玄铁安全系统安全工具用户手册

文档编号 STG30003

版权声明

Notice

The purchased products, services and features are stipulated by the contract made between T-HEAD and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

平头哥半导体有限公司 T-HEAD Semiconductor Co.,LTD
地址:杭州市余杭区向往街1122号欧美金融城(EFC)英国中心西楼T6 43层 邮编: 311121
网址:www.T-head.cn

修订历史

日期	版本	新增修改内容	作者
20230216	v1.2	• 新增boot.ext4签名功能	肖佳涛
20220420	v1.1	• 新增KDF工具	夏狼
20220328	v1.0	• 支持sectool工具说明	夏狼
20220311	v0.1	• 初版	夏狼

目录

[TOC]

1. 工具总览

工具名	说明	版本	仓库地址/附件

product	镜像签名工具	>= v1.0.23	随yoctool一起安装，参考yoctool用户手册： https://yuque.antfin-inc.com/occ/xyrz27/cywaud https://code.aone.alibaba-inc.com/thead/yoc_tools
efuse_utility	eFuse烧写工具	v1.2	仓库地址：git@gitlab.alibaba-inc.com:yocopen/efusehost.git 工具包：📎efuse_host_v1.2.tar.zip 工具包md5um: 5d8257ce606232f21a69110e1654f192
	efuse key加密工具	v1.0	📎efuse加解密脚本及使用方法说明 v1.0.7z
fastboot	镜像升级工具	v1.0	📎fastboot v1.0.zip
sectool	imagesign.sh	v1.0	git@gitlab.alibaba-inc.com:thead-os-platform/sectool.git
	bin2ext4.sh		
	kdf_gen.exe		

2. 工具介绍

2.1 product

product用于系统镜像的签名，支持指定的密钥证书等功能。

功能命令:

sigx

说明:

对已生成好的单个镜像(或者公钥PEM文件)进行签名。

输入product sigx可查看帮助信息。

示例:

2级公钥的签名

- 国际算法例子:

product sigx keystore/pubkeyB.pem -pvk keystore/privatekeyA.pem -pubk keystore/pubkeyA.pem -ss RSA2048 -ds SHA256 -npubk keystore/pubkeyB.pem -nss

RSA2048 –nds SHA256 –o sign_2nd_pubkey.bin

- 国密算法例子：

product sigx keystore_sm/pubkeyB.pem –pvk keystore_sm/privatekeyA.pem –pubk keystore_sm/pubkeyA.pem –ss SM2 –ds SM3 –npubk keystore_sm/pubkeyB.pem –nss SM2 –nds SM3 –o sign_2nd_pubkey.bin

镜像的签名(带有下级公钥)

- 国际算法例子：

product sigx iw.bin –pvk keystore/privatekeyB.pem –ss RSA2048 –ds SHA256 –npubk keystore/pubkeyC.pem –nss RSA2048 –nds SHA256 –iv 0 –ra 0xFFE0000000 –o sign_iw.bin

- 国密算法例子：

product sigx iw.bin –pvk keystore_sm/privatekeyB.pem –ss SM2 –ds SM3 –npubk keystore_sm/pubkeyC.pem –nss SM2 –nds SM3 –iv 0 –ra 0xFFE0000000 –o sign_iw.bin

镜像的加密签名(带有下级公钥)

- 国际算法例子：

product sigx iw.bin –pvk keystore/privatekeyB.pem –ss RSA2048 –ds SHA256 –npubk keystore/pubkeyC.pem –nss RSA2048 –nds SHA256 –ent AES_256_CBC –enk keystore/aes_256_cbc.key –iv 0 –ra 0xFFE0000000 –o sign_iw.bin

- 国密算法例子：

product sigx iw.bin –pvk keystore_sm/privatekeyB.pem –ss SM2 –ds SM3 –npubk keystore_sm/pubkeyC.pem –nss SM2 –nds SM3 –ent SM4_CBC –enk keystore_sm/sm4.key –iv 0 –ra 0xFFE0000000 –o sign_iw.bin

镜像的签名(不带有下级公钥)

