更新历史

2016-12-19	创建文档	V1.0	liwei@trustkernel.com
2016-12-28	添加一些细节	V1.1	liwei@trustkernel.com
2017-1-15	更新 bsp	V1.2	liwei@trustkernel.com
2017-10-25	增加 1708 版本	V1.3	liwei@trustkernel.com
2017-12-2	增加 SDRPMB	V1.4	liwei@trustkernel.com
2018-2-5	Android8.0 支持	V1.5	liwei@trustkernel.com

简介

本文档介绍如何使用 TrustKernel BSP，将 Android 初始工程代码修改为集成并支持 TrustKernel TEE 以及 TA 的 Android 版本。

本文档将以 Mediatek 6755 SOC，标准安卓 MT6755 EVB 代码为例，简单描述适配所需步骤。方便起见，将 MT6755 EVB 代码的根目录定义为 alps，工程名为 hct6750_66_n。BSP 所在的位置为 tkcore_bsp，具体应用时请根据具体情况调整。

1. 准备工作

在 alps/vendor/mediatek/proprietary/trustzone 目录下创建 trustkernel 目录，并且将 tkcore_bsp 整个文件包拷到 trustkernel 目录下，并且将 tkcore_bsp 重命名为 source。

```
mkdir -pv alps/vendor/mediatek/proprietary/trustzone/trustkernel/
mv tkcore_bsp alps/vendor/mediatek/proprietary/trustzone/trustkernel/source
```

以上步骤完成以后，会在 source 目录下看见如下内容：

```
Android.mk  build  client  external         README        ta
bsp         ca     driver  listmakefile.sh  README.build  tools
```

下面对各个文件目录进行简单介绍：

Bsp： 该目录下包含了各个平台的补丁和配置文件，包括需要打在系统中的 patch,需要额外引入的启动脚本，selinux 相关的配置文件，针对各个平台的 tee 镜像。

Ca： 该目录下包含需要编译进系统中库和可执行文件，gatekeeper, keymaster ,kph

Client: 该目录下包含了系统运行时 ta 和 ca 交互时用到的库和 daemon

External ： 该目录下包含 gatekeeper 和 keystore_v1，keystore_v2 源码。

Driver: 需要编译进内核的 TEE 驱动。

Ta: 需要编译进 system 镜像的 ta。

2. 集成工作

集成工作主要是包括将各个部分的 patch 打到系统的各个位置上，然后将相关文件拷贝到指定位置。以 mt6755 为例，需要打的 patch 主要分为如下几类，位于 tkcore_bsp/bsp/platform/mt6755/patches 目录下，具体文件如下：

```
lw@lw:~/lw_work/data/tkcore-bsp/bsp/platform/common/patches$ ls
custom_build.patch  device.mk.patch  file_contexts.patch  meta.patch
```

```
lw@lw:~/lw_work/data/tkcore-bsp/bsp/platform/mt6755/patches$ ls
atf.patch  init.rc.patch  kernel-3.18.patch  preloader.patch  ProjectConfig.mk.patch
```

该部分 patch 有两部分构成，一个是适用于所有的平台相关的 patch,另一部分是适用于跟平台相关的 patch.通过名字可以看出：

Atf.patch ： 是需要打到源码的 vendor/mediatek/proprietary/trustzone/atf 目录。

Custom_build.patch：是需要打到

vendor/mediatek/proprietary/trustzone/custom/build

preloader.patch: 是需要打到
vendor/mediatek/proprietary/bootable/bootloader/preloader
kernel-3.18.patch: 是需要打到 kernel-3.18 目录下。
ProjectConfig.mk.patch: 是需要打到 device/厂商名/工程名/ProjectConfig.mk 文件。
File_contexts.patch: 是需要打到 device/mediatek/common/sepolicy/file_contexts 文件。
注：如果是 Android7.0,请将该 patch 打到
device/mediatek/common/sepolicy/bsp/file_contexts 文件。
Init.rc.patch: 将该 patch 打到 device/mediatek/mt6755/目录下的 init.mt6755.rc, factory_init.rc, meta_init.rc 文件上。

