AN2748

SAMA5D2 Linux®安全引导

简介

SAMA5D2 系列 MPU 支持两种引导模式：正常引导和安全引导。

- 如果引导时从外部存储器加载未加密/未签名的程序，则使用正常引导模式。这种工作模式适用于许多设计，而且非常适合开发过程，因为经过修改的代码稍加调试即可运行。
- 如果引导时加载加密/签名的程序，则使用安全引导模式。使用这种模式的设计通常会要求保证在引导时加载的映像是可信的，并且需经过授权才能在安全系统上运行。此外，某些软件还会通过加密来隐藏内容。

本应用笔记介绍了如何使用 SAMA5D2 MPU 将 Linux 内核作为安全应用程序进行引导。安全引导有助于防止在 SAMA5 MPU 上引导未经授权的软件。

本应用笔记为 SAMA5D2-RevC Xplained 板而编写，但经过定制修改后适用于任何 SAMA5D2 系统。

参考文档

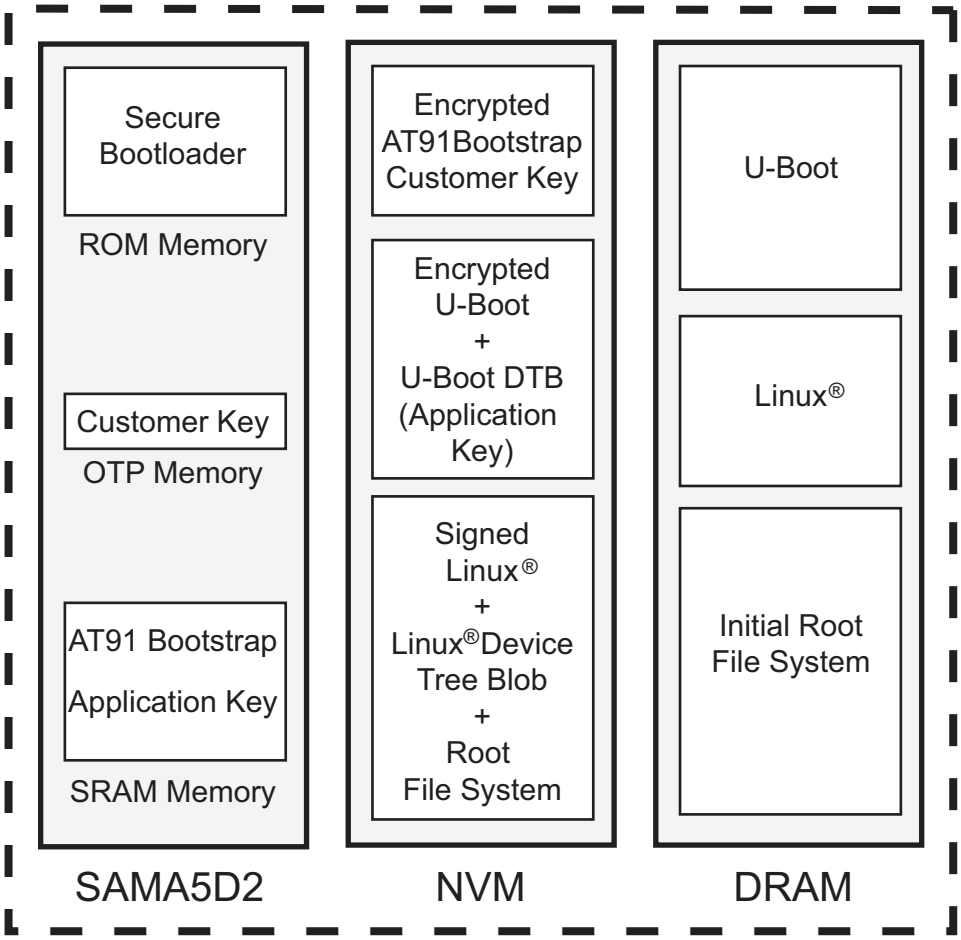
- *SAMA5D2 Series Data Sheet*（文档编号：DS60001476）。可从 www.microchip.com 获取。
- *SAMA5D2 Series Secure Boot Strategy* application note（AN2435，文档编号：DS00002435）。在签署保密协议（Non-Disclosure Agreement，NDA）的条件下由当地的 Microchip 销售办事处提供。
- 《SAMA5D2C（版本 C）Xplained Ultra 评估工具包用户指南》（文档编号：DS50002691B_CN）。可从 www.microchip.com 获取。
- Secure-sam-ba-cipher-3.2 自述文件
- Secure-sam-ba-loader-3.2 自述文件
- AT91Bootstrap 源代码
- U-Boot 文档（位于./doc/uImage.FIT 目录下）

目录

1. 系统的软件组件

- ROM 代码（第一阶段自举程序）
- AT91bootstrap 自举程序（第二阶段自举程序）
- U-Boot 自举程序（可选的第三阶段自举程序）
- Linux 内核/设备树二进制文件
- 根文件系统

图 1-1. 引导组件



1.1 ROM 代码

当处理器退出复位模式时，会开始执行引导 ROM 代码。在加载引导映像之前，ROM 会配置：

- Arm®监控器堆栈
- PLLA（使用 12 MHz 快速 RC 时钟作为输入）
- 处理器时钟（PCK）和主时钟（MCK）

ROM 代码通过读取熔丝区域中引导配置字的位来确定引导序列。然后将第二阶段自举程序从非易失性存储器装载到片上 SRAM。有关 ROM 代码配置和操作的详细信息，请参见应用笔记 *SAMA5D2 Series Secure Boot Strategy*（见参考文档）。

1.2 AT91bootstrap 自举程序

第二阶段自举程序 AT91bootstrap 负责初始化 SDRAM，并装载第三阶段自举程序或 Linux 内核和设备树二进制文件。第二阶段自举程序可位于下列其中一个 NVM 存储单元中：

- SDMMC（1 和 0）
- NAND 闪存
- 串行闪存（0 和 1）
- QSPI 闪存（0 和 1）

1.3 U-Boot 自举程序

U-Boot 是一款非常强大且灵活的自举程序，可加载来自各种源的应用程序。如果 AT91bootstrap 提供的功能不足以引导映像时（例如从网络引导时，或者从第二阶段自举程序不支持的设备或文件系统引导时），则可以使用第三阶段自举程序（如 U-Boot）。U-Boot 与 Linux 内核类似，同样使用扁平设备树（Flattened Device Tree，FDT）来配置 GPIO、串行端口和自举过程中使用的其他硬件设备等功能。

1.4 Linux 内核

Linux 内核是 Linux 系统的主要组件。它负责处理计算机系统的大部分关键功能，例如系统资源管理、调度、处理器中断、虚拟存储器和设备驱动程序。此外，该内核还有几个不同的普通“用户”程序接口。

Linux 内核由 AT91bootstrap 第二阶段自举程序或 U-Boot 第三阶段自举程序装载到 SDRAM 中。

1.5 设备树二进制文件

Linux 内核使用扁平设备树来描述运行它的系统硬件组件。在引导期间，首先会通过 AT91bootstrap 或 U-Boot 将设备树二进制（dtb）文件装载到存储器中，之后内核才会执行操作。

1.6 根文件系统

Linux 内核需要使用根文件系统。根文件系统包含 Linux 系统正常使用所需的全部必需库、程序、实用程序和设备节点。在本应用笔记中，我们通过将 Linux initRAM 文件系统包含在内核映像中来保护初始根文件系统。

2. 安全引导任务

关于安全引导的内容已全部介绍完毕，接下来我们可以介绍使能安全引导系统所需进行的活动。

1. 准备加密材料。需要生成密钥和证书并将其置于适当的位置。
2. 准备要装载的软件（Linux 内核、Linux 设备树和自举程序）。
3. 对第二阶段自举程序（AT91bootstrap）进行加密和签名。
4. 对第三阶段自举程序（U-Boot）进行配置、对公钥进行存储、加密和签名。
5. 将 Linux 内核和设备树打包到已签名的 FIT 文件中并正确签名。
6. 将 SAMA5D MPU 设置为安全引导模式。
7. 将密钥装载到 MPU 中。
8. 如有需要，可将安全引导模式和密钥永久编程到器件熔丝中。

3. 安全引导下的加密使用

3.1 加密

对于 Linux 安全引导环境，将对以下两个程序进行加密：AT91bootstrap 程序和 U-Boot。通过对 AT91bootstrap 程序进行加密，可阻止访问用于验证 U-Boot 的密钥。如果 AT91bootstrap 未加密，则可以在引导过程中十分轻松地伪造下一个软件阶段。这是因为 AT91bootstrap 使用对称密钥来执行身份验证。

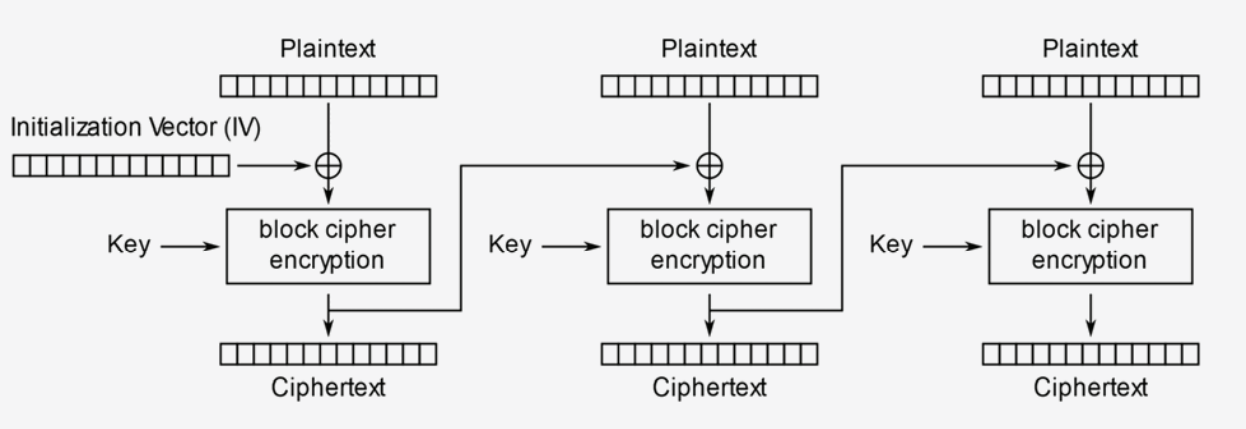
AT91bootstrap 和 U-Boot 均采用 AES 算法进行加密。AES 加密是一种对称算法，这意味着使用相同的密钥进行加密和解密。AES 密钥长度可以是 128 位、192 位或 256 位。密钥必须在加密数据的系统和解密数据的系统之间共享。

AES 是分组密码，这意味着它基于数据块（而非数据流）进行操作。无论所使用的密钥大小如何，块的大小都是 128 位。流数据加密与其他机制（如链接）结合使用时可使用 AES 块级加密。