- 国际算法例子：

product sigx iw.bin –pvk keystore/privatekeyB.pem –ss RSA2048 –ds SHA256 –iv 0 –ra 0xFFE0000000 –o sign_iw.bin

- 国密算法例子：

product sigx iw.bin –pvk keystore_sm/privatekeyB.pem –ss SM2 –ds SM3 –iv 0 –ra 0xFFE0000000 –o sign_iw.bin

镜像的加密签名(不带有下级公钥)

- 国际算法例子：

product sigx iw.bin –pvk keystore/privatekeyB.pem –ss RSA2048 –ds SHA256 –ent AES_256_CBC –enk keystore/aes_256_cbc.key –iv 0 –ra 0xFFE0000000 –o sign_iw.bin

- 国密算法例子：

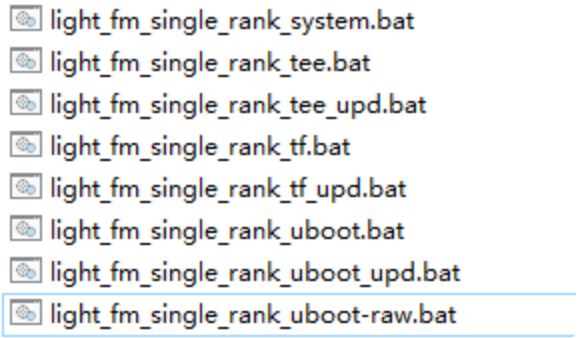
product sigx iw.bin –pvk keystore_sm/privatekeyB.pem –ss SM2 –ds SM3 –ent SM4_CBC –enk keystore_sm/sm4.key –iv 0 –ra 0xFFE0000000 –o sign_iw.bin

2.2 efuse utilty

efuse utility是用于芯片efuse初次烧写，利用SPI Blaster来进行烧写，运行在Windows下的一个工具。

2.3 fastboot

fastboot是一种比recovery更底层的刷机模式（俗称引导模式）。就是使用USB数据线连接终端的一种刷机模式。我们利用fastboot进行系统镜像的更新。



- light_fm_single_rank_system.bat – 用于直接烧写所有系统镜像
- light_fm_single_rank_uboot.bat – 用于直接烧写uboot，不进行升级流程
- light_fm_single_rank_uboot_upd.bat – 用于更新uboot镜像，进行升级路程
- light_fm_single_rank_uboot_raw.bat – 用于更新烧写uboot镜像(uboot损坏)，不进行升级流程
- light_fm_single_rank_tee.bat – 用于直接烧写tee，不进行升级流程
- light_fm_single_rank_tee_upd.bat – 用于直接更新tee镜像，进行升级流程
- light_fm_single_rank_tf.bat – 用于直接烧写tf，不进行升级流程
- light_fm_single_rank_tf_upd.bat – 用于直接更新tf镜像，进行升级流程