2.1. atf 补丁

atf 在 TEE 系统中起到一个 monitor 的作用，在 TEE 和 REE 之间切换。

```
cd alps/vendor/mediatek/proprietary/trustzone/atf/v1.0
patch -p1 < tkcore_bsp/bsp/platform/mt6755/patches/atf.patch
```

注：在应用 ATF 补丁的时候，需要根据工程实际用的 ATF 版本来打，一般默认的版本都为 v1.0,但是像 mtk6795 用的就是 v0.4 版本。

2.2. preloader 补丁

该部分的 Patch 主要使其支持 TKCore 的引导和启动，该部分涉及到 RPMB 密钥的生成。

```
cd alps/vendor/mediatek/proprietary/bootable/preloader
patch -p1 < tkcore_bsp/bsp/platform/mt6755/patches/preloader.patch
```

注：Proloader 控制着系统的启动，该部分涉及到了 TEE 的启动，所以在配置 Proloader 时请将系统 TEE 支持打开。配置如下，请确保 CFG_TEE_SUPPORT = 1,其他厂商的 TEE_SUPPORT 等于 0。该配置文件位于 preloader/custom/hct6750_66_n(工程名)/ cust_bldr.mak 如下：

```
CFG_UART_LOG :=UART1
CFG_UART_META :=UART1
CFG_TEE_SUPPORT = 1
CFG_TRUSTONIC_TEE_SUPPORT = 0
CFG_MICROTRUST_TEE_SUPPORT = 0
CFG_GOOGLE_TRUSTY_SUPPORT=0
CFG_TRUSTKERNEL_TEE_SUPPORT=1
```

2.3. custom_build 补丁

该 Patch 用于系统编译 trustzone,

```
cd alps/vendor/mediatek/proprietary/trustzone/custom/build
patch -p1
    < tkcore_bsp/bsp/platform/mt6755/patches/custom_build.patch
```

由于在 custom/build 目录下控制着是否编译关于 trustzone 相关的内容，一些厂商在自己调试时，会将相关的 TEE 开关关掉，所以在打该 Patch 时，请注意是否打开了 TEE 和 ATF 支持。Custom/build/hct6750_66_n.m(工程名)如下：

```
1 MTK_ATF_SUPPORT = yes
2 MTK_TEE_SUPPORT = yes
3 TRUSTONIC_TEE_SUPPORT = no
4 MICROTRUST_TEE_SUPPORT = no
5 MTK_GOOGLE_TRUSTY_SUPPORT=no
6 TRUSTKERNEL_TEE_SUPPORT = yes
7 MTK_TEE_DRAM_SIZE=0x4000000
```

确保打过 patch 后，MTK_ATF_SUPPORT 和 MTK_TEE_SUPPORT 是处于 yes 状态。除 TRUSTKERNEL_TEE_SUPPORT 宏处于 yes 状态以外，其他厂商的 TEE 支持都处于 no 状态。

2.4. kernel 补丁

内核 patch 主要跟编译 tee driver 有关，另外就是配置系统的 SPI，mmc。

```
cd alps/kernel-3.18
patch -p1 < tkcore_bsp/bsp/platform/mt6755/patches/kernel-3.18.patch
```

在打 kernel patch 时，还需要注意，内核的配置文件里的相关配置，该文件在 arch/arm64/configs/(平台配置文件)下，具体的配置请咨询 OEM/IDM 工程师，在该文件中，必须确保拥有一样内容：

```
38
39 # CFG_GOOGLE_TRUSTY_SUPPORT is not set
40 # CONFIG_TRUSTONIC_TEE_SUPPORT is not set
41 # CONFIG_MICROTRUST_TEE_SUPPORT is not set
42 CONFIG_TRUSTKERNEL_TEE_SUPPORT=y
43 CONFIG_TRUSTKERNEL_TEE_RPMB_SUPPORT=y
44 CONFIG_TRUSTKERNEL_TEE_FP_SUPPORT=y
45
```

