QSEE 4.0.5 TA 移植说明

Prepared by		Date	
拟制	_____	日期	_____
Reviewed by		Date	
审阅	_____	日期	_____
Approved by		Date	
批准	_____	日期	_____

Revision Record 修订记录

Date 日期	Reversion Version 修订版本	Sec No. 修改章节	Change Description 修改描述	Author 作者
2016.12.29	V1.0.0		Initial version	
2017.02.10	V1.0.1		Add TA verify section	

目录

Overview

Prerequisites

在进行 fingerprint TA 移植前，请确认：

1) 本文档适用于 QSEE TZ.4.0.5 版本，如果使用的 QSEE 版本不是 TZ.4.0.5，请参考 QSEE_TZ.4.0.x_porting_guide 。 QSEE 的版本信息位于： trustzone_images/build/manifest.xml 文件中，如下：

```
<config>
  <image_tree>
    <name>TZ.BF.4.0.5</name>
    <build_id>TZ.BF.4.0.5-00030-M8937AAAAANAZT-2</build_id>
    <branch_point_revision/>
    <revision>00030</revision>
    <spin>2</spin>
    <client>M8937AAAAANAZT</client>
  </image_tree>
</config>
```

2) QSEE SDK 已准备好，且编译、开发 QSEE TA 所需要的 toolchain 如 RVDS/LLVM/PYTHON/MAKE/GCC 等已经安装配置好。该步骤可通过编译 sampelapp TA 来进行确认，命令如下（CHIPSET 请根据具体情况设置）：

```
cd TZ.4.0.5/trustzone_images/build/ms/
./build.sh CHIPSET=msm8937 sampleapp
```

3) 指纹模组与高通硬件平台连接无误，指纹的 TEE Linux driver 及 selinux sepolicy 已经移植完毕，且指纹模组使用的 SPI BUS 已经配置到了 trustzone 侧（对于高通 QSEE 平台，指纹相关模块的移植最好遵循 Linux driver/sepolicy -> QSEE TA -> CA HAL 的顺序）。

1 Edit secimage.xml

1.1 trustzone_images/apps/bsp/trustzone/qsapps/build

cd trustzone_images/apps/bsp/trustzone/qsapps/build, 编辑 secimage.xml 文件，增加以下内容:

```
<image sign_id="fngap32" name="fngap32.mbn" image_type="elf_has_ht">
    <general_properties_overrides>
        <sw_id>0x000000000000000C</sw_id>
        <app_id>0x0000000000112345</app_id>
    </general_properties_overrides>
```

```
    </image>
    <image sign_id="fngap64" name="fngap64.mbn" image_type="elf_has_ht">
        <general_properties_overrides>
            <sw_id>0x000000000000000C</sw_id>
            <app_id>0x0000000000112345</app_id>
        </general_properties_overrides>
    </image>
```

注：app_id 请客户根据情况自己定义，如客户未要求，可以自己定义，但不得与其它 TA 的 app_id 相同。fngap32 和 fngap64 分别是指纹 TA 的 32bit 和 64bit 版本名字，在默认情况下，只编译 64bit 版本，即 fngap64。

1.2 trustzone_images/core/bsp/trustzone/qsapps/build

与 1.1 相同，cd trustzone_images/core/bsp/trustzone/qsapps/build，编辑 secimage.xml 文件，增加以下内容（app_id 请客户根据情况自己定义，但必须与 1.1 中的相同且不可与其它 TA 的 app_id 重复）：

```
    <image sign_id="fngap32" name="fngap32.mbn" image_type="elf_has_ht">
        <general_properties_overrides>
            <sw_id>0x000000000000000C</sw_id>
            <app_id>0x0000000000112345</app_id>
        </general_properties_overrides>
    </image>
    <image sign_id="fngap64" name="fngap64.mbn" image_type="elf_has_ht">
        <general_properties_overrides>
            <sw_id>0x000000000000000C</sw_id>
            <app_id>0x0000000000112345</app_id>
        </general_properties_overrides>
    </image>
```

