷贝到 silabs_examples/credentials/目录

- 执行如下命令：

```
> cp ./Silabs-Test-* ./silabs_examples/credentials/

> ls -l ./silabs_examples/credentials/


total 56
```

```
-rwxrwxr-x  1 lesun lesun  7968 1 月  10 15:26 creds.py
-rw-rw-r--  1 lesun lesun 11269 1 月  10 15:26 csa_openssl.cnf
drwxrwxr-x 19 lesun lesun  4096 1 月  10 15:26 device
-rw-rw-r--  1 lesun lesun   512 1 月  10 15:26 efr32_creds.tmpl
drwxrwxr-x  3 lesun lesun  4096 1 月  10 15:26 host
-rw-rw-r--  1 lesun lesun 10635 1 月  10 15:26 README.md
-rw-rw-r--  1 lesun lesun   233 1 月  16 10:50 Silabs-Test-CD.bin
-rw-rw-r--  1 lesun lesun   635 1 月  16 10:50 Silabs-Test-PAI-Cert.pem
-rw-rw-r--  1 lesun lesun   227 1 月  16 10:50 Silabs-Test-PAI-Key.pem
```

5.4 搭建 Silicon Labs 的 creds.py 脚本执行环境

- 执行如下命令：

```
> cd ./silabs_examples/credentials
> export BASE_SDK_PATH=~/matter/third_party/silabs/gecko_sdk
> export ARM_GCC_DIR=~/Downloads/gcc-arm-11.2-2022.02-x86_64-arm-none-eabi
```

5.5 修改 creds.py 脚本以适配 Matter Breakout 开发板的 JLINK 和 USB 转串口

5.5.1 修改 creds.py 脚本中的 Commander 命令，添加对“--device”的支持

```
# 将 python 脚本中执行 commander 命令的所有地方添都加'--device', mcu_family 参数：
# 在第 13 行添加
mcu_family = 'EFR32MG24'
# 第 69 行改为
execute('Flashing device app', [ "commander", "flash", device_bin, "--serialno", serial_num, "--device", mcu_family, ])
# 第 137 行到 140 行改为
execute('Erasing Page', ["commander", "device", "pageerase", "--range", "{}:+{}".format(hex(page.address), page.size), "--serialno", serial_num, "--device", mcu_family, ])
```

```
execute('Flashing PAI', ["commander", "flash", _pai_der, "--binary", "--address", hex(page.address + pai_offset), "--serialno", serial_num, "--device", mcu_family, ])

execute('Flashing DAC', ["commander", "flash", _dac_der, "--binary", "--address", hex(page.address + dac_offset), "--serialno", serial_num, "--device", mcu_family, ])

execute('Flashing CD', ["commander", "flash", cd_file, "--binary", "--address", hex(page.address + cd_offset), "--serialno", serial_num, "--device", mcu_family, ])

# 第 143 行改为

subprocess.run(["commander", "readmem", "--range", "{}:+{}".format(hex(page.address), end_offset), "--serialno", serial_num, "--device", mcu_family])
```

- 注：BRD2601B 的 mcu family 是 EFR32MG24，所以 mcu_family 设置为该值，Matter 设备还支持 EFR32MG12，例如 BRD4161A 等。该 mcu_family 变量需要设定为实际使用的芯片类型。

5.5.2 修改 host_creds.c 脚本内容，使其支持 Matter Breakout 板子的 USB 转串口

- 执行如下命令：

```
> cd ./silabs_examples/credentials/host

# 如果不支持 Vim 工具请自行安装，或者使用其它编辑方法

> vim host_creds.c
```

