前言

这篇序言介绍了 ARM 安全技术文档。它包含以下部分:

关于此文档

本文概述了 ARM TrustZone 技术，以及该技术如何通过精心的片上系统(SoC)配置和软件设计来提供实用的安全性。

ARM TrustZone 技术包括对处理器的 ARM 安全扩展、添加到 AMBA 3 总线基础设施的安全信号，以及大量可用于在处理器架构和系统架构之上构建安全性的外设知识产权(IP)。

警告

本文档不提供任何相关技术的完整规范。请参见相应的技术参考手册。

目标受众

本白皮书适用于所有使用 ARM TrustZone 技术的开发人员，无论他们是编写安全要求、设计 SoC、开发软件还是审核安全设计。

本文假设您熟悉 ARM 处理器架构以及常见的硬件和软件术语。

使用此文档

本文档分为以下几章:

Chapter 1 Introduction

嵌入式安全生态系统的背景材料，讨论谁可能攻击设备，以及攻击可能采取的形式。

Chapter 2 System Security

概述部署到嵌入式设备中的一些现有安全技术，包括它们的一些优点和缺点。本章还概述了 ARM TrustZone 技术的设计理念，以及它如何包含许多替代解决方案的优势。

Chapter 3 TrustZone Hardware Architecture

ARM TrustZone 技术的详细描述，以及它如何影响基本系统组件。

本章分为三个部分:第一部分介绍 TrustZone 技术对系统基础设施的影响，第二部分介绍该技术对 ARM 处理器内核的影响，最后一部分介绍调试架构的变化。

Chapter 4 TrustZone Hardware Library

ARM 提供的 TrustZone 感知外设 IP 概述。

本节还包括一些设计建议，这些建议支持在系统设计中有限集成基于 AMBA2 AHB 接口的现有 IP。

Chapter 5 TrustZone Software Architecture

介绍使用 ARM 处理器实现 ARM 安全扩展时的一些可能的软件设计选择。

Chapter 6 TrustZone System Design

使用数字版权管理和移动支付作为示例用例的示例系统设计。

Chapter 7 Design Checklists

在设计或审查使用 TrustZone 技术的系统时，请使用本章中的清单作为提示。

进一步阅读

本节列出了来自 ARM Limited 和第三方的出版物，提供了更多信息。

ARM 定期对其文档进行更新和更正。看见 http://infocenter.arm.com 获取最新版本的技术参考手册、用户指南和 ARM 常见问题列表。

ARM 出版物

请参考 ARM 有限公司的以下出版物:

- *ARM 架构参考手册 ARM v7-A 和 ARM v7-R 版* (ARM DDI 0406)

外部出版物

请参考第三方的以下出版物:

- 可信计算小组:保护融合网络上的移动设备

  https://w（同 whitewater）白水.trustedcomputinggroup.org/groups/mobile/Final_iGR_mobile_security_white_paper_sept_2006.pdf
- 路透社英国版:你收到关于被盗手机的消息了吗？

  http://uk.reuters.com/article/domesticNews/idUKL2175341320070621
- 美国运输部:里程表欺诈的发生率

  http://www.nhtsa.dot.gov/cars/rules/regrev/evaluate/pdf/809441.pdf
- 华盛顿邮报:丢了一部黑莓手机？数据可能会导致安全漏洞

  http://www.washingtonpost.com/wp-dyn/content/article/2005/07/24/AR2005072401135.html
- 保护 Java，G. McGraw 和 E. Felten

  http://www.securingjava.com

反馈

ARM Limited 欢迎对本文档的反馈。

对此文档的反馈

如果您对本文档有任何意见，请发送电子邮件至 errata@arm.com 给予:

- 文档标题
- 文件编号
- 您的注释所适用的页码
- 对你的评论的简明解释。

也欢迎对添加和改进提出一般性建议。

第一章
介绍

本章提供了一些与嵌入式系统安全性相关的背景知识。本章包括以下几节:

1.1 什么是安全？

在非常抽象的术语中，术语“安全性”可以用来涵盖设计中许多非常不同的底层功能。然而，它本质上是系统的一个属性，确保有价值的资源不能被复制、损坏或使真正的用户不可用。每个系统设计都需要一组不同的安全属性，这取决于它试图抵御恶意攻击的资产的类型和价值。

值得保护的有价值的资源。资产可以是有形的对象，如用户密码，也可以是无形的资产，如网络可用性。

攻击故意试图获取、损坏或破坏您无权访问的资产的行为。

攻击可能包括使用恶意软件、硬件监控和硬件篡改。

为设计一个包括硬件或软件机制的系统的行为辩护，该系统提供对抗攻击的对策。

1.1.1 基本安全属性

几乎每一个更高级别的属性所基于的基本安全属性是机密性和完整性。

机密性机密资产不能被一组已定义的攻击复制或窃取。

该属性对于密码和密钥等资产至关重要。

完整性完整性得到保证的资产可以抵御一组定义的攻击的修改。

该属性对于其他系统安全性所基于的一些设备上的根秘密是必不可少的，并且对于正在运行的安全软件也是如此。

真实性在某些情况下，设计不能提供完整性，因此提供真实性的属性。在这种情况下，资产的价值可以被攻击者改变，但是防御者将能够在资产被使用之前检测到这种改变，从而在攻击导致安全故障之前检测到这种改变。