2 Edit TZ SConscript and copy project files

2.1 trustzone_images/core/kernel/libstd/build

a). copy project files

cd trustzone_images/core/kernel/libstd/build

cp –arf sampleapp fingerapp

b). edit SConscript, add FINGERAPP_IMAGE and FINGERAPP64_IMAGE:

```
#-------------------------------------------------------------------------------
# Add Libraries to image
#-------------------------------------------------------------------------------

env.AddBinaryLibrary(['TZOS_IMAGE','MONITOR_IMAGE','HYPERVISOR_IMAGE', 'CTZL_IMAGE', 'CTZL64_IMAGE'
                      'GPSAMPLE', 'GPTEST_IMAGE', 'GPTEST2', 'TTAARI1', 'TTACAPI1', 'TTACAPI2', 'TT
                      'FINGERAPP_IMAGE', 'FINGERAPP64_IMAGE', 'SAMPLEAPP_IMAGE', 'QMPSECAPP_IMAGE',
                      'HDCPSRM_IMAGE', 'LKSECAPP_IMAGE', 'TZBSPTEST_IMAGE', 'APTTESTAPP64_IMAGE', '
                      'CRIKEYMGMTAPP_IMAGE', 'CRIKEYMGMTAPP64_IMAGE', 'APTLKSECAPP_IMAGE', 'APTLKSE
                     '${BUILDPATH}/libstd',
                     LIBSTD_SOURCES)
if 'LIBSTD_TEST' in env:
    env.AddBinaryLibrary(['TZOS_IMAGE', 'HYPERVISOR_IMAGE', 'FINGERAPP_IMAGE', 'FINGERAPP64_IMAGE',
                          'TZBSPTEST_IMAGE', 'QPAY_IMAGE', 'QPAY64_IMAGE'],
                          '${BUILDPATH}/libstd_test',
                          LIBSTD_TEST_SOURCES)
```

2.2 trustzone_images/core/securemsm/trustzone/qsee/mink/libstd/build

a). copy project files

cd trustzone_images/core/securemsm/trustzone/qsee/mink/libstd/build

cp –arf sampleapp fingerapp

b). edit SConscript, **ONLY add FINGERAPP_IMAGE:**

```
#-------------------------------------------------------------------------------
# Add Libraries to image
#-------------------------------------------------------------------------------

images = ['TZOS_IMAGE', 'MONITOR_IMAGE', 'HYPERVISOR_IMAGE', 'FINGERAPP_IMAGE',
          'CTZL_IMAGE', 'CTZL64_IMAGE', 'TZTESTEXEC_IMAGE',
          'WIDEVINE_IMAGE', 'PLAYREADY_IMAGE', 'MACCHIATO_SAMPLE_IMAGE',
          'GPSAMPLE', 'GPTEST_IMAGE', 'TTAARI1', 'TTACAPI1', 'TTACAPI2', 'TTACAPI3
          'TTACAPI4',  'TTACAPI5',  'TTACRP1',  'TTADS1',  'TTATIME1',
```

2.3 trustzone_images/core/securemsm/trustzone/qsapps/libs/biometric/build

a). copy project files

cd trustzone_images/core/securemsm/trustzone/qsapps/libs/biometric/build

cp –arf sampleapp fingerapp

b). edit SConscript, add FINGERAPP_IMAGE and FINGERAPP64_IMAGE:

```
#-------------------------------------------------------------------------------
# Add Libraries to image
#-------------------------------------------------------------------------------
env.AddBinaryLibrary (['FIDOCRYPTO_IMAGE'],biometric_lib, [BIOMETRIC_LIB_SOURCES])
env.AddBinaryLibrary (['SAMPLEAPP_IMAGE'],biometric_lib, [BIOMETRIC_LIB_SOURCES])
env.AddBinaryLibrary (['SAMPLEAPP64_IMAGE'],biometric_lib, [BIOMETRIC_LIB_SOURCES])
env.AddBinaryLibrary (['FINGERPRINT_IMAGE'],biometric_lib, [BIOMETRIC_LIB_SOURCES])
env.AddBinaryLibrary (['FINGERPRINT64_IMAGE'],biometric_lib, [BIOMETRIC_LIB_SOURCES])
env.AddBinaryLibrary (['FINGERAPP_IMAGE'],biometric_lib, [BIOMETRIC_LIB_SOURCES])
env.AddBinaryLibrary (['FINGERAPP64_IMAGE'],biometric_lib, [BIOMETRIC_LIB_SOURCES])
env.AddBinaryLibrary (['IRIS_IMAGE'],biometric_lib, [BIOMETRIC_LIB_SOURCES])
```