安全引导 ROM 代码和 AT91bootstrap 使用密码块链接（Cipher Block Chaining，CBC）模式对系统上运行的固件进行加密/解密。

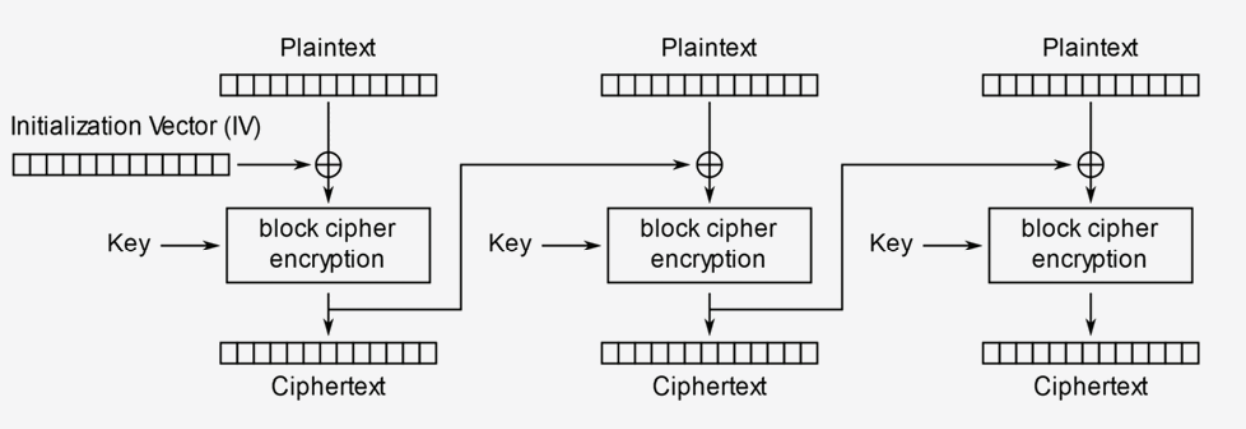
映像使用 AES-CBC 模式进行加密。

图 3-1. 密码块链接模式加密



引导 ROM 和自举程序使用 AES-CBC 解密，如下所述。

图 3-2. 密码块链接模式解密



3.2 身份验证

安全引导系统的一个重要特性是能够验证要引导的映像是否来自可信源，并且尚未以某种方式遭到篡改或损坏。通过将报文验证代码（Message Authentication Code，MAC）附加到映像末尾可确保通过身份验证和完整性检查。向映像附加 MAC 也被称为对映像进行“签名”。ROM 代码通过检查 AT91bootstrap 程序的签名来确保它是可信的。之后，AT91bootstrap 程序会检查 U-Boot 的签名以确保它是可信的。接着，U-Boot 会检查 Linux 映像、Linux 设备树和根文件系统的签名。

MAC 可通过多种方法来创建。其中两种比较常见的方法是：1）HMAC——基于哈希的报文验证代码；2）CMAC——基于密码的报文验证代码。SAMA5Dx MPU 使用 AES-CMAC 算法或 RSA + SHA-256 HMAC 来执行此检查。AT91bootstrap 程序使用 AES-CMAC 来验证 U-Boot。

4. 开发流程

本应用笔记将以一种非常规的顺序来呈现开发流程——从完全开放、未加密、未签名的平台开始；然后是运行的操作系统“上游”部分的安全性；最后回到 ROM 代码。这样，便可一次只添加一层安全功能，而且不必一开始即设法实现所有部分协同工作。

本应用笔记依赖 Buildroot 工具来管理工作 SD 卡映像（可在 SAMA5D2 开发板上运行）的编译过程。

4.1 创建一个工作 SD 卡映像

第一步是签出 Buildroot 并编译默认映像。

```
$ git clone https://git.buildroot.net/buildroot
$ cd buildroot
$ git checkout 2018.02.x
$ make atmel_sama5d2_xplained_mmc_defconfig
$ make menuconfig
```

修改使用的 U-Boot 版本。转到 Bootloaders→Uboot Version (Custom)（自举程序→Uboot 版本（自定义）），然后选择 2018.03。

```
$ make
```

这将为 SAMA5D2 Xplained 板创建一个工作 SD 卡映像。将映像复制到 SD 卡并确保它能在 SAMA5D2_Xplained 板上工作。

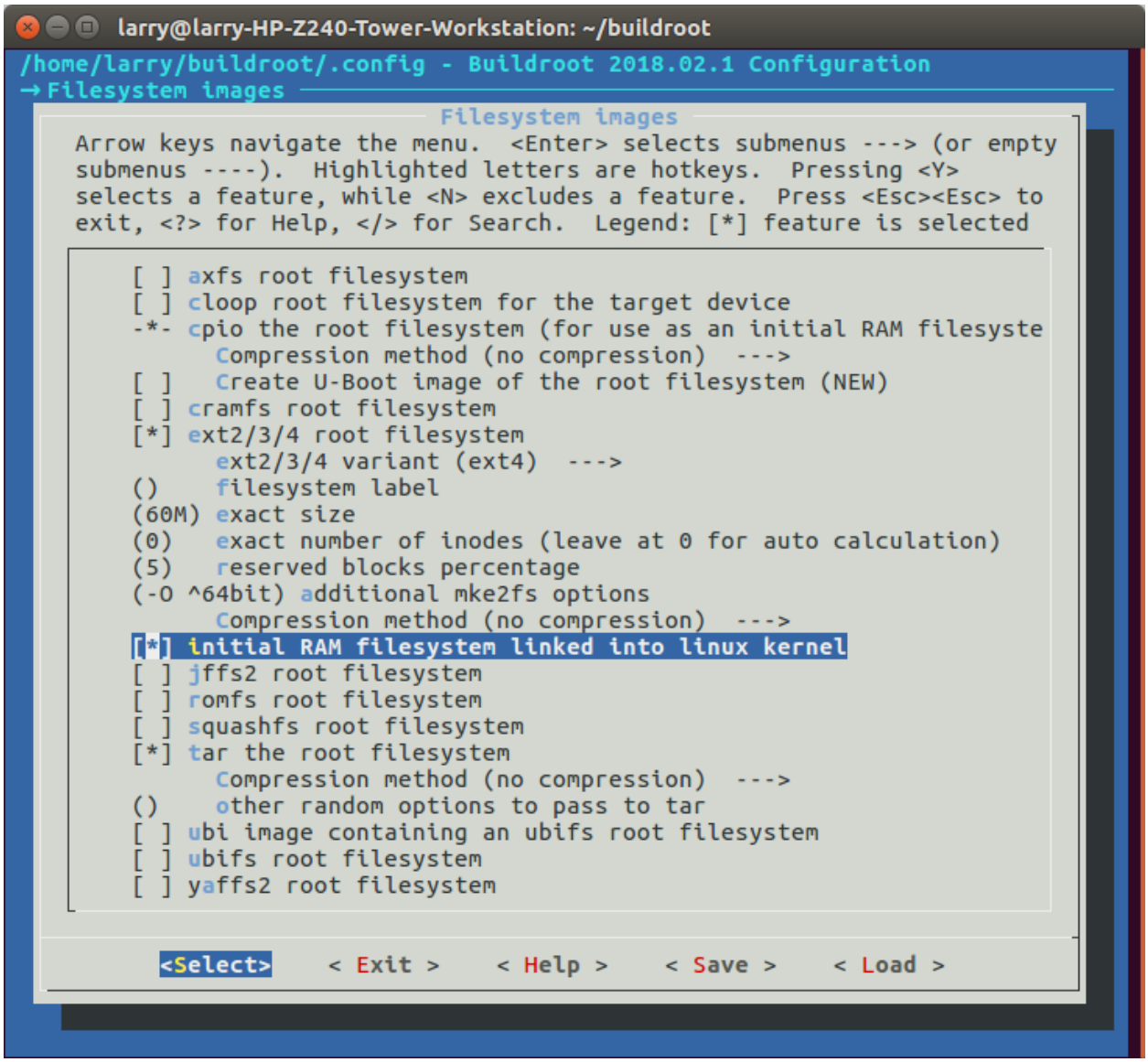
映像的名称为 output/images/sdcard.img

4.2 添加初始 RAM 文件系统

将根文件系统包含在 Linux 内核二进制文件中是验证根文件系统的最简单方法之一。为此，Buildroot 包含一个配置选项。

```
$ make menuconfig
```

转到 Filesystem Images（文件系统映像）并选择“initial RAM filesystem linked into kernel”（链接到内核的初始 RAM 文件系统）。



为了使初始 RAM 文件系统正常工作，需要事先手动重新配置 Linux 内核。编译映像并再次进行测试，以确保根文件系统能够正常工作。

```
$ make linux-reconfigure
```

```
$ make
```

我们已创建一个工作 Linux 映像，接下来我们需要使能 U-Boot 的验证引导机制。

5. U-Boot 验证引导

在 U-Boot 文档中，对验证引导进行了相关介绍。U-Boot 可使用验证引导过程来验证映像是否正确以及是否可在平台上运行。使用验证引导时需要：

1. 特殊配置
2. 创建密钥和证书
3. 将公钥存储在 U-Boot 控制 DTB 中
4. 创建 FIT 映像
5. 对 FIT 映像进行签名

本应用笔记使用了验证引导的“签名配置”方法。FIT 文件包含两个使用 SHA256 进行哈希处理的映像，以及一个使用 SHA256 进行哈希处理并使用 RSA 进行加密的配置。

5.1 U-Boot FIT 映像

扁平映像树（Flattened Image Tree，FIT）文件是扁平设备树（FDT）文件的特殊实例。U-Boot FIT 文件包含用于引导 Linux 等应用程序的文件和元数据（节点和属性形式），并非用于描述硬件。验证引导的 FIT 文件包含 Linux 内核、Linux 设备树、配置数据和哈希值（供 U-Boot 用于验证 FIT 文件中的数据）。U-boot 使用“bootm”命令来验证和引导 FIT 格式的映像。

5.2 配置 U-Boot

由于 U-Boot 非常灵活，因此需要仔细评估许多功能以确保安全性。其中一项配置设置 CONFIG_FIT_SIGNATURE 允许进行映像签名检查，它是 U-Boot 验证引导方法的核心。如果没有 CONFIG_FIT_SIGNATURE 设置，则无法检查甚至生成正确签名的 FIT 文件。

