Rockchip TEE安全SDK 开发指南

发布版本：V1.60

日期：2021.01

文件密级：公开资料

瑞芯微电子股份有限公司

Fuzhou Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-591-83991906

客户服务传真： +86-591-83951833

客户服务邮箱： www.rock-chips.com

前言

概述

本文档主要介绍Rockchip TEE安全相关固件说明、TEE环境搭建、CA/TA开发测试、TA调试方法、TA签名方法以及注意事项。

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

修订记录

日期	版本	作者	修改说明
2018-4-26	V1.00	张志杰	初始版本
2019-3-18	V1.10	张志杰	新增uboot TEE环境说明；优化V1与V2版本区分说明
2019-6-4	V1.20	林平	新增安全存储说明
2019-7-4	V1.30	林平	修改安全存储说明
2019-7-11	V1.40	林平	新增parameter.txt说明；新增TEE相关内核节点说明
2019-8-8	V1.50	林平	新增编译rk_tee_user报错说明
2021-1-27	V1.60	林平	optee v1内核驱动变更说明

Rockchip TEE安全SDK 开发指南

1. TrustZone简介

1.1 什么是TrustZone

ARM TrustZone技术是系统范围的安全方法，针对高性能计算平台上的大量应用，包括安全支付、数字版权管理(DRM)、企业服务和基于Web的服务。

TrustZone技术与Cortex™-A处理器紧密集成，并通过AMBA-AXI总线和特定的TrustZone系统IP块在系统中进行扩展。此系统方法意味着可以保护安全内存、加密块、键盘和屏幕等外设，从而可确保它们免遭软件攻击。

按照TrustZone Ready Program建议开发并利用TrustZone技术的设备提供了能够支持完全可信执行环境(TEE)以及安全感知应用程序和安全服务的平台。

智能手机和平板电脑等最新设备为消费者提供了基于扩展服务集的高价值体验，移动设备已发展为能够从Internet下载各种大型应用程序的开放软件平台。这些应用程序通常由设备OEM进行验证以确保质量，但并非可对所有功能进行测试，并且攻击者正在不断创建越来越多以此类设备为目标的恶意代码。

同时，移动设备处理重要服务的需求日益增加。从能够支付、下载和观看某一特定时段的最新好莱坞大片，到能够通过手机远程支付帐单和管理银行帐户，这一切都表明，新的商业模式已开始出现。

这些发展趋势已使手机有可能成为恶意软件、木马和rootkit等病毒的下一软件攻击目标。但是，通过应用基于ARM TrustZone技术的高级安全技术并整合SecurCore™防篡改元素，可开发出能够提供功能丰富的开放式操作环境和强大安全解决方案的设备。

可信应用程序采用基TrustZone技术的SoC（运行可信执行环境），与主OS分开，可防止软件/恶意软件攻击。TrustZone可切换到安全模式，提供硬件支持的隔离。可信应用程序通常是可集装箱化的，如允许不同支付公司的可信应用程序共存于一台设备上。处理器支持ARM TrustZone技术是所有Cortex-A类处理器的基本功能，是通过ARM架构安全扩展引入的。这些扩展可在供应商、平台和应用程序中提供一致的程序员模型，同时提供真实的硬件支持的安全环境。

1.2 TrustZone软硬件架构

1.2.1 硬件架构

TrustZone硬件架构旨在提供安全框架，从而使设备能够抵御将遇到的众多特定威胁。TrustZone技术可提供允许SoC设计人员从大量可在安全环境中实现特定功能的组件中进行选择的基础结构，而不提供固定且一成不变的安全解决方案。

架构的主要安全目标是支持构建可编程环境， 以防止资产的机密性和完整性受到特定攻击。具备这些特性的平台可用于构建一组范围广泛的安全解决方案，而使用传统方法构建这些解决方案将费时费力。