2.4 trustzone_images/core/securemsm/trustzone/qsapps/libs/applib/qsee/build

a). copy project files

cd trustzone_images/core/securemsm/trustzone/qsapps/libs/applib/qsee/build

cp –arf sampleapp fingerapp

b). edit SConscript, **ONLY add FINGERAPP_IMAGE:**

```
elif env.has_key('FINGERPRINT_IMAGE'):
    LIB_ENTRY_SOURCES += [ '${BUILDPATH}/src/tzapp_lib_main.c',]
elif env.has_key('FINGERAPP_IMAGE'):
    LIB_ENTRY_SOURCES += [ '${BUILDPATH}/src/tzapp_lib_main.c',]
elif env.has_key('VOICEPRINT_IMAGE'):
```

and:

```
env.AddBinaryObject('FINGERPRINT_IMAGE', IMG_ENTRY_SOURCES)
env.AddBinaryLibrary('FINGERPRINT_IMAGE', '${BUILDPATH}/tzapp_entrylib', LIB_ENTRY_SOURCES)

env.AddBinaryObject('FINGERAPP_IMAGE', IMG_ENTRY_SOURCES)
env.AddBinaryLibrary('FINGERAPP_IMAGE', '${BUILDPATH}/tzapp_entrylib', LIB_ENTRY_SOURCES)

```

2.5 trustzone_images/core/securemsm/trustzone/qsapps/libs/applib/qsee64/build

a). copy project files

cd trustzone_images/core/securemsm/trustzone/qsapps/libs/applib/qsee64/build

cp –arf sampleapp fingerapp

b). edit SConscript, **ONLY add FINGERAPP64_IMAGE:**

```
elif env.has_key('FINGERPRINT64_IMAGE'):
    LIB_ENTRY_SOURCES += [ '${BUILDPATH}/qsee/src/tzapp_lib_main.c',]
elif env.has_key('FINGERAPP64_IMAGE'):
    LIB_ENTRY_SOURCES += [ '${BUILDPATH}/qsee/src/tzapp_lib_main.c',]
elif env.has_key('SECUREUISAMPLE64_IMAGE'):
```

And:

```

env.AddBinaryObject('FINGERPRINT64_IMAGE', IMG_ENTRY_SOURCES)
env.AddBinaryLibrary('FINGERPRINT64_IMAGE', '${BUILDPATH}/tzapp_entrylib', LIB_ENTRY_SOURCES)

env.AddBinaryObject('FINGERAPP64_IMAGE', IMG_ENTRY_SOURCES)
env.AddBinaryLibrary('FINGERAPP64_IMAGE', '${BUILDPATH}/tzapp_entrylib', LIB_ENTRY_SOURCES)

```

2.6 trustzone_images/core/securemsm/trustzone/qsapps/libs/applib/proxy/build

a). copy project files

cd trustzone_images/core/securemsm/trustzone/qsapps/libs/applib/proxy/build

cp –arf sampleapp fingerapp

b). edit SConscript, add FINGERAPP_IMAGE and FINGERAPP64_IMAGE:

```

env.AddBinaryLibrary(['KEYMASTER_IMAGE', 'FINGERAPP_IMAGE', 'FINGERAPP64_IMAGE',
                      'GPSAMPLE', 'GPTEST_IMAGE', 'GPTEST2', 'TTAARI1', 'TTACAPI1', 'T
                      'TTATCF4', 'TTATCF5', 'SAMPLEAPP_IMAGE', 'SAMPLEAPP64_IMAGE', 'A
```

