一.TEE相关知识

1.1 TEE简介

TEE (Trusted Execution Environment) 可信执行环境被广泛应用在手机，平板电脑中（指纹支付，生物识别）。越来越多的手机厂商开始集成TEE以及相关的可信应用。具体学习可以参考ARM开源社区OP-TEE。

github地址：GitHub - OP-TEE/optee_os: Trusted side of the TEE

TEE是基于TrustZone技术建立起来的更好安全级别的可信执行环境，是典型的软硬件协同合作的概念。

1.2 TEE解决方案

TEE是套完整的安全解决方案，主要有

- CA Client Application 运行在正常世界状态的客户端应用。
- TA Trusted Application 运行在TEE环境下的应用叫可信应用程序。
- Secure Driver SD 可信硬件驱动
- TEEOS （Trust execution Environment Operation System TEEOS）可信内核系统，用来运行TA的一个小型系统。真实存在的。
- 其他系统配置，内部逻辑，安全设备以及与CPU有关的集成电路

TEE就是通过软件和硬件结合构建的一个可信任的安全执行环境。目前国内外各种TEE解决方案都遵循GP(global platform)规范。GP规范规定了tee解决方案的架构以及相应的api原型。同时也有使用较多的PSA方案。

1.3 TrustZone技术

TrustZone是ARM架构中的一种技术（从ARMv6开始）。支持TrustZone技术在运行时，根据CPU的工作状态分为正常世界状态（非安全状态度）和安全世界状态（安全状态）。

TEE就是基于TrustZone技术提供了可信运行环境。

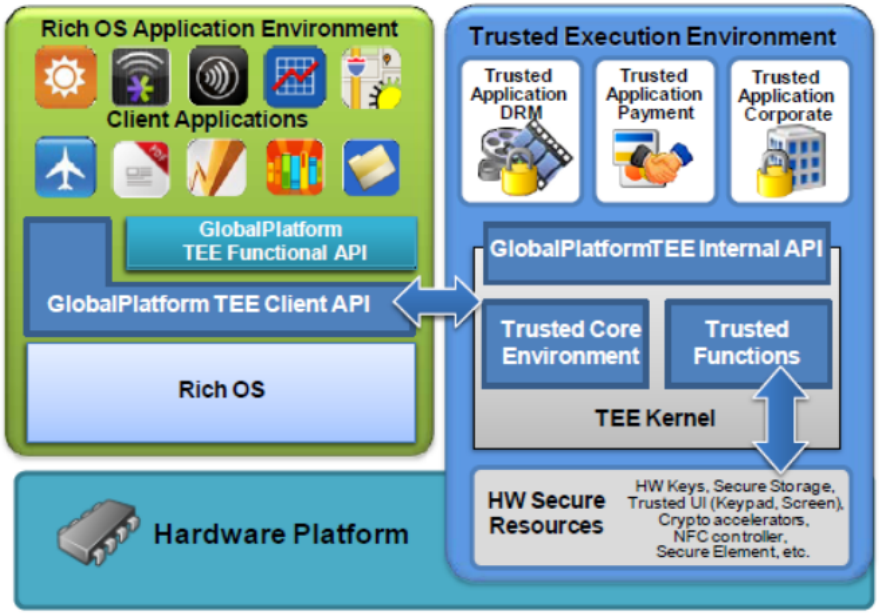
1.3.1 REE与TEE区别

富执行环境：Rich Execution Environment

从系统的软件层面看，一般的操作系统以及应用都运行在正常世界状态。正常世界状态内的开发资源相对于安全世界状态较为丰富，所以在正常世界状态的环境叫富执行环境（REE）

可信执行环境：Trust Execution Enviroment

Tee运行在安全世界状态，可信任的操作系统以及上层的可信应用（Trust Application TA）运行在安全世界状态。这部分环境就是TEE。



通俗来说安全世界状态就在系统中提供了一个相对可信赖的运行环境，通过用户的关键数据或应用可以在这个安全环境中使用和运行。处于正常世界中的linux等即使系统攻破，攻击者也无法访问安全世界状态中的重要数据。