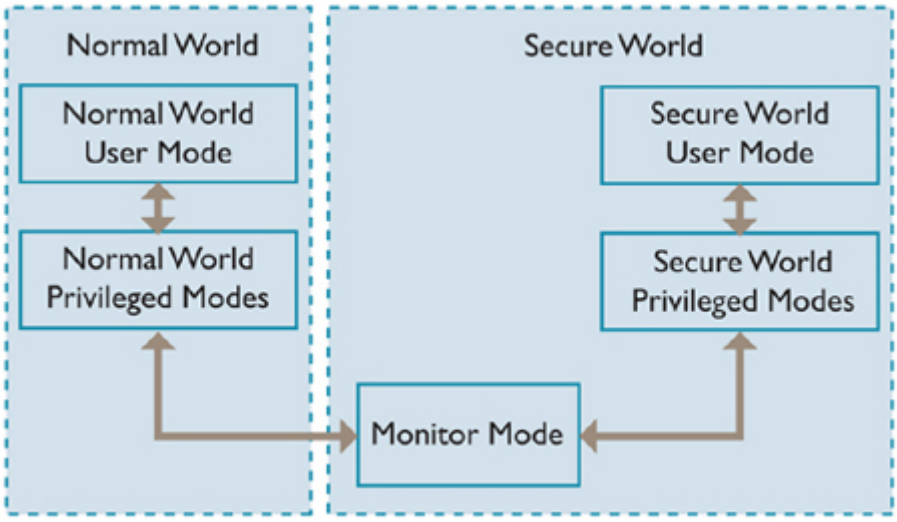

图1-1 TrustZone安全与非安全环境

可通过以下方式确保系统安全：隔离所有SoC硬件和软件资源，使它们分别位于两个区域（用于安全子系统的安全区域以及用于存储其他所有内容的普通区域）中。支持TrustZone的AMBA3 AXI™总线构造中的硬件逻辑可确保普通区域组件无法访问安全区域资源，从而在这两个区域之间构建强大边界。将敏感资源放入安全区域的设计，以及在安全的处理器内核中可靠运行软件可确保资产能够抵御众多潜在攻击，包括那些通常难以防护的攻击（例如，使用键盘或触摸屏输入密码）。通过在硬件中隔离安全敏感的外设，设计人员可限制需要通过安全评估的子系统的数目，从而在提交安全认证设备时节省成本。

TrustZone硬件架构的第二个方面是在一些ARM处理器内核中实现的扩展。通过这些额外增加的扩展，单个物理处理器内核能够以时间片的方式安全有效地同时从普通区域和安全区域执行代码。这样，便无需使用专用安全处理器内核，从而节省了芯片面积和能源，并且允许高性能安全软件与普通区域操作环境一起运行。

更改当前运行的虚拟处理器后，这两个虚拟处理器通过新处理器模式（称为监视模式）来进行上下文切换。

物理处理器用于从普通区域进入监视模式的机制受到密切控制， 并且这些机制始终被视为监视模式软件的异常。要监视的项可由执行专用指令（安全监视调用 (SMC)指令）的软件触发，或由硬件异常机制的子集触发。可对IRQ、FIQ、外部数据中止和外部预取中止异常进行配置，以使处理器切换到监视模式。

在监视模式中执行的软件是实现定义的，但它通常保存当前区域的状态，并还原将切换到的区域位置的状态。然后，它会执行从异常返回的操作，以在已还原区域中重新启动处理过程。TrustZone硬件架构的最后一个方面是安全感知调试基础结构，它可控制对安全区域调试的访问，而不会削弱普通区域的调试可视化。

1.2.2 软件架构

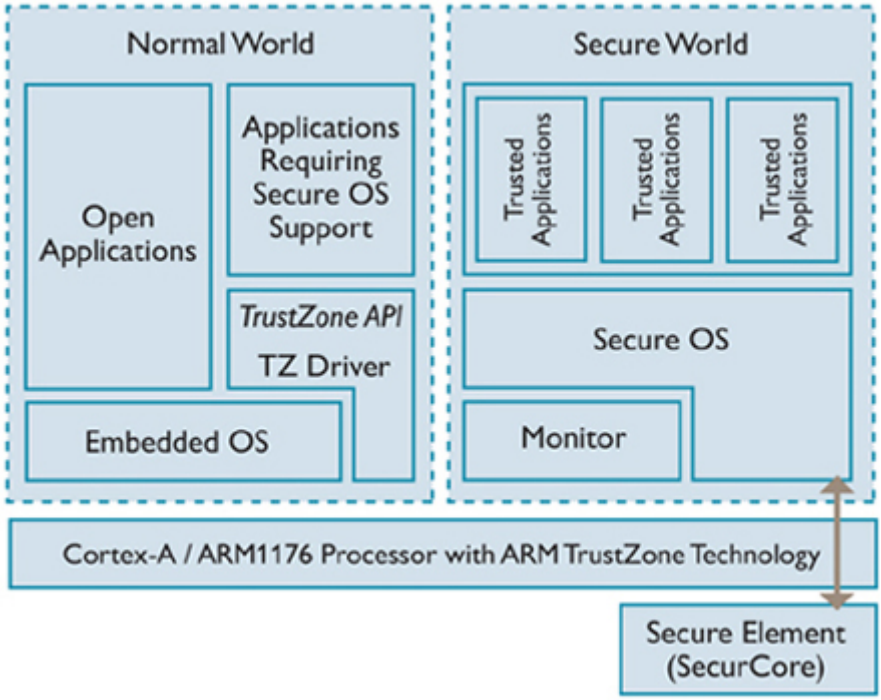

图1-2 TrustZone软件架构

在SoC硬件中实现安全区域要求在其中运行某些安全软件，并利用存储在其中的敏感资产。

可能有许多支持TrustZone的处理器内核上的安全区域软件堆栈可实现的软件架构。最高级的软件架构是专用安全区域操作系统；最简单的是放置在安全区域中的同步代码库。这两个极端架构之间有许多中间选项。

专用安全内核可能是一种复杂但强大的设计。它可模拟多个独立安全区域应用程序的并发执行、新安全应用程序的运行时下载以及完全与普通区域环境独立的安全区域任务。

这些设计与将在SoC中看到的软件堆栈非常类似，它们在非对称多处理(AMP)配置中使用两个单独的物理处理器。在每个虚拟处理器上运行的软件是独立的操作系统，并且每个区域使用硬件中断来抢占当前运行的区域和获得处理器时间。

