

了解TrustZone，读了这篇就够了



XtremeDV

了解TrustZone，读了这篇就够了

这篇文章源于老板想了解TrustZone，要求我写一篇文章简单介绍TrustZone的原理。既然是给领导看的，只介绍原理哪里够，因此也添加了公司自己现有TEE环境的设计、实现和发展，也顺便加入了一些题外话。也是因为要给领导看，所以文章也不能涉及太多技术细节，包括TrustZone模块的详细设计以及示例代码等，所以只从总体上讲解了什么是TrustZone，TrustZone是如何实现安全隔离的、TrustZone相关的一些资源等。

如果你之前对TrustZone亦无所知，好吧，本文或许值得你一看；如果你已经了解了TrustZone，想知道更多的实现细节，抱歉，本文并不适合你，或许阅读ARM官方网站和文档以及各开源项目源码是更好的选择。

本文先交代TrustZone的安全背景，然后从较高层次展开介绍TrustZone的工作机制和原理（包括AXI总线架构、CPU、内存和中断模型，以及安全隔离机制），列举了几个常见ARM平台上的实现以及当前博通ARM平台上的状况，最后附带一些TrustZone相关的开源项目以及其他资源链接，全文约7500字。（由于涉及安全的原因，本文已经删掉介绍公司自己平台相关的部分）。

本文内容主要来源于网络，综合了网上的多篇文章，也加入了一些自己的理解，重新组织了文章结构使其便于理解。

主要参考的文章包括：

- [TrustZone领域先行者](#)
- [TrustZone技术简介](#)
- [trust zone之我见](#)
- [简谈高通Trustzone的实现](#)

除上面列举的资源外，本文主要资料参考了ARM官方对TrustZone的介绍，主要有：

- 网站 developer.arm.com/technologies
- 文档 [Building a Secure System using TrustZone Technology](#)

事实上，前面多篇文章的细节也来源于官方文档。本人不保留本文的所有权，欢迎转载本文，让更多的人来了解TrustZone。由于不想再次以类似《TrustZone原理介绍》一类作为标题，但又不知道以什么作为标题贴切，所以随手用了现在标题党的套路，抱歉。

1. TrustZone介绍

1.1 安全背景

在介绍TrustZone前有必要简单回顾下目前的一些安全手段。

CPU通过内存映射手段给每个进程营造一个单独的地址空间来隔离多个进程的代码和数据，通过内核空间 and 用户空间不同的特权级来隔离操作系统和用户进程的代码和数据。但由于内存中的代码和数据都是明文，容易被同处于内存中的其它应用偷窥，因此出现了扩展的安全模块，应用将加密数据送往安全模块，由安全模块处理完后再返回结果给相应的应用。

很多消费电子设备都使用扩展的安全模块来确保数据安全，目前常见的方式有：

1. 外部挂接硬件安全模块

数据的处理交由外部的安全模块实现，这些模块能够保护自己的资源和密钥等数据的安全，如SIM卡、各种智能卡或连接到外部的硬件加解密模块等，但其同主芯片的通信线路暴露在外部，容易被监听破解。另外，通信的速率比较低。

2. 内部集成硬件安全模块

将外部安全模块的功能集成到芯片内，因此一个芯片上至少有两个核：一个普通核和一个安全核。优点是核与核之间的通信在芯片内部实现，不再暴露在外面。缺点是核之间的通信速度仍然较低，而且单独的安全核性能有限，还会占用SoC面积，成本较高。

1.2 TrustZone是个什么鬼？

TrustZone在概念上将SoC的硬件和软件资源划分为安全(Secure World)和非安全(Normal World)两个世界，所有需要保密的操作在安全世界执行（如指纹识别、密码处理、数据加解密、安全认证等），其余操作在非安全世界执行（如用户操作系统、各种应用程序等），安全世界和非安全世界通过一个名为Monitor Mode的模式进行转换，如图1：

[ARM的安全世界和非安全世界](#)

图1. ARM的安全世界和非安全世界

处理器架构上，TrustZone将每个物理核虚拟为两个核，一个非安全核（Non-secure Core, NS Core），运行非安全世界的代码；和另一个安全核（Secure Core），运行安全世界的代码。

两个虚拟的核以基于时间片的方式运行，根据需要实时占用物理核，并通过Monitor Mode在安全世界和非安全世界之间切换，类似同一CPU下的多应用程序环境，不同的是多应用程序环境下操作系统实现的是进程间切换，而Trustzone下的Monitor Mode实现了同一CPU上两个操作系统间的切换。

