PHYTIUM 飞腾

腾锐 D2000 安全 CPU

QuickStart

v1.1

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	2021-06-11	初稿
1.10	飞腾通用软件部	邓强	2021-09-15	根据软件新框架修改，增加扩大 TEE 系统内存方法的说明

目录

1 目录说明.....1
2 环境搭建.....1
2.1 软硬件环境要求.....1
2.2 编译源码.....1
2.2.1 编译脚本.....1
2.2.2 编译 pbtee.....3
2.2.3 编译 Linux 内核.....3
2.3 固件打包.....4
2.4 演示环境.....4
2.4.1 演示 helloworld 应用程序.....4
2.5 经验分享.....5
3 调试方法.....6
3.1 一种基于 U 盘的轻量化调试方法.....6
3.1.1 概述.....6
3.1.2 准备工作.....6
3.1.3 基于 uboot 的调试步骤.....6
3.1.4 基于 uefi 的调试步骤.....9
3.2 一种扩大 TEE 系统内存的方法.....9
3.2.1 功能背景.....9
3.2.2 原理说明.....10
3.2.3 配置要求.....10
3.2.4 注意事项.....11
4 SDK 文档.....11

1 目录说明

表 1.1 目录列表

目录名	子目录	目录说明
doc		存放 SDK 所有说明文档
SecureOS	debug	调试用有关脚本和工具
	patch	SDK 使用到的所有补丁文件
	source	SDK 提供的所有源码包
固件打包		固件打包工具和 PBF 发布版本

2 环境搭建

2.1 软硬件环境要求

搭建自己熟悉的编译环境，无论是 ARM 的编译环境，还是 x86 的交叉编译环境都有成功的范例。

推荐安装 aarch64-linux-gnu-gcc，版本高于或等于 7.3.0 的 ARM 编译环境，用于编译 pbtee 和 REE OS，REE OS 采用 Linux kernel，内核版本要求高于 4.12.0。

编译 pbtee 过程中需要 Python3 的库支持。可以根据编译错误的提示逐步增加相关的组件支持。根据经验，注意一下几个库的安装和支持:

pycrypto

pyelftools

Pycryptodomex

硬件环境要求是基于腾锐 D2000 安全 CPU 的开发环境。可以请相关支持人员确认。

2.2 编译源码

2.2.1 编译脚本

软件编译时使用的自动化脚本文件，通过多次的优化修改，目前 PBTEE v3.0 已经形成了一套方便开发者和客户使用的脚本系统。

1) 主目录脚本 auto-clean-all

可以在主目录下直接运行./auto-clean-all，常用于打包 pbtee 主目录代码前的清理冗余文件，或者一次完整的 distclean。

可以带参数 os，client，ca 来分别单独清理子目录。

2) 主目录脚本 auto-compile-all

可以在主目录下直接运行./auto-compile-all，如果不带参数，会显示帮助信息，请根据帮助信息增加 target 信息，然后可以一次性编译出所有需要的东西，包括 TEE、REE 侧的 target 文件，存放在根目录的 out 目录下。

3) tee_os 目录脚本 build_tee_os

可以在 tee_os 目录下直接运行./build_tee_os，如果不带参数，会显示帮助信息，请根据帮助信息增加参数。该脚本主要负责 TEE OS 内核的编译，支持 D2000，FT2004 等多种不同 CPU 的环境，具体请参考脚本自带的帮助信息。

4) ree_client 目录脚本 build_ree_client

可以在 ree_client 目录下直接运行./build_ree_client，没有参数。该脚本主要负责 REE 支持环境的编译。

5) ree_ca 目录脚本 build_ree_ca

目前 PBTEE v3.0 版本下，ree_ca 目录下有 5 个参考 CA 例程，分别是 hello_world、scto_selftest、scto、efuse_rw 和 crypto_services。该脚本主要负责这 5 个参考 CA 例程的编译，该脚本的运行有个前提条件是必须先运行 ree_client 目录下的脚本 build_ree_client，CA 例程的编译需要用到支持环境的库链接。

该脚本可以加参数单独编译某一个例程，比如./build_ree_ca scto 表示只编译 scto 这个 CA 例程。如果脚本不加参数运行，表示编译全部 CA 例程。