使用将安全区域任务与请求这些任务的普通区域威胁相关联的通信协议的紧密集成设计可提供对称多处理(SMP)设计的许多优点。例如，在这些设计中，安全区域应用程序可继承它支持的普通区域任务的优先级。这将导致对媒体应用程序做出某些形式的软实时响应。

安全扩展是ARM架构的开放式组件，因此任何开发人员都可创建自定义安全区域软件环境，以满足其要求。

1.2.3 TrustZone与TEE

支付、网上银行、内容保护和企业身份验证之类的应用可通过利用TrustZone技术增强型设备所提供的三个关键要素来提高其完整性、功能和用户体验：

1. 面向软件的安全执行环境，可防止从富操作系统发起恶意软件攻击
2. 已知良好的硬件信任根，可在富操作领域检查数据和应用程序的完整性，确保安全环境不受到损害
3. 按需访问安全外设，如内存、键盘/触摸屏，甚至显示器

基于ARM TrustZone技术的设备与开放API相结合，提供了可信执行环境(TEE)，开发人员需要通过一种新型软件才能实现其功能和一致性：这种软件就是可信应用程序。典型可信应用程序可在普通区域和安全区域各包含部分代码，例如，处理关键存储和操控。TEE还提供了与其他可信应用程序的隔离，使多个可信服务可以共存。

TEE API的标准化（由GlobalPlatform管理）将会使服务提供商、运营商和OEM的可互操作可信应用程序和服务实现市场化。

ARM TrustZone技术无需单独的安全硬件来验证设备或用户的完整性。它通过在主手机芯片集中提供真正的硬件信任根来实现这一点。

为确保应用程序的完整性，TrustZone还提供了安全执行环境（即可信执行环境 (TEE)），在此环境中只有可信应用程序才能运行，从而防止遭到黑客/病毒/恶意软件形式的攻击。

TrustZone硬件提供了TEE与软件攻击媒介的隔离。硬件隔离可扩展为保护一直到物理外设（包括键盘/触摸屏等）的数据输入和输出。

正是具备了这些关键功能，采用TrustZone技术的芯片集提供了众多机会来重新定义用户可以访问的服务（更多、更好的服务），如何访问服务（更快、更轻松）以及在何处访问服务（随时随地）。

在大多数 Android设备上，Android Boot加载程序都不会验证设备内核的真实性。希望进一步控制其设备的用户可能会安装破解的Android内核来对设备进行root。破解的内核可让超级用户访问所有数据文件、应用程序和资源。一旦破解内核损坏，则会导致服务被拒绝。如果内核包含恶意软件，则将危害企业数据的安全性。

而Secure Boot可有效防止上述问题，Secure Boot是一种安全机制，它可防止在启动过程中加载未经授权的启动加载程序和内核。由值得信任的已知权威机构以加密方式签名的固件映像（如操作系统和系统组件）会被视为经过授权的固件。安全启动组件可以形成第一道防线，用以防范恶意软件对设备进行攻击。

2. TEE环境

2.1 平台说明

Rockchip平台中Android 7.1及更高版本SDK默认均支持TEE环境，Android7.1以下版本默认不支持TEE环境。Linux版本SDK默认不支持TEE环境。

Rockchip平台采用的TEE方案为OP-TEE，TEE API符合GlobalPlatform标准。

目前运行在rockchip平台上的OP-TEE有两个版本，OP-TEE V1与OP-TEE V2。

1.RK312x、RK322x、RK3288、RK3328、RK322xh、RK3368、RK3399、RK3399Pro采用OP-TEE V1版本。

2.RK3326、RK3308、RK1808、RV1109/RV1126、RK3566/RK3568和后续新平台均采用OP-TEE V2版本。

两个版本在TEE库文件、TA文件、Secure OS固件方面均有不同，需根据具体平台采用不同版本TEE相关组件。

2.2 Parameter.txt说明

Parameter.txt文件记录了各镜像及分区的位置与大小信息，Rockchip的OP-TEE目前同时支持security与rpmb两种安全存储文件系统，具体使用哪种文件系统由TA中设置storageID参数来决定，若parameter.txt中没有定义security分区，则TA无法使用security安全存储文件系统，设置security分区的方法只需在parameter.txt中添加0x00002000@0x000xxxxx(security)即可，0x00002000表示大小4M, 0x000xxxxx表示起始地址，根据用户的parameter.txt来修改。

2.3 TEE固件

TEE Secure OS的源码默认不开源，binary位于目录u-boot/tools/rk_tools/bin或rkbin/bin下。

1. ARMv7平台的TEE binary由工具u-boot/tools/loaderimage打包成固件trust.img，TEE binary的命名如下：

```
<platform>_tee_[ta]_<version>.bin
```

名称中带ta的为支持外部TA运行，不带ta则不支持运行外部TA。

2. ARMv8平台的TEE binary由工具u-boot/tools/trust_merger将BL31/BL32等bin打包成固件trust.img，TEE binary的命名如下：

```
<platform>_bl32_<version>.bin
```

3. 若rkbin/RKTRUST/.ini中[BL32_OPTION]下SEC=0，则需要将其改成SEC=1，否则trust.img将不包含Secure OS，无法运行TEE相关服务。