AMBA3 AXI(AMBA3 Advanced eXtensible Interface)系统总线作为TrustZone的基础架构设施，提供了安全世界和非安全世界的隔离机制，确保非安全核只能访问非安全世界的系统资源，而安全核能访问所有资源，因此安全世界的资源不会被非安全世界（或普通世界）所访问。

设计上，TrustZone并不是采用一刀切的方式让每个芯片厂家都使用同样的实现。总体上以AMBA3 AXI总线为基础，针对不同的应用场景设计了各种安全组件，芯片厂商根据具体的安全需求，选择不同的安全组件来构建他们的TrustZone实现。

其中主要的组件有：

- 必选组件
 - AMBA3 AXI总线，安全机制的基础设施
 - 虚拟化的ARM Core，虚拟安全和非安全核
 - TZPC (TrustZone Protection Controller)，根据需要控制外设的安全特性
 - TZASC (TrustZone Address Space Controller)，对内存进行安全和非安全区域划分和保护

- TZMA (TrustZone Memory Adapter), 片上ROM或RAM安全区域和非安全区域的划分和保护
- AXI-to-APB bridge, 桥接APB总线, 配合TZPC使APB总线外设支持TrustZone安全特性

除了以上列出的组件外, 还有诸如 Level 2 Cache Controller, DMA Controller, Generic Interrupt Controller等。

逻辑上, 安全世界中, 安全系统的OS提供统一的服务, 针对不同的安全需求加载不同的安全应用TA(Trusted Application)。例如: 针对某具体DRM的TA, 针对DTCP-IP的TA, 针对HDCP 2.0验证的TA等。

图2是一个ARM官网对TrustZone介绍的应用示意图:

 [基于TrustZone的应用示意图](#)

图2. 基于TrustZone的应用示意图

图中左边蓝色部分Rich OS Application Environment(REE)表示用户操作环境, 可以运行各种应用, 例如电视或手机的用户操作系统, 图中右边绿色部分Trusted Execution Environment(TEE)表示系统的安全环境, 运行Trusted OS, 在此基础上执行可信任应用, 包括身份验证、授权管理、DRM认证等, 这部分隐藏在用户界面背后, 独立于用户操作环境, 为用户操作环境提供安全服务。

可信执行环境 (TEE, Trusted Execution Environment) 是Global Platform (GP) 提出的概念。对应于TEE还有一个REE(Rich Execution Environment)概念, 分别对应于安全世界(Secure World)和非安全世界(Non-secure World, Normal World)。

GlobalPlatform (GP) 是跨行业的国际标准组织, 致力于开发、制定并发布安全芯片的技术标准, 以促进多应用产业环境的管理 及其安全、可互操作的业务部署。目标是创建一个标准化的基础架构, 加快安全应用程序及其关联资源的部署, 如数据和密钥, 同时保护安全应用程序及其关联资源免受软件方面的攻击。

2. TrustZone原理和设计

2.1 总线设计

- 总线

设计上, TrustZone 在系统总线上针对每一个信道的读写增加了一个额外的控制信号位, 这个控制位叫做Non-Secure或者NS位, 是AMBA3 AXI总线针对TrustZone作出的最重要、最核心的扩展设计。

这个控制信号针对读和写分别叫做ARPORT[1]和AWPORT[1]:

- **ARPORT[1]**: 用于读操作(Read transaction), 低表示Secure, 高表示Non-Secure
- **AWPORT[1]**: 用于写操作(Write transaction), 低表示Secure, 高表示Non-Secure

总线上的所有主设备(master)在发起新的操作(transaction)时会设置这些信号, 总线或从设备(slave)上解析模块会对主设备发起的信号进行辨识, 来确保主设备发起的操作在安全上没有违规。

例如: 硬件设计上, 所有非安全世界的主设备 (Non-Secure masters) 在操作时必须将信号的NS位置高, 而NS位置高又使得其无法访问总线上安全世界的从设备 (Secure Slaves), 简单来说就是对非安全世界主设备发出的地址信号进行解码时在安全世界中找不到对应的从设备, 从而导致操作失败。

NS控制信号在AMBA3 AXI总线规范中定义。可以将其看作为原有地址的扩展位, 如果原有32为寻址, 增加NS可以看成是33位寻址, 其中一半的32位物理寻址位于安全世界, 另一半32位物理寻址位于非安全世界。