Buildroot 提供了一种使用语法来配置包的简单机制：

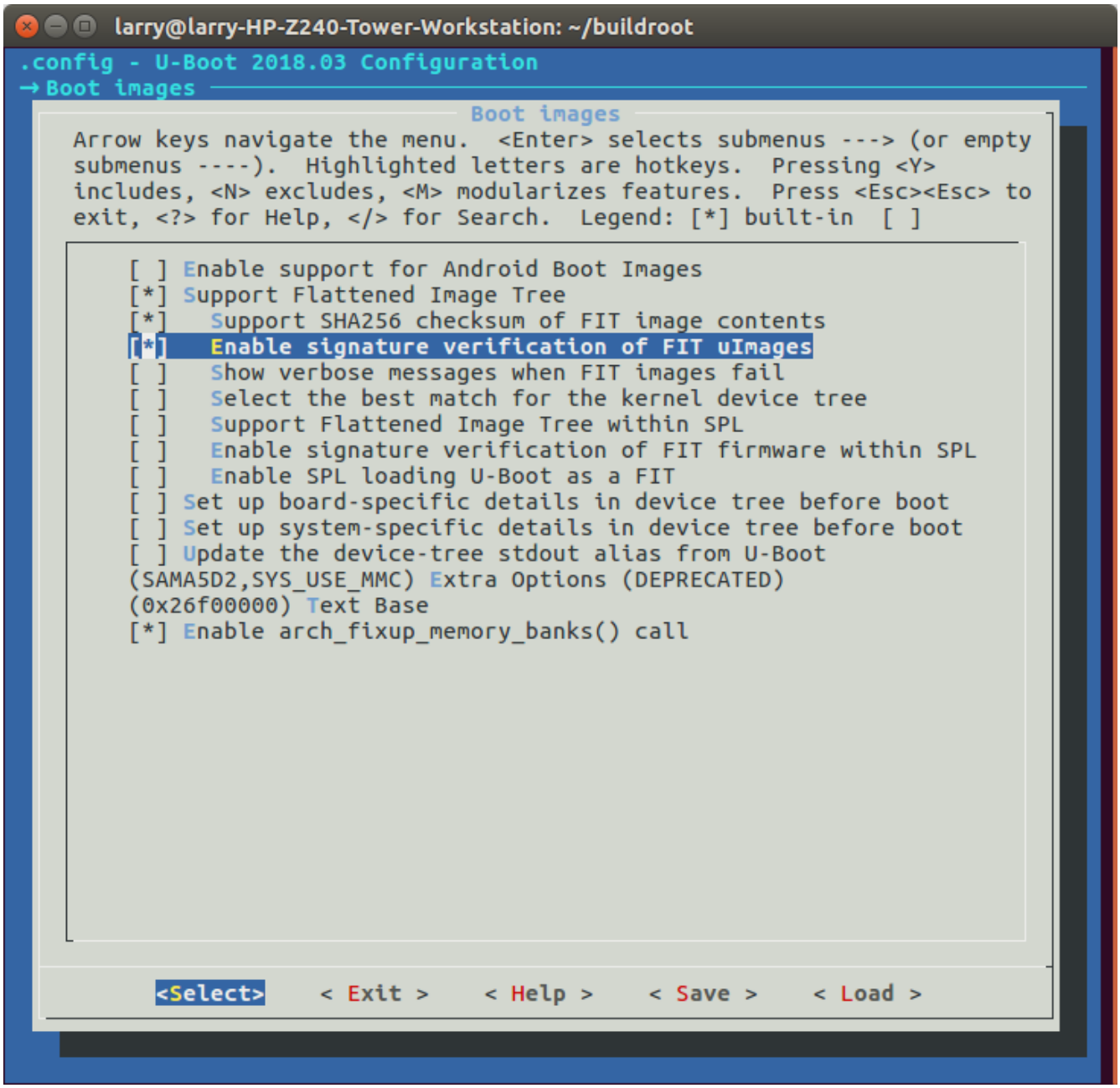
```
$ make <package name>-menuconfig
```

要运行 U-Boot 的 menuconfig，只需输入以下命令：

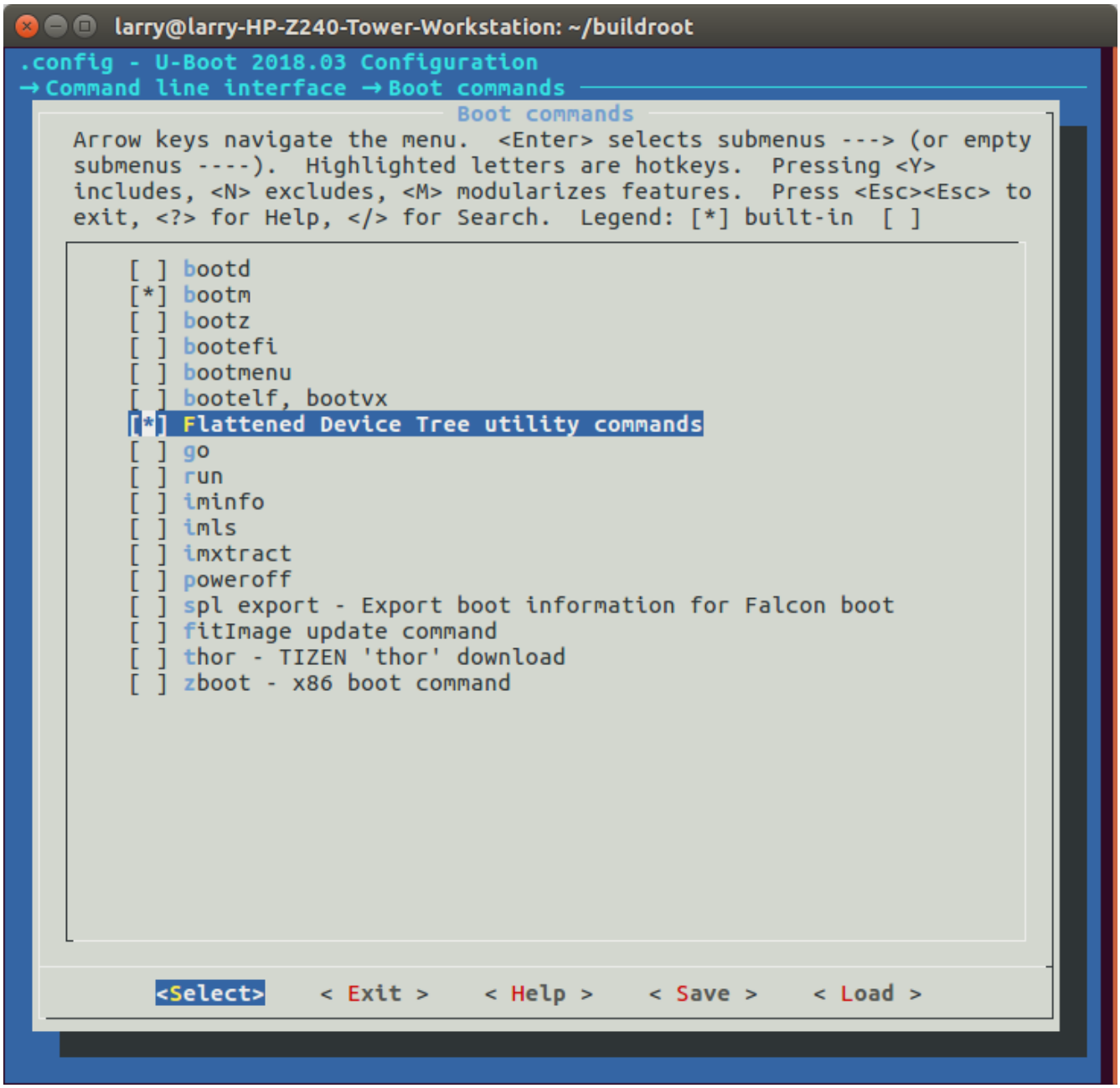
```
$ make uboot-menuconfig
```

以下 menuconfig 屏幕截图显示了为安全引导而修改的各种参数。

通过选择“Boot images -> Enable signature verification of FIT uImages”（引导映像 -> 使能 FIT 映像的签名验证）来设置 CONFIG_FIT_SIGNATURE。



在“Command Line Interface → boot commands”（命令行接口 → 引导命令）下，确保已使能 bootm 命令以及“扁平设备树实用程序命令”。



U-Boot 现在能检查已签名的 FIT 映像，但此时仍然缺少一些信息。U-Boot 具有默认控制 DTB，其不包含检查映像签名所需的公钥。如下图所示，在没有正确 U-Boot 控制 DTB 的情况下运行时，会生成一条消息，例如“Verifying Hash Integrity … OK”（正在验证哈希完整性 …正常）。虽然此消息看似正常，但无签名测试。哈希值正确，但出于安全引导的目的，还需要增加一个 RSA 步骤。以下部分介绍如何将正确的密钥插入 U-Boot 控制 DTB。

```
=> bootm 0x20000000
## Loading kernel from FIT Image at 20000000 ...
   Using 'conf@1' configuration
   Verifying Hash Integrity ...OK
   Trying 'kernel@1' kernel subimage
     Description:  unavailable
     Type:         Kernel Image
     Compression:  uncompressed
     Data Start:   0x200000c4
     Data Size:    4938272 Bytes = 4.7 MiB
     Architecture: ARM
     OS:           Linux
```

```
     Load Address: 0x21000000
     Entry Point:  0x21000000
     Hash algo:    sha256
     Hash value:   9edcd903bc251f78c1eda0d32f175140a172121d1a472d3f50f549b758f8a78a
   Verifying Hash Integrity ... sha256+ OK
## Loading fdt from FIT Image at 20000000 ...
   Using 'conf@1' configuration
   Trying 'fdt@1' fdt subimage
     Description:  unavailable
     Type:         Flat Device Tree
     Compression:  uncompressed
     Data Start:   0x204b5bbc
     Data Size:    31955 Bytes = 31.2 KiB
     Architecture: ARM
     Hash algo:    sha256
     Hash value:   9aa95c7aed6509bb294feb21dd904293620ef9e860f5ad6255771538945e4c39
   Verifying Hash Integrity ... sha256+ OK
   Booting using the fdt blob at 0x204b5bbc
   Loading Kernel Image ...OK
   Loading Device Tree to 3f95e000, end 3f968cd2 ...OK
Starting kernel ...
```

5.3 创建 RSA 签名凭据

本应用笔记不会详细介绍公钥基础架构（Public Key Infrastructure，PKI），但会介绍和描述执行安全引导所需的组件。执行 PKI 操作时，必须提前获取或构建几个重要组件。具体包括：

- RSA 根密钥和证书——用于签署“签名”证书
- RSA 签名密钥和证书——用于对映像进行签名

虽然 U-Boot 用于对映像进行签名的工具并未强制要求，但用于对代码进行签名的证书应与用于签署证书的证书不同。在本应用笔记中，我们将生成单个根 CA 证书和单个代码签名证书。通常，在未处于网络中的计算机上创建根 CA 证书。具体创建以下两个部分：

- CA 私钥
- CA 证书

私钥不得共享。因为凭借私钥，任何人都可以创建和签署伪造的证书，但可将其用作可信证书。

5.4 OpenSSL 配置文件

在创建用于对代码映像进行签名的证书之前，应修改默认的 OpenSSL 配置文件。修改的目的是更改在证书中找到的“可辨别名称”的默认值，以及为代码签名证书创建“密钥用法”值。

具体步骤如下：

1. 创建用于配置文件、密钥和证书的存储目录。

```
$ mkdir keys
$ cd keys
```

2. 复制 OpenSSL 默认配置：

```
$ cp /etc/ssl/openssl.cnf .
```

3. 编辑配置文件（openssl.cnf），使用代表您所属组织的值填写 req_distinguished_name 部分：

```
[ req_distinguished_name ]
countryName         = Country Name (2 letter code)
countryName_default     = US
countryName_min         = 2
countryName_max         = 2
stateOrProvinceName     = State or Province Name (full name)
stateOrProvinceName_default = Arizona
localityName            = Locality Name (eg, city)
localityName_default    = Chandler
```

```
0.organizationName      = Organization Name (eg, company)
0.organizationName_default  = Microchip Technology
```

4. 使用以下字段创建名为“code_sign”的部分：

```
[ code_sign ]
basicConstraints=CA:FALSE
keyUsage = digitalSignature
extendedKeyUsage = codeSigning
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer
```

5.5 创建 CA 证书和密钥

以下命令可用于创建有效期为 10 年的 CA 证书：

```
$ openssl req -x509 -newkey rsa:4096 -keyout cacert.key -out cacert.crt -days 3652 -sha256 -
config openssl.cnf
```