2.5. ProjectConfig 补丁

该补丁用于开启 Android 工程 TEE 支持，该文件在 device/ haocheng/hct6750_66_n/（假设厂商名为 haocheng，工程名为 hct6750_66_n）。

在该工程底下，确保关于 TEE 的配置如上，并且还需要将其他 TEE 厂商的 tee_support 宏关闭。具体如下：

```
723 ########## TrustKernel TEE ON----------------
724 MTK_ATF_SUPPORT=yes
725 MTK_TEE_SUPPORT=yes
726 TRUSTKERNEL_TEE_SUPPORT = yes
727 TRUSTONIC_TEE_SUPPORT=no
728 MTK_GOOGLE_TRUSTY_SUPPORT=no
729 MICROTRUST_TEE_SUPPORT=no
```

注：有的厂商对工程进行定制化，该配置文件不一定在该目录下，具体情况请咨询相关的工程师。

2.6. Init 初始化脚本补丁

该脚本根系统的启动有关，表明系统在启动时，启动哪些系统服务。该补丁用于启动 init.Trustkernl.rc.所以只需要在 init.mt6750.rc,factory_init.rc,meta_init.rc 文件中导入 init.trustkernel.rc 即可，系统会在导入时自动启动。所以只要在上面三个文件的 import 语句最后添加如下高亮语句即可，这些文件都为于“device/mediatek/mt6755(平台名 mt6755)/”。

```
15 import init.common_svc.rc
16 import init.microtrust.rc
17 import init.chiptest.rc
18 import init.trustkernel.rc
```

注：如果系统高于 Android8.0，则 rc 文件的 import 需要按照如下的写法：

```
 9 import ${ro.mtkrc.path}meta_init.connectivity.rc
10 import /vendor/etc/init/hw/meta_init.project.rc
11 import /vendor/etc/init/microtrust.rc
12 import /vendor/etc/init/trustkernel.rc
13 import /odm/etc/init/init.odm.rc
14 import /odm/etc/init/init.odm_project.rc
```

注:为了对产线支持,需要将 meta_tst 服务改为 late_start.具体修改位置为 meta_init.rc

```
486 #INTERNAL_START
487 service meta_tst /vendor/bin/meta_tst
488     class late_start
```

另外，由于系统需要加密手机，所以指纹服务同样需要修改为 late_start。
通常位于“device/mediatek/mt6755(平台名 mt6755)/init.mt6750.rc”文件中。或者位于 system 文件夹下。

```
1602 service fingerprintd /system/bin/fingerprintd
1603     class late_start
1604     user system
1605     group system
1606
1607 #sunwave
1608 service fpserver /system/bin/fpserver
1609     class late_start
1610     user root
1611     disabled
1612 on property:sys.fingerprint.chip=sunwave
1613     start fpserver
1614 #fingerprint
```

2.7. Device.mk 补丁

该补丁很简单，用于在系统编译时，将指定位置的文件拷贝的指定位置，而拷贝的脚本即为 device_trustkernel.mk 文件。

所以只需要在 device/mediatek/mt6755/device.mk 文件中将补丁中的内容加上即可。加的内容如下：

```
5 ifeq ($(strip $(TRUSTKERNEL_TEE_SUPPORT)), yes)
6 $(call inherit-product, vendor/mediatek/proprietary/trustzone/trustkernel/source/bsp/platform/mt6755/scripts/device_trustkernel.mk)
7 endif
8
9 PRODUCT_COPY_FILES += frameworks/native/data/etc/android.hardware.location.gps.xml:system/etc/permissions/android.hardware.location.gps.xml
```