6) 其他 CA 脚本 build_hello_world、build_scto 和 build_crypte_services

这些脚本应用于需要动态加载 TA 到 TEE 侧环境的 CA 例程中。可以单独执行，也被上一级脚本 build_ree_ca 间接调用。

这类脚本没有参数输入。

2.2.2 编译 pbtee

以 PBTEE v3.0 为例，从 SDK 中获取 pbtee-v3.0-release.tgz 文件，解压后是一个 tar 包文件，文件名的后六位是 MD5 值，pbtee 的 v3.0 版本的 MD5 值后六位是 901590，不会有任何其他不同的值，可以进行验证是否被篡改。

解包后进入 pbtee 主目录，介于上一节比较完善的脚本系统，编译 pbtee 可以使用 auto-compile-all 进行一键编译。

即在主目录下运行 auto-compile-all 脚本，本文档考虑腾锐 D2000 安全 CPU 运行环境，所以只考虑 d2000r 和 d2000d 两种配置，r 表示 release，d 表示 debug，用于常见的两种编译情况。

例如，./auto-compile-all d2000d，运行后，会自动在主目录生成一个 out 目录，out 目录中的 data-xxxx.tgz 和 tee-phytium-xxxx.bin 文件分别就是 REE 侧的支持包和 TEE 侧的映像。

如果编译过程中出现错误，可以采用错误出现目录中的编译脚本先进行手动编译，等待问题解决后，可以再用 auto-clean-all 清理下所有冗余文件，再用 auto-compile-all 整体编译。

2.2.3 编译 Linux 内核

1 获取 Linux 内核补丁，补丁在 SDK 子目录 SecureOS 的"patch"目录下，文件名为 linux_for_trustzone.patch。

2 将补丁 patch 到 Linux 内核中，再编译 Linux 内核，采用 uboot 作为引导程序，则 make uImage，编译完成后得到 uImage 文件，如果采用 uefi 为引导程序则直接 make，得到 Image.gz。

注意：如果内核是飞腾内部发布的参考内核，则不需要打补丁，该补丁默认已经打上。

3 补丁支持 ACPI 和设备树两种方式启动对 optee 的支持。如果使用设备树方式启动 Linux，设备树文件中必须添加的一行说明，如下：

```
firmware {
        optee{
                compatible = "linaro,optee-tz";
                method = "smc";
        };
};
```

ACPI 表也是类似添加一行配置。

注意：Linux 版本必须采用 4.12 以上的版本。

如果 patch 打补丁有错误提示，应该是内核版本不一致的问题，请阅读 patch 文件内容，理解方法，并根据实际情况灵活修改。某些内核版本默认已经打上该补丁。

输出结果：

文件	说明
uImage 或 Image.gz 文件	内核映像

2.3 固件打包

固件打包的详细流程请参考《腾锐 D2000 安全 CPU-固件打包工具说明》。

上面生成的 tee-phytium-xxxx.bin 就对应该文档中的 bl32 固件。

2.4 演示环境

2.4.1 演示 helloworld 应用程序

结合前面的准备工作，演示 Trustzone 模式下的 helloworld 需要具备以下文件：

文件	说明
uImage 或 Image.gz 文件	支持 Trustzone 模式的 Linux 内核映像
fip-all.bin	打包工具输出的最终固件烧录文件
data-xxxx.tgz 包	REE 端和 TEE 交互的 client 端文件包

演示步骤：

1 先烧写 fip-all.bin 到硬件环境中。

2 用 uImage 或 Image.gz 替换之前的 Linux 内核。启动 ubuntu 环境或其他类 Linux 环境。

3 将 data-xxxx.tgz.tgz 拷贝到启动环境后桌面环境中，并解压包内容到根目录下。

4 进入/data 目录，先执行 source ./env.source 设置环境，再执行 hello_world 程序，如果程序运行无错误，并有以下类似显示，就可以了：

```
D/LD:  ldelf:169 ELF (8aaaf200-2450-11e4-abe2-0002a5d5c51b) a
F/TC:? 0 plat_prng_add_jitter_entropy:72 0x01
Invoking TA to increment 42
TA iD/TC:? 0 tee_ta_close_session:515 csess 0xfc0b4b30 id 1
ncremented value to 43
D/TC:? 0 tee_ta_close_session:533 Destroy session
```

