玄铁安全系统安全工具用户手册

文档编号 STG30003

版权声明

修订历史

目录

- 1. 工具总览
- 2. 工具介绍
 - 2.1 product
 - 2.2 efuse utilty
 - 2.3 fastboot
 - 2.4 sectool
 - 2.4.1 签名证书
 - 2.4.2 imagesign
 - 2.4.2.1 生成SBMETA签名镜像
 - 2.4.2.2 生成UBOOT签名镜像
 - 2.4.2.3 生成TF签名镜像
 - 2.4.2.4 生成TEE签名镜像
 - 2.4.2.5 生成签名的boot.EXT4
 - 2.4.3 bin2ext4
 - 2.4.3.1 生成uboot.EXT4
 - 2.4.3.2 生成TF.EXT4
 - 2.4.3.3 生成TEE.EXT4
 - 2.4.3.4 生成TEE.<BOARD NAME>.EXT4
 - 2.4.3.5 生成SBMETA.EXT4
 - 2.4.4 KDF_GEN
 - 2.4.5 KDF_GEN
- 3. 用户手册

文档编号 STG30003

版权声明

Copyright © 2022 T-HEAD Semiconductor Co.,Ltd. All rights reserved.

This document is the property of T-HEAD Semiconductor Co.,Ltd. This document may only be distributed to: (i) a T-HEAD party having a legitimate business need for the information contained herein, or (ii) a non-T-HEAD party having a legitimate business need for the information contained herein. No license, expressed or implied, under any patent, copyright or trade secret right is granted or implied by the conveyance of this document. No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise without the prior written permission of T-HEAD Semiconductor Co.,Ltd.

Trademarks and Permissions

The T-HEAD Logo and all other trademarks indicated as such herein are trademarks of T-HEAD Semiconductor Co.,Ltd. All other products or service names are the property of their respective owners.

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.

Copyright © 2022 平头哥半导体有限公司,保留所有权利.

本文档的产权属于平头哥半导体有限公司(下称平头哥)。本文档仅能分布给:(i)拥有合法雇佣关系,并需要本文档的信息的平头哥员工,或(ii)非平头哥组织但拥有合法合作关系,并且其需要本文档的信息的合作方。对于本文档,禁止任何在专利、版权或商业秘密过程中,授予或暗示的可以使用该文档。在没有得到平头哥半导体有限公司的书面许可前,不得复制本文档的任何部分,传播、转录、储存在检索系统中或翻译成任何语言或计算机语言。

商标申明

平头哥的LOGO和其它所有商标归平头哥半导体有限公司所有,所有其它产品或服务名称归其所有者拥有。

注意

您购买的产品、服务或特性等应受平头哥商业合同和条款的约束,本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定,平头哥对本文档内容不做任何明示或 默示的声明或保证。

由于产品版本升级或其他原因,本文档内容会不定期进行更新。除非另有约定,本文档仅作为使用指导,本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

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 位库地址/附件	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_util ty	eFuse烧写工具	v1.2	仓库地址: git@gitlab.alibaba- inc.com:yocopen/efusehost.git 工具包:
	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	
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

• 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
- light_fm_single_rank_tee.bat
- light_fm_single_rank_tee_upd.bat
- light_fm_single_rank_tf.bat
- light_fm_single_rank_tf_upd.bat
- light_fm_single_rank_uboot.bat
- light_fm_single_rank_uboot_upd.bat
- light fm single rank uboot-raw.bat
- light_fm_single_rank_system.bat
- light_fm_single_rank_uboot.bat
- light_fm_single_rank_uboot_upd.bat
- light_fm_single_rank_uboot_raw.bat
- light_fm_single_rank_tee.bat
- light_fm_single_rank_tee_upd.bat
- light fm single rank tf.bat
- light_fm_single_rank_tf_upd.bat