注意：

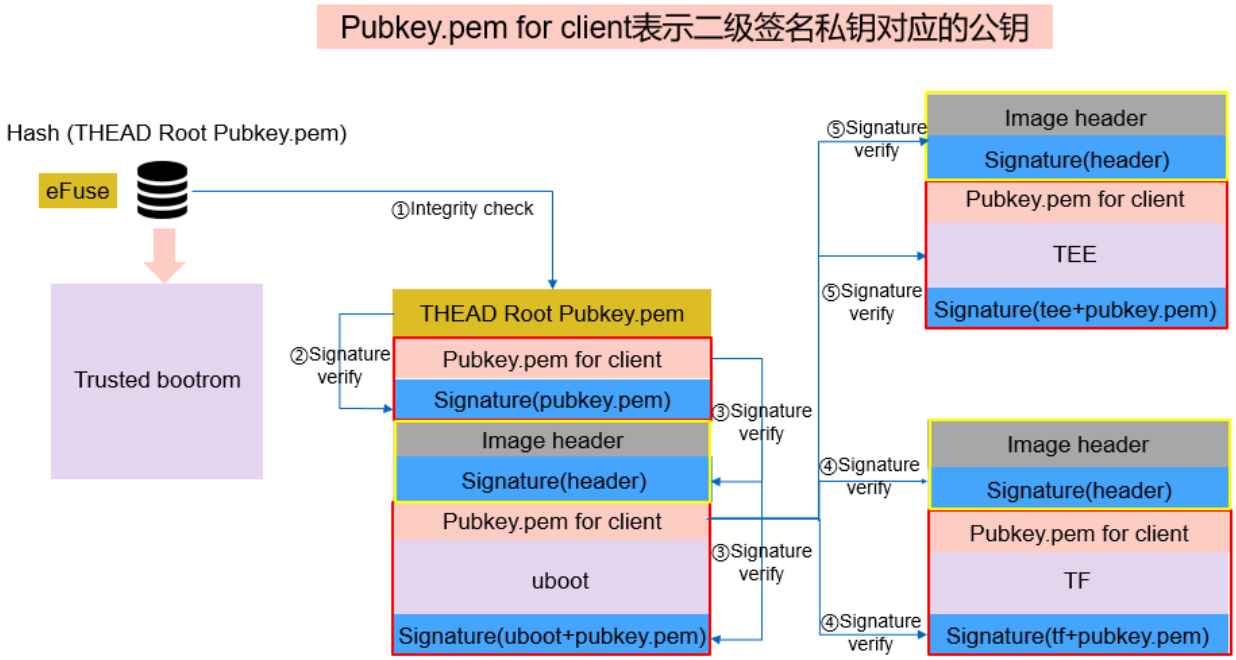
Fastboot工具使用前需要安装ADB驱动，请参考README指导安装。

2.4 sectool

sectool用于镜像签名打包的工具，包括支持二进制文件打包成EXT4文件。其主要包括imagesign.sh和bin2ext4.sh脚本。

```bash
cxx194832@docker-ubuntu18:~/sectool$ ls
bin2ext4  imagesign.sh  keystore  keystore_sm  README.md  tee  tf  tool  uboot
```

2.4.1 签名证书



在进行镜像签名前，可以在imagesign.sh脚本里通过修改以下两个变量来指定密钥证书。

- `client_public_cert`
- `client_private_cert`

注意：国际算法证书必须放在keystore文件夹下，国密算法证书必须放在keystore_sm文件夹下。以下证书用于uboot镜像签名，一般情况下由平头哥进行提供管理:

- `thead_root_public_cert`
- `thead_root_private_cert`
- `thead_b1_public_cert`
- `thead_b1_private_cert`

2.4.2 imagesign

imagesign工具是一个运行在shell下的一个脚本文件，其用于将一个二进制文件用指定的算法进行签名，同时还能指定镜像是否加密和版本号等信息，最后进行镜像内容和签名数据等信息打包，生成一个签名文件,注意最后一个参数image board type参数含义为镜像对应板子板子类型。

```bash
imagesign.sh utility version v1.0
The utility is designed for aim to help user generate new image file
with signature with desired sign scheme.

Usage:
imagesign.sh [chkuboot] [ia/sm] [enc/nor] [tf/tee/uboot] [ver] [image board type]
chkuboot: check uboot binary file is signed or not, if yes, it generates the original bboot binary file
signed algorithms
  ia - international algorithm
  sm - china algorithms
secure attribution
  enc - signed image with encryption
  nor - signed image without encryption
image file type
  tf   - trust_firmware binary image
  tee  - tee binary image
  uboot - uboot binary image
  ree - ree binary image
version
  ver - image version (x.y), eg 1.1, 2.1
image board type
  LA  - board light A
  LB  - board light B
  LP  - board lpi4a
  LG  - board beagle
  LD  - board ant_ref
```