2.8. File_contexts 补丁

该补丁定义 TKCORE 相关的 sepolicy 相关定义。而 file_contexts 文件专门用于定义 sepolicy 相关变量的。

```
15 ##############################
16 # TrustKernel ADD
17 ##############################
18 /data/tee(/.*)? u:object_r:tkcore_data_file:s0
19 /data/teec.log u:object_r:tkcore_log_file:s0
20 /dev/tkcoredrv u:object_r:tkcore_admin_device:s0
21 /system/bin/teed u:object_r:tkcore_exec:s0
22 /data/ifaa-test.log u:object_r:ifaa_log_file:s0
23 /system/lib/t6(/.*)? u:object_r:tkcore_systa_dir:s0
24 /data/tee/t6(/.*)? u:object_r:tkcore_spta_dir:s0
25
```

注：该文件在不同的 Android 版本上所在的路径不一样

如果是 Android6.0，该文件在 device/mediatek/common/sepolicy/下

如果是 Android7.x，该文件在 device/mediatek/common/sepolicy/bsp/下

如果是 Android8.x，该文件在 device/mediatek/sepolicy/bsp/non_plat 下

3. 拷贝相关文件到制定位置

拷贝相关文件到指定位置，主要包括两方面：

(1)将 sepolicy 相关文件拷贝到指定位置

(2)将 TEE 驱动拷贝到指定位置

3.1. 拷贝 sepolicy 相关文件

如果是 Android6.0，请将

tkcore_bsp/bsp/platform/mt6755/scripts/sepolicy/Android6.0/tkcore_daemon.te

拷贝到 alps/device/mediatek/common/sepolicy/

如果是 Android7.0，请将

tkcore_bsp/bsp/platform/mt6755/scripts/sepolicy/Android7.0/tkcore_daemon.te

拷贝到 Device/mediate/commom/sepolicy/bsp/

并且将 untrusted_app.te.patch 应用到 7.0 版本的 untrusted_app.te 中。

3.2. 拷贝驱动到内核

拷贝驱动到内核执行如下命令：

```
cp -a tkcore_bsp/driver/tkcore alps/kernel-3.18/drivers/misc/mediatek/
```

4. 替换项目证书

在 1708 版本以后，各个项目都是使用了证书来管理各个项目。证书存放的目录位于“Tkcore-bsp/build/项目名/cert.dat“，通常将 sample 文件夹重命名为项目名。

另外，每一个项目都会拥有不同的证书，这是因为每个项目的 brand,model,platform 都不一定一致，所以在项目集成时，请向开发人员获取证书。具体需要提供的信息在 ”out/target/product/ 项 目 名 /system/build.prop” 文 件 中 。 请 提 供 ” ro.product.model”,”ro.product.brand”,”ro.board.platform”这三个字段。

5. SDRPMB 集成

目前该方案由于商务原因没有大规模使用在 BSP 1708 版本以后，我们实现了 SDRPMB 方案。该方案用于替代手机 RPMB，可以在该区域存储产线数据。这涉及到 preloader，lk, tee.bin 以及 teed 的支持才行。目前在 bsp 版本中默认的补丁已经支持，但是，需要额外释放正式版本的 teed 和 tee.bin。

1).使用 Lk 补丁和 preoloader 补丁

将 lk.patch 拷贝到 vendor/mediatek/proprietary/bootable/bootloader/lk/目录下。执行如下命令：

```
Patch  -p1 < lk.patch
```

```
 7 else
 8 CFG_BOOT_DEV :=BOOTDEV_NAND
 9 endif
10 CFG_UART_LOG :=UART1
11 CFG_UART_META :=UART1
12 CFG_TEE_SUPPORT = 1
13 CFG_TRUSTONIC_TEE_SUPPORT = 0
14 CFG_MICROTRUST_TEE_SUPPORT = 0
15 CFG_GOOGLE_TRUSTY_SUPPORT=0
16 CFG_TRUSTKERNEL_TEE_SUPPORT=1
17 CFG_TRUSTKERNEL_TEE_SDRPMB_SUPPORT=1
```