- 用于直接烧写所有系统镜像
- 用于直接烧写uboot,不进行升级流程
- 用于更新uboot镜像, 进行升级路程
- 用于更新烧写uboot镜像(uboot损坏),不进行升级流程
 - 用于直接烧写tee,不进行升级流程
 - 用于直接更新tee镜像,进行升级流程
- 用于直接烧写tf, 不进行升级流程
- 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。

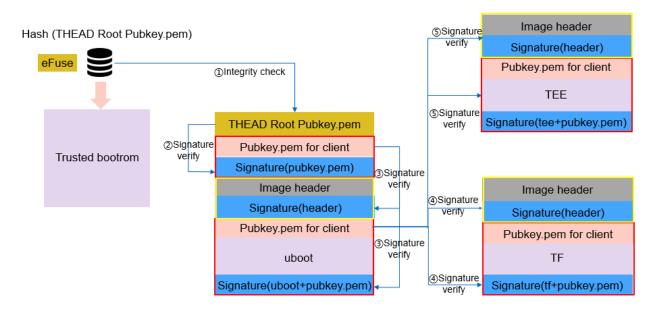
1 cxx194832@docker-ubuntu18:~/sectool\$ ls

bin2ext4 imagesign.sh keystore keystore_sm README.md tee tf tool ub oot

Bash

2.4.1 签名证书

Pubkey.pem for client表示二级签名私钥对应的公钥



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
1
     imagesign.sh utility version v1.0
    The utility is designed for aim to help user generate new image file
2
3
    with signature with desired sign scheme.
4
5
    Usage:
 6 • imagesign.sh [chkuboot] [ia/sm] [enc/nor] [tf/tee/uboot] [ver] [image boar
     d type]
     chkuboot: check uboot binary file is signed or not, if yes, it generates t
7
     he original bboot binary file
     signed algorithms
8
9
       ia - international algorithm
       sm - china algorithms
10
     secure attribution
11
12
       enc - signed image with encryption
13
       nor - signed image without encryption
14
     image file type

    trust firmware binary image

15
       tee - tee binary image
16
       uboot - uboot binary image
17
18
       ree - ree binary image
19
     version
20
       ver - image version (x.y), eg 1.1, 2.1
21
     image board type
22
       LA - board light A
       LB - board light B
23
24
       LP - board lpi4a
25
       LG - board beagle
26
       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
1
     image_configs:
2
       #config of trust_firmware.bin
 3
4
         name: trust_firmware.bin
5
         device: 0
6
         partition: 3
7
         type: tf
         digest_scheme: sha256
8
9
         sign_scheme: rsa_2048
         is_image_encrypted: 0
10
11
         medium_type: emmc
         image_path: ./tf/trust_firmware.bin #recommend to this default path
12
13
14
       #config of tee.bin
15
16
         name: tee.bin
17
         device: 0
18
         partition: 4
         type: tee
19
20
         digest scheme: sha256
         sign scheme: rsa 2048
21
22
         is_image_encrypted: 0
23
         medium_type: emmc
24
         image_path: ./tee/tee.bin #recommend to this default path
25
26
       #config of dtb
27
28
         name: light-lpi4a-sec.dtb
29
         device: 0
30
         partition: 2
31
         type: dtb
32
         digest scheme: sha256
         sign_scheme: rsa_2048
33
34
         is_image_encrypted: 0
35
         medium_type: emmc
         image_path: ./dtb/light-lpi4a-sec.dtb
36
```

b. 生成sbmeta.bin镜像

```
▼ Bash

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

2 
3 # 查看生成的镜像

4 kjs@kjs-VirtualBox:~/sectool/sbmeta$ ls sbmeta

5 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
 1
    kjs@kjs-VirtualBox:~/sectool$ ./imagesign.sh ia nor sbmeta 1.0
 2
    Dump all parameters from user input.
 3
    Signed algorithem: ia
 4
   Secure attribution: nor
 5
    Image file: sbmeta
 6
7
    Image version: 1.0
8
9
    Enter into image sign process ...
10 Start sbmeta Image (1.0) signed with international algorithms with secure
    attr (nor)
11 sign tool path: ./tool/product
12
    Original file: sbmeta/sbmeta.bin
    Signed file: sbmeta/signed_ia_nor_v1.0_sbmeta.bin
13
    Image Version: 256
14
    Relocate Addr: 0x100000
15
16 • [2023-08-16 11:17:56] Sign a image file.
17 - [2023-08-16 11:17:56] Sign ok.
    Exit from image sign process ...
18
```

3. 查看生成的SBMETA签名镜像