演示说明：

这个交互是 REE 环境下发了一个等于 42 的变量给了 TEE 环境，TEE 环境收到后，进行了++操作，然返回给 REE 环境。具体可以参考源代码 hello_world.c（REE 环境运行）和 hello_world_ta.c（TEE 环境运行），代

码在 pbtee/ree_ca/hello_world 目录下。

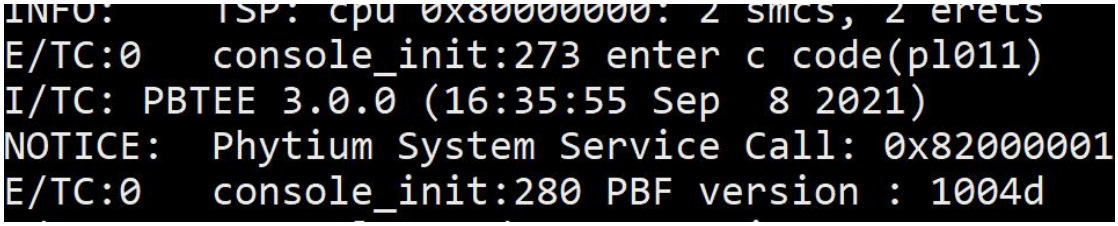
用户可以参考 helloworld 的模式开发客户自己的 CA 和 TA 程序。也可以参考 ree_ca 目录下提供的其他开发例子。

2.5 经验分享

如果整个操作不那么顺利，可以按以下一些经验一步步来。

1 先确保 pbtee 启动，打包固件正确。

先烧录 fip-all.bin 到硬件环境直接启动，看启动 log，要出现类似以下 log：



内容可能不一样没关系，但必须有类似“E/TC”, “I/TC”这样标志性的打印 title 即可。

如果没有，说明 TEE 环境没有正常启动，可以用别人已经验证 OK 的 fip-all.bin 烧录下，确认硬件环境没问题。

如果好的 fip-all.bin 可以，自己的不行，则检查打包步骤（比如是不是 pbf 没更新，打包命令没加 bl32 等）和 pbtee/tee_os 中的编译（是不是编译有警告或错误，bin 文件没有正确拷贝）两个方向。

2 设备文件的检查

在 Linux 操作系统启动后的文件系统中，必须要生成

/dev/tee0

/dev/teepriv0

两个设备文件。

如果设备文件正常生成，既可以开始演示操作，如果设备文件不存在，则检查下 Linux 的驱动补丁是否正确打上，设备树或 ACPI 表是否加入相关配置。可以用已经验证 OK 的 uImage 或 Image.gz 试试，排除其他硬件问题干扰。

3 调试方法

3.1 一种基于 U 盘的轻量化调试方法

3.1.1 概述

按照前面的内容，完成一个 helloworld 和一个成熟的 TEE 调试是没有问题的。但当用户的 TEE 映像并没有稳定，需要经常修改时，反复打包和烧片将会带来很繁重，且意义不大的工作量。而有的用户环境是不能烧片，也不能修改硬盘中桌面 OS 和 Kernel 的，这时候就结合 SDK 文档《腾锐 D2000 安全 CPU-Tboot 使用说明》上面提到的 Tboot 辅助工具，推荐使用一种基于 U 盘的轻量化调试方法。

3.1.2 准备工作

准备 checklist 如下：

1 硬件方面准备 U 盘一个。

2 已经编译好，等待调试的 pbtee 的映像文件。本文档以 tee-phytium-d2000.bin 为例。

3 打包好支持 Tboot 功能的烧录文件。本文档以 fip-all-tboot.bin 为例，该烧录文件有可能有两种，一种是 bl33 为 uboot 的版本，一种是 bl33 为 uefi 的版本。

4 参考本文档相应章节，编译好正确的内核映像，本文档编译后命名为 uImage-d2000-tztp 为例。

5 轻量化的 ramdisk 文件系统。本文档以 ramfs.img 为例。

6 有条件的准备：如果 bl33 是 uefi 的情况下，额外需要准备一个 grub 工具用于启动。

3.1.3 基于 uboot 的调试步骤

如果用户的 bl33 是 uboot 映像，请参考本节的步骤，本文档使用的 uboot 是飞腾版本的 uboot，提供 tboot 这样的自研命令。

步骤 1 将支持 tboot 功能的烧录文件（fip-all-tboot.bin）烧录进 D2000 的 flash 中。如果 flash 已经支持 tboot，则可以忽略该步骤。

