PHYTIUM 飞腾

腾锐 D2000 安全 CPU

可信交互机制

v1.0

飞腾信息技术有限公司

2021 年 6 月

更新记录

版本号	发布部门	作者	发布日期	备　注
1.00	飞腾通用软件部	邓强	2021-04-14	初稿

注意

飞腾信息技术有限公司对其发行的或与合作公司共同发行的包括但不限于产品的全部内容及材料所拥有版权等知识产权，受法律保护。非经本公司书面许可，任何单位及个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

免责声明

我们仅仅提供技术上的咨询，对利用文档搭建环境所从事的研发活动没有技术支持责任，对相关研发成果没有连带责任。

目录

1 概述

腾锐 D2000 安全 CPU 支持两种执行环境：可信执行环境（TEE）和通用执行环境（REE）。可信执行环境和通用执行环境间可以通过 TrustZone 和 TPCM 两种方式进行交互。

在 TrustZone 方式下，处理核可以通过 SMC 指令的方式，进行可信执行环境和通用执行环境之间的切换。在 TPCM 方式下，处理核分为可信核和计算核，可信核上运行的是可信执行环境，计算核上运行的是通用执行环境，两个环境使用中断方式交互。

2 TrustZone

TrustZone 是 ARM 提出的一套可信框架。本文档只说明 TrustZone 方式下，可信执行环境和通用执行环境之间的交互机制。

2.1 SMC 指令

SMC 指令交互主要指 CPU 调用 SMC 指令在可信执行环境和通用执行环境间切换，并通过 CPU 通用寄存器传递参数。例如通用操作系将安全功能参数保存在 CPU 通用寄存器中，然后调用 SMC 指令进到可信操作系统中，并根据安全功能参数执行相关安全服务。功能参数是一个 32bit 的数据类型，保存在 CPU 通用寄存器 R0 中，其定义必须符合 ARM TrustZone 规范。

表 2.1 SMC 功能参数 ID

位域	屏蔽位	功能描述
31	0x80000000	0：表示标准调用 1：表示快速调用；执行过程不能被中断
30	0x40000000	0：表示 SMC32 调用 1：表示 SMC64 调用
29:24	0x3F000000	0-47：预留 48-49：可信应用程序使用的 ID 范围 50-63：可信操作系统使用的 ID 范围
23-16	0x00FF0000	当 bit31 为 1 时，该域必须位 0 当 bit31 为 0 时，该域的值未定义
15-0	0x0000FFFF	用户自定义，每个值对应一个功能

2.2 SMC 调用示例

以 OP-TEE+Linux 环境为例，针对 OP-TEE 的 Linux 内核驱动出现在内核版本 4.12 及其以上的版本。

当驱动启动时，需要通过 SMC 指令，切换到 TEE 环境，并获取 OP-TEE OS 的版本信息。

2.2.1 示例代码环境

REE 软件：Linux4.19.88

TEE 软件：OP-TEE OS 3.2.0-rc1

2.2.2 代码运行流程

Linux 的相关代码在 drivers/tee/optee/core.c 中，内核驱动获取 OP-TEE 版本信息的函数是 optee_msg_get_os_revision，该函数被驱动的入口函数 optee_probe 调用。

```
static void optee_msg_get_os_revision(optee_invoke_fn *invoke_fn)
{
    union {
        struct arm_smccc_res smccc;
        struct optee_smc_call_get_os_revision_result result;
    } res = {
        .result = {
            .build_id = 0
        }
    };

    invoke_fn(OPTEE_SMC_CALL_GET_OS_REVISION, 0, 0, 0, 0, 0, 0, 0,
          &res.smccc);

    if (res.result.build_id)
        pr_info("revision %lu.%lu (%08lx)", res.result.major,
            res.result.minor, res.result.build_id);
    else
        pr_info("revision %lu.%lu", res.result.major, res.result.minor);
} « end optee_msg_get_os_revision »
```

函数 invoke_fn 在 Linux 设备树相关配置中已经被定义为 SMC 指令的调用，具体参考函数 optee_probe 中的 get_invoke_func 函数。

SMC 指令最多支持 8 个参数，读取 OP-TEE 版本信息只需要第一个参数，具体参数定义参考本文档的表 1，这里的值为 OPTEE_SMC_CALL_GET_OS_REVISION。

OPTEE_SMC_CALL_GET_OS_REVISION 通过一系列的宏定义封装：

```
ARM_SMCCC_CALL_VAL(ARM_SMCCC_FAST_CALL, ARM_SMCCC_SMC_32, \
                   ARM_SMCCC_OWNER_TRUSTED_OS, (func_num))
```