处理器核状态	运行代码	访问权限
非安全状态	只能运行REE侧代码	只能REE侧地址空间访问权限，通过特定接口访问TE侧数据
安全状态	只能运行TEE侧代码	两侧地址空间都可访问

这种安全保护是系统资源硬件级别的隔离

1.3.2 安全状态读写信号位

系统通过安全监控模式调用(SMC，Secure Monitor Call)指令实现ARM核的安全状态与非安全状态之间的切换。

通俗就是如何区分安全世界和正常世界的访问请求的。

CPU在访问安全设备或者安全内存地址空间（访问安全世界）时，芯片级别的安全扩展组件会去校验CPU发送的访问请求的安全状态读写信号位（Non-secure bit，NS bit）是0还是1，以此来判定当前CPU发送的资源访问请求是安全请求还是非安全请求。

当ARM核处于安全状态时：发送到系统总线上读写请求的安全状态读写标志位会被强制设置为0。表示当前CPU请求为安全请求。才能访问对应TEE侧的数据资源。

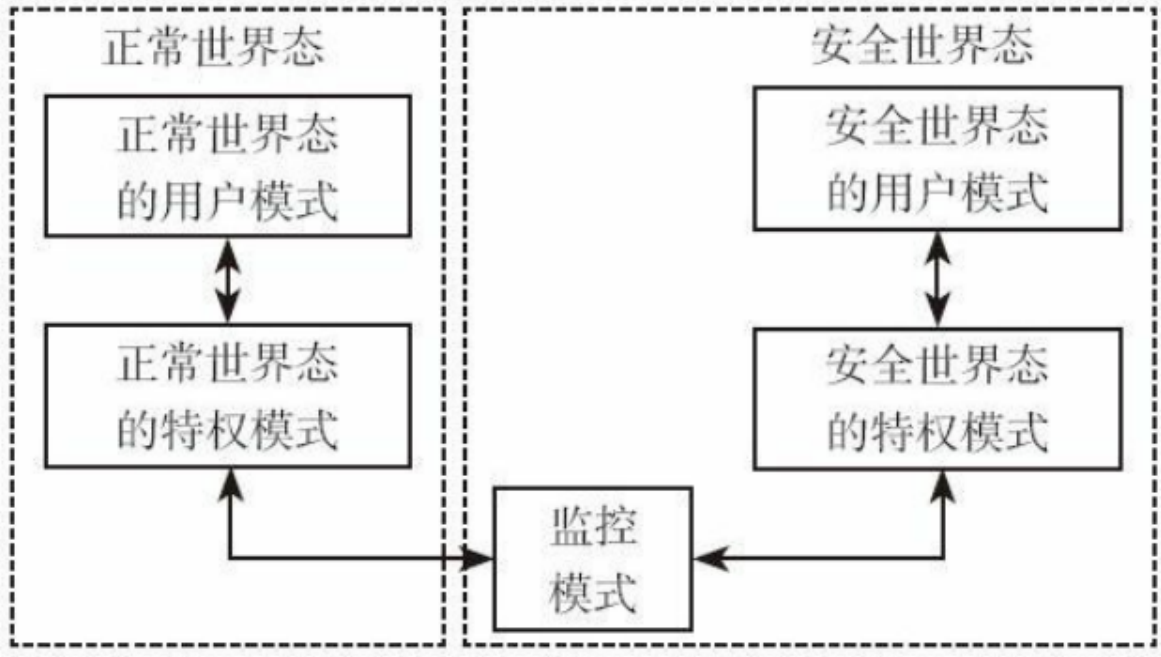
当ARM核处于非安全状态时，发送到系统总线上读写请求的安全状态读写标志位会被强制设置为1。表示当前CPU请求为非安全请求。芯片安全组件会根据对资源的访问权限配置来决定是否响应。