2.4 U-Boot 中TEE驱动

目前一些安全的操作需要在U-Boot这级操作，比如读取一些数据必须需要OP-TEE帮忙获取。U-Boot里面实现了OP-TEE Client代码，可以通过该接口与OP-TEE通信。OP-TEE Client驱动在lib/optee_client目录下，API符合GP规范。目前尚不支持客户在U-Boot中开发自己的CA/TA应用。

2.4.1 宏定义说明

CONFIG_OPTEE_CLIENT，U-Boot调用OP-TEE总开关。

CONFIG_OPTEE_V1，采用OP-TEE V1的平台使用。

CONFIG_OPTEE_V2，采用OP-TEE V2的平台使用。

CONFIG_OPTEE_ALWAYS_USE_SECURITY_PARTITION， 当emmc的rpmb不能用，才开这个宏，默认不开。

2.4.2 共享内存说明

U-Boot与OP-TEE通信时，数据需放在共享内存中，可以通过TEEC_AllocateSharedMemory()来申请共享内存，但各个平台共享内存大小不同，建议不超过1M，若超过则建议分割数据多次传递，使用完需调用TEEC_ReleaseSharedMemory()释放共享内存。

2.4.3 测试命令

测试安全存储功能，需进入U-Boot串口命令，执行：

```
=> mmc testsecurestorage
```

该测试用例将循环测试安全存储读写功能，测试程序会自动检查硬件，当硬件使用emmc时将测试rpmb与security分区两种安全存储方式，当硬件使用nand时只测试security分区安全存储。

2.4.4 常见错误打印

```
"TEEC: Could not find device"
```

没有找到emmc或者nand设备，请检查U-Boot中驱动，或者硬件是否损坏。

```
"TEEC: Could not find security partition"
```

当采用security分区安全存储时，加密数据会存储在该分区，请检查parameter.txt中是否定义了security分区。

```
"TEEC: verify [%d] fail, cleanning ...."
```

第一次使用security分区进行安全存储时，或者security分区数据被非法篡改时出现，security分区会全部清空。

```
"TEEC: Not enough space available in secure storage !"
```

安全存储的空间不足，请检查存储的数据是否过大，或者存储的文件数量过多，或者之前是否存储过大量的数据而没有删除。

```
INF [0x0] TEE-CORE:storage_read_obj:201: Warning! head data not find!
ERR [0x0] TEE-CORE:storage_read_obj:210: cpu or emmc was replaced!
```

U-Boot启动过程中关键数据会调用TEE进行安全存储，关键数据经过TEE加密存储在security分区或者rpmb分区，而且加密密钥与CPU绑定；若客户更换CPU将导致关键数据无法正常解密，导致U-Boot启动失败；若客户更换emmc且emmc之前被使用过，security分区或者rpmb分区存在旧数据，也会出现该错误导致U-Boot启动失败；解决办法为需要清空security分区或者rpmb分区中的旧数据，若使用的是security分区则直接格式化emmc即可，若是使用rpmb则需要联系技术支持提供特殊固件清除rpmb中旧数据。

```
“optee check api revision fail”
```

U-Boot与TEE版本不匹配，U-Boot版本高于TEE版本，解决办法如下（二选一）：

1. 回退U-Boot版本至cf13b78438 (tag: android-10.0-mid-rkr9) rockchip: spl: add rollback index check with otp。
2. revert以下几个提交：

   396e3049bd rochchip: board: only map op-tee share memory as dcache enabled

   7a349fdcbd lib: optee_client: add optee initialize flag

   74eb602743 lib: optee_client: update to new optee msg for optee v1 platform

   102dfafc4a rochchip: board: map op-tee memory as dcache enabled

正常情况下对外释放的SDK版本都是匹配的。

```
"optee api revision mismatch with u-boot/kernel, panic"
```

若在U-Boot启动阶段打印，则是U-Boot版本与TEE版本不匹配，U-Boot版本低于TEE版本，可升级U-Boot版本至396e3049bd (tag: android-10.0-mid-rkr11, tag: android-10.0-mid-rkr10) rochchip: board: only map op-tee share memory as dcache enabled及以上版本。

若在Android系统启动阶段打印，则升级android/vendor/rockchip/common版本至8bc7bf97 (tag: android-10.0-mid-rkr10) vpu: librockit: add Rockit MetadataRetriever及以上版本。

若在Linux系统启动阶段打印，则升级linux/external/security/bin版本至f59085c optee_v1: lib: arm&arm64: update binary and library及以上版本。

正常情况下对外释放的SDK版本都是匹配的。

2.5 TEE linux kernel驱动

TEE linux kernel驱动位于security/optee_linuxdriver/与drivers/tee/中。

2.5.1 OP-TEE V1

采用OP-TEE V1的芯片的驱动位于security/optee_linuxdriver/，默认均有开启。开启方法如下：

config中添加以下配置：

```
CONFIG_TEE_SUPPORT=y
```

目前我们将逐步废弃OP-TEE V1的TEE linux kernel驱动，OP-TEE V1平台将使用OP-TEE V2的TEE linux kernel驱动，若rkbin/bin目录下TEE binary文件名中 version >= v2.00，则还需要开启OP-TEE V2的TEE linux kernel驱动。

