|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

**更新历史**

**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：



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。

1. **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：





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。

* 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 |

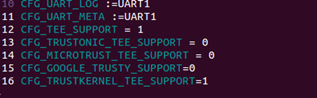
**注：在应用ATF补丁的时候，需要根据工程实际用的ATF版本来打，一般默认的版本都为v1.0,但是像mtk6795用的就是v0.4版本**。

* 1. **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：

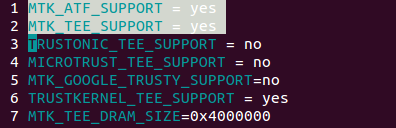


* 1. **custom\_build 补丁**

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：



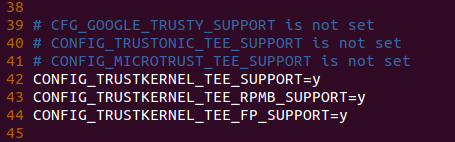
确保打过patch后，MTK\_ATF\_SUPPORT 和MTK\_TEE\_SUPPORT是处于yes状态。除TRUSTKERNEL\_TEE\_SUPPORT宏处于yes状态以外，其他厂商的TEE支持都处于no状态。

* 1. **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：

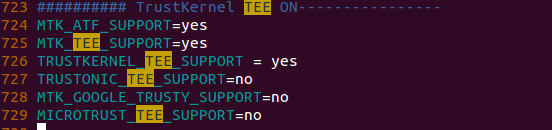


* 1. **ProjectConfig补丁**

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：

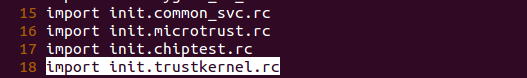


注：有的厂商对工程进行定制化，该配置文件不一定在该目录下，具体情况请咨询相关的工程师。

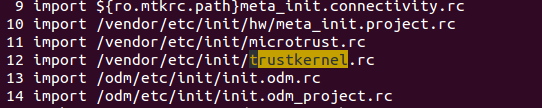
* 1. **Init初始化脚本补丁**

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)/”。



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



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



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。

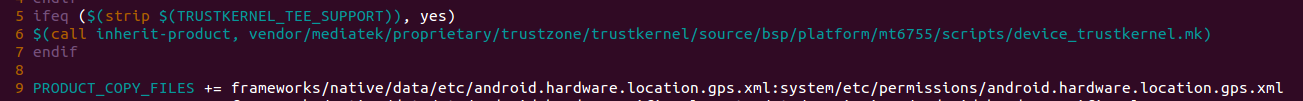


* 1. **Device.mk补丁**

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。

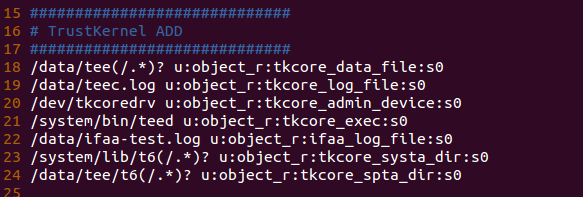
所以只需要在device/mediatek/mt6755/device.mk文件中将补丁中的内容加上即可。加的内容如下：



* 1. **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。



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

1. **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/ |

1. **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。

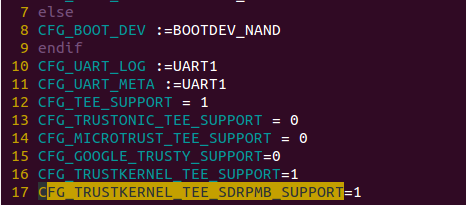
1. **SDRPMB集成**

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

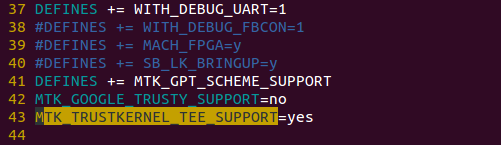
将lk.patch拷贝到vendor/mediatek/proprietary/bootable/bootloader/lk/目录下。执行如下命令：

|  |
| --- |
| Patch –p1 < lk.patch |



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。

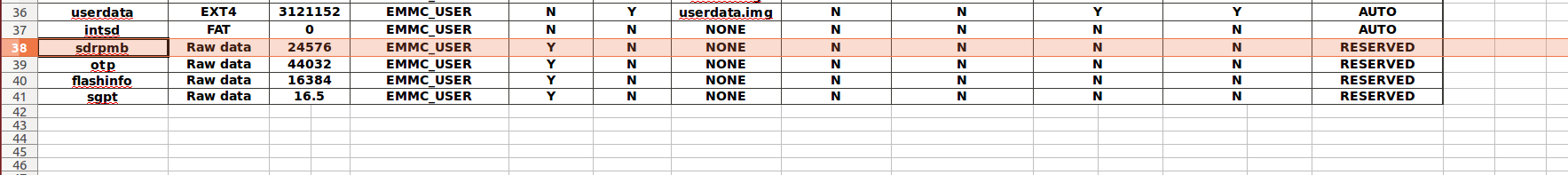
另外，确保lk/lk/project/工程名.mk文件中TRUSTKERNEL相关的宏处于打开状态，如下：