非安全请求试图去访问安全资源时会被安全扩展组件认为是非法访问的，于是就禁止其访问安全资源，因此该CPU访问请求的返回结果要么是访问失败，要么就是返回无效结果

ARM核状态	cpu请求	安装状态读写标志位
非安全状态	非安全请求	1
安全状态	安全请求	0

以上就是TrustZone技术能实现对系统资源硬件级别保护和隔离的根本原因。

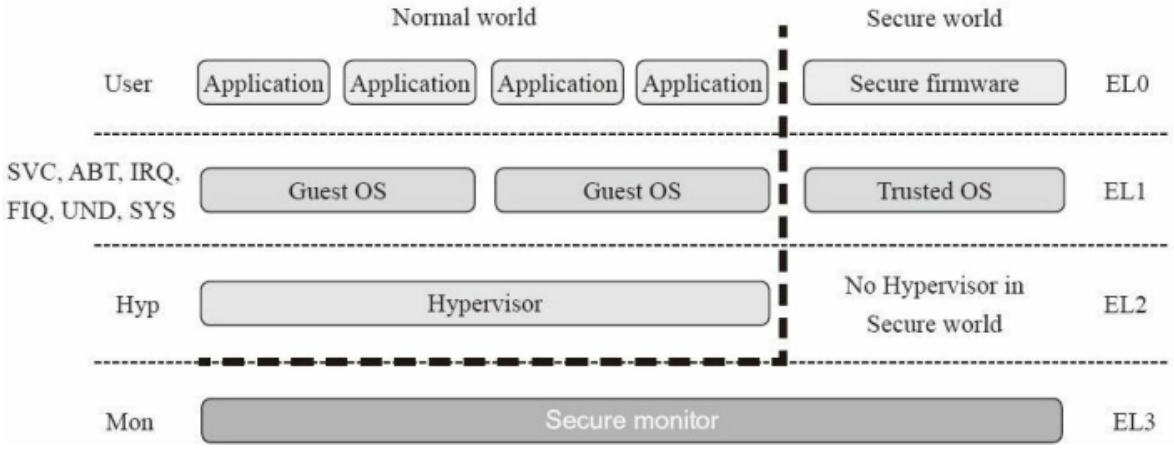
1.3.3 ARMV7 架构的TrustZone技术



- 正常世界态=====>非安全世界=======>REE环境
- 安全世界态=====>安全世界=======>TEE环境
- 监控模式　用来 切换两种世界的状态，smc指令、

  该图总结就是：
- 系统在REE侧或者TEE侧运行时，通过安全监控模式调用smc指令进入monito模式
- 通过判定系统化SCR寄存器中对应的值来确定请求来源以及发送目标

1.3.4 ARMv8架构的TrustZone



ARMV8中用EL(execution Level)来定义arm核的运行等级。

EL0：运行应用程序

EL1：运行操作系统

EL2: 运行虚拟机、监视器

EL3: 运行 安全管理

EL0 ， EL1 ,EL2 每个等级都有安全态，非安全态

EL3权限最高

ARM处理器的7种工作模式

- 用户模式（USR）:正常程序执行模式，不能直接切换到其他模式
- 系统模式（SYS）: 运行操作系统的特权任务，与用户模式类似，但具有可以直接切换到其他模式等特权
- 快中断模式（FIQ）： 支持高速数据传输及通道处理，FIQ异常响应时进入此模式。
- 中断模式（IRQ）： 用于通用中断处理，IRQ异常响应时进入此模式
- 管理模式（SVC）： 操作系统保护模式，系统复位和软件中断响应时进入此模式。由系统调用执行软中断SWI命令触发
- 中止模式（ABT）： 用于支持虚拟内存或存储器保护
- 未定义模式（UND）： 支持硬件协处理器的软件仿真，未定义指令异常响应时进入此模式。

处理器工作模式	**特权模式**	**异常模式**	**说明**
用户（user）模式			用户程序运行模式
系统（system）模式	该组模式下可以任意访问系统资源		运行特权级的操作系统任务
一般中断（IRQ）模式	该组模式下可以任意访问系统资源	通常由系统异常状态切换进该组模式	普通中断模式
快速中断（FIQ）模式	该组模式下可以任意访问系统资源	通常由系统异常状态切换进该组模式	快速中断模式
管理（supervisor）模式	该组模式下可以任意访问系统资源	通常由系统异常状态切换进该组模式	提供操作系统使用的一种保护模式，swi命令状态
中止（abort）模式	该组模式下可以任意访问系统资源	通常由系统异常状态切换进该组模式	虚拟内存管理和内存数据访问保护
未定义指令终止（undefined）模式	该组模式下可以任意访问系统资源	通常由系统异常状态切换进该组模式	支持通过软件仿真硬件的协处理