最终值分为 4 个位域:

ARM_SMCCC_FAST_CALL	bit31 为 1，表示快速调用
ARM_SMCCC_SMC_32	bit30 为 0，表示 SMC32 调用
ARM_SMCCC_OWNER_TRUSTED_OS	bit29:24 为 50，表示 OP-TEE OS 使用的 ID

func_num 为 0x0001，参数的低 16 位，表示用户自定义的功能号，Linux 定义的 0x0001 表示读取 OP-TEE OS 的版本号。

函数 invoke_fn 执行后，REE 环境的 Linux 调用 SMC 指令，切换到 TEE 环境下，下条指令要等待 TEE 环境的代码执行完毕后，再通过 SMC 指令切回 REE 环境后执行。版本信息保存在函数 invoke_fn 的最后一个参数 res.smccc 中。

OP-TEE OS 相关处理代码的总入口在 optee_os/core/arch/arm/plat-phytium/main.c，通过代码中 handlers 结构可以看到 SMC 快速调用的处理函数是 tee_entry_fast，该函数定义在 optee_os/core/arch/arm/tee/entry_fast.c 中。

函数 tee_entry_fast 是一个 switch 语句，根据 REE 传递过来的第一个参数值不同进行不同的操作。OPTEE_SMC_CALL_GET_OS_REVISION 定义的操作也在操作列表中，其中宏的定义和 Linux 中同一个宏的定义一模一样。这样确保当 SMC 指令的第一个参数为该宏定义的值时，OP-TEE OS 将调用 tee_entry_get_os_revision 函数处理读取版本信息的请求。

```
void __weak tee_entry_get_os_revision(struct thread_smc_args *args)
{
    args->a0 = CFG_OPTEE_REVISION_MAJOR;
    args->a1 = CFG_OPTEE_REVISION_MINOR;
    args->a2 = TEE_IMPL_GIT_SHA1;
}
```

函数 tee_entry_get_os_revision 将相关信息保存在 SMC 指令的前三个参数中，操作完成后，OP-TEE OS 调用 SMC 指令切换回 REE 环境的 Linux，SMC 指令的参数保存在 res.smccc 中，Linux 可以打印出版本信息，完成一次 REE 和 TEE 之间的交互。

3 TPCM

上一章的 Trustzone 交互模式是 ARM 标准的交互模式，另外，本文档还介绍 D2000 芯片可以支持的另一种可信中断交互机制 TPCM 模式，即可信核与计算核可以通过中断和 share memory 进行交互。

该模式只适用于多核系统，需要将至少一个核单独运行在安全世界，而其他的核运行在非安全世界，

两个世界之间通过中断和共享内存的方式进行交互。

该文档所有内容的编写都是基于以下前提条件:

计算核，主要用来运行通用固件和 linux 软件，处于 normal world;

可信核，主要用来运行 TPCM 软件，处于 secure world;

Memory 的分布如下:

0xfc000000 - 0x100000000 // PBF

0xf0000000 - 0xfa000000 //TPCM

0xfa000000 - 0xfc000000 // share memory

3.1 示例代码的使用

3.1.1 固件打包步骤

进入 image_fix 目录;

将 bl32_new.bin 连接到 optee(trust os)的 bin 文件;

将 bl33_new.bin 连接到 第三方固件厂商 uefi;

打包环境如果为 X86 机器，则使用 fiptool_86 覆盖 my_scripts/fiptool

在 image_fix 目录下执行 my_scripts/image-fix.sh bl32;