2.7

trustzone_images/core/securemsm/trustzone/qsapps/libs/applib/common_applib/buil

d

a). copy project files
cd trustzone_images/core/securemsm/trustzone/qsapps/libs/applib/common_applib/build
cp –arf sampleapp fingerapp

b). edit SConscript, add FINGERAPP_IMAGE and FINGERAPP64_IMAGE:

```
145 env.AddBinaryLibrary('FINGERPRINT_IMAGE',    '${BUILDPATH}/tzapp_entrylib', LIB_ENTRY_SOURCES)
146 env.AddBinaryLibrary('FINGERPRINT64_IMAGE',  '${BUILDPATH}/tzapp_entrylib', LIB_ENTRY_SOURCES)
147 env.AddBinaryLibrary('FINGERAPP_IMAGE',     '${BUILDPATH}/tzapp_entrylib', LIB_ENTRY_SOURCES)
148 env.AddBinaryLibrary('FINGERAPP64_IMAGE',  '${BUILDPATH}/tzapp_entrylib', LIB_ENTRY_SOURCES)
149 env.AddBinaryLibrary('VOICEPRINT_IMAGE',    '${BUILDPATH}/tzapp_entrylib', LIB_ENTRY_SOURCES)
```

2.8 trustzone_images/core/securemsm/secrsa/build

edit SConscript, add FINGERAPP_IMAGE and FINGERAPP64_IMAGE:

```
72 if env.has_key('FINGERAPP_IMAGE'):
73   env.Append(CPPPATH = "${COREBSP_ROOT}/securemsm/trustzone/qsapps/libs/applib/common/src")
74   env.Append(CPPPATH = "${COREBSP_ROOT}/securemsm/secrsa/env/sampleapp/inc")
75   env.Append(CPPPATH = "${COREBSP_ROOT}/securemsm/secmath/env/sampleapp/inc")
76   env.Append(CPPPATH = "${COREBSP_ROOT}/api/securemsm/trustzone/qsee")
77   env.Append(CPPPATH = "${COREBSP_ROOT}/api/securemsm/crypto")
78 if env.has_key('FINGERAPP64_IMAGE'):
79   env.Append(CPPPATH = "${COREBSP_ROOT}/securemsm/trustzone/qsapps/libs/applib/common/src")
80   env.Append(CPPPATH = "${COREBSP_ROOT}/securemsm/secrsa/env/sampleapp/inc")
81   env.Append(CPPPATH = "${COREBSP_ROOT}/securemsm/secmath/env/sampleapp/inc")
82   env.Append(CPPPATH = "${COREBSP_ROOT}/api/securemsm/trustzone/qsee")
83   env.Append(CPPPATH = "${COREBSP_ROOT}/api/securemsm/crypto")
84
```

And:

```
100 elif env.has_key('WINSECAPP_IMAGE'):
101   env.Append(CCFLAGS = " -O3  ")
102 elif env.has_key('FINGERAPP_IMAGE'):
103   env.Append(CCFLAGS = " -O3  ")
104 elif env.has_key('FINGERAPP64_IMAGE'):
105   env.Append(CCFLAGS = " -O3  ")
106 else:
107   env.Append(CCFLAGS = " -O3  -DTZ_APP_LEGACY")
108
```

3 Edit TZ memory allocation

注意：

该步骤在移植阶段可以先略过，在进行步骤 8 时，如 TA 加载失败，可以再进行。
在修改前，请首先咨询客户开发板 memory 配置情况，该步骤在大部分情况下不需要，如有

需要，请在客户协助下进行。

某些情况下，在 CA 打开 TA 时（如第八节的例子），会发现 TA 总是打开失败，查询 logcat，如发现类似以下输出信息：

```
E/QSEECOMAPI(   902): Error::Failed to open /dev/qseecom device
E/QSEECOMAPI(   531): Error::Cannot open the file /system/etc/firmware/fngap64.mdt errno =2
E/QSEECOMAPI(   531): Error::Loading image failed with ret = -1
```

如果搜索发现系统自带的一些 TA，如 keymaster, mdtp 等也 open 失败，可以初步怀疑是 QSEE 环境问题，尤其是 TA 的堆和栈分配有问题。