ARMv7和ARMv8架构下 各模式对应关系

运行模式	**ARMv7 特权等级**	**ARMv8 特权等级**
USR	PL0	EL0
SVC/ABT/IRQ/FIQ/UND/SYS	PL1	EL1
Hyp	Hyp	EL2
Monitor	Mon	EL3

ARMv8架构同样也是使用安全监控模式调用使处理器进入EL3运行等级。

要注意的是安全监控模式调用（secure monitor）是运行在EL3中的。

EL3中的代码负责处理器安全状态和非安全状态的切换

1.4 ARM可信固件

ARM 可信固件（ARM Trusted Firmware ,ATF）是ARM官方提供的底层固件。

这个固件统一了ARM底层接口标准，如电源状态控制接口，安全启动需求，安全世界状态与正常世界状态切换的安全监控模式调用操作等。

ATF的源代码共分为bl1，bl2，bl31，bl32，bl33。

bl1，bl2，bl31 部分属于固定的固件。

bl32 用来加载TEE OS镜像。

bl33 用来REE侧的镜像。（linux内核镜像）

ATF主要完成的功能如下：

- 初始化安全世界状态运行环境，异常向量，控制寄存器，中断控制器，配置平台的中断。
- 初始化ARM通用中断控制器
- 执行ARM系统ip的标准初始化操作，以及安全拓展组件的基本配置。
- 安全监控模式调用请求的逻辑处理代码
- 实现可信板级引导功能，对引导过程中加载的镜像文件进行电子签名检查
- 支持自有固件的引导。

ATF与TEE的关系

ATF是ARM的底层固件，是开源的。

其主要完成系统中bootloader,linux内核，TEE os的加载和启动，以及正常世界和安全世界的切换。

ATF将整个启动过程划分成不同的启动阶段，由BLX表示

在ARMv8架构中ARM提 供了ARM可信固件（ATF）。Bootloader、Linux内 核、TEE OS的启动都由ATF来加载和引导。对于 ARMv8，Bootloader、Linux内核和TEE OS镜像文 件的验签工作都是在ATF中完成的。

```
安全世界状态和正常世界状态 之间的切换是由bl31来完成
```