- 将 serial_port_open 函数改为如下代码：

```
# 宏 ADAPT_SELF_BRD2601B_UART 包含的代码为添加内容


#define ADAPT_SELF_BRD2601B_UART 1


static int serial_port_open(const char * port)
{
 struct termios tty;
 int fd = open(port, O_RDWR | O_NOCTTY);
 if (fd < 0)
 {
   return -1;
```

```
}


#if defined(ADAPT_SELF_BRD2601B_UART)
 tcflush(fd, TCIOFLUSH);
#endif


 if (tcgetattr(fd, &tty) != 0)
 {
   printf("Error %i from tcgetattr: %s\n", errno, strerror(errno));
   return -2;
 }


#if defined(ADAPT_SELF_BRD2601B_UART)
 printf("cfset for self brd2601b\n");


 speed_t baud = B115200;
 cfsetispeed(&tty, baud);
 cfsetospeed(&tty, baud);
 tty.c_cflag |= CLOCAL;              // ignore modem inputs
 tty.c_cflag |= CREAD;               // receive enable (a legacy flag)
 tty.c_cflag &= ~CSIZE;              // 8 data bits
 tty.c_cflag |= CS8;
 tty.c_cflag &= ~PARENB;               // no parity
 tty.c_cflag &= ~CSTOPB;
 //tty.c_cflag &= ~CRTSCTS;
 tty.c_iflag &= ~(BRKINT | INLCR | IGNCR | ICRNL | INPCK
                  | ISTRIP | IMAXBEL | IXON | IXOFF | IXANY);


 tty.c_iflag &= ~(IXON | IXOFF);
```

```
 tty.c_lflag &= ~(ICANON | ECHO | IEXTEN | ISIG);

 tty.c_oflag &= ~OPOST;

 (void) memset(tty.c_cc, _POSIX_VDISABLE, NCCS);

 tty.c_cc[VMIN] = 1;

 tty.c_cc[VTIME] = 0;


 if (tcsetattr(fd, TCSAFLUSH, &tty) != 0) {

   return -3;

 }

#else

 printf("cfset for wstk vcom\n");

 tty.c_cflag |= CREAD | CLOCAL; // Allows read, and ignore control lines

 tty.c_lflag &= ~ICANON; // Non-canonical mode

 tty.c_lflag &= ~ISIG;   // Ignore signal characters (INTR, QUIT, SUSP, or DSUSP)

 tty.c_iflag &= ~(INLCR | ICRNL);  // Disable mapping NL to CR-NL on input

 tty.c_cc[VTIME] = 10;   // Wait up to 1 second

 tty.c_cc[VMIN] = 0;     // Minimum number of characters for noncanonical

 // Save tty settings, also checking for error

 if (tcsetattr(fd, TCSANOW, &tty) != 0) {

   return -3;

 }

#endif

 return fd;

}
```

5.6 利用 creds.py 脚本生成 DAC 证书，并将 Matter 安全证书(DAC/PAI/CD)烧录到 Breakout 开发板

- 将 Matter Breakout 板子的 Type-C 的 USB 线和 J-Link 烧录器的 USB 线插入电脑的 USB 端口，确保两个 USB 都能在 Ubuntu 系统中识别到。

- 检查 J-Link 是否被 Ubuntu 识别到，输入 commander 命令，启动 Commander 工具，点击“Select Kits”，确保能识别到 J-Link 的 USB 端口，如下图 1 所示。如果识别不到 USB 端口，需要先解决该问题再进行下一步操作。