步骤 2 启动系统进入 uboot 命令行界面，log 如下：

```
scsi scan failed
nvme scan start
get flash cmd
NOTICE:  Phytium System Service Call: 0xc200000c
WARNING: Unimplemented PHYTIUM System Service Call: 0xc200000c
cmd write is 0x0,cmd erase is 0x0,cmd pp is 0x0
Hit any key to stop autoboot:  0
D2000#
D2000#
D2000#
```

步骤 3 执行 usb start 启动 uboot 下 usb 支持。如果没有任何反应，可以考虑先执行 usb stop，再重新执行 usb start。同理，如果这时候要插拔 U 盘，也可以 usb stop 下，然后插拔 U 盘，再 usb start，log 如下：

```
D2000#usb start
D2000#
D2000#usb stop
stopping USB..
D2000#usb start
starting USB...
Bus xhci_pci: Register 8000820 NbrPorts 8
Starting the controller
USB XHCI 1.00
Bus xhci_pci: Register 8000820 NbrPorts 8
Starting the controller
USB XHCI 1.00
scanning bus xhci_pci for devices... 2 USB Device(s) found
scanning bus xhci_pci for devices... 2 USB Device(s) found
       scanning usb for storage devices... 1 Storage Device(s) found
D2000#
```

注意：腾锐 D2000 安全 CPU 并没有 USB host 设备，这里指的 USB host 都是基于 PCIe 的，所以这里 log 可以看出有 PCIe devices 找到，另外找到一个 storage 设备，这个才是 U 盘设备。

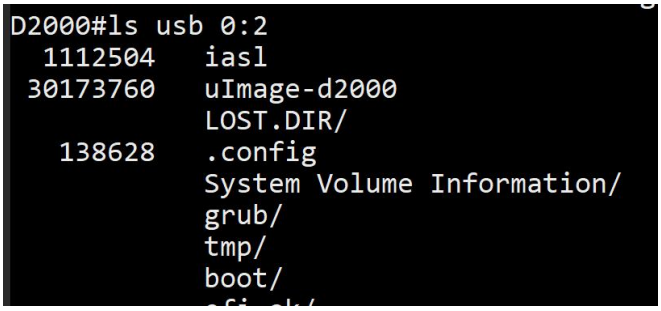
步骤 4 可以通过 usb strorge 命令查看下有没有 storage 设备，确定 U 盘的设备号和分区号，log 如下：

```
D2000#usb storage
  Device 0: Vendor: TOSHIBA Rev:  Prod: USB FLASH DRIVE
            Type: Removable Hard Disk
            Capacity: 29600.0 MB = 28.9 GB (60620800 x 512)
```

可以看到设备号是 0，记录下来。

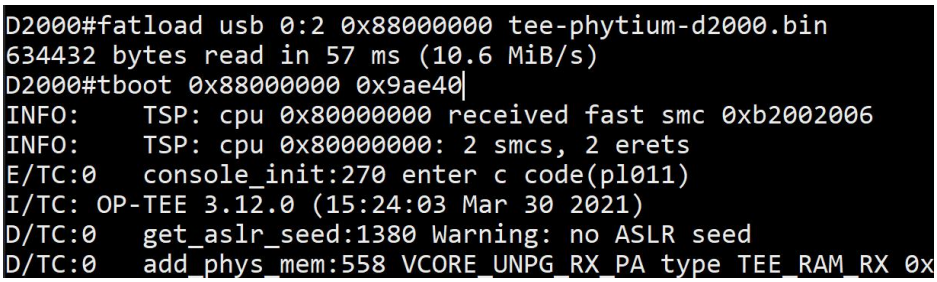
这里没有显示分区号，但可以通过查询 U 盘内容来确定分区号。

一般来说，U 盘只分一个区，分区号就是 1，为了测试，U 盘分了两个区，第一个区是 NTFS 格式，第二个区是 FAT32 格式。设备号固定是 0，可以尝试用 ls usb 读 U 盘内容的方式扫一下分区号看看，飞腾 uboot 版本目前只支持 FAT 格式的分区。