此命令将提示您填写可辨别名称字段，并要求提供用于加密私钥的密码。由于配置文件已修改为包含大多数可辨别名称的默认值，因此大多数条目按 Enter 键即可。请确保为“Common Name”（通用名称）字段输入适当的字符串。在下面的示例中，通用名称为“Test CA”

以下是 req 命令的部分示例输出。请注意，PEM 密码和通用名称是需要输入的部分。

```
$ openssl req -x509 -newkey rsa:4096 -keyout cacert.key -out cacert.crt -days 3652 -sha256 -
config openssl.cnf
Generating a 4096 bit RSA private key
.....................................................++
..............................++
writing new private key to 'cacert.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [Arizona]:
Locality Name (eg, city) [Chandler]:
Organization Name (eg, company) [Microchip Technology]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:Test CA
Email Address []:
```

“req”命令使用的选项如下：

表 5-1. req 命令选项

选项	说明
`-x509`	创建自签名 CA 证书而不是证书请求。
`-newkey rsa:4096`	创建 RSA 4096 位私钥。
`-keyout cacert.key`	将 CA 私钥写入“cacert.key”。
`-out cacert.crt`	将 CA 证书写入“cacert.crt”。

.....（续）

选项	说明
`-days 3652`	证书有效期为 3652 天。
`-sha256`	使用 SHA256 哈希算法作为签名算法。
`-config openssl.cnf`	使用“openssl.cnf”作为配置文件。

5.6 创建证书请求和私钥

```
openssl req -nodes -newkey rsa:4096 -keyout samkey.key -out samkey.csr -sha256 -config
openssl.cnf
```

此命令没有-x509 选项，因此它将创建证书签名请求（Certificate-Signing-Request，CSR），而非自签名证书。此命令还指定-nodes 选项，该选项会导致私钥未加密。U-Boot 目前不支持受密码保护的 RSA 密钥。未来可能会支持。

```
$ openssl req -nodes -newkey rsa:4096 -keyout samkey.key -out samkey.csr -sha256 -config
openssl.cnf
Generating a 4096 bit RSA private key
.....................................................++
..............................++
writing new private key to 'samkey.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [Arizona]:
Locality Name (eg, city) [Chandler]:
Organization Name (eg, company) [Microchip Technology]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:U-Boot Image Signing
Email Address []:
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

5.7 对证书请求进行签名

```
openssl x509 -req -in samkey.csr -days 365 -CA cacert.crt -CAkey cacert.key -CAcreateserial -
out samkey.crt -extfile openssl.cnf -extensions code_sign
```

此命令使用先前创建的 CA 证书对 samkey.csr 证书请求进行签名。U-Boot 要求密钥和证书的名称必须相同，但密钥的文件扩展名必须为.key，证书的文件扩展名必须为.crt。签名人必须输入 CA 私钥的密码才能成功对证书请求进行签名。

```
$ openssl x509 -req -in samkey.csr -days 365 -CA cacert.crt -CAkey cacert.key -CAcreateserial
-out samkey.crt -extfile openssl.cnf -extensions code_sign
Signature ok
subject=C = US, ST = Arizona, L = Chandler, O = Microchip Technology, OU = MPU32
Applications, CN = U-Boot Image Signing
Getting CA Private Key
Enter pass phrase for cacert.key:
```

5.8 检查签名证书

如果要检查 CA 签名的证书，可使用“x509”命令以人们可阅读的形式打印证书。

```
$ openssl x509 -in samkey.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            81:9b:fb:48:3b:da:22:94
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = Arizona, L = Chandler, O = Microchip Technology, CN = Test CA
        Validity
            Not Before: Apr 30 03:26:41 2018 GMT
            Not After : Apr 30 03:26:41 2019 GMT
        Subject: C = US, ST = Arizona, L = Chandler, O = Microchip Technology, CN = U-Boot
Image Signing
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
                    00:ed:04:19:ea:79:01:f8:ef:74:ff:5c:01:20:50:
                    b3:f0:6f:03:10:62:66:9c:59:95:c6:ff:4c:5d:be:
                    33:c5:3c:36:28:c7:44:0c:95:91:66:e8:0e:5d:81:
                    7f:11:95:f6:c8:37:0d:5e:30:2c:0f:d5:84:0f:14:
                    b2:b3:2b:72:b0:65:6f:15:97:31:38:26:88:15:34:
                    d4:4e:20:7c:94:1c:35:9b:ec:62:06:5d:7f:6f:db:
                    93:12:b5:e9:5c:fd:45:a8:9c:5a:10:a7:37:09:2b:
                    d1:35:19:d3:5e:8e:92:4e:9a:53:a5:ad:11:41:e0:
                    06:4b:37:7e:77:c8:82:81:7b:ee:5f:be:71:4f:18:
                    66:d4:fe:2a:74:55:ef:ab:dc:df:80:44:6c:c8:f8:
                    5f:5c:da:76:04:08:68:7c:4b:ec:68:06:81:a0:e8:
                    f5:6e:57:01:38:66:43:9c:e2:4a:d2:d2:ed:f6:64:
                    3c:99:04:f4:af:8b:46:bd:90:45:52:41:1a:45:f5:
                    48:77:28:b0:a8:9f:00:e2:5b:8a:00:d4:01:8c:6f:
                    09:ea:58:a1:2e:73:9d:d2:44:b9:e8:b7:d7:9b:5e:
                    24:a5:2d:86:65:2f:66:ec:41:83:5a:18:e5:47:d3:
                    d1:9a:0d:5d:b9:cd:c2:bd:71:a1:39:ec:5b:88:ad:
                    1b:f7:99:7f:0b:de:89:59:c5:26:9c:de:00:8f:41:
                    76:0b:c2:76:8d:31:84:7f:1b:c4:6f:f7:2d:45:45:
                    12:80:cb:54:ae:f7:65:15:55:dc:d8:c0:9d:74:27:
                    81:00:4a:f2:a9:31:9f:4b:fc:01:16:9a:50:df:96:
                    4d:26:30:f4:4c:11:6e:02:f8:cb:8a:80:37:81:83:
                    84:81:ac:91:46:97:ea:a2:9f:46:36:56:4c:6d:c9:
                    c6:f8:42:e2:e9:90:97:49:bf:9a:5d:21:2d:50:74:
                    bf:ef:a0:53:c4:7b:4a:e6:18:6c:1d:77:ee:94:fc:
                    4c:c5:ae:06:20:32:52:b5:0c:0c:2d:a8:56:76:1b:
                    b8:0c:d6:81:b3:1e:26:1a:f8:69:3f:97:a8:c9:61:
                    c0:66:55:cf:b4:de:67:9e:c5:ad:a7:88:16:3b:c6:
                    e0:47:82:01:92:e4:1f:b1:d3:dd:7f:70:1a:ab:47:
                    9f:50:98:21:3e:92:fe:45:d4:b0:fd:6f:01:dc:01:
                    d5:f9:9e:57:5e:e1:17:70:c9:42:05:8c:c8:bf:8f:
                    26:fd:19:e7:9e:89:cf:f4:25:6f:40:d8:37:4d:0f:
                    61:5c:5b:58:9e:16:d1:5b:43:d3:c4:3e:81:b1:1c:
                    43:73:b2:11:fe:a2:6f:a1:f5:a9:f5:73:33:99:02:
                    86:ff:57
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature
            X509v3 Extended Key Usage:
                Code Signing
            X509v3 Subject Key Identifier:
                1E:C2:6A:E1:57:03:C2:DC:BB:DD:34:1E:CD:65:49:0C:56:FC:B7:3E
            X509v3 Authority Key Identifier:
                keyid:BC:C6:9B:7C:56:98:B7:CB:37:34:24:BE:A8:CD:B1:BD:9D:66:B7:AF
    Signature Algorithm: sha256WithRSAEncryption
         cd:ba:53:1f:31:b0:3c:51:d3:0f:dc:87:e4:d0:b5:1c:55:98:
         21:8b:6e:34:29:44:a2:24:14:08:26:0f:d8:74:42:94:44:09:
         b7:64:15:ee:5d:b4:6c:5e:dc:7e:5c:0a:c1:00:17:4f:4c:c1:
         4a:93:b9:61:21:a6:55:a4:c8:c8:f9:75:c8:29:f3:4f:cf:0b:
         d0:60:95:0f:de:27:21:ee:5e:68:8d:af:17:6b:ff:a8:06:9f:
```

```
          3e:31:05:3d:e4:d3:a8:db:97:5f:d7:75:47:94:ea:31:f5:3b:
          87:03:a9:47:cc:36:4e:38:0a:9b:c7:4e:3a:a4:18:3e:42:2f:
          75:60:3d:07:66:89:85:7c:45:b4:a8:a4:c8:87:34:45:dd:99:
          21:37:78:5f:06:f5:f7:f0:e5:5b:eb:7b:ea:59:8a:a6:0d:3d:
          18:ab:3b:fb:71:49:2b:9d:7d:2c:99:3d:42:06:91:4a:5f:42:
          37:c0:5b:ba:df:0b:df:47:80:e5:c8:cd:9c:52:94:a4:cf:c1:
          aa:11:e9:db:83:e2:2b:e9:42:ae:3d:c5:20:ad:2c:22:c8:15:
          68:7d:3a:e8:69:21:a3:bb:f0:6e:8e:8a:e3:a8:f4:11:20:7d:
          eb:e3:23:92:b6:27:28:f6:c1:da:e9:ab:1e:71:95:75:f4:7a:
          52:c2:91:b2:56:da:f0:74:c4:3a:53:3d:77:d9:65:d9:9c:1f:
          8e:2a:90:0f:c9:8a:c8:ab:7b:4f:58:d2:71:49:a5:e8:cb:16:
          53:56:a1:a3:73:d1:1b:3a:8d:e2:9c:ef:26:b5:75:9b:17:3c:
          1c:70:a8:6d:d4:cb:37:0e:4a:a2:dc:5e:15:38:6d:05:96:d8:
          6e:8d:8a:13:86:de:a6:fc:1b:65:f8:85:e5:3b:92:1a:61:cc:
          b2:8f:50:2a:73:ba:8b:36:85:6a:19:98:8a:3f:0b:ca:69:e5:
          59:33:02:c1:75:e3:82:ed:d4:c7:03:da:ab:94:c9:24:ec:00:
          2c:96:a1:56:0e:f7:50:9a:1b:68:c0:83:f7:7f:85:d8:b0:81:
          2e:24:58:c9:6a:c8:c4:3e:85:4c:7f:9e:6c:fa:f6:4e:bd:12:
          37:08:6b:91:92:50:78:8f:09:db:cb:f4:ce:a8:fe:5d:33:0e:
          a6:0f:5f:7b:bf:f4:99:96:f2:54:f7:14:54:a8:88:b0:f4:3d:
          c3:85:21:ee:ef:13:53:64:cd:a8:fd:14:da:6b:ee:cf:57:56:
          43:8f:26:75:aa:fc:75:03:3f:b9:80:60:70:2c:5c:b7:1d:b1:
          b4:82:df:cb:4d:0f:92:5d:80:30:63:49:a4:1a:78:ef:ce:9c:
          ff:f6:cf:6e:59:7a:1e:58
```

5.9 FIT 模板

FIT 文件由 mkimage 工具使用 DTS 语法输入文件创建。这是因为 mkimage 工具依赖于设备树编译器工具（作为签名 FIT 文件创建过程的一部分）。U-Boot 版本 DTC 具有“/incbin/”保留字，允许将整个二进制映像作为属性值包含在最终 DTB 中。Linux 内核和 Linux DTB 即通过这种方式置于 FIT 文件中。

在 Buildroot 基本目录中，创建一个名为“linux.its”的文件，内容如下：