2.4.2.1 生成SBMETA签名镜像

1. 需要签名的 `sbmeta.bin` 必须在 `./sbmeta` 文件夹下。 `sbmeta.bin` 可以由sectool默认生成，也可以由用户自定义生成。生成步骤为：
   a. 更改 `sbmeta.yaml` 文件，指定校验镜像

```bash
image_configs:
  #config of trust_firmware.bin
  -
    name: trust_firmware.bin
    device: 0
    partition: 3
    type: tf
    digest_scheme: sha256
    sign_scheme: rsa_2048
    is_image_encrypted: 0
    medium_type: emmc
    image_path: ./tf/trust_firmware.bin #recommend to this default path

  #config of tee.bin
  -
    name: tee.bin
    device: 0
    partition: 4
    type: tee
    digest_scheme: sha256
    sign_scheme: rsa_2048
    is_image_encrypted: 0
    medium_type: emmc
    image_path: ./tee/tee.bin #recommend to this default path

  #config of dtb
  -
    name: light-lpi4a-sec.dtb
    device: 0
    partition: 2
    type: dtb
    digest_scheme: sha256
    sign_scheme: rsa_2048
    is_image_encrypted: 0
    medium_type: emmc
    image_path: ./dtb/light-lpi4a-sec.dtb
```

b. 生成sbmeta.bin镜像

```bash
kjs@kjs-VirtualBox:~/sectool$ python sbmeta.py

# 查看生成的镜像
kjs@kjs-VirtualBox:~/sectool/sbmeta$ ls sbmeta
sbmeta.bin
```

注意：文件名必须是 `sbmeta.bin`， sbmeta文件夹名字不可以更改

2. 执行下面命令进行国际镜像算法签名

```bash
kjs@kjs-VirtualBox:~/sectool$ ./imagesign.sh ia nor sbmeta 1.0
Dump all parameters from user input.
-----------------------------------
Signed algorithem: ia
Secure attribution: nor
Image file: sbmeta
Image version: 1.0
-----------------------------------
Enter into image sign process ...
Start sbmeta Image (1.0) signed with international algorithms with secure attr (nor)
sign tool path: ./tool/product
Original file: sbmeta/sbmeta.bin
Signed file: sbmeta/signed_ia_nor_v1.0_sbmeta.bin
Image Version: 256
Relocate Addr: 0x100000
[2023-08-16 11:17:56] Sign a image file.
[2023-08-16 11:17:56] Sign ok.
Exit from image sign process ...
```

3. 查看生成的SBMETA签名镜像

```bash
kjs@kjs-VirtualBox:~/sectool$ ls sbmeta
sbmeta.bin  signed_ia_nor_v1.0_sbmeta.bin
```

生成的 `signed_ia_nor_v1.0_sbmeta.bin` 位于sbmeta文件夹下。

注意：

- **使用的时候必须将 `signed_ia_nor_v1.0_sbmeta.bin` 改名为 `sbmeta.bin`**

2.4.2.2 生成UBOOT签名镜像

1. 将需要签名的u-boot-with-spl.bin复制到uboot文件夹里

注意：文件名必须是 `u-boot-with-spl.bin`， uboot文件夹名字不可以更改

```bash
cxx194832@docker-ubuntu18:~/sectool$ ls uboot/
u-boot-with-spl.bin
```

2. 执行下面命令进行镜像签名

```bash
cxx194832@docker-ubuntu18:~/sectool$ ./imagesign.sh ia enc uboot 1.2
Dump all parameters from user input.
-----------------------------------
Signed algorithem: ia
Secure attribution: enc
Image file: uboot
Image version: 1.2
-----------------------------------
Enter into image sign process ...
Start uboot Image (1.2) signed with international algorithms with secure attr (enc)
sign tool path: ./tool/product
Original file: uboot/u-boot-with-spl.bin
Signed file: uboot/signed_ia_enc_u-boot-with-spl.bin
Image Version: 258
Relocate Addr: 0xFFE0000800
[2022-03-23 14:09:44] Sign a public key file.
[2022-03-23 14:09:44] rsa verify ok....
[2022-03-23 14:09:44] rsa verify ok....
[2022-03-23 14:09:44] Sign ok.
[2022-03-23 14:09:44] Sign a image file.
[2022-03-23 14:09:44] Sign ok.
Exit from image sign process ...