从这个 log 可以看出测试用的 U 盘，能访问的 FAT32 分区的分区号是 2。

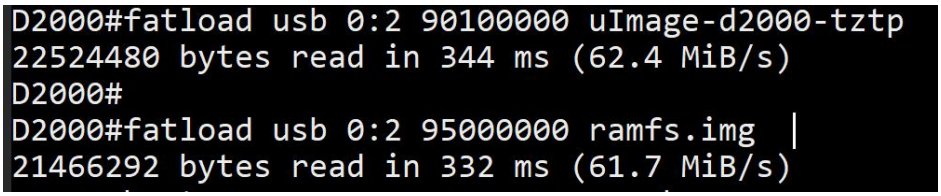
步骤 5 之前将编译好的 TEE Image（tee-phytium-d2000.bin）已经存放在 U 盘里，这时候可以通过 fatload 命令加载 Image 到内存临时区域，并运行 tboot 命令启动，log 如下：



从 log 可以看出 TEE Image 正常运行起来了，关于 tboot 的原理，请参考《腾锐 D2000 安全 CPU-Tboot 使用说明》，这里就不做详细讲解了，可以简单理解为“相当于是 ATF 的启动顺序中，bl32 延后运行了”。

注意：tboot 命令的第二个参数是 TEE Image 的 size，是 16 进制的，前一个 fatload 命令运行完的提示 size 是十进制的，实质是同一个数。uboot 命令默认参数都是 16 进制的。

步骤 6 接下来加载内核和文件系统到内存临时存放。同样也是使用 fatload 命令，log 如下：



步骤 7 使用 bootm 命令启动内核和文件系统，bootm 命令的格式如下：

bootm 内核首地址 文件系统首地址:文件系统 size（16 进制表示） FDT 首地址

```
D2000#bootm 90100000 95000000:1478cb4 17c000
## Booting kernel from Legacy Image at 90100000 ...
   Image Name:   Linux-4.19.0phytium
   Image Type:   AArch64 Linux Kernel Image (uncompressed)
   Data Size:    22524416 Bytes = 21.5 MiB
   Load Address: 80080000
   Entry Point:  80080000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 0017c000
   Booting using the fdt blob at 0x17c000
   Loading Kernel Image ... OK
```

FDT 的地址是由打包工具在打包的时候决定的。

经过以上步骤后，系统能进入到一个 Shell 环境下，用户可以在这个环境下运行自己的 CA，TA 来进行调试了。

3.1.4 基于 uefi 的调试步骤

基于 uefi 的调试相对于 uboot 的调试，手动操作较少，比较简单，本文档进行一些关键说明。

步骤 1 首先要进入到 uefi shell 界面，找到 U 盘文件系统。比如 fs0。

步骤 2 执行 ShellTbeTest.efi 脚本。该脚本存放在本 SDK 的 SecureOS/debug 目录下。该脚本可以带一个参数，参数就是 TEE 待调试的映像存放的绝对路径，没有盘符，所以该映像应该存放在和 ShellTbeTest.efi 脚本文件同一个盘符下，即都在 U 盘的同一个分区里。log 如下：

```
FS0:\newd\> ShellTbeTest.efi \tee-phytium-d2000.bin
INFO:    TSP: cpu 0x80000000 received fast smc 0xb2002006
INFO:    TSP: cpu 0x80000000: 2 smcs, 2 erets
E/TC:0   console_init:270 enter c code(pl011)
```