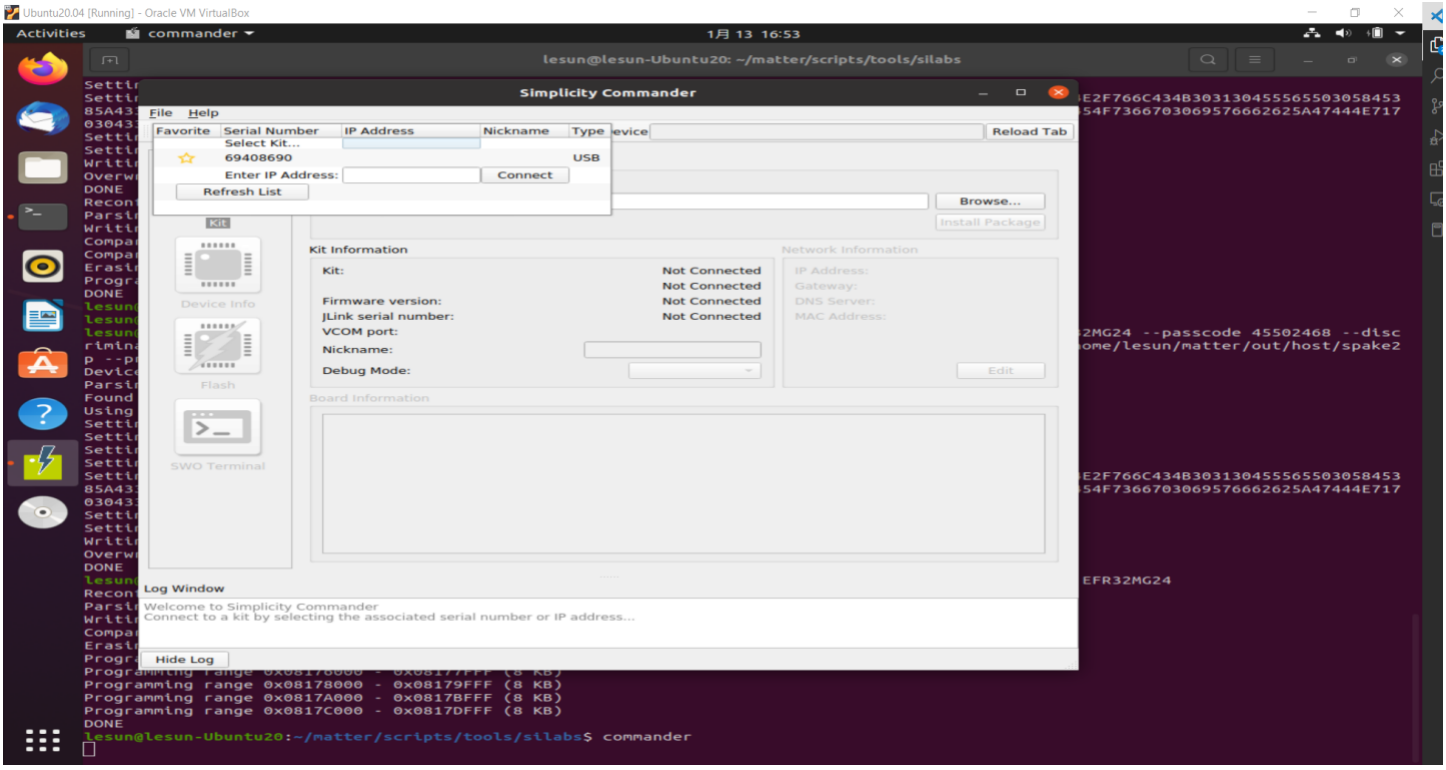

图 1 – 使用 commander 工具识别 J-Link 的 USB 端口

- 查看/dev/目录下是否有 ttyUSB0 设备

```
> ls /dev/ttyUSB*

/dev/ttyUSB0

> sudo chmod 777 /dev/ttyUSB0
```

- 执行如下命令：

```
# DAC 证书的生成和 Matter 安全证书(DAC/PAI/CD)的烧录

> python3 ./creds.py -p /dev/ttyUSB0 -S 69408690 -B brd2601b -N  "Matter Test DAC 01" -V 4169 -P 21845 -C Silabs-Test-PAI-Cert.pem -K Silabs-Test-PAI-Key.pem -D Silabs-Test-CD.bin
```

- 该脚本实现以下功能：
    - 编译 Host 和设备端应用程序。
    - 将设备端应用程序烧录到目标设备。

  - 将 PEM 格式的证书转化为 DER 格式。
  - 从目标设备中获取 CSR 文件。
  - 使用 OpenSSL 生成 DAC 证书。
  - 计算 DAC、PAI、CD 的偏移量。
  - 使用 Simpicity Commander 工具烧录 DAC、PAI、CD 到目标设备。
  - 生成 efr32_creds.h 头文件，后续替换设备的 Matter 工程里面的同名头文件。

- 参数说明：
  - -p 是指 USB 的端口号。
  - -S 是指 J-Link 的序列号。
  - -B 是指支持的开发板型号。
  - -N 是指 DAC 证书的名字。
  - -V 是指 Vendor ID。
  - -P 是指 Product ID。
  - -C 是指 PAI 证书。
  - -K 是指 PAI 的私钥。
  - -D 是指 CD 文件。