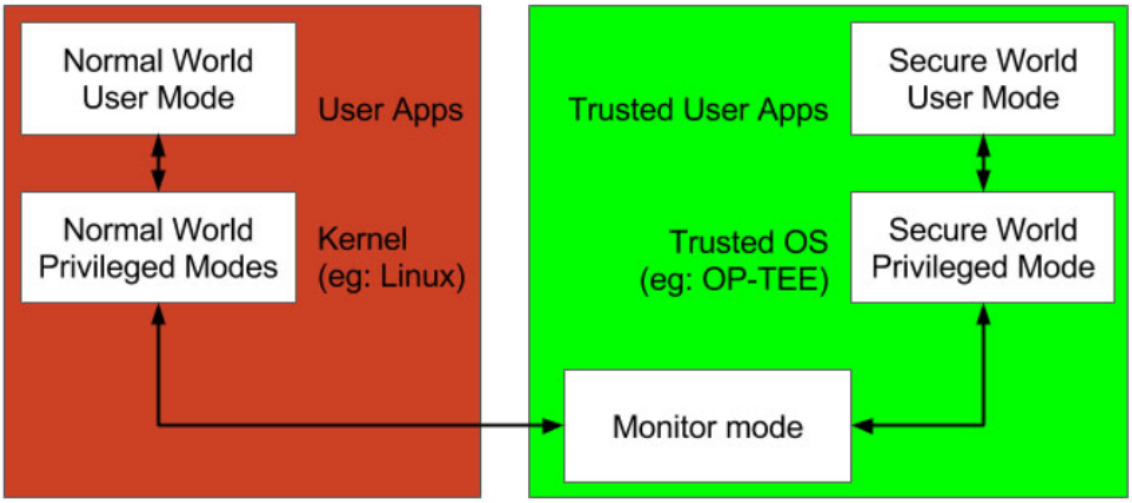
TEE os的加载是有ATF的bl32完成

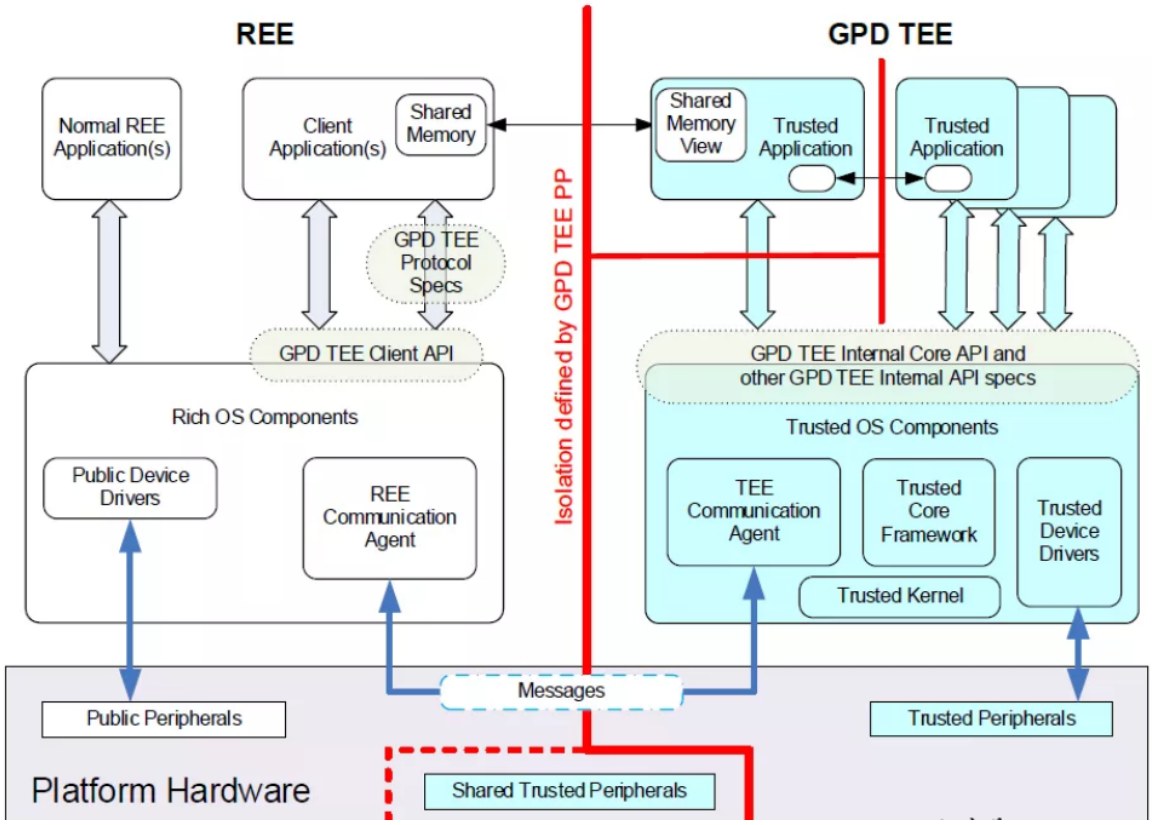
linux内核的加载是由ATF的bl33完成

其中涉及到一系列的安全启动镜像验签等内容。

二.TEE架构简析

2.1 框架简图





整个软件架构分为REE和TEE两部分，通过Monitor Mode进行安全和非安全状态切换。REE部分是指normal world，在REE中运行的系统和应用分别是Rich OS和CA。而TEE部分是指secure world，分别对应Trusted OS和TA