当然, 非安全世界的主设备尝试访问安全世界的从设备会引发访问错误, 可能是SLVERR(slave error)或者DECERR(decode error), 具体的错误依赖于其访问外设的设计或系统总线的配置。

- 外设

在TrustZone出现前, ARM的外设基于AMBA2 APB (Advanced Peripheral Bus)总线协议, 但是APB总线上不存在类似AXI总线上的NS控制位。为了兼容已经存在的APB总线设计, AMBA3规范中包含了AXI-to-APB bridge组件, 这样就确保基于AMBA2 APB的外设同AMBA3 AXI的

例如：新一代的芯片可以通过增加AXI-to-APB bridge组件来沿用上一代芯片的设计来使其外围设备可以支持TrustZone。

2.2 处理器设计

2.2.1 处理器模型

TrustZone中，每个物理处理器核被虚拟为一个安全核(Secure)和一个非安全核(Non-Secure)，安全核运行安全世界的代码，非安全核运行除安全世界外的其它代码。由于安全世界和非安全世界的代码采用时间片机制轮流运行在同一个物理核上，相应的节省了一个物理处理器核。

多核处理器上，也有建议说让将某一个或几个核指定为安全专用核，只运行安全系统代码来构建安全世界，其余核运行非安全代码，暂不清楚目前有哪些平台采用这个实现。

图3中，系统有4个物理核，每个又分为两个虚拟核（安全核和非安全核）的情况：

多核处理器上的安全核和非安全核

图3. 多核处理器上的安全核和非安全核

2.2.2 L1内存模型

- MMU

MMU是一种硬件电路，它包含两类部件，一类是分段部件，一类是分页部件，对应于内存管理的分段机制和分页机制。分段机制把一个逻辑地址转换为线性地址；接着，分页机制把一个线性地址转换为物理地址。

当CPU访问一个虚拟地址时，这个虚地址被送到MMU翻译，硬件首先把它和TLB中的所有条目同时(并行地)进行比较，如果它的虚页号在TLB中，并且访问没有违反保护位，它的页面会直接从TLB中取出而不去访问页表，从而提高地址转换的效率。

安全世界和非安全世界都有自己的虚拟MMU，各自管理物理地址的映射。实际上只是两个世界都有一份TTBR0、TTBR1、TTBCR寄存器，因此就会对应两个MMU表。

TLB，提高执行效率。

对于TLB共享并不是硬性规定的，部分芯片在两个世界间切换时可能通过硬件部分或全部刷新TLB。

- Cache

同TLB类似，硬件上两个世界共享一套Cache，具体的Cache数据属于哪一个世界也由其NS位指定，在世界间切换也不需要刷新Cache。

2.2.3 中断模型

基于TrustZone的处理器有三套异常向量表：

- 一套用于非安全世界，
- 一套用于安全世界，
- 还有一套用于Monitor模式。

与之前非TrustZone的处理器不同的是，这三套中断向量表的基地址在运行时可以通过CP15的寄存器VBAR(Vector Base Address Register)进行修改。

复位时，安全世界的中断向量表由处理器的输入信号VINITHI决定，没有设置时为0x00000000，有设置时为0xFFFF0000；非安全世界和Monitor模式的中断向量表默认没有设置，需要通过软件设置后才能使用。

默认情况下，IRQ和FIQ异常发生后系统直接进入Monitor模式，由于IRQ是绝大多数环境下最常见的中断源，因此ARM建议配置IRQ作为非安全世界的中断源，FIQ作为安全世界的中断源。这样配置有两个优点：

- 当处理器运行在非安全世界时，IRQ直接进入非安全世界的处理函数；如果处理器运行在安全世界，当IRQ发生时，会先进入到Monitor模式，然后跳到非安全世界的IRQ处理函数执行
- 仅将FIQ配置为安全世界的中断源，而IRQ保持不变，现有代码仅需做少量修改就可以满足

将IRQ设置为非安全世界的中断源时系统IRQ的切换见图4：

 [IRQ作为非安全世界的中断源](#)

2.2.4 系统模式切换

基于TrustZone的系统有三种状态，安全世界、非安全世界和用于二者切换的Monitor Mode。

协处理器CP15的寄存器SCR(Secure Configuration Register)有一个NS位用于指示当前处理器位于哪一个世界，该寄存器在非安全世界是不能访问的。当CPU处于Monitor Mode时，无论NS位是0还是1，处理器都是在安全世界运行代码。因此Monitor Mode下总是安全世界，但如果此时NS为1，访问CP15的其它寄存器获取到的是其在非安全世界的值。