- 执行成功的日志信息：

```
> python3 ./creds.py -p /dev/ttyUSB0 -S 69408690 -B brd2601b -N  "Matter Test DAC 01" -V 4169 -P 21845 -C Silabs-Test-PAI-Cert.pem -K Silabs-Test-PAI-Key.pem -D Silabs-Test-CD.bin


Building device app
 make -C device/brd2601b -f device-creds.Makefile


make: Entering directory '/home/lesun/matter/silabs_examples/credentials/device/brd2601b'
Building /home/lesun/matter/third_party/silabs/gecko_sdk/hardware/board/src/sl_board_control_gpio.c
```

```
.....

.....

Building
/home/lesun/matter/third_party/silabs/gecko_sdk/util/third_party/crypto/sl_component/sl_psa_driver/src/sli_se_version_dependencies.c

Building ../device_creds.c

Building ../main.c

Building autogen/sl_board_default_init.c

Building autogen/sl_device_init_clocks.c

Building autogen/sl_event_handler.c

Building autogen/sl_iostream_handles.c

Building autogen/sl_iostream_init_usart_instances.c

Linking build/debug/device-creds.out

Done.

make: Leaving directory '/home/lesun/matter/silabs_examples/credentials/device/brd2601b'


Flashing device app

 commander flash device/brd2601b/build/debug/device-creds.s37 --serialno 69408690 --device EFR32MG24


Reconfiguring debug connection with detected device part number: EFR32MG24B210F1536IM48

Parsing file device/brd2601b/build/debug/device-creds.s37...

Writing 65536 bytes starting at address 0x08000000

Comparing range 0x08000000 - 0x0800FFFF (64 KB)

DONE


Building host app

 make -C host/app/ -f host-creds.Makefile


make: Entering directory '/home/lesun/matter/silabs_examples/credentials/host/app'

Building ../host_creds.c

../host_creds.c: In function ‘host_creds_csr’:
```

```
../host_creds.c:114:22: warning: implicit declaration of function ‘strnlen’; did you mean ‘strlen’? [-Wimplicit-function-declaration]
  114 |     size_t cn_size = strnlen(common_name, CREDS_COMMON_NAME_MAX) + 1;
      |                      ^~~~~~~
      |                      strlen
Building ../main.c
Linking build/debug/host-creds
Done.
make: Leaving directory '/home/lesun/matter/silabs_examples/credentials/host/app'


Creating temp dir
  mkdir -p ./temp/



Requesting CSR
  ./host/app/build/debug/host-creds -p /dev/ttyUSB0 -f ./temp/csr.pem -N Matter Test DAC 01 -V 4169 -P 21845


Requesting CSR (cn:'Matter Test DAC 01',vid:0x1049, pid:0x5555) from '/dev/ttyUSB0' as './temp/csr.pem'
cfset for self brd2601b
Requesing CSR...
CSR generated, size:432
Writting './temp/csr.pem' (441)...
-----BEGIN CERTIFICATE REQUEST-----
MIIBBjCBqwIBADBJMRswGQYDVQQDDBJNYXR0ZXIgVGVzdCBEQUMgMDExFDASBgor
BgEEAYKifAIBDAQxMDQ5MRQwEgYKKwYBBAGConwCAgwENTU1NTBZMBMGByqGSM49
AgEGCCqGSM49AwEHA0IABIHV20ADN6wpLlg5IKKKQNtwBAUM2PPgUb2XENqRzSx8
SX8nLDUPuhpRmK8d49eqc/G3j7HCo9qJR+VaZtgwCfWgADAMBggqhkjOPQQDAgUA
A0gAMEUCIQDxGuggr6U3CmI0PWqZ3YWebxh5KEqHY0ZNBykt0LBsFAIgBzwQZCQS
rkm0NjCAHAhY2ZA7rZ4ZHJaftW7FpvDY/zA=
-----END CERTIFICATE REQUEST-----
```

```
Generating DAC (from device CSR)

 openssl x509 -sha256 -req -days 18250 -extensions v3_ica -extfile ./csa_openssl.cnf -set_serial 69408690 -CA Silabs-Test-PAI-Cert.pem -CAkey Silabs-Test-PAI-Key.pem -in ./temp/csr.pem -outform der -out ./temp/dac_cert.der


Signature ok

subject=CN = Matter Test DAC 01, 1.3.6.1.4.1.37244.2.1 = 1049, 1.3.6.1.4.1.37244.2.2 = 5555

Getting CA Private Key


Parsing PAI

 openssl x509 -outform der -in Silabs-Test-PAI-Cert.pem -out ./temp/pai_cert.der


PAI:            0x817e000 + 0x0                 = 0x817e000 (427)

DAC:            0x817e000 + 0x1c0       = 0x817e1c0 (466)

DC:             0x817e000 + 0x3c0       = 0x817e3c0 (233)


Erasing Page

 commander device pageerase --range 0x817e000:+8192 --serialno 69408690 --device EFR32MG24


Reconfiguring debug connection with detected device part number: EFR32MG24B210F1536IM48

Erasing range 0x0817e000 - 0x08180000

DONE


Flashing PAI

 commander flash ./temp/pai_cert.der --binary --address 0x817e000 --serialno 69408690 --device EFR32MG24


Reconfiguring debug connection with detected device part number: EFR32MG24B210F1536IM48

Parsing file ./temp/pai_cert.der...

Writing 8192 bytes starting at address 0x0817e000

Comparing range 0x0817E000 - 0x0817FFFF (8 KB)
```

```
Programming range 0x0817E000 - 0x0817FFFF (8 KB)

DONE


Flashing DAC

 commander flash ./temp/dac_cert.der --binary --address 0x817e1c0 --serialno 69408690 --device EFR32MG24


Reconfiguring debug connection with detected device part number: EFR32MG24B210F1536IM48

Parsing file ./temp/dac_cert.der...

Writing 8192 bytes starting at address 0x0817e000

Comparing range 0x0817E000 - 0x0817FFFF (8 KB)

Erasing range 0x0817E000 - 0x0817FFFF (1 sector, 8 KB)

Programming range 0x0817E000 - 0x0817FFFF (8 KB)

DONE


Flashing CD

 commander flash Silabs-Test-CD.bin --binary --address 0x817e3c0 --serialno 69408690 --device EFR32MG24


Reconfiguring debug connection with detected device part number: EFR32MG24B210F1536IM48

Parsing file Silabs-Test-CD.bin...

Writing 8192 bytes starting at address 0x0817e000

Comparing range 0x0817E000 - 0x0817FFFF (8 KB)

Erasing range 0x0817E000 - 0x0817FFFF (1 sector, 8 KB)

Programming range 0x0817E000 - 0x0817FFFF (8 KB)

DONE

Reconfiguring debug connection with detected device part number: EFR32MG24B210F1536IM48

Reading 2048 bytes from 0x0817e000...

{address:  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F}

0817e000: 30 82 01 A7 30 82 01 4E A0 03 02 01 02 02 08 71

0817e010: A4 7D E5 6E AD 88 C1 30 0A 06 08 2A 86 48 CE 3D
```

```
0817e020: 04 03 02 30 1A 31 18 30 16 06 03 55 04 03 0C 0F
0817e030: 4D 61 74 74 65 72 20 54 65 73 74 20 50 41 41 30
0817e040: 20 17 0D 32 33 30 31 31 35 31 34 33 30 30 30 5A
0817e050: 18 0F 39 39 39 39 31 32 33 31 32 33 35 39 35 39
0817e060: 5A 30 30 31 18 30 16 06 03 55 04 03 0C 0F 4D 61
0817e070: 74 74 65 72 20 54 65 73 74 20 50 41 49 31 14 30
0817e080: 12 06 0A 2B 06 01 04 01 82 A2 7C 02 01 0C 04 31
0817e090: 30 34 39 30 59 30 13 06 07 2A 86 48 CE 3D 02 01
0817e0a0: 06 08 2A 86 48 CE 3D 03 01 07 03 42 00 04 0C EC
0817e0b0: 88 31 3C 84 70 95 29 97 74 CC E1 7E B5 6D EA E9
0817e0c0: 86 A5 56 B8 D9 2C 6B 4F 7F DC 0D 5A D4 65 30 42
0817e0d0: 5D 88 5E CB 2A 3F 39 7B A2 43 94 EC 6B E8 67 D5
0817e0e0: E8 49 AE 3D 21 AE 94 88 4E 62 E2 F4 37 F1 A3 66
0817e0f0: 30 64 30 12 06 03 55 1D 13 01 01 FF 04 08 30 06
0817e100: 01 01 FF 02 01 00 30 0E 06 03 55 1D 0F 01 01 FF
0817e110: 04 04 03 02 01 06 30 1D 06 03 55 1D 0E 04 16 04
0817e120: 14 73 D0 1D E7 10 0B A4 43 39 A5 9C 4B 8D D3 BC
0817e130: F2 E4 83 81 C6 30 1F 06 03 55 1D 23 04 18 30 16
0817e140: 80 14 78 5C E7 05 B8 6B 8F 4E 6F C7 93 AA 60 CB
0817e150: 43 EA 69 68 82 D5 30 0A 06 08 2A 86 48 CE 3D 04
0817e160: 03 02 03 47 00 30 44 02 20 4F EF 72 89 23 10 A3
0817e170: 62 DD 30 83 07 C9 1C D4 EE DE 73 C6 99 0B 61 46
0817e180: 0F F2 41 8D A6 ED DB A6 9C 02 20 69 AE 29 3F 8C
0817e190: FA F3 BB DE 09 64 71 90 C4 B8 BE FE 06 A2 EE 41
0817e1a0: 96 96 DC F9 1C 43 D3 7F 82 D8 CC FF FF FF FF FF
0817e1b0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e1c0: 30 82 01 CE 30 82 01 73 A0 03 02 01 02 02 04 04
0817e1d0: 23 17 B2 30 0A 06 08 2A 86 48 CE 3D 04 03 02 30
0817e1e0: 30 31 18 30 16 06 03 55 04 03 0C 0F 4D 61 74 74
```

```
0817e1f0: 65 72 20 54 65 73 74 20 50 41 49 31 14 30 12 06
0817e200: 0A 2B 06 01 04 01 82 A2 7C 02 01 0C 04 31 30 34
0817e210: 39 30 20 17 0D 32 33 30 31 31 36 30 37 35 34 31
0817e220: 36 5A 18 0F 32 30 37 33 30 31 30 33 30 37 35 34
0817e230: 31 36 5A 30 49 31 1B 30 19 06 03 55 04 03 0C 12
0817e240: 4D 61 74 74 65 72 20 54 65 73 74 20 44 41 43 20
0817e250: 30 31 31 14 30 12 06 0A 2B 06 01 04 01 82 A2 7C
0817e260: 02 01 0C 04 31 30 34 39 31 14 30 12 06 0A 2B 06
0817e270: 01 04 01 82 A2 7C 02 02 0C 04 35 35 35 35 30 59
0817e280: 30 13 06 07 2A 86 48 CE 3D 02 01 06 08 2A 86 48
0817e290: CE 3D 03 01 07 03 42 00 04 81 D5 DB 40 03 37 AC
0817e2a0: 29 2E 58 39 20 A2 8A 40 DB 70 04 05 0C D8 F3 E0
0817e2b0: 51 BD 97 10 DA 91 CD 2C 7C 49 7F 27 2C 35 0F BA
0817e2c0: 1A 51 98 AF 1D E3 D7 AA 73 F1 B7 8F B1 C2 A3 DA
0817e2d0: 89 47 E5 5A 66 D8 30 09 F5 A3 60 30 5E 30 0C 06
0817e2e0: 03 55 1D 13 01 01 FF 04 02 30 00 30 0E 06 03 55
0817e2f0: 1D 0F 01 01 FF 04 04 03 02 07 80 30 1D 06 03 55
0817e300: 1D 0E 04 16 04 14 D2 A8 AA 96 15 26 86 AD 9D B7
0817e310: 63 9F 1D 55 40 3B D0 08 69 6D 30 1F 06 03 55 1D
0817e320: 23 04 18 30 16 80 14 73 D0 1D E7 10 0B A4 43 39
0817e330: A5 9C 4B 8D D3 BC F2 E4 83 81 C6 30 0A 06 08 2A
0817e340: 86 48 CE 3D 04 03 02 03 49 00 30 46 02 21 00 EB
0817e350: ED F6 99 95 AB D8 9C 7B D6 F5 57 6E B7 10 67 57
0817e360: 61 B3 01 DF B6 B5 FF 6D 1A 4F F9 CE 11 0C 60 02
0817e370: 21 00 85 8A 5B A6 F7 40 73 E6 1E 69 14 7D A2 8E
0817e380: 79 D4 54 56 3B 0B 40 9C D5 DC A3 F8 56 AA A6 11
0817e390: E0 EC FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e3a0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e3b0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