```
/dts-v1/;
/ {
    description = "Linux kernel image with one or more FDT blobs";
    #address-cells = <1>;
    images {
        kernel@1 {
            data = /incbin/("output/images/zImage");
            type = "kernel";
            arch = "arm";
            os = "linux";
            compression = "none";
            load = <0x21000000>;
            entry = <0x21000000>;
            hash@1 {
                algo = "sha256";
            };
        };
        fdt@1 {
            data = /incbin/("output/images/at91-sama5d2_xplained.dtb");
            type = "flat_dt";
            arch = "arm";
            compression = "none";
            fdt-version = <1>;
            hash@1 {
                algo = "sha256";
            };
        };
    };
    configurations {
        default = "conf@1";
        conf@1 {
            kernel = "kernel@1";
            fdt = "fdt@1";
            signature@1 {
                algo = "sha256,rsa4096";
                key-name-hint = "samkey";
                sign-images = "fdt", "kernel";
            };
        };
```

```
    };
};
```

5.10 公钥提取

公钥需要存储在 U-Boot 控制 DTB 中。为更好地适应 U-Boot 编译过程，我们将提取公钥并将其置于源 DTS 文件中，而不是直接修改控制 DTB。只有在创建新的签名证书时才需要执行此步骤。借助 DTS 文件中的公钥，无论何时执行 U-Boot 编译，都会将正确的公钥写入 U-Boot 二进制文件。即使执行“清除”编译亦如此。

创建新的签名证书时，可以执行以下过程以将获取的公钥置于控制设备树源中。

中间文件 pubkey.dtb 将用作公钥的保存位置，随后将合并到相应的 dts 文件中。

1. 创建一个 DTS 文件，仅作为用于创建空 DTB 文件的 shell。本例中名为“pubkey.dts”的文件应如下所示：

```
/dts-v1/;
/ {
};
```

2. 编译 DTS 文件以获取空 DTB：

```
$ dtc -O dtb pubkey.dts > pubkey.dtb
```

3. 运行位于 U-Boot 编译目录下的 mkimage 工具以创建 FIT 映像，并提取公钥。确保用于签名的密钥和证书位于同一目录下。在本示例中，目录名为“keys”。“keys”目录中应存在两个文件：samkey.crt 和 samkey.key。文件 samkey.crt 为证书，文件 samkey.key 是用于签署 FIT 文件的私钥。

```
$ output/build/uboot-2018.03/tools/mkimage -f linux.its -k keys -r -K pubkey.dtb
linux.itb
```

4. 取消编译 DTB 文件并改写 pubkey.dts：

```
$ dtc -I dtb pubkey.dtb > pubkey.dts
```

5. 编辑用作控制 DTB 的 U-Boot DTS 源文件，并将 pubkey.dts 插入到结构中。在本示例中，我们将编辑 output/build/uboot-2018.03/arch/arm/dts/at91-sama5d2_xplained.dts。请确保仅复制“签名”节点。编辑之前，at91-sama5d2_xplained.dts 的前几行如下所示：

```
/dts-v1/;
#include "sama5d2.dtsi"
#include "sama5d2-pinfunc.h"
/ {
    model = "Atmel SAMA5D2 Xplained";
    compatible = "atmel,sama5d2-xplained", "atmel,sama5d2", "atmel,sama5";
    chosen {
        u-boot,dm-pre-reloc;
        stdout-path = &uart1;
    };
    ahb {
        usb1: ohci@00400000 {
            num-ports = <3>;
            atmel,vbus-gpio = <&pioA 42 0>;
            pinctrl-names = "default";
            pinctrl-0 = <&pinctrl_usb_default>;
            status = "okay";
        };
```

在包含 pubkey.dts 的签名节点之后，at91-sama5d2_xplained.dts 的前几行如下所示：

```
/dts-v1/;
#include "sama5d2.dtsi"
#include "sama5d2-pinfunc.h"
/ {
    model = "Atmel SAMA5D2 Xplained";
    compatible = "atmel,sama5d2-xplained", "atmel,sama5d2", "atmel,sama5";
    signature {
        key-samkey {
            required = "conf";
```

```
            algo = "sha256,rsa4096";
            rsa,r-squared = <0xdbfec24b 0xa5ea2b1a 0x68400641 0x7326b903 0x56304c98
0x506293aa 0xd8f2b88c 0xff588c74 0x7ff0a71b 0xfe7d3a5d 0x22f305d2 0x9de61df7 0x5caf73fb
0x383e5828 0xc6b78b09 0x7f8bea2b 0xae04338d 0xd2aabc16 0xe755d903 0x4edd0d2e 0x81d12585
0x410a695d 0x191a5d10 0xc4215770 0x6b35edfa 0x94b6cca 0xb6d4f75e 0x12b8c3ac 0x38fa2bc0
0xeb198a6c 0x5063d6ce 0xdda0ad2b 0xfd31ab15 0xb5a2de0b 0xb9e9f906 0x84d7ab9d 0xe3d481f3
0xd0d66125 0x4f4500b6 0x35a90317 0xc27509f1 0x950266e1 0xecaad051 0x4ef9e854 0x4bafc3ad
0x977d78e4 0xdb28e6f9 0x6e3cd28d 0x8626bb81 0x79018bef 0xf952fcc3 0x143d6e9f 0x472cc55f
0xbd793e29 0xd1973294 0x30b01413 0xa06a8d5b 0x59b87ecb 0x36fe25df 0x8c0240ee 0x9683c6e2
0xac4fab5b 0xa0a41089 0x77a89400 0x563c9cdb 0xf18e1984 0xbaf14849 0xfc0ee81a 0x63e90647
0x608656a3 0xf4d8a777 0xc9c0d23b 0x180c4dd6 0xefba6909 0x5dbd1bfd 0x6b1e1f50 0x8b4a764d
0x99495e21 0x7db3b75e 0x948b4ca8 0xf37dda85 0x352ca2f8 0x43ac5883 0xc0e745f0 0x6fc585b0
0x60e6f9ec 0x49457b4f 0x251948cf 0x48aef777 0xd2f8d3b3 0x6c880548 0xb147d8c3 0x353589be
0x3030a4fa 0x3a0de758 0xdc3c58aa 0x463f4d12 0x3c858599 0xcf0a8e24 0x1d47a6bc 0xc456ebde
0x8f774b0c 0xc79324e7 0xb329cdca 0xd72bf02a 0xdc992be8 0xf6f75455 0x8d7add99 0x85cb9ef5
0xd026d74 0x81da0447 0x1ad13f10 0xaafb6612 0xd4f288ea 0xdb95ff5 0xe3a4786f 0xa746913a
0x8a64781d 0x4f300d3b 0xccf8dd37 0xdb7dc010 0x54098c99 0x3f1e5ea2 0xffc98e8f 0x833f7267
0x740f3d0 0x9b981910 0xd8d558b5>;
            rsa,modulus = <0xe4d61420 0xe402426a 0xb7e8a8e 0x7fed6cf3 0x2d105d06 0x58070696
0x9d12b3d3 0x1a549c06 0xf863d237 0x4ed637b3 0xd1e9172c 0xe9a08063 0x8c69eeda 0x2c31e009
0x75651b8e 0x39773276 0x4ae7d620 0x26ad6d5c 0x94f0368a 0x877fb943 0x8ac20c66 0x527ed3d
0xc1a3d0 0xf86bda13 0xaeb60f4b 0xbfbfd887 0x3fd6c20a 0x3212150a 0x68537bd 0xaf00b12e
0x39941c70 0x359d2602 0xbb2ca80e 0xd3be6c23 0x2c6929cc 0xf26f0249 0x7c93878d 0x435eebf5
0xc76479b1 0x960228cd 0x69d9a190 0x8de8d691 0xe93bebe3 0x708c5dfe 0x2d2013d6 0xfdb2c2f9
0xc8f46fb6 0x27ecd502 0x62737dc5 0xb2797bfd 0x8415de8 0xc17d85b7 0xd4e94000 0xada5f467
0x9b16c556 0x29e74ade 0x34b62261 0x61532487 0x3f7563b8 0x1c8bb911 0x6c92201a 0xb3892cd7
0x82fd1935 0x76cb4571 0x96edc77f 0x12e0cd53 0xacf61aa9 0x8192b68e 0x722e4231 0xdfb6fb75
0x581502eb 0xdaff4c38 0x2109510 0x9beb7e5e 0x7b639e13 0xc3caa133 0xc4bc8884 0x7535ca33
0x3276a627 0x943cdbc7 0xeb3f2ca0 0x39005a10 0x33323f82 0xfe16ea62 0x414286 0xc26e39e8
0xf72f18f 0x78b1c4ac 0x5e8230f 0xa785d73b 0xf60ba5d1 0x416fa25c 0x6395387c 0xa836b42
0xb9649d5c 0x92b4e 0x871a656c 0x5e970991 0x545b3df7 0xd19fb193 0x84b55c49 0x388111a5
0xbd027fe 0xcb39dd8a 0xd4f5856 0xd3434afb 0xc74df655 0xeec9d781 0x6d7c28f6 0xde2f5d14
0xb266c3b 0xb9f322cb 0x24e4de6c 0x4952433b 0xf1011ad7 0xda4c2c38 0x8a92543f 0xf4db6ab7
0xe4bd1c5b 0x6c2bcc2f 0xd4e59af1 0x988456dd 0xd1176b8f 0x7da860fa 0xdf1b7530 0x240ef9f
0x40e63e 0x6f1fb023>;
            rsa,exponent = <0x0 0x10001>;
            rsa,n0-inverse = <0x55718075>;
            rsa,num-bits = <0x1000>;
            key-name-hint = "samkey";
        };
    };
```

重新编译 U-Boot 时，此设备树将用作控制 DTB，并附加到 U-Boot 可执行文件以创建单个二进制文件。

要重新编译 U-Boot，请使用以下命令：

```
$ make uboot-rebuild
$ make
```

5.11 测试新映像

将映像 output/images/sdcard.img 复制到 SD 卡，然后将文件“linux.itb”复制到 SD 卡的 FAT 分区。

在按下 SAMA5D2 板的复位按钮后 3 秒内按 Enter 键进入 U-Boot 的命令 shell。使用“fatload”命令（如下所示）将“linux.itb”FIT 文件装载到 SDRAM 中。然后使用“bootm”命令（如下所示）执行签名检查并引导 FIT 映像。请注意，“Verifying Hash Integrity”（验证哈希完整性）消息显示“rsa4096:samkey+”。这表明确实检查了 FIT 签名，并且发现它是有效的。