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”

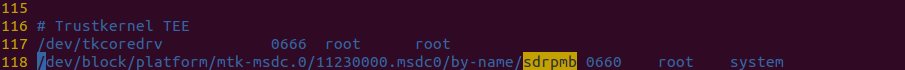
打开partition\_table\_平台名.xls文件，添加如下内容：



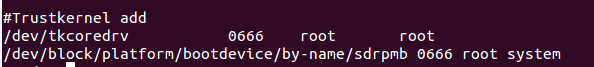
注意，一定要在如图中的对应位置加入对应的内容。

3). 修改uevent.rc文件

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

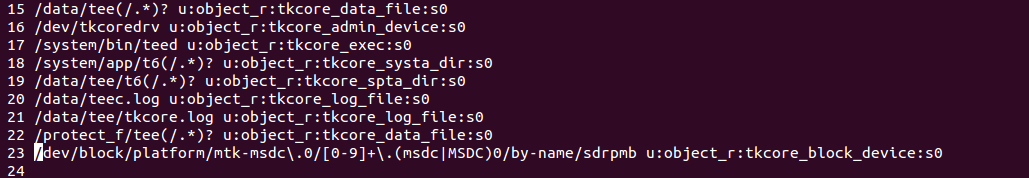


其中mt6763平台sdrpmb所在路径有所不同，如下：



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。



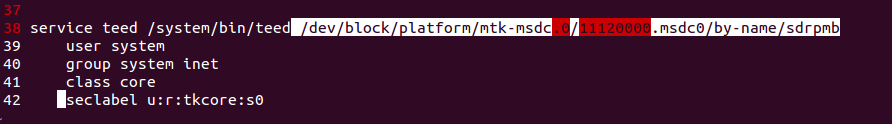
如下图为手机里的路径例子，仅做参考：



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：



1. **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。

1. **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 |

1. **测试集成**

第一次在厂商那集成，我们需要确定以下问题：

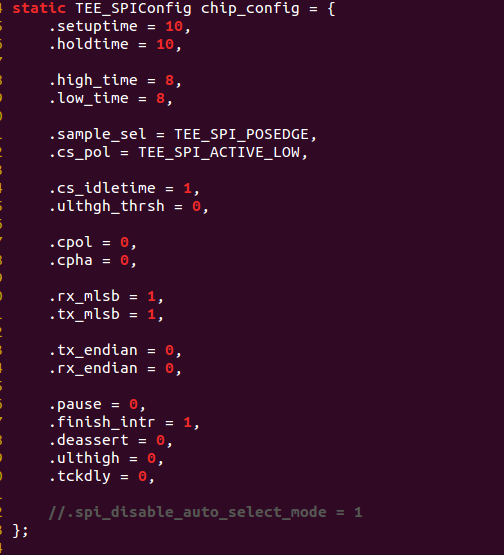
1. 确定能够正常开机？
2. 设备的RPMB是否能够工作？

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\*” |

1. 能否正常读取指纹模组的chipid？

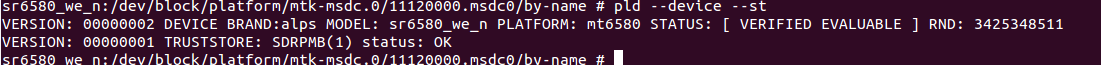
部署完毕后，还需要确定设备的SPI是否能够正常work。此时需要写一个读chipid的TA.请参考TK\_APPS/fingerprint\_ta下关于读取chipid的TA,此时你需要与指纹厂商的工程师确认TEE\_SPIConfig和读取寄存器命令。然后在代码中写入。如下：



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。

1. SDRPMB测试

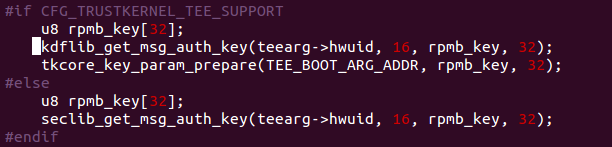
SDRPMB是在1708以后版本才会出现的，所以如果版本早于1708版本，请跳过该项测试。执行命令即可查看sdrpmb是否Ok.“pld –device --st”命令即可查看。如果输出的TRUSTSTORE类型为SDRPMB,并且状态为OK,可以判定为可以SDRPMB可以使用，例子如下图所示。



1. **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：



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。