将新生成的 fip-all.bin 烧到 flash 中。

3.1.2 Linux 代码修改

将 linux_for_tpcm.patch 合入到内核源码中。

3.1.3 测试命令

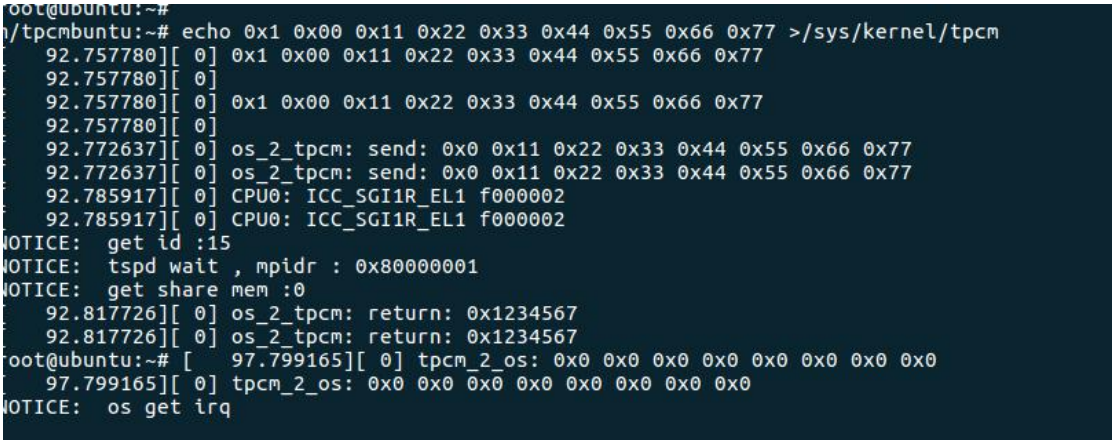
echo 0x1 0x00 0x11 0x22 0x33 0x44 0x55 0x66 0x77 >/sys/kernel/tpcm/tpcm

echo 的参数说明:

第一个参数为可信核 id，此处为 1 核（0x1）;

发送内容: 0x00 0x11 0x22 0x33 0x44 0x55 0x66 0x77 为 linux 通过 share memory 发送给 TPCM 的数据。每个数据为 64b，最多支持 8 个数据。

3.1.4 测试结果 log



3.1.5 测试方法的实现

Linux 向 TPCM 发送命令的流程：

Linux 将命令参数填充到 share memory 0xfa000000；

Linux 发送 SGI 15 给 TPCM 可信核；

TPCM 的中断响应函数会读取和打印 share memory 0xfa000000 中的内容；

TPCM 填充 0x1234567 到地址 0x fa000000；

Linux 的查询到 share memory 0xfa000000 处的值为 0x1234567，则表示命令发送成功；

TPCM 向 Linux 发送命令的流程：

当 TPCM 的可信核接受并处理 SGI 15 中断后，会在 5 秒之后将接受到的命令内容填充到 share memory 0xfa000800 处；

TPCM 发送 SGI 7 给 Linux；

Linux 的中断处理函数会读取和打印 share memory 0xfa000800 中的内容；

Linux 填充 0x1234567 到地址 0xfa000800；

TPCM 查询到 share memory 地址 0xfa000800 处的值为 0x1234567，则表示命令发送成功；

3.2 中断的配置和触发

中断示例约定采用固定的 SGI 中断号进行交互。TPCM 向计算核发送 7 号中断，计算核向 TPCM 发送 15 号中断。

3.2.1 中断配置

可信核 TPCM(S-EL1)下的中断配置

配置 GICR_NSACR 寄存器的 Bit31-30 为 b'10，允许非安全态 CPU 发送 15 号安全中断；

配置 DAIF 的 I 位为 0，表示不屏蔽 IRQ 中断。

计算核 EL2 固件下的中断配置

配置 HCR_EL2 寄存器的 IMO 位为 1，目的是让中断路由到 EL2；引导 OS 前，必须将该位清 0。

配置 DAIF 的 I 位为 0，表示不屏蔽 IRQ 中断；

计算核 EL1 Kernel 的中断配置

采用 Kernel 缺省配置即可，无需额外定制。

3.2.2 中断触发

可信核向计算核发送中断：写 SGI 发送寄存器 ICC_ASGI1R_EL1；

2000/4CPU 共 4 个核，分成 2 个 cluster，每个 cluster 包含 2 个核。即 Core0-1 属于 cluster 0，Core 2-3 属于 cluster 1。如下图所示，Aff1 为目标 CPU 的 Cluster 号，TargetList 为目标 CPU 在 Cluster 内的 ID，即 0-1。INTID 为 SGI 中断号。

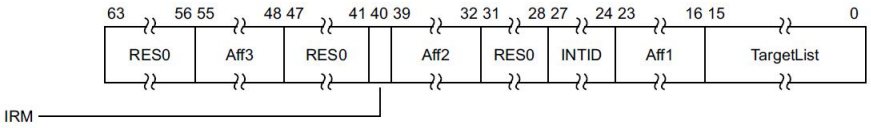


图 3.1 ICC_ASGI1R_EL1

计算核向可信核发送中断：写 SGI 发送寄存器 ICC_SGI1R_EL1，寄存器格式参见 ICC_ASGI1R_EL1。

4 参考文献

[1] ARM,《SMC CALLING CONVENTION System Software on ARM Platforms》

[2] ARM, 《Generic Interrupt Controller Architecture Specification GIC architecture version 3.0 and version 4.0》, Issue C.

[3] ARM, 《ARM Architecture Reference Manual ARMv8, for ARMv8-A architecture profile》