2020年8月份以后释放的Android10以及以上版本默认使用OP-TEE V2的kernel驱动。

2.5.2 OP-TEE V2

采用 OP-TEE V2 的芯片的驱动位于 drivers/tee/下，开启方法如下： 确认对应平台 dtsi 中添加了如下节点：

```
firmware {
        optee: optee {
            compatible = "linaro,optee-tz";
            method = "smc";
            #status = "disabled";
        };
    };
```

各平台默认情况下都有添加该节点，但部分平台会设置 **status = "disabled";** 导致该驱动默认关闭，所以如果要开启optee驱动，只要去除 **status = "disabled";** 即可。

config 中添加以下两个配置：

```
CONFIG_TEE=y
CONFIG_OPTEE=y
```

2.5.3 确认驱动开启

若出现 “/dev/opteearmtz00” 节点，说明optee v1的TEE linux kernel驱动已开启； 若出现 “/dev/tee0” 和 “/dev/teepriv0“ 节点，说明optee v2的TEE linux kernel驱动已开启。

2.6 TEE库文件

2.6.1 Android

TEE环境相关组件在Android工程目录vendor/rockchip/common/security下（包含V1与V2版本，需根据不同平台采用不同版本文件）：

1. lib：包含32bit与64bit平台编译出来的tee-supplicant、libteec.so以及keymaster/gatekeeper相关库文件。
2. ta：存放编译好的keymaster/gatekeeper等相关TA文件。

2.6.2 Linux

TEE环境相关组件在linux工程目录external/security/bin下（包含V1与V2版本，需根据不同平台采用不同版本文件）：

1. lib：包含32bit与64bit平台编译出来的tee-supplicant、libteec.so以及其他CA相关库文件。

2. ta：存放编译好的TA文件。

3. CA/TA开发与测试

3.1 Android

3.1.1 目录介绍

TEE CA/TA开发环境在安卓工程目录external/rk_tee_user下：

1. Android.mk：其中决定了编译的工具和需要编译的ca 文件。
2. host：存放CA的相关源文件。
3. ta：存放TA的源文件。
4. export*：存放编译TA 所依赖的环境。

3.1.2 编译开发说明

若external/rk_tee_user目录下只有v1/ v2/两个目录，说明master分支代码已经合并到develop-next分支，master分支将被废弃，合并点为master分支492f1cbf testapp: support new OP-TEE MSG，执行如下命令开始编译。

```
#OP-TEE V1平台进入v1目录
cd external/rk_tee_user/v1
#OP-TEE V2平台进入v2目录
cd external/rk_tee_user/v2
rm -rf out/
./build.sh ta
mm
```

若external/rk_tee_user目录下没有v1/ v2/两个目录，说明依然使用两个分支，OP-TEE V1请先切换到master分支，OP-TEE V2请先切换到develop-next分支，执行如下命令开始编译。

```
cd external/rk_tee_user/
rm -rf out/
./build.sh ta (git log包含“Android.mk: remove build ta from android”则执行，否则不执行)
mm
```

编译成功后会得到相应的执行程序，执行程序分为CA（Client Application，运行在normal world）和TA（Trust Application，运行在secure world）。

CA为普通执行文件，编译后生成于Android工程out目录下中，testapp与testapp_storage为RK编写的demo程序。TA是文件名为uuid，后缀为.ta的文件，编译后生成于rk_tee_user/ta、rk_tee_user/out/ta、rk_tee_user/v1/out/ta、rk_tee_user/v2/out/ta其中一个目录对应的文件夹中。

若编译报错“No module named Crypto.Signature”，这是用户电脑没有安装python的算法库导致的，执行如下命令即可：

```
pip uninstall Crypto
pip uninstall pycrypto
pip install pycrypto
```

3.1.3 运行测试TEE环境

1. adb shell进入设备

2. Android 7：libteec.so放置到/system/lib或/system/lib64目录下；tee-supplicant， testapp放置到/system/bin目录下；创建/system/lib/optee_armtz目录，8cccf200-2450-11e4-abe20002a5d5c52c.ta或8cccf200-2450-11e4-abe2-0002a5d5c52c.ta放置到/system/lib/optee_armtz目录下。

   Android 8及更高版本: libteec.so放置到/vendor/lib或/vendor/lib64目录下；tee-supplicant，testapp放置到/vendor/bin目录下；创建/vendor/lib/optee_armtz目录，8cccf200-2450-11e4-abe20002a5d5c52c.ta或8cccf200-2450-11e4-abe2-0002a5d5c52c.ta放置到/vendor/lib/optee_armtz目录下。

   （若开机tee-supplicant自启动，则tee-supplicant和libteec.so不用再push，系统中已有这两个文件；libteec.so和tee-supplicant注意区分OP-TEE V1与OP-TEE V2，注意区分32位和64位；

   push后检查下tee-supplicant和testapp是否有执行权限）

3. 若开机未自动运行tee-supplicant，则需手动root权限后台运行tee-supplicant：

```
# tee-supplicant &
```