```

3. 查看生成的uboot签名镜像

```bash
cxx194832@docker-ubuntu18:~/sectool$ ls uboot/
signed_ia_enc_u-boot-with-spl.bin  signed_image_u-boot-with-spl.bin
signed_pubkey_u-boot-with-spl.bin  u-boot-with-spl.bin
```

生成的 `signed_ia_enc_u-boot-with-spl.bin` 位于uboot文件夹下。

注意：

- uboot签名镜像由 **`signed_pubkey_u-boot-with-spl.bin`** 和 **`signed_image_u-boot-with-spl.bin`** 两部分签名镜像组成。
- **`signed_pubkey_u-boot-with-spl.bin`** 由平头哥管理
- **`signed_image_u-boot-with-spl.bin`** 用于镜像升级，但签名镜像的私钥必须和 **`signed_pubkey_u-boot-with-spl.bin`** 了用的公钥配对。
- 使用的时候必须将 **`signed_ia_enc_u-boot-with-spl.bin`** 改名为 **`u-boot-with-spl.bin`**
- 当不确定当前的u-boot-with-spl.bin是否已经签名，可以通过 **`imagesign.sh chkuboot u-boot-with-spl.bin`** 命令进行检查，如果已经签名，则会生成原始未签名的u-boot-with-spl.bin文件(注:目前chkuboot不支持加密镜像的还原,即国密加密或者国际加密镜像无法使用chkuboot进行还原)，可以使用该文件进行其他算法的签名。
- 如果要支持镜像烧写防错措施，需要给签名命令里加上Board ID参数，该参数会作为板子的Identifier ID。

2.4.2.3 生成TF签名镜像

1. 将需要签名的trust_firmware.bin复制到tf文件夹里

注意：文件名必须是 `trust_firmware.bin`， tf文件夹名字不可以更改

```bash
cxx194832@docker-ubuntu18:~/sectool$ ls tf
trust_firmware.bin
```

2. 执行下面命令进行镜像签名

```bash
cxx194832@docker-ubuntu18:~/sectool$ ./imagesign.sh ia nor tf 1.2
Dump all parameters from user input.
-----------------------------------
Signed algorithem: ia
Secure attribution: nor
Image file: tf
Image version: 1.2
-----------------------------------
Enter into image sign process ...
Start tf Image (1.2) signed with international algorithms with secure att
r (nor)
sign tool path: ./tool/product
Original file: tf/trust_firmware.bin
Signed file: tf/signed_ia_nor_trust_firmware.bin
Image Version: 258
Relocate Addr: 0x0
[2022-03-23 14:12:36] Sign a image file.
[2022-03-23 14:12:36] Sign ok.
Exit from image sign process ...
```

3. 查看生成的TF签名镜像

```bash
cxx194832@docker-ubuntu18:~/sectool$ ls tf
signed_ia_nor_v1.2_trust_firmware.bin  trust_firmware.bin
```

生成的 `signed_ia_nor_trust_firmware.bin` 位于tf文件夹下。

注意：

- **使用的时候必须将 `signed_ia_nor_v1.2_trust_firmware.bin` 改名为 `trust_firmwar e.bin`**

2.4.2.4 生成TEE签名镜像

1. 将需要签名的tee.bin复制到tee文件夹里

注意：文件名必须是 `tee.bin` ， tee文件夹名字不可以更改

```bash
cxx194832@docker-ubuntu18:~/sectool$ ls tee
tee.bin
```

2. 执行下面命令进行镜像国密算法签名

```bash
cxx194832@docker-ubuntu18:~/sectool$ ./imagesign.sh ia nor tee 1.2
Dump all parameters from user input.
-----------------------------------
Signed algorithem: ia
Secure attribution: nor
Image file: tee
Image version: 1.2
-----------------------------------
Enter into image sign process ...
Start tee Image (1.2) signed with international algorithms with secure att
r (nor)
sign tool path: ./tool/product
Original file: tee/tee.bin
Signed file: tee/signed_ia_nor_tee.bin
Image Version: 258
Relocate Addr: 0xff000000
[2022-03-23 14:13:31] Sign a image file.
[2022-03-23 14:13:31] Sign ok.
Exit from image sign process ...