步骤 3 然后通过 grub 手动启动正常的启动流程即可。这里不一定要启动 U 盘中的 kernel 和 ramdisk，也可以启动硬盘中的相应文件。

基于 uefi 的调试，由于飞腾提供自动化的运行脚本 ShellTbeTest.efi，简便了所有的输入步骤。

3.2 一种扩大 TEE 系统内存的方法

3.2.1 功能背景

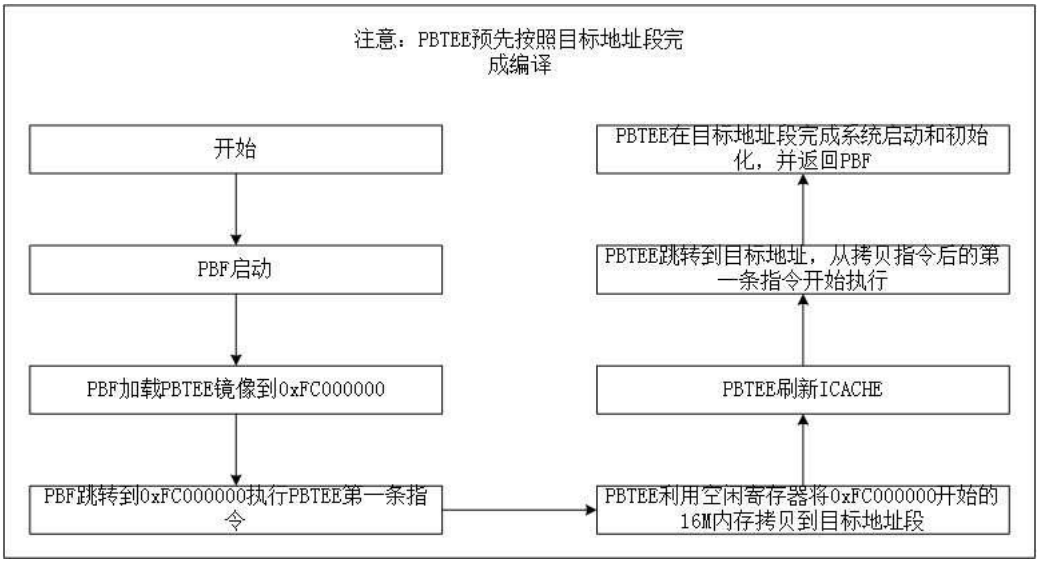
目前 PBTEE 的默认系统内存配置在 tee_os/core/arch/arm/plat-phytium/conf.mk 配置文件中定义，定义如下：

```
#
# Config for d2000
#
ifeq ($(PLATFORM_FLAVOR),d2000)
CFG_TZDRAM_START ?= 0xfc000000
CFG_TZDRAM_SIZE ?= 0x02200000
CFG_SHMEM_START ?= 0xfe200000
CFG_SHMEM_SIZE ?= 0x00c00000
```

可以看到 TEE 环境总共运行内存为 48MB，随着功能的添加，系统内存并不能满足需要，需要扩大内存支持。

3.2.2 原理说明

代码启动过程中，PBTEE 的代码段最开始一段汇编代码是一个 mini 搬移头程序，将自己整体搬移到正确的内存空间中运行。具体搬移步骤如下图所示:



3.2.3 配置要求

要实现扩大系统内存，只需要重新配置 conf.mk 文件中对应的宏，将对应的 START，SIZE 宏修改成需要的值。

另外要注意，内存不能有冲突，即不能让 REE 和 TEE 同时管理一段内存空间，这样势必产生错误结果。也就是说，pbtee 的系统内存扩大了，相应的 REE 管理的内存要配置缩小。

举例说明，客户假设希望将 TEE 系统内存扩大到（256-16）MB 空间，最上面 16MB 是 PBF 空间，不可覆盖。那么可以将 CFG_TZDRAM_START 改为 0xF0000000，CFG_TZDARM_SIZE 做相应修改，REE 侧则不可再配置这段内存。

3.2.4 注意事项

1 目前该方法无法获取 Image 具体大小，所以硬搬移 16MB 空间，如果未来 PBTEE 编译后的 Image 大于 16MB，需要进行调整。

2 搬移操作是必选项，但 16MB 数据在 DDR 中的搬移，对启动时间消耗影响甚微。

3 编译地址和加载地址不相同，会影响内部所有跳转和调用，但搬移头在程序代码段最开始位置，没有相关问题的困扰。

4 SDK 文档

表 4.1 SDK 文档列表和说明

文档名	说明
腾锐 D2000 安全 CPU-PSPA 软件编程手册	为用户在飞腾 PSPA 框架下进行软件上层开发提供相关硬件接口信息。
腾锐 D2000 安全 CPU-QuickStart	为用户在本地快速搭建编译和 helloworld 演示环境，已经常规调试方法进行了说明。
腾锐 D2000 安全 CPU-固件打包工具说明	为用户使用打包工具进行了详细说明。
腾锐 D2000 安全 CPU-Tboot 使用说明	为用户讲解了飞腾自由专利技术 Tboot 的原理。
腾锐 D2000 安全 CPU-TEE 编程接口手册	为用户提供更多的飞腾特有的 TEE 底层调用接口的说明。
腾锐 D2000 安全 CPU-可信交互机制	为用户讲解了飞腾 CPU 支持的多种可信交互机制的原理。