```
0817e3c0: 30 81 E6 06 09 2A 86 48 86 F7 0D 01 07 02 A0 81
0817e3d0: D8 30 81 D5 02 01 03 31 0D 30 0B 06 09 60 86 48
0817e3e0: 01 65 03 04 02 01 30 43 06 09 2A 86 48 86 F7 0D
0817e3f0: 01 07 01 A0 36 04 34 15 24 00 01 25 01 49 10 36
0817e400: 02 05 55 55 18 24 03 16 2C 04 13 5A 49 47 30 30
0817e410: 30 30 30 30 30 30 30 30 30 30 30 30 31 24 05
0817e420: 00 24 06 00 24 07 05 24 08 00 18 31 7C 30 7A 02
0817e430: 01 03 80 14 62 FA 82 33 59 AC FA A9 96 3E 1C FA
0817e440: 14 0A DD F5 04 F3 71 60 30 0B 06 09 60 86 48 01
0817e450: 65 03 04 02 01 30 0A 06 08 2A 86 48 CE 3D 04 03
0817e460: 02 04 46 30 44 02 20 14 5C 0E 32 54 95 E3 C0 EE
0817e470: A9 70 0D FF 94 9C B8 2A 4F 69 69 52 46 12 5E 86
0817e480: EC 8F EE AA 36 12 5C 02 20 08 04 B6 9D A0 0E 6F
0817e490: 61 A6 C1 93 3F F0 7C 37 BC 3B E3 CF 20 89 89 23
0817e4a0: 6C 36 5B 42 86 ED 4C 98 F4 FF FF FF FF FF FF FF
0817e4b0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e4c0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e4d0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e4e0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e4f0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e500: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e510: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e520: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e530: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e540: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e550: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e560: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e570: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e580: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

```
0817e590: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e5a0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e5b0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e5c0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e5d0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e5e0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e5f0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e600: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e610: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e620: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e630: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e640: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e650: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e660: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e670: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e680: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e690: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e6a0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e6b0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e6c0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e6d0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e6e0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e6f0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e700: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e710: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e720: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e730: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e740: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

0817e750: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