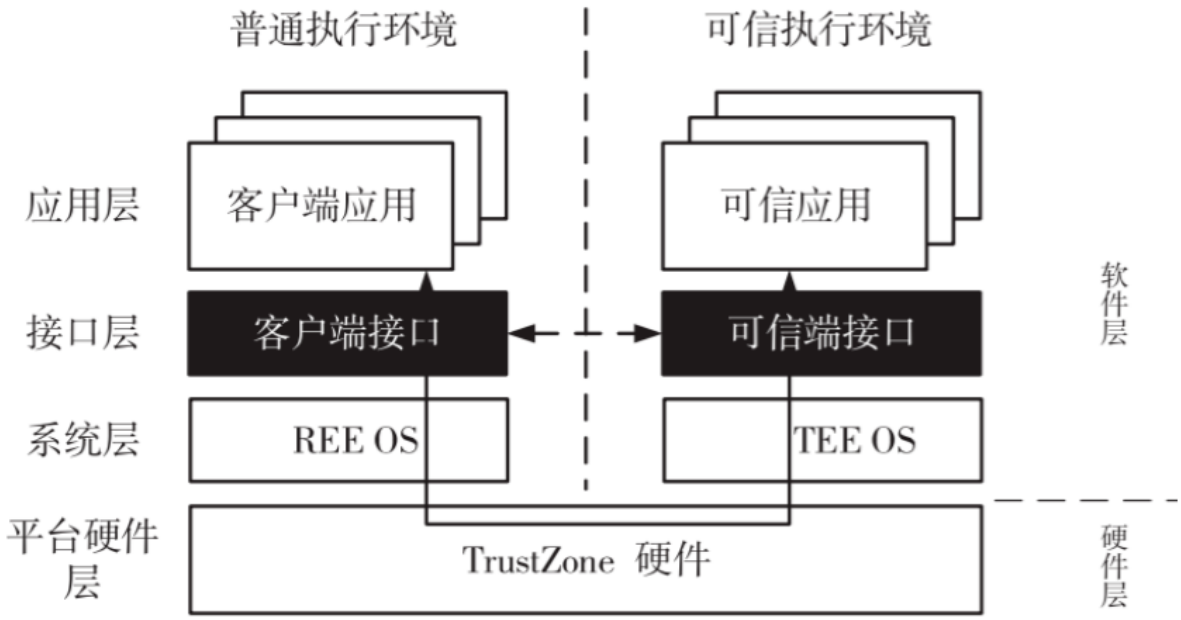
REE中的系统结构：

- CA（Client APP）对应一些上层应用，比如指纹采集、支付应用等，通过调用TEE Client API实现与TEE环境的交互。
- REE Communication Agent为TA和CA之间的消息传递提供了REE支持
- TEE Client API是REE中的TEE驱动程序提供给外部的接口，可以使运行在REE中的CA能够与运行在TEE中的TA交换数据。

TEE中的系统结构：

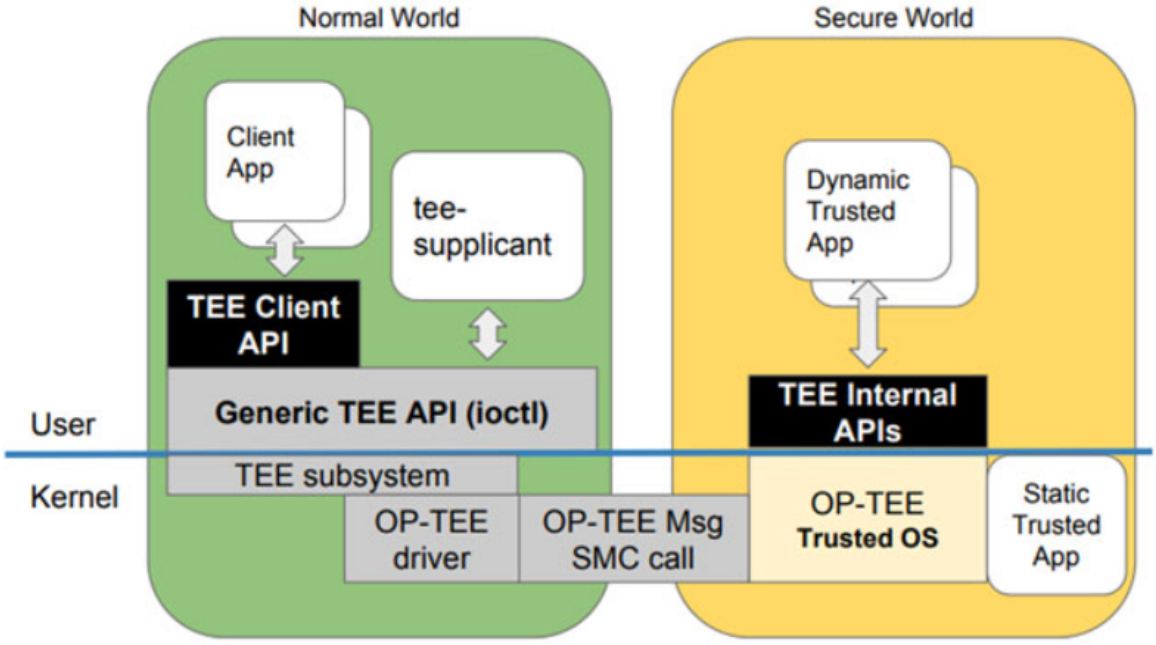
- TA（Trusted Application）是TEE中完成特定功能的应用。由于TEE中完成计算因此具有较高的安全性。每一个TA在REE中有一个或者多个对应的CA，在REE环境中可以通过调用CA的接口，将信息传送到TEE环境中执行TA，完成对应功能然后返回计算结果。