当把 preloader 相关的 patch 合入以后，需要确保工程的 custom/工程名/cust_bldr.mak 文件中 SDRPMB 宏处于开的状态。

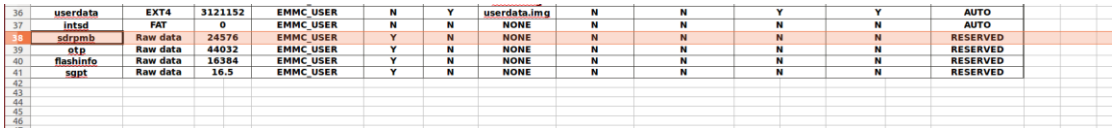
另外，确保 lk/lk/project/工程名.mk 文件中 TRUSTKERNEL 相关的宏处于打开状态，如下：

```
37 DEFINES += WITH_DEBUG_UART=1
38 #DEFINES += WITH_DEBUG_FBCON=1
39 #DEFINES += MACH_FPGA=y
40 #DEFINES += SB_LK_BRINGUP=y
41 DEFINES += MTK_GPT_SCHEME_SUPPORT
42 MTK_GOOGLE_TRUSTY_SUPPORT=no
43 MTK_TRUSTKERNEL_TEE_SUPPORT=yes
44
```

2). 修改分区表

SDRPMB 需要在 emmc 上额外留出一块分区作为 SDRPMB。分区表 excel 文件所在路径为”device/mediatek/build/build/tools/ptgen/平台名”

打开 partition_table_平台名.xls 文件，添加如下内容：



36	userdata	EXT4	3121152	EMMC_USER	N	Y	userdata.img	N	N	Y	Y	AUTO
37	intsd	FAT	0	EMMC_USER	N	N	NONE	N	N	N	N	AUTO
38	sdrpmb	Raw data	24576	EMMC_USER	Y	N	NONE	N	N	N	N	RESERVED
39	otp	Raw data	44032	EMMC_USER	Y	N	NONE	N	N	N	N	RESERVED
40	flashinfo	Raw data	16384	EMMC_USER	Y	N	NONE	N	N	N	N	RESERVED
41	sgpt	Raw data	16.5	EMMC_USER	Y	N	NONE	N	N	N	N	RESERVED
42												
43												
44												
45												
46												

注意，一定要在如图中的对应位置加入对应的内容。

3). 修改 uevent.rc 文件

该文件在 device/mediatek/平台名/ueventd.平台名.rc 文件中

```
115
116 # Trustkernel TEE
117 /dev/tkcoredrv            0666  root      root
118 /dev/block/platform/mtk-msdc.0/11230000.msdc0/by-name/sdrpmb 0660    root    system
```

其中 mt6763 平台 sdrpmb 所在路径有所不同，如下：

```
#Trustkernel add
/dev/tkcoredrv            0666    root       root
/dev/block/platform/bootdevice/by-name/sdrpmb 0666 root system
```

4).selinux

在 file_contexts 文件中，对 sdrpmb 进行声明的时，需要注意标注的内容。必须和手机里的路径保持一致，当然如下图的为通配符，可以匹配任意的数字。

```
15 /data/tee(/.*)? u:object_r:tkcore_data_file:s0
16 /dev/tkcoredrv u:object_r:tkcore_admin_device:s0
17 /system/bin/teed u:object_r:tkcore_exec:s0
18 /system/app/t6(/.*)? u:object_r:tkcore_systa_dir:s0
19 /data/tee/t6(/.*)? u:object_r:tkcore_spta_dir:s0
20 /data/teec.log u:object_r:tkcore_log_file:s0
21 /data/tee/tkcore.log u:object_r:tkcore_log_file:s0
22 /protect_f/tee(/.*)? u:object_r:tkcore_data_file:s0
23 /dev/block/platform/mtk-msdc\.0/[0-9]+\.(msdc|MSDC)0/by-name/sdrpmb u:object_r:tkcore_block_device:s0
24
```