如果怀疑是 QSEE 环境问题，可以先进行步骤 9， 步骤 9 里使用的 test TA 堆和栈都是使用的 QSEE 默认配置，如果 test TA 可以打开，那么就需要修改系统的 memory 配置。改动在如下三个部分：

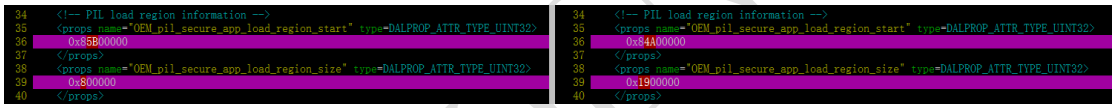
3.1 In QSEE SDK

Edit file(msm8937 for example):

trustzone_images/core/securemsm/trustzone/qsee/mink/oem/config/msm8937/oem_config.xml

0X85B00000---->0x84a00000

0x800000 ---->0x1900000 //8M -> 25M



修改完成后，需要重新编译并通过 fastboot 烧写 tz.mbn

3.2 In Linux kernel

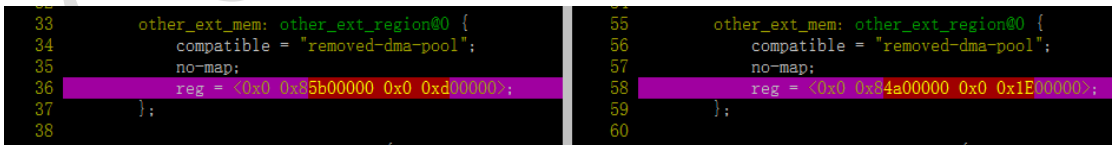
Edit file msm8937.dtsi(msm8937 for example):

Android/kernel/arch/arm64/boot/dts/qcom/msm8937.dtsi

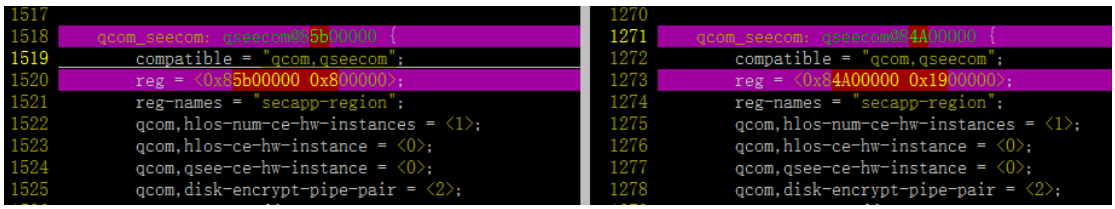
0X85B00000---->0x84a00000

0x800000 ---->0x1900000 //8M -> 25M

1.



2.



修改完成后，需要重新编译 boot.img 并且通过 fastboot 烧写 kernel

3.3 In bootloader LK

Edit file iomap.h, (msm8937 for example):
Android/bootable/bootloader/lk/platform/msm8952/include/platform/iomap.h
0X85B00000---->0x84a00000
0x800000 ---->0x1900000 //8M -> 25M

```
174 //#define APP_REGION_ADDR platform_get_tz_app_add()
175 //#define APP_REGION_SIZE platform_get_tz_app_size()
176 #define APP_REGION_ADDR 0x84A00000
177 #define APP_REGION_SIZE 0x1900000
178 #define APP_REGION_ADDR_8952 0x85E00000
179 #define APP_REGION_SIZE_8952 0x500000
180                          0x85B00000
181                          0x800000
182
```

```
174 //#define APP_REGION_ADDR platform_get_tz_app_add()
175 //#define APP_REGION_SIZE platform_get_tz_app_size()
176 #define APP_REGION_ADDR 0x84A00000
177 #define APP_REGION_SIZE 0x1900000
178 #define APP_REGION_ADDR_8952 0x85E00000
179 #define APP_REGION_SIZE_8952 0x500000
180                          0x84A00000
181                          0x1900000
182
```