```
0817e760: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e770: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e780: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e790: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e7a0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e7b0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e7c0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e7d0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e7e0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0817e7f0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
DONE
```

5.7 将步骤 6 生成的./temp/efr32_creds.h 复制到 Matter 工程，并编译出新的 Matter 固件

- 如果是使用 Simplicity Studio v5 编译 Matter 工程，则将步骤 6 生成的./temp/efr32_creds.h 头文件替换{MATTER_HOME}/examples/platform/efr32/目录下的同名文件（具体 Ubuntu 和 Windows OS 如何拷贝文件，在此不做详细说明），并在 Simplicity Studio V5 中编译出新的 Matter 固件。注意：在拷贝替换 efr32_creds.h 之后，需要在 Simplicity Studio V5 中定义宏 EFR32_ATTESTATION_CREDENTIALS，然后再进行编译。设置方法为：在 Simplicity Studio V5 中，左键选中工程，File->Properties->C/C++ Build->Setting->GNU ARM C++ Compiler->Preprocessor，然后添加宏定义“EFR32_ATTESTATION_CREDENTIALS=1”。点击“Apply and Close”。
- 备注：本次培训为止，SSV5 里面的 Matter GSDK 编译固件验证该功能时暂时有点小问题，可以在 Linux Ubuntu 平台使用如下命令行的方式编译固件。