如下图为手机里的路径例子，仅做参考：

```
sr6580_we_n:/dev/block/platform/mtk-msdc.0/11120000.msdc0/by-name # ls sdrpmb
sdrpmb
sr6580_we_n:/dev/block/platform/mtk-msdc.0/11120000.msdc0/by-name #
```

5).启动脚本 rc 文件

本文件用于启动系统服务 teed,在 teed 中会对 SDRPMB 进行初始化,所以需要告诉 teed 服务 sdrpmb 的正确路径时多少，其中具体的路径为手机里的路径，如上图所示。

其中在 trustkernel.rc 文件中,应该将 sdrpmb 具体的路径填写正确,可以参考如下例子：

```
37
38 service teed /system/bin/teed /dev/block/platform/mtk-msdc.0/11120000.msdc0/by-name/sdrpmb
39     user system
40     group system inet
41     class core
42     seclabel u:r:tkcore:s0
```

6. 指纹集成

指纹集成通常有指纹厂商的工程师进行。在此我们不需要管。

如果指纹厂商需要给了指纹源码（hal 库代码），我们集成需要将这些源代码放在如下位置。通常都是给编译好的库代码，所以以下内容酌情处理。

配置 GPIO 模式。如果对应 Pin 脚的默认模式与指纹芯片的要求不符，需要修改 codegen.dws 文件以配置 GPIO 的默认模式。在本例中，存在 4 处 codegen.dws 文件需要修改，分别位于

1. alps/kernel-3.18/drivers/misc/mediatek/mach/mt6755/evb6755_64_teei/dct/dct/codegen.dws
2. vendor/mediatek/proprietary/bootable/bootloader/lk/target/evb6755_64_teei/dct/dct/codegen.dws
3. vendor/mediatek/proprietary/custom/evb6755_64_teei/kernel/dct/dct/codegen.dws
4. endor/mediatek/proprietary/bootable/bootloader/preloader/custom/evb6755_64_teei/dct/dct/codegen.dws

在本例中，需要修改 codegen.dws 文件,将 GPIO95、96、97、98 配置为 mode 1(分别为 SPI_MO, SPI_CSB, SPI_MI, SPI_CLK，如图所示 TODO)。需要注意的是，如果 Android 编译完成之后再次修改了上述 codegen.dws 文件，为使改动生效,请重新编译 lk, preloader 和 bootimage。

7. 编译

通常进行编译只需要按照 Android 标准的编译命令编译即可，但是有的厂商会进行定制化编译。需要咨询相应的工程师。如本例中，执行如下命令编译。

```
source build/envsetup.sh
lunch full_hct6750_66_n-eng
make -j8 2>&1 | tee build.log
```

8. 测试集成

第一次在厂商那集成，我们需要确定以下问题：

1. 确定能够正常开机？
2. 设备的 RPMB 是否能够工作？

   我们利用 tkcore_apps 下的 teetest TA 来验证，所以在集成完毕以后，正常开机，请将 teetest ta 安装到手机。

```
./install_ta.sh  tee-test  #pc
./install_ca.sh  teetest  #pc
//以下命令在手机 shell
teetest  -gtest_filter=”*RPMB*”
```