4. 运行testapp，成功提示PASS，失败提示Fail：

```
# testapp
```

5. 若testapp运行通过，则TEE环境正常，可进行TEE相关开发。

   若运行报错，请先检查驱动及各组件；

   也可能是rk_tee_user版本与TEE OS版本不匹配导致，以下为常用匹配关系：

   OP-TEE V1:

   rkbin/bin目录下TEE binary文件名中 version >= v2.00，

   对应492f1cbf testapp: support new OP-TEE MSG

   rkbin/bin目录下TEE binary文件名中 version < v2.00，

   对应e8d7215d Android.mk: support build in android R

   或者466515ec add tools for user to resign TA

   OP-TEE V2:

   TEE启动阶段串口打印"OP-TEE version: 3.6.0"，

   对应1aa969e2 Android.mk: support build in android R

   TEE启动阶段串口打印"OP-TEE version: 3.3.0"

   对应aa0a0c00 Android.mk: remove build ta from android

   TEE启动阶段串口打印"OP-TEE version: 2.5.0"，

   对应1ec9913a add tools for user to resign TA

6. 同理，可利用testapp_storage和对应TA测试Secure Storage环境是否正常。

   执行testapp_storage测试前，需要确保内核对应节点存在，/dev/block/by-name/security对应security分区；rpmb安全存储需要三个节点，/dev/block/mmcblk%u，/dev/block/mmcblk%urpmb，

   /sys/class/mmc_host/mmc%u/mmc%u:0001/cid，%u值为0 1 2任意一个；若节点不存在请链接到对应节点。

3.1.4 开发CA/TA

可参考testapp，TA中的Makefile与头文件的UUID需要修改成新生成的UUID，可用uuidgen命令生成。

在每个TA的include目录下的头文件user_ta_header_defines.h中定义了堆栈的大小，堆的大小为32KB（TA_DATA_SIZE），栈的大小为2KB（TA_STACK_SIZE）。一般情况下最好不要去修改，若实在无法满足需求，可适当改大一些，堆的大小不要超过1MB，栈的大小不要超过64KB。

```
#define TA_STACK_SIZE                (2 * 1024)
#define TA_DATA_SIZE                 (32 * 1024)
```

3.2 Linux

3.2.1 目录介绍

TEE CA/TA开发环境在linux工程目录external/security/rk_tee_user下：

1. build.sh：编译执行脚本，编译说明请参考脚本中的注释。
2. Makefile：其中决定了编译的工具和需要编译的ca 文件。
3. host：存放CA的相关源文件以及对应Makefile。
4. ta：存放TA的源文件。
5. export*：存放编译TA 所依赖的环境。

3.2.2 编译开发说明

若external/security/rk_tee_user目录下只有v1/ v2/两个目录，说明master分支代码已经合并到develop-next分支，master分支将被废弃，合并点为master分支492f1cbf testapp: support new OP-TEE MSG，执行如下命令开始编译。

```
#OP-TEE V1平台进入v1目录
cd external/security/rk_tee_user/v1
#OP-TEE V2平台进入v2目录
cd external/security/rk_tee_user/v2
rm -rf out/
./build.sh 3232 （32位平台执行，CA 32bits，TA 32bits）
./build.sh 6432 （64位平台执行，CA 64bits，TA 32bits）
```

若external/security/rk_tee_user目录下没有v1/ v2/两个目录，说明依然使用两个分支，OP-TEE V1请先切换到master分支，OP-TEE V2请先切换到develop-next分支，执行如下命令开始编译。

```
cd external/security/rk_tee_user/
rm -rf out/
./build.sh 3232 （32位平台执行，CA 32bits，TA 32bits）
./build.sh 6432 （64位平台执行，CA 64bits，TA 32bits）
```

编译成功后会得到相应的执行程序，执行程序分为CA（Client Application，运行在normal world）和TA（Trust Application，运行在secure world）。

CA为普通执行文件，编译后生成于rk_tee_user/out、rk_tee_user/v1/out、rk_tee_user/v2/out其中一个目录下对应的文件夹中，testapp与testapp_storage为RK编写的demo程序。TA是文件名为uuid，后缀为.ta的文件，编译后生成于rk_tee_user/out/ta、rk_tee_user/v1/out/ta、rk_tee_user/v2/out/ta其中一个目录对应的文件夹中。

若编译报错“No module named Crypto.Signature”，这是用户电脑没有安装python的算法库导致的，执行如下命令即可：

```
pip uninstall Crypto
pip uninstall pycrypto
pip install pycrypto
```

3.2.3 运行测试TEE环境

1. 进入设备。
2. libteec.so*等库文件放置到/lib或/lib64目录下；tee-supplicant，testapp放置到/usr/bin目录下；创建/lib/optee_armtz目录，8cccf200-2450-11e4-abe20002a5d5c52c.ta或8cccf200-2450-11e4-abe2-0002a5d5c52c.ta放置到/lib/optee_armtz目录下。

   （若开机tee-supplicant自启动，则tee-supplicant和libteec.so不用再push，系统中已有这两个文件；

   libteec.so和tee-supplicant注意区分OP-TEE V1与OP-TEE V2，注意区分32位和64位；

   push后检查下tee-supplicant和testapp是否有执行权限）
3. 若开机未自动运行tee-supplicant，则需手动root权限后台运行tee-supplicant：