非安全世界到Monitor模式的切换

处理器从非安全世界进入Monitor Mode的操作由系统严格控制，而且所有这些操作在Monitor Mode看来都属于异常。从非安全世界到Monitor Mode的操作可通过以下方式触发：

- 软件执行SMC (Secure Monitor Call)指令
- 硬件异常机制的一个子集（换言之，并非所有硬件异常都可以触发进入Monitor Mode），包括：
 - IRQ
 - FIQ
 - external Data Abort
 - external Prefetch Abort

Monitor Mode

Monitor Mode内执行的代码依赖于具体的实现，其功能类似于进程切换，不同的是这里是不同模式间CPU状态切换。

软件在Monitor Mode下先保存当前世界的状态，然后恢复下一个世界的状态。操作完成后以从异常返回的方式开始运行下一个世界的代码。

为什么安全模式和非安全模式不能直接切换？

非安全世界无权访问CP15的SCR寄存器，所以无法通过设置NS来直接切换到安全世界，只能先转换到Monitor Mode，再到安全世界。

模式下的应用可以获取到这些数据，会有极大的安全风险。因此，只建议在Monitor Mode下通过设置NS位来切换到非安全模式。

综上，安全世界和非安全世界不存在直接的切换，所有切换操作都通过Monitor Mode来执行。

图5展现了安全世界和非安全世界之间的切换方式：

 [安全世界和非安全世界之间的切换](#)

图5. 安全世界和非安全世界之间的切换

2.3 隔离机制

除了CPU执行时实行安全世界和非安全世界的隔离外，AMBA3 AXI总线提供了外设隔离的基础。

2.3.1 内存隔离机制

这里的内存指外部的DDR和片上的ROM以及SRAM，其隔离和保护通过总线组件TZASC和TZMA的设置来实现。

- TZASC (TrustZone Address Space Controller)
 - TZASC可以把外部DDR分成多个区域，每个区域可以单独配置为安全或非安全区域，非安全世界的代码和应用只能访问非安全区域。TZASC只能用于内存设备，不适合用于配置块设备，如Nand Flash。
- TZMA (TrustZone Memory Adapter)
 - TZMA可以把片上ROM和SRAM隔离出安全和非安全区域。TZMA最大可以将片上存储的低2MB配置为安全区域，其余部分配置为非安全区域。大小划分上，片上安全区域可以在芯片出厂前设置为固定大小，或运行时通过TZPC动态配置。TZMA使用上有些限制，其不适用于外部内存划分，而且也只能配置一个安全区域。

2.3.2 外设隔离机制

总线上的设备提供类似AXI上的NS控制信号。

由于TZPC可以在运行时动态设置，这就决定了外设的安全特性是动态变化的，例如键盘平时可以作为非安全的输入设备，在输入密码时可以配置为安全设备，只允许安全世界访问。

2.3.3 隔离机制示意图

整个系统内存和外设隔离机制示意图见图6。



图6. 系统内存和外设隔离机制示意图

此图来源于网上，实际上TZPC还连接到片内的ROM/RAM设备上，用于配置片上存储的安全区域。

2.4 安全启动

AMBA3 AXI总线机制隔离出安全世界和非安全世界，但这是系统启动之后的事情。如何确保系统本身是安全的呢？这就涉及到系统启动的过程。

系统上电复位后，先从安全世界开始执行。安全世界会对非安全世界的bootloader进行验证，确保非安全世界执行的代码经过授权而没有被篡改过。然后非安全世界的bootloader会加载非安全世界的OS，完成整个系统的启动。

在非安全系统的bootloader加载OS时，仍然需要安全世界对OS的代码进行验证，确保没有被篡改。

图7是典型的TrustZone芯片的启动流程：



图7. 典型的TrustZone芯片启动流程

整个启动流程跟目前博通平台的安全启动原理基本一致，上电后安全芯片先启动，然后校验主芯片的bootloader，接下来bootloader提交系统的OS和文件系统给BSP进行校验，通过后加载主系统，确保主系统是安全的。

导致整个系统不安全。

3. 各家TrustZone实现

基于安全考虑，各家TrustZone都实行闭源，关于其实现细节的介绍都较少。

网上能找到少许关于高通方案上TrustZone的介绍：


- 安全世界 QSEE (Qualcomm Secure Execution Environment)
- 非安全世界 HLOS (High Level OS)