```
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

```
The state of the s
```

2. 执行下面命令进行镜像签名

```
Bash
 1
     cxx194832@docker-ubuntu18:~/sectool$ ./imagesign.sh ia enc uboot 1.2
 2
    Dump all parameters from user input.
 3
 4
    Signed algorithem: ia
 5
    Secure attribution: enc
6
    Image file: uboot
7
    Image version: 1.2
8
9
    Enter into image sign process ...
10
    Start uboot Image (1.2) signed with international algorithms with secure a
    ttr (enc)
11
    sign tool path: ./tool/product
12
    Original file: uboot/u-boot-with-spl.bin
    Signed file: uboot/signed ia enc u-boot-with-spl.bin
13
14
    Image Version: 258
    Relocate Addr: 0xFFE0000800
16 • [2022-03-23 14:09:44] Sign a public key file.
17 [2022-03-23 14:09:44] rsa verify ok....
18 [2022-03-23 14:09:44] rsa verify ok....
19 [2022-03-23 14:09:44] Sign ok.
20 = [2022-03-23 \ 14:09:44] Sign a image file.
21 * [2022-03-23 14:09:44] Sign ok.
22
    Exit from image sign process ...
23
```

3. 查看生成的uboot签名镜像

t
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-w ith-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.bi
- 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-b oot-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

Toxx194832@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
 1
 2
    Dump all parameters from user input.
 4
    Signed algorithem: ia
    Secure attribution: nor
 5
    Image file: tf
6
7
    Image version: 1.2
8
9
    Enter into image sign process ...
    Start tf Image (1.2) signed with international algorithms with secure att
10
     r (nor)
    sign tool path: ./tool/product
11
    Original file: tf/trust firmware.bin
12
13
    Signed file: tf/signed ia nor trust firmware.bin
14
    Image Version: 258
    Relocate Addr: 0x0
15
16 = [2022-03-23 \ 14:12:36] Sign a image file.
17 • [2022-03-23 14:12:36] Sign ok.
    Exit from image sign process ...
```

3. 查看生成的TF签名镜像

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_firmware.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

```
The 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
 1
     cxx194832@docker-ubuntu18:~/sectool$ ./imagesign.sh ia nor tee 1.2
 2
    Dump all parameters from user input.
4
    Signed algorithem: ia
 5
    Secure attribution: nor
    Image file: tee
 6
7
    Image version: 1.2
 8
9
    Enter into image sign process ...
    Start tee Image (1.2) signed with international algorithms with secure att
10
     r (nor)
11
    sign tool path: ./tool/product
    Original file: tee/tee.bin
12
13
    Signed file: tee/signed_ia_nor_tee.bin
14
    Image Version: 258
    Relocate Addr: 0xff000000
15
16 • [2022-03-23 14:13:31] Sign a image file.
17 • [2022-03-23 14:13:31] Sign ok.
18
     Exit from image sign process ...
19
```

3. 查看生成的TEE签名镜像

```
The state of the s
```

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

```
t
    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 Imag e 和 light-val-a-sec.dtb 文件签名为例(多个文件用空格隔开)

```
#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
 1
 2
    Dump all parameters from user input.
 3
 4
    Signed algorithem: ia
   Secure attribution: nor
5
 6 Image file: ree
7
    Image version: 1.0
    Board id:
8
9
    Enter into image sign process ...
10
11
    Start ree Image (1.0) signed with international algorithms with secure att
     r (nor)
    sign tool path: ./tool/product
12
    Original file: bootimg/Image
13
    Signed file: ree/signed ia nor v1.0 Image
14
    Image Version: 256
15
    Relocate Addr: 0x200000
16
17 - [2023-09-21 \ 21:36:43] Sign a image file.
18
19 • [2023-09-21 21:36:43] Sign ok.
20
     Exit from image sign process ...
```