   ```
   # tee-supplicant &
   ```

4. 运行testapp，成功提示PASS，失败提示Fail：

   ```
   # testapp
   ```

5. 若testapp运行通过，则TEE环境正常，可进行TEE相关开发。

   若运行报错，请先检查驱动及各组件；

   也可能是rk_tee_user版本与TEE OS版本不匹配导致，以下为常见匹配关系：

   OP-TEE V1:

   rkbin/bin目录下TEE binary文件名中 version >= v2.00，

   对应492f1cbf testapp: support new OP-TEE MSG

   rkbin/bin目录下TEE binary文件名中 version < v2.00，

   dd23392a makefile: support build CA when run build.sh

   OP-TEE V2:

   TEE启动阶段串口打印"OP-TEE version: 3.6.0"，

   对应d39ab494 Update export-ta_arm32/ and export-ta_arm64/

   TEE启动阶段串口打印"OP-TEE version: 3.3.0"，

   对应44def952 makefile: support compile 32 bits CA

   TEE启动阶段串口打印"OP-TEE version: 2.5.0"，

   对应1ec9913a add tools for user to resign TA
6. 同理，可利用testapp_storage和对应TA测试Secure Storage环境是否正常。

   执行testapp_storage测试前，需要确保内核对应节点存在，/dev/block/by-name/security对应security分区；rpmb安全存储需要三个节点，/dev/mmcblk%u，/dev/mmcblk%urpmb，/sys/class/mmc_host/mmc%u/mmc%u:0001/cid，%u值为0 1 2任意一个；若节点不存在请链接到对应节点。

3.2.4 开发CA/TA

可参考testapp，TA中的Makefile与头文件的UUID需要修改成新生成的UUID，可用uuidgen命令生成。

在每个TA的include目录下的头文件user_ta_header_defines.h中定义了堆栈的大小，堆的大小为32KB（TA_DATA_SIZE），栈的大小为2KB（TA_STACK_SIZE）。一般情况下最好不要去修改，若实在无法满足需求，可适当改大一些，堆的大小不要超过1MB，栈的大小不要超过64KB。

```
#define TA_STACK_SIZE                (2 * 1024)
#define TA_DATA_SIZE                 (32 * 1024)
```

4. TA签名方法

4.1 签名TA过程

在编译TA时，编译脚本将自动使用rk_tee_user工程export-user_ta/keys目录或者export-ta_arm32/keys目录下的密钥对TA应用进行签名，该密钥为pem格式的2048长度RSA密钥。

为防止客户A的TA应用运行在客户B的板子上，建议客户生成一个2048长度RSA密钥，替换上诉目录中的密钥并重新编译TA应用。

4.2 验证TA过程

在加载运行TA时，TEE OS将验证TA的合法性，验证通过才能正常运行TA应用。若客户替换了签名TA的密钥，则TEE OS中用于验证TA合法性的公钥也需要随之替换，客户可以使用工具替换TEE binary中的公钥，所需工具在rk_tee_user工程tools/目录中。

Linux下替换：

```
./change_puk  --teebin  <TEE binary>
```

该命令将自动生成一个2048长度的RSA密钥oemkey.pem并保存在当前目录下，并自动使用该密钥中的公钥替换TEE binary中的原始公钥。

```
./change_puk  --teebin  <TEE binary> --key oemkey.pem
```

使用客户指定的密钥中的公钥来替换TEE binary中的原始公钥，密钥长度须2048长度。

Windows下替换：

打开Windows_change_puk.exe点击“生成oemkey.pem”按钮生成并保存密钥。

选择刚刚生成的密钥和镜像，点击修改公钥。

（由于Windows_change_puk.exe会调用BouncyCastle.Crypto.dll第三方库，请确保BouncyCastle.Crypto.dll与Windows_change_puk.exe在同一目录下）

5. TA调试方法

当TA出现异常时会打印如下信息。

```
user TA data-abort at address 0x8888

 fsr 0x00000805  ttbr0 0x6846c46a  ttbr1 0x6846806a  cidr 0x2

 cpu #0          cpsr 0x00000030

 r0 0x60000013      r4 0x001007b8    r8 0x68471754   r12 0x000000ab

 r1 0x0000003a      r5 0x00200da9    r9 0x68415491    sp 0x00100720

 r2 0x00000031      r6 0x001005a0   r10 0x00000000    lr 0x0020265f

 r3 0x00008888      r7 0x00100728   r11 0x00000000    pc 0x00200104

Status of TA 8cccf200-2450-11e4-abe20002a5d5c52c (0x68467450) (active)
- load addr : 0x200000    ctx-idr: 2
- code area : 0x68700000 1048576
- stack: 0x68800000 stack:2048
DBG [0x0] TEE-CORE:get_fault_type:455: [abort] abort in User mode (TA will panic)
DBG [0x0] TEE-CORE:user_ta_enter:465: tee_user_ta_enter: TA panicked with code
0xdeadbeef
```

图5-1 TA异常log信息

图中pc 0x00200104就是异常位置。进入rkdemo目录下，输入下面命令

```
arm-eabi-objdump -S 8cccf200-2450-11e4-abe20002a5d5c52c.elf | less
```