```
按任意键停止自动引导：0
=> fatload mmc 1:1 0x20000000 linux.itb
4975090 bytes read in 307 ms (15.5 MiB/s)
=> bootm 0x20000000
## Loading kernel from FIT Image at 20000000 ...
   Using 'conf@1' configuration
   Verifying Hash Integrity ... sha256,rsa4096:samkey+ OK
   Trying 'kernel@1' kernel subimage
     Description:  unavailable
     Type:         Kernel Image
     Compression:  uncompressed
     Data Start:   0x200000c4
     Data Size:    4937736 Bytes = 4.7 MiB
```

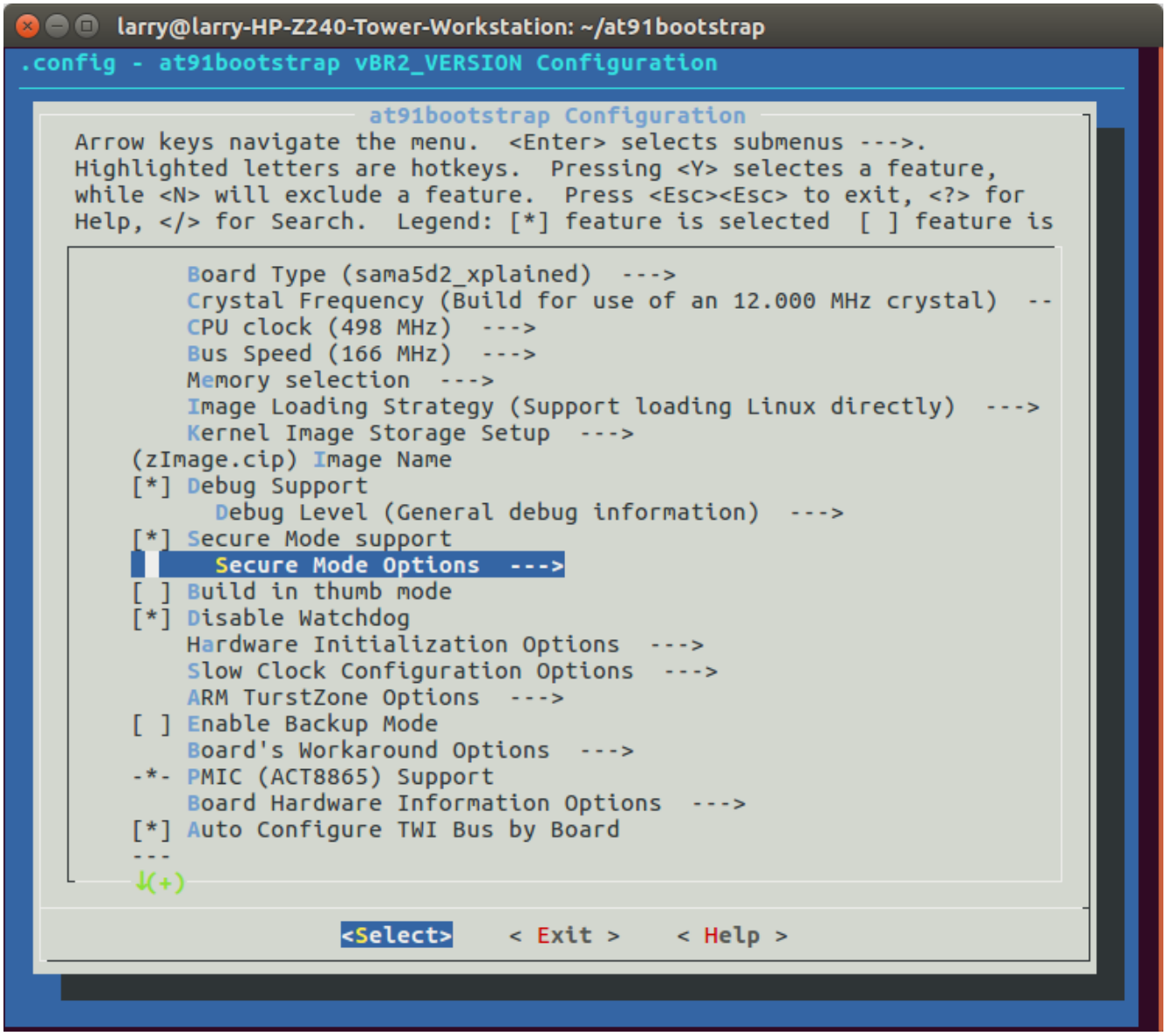
```
     Architecture: ARM
     OS:           Linux
     Load Address: 0x21000000
     Entry Point:  0x21000000
     Hash algo:    sha256
     Hash value:   aaf680f3fbfc09d0a102a75959bce1c423f8b096b125ae54ca3f98c08b8caa1e
   Verifying Hash Integrity ... sha256+ OK
## Loading fdt from FIT Image at 20000000 ...
   Using 'conf@1' configuration
   Trying 'fdt@1' fdt subimage
     Description:  unavailable
     Type:         Flat Device Tree
     Compression:  uncompressed
     Data Start:   0x204b59a4
     Data Size:    33480 Bytes = 32.7 KiB
     Architecture: ARM
     Hash algo:    sha256
     Hash value:   4e338db79f27d05fc666aaffce73c5e03562562da7b59913eec2b374aaef4b07
   Verifying Hash Integrity ... sha256+ OK
   Booting using the fdt blob at 0x204b59a4
   Loading Kernel Image ...OK
   Loading Device Tree to 3f95d000, end 3f9682c7 ...OK
Starting kernel ...
```

6. AT91bootstrap 配置

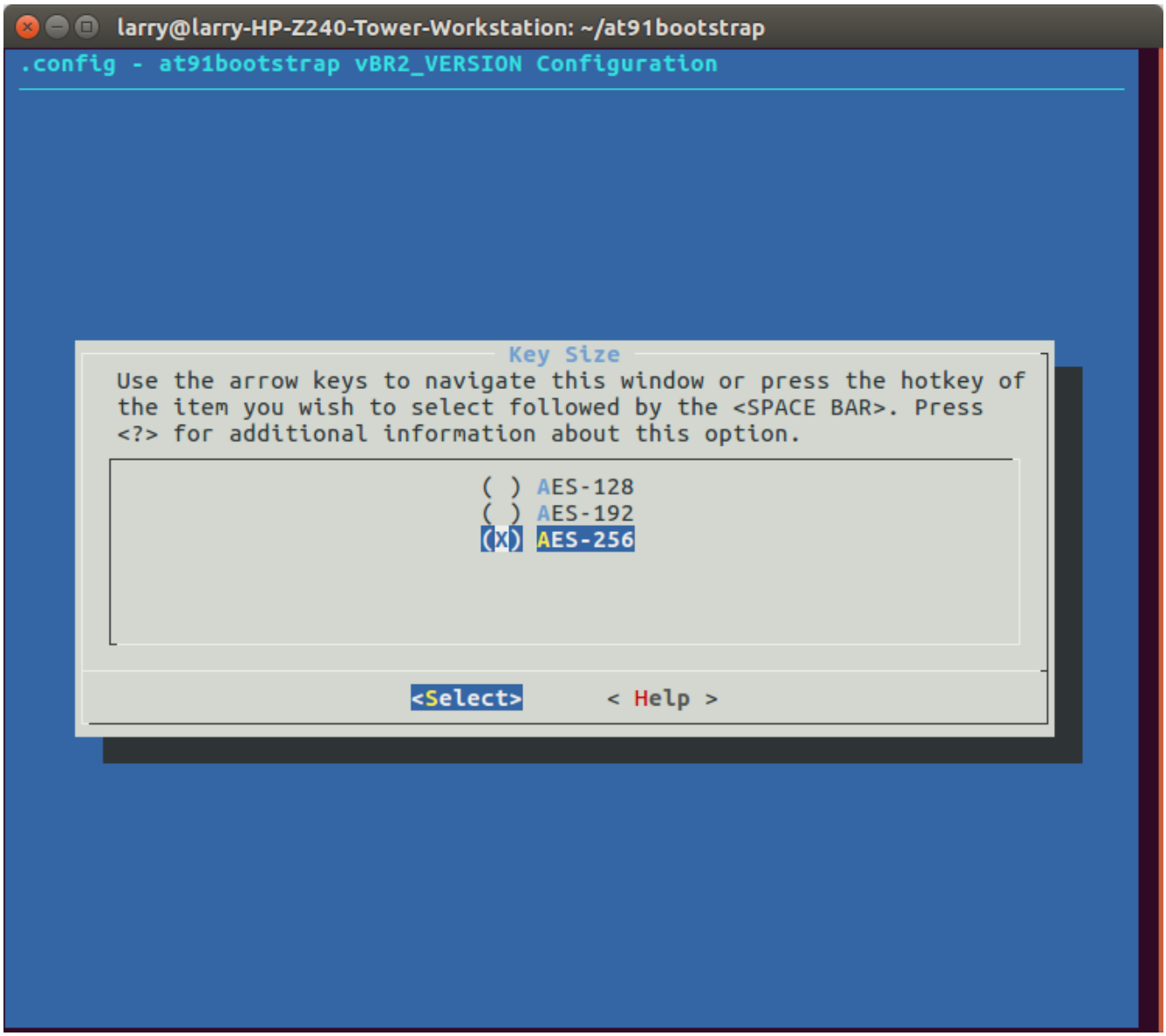
现在已使 U-Boot 仅引导已签名的映像，必须配置 AT91bootstrap 程序才能进行安全引导。要配置 AT91bootstrap 程序，请运行以下命令：