- TEE Communication Agent是可信操作系统的特殊组成部分，它与REE Communication Agent一起工作，使TA与CA之间安全地传输消息。
- TEE Internal Core API是TEE操作系统提供给TA调用的内部接口，包括密码学算法，内存管理等功能。
- Trusted Device Drivers可信设备驱动程序，为专用于TEE的可信外设提供通信接口。
- Shared Memory是一块只有CA和TA可以访问的一块安全内存，CA和TA通过共享内存来快速有效传输指令和数据



Tee所能访问的软硬件资源是与Richos 分离的。TEE提供了授权安全软件（可信应用，TA）的安全执行环境，同时也保护TA的资源和数据的保密性，完整性和访问权限。为了保证TEE本身的可信根，TEE在安全启动过程中是要通过验证并且与Rich OS隔离的。在TEE中，每个TA是相互独立的，而且不能在未授权的情况下不能互相访问

2.2 CA与TA的通信



TA部分

TA分为static TA和Dynamic TA两种。

静态TA运行在内核模式中，动态TA运行在用户模式中，一般存储在文件系统中，通过TEE-Supplicant被OP_tee加载。

CA与TA交互流程如下：CA首先调用TEE Client API触发系统调用，进入REE的操作系统内核态，根据CA调用的参数找到对应的REE驱动程序，REE驱动程序通过调用SMC汇编指令进入Monitor模式，并将处理器切换到安全内核状态，进入安全模式。切换进入TEE以后，CA的服务请求通过总线传到TEE侧，然后TEE OS通过TEE Internal API调用对应的TA，最后TA运行结束后将运行结果和数据返回给CA，执行完以后回到TEE内核态调用SMC汇编指令进入Monitor切回REE环境。