得到反汇编信息，由于TA的运行地址从2M位置开始，所以在反汇编信息中搜索104（PC – 0x200000），得到如下图反汇编信息，图中红色就是异常位置。

```
      e8:       4b3f            ldr     r3, [pc, #252]  ; (1e8
<TA_InvokeCommandEntryPoint+0x128>)
      ea:       447b            add     r3, pc
      ec:       9300            str     r3, [sp, #0]
      ee:       4b3f            ldr     r3, [pc, #252]  ; (1ec
<TA_InvokeCommandEntryPoint+0x12c>)
      f0:       447b            add     r3, pc
      f2:       4618            mov     r0, r3
      f4:       215e            movs    r1, #94 ; 0x5e
      f6:       2202            movs    r2, #2
      f8:       2301            movs    r3, #1
      fa:       f004 ff8d       bl      5018 <trace_printf>
        *(char*)0x8888 = '1';
      fe:       f648 0388       movw    r3, #34952      ; 0x8888
     102:       2231            movs    r2, #49 ; 0x31
     104:       701a            strb    r2, [r3, #0]
        IMSG("=========2=====");
     106:       4b3a            ldr     r3, [pc, #232]  ; (1f0
<TA_InvokeCommandEntryPoint+0x130>)
     108:       447b            add     r3, pc
     10a:       9300            str     r3, [sp, #0]
     10c:       4b39            ldr     r3, [pc, #228]  ; (1f4
<TA_InvokeCommandEntryPoint+0x134>)
     10e:       447b            add     r3, pc
     110:       4618            mov     r0, r3
     112:       2160            movs    r1, #96 ; 0x60
     114:       2202            movs    r2, #2
     116:       2301            movs    r3, #1
     118:       f004 ff7e       bl      5018 <trace_printf>
```

图5-2 TA反汇编信息

6. 内存相关说明

6.1 OP-TEE V1

ARMv8架构芯片中TEE内存分配情况如下：

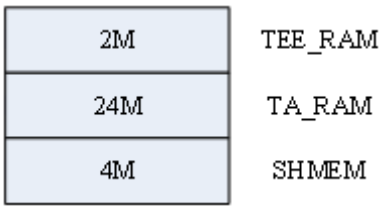

图6-1 ARMv8架构TEE内存分配情况

说明：占用内存总共30M，Secure OS运行在TEE_RAM，TA运行在 TA_RAM ，共享内存占用4M空间。

ARMv7架构芯片中TEE内存分配情况如下：

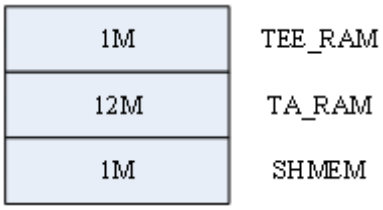

图6-2 ARMv7架构TEE内存分配情况

说明：占用内存总共14M, Secure OS运行在TEE_RAM，TA运行在TA_RAM，共享内存占用1M空间。

6.2 OP-TEE V2

各平台大小不固定，并且运行大小可能会调整，这里就不统一说明了。

7. 安全存储说明

1. 安全存储是OPTEE OS重要的功能之一，一般用于存储用户重要数据，数据经过OPTEE OS加密存储于security分区或者rpmb分区，具体存储于哪个分区由CA传递参数告知TA，由TA负责存储。
2. Uboot端安全存储，uboot端optee client代码存放于lib/optee_client目录下，一般情况下使用emmc时安全存储存于rpmb，使用nand时安全存储存于security分区，当emmc的rpmb不可用时可以开启CONFIG_OPTEE_ALWAYS_USE_SECURITY_PARTITION强制使用security分区，相关宏定义如下：

   CONFIG_OPTEE_CLIENT，U-Boot调用OP-TEE总开关。

   CONFIG_OPTEE_V1，采用OP-TEE V1的平台使用。

   CONFIG_OPTEE_V2，采用OP-TEE V2的平台使用。

   CONFIG_OPTEE_ALWAYS_USE_SECURITY_PARTITION， 当emmc的rpmb不能用，才开这个宏，默认不开。

   Uboot端不支持用户开发自己的TA应用，但是可以开发自己的CA调用OPTEE OS内部静态TA存储用户自己的数据，用户只需要设置对应的文件名和数据即可进行安全存储，细节请参考lib/optee_client/OpteeClientInterface.c中的函数。
3. Android端安全存储，定义PRODUCT_PROPERTY_OVERRIDES += ro.tee.storage=auto代表根据硬件来选择安全存储区域，emmc使用rpmb，nand使用security；

   定义PRODUCT_PROPERTY_OVERRIDES += ro.tee.storage=rpmb代表使用rpmb

   定义PRODUCT_PROPERTY_OVERRIDES += ro.tee.storage=rkss代表使用security分区

4. 在进行安全存储写入数据时请确保设备不发生断电，虽然我们做了断电保护测试，但不保证文件系统完整性，所以建议用户减少对重要数据的写入次数，确保操作安全数据时环境的稳定性。

8. 相关资料扩展

ARM官方TrustZone

https://developer.arm.com/ip-products/security-ip/trustzone

GlobalPlatform官网：

https://globalplatform.org/

该网站可下载CA开发API参考文档：TEE Client API Specification v1.0

TA开发API参考文档：TEE Internal Core API Specification v1.1

以及其他架构方面参考文档。