```
$ make at91bootstrap3-menuconfig
```

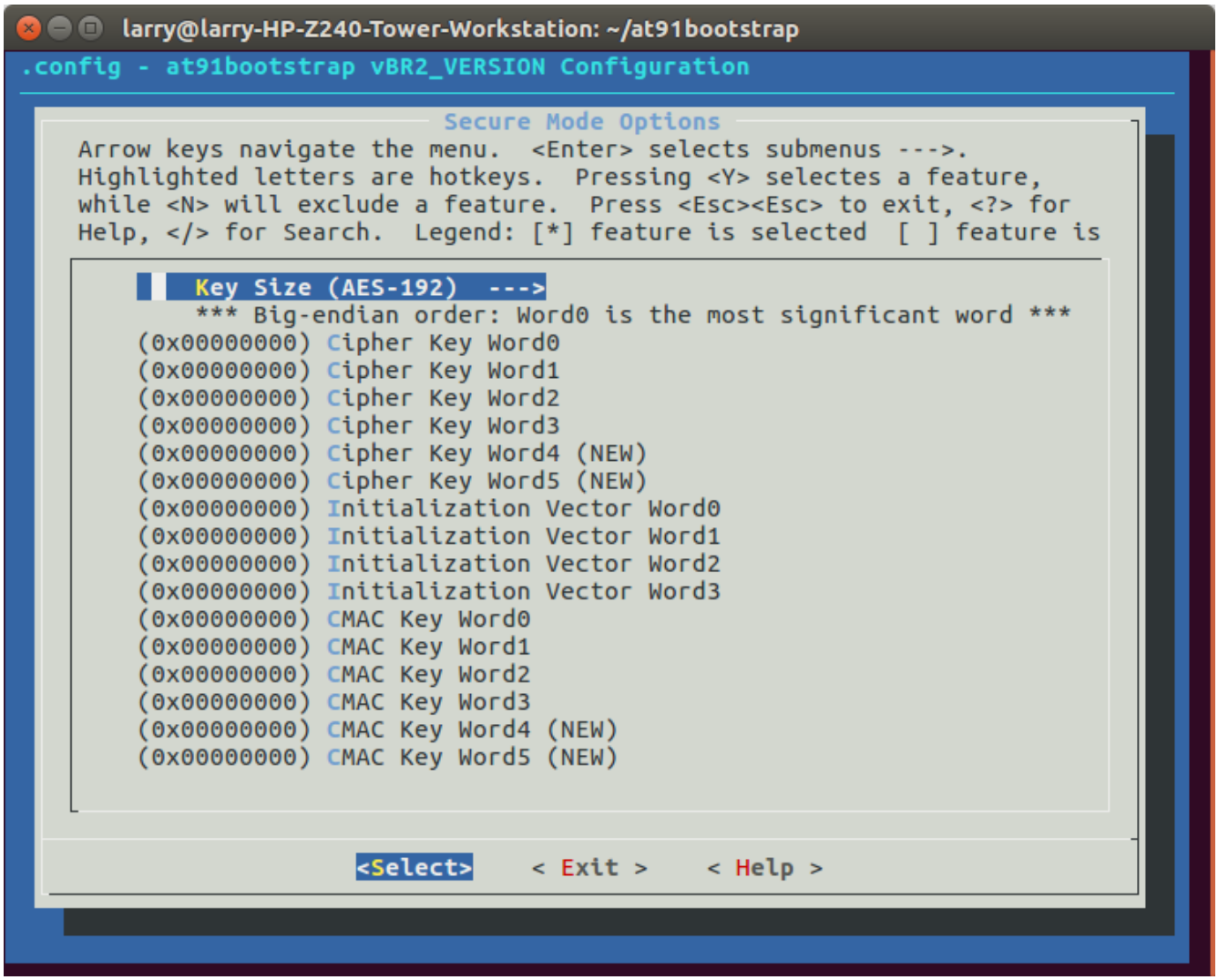
使能“安全模式支持”，并确保选择的密钥大小与将在系统中使用的引导配置相匹配。对于所有密钥大小，初始化向量均保持不变（128 位，4 个条目）。



加密和验证的密钥大小选项如下图所示。

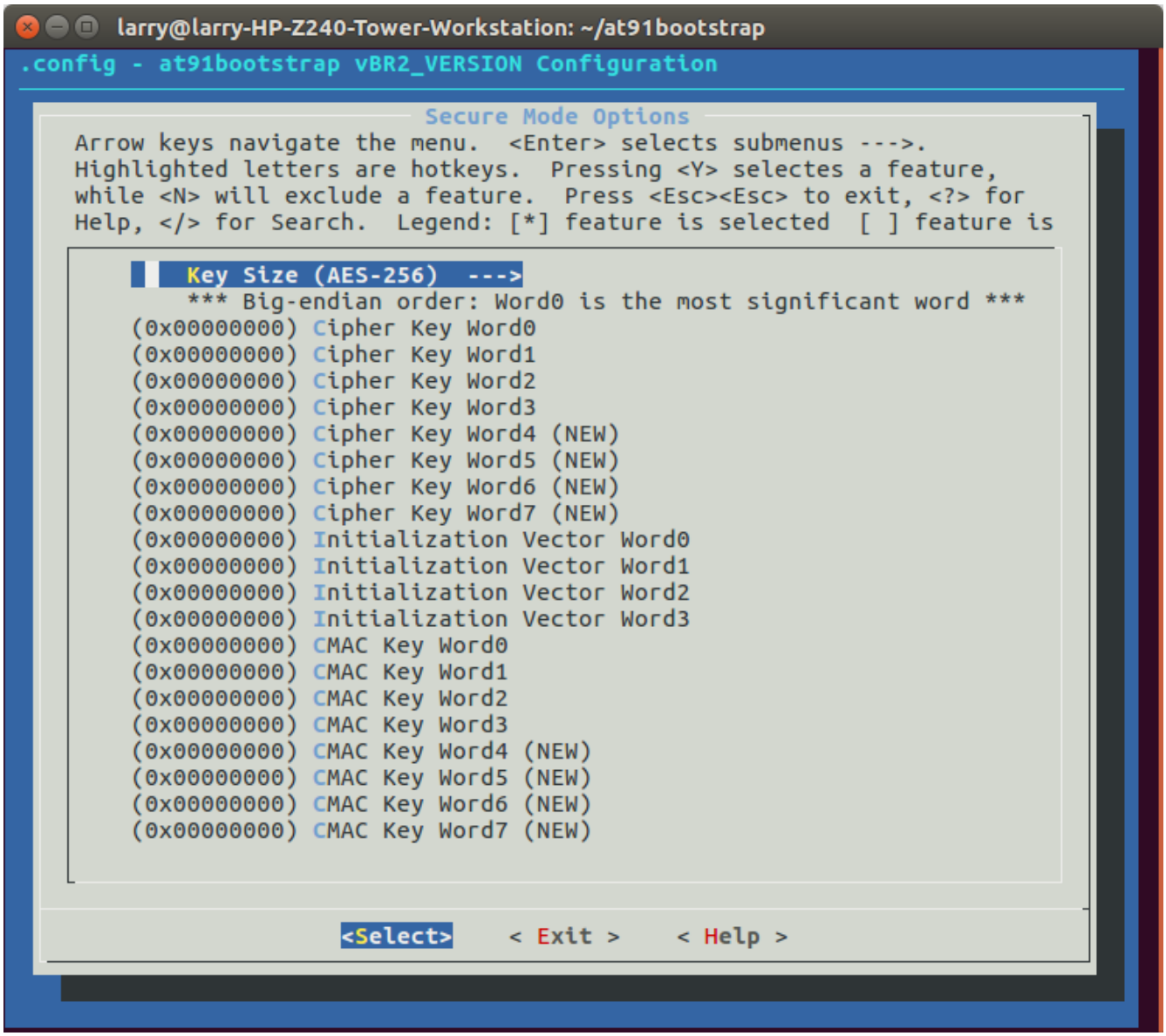


对于 128 位密钥，密码密钥和 CMAC 密钥都是 128 位。



选择 192 位密钥时，密码密钥和 CMAC 密钥都需要额外输入 64 位。

选择 256 位密钥时，密码密钥和 CMAC 密钥长度均为 8 个字。



确保要引导的 Linux 映像在 AT91bootstrap 配置中命名为 zImage.cip，以便与应用程序加密步骤中将使用的名称相匹配。

6.1 编译 AT91bootstrap

配置 AT91bootstrap 后，运行“make”以编译 boot.bin 文件：

```
$ make
```

这将编译可引导加密 U-Boot 的未加密引导文件。最好在将 SAMA5 置于安全引导模式之前测试自举程序。

6.2 安全 SAM-BA 工具

Microchip 根据 NDA 提供加密和签名应用程序时所需的工具。

欲了解更多信息，请联系您当地的 Microchip 销售办事处。

安全 SAM-BA 工具由两部分组成。

1. 安全 sam-ba 密码——在开发和准备安全引导密钥、AT91bootstrap 程序和应用程序期间执行各种操作的实用程序
2. 安全 sam-ba 加载程序——用于与终端平台通信以配置器件、装载密钥、哈希值、AT91bootstrap 程序和应用程序的实用程序

6.3 许可证请求

在运行安全 SAM-BA 工具以执行加密操作之前，负责执行签名的主机必须具备有效的许可证文件。许可证文件仅限于单个主机，每次运行 secure-sam-ba-cipher 时都会对其进行检查。

安全 SAM-BA 密码用于生成许可证请求，并且必须在将用于执行安全 SAM-BA 密码操作（例如为 bootstrap 程序和应用程序签名）的主机中运行。

此许可证请求必须由 Microchip 验证，通过验证后会有一个许可证文件发送回请求方。

示例：

以下命令行将在 request.txt 文件中生成许可证请求。

```
secure-sam-ba-cipher.py request-license -o request.txt
```

6.4 加密/签名 U-Boot

除了有效的许可证文件之外，安全 SAM-BA“application”命令还需要应用程序密钥。虽然签名计算机上只能使用单个许可证文件，但可使用的应用程序密钥数量没有限制。需要注意的是，应用程序密钥的大小和值必须与 AT91bootstrap 程序中配置的大小和值完全匹配。

6.5 应用程序密钥文件格式

app.key 文件必须包含十六进制的应用程序密钥，格式如下：

```
KEY=00000000000000000000000000000000000000000000000000000000000000000000000000000000
IV_KEY=00000000000000000000000000000000
CMAC_KEY=00000000000000000000000000000000000000000000000000000000000000000000000000000000
```

AES 密钥和 AES CMAC 密钥的长度可以是以下几种：

- 128 位（16 字节，32 个十六进制数字）
- 192 位（24 字节，48 个十六进制数字）
- 256 位（32 字节，64 个十六进制数字）

AES IV 始终为 128 位（16 字节，32 个十六进制数字）。

6.6 应用程序加密/签名示例

在此示例中，AT91bootstrap 程序配置为使用 128 位 AES 安全地引导名为“uboot.cip”的 U-boot 映像。必须为 AT91bootstrap 程序配置正确的密钥大小选项，并确保 AT91bootstrap 配置文件中的密钥值与安全 SAM-BA“application”命令使用的密钥文件相匹配。此示例假设已请求安全 SAM-BA 许可证文件，并从 Microchip 收到了有效的许可证文件。

6.7 应用程序密钥文件

此示例中的密钥文件名为 zImage.key，采用 128 位密钥格式。

```
$ cat uboot.key
KEY=1234567811111111abcdef01ffffffff
```

```
IV_KEY=00000001000000020000000300000004
CMAC_KEY=876543211111111110fedcbaffffffff
```

6.8 AT91bootstrap .config 文件

AT91bootstrap 的.config 文件使用与 uboot.key 文件匹配的值进行配置。

```
CONFIG_SECURE=y
#
# Secure Mode Options
#
CONFIG_AES_KEY_SIZE_128=y
# CONFIG_AES_KEY_SIZE_192 is not set
# CONFIG_AES_KEY_SIZE_256 is not set
#
# Big-endian order: Word0 is the most significant word
#
CONFIG_AES_CIPHER_KEY_WORD0=0x12345678
CONFIG_AES_CIPHER_KEY_WORD1=0x11111111
CONFIG_AES_CIPHER_KEY_WORD2=0xabcdef01
CONFIG_AES_CIPHER_KEY_WORD3=0xffffffff
CONFIG_AES_IV_WORD0=0x00000001
CONFIG_AES_IV_WORD1=0x00000002
CONFIG_AES_IV_WORD2=0x00000003
CONFIG_AES_IV_WORD3=0x00000004
CONFIG_AES_CMAC_KEY_WORD0=0x87654321
CONFIG_AES_CMAC_KEY_WORD1=0x11111111
CONFIG_AES_CMAC_KEY_WORD2=0x10fedcba
CONFIG_AES_CMAC_KEY_WORD3=0xffffffff
```

6.9 运行 secure-sam-ba-cipher“application”命令

“application”命令将应用程序密钥文件和应用程序作为输入，并创建加密和签名的输出文件。