5. 查看生成的REE签名镜像

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

2. 执行下面命令生成 uboot ext4

```
Bash
     cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ./bin2ext4.sh uboot.ext4 u-b
 1
     oot-with-spl.bin
     bin2ext4 utility version v1.0
2
3
    1+0 records in
4
 5
    1+0 records out
    1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00185016 s, 567 MB/s
7
    mke2fs 1.44.1 (24-Mar-2018)
8
    Filesystem too small for a journal
9
    Discarding device blocks: done
10
     Creating filesystem with 1024 1k blocks and 128 inodes
11
12
13
    Allocating group tables: done
14
    Writing inode tables: done
15
    Writing superblocks and filesystem accounting information: done
```

3. 查看生成的 uboot ext4 文件

```
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

2. 执行下面命令生成tf.ext4

```
Bash
     cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ./bin2ext4.sh tf.ext4 trust_
 1
     firmware.bin
2
     bin2ext4 utility version v1.0
3
    1+0 records in
4
5
    1+0 records out
    1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00112569 s, 931 MB/s
7
    mke2fs 1.44.1 (24-Mar-2018)
8
    Filesystem too small for a journal
9
    Discarding device blocks: done
10
    Creating filesystem with 1024 1k blocks and 128 inodes
11
12
13
    Allocating group tables: done
14
    Writing inode tables: done
15
    Writing superblocks and filesystem accounting information: done
```

3. 查看生成的tf.ext4文件

```
t
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

2. 执行下面命令生成tee.ext4

```
Bash
     cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ./bin2ext4.sh tee.ext4 tee.b
 1
2
     bin2ext4 utility version v1.0
 3
    1+0 records in
4
    1+0 records out
 5
    1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00112569 s, 931 MB/s
    mke2fs 1.44.1 (24-Mar-2018)
7
    Filesystem too small for a journal
9
    Discarding device blocks: done
10
     Creating filesystem with 1024 1k blocks and 128 inodes
11
12
13
    Allocating group tables: done
14
    Writing inode tables: done
    Writing superblocks and filesystem accounting information: done
15
```

3. 查看生成的tee.ext4文件

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

2. 执行下面命令生成tee.ext4

```
Bash
     cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ./bin2ext4.sh tee.evb_light.
 1
     ext4 tee.bin
2
     bin2ext4 utility version v1.0
3
   8+0 records in
4
   8+0 records out
5
    8388608 bytes (8.4 MB, 8.0 MiB) copied, 0.00767809 s, 1.1 GB/s
    mke2fs 1.44.1 (24-Mar-2018)
7
    Discarding device blocks: done
9
     Creating filesystem with 8192 1k blocks and 2048 inodes
10
    Allocating group tables: done
11
    Writing inode tables: done
12
13
    Creating journal (1024 blocks): done
14
    Writing superblocks and filesystem accounting information: done
15
```

3. 查看生成的tee.ext4文件

```
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

2. 执行下面命令生成 tee.ext4

```
Bash
     cxx194832@docker-ubuntu18:~/sectool/bin2ext4$ ./bin2ext4.sh sbnmeta.ext4 s
 1
     bmeta.bin
 2
     bin2ext4 utility version v1.0
 3
4 8+0 records in
 5
   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)
7
    Discarding device blocks: done
9
     Creating filesystem with 8192 1k blocks and 2048 inodes
10
11
    Allocating group tables: done
    Writing inode tables: done
12
13
    Creating journal (1024 blocks): done
14
    Writing superblocks and filesystem accounting information: done
15
```

3. 查看生成的 tee ext4 文件

```
Toxx194832@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