可信应用的设计分为两个部分：客户端应用（CA）和可信端应用(TA)。CA和TA通过uuid进行识别，只有使用相同的uuid，双方才能实现交互。

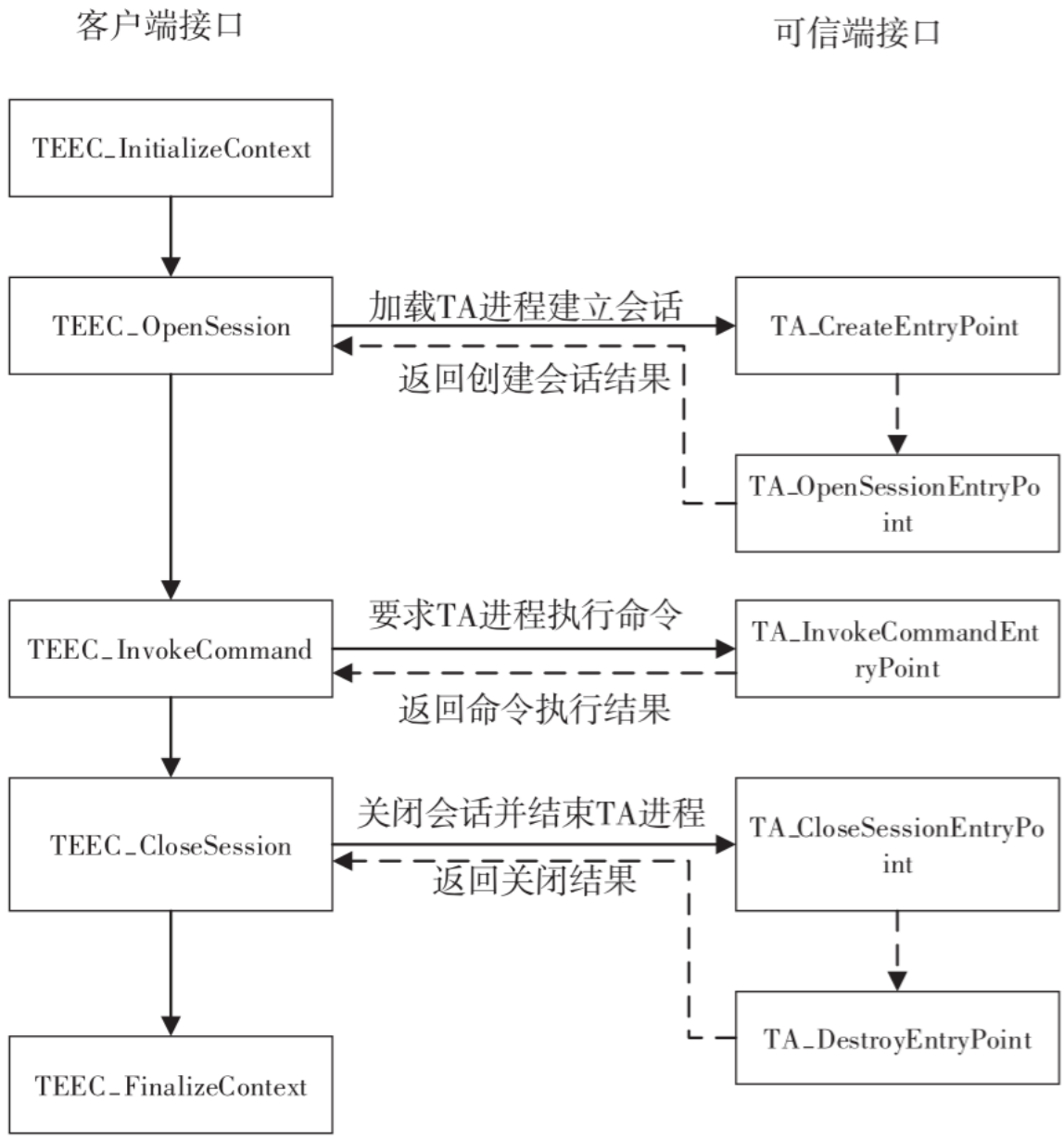


图5 客户端应用和可信端应用的端口交互过程

CA部分函数:

TEEC_InitializeContext： 建立context，提供CA连接TA的方式

TEEC_OpenSession：根据UUID建立CA呼叫TA的通道，为连接CA-TA的起始点。（可能会加载TA）

TEEC_InvokeCommand：传送指令与需要的参数给TA

TEEC_CloseSession：关闭CA-TA的通道

TEEC_FinalizeContext：移除建立好的上下文。

TA部分函数：

TA_CreateEntryPoint：建立进入点，让TA可以被呼叫

TA_DestroyEntryPoint：移除进入点，结束TA的功能

TA_OpenSessionEntryPoint：建立CA呼叫TA的通道，为连接CA-TA的起始点

TA_CloseSessionEntryPoint：关闭CA-TA的通道

TA_InvokeCommandEntryPoint：接收CA传送的指令，并在这边执行

四.参考链接

https://deepinout.com/android-system-analysis/android-security-related/easy-to-understand-tee.html

https://www.jianshu.com/p/c238bfea3e46