```
$ secure-sam-ba-cipher.py application -l license_D2.txt -k uboot.key -i uboot.bin -o uboot.cip
```

6.10 测试应用程序

使用配置为引导安全 U-Boot 映像的新 AT91bootstrap 程序替换 SD 卡上的 AT91bootstrap 程序。

将以下文件复制到 SD 卡的 FAT 分区：

- boot.bin——新 AT91bootstrap 程序
- uboot.cip——新的已加密和签名的 U-Boot 映像，与 boot.bin 中的配置匹配

将 SD 卡置于 SAMA5 开发板上并确保映像能够正常引导。

6.10.1 保护 AT91bootstrap 程序

在 AT91bootstrap 程序正确引导 uboot.cip 文件后，可以从该 boot.bin 创建一个名为 boot.cip 的安全 AT91bootstrap 二进制文件。正如 uboot.cip 文件需要应用程序密钥一样，AT91bootstrap 程序也需要一个密钥，供 ROM 代码用于安全地验证和解密 AT91bootstrap 映像。ROM 代码解密和验证 AT91bootstrap 程序时所使用的密钥源自一个名为“customer key”的密钥。

6.11 AT91bootstrap 密钥文件格式

cust.key 文件必须包含十六进制的客户密钥，格式如下：

```
KEY_CUST=0000000000000000000000000000000000000000000000000000000000000000
```

SAMA5D2 的客户密钥长度为 256 位。

6.12 运行 secure-sam-ba-cipher“customer-key”命令

“customer key”命令将 AT91bootstrap 密钥文件作为输入，并创建一个加密和签名的输出文件（最终通过一个单独步骤装载到 SAMA5 MPU）。

```
secure-sam-ba-cipher.py customer-key -d sama5d2x -l license_D2.txt -k cust.key -o customer-key.cip
```

6.13 运行 secure-sam-ba-cipher“bootstrap”命令

“bootstrap”命令将 bootstrap 密钥（客户密钥）文件和 bootstrap 二进制文件作为输入，并创建一个加密和签名的输出文件。

```
$ secure-sam-ba-cipher.py bootstrap -d sama5d2x -l license_D2.txt -k cust.key -i boot.bin -o boot.cip
```

6.14 测试 AT91bootstrap 程序

boot.cip 文件是之前在非安全模式下测试的 boot.bin 的加密和签名版本。

将 boot.cip 文件复制到 SD 卡的 FAT 分区。

6.15 使用 secure-sam-ba-loader 配置电路板

之前，已使用 secure-sam-ba-cipher 程序准备以下加密/签名文件：

- zImage.cip——加密和签名的 Linux 映像
- boot.cip——加密和签名的 AT91bootstrap 程序
- customer-key.cip——加密和签名的 AT91bootstrap 密钥

zImage.cip 文件和 boot.cip 文件被复制到引导介质——在我们的示例中为 SD 卡。customer-key.cip 文件存储在 SAMA5 MPU 中，不应存储在引导介质上。

应使用 secure-sam-ba-loader 实用程序将 SAMA5 置于安全模式。在运行命令之前，请确保最终设备正在运行 SAM-BA 监视器，并确保 USB 电缆正确连接到主机。请注意插入 USB 电缆时枚举的设备名称。在此示例中，控制台是/dev/ttyACM0，SAM-BA 接口是/dev/ttyACM1。

出现 RomBOOT 提示后，使用 secure-sam-ba-loader 将 MPU 置于安全引导模式：

```
$ secure-sam-ba-loader.py secure-mode -d sama5d2x -p /dev/ttyACM1
```

运行 secure-sam-ba-loader“secure-mode”命令后，应在复位后出现如下提示：

```
Secure Boot Mode
```

下一步是将客户密钥装载到 SAMA5 MPU 中。

```
$ secure-sam-ba-loader.py customer-key -d sama5d2x -p /dev/ttyACM1 -i customer-key.cip
```

6.16 测试映像

将 SD 卡置于电路板中，然后按 Reset（复位）。控制台上应显示“Secure Boot Mode”（安全引导模式）提示，然后依次运行 AT91bootstrap 和 Linux。此时，所有测试均已使用 MPU 的备份寄存器之一（BUREG0）执行。如果将器件掉电，将撤消所有安全模式设置。

6.17 烧写熔丝

当 AT91bootstrap 程序在安全模式下正常工作后，可以使用以下命令在器件熔丝中将 secure-mode 位永久置 1：

```
$ secure-sam-ba-loader.py secure-mode --fuse -d sama5d2x -p /dev/ttyACM1
Connecting to serial port /dev/ttyACM1...
Connected to /dev/ttyACM1.
Enabling secure mode on sama5d2x...
Secure mode successfully enabled, please power cycle the device.
```

然后，可使用以下命令将 AT91bootstrap 密钥永久编程到熔丝中：

```
$ secure-sam-ba-loader.py customer-key -d sama5d2x -p /dev/ttyACM1 --fuse -i customer-key.cip
```

从现在起，器件将永久处于安全引导模式。

请注意，熔丝编程是一种永久性操作，无法撤消。

7. 更多步骤

本应用笔记介绍了安全引导 SAMA5 MPU 系统的过程。用于加密和签名的设置仅限于 128 位 AES 和 AES CMAC。关于其他可供使用的验证模式，可参见遵循 NDA 提供的安全引导包。

8. 版本历史

8.1 版本 A——2018 年 06 月

本应用笔记的初始版本。

Microchip 网站

Microchip 网站 http://www.microchip.com/为客户提供在线支持。客户可通过该网站方便地获取文件和信息。只要使用常用的互联网浏览器即可访问，网站提供以下信息：

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持**——常见问题（FAQ）、技术支持请求、在线讨论组以及 Microchip 顾问计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

变更通知客户服务

Microchip 的变更通知客户服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请登录 Microchip 网站 http://www.microchip.com/。在“支持”（Support）下，点击“变更通知客户”（Customer Change Notification）服务后按照注册说明完成注册。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师（FAE）
- 技术支持

客户应联系其代理商、代表或应用工程师（FAE）寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过以下网站获得技术支持： http://www.microchip.com/support

Microchip 器件代码保护功能

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下，Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿意与关心代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。

代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案（Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

法律声明

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc.及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc.的英文原版文档。

商标

ISBN： 978-1-5224-4680-4

质量管理体系

有关 Microchip 质量管理体系的更多信息，请访问 www.microchip.com/quality。

全球销售及服务网点

美洲	亚太地区	亚太地区	欧洲
公司总部 2355 West Chandler Blvd. 钱德勒，亚利桑那州 85224-6199 电话：480-792-7200 传真：480-792-7277 技术支持： http://www.microchip.com/support 网址： www.microchip.com **亚特兰大** 德卢斯，佐治亚州 电话：678-957-9614 传真：678-957-1455 **奥斯汀，德克萨斯州** 电话：512-257-3370 **波士顿** 韦斯特伯鲁，马萨诸塞州 电话：774-760-0087 传真：774-760-0088 **芝加哥** 艾塔斯卡，伊利诺伊州 电话：630-285-0071 传真：630-285-0075 **达拉斯** 阿迪森，德克萨斯州 电话：972-818-7423 传真：972-818-2924 **底特律** 诺维，密歇根州 电话：248-848-4000 **休斯顿，德克萨斯州** 电话：281-894-5983 **印第安纳波利斯** 诺布尔斯维尔，印第安纳州 电话：317-773-8323 传真：317-773-5453 电话：317-536-2380 **洛杉矶** 米慎维荷，加利福尼亚州 电话：949-462-9523 传真：949-462-9608 电话：951-273-7800 **罗利，北卡罗来纳州** 电话：919-844-7510 **纽约，纽约州** 电话：631-435-6000 **圣何塞，加利福尼亚州** 电话：408-735-9110 电话：408-436-4270 **加拿大 - 多伦多** 电话：905-695-1980 传真：905-695-2078	**澳大利亚 - 悉尼** 电话：61-2-9868-6733 **中国 - 北京** 电话：86-10-8569-7000 **中国 - 成都** 电话：86-28-8665-5511 **中国 - 重庆** 电话：86-23-8980-9588 **中国 - 东莞** 电话：86-769-8702-9880 **中国 - 广州** 电话：86-20-8755-8029 **中国 - 杭州** 电话：86-571-8792-8115 **中国 - 香港特别行政区** 电话：852-2943-5100 **中国 - 南京** 电话：86-25-8473-2460 **中国 - 青岛** 电话：86-532-8502-7355 **中国 - 上海** 电话：86-21-3326-8000 **中国 - 沈阳** 电话：86-24-2334-2829 **中国 - 深圳** 电话：86-755-8864-2200 **中国 - 苏州** 电话：86-186-6233-1526 **中国 - 武汉** 电话：86-27-5980-5300 **中国 - 西安** 电话：86-29-8833-7252 **中国 - 厦门** 电话：86-592-2388138 **中国 - 珠海** 电话：86-756-3210040	**印度 - 班加罗尔** 电话：91-80-3090-4444 **印度 - 新德里** 电话：91-11-4160-8631 **印度 - 浦那** 电话：91-20-4121-0141 **日本 - 大阪** 电话：81-6-6152-7160 **日本 - 东京** 电话：81-3-6880- 3770 **韩国 - 大邱** 电话：82-53-744-4301 **韩国 - 首尔** 电话：82-2-554-7200 **马来西亚 - 吉隆坡** 电话：60-3-7651-7906 **马来西亚 - 槟榔屿** 电话：60-4-227-8870 **菲律宾 - 马尼拉** 电话：63-2-634-9065 **新加坡** 电话：65-6334-8870 **台湾地区 - 新竹** 电话：886-3-577-8366 **台湾地区 - 高雄** 电话：886-7-213-7830 **台湾地区 - 台北** 电话：886-2-2508-8600 **泰国 - 曼谷** 电话：66-2-694-1351 **越南 - 胡志明市** 电话：84-28-5448-2100	**奥地利 - 韦尔斯** 电话：43-7242-2244-39 传真：43-7242-2244-393 **丹麦 - 哥本哈根** 电话：45-4450-2828 传真：45-4485-2829 **芬兰 - 埃斯波** 电话：358-9-4520-820 **法国 - 巴黎** 电话：33-1-69-53-63-20 传真：33-1-69-30-90-79 **德国 - 加兴** 电话：49-8931-9700 **德国 - 哈恩** 电话：49-2129-3766400 **德国 - 海尔布隆** 电话：49-7131-72400 **德国 - 卡尔斯鲁厄** 电话：49-721-625370 **德国 - 慕尼黑** 电话：49-89-627-144-0 传真：49-89-627-144-44 **德国 - 罗森海姆** 电话：49-8031-354-560 **以色列 - 若那那市** 电话：972-9-744-7705 **意大利 - 米兰** 电话：39-0331-742611 传真：39-0331-466781 **意大利 - 帕多瓦** 电话：39-049-7625286 **荷兰 - 德卢内市** 电话：31-416-690399 传真：31-416-690340 **挪威 - 特隆赫姆** 电话：47-72884388 **波兰 - 华沙** 电话：48-22-3325737 **罗马尼亚 - 布加勒斯特** 电话：40-21-407-87-50 **西班牙 - 马德里** 电话：34-91-708-08-90 传真：34-91-708-08-91 **瑞典 - 哥德堡** 电话：46-31-704-60-40 **瑞典 - 斯德哥尔摩** 电话：46-8-5090-4654 **英国 - 沃金厄姆** 电话：44-118-921-5800 传真：44-118-921-5820