```

3. 查看生成的TEE签名镜像

```bash
cxx194832@docker-ubuntu18:~/sectool$ ls tee
signed_sm_nor_v1.2_tee.bin  tee.bin
```

生成的 `signed_sm_nor_v1.2_tee.bin` 位于tee文件夹下。

注意：

- **使用的时候必须将 `signed_sm_nor_v1.2_tee.bin` 改名为 `tee.bin`**

2.4.2.5 生成签名的boot.EXT4

步骤如下:

1. sectool根目录下新建ree文件夹
2. 将需要签名boot.ext4复制到ree文件夹

```bash
kjs@kjs-VirtualBox:~/sectool$ ls ./ree
boot.ext4
```

3. 将imagesign.sh文件中 `ree_sign_file_list` 改成需要签名的文件名，以对Kernel镜像 `Image` 和 `light-val-a-sec.dtb` 文件签名为例（多个文件用空格隔开）

```bash
#ree sign file list
ree_sign_file_list="Image light-a-val-sec.dtb"
```

4. 执行下面命令进行镜像国际算法签名

```bash
kjs@kjs-VirtualBox:~/sectools/sectool$ ./imagesign.sh ia nor ree 1.0
Dump all parameters from user input.
-----------------------------------
Signed algorithem: ia
Secure attribution: nor
Image file: ree
Image version: 1.0
Board id:
-----------------------------------
Enter into image sign process ...
Start ree Image (1.0) signed with international algorithms with secure attr (nor)
sign tool path: ./tool/product
Original file: bootimg/Image
Signed file: ree/signed_ia_nor_v1.0_Image
Image Version: 256
Relocate Addr: 0x200000
[2023-09-21 21:36:43] Sign a image file.

[2023-09-21 21:36:43] Sign ok.
Exit from image sign process ...
```

5. 查看生成的REE签名镜像

```bash
kjs@kjs-VirtualBox:~/sectools/sectool/ree$ ls

boot.ext4  signed_ia_nor_v1.0_Image
```

生成的签名的 `boot.ext4` 位于ree文件夹下。

注意：如果已经对 `./ree/boot.ext4` 的文件签过名，需要重新上传未签名过的 `boot.ext4` 文件

2.4.3 bin2ext4

bin2ext4工具用于将一个文件打包生成EXT4文件。

注意：在默认情况下，打包的EXT4文件大小是8M，如果文件真实大小大于8M，需要调整成合适的值。

```bash
dd if=/dev/zero of=$1 bs=1M count=8
```

2.4.3.1 生成uboot.EXT4

1. 将 `u-boot-with-spl.bin` 复制到bin2ext4文件下

```bash
cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ls
bin2ext4.sh Readme u-boot-with-spl.bin
```

2. 执行下面命令生成 `uboot.ext4`

```bash
cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ./bin2ext4.sh uboot.ext4 u-boot-with-spl.bin
bin2ext4 utility version v1.0

1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00185016 s, 567 MB/s
mke2fs 1.44.1 (24-Mar-2018)

Filesystem too small for a journal
Discarding device blocks: done
Creating filesystem with 1024 1k blocks and 128 inodes

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

3. 查看生成的 `uboot.ext4` 文件

```bash
cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ls
bin2ext4.sh Readme u-boot-with-spl.bin  uboot.ext4
```

2.4.3.2 生成TF.EXT4

1. 将 `trust_firmware.bin` 复制到bin2ext4文件下

```bash
cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ls
bin2ext4.sh Readme trust_firmware.bin
```

2. 执行下面命令生成tf.ext4

```bash
cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ./bin2ext4.sh tf.ext4 trust_firmware.bin
bin2ext4 utility version v1.0

1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00112569 s, 931 MB/s
mke2fs 1.44.1 (24-Mar-2018)

Filesystem too small for a journal
Discarding device blocks: done
Creating filesystem with 1024 1k blocks and 128 inodes

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

3. 查看生成的tf.ext4文件

```bash
cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ls
bin2ext4.sh Readme trust_firmware.bin  tf.ext4
```