```
> cd matter/
> ./scripts/examples/gn_efr32_example.sh ./examples/lighting-app/silabs/efr32/ ./out/lighting-app BRD2601B
chip_build_platform_attestation_credentials_provider=true
```

5.8 使用 Simplicity Commander 将编译生成的固件烧录到 Breakout 开发板

- 将 JLink 烧录器连上 Matter Breakout 开发板，并将 JLink 的 USB 线连上电脑。从 SSV5 中打开 Simplicity Commander 并烧录新的固件。如下图 2 所示：

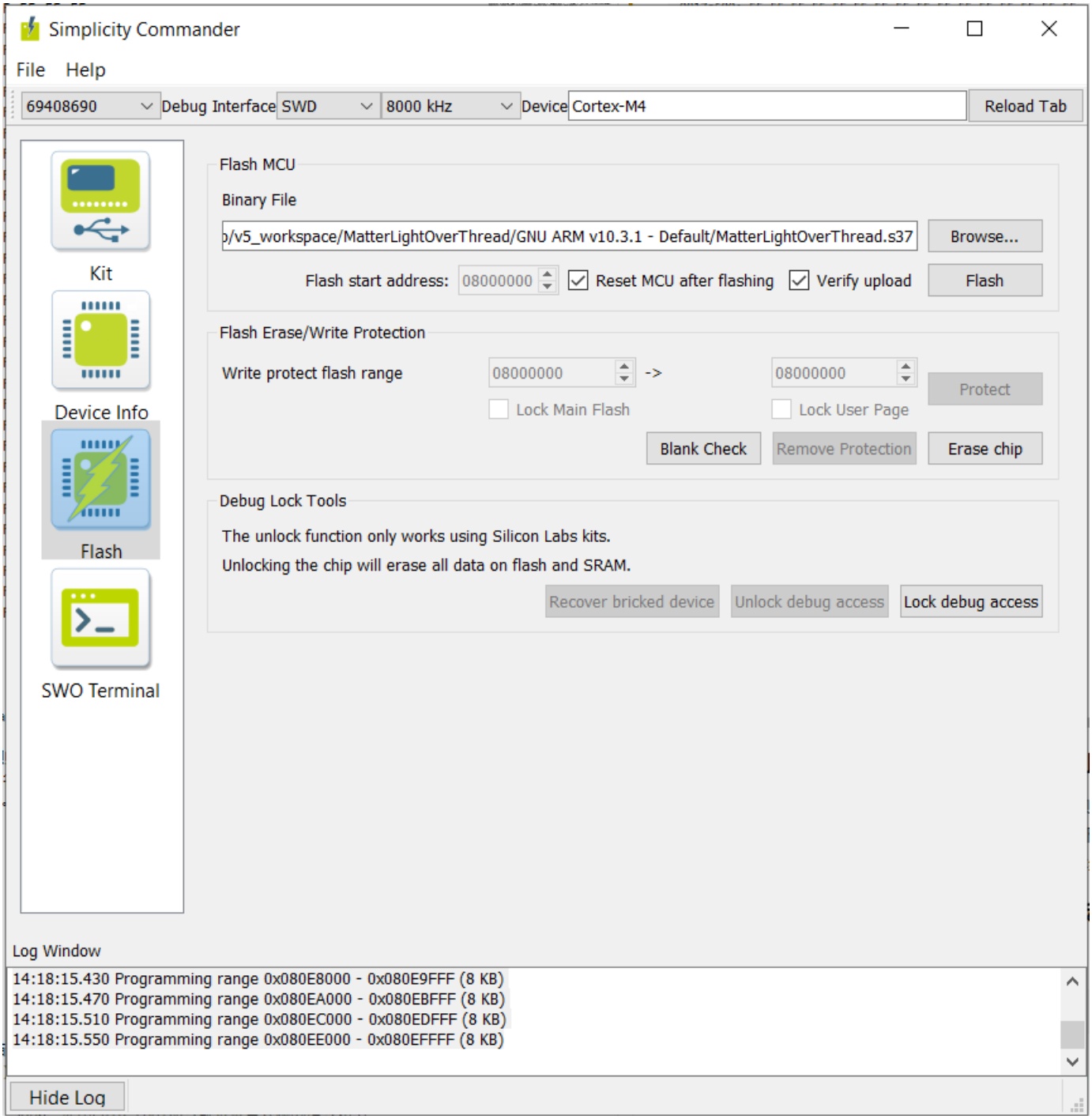

图 2 – Commander 工具烧录.s37 固件

6. 利用树莓派的 Chip-Tool 进行 Commissioning 配网验证

- 在树莓派中，使用 chip-tool 工具入网设备，这里不再描述，需要可点击这里查以前 Matter 相关的实验；设备成功入网，则表明烧入的 Spake2p NVM3 分区成功生效。
- 本实验指定 pincode 为：45502684，discriminator 为：3860，使用 chip-tool 命令触发设备入网的指令如下：