修改完成后，需要重新编译并通过 fastboot 烧写 lk

4 Copy fingerprint TA source files

a).Copy ta_src/core/bsp/trustzone/qsapps/fngap64
to TZ.4.0.5/trustzone_images/core/bsp/trustzone/qsapps/
b).Copy ta_src/core/securemsm/trustzone/qsapps/fngap64
to TZ.4.0.5/trustzone_images/core/securemsm/trustzone/qsapps/fngap64

5 SPI bus configuration

根据具体的硬件连接情况，修改：
trustzone_images/core/securemsm/trustzone/qsapps/fngap64/src/fp_ta_config_qsee.c 中变量 fpsensor_qsee_spi_id 的初始值即可，如下(for example, SPI 7)：

```
 9
10 // which SPI bus is used by fpsensor fingerprint in QSEE
11 qsee_spi_device_id_t fpsensor_qsee_spi_id = QSEE_SPI_DEVICE_7;
12
```

6 Build fngap64 TA

1. cd TZ.4.0.x/trustzone_images/build/ms/
2. ./build.sh CHIPSET=msm8937 fngap64
3. 如果编译无误，生成的文件分别位于：
   a) ELF 文件：
   trustzone_images/core/bsp/trustzone/qsapps/fngap64/build/ZALAANAA/fngap64.elf
   b) mbn 文件：
   trustzone_images/build/ms/bin/ZALAANAA/fngap64.mbn

c) flist/mdt 文件

trustzone_images/build/ms/bin/PIL_IMAGES/SPLITBINS_ZALAANAA/fngap64.flist

trustzone_images/build/ms/bin/PIL_IMAGES/SPLITBINS_ZALAANAA/fngap64.mdt

d) SPLITBINS 文件

trustzone_images/build/ms/bin/PIL_IMAGES/SPLITBINS_ZALAANAA/fngap64.b00

…

trustzone_images/build/ms/bin/PIL_IMAGES/SPLITBINS_ZALAANAA/fngap64.b0*

…

trustzone_images/build/ms/bin/PIL_IMAGES/SPLITBINS_ZALAANAA/fngap64.b06

7 Install fngap64 TA

```
adb root
adb remount
adb push fngap64.mdt /etc/firmware/    (or /system/etc/firmware,    /firmware/image)
adb push fngap64.b00 /etc/firmware/    (or /system/etc/firmware,    /firmware/image)
…
adb push fngap64.b0* /etc/firmware/    (or /system/etc/firmware,    /firmware/image)
…
adb push fngap64.b06 /etc/firmware/    (or /system/etc/firmware,    /firmware/image)
```

8 Run&Verify fngap64 TA

在步骤 7 中，已经通过 ADB 将前面编译好的 fngap64 TA push 到了目标板的 Android 系统中， 为方便后继工作，减少差错可能，可以借助高通平台自带的 qseecom_sample_client 工具来加载和运行 fngap64 TA，验证 fngap64 TA 是否可以在目标板上运行，以便分析 TA 移植是否成功。详细步骤如下：

1. 打开一个 adb 终端，观察 qsee log 的输出(qsee TEE log 都会输出到文件 /d/tzdbg/qsee_log 里)：adb shell cat /d/tzdbg/qsee_log
2. 打开另一个 adb 终端，执行命令: adb shell qseecom_sample_client –v fngap64 0 1

如果移植的 fngap64 TA 没有问题，那么 qsee_log 里会输出以 fingerprint 字段开头的 log，具体如下：

```
<8>fingerprint: "tz_app_fngap64 init, Apr 10 2017-19:54:21\n"
<8>fingerprint: "tz_app_cmd_handler cmd length 0\n"
<8>fingerprint: "cmd length is 0, is this a test cmd sent by qseecom_sample_client?"
<8>fingerprint: "tz_app_shutdown invoked"
```