2.4.3.3 生成TEE.EXT4

1. 将 `tee.bin` 复制到bin2ext4文件下

```bash
cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ls
bin2ext4.sh Readme tee.bin
```

2. 执行下面命令生成tee.ext4

```bash
cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ./bin2ext4.sh tee.ext4 tee.bin
bin2ext4 utility version v1.0

1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00112569 s, 931 MB/s
mke2fs 1.44.1 (24-Mar-2018)

Filesystem too small for a journal
Discarding device blocks: done
Creating filesystem with 1024 1k blocks and 128 inodes

Allocating group tables: done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

3. 查看生成的tee.ext4文件

```bash
cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ls
bin2ext4.sh Readme tee.bin  tee.ext4
```

2.4.3.4 生成TEE.<BOARD NAME>.EXT4

TEE.<BOARD NAME>.EXT4文件包括tee.bin和trust_firmware.bin文件，用于烧录至U-Boot tee分区。**LA开发板的文件名为tee.evb_light.ext4，LB为tee.light_b.ext4，SOM板为tee.lichee_pi_4a.ext4。**

操作如下：

1. 将 `tee.bin` 和 `trust_firmware.bin` 复制到bin2ext4文件下

```bash
cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ls
bin2ext4.sh Readme tee.bin trust_firmware.bin
```

2. 执行下面命令生成tee.ext4

```bash
cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ./bin2ext4.sh tee.evb_light.ext4 tee.bin
bin2ext4 utility version v1.0

8+0 records in
8+0 records out
8388608 bytes (8.4 MB, 8.0 MiB) copied, 0.00767809 s, 1.1 GB/s
mke2fs 1.44.1 (24-Mar-2018)
Discarding device blocks: done
Creating filesystem with 8192 1k blocks and 2048 inodes

Allocating group tables: done
Writing inode tables: done
Creating journal (1024 blocks): done
Writing superblocks and filesystem accounting information: done

```

3. 查看生成的tee.ext4文件

```bash
cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ls
bin2ext4.sh Readme tee.bin  tee.evb_light.ext4  trust_firmware.bin
```

2.4.3.5 生成SBMETA.EXT4

1. 将 `sbmeta.bin` 复制到bin2ext4文件下

```bash
cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ls
bin2ext4.sh Readme sbmeta.bin
```

2. 执行下面命令生成 `tee.ext4`

```bash
cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ./bin2ext4.sh sbnmeta.ext4 sbmeta.bin
bin2ext4 utility version v1.0

8+0 records in
8+0 records out
8388608 bytes (8.4 MB, 8.0 MiB) copied, 0.00836204 s, 1.0 GB/s
mke2fs 1.44.1 (24-Mar-2018)
Discarding device blocks: done
Creating filesystem with 8192 1k blocks and 2048 inodes

Allocating group tables: done
Writing inode tables: done
Creating journal (1024 blocks): done
Writing superblocks and filesystem accounting information: done

```

3. 查看生成的 `tee.ext4` 文件

```bash
cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ls
bin2ext4.sh Readme sbmeta.bin  sbmeta.ext4
```

2.4.4 KDF_GEN

KDF_GEN工具根据输入的CV_ROOTKEY和CV_COMMONKEY来派生KDF密钥，存在kdf_derived_key.dat，同时产生测试向量存在kdf_test_vector.dat里。

1. 将CV_ROOTKEY写入cv_rootkey.dat文件
2. 将CV_COMMONKEY写入cv_commonkey.dat文件
3. 执行key_gen.elf
4. 查看kdf_derived_key.dat和kdf_test_vector.dat

其中kdf_derived_key.dat包含所有预期生成的派生密钥， kdf_test_vector.dat包含生成派生密钥的测试向量。

2.4.5 KDF_GEN

3. 用户手册

文档名	文档链接
product用户手册	📎product用户手册_v1.2.pdf
fastboot用户手册	📎USB-Fastboot用户手册_v1.0.0.pdf
efuse上位机用户手册	📎eFuse上位机用户手册_v1.0.0.pdf