整个系统的架构如图8：



图8. 高通QSEE系统架构图

4. 其它

- ARMv8-A架构定义了四个异常等级，分别为EL0到EL3，其中数字越大代表特权(privilege)越大：
 - EL0: 无特权模式(unprivileged)
 - EL1: 操作系统内核模式(OS kernel mode)
 - EL2: 虚拟机监视器模式(Hypervisor mode)
 - EL3: TrustZone monitor mode  [ARMv8-A Security Level](#)
- TrustZone设计的相关方
 - ARM公司，定义TrustZone并实现硬件设计，TEE，TZAPI等
 - 芯片厂家，在具体芯片上实现TrustZone设计，包括三星、高通、MTK、TI、ST、华为等
 - 应用提供方，如DRM厂家和安全应用开发商，实现DRM、Playready、DTCP-IP和一些其它安全应用开发和认证

- Trust OS

Trustzone里面我们一般叫TrustApp，当然TEE里面每个TrustApp都在一个沙盒里，互相之间是隔离的。比如说支付，就可以做成一个App（需要注意的是，和Normal World里面的App是两个概念），这个App简单来说就负责用私钥把网上发来的Challenge签个名，而这个签名的动作是需要在Secure World里面做的，避免恶意程序窃取到私钥来伪造签名。

例如支付宝，其实支付宝也是只支持几个Trust OS的。同时，支付宝还定义了一系列标准，用来完成他的行为。

现在的Trust OS大都会遵循GlobalPlatform的规范，这个组织致力于制定统一的Trust OS的API的接口规范，这样一个TrustApp只要用GP API，就可以方便移植到各个不同的TEE操作系统上了。

- Intel 平台的 SGX

针对可信计算，类似ARM的TrustZone，Intel也针对x86平台提出了自己的安全架构SGX：

Intel® Software Guard Extensions (Intel® SGX)

software.intel.com/zh-cn/sgx-s...

SGX全称Intel Software Guard Extensions，顾名思义，其是对英特尔体系（IA）的一个扩展，用于增强软件的安全性。这种方式并不是识别和隔离平台上的所有恶意软件，而是将合法软件的安全操作封装在一个enclave中，保护其不受恶意软件的攻击，特权或者非特权的软件都无法访问enclave，也就是说，一旦软件和数据位于enclave中，即便操作系统或者和VMM（Hypervisor）也无法影响enclave里面的代码和数据。Enclave的安全边界只包含CPU和它自身。SGX创建的enclave也可以理解为一个可信执行环境TEE（Trusted Execution Environment）。不过其与ARM TrustZone（TZ）还是有一点小区别的，TZ中通过CPU划分为两个隔离环境（安全世界和正常世界），两者之间通过SMC指令通信；而SGX中一个CPU可以运行多个安全enclaves，并发执行亦可。

简单来讲，Intel SGX最关键的优势在于将程序以外的software stack如OS和BIOS都排除在了TCB（Trusted Computing Base）以外。换句话说，就是在容器enclave里的code只信任自己和intel的CPU。

网上有人是这样对比TrustZone和SGX的：

app可能也进去了，而且保险柜钥匙在管理员手上，必须相信管理员。SGX每个app有自己的保险柜，钥匙在自己手上

SGX要进入工业界应用尚需时间，一个重要的问题是现在在intel发行的服务器芯片上还没有SGX，而SGX的重要应用就是在数据中心和云端的应用。

5. TrustZone开源项目

除了各家私有实现外，ARM也有不少开源项目，知名度较高的有：

- Arm Trusted Firmware
 - 基于ARMv8-A应用处理器，ARM官方提供了一个开源参考实现BL31。
 - [github.com/ARM-softwar...](https://github.com/ARM-software/TrustedFirmware)
- Openvirtualization
 - 带有一些商业属性的开源项目，部分TEE实现只有商业版支持
 - www.openvirtualization.org/
- Op-Tee
 - Linaro 推出的开源TEE
 - github.com/OP-TEE

6. 参考

- Architecture
 - [developer.arm.com/products/ar...](https://developer.arm.com/products/architecture)
- TrustZone: <https://developer.arm.com/technologies/trustzone>
- [Building a Secure System using TrustZone Technology](#)
- ARM Trusted Firmware

- Development of TEE and Secure Monitor Code

- www.arm.com/products/se...

分类: 前端 标签: [Classyshark](#)