```
> cd connectedhomeip

> sudo rm -rf /tmp/chip_*

> mattertool startThread

> sudo ot-ctl dataset active -x

> ./out/standalone/chip-tool pairing ble-thread 1234 hex:<thread-dataset> 45502468 3860

> ./out/standalone/chip-tool onoff toggle 1234 1
```

7. 常见问题

- 在 Ubuntu 中安装完 Simplicity Commander 工具之后，需要设置环境变量。
- 如果 JLink 的 USB 端口无法在 Ubuntu 下识别，检查虚拟机的 USB 设置是否正确。也可以重新拔插与电脑相连的 JLink 的 USB 线。
- 在做 Commissioning 配网验证开始前，需要先烧入匹配的 Bootloader 和 Matter 设备的固件。
- 本实验中使用的 PAA 证书为测试的 Chip-Test-PAA-NoVID-Cert 证书。如果厂商后续开发产品的过程中需要使用其它的 PAA，则对应的 Chip-Tool 也需要添加对该 PAA 的支持。可以在 Matter 源码中 DefaultDeviceAttestationVerifier.cpp 文件中的 kTestPaaRoots[]数组中添加。

8. 参考资料

- Matter 源码 —— Silicon Labs 提供的 Matter 源码。
- chip-cert 工具 —— CSA 联盟提供的 chip-cert 工具，用于生成测试 PAI 证书、PAI 私钥以及 CD。

- creds.py 脚本 —— Silicon Labs 提供的 Python 脚本，用于烧录 Matter 设备 DAC 等安全证书信息。