如果 qsee_log 没有输出或者系统崩溃，那么证明 fngap64 TA 的移植和编译有问题，需要进一步检查和分析。

9 Test&Verify fingerprint hardware

在移植指纹 TA/CA 的过程中，比较容易出问题的地方是硬件配置（主要是 SPI 和 GPIO）。但由于 TEE 的运行环境，需要分别移植的软件模块有三个部分：1. Linux driver 2. Android CA fpsensor_fingerprint.default.so 3. QSEE fngap64 TA，正常情况下，只有这三个部分都移植完毕并运行正常，才能知道指纹硬件配置是否正常。为了简化环境，我们提供了一个名叫 fngap64_test 的 TA 和 fp_ca 的测试 CA，主要目的是在开发前期阶段来验证硬件配置是否可以正常工作。

fngap64_test 是一个简化版的 fngap64 TA，他的主要工作是接收 fp_ca 发送的 CMD，然后针对具体 CMD 进行相应的硬件操作，然后将操作结果返回 fp_ca，供参考硬件是否正常。需要注意的是，进行该步骤的前提条件是指纹的 Linux driver 已经移植完毕， 且系统启动后可以生成/dev/fpsensor 设备节点。

验证硬件的步骤如下：

1. 将 fp_ca 和 fngap64_test 通过 adb push 到目标板相应位置
2. chmod 777 /dev/fpsensor， setenforce 0， 确保没有权限问题
3. 进入 adb shell 运行/system/bin/fp_ca（推荐以 root 用户来执行 fp_ca 命令），测试选项如下：

```
fp_ca.cpp -------------Tiny TEE Software--------------
Please select test case:

1: gpio irq test
2: spi test
3: tee file sys test
4: tee crypto test(only for tbase!)
5: Exit
```

4. 主要的测试选项是 1 和 2，以 spi test 为例，如果硬件无误，会读到并打印 hardware ID(如 0x7153)。测试结果同样可以观察 qsee_log 或 logcat 输出，正确的 qsee_log 如下：

```
<8>FingerApp64: "tz_app_fngap64 init, Feb  6 2017-15:19:59\n"
<8>FingerApp64: "fpsensor_spi_init init spi port 2 ok\n"
<8>FingerApp64: "fpsensor_reg_access WRITE 0x8/8 (1 bytes) 55 0 0 0 : 0 0 0 0\n"
<8>FingerApp64: "fpsensor_reg_access READ 0x0/0 (2 bytes) 71 53 0 0 : 0 0 0 0\n"
<8>FingerApp64: "tz_app_cmd_handler cmd process rsp 0, rsp_len 8\n"
<8>FingerApp64: "tz_app_shutdown invoked"
```

10 Enable Android fingerprint feature

Android 6.0 和 Android 7.0 系统原生支持指纹功能，但是在默认情况下，指纹相关的组件是没有开启的。在移植 fpsensor fingerprint 相关软件时，需要先确保 Android 指纹功能已经使能，并且 Fingerprint Service 和/system/bin/fingerprintd 可以开机正常启动。如果在调试时，发现 logcat 有如下相关 log 输出：

```
V FingerprintManager: FingerprintManagerService was null
V KeyguardUpdateMonitor: startListeningForFingerprint()
W FingerprintManager: isFingerprintHardwareDetected(): Service not connected!
W FingerprintManager: addLockoutResetCallback(): Service not connected!
D KeyguardViewMediator: setShowingLocked() - showing = true, mShowing = false
D KeyguardViewMediator: isInputRestricted: showing=true, needReshow=false
I art     : System.exit called, status: 0
I AndroidRuntime: VM exiting with result code 0, cleanup skipped.
D KeyguardViewMediator: isInputRestricted: showing=true, needReshow=false
```

此时可以 adb shell 进系统，查看/system/etc/permissions/目录下有无一个 android.hardware.fingerprint.xml 的文件，如果该文件也没有，那么证明 Android 系统的 fingerprint 功能没有使能，此时需要进行以下两步，确保使能 fingerprint 组件及功能：

1. 将 Android/frameworks/native/data/etc/android.hardware.fingerprint.xml push 到目标系统的/system/etc/permissions/目录下。
2. 确保系统/system/bin/fingerprintd 存在并且有执行权限。如果没有，请到 android/framework/core/fingerprintd/下编译，并 push 进系统。同时，在系统的开机启动脚本 init.rc 里，添加如下内容：

```
# add for Fingerprint
service fingerprintd /system/bin/fingerprintd
    class late_start
    user  system
    group system
```