3. 能否正常读取指纹模组的 chipid？

   部署完毕后，还需要确定设备的 SPI 是否能够正常 work。此时需要写一个读 chipid 的 TA.请参考 TK_APPS/fingerprint_ta 下关于读取 chipid 的 TA,此时你需要与指纹厂商的工程师确认 TEE_SPIConfig 和读取寄存器命令。然后在代码中写入。如下：

```
static TEE_SPIConfig chip_config = {
    .setuptime = 10,
    .holdtime = 10,

    .high_time = 8,
    .low_time = 8,

    .sample_sel = TEE_SPI_POSEDGE,
    .cs_pol = TEE_SPI_ACTIVE_LOW,

    .cs_idletime = 1,
    .ulthgh_thrsh = 0,

    .cpol = 0,
    .cpha = 0,

    .rx_mlsb = 1,
    .tx_mlsb = 1,

    .tx_endian = 0,
    .rx_endian = 0,

    .pause = 0,
    .finish_intr = 1,
    .deassert = 0,
    .ulthigh = 0,
    .tckdly = 0,

    //.spi_disable_auto_select_mode = 1
};
```

这些请指纹工程师协助，编译好以后，push 进手机进行测试，如果能够正常读取 id，即打印出来的 id 值不为 0，即读取成功。

4. SDRPMB 测试

SDRPMB 是在 1708 以后版本才会出现的，所以如果版本早于 1708 版本，请跳过该项测试。执行命令即可查看 sdrpmb 是否 Ok. “pld -device --st”命令即可查看。如果输出的 TRUSTSTORE 类型为 SDRPMB,并且状态为 OK,可以判定为可以 SDRPMB 可以使用，例子如下图所示。

```
sr6580_we_n:/dev/block/platform/mtk-msdc.0/11120000.msdc0/by-name # pld --device --st
VERSION: 00000002 DEVICE BRAND:alps MODEL: sr6580_we_n PLATFORM: mt6580 STATUS: [ VERIFIED EVALUABLE ] RND: 3425348511
VERSION: 00000001 TRUSTSTORE: SDRPMB(1) status: OK
sr6580_we_n:/dev/block/platform/mtk-msdc.0/11120000.msdc0/by-name #
```

9. 常见问题

1. 有的厂商会有自己的工程配置，比如 ProjectConfig.mk 和内核的 config 文件配置 不一定在源目录下，如果该原目录下的文件有可能被覆盖和修改，所以在修改这两个文件时，请咨询对接工程师，然后在他们指定位置修改。

2. 在测试 RPMB 时，有可能是失败的，如果测试执行 RPMB 测试时，返回值为 0xffff000f,此时有可能是由于该设备在以前用过其他厂商的 TEE,并且往 RPMB 中写入过 Key。造成此现象的原因有可能是我们生成 rpmb key 的算法和原来的不一样。请调整 preloader 中生成 rpmbkey 的算法。具体做法如下：

```
#if CFG_TRUSTKERNEL_TEE_SUPPORT
    u8 rpmb_key[32];
    kdflib_get_msg_auth_key(teearg->hwuid, 16, rpmb_key, 32);
    tkcore_key_param_prepare(TEE_BOOT_ARG_ADDR, rpmb_key, 32);
#else
    u8 rpmb_key[32];
    seclib_get_msg_auth_key(teearg->hwuid, 16, rpmb_key, 32);
#endif
```

请在”CFG_TRUSTKERNEL_TEE_SUPPORT”宏中，在函数中 kdflib_get_msg_auth_key 和 seclib_get_msg_auth_key 函数中进行选择切换。

3.在测试 SPI 是否 OK 时，有可能读出来的数据完全为 0，此时造成的原因有两个。

1.有可能 TEE_SPIConfig 配置不对。需要与指纹工程师确认。

2.有可能在 TEE_SPIReadWrite()函数中，发送的命令部队，需要与指纹工程师确认。

3.有可能是指纹芯片没有上电，如果有条件请测量一下 gpio 口电压是否被拉高。如果没有被拉高，请在指纹驱动里设置。

4.成完毕后不能够启动，通常由两个原因造成的

1. spi 驱动没有正确配置好，请检查是否根补丁包中 spi 部分一致。

2. 可能有其他的设备在用 SPI，而此时的 SPI 被配置成安全状态，导致使用者无法使用，此时会导致内核 PANIC.

3. 其他原因，需要根据具体 log 来判断。