这个属性对于安全软件来说是必不可少的。如果攻击者能够在程序代码被加载到安全的执行位置之前篡改程序代码而不被检测到，那么软件提供的安全性就可以被绕过。

1.1.2 安全解决方案的局限性

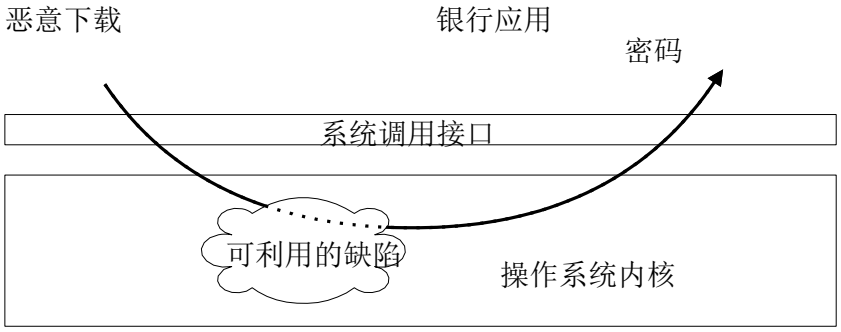
所有安全解决方案的设计都是为了防御他们可能遇到的攻击的一个子集。防御所有可能的攻击是不可能完成的任务；总有人愿意花费大量的时间和金钱，使用非常复杂的攻击来破解任何安全方案。因此，设计必须决定它想要保护哪些资产，以及它想要保护这些资产免受哪些可能的攻击。这可能是设计过程中最关键的部分；针对不正确或不完整的攻击列表保护错误资产的设计很容易被破坏。

如果有足够的时间和金钱，所有的安全性都可以被破解，那么设计的安全性需求不应该被描述为“无法绕过”，而应该用价值术语来描述:“对资产 B 的攻击 A 应该至少花费 Y 天和 Z 美元”。如果一套对策意味着一次成功的攻击将花费太长时间或代价太大，那么防御就是成功的。如果发现自己处于这种情况，大多数攻击者会转移到不同的目标。

1.2 对安全的需求

嵌入式设备正在处理越来越有价值的数据，例如消费者银行凭证。它们还稳步成为开放的软件平台，具有高度的设备外连接性，使消费者能够下载任意的第三方应用程序。这将这些设备置于高风险位置。在桌面环境中，开放软件平台、任意应用程序下载和设备上存在的宝贵资产的组合已被证明会显著增加安全漏洞被利用的风险。此外，如果通过这些系统的数据的价值不断增加，攻击者就更有可能投资破坏这些系统。

图 1-1:攻击应用程序



ARM TrustZone 技术的目标是让设备同时受益于功能丰富的开放式操作环境和强大的安全解决方案。设计良好的系统硬件架构和适当的安全软件设计可以确保敏感数据保持安全，无论不太可信的操作环境如何。

1.2.1 硬件增强的安全性

使嵌入式产品免受恶意攻击会对整个系统设计产生影响。设备中的硬件和软件必须协同工作，以实现针对正确攻击类型的强大安全对策。可信计算组织的白皮书《保护融合网络上的移动设备》指出，最受保护的系统从设计之初就包含了专用的硬件安全措施，从处理器内核和 SoC 基础设施的规格开始。在早期阶段仔细考虑硬件安全性，可以在设备中内置保护功能，而这在设计过程的后期是不可能添加的。

ARM TrustZone 技术通过将保护措施集成到 ARM 处理器、总线结构和系统外设 IP 中，实现了这种级别的系统范围安全性。通过集成硬件和软件组件的组合，TrustZone 技术提供了一个框架，允许在对设备成本影响最小的情况下实施各种安全系统架构。

1.3 威胁是什么？

在探究 TrustZone 硬件体系结构的细节之前，有必要了解在这种情况下安全的含义，以及攻击带来的风险与预防成本的比较。只有有了这些信息，系统设计师才能就保护什么以及在硬件或软件防御上投资多少做出合理的设计选择。

有许多例子，跨越多个应用和行业，与嵌入式系统抵御攻击失败相关的成本。一些攻击(如支付欺诈)会导致服务提供商必须支付的直接成本，而其他攻击(如向机顶盒添加修改过的硬件芯片)会导致设备寿命期间的长期收入损失。

1.3.1 市场部门概述

每个市场部门试图防范攻击的敏感资产各不相同。例如，移动手机旨在保护无线电网络的完整性，而电视机顶盒则防止未经授权访问订阅频道。受保护资产的不同类型和价值，加上不同的底层系统实施，意味着每种资产所遭受的攻击也各不相同。本节旨在概述可能部署 TrustZone 技术的一些市场领域的一小部分安全历史。

移动部门

GSM 手机的两个关键部分是国际移动设备识别码(IMEI),这是一个唯一的 15 位代码，用于在手机连接到网络时识别手机，以及用于将特定设备绑定到特定网络运营商的 SIM 卡的低级 SIMLock 协议。

这两个组件都用于提供安全功能:IMEI 用于阻止被盗手机访问网络，SIMLock 协议用于在合同期限内将设备与运营商绑定。在许多手机上，这两种保护机制都可以轻而易举地绕过，通常使用 USB 电缆和桌面工作站上运行的重新编程工具。

实施中的这些不安全性的结果是大规模实施欺诈的机会，路透社英国报道的统计数据表明，通过手机盗窃驱动了所有街头犯罪的一半，并使该行业每年损失数十亿美元。

对新移动设备的安全要求不再仅仅涉及网络，还涉及设备上本地可用的内容和服务。随着运营商和用户都试图从他们的设备中获得更多价值，通过数字版权管理(DRM)保护数字媒体内容以及保护机密用户数据(如同步电子邮件帐户)变得越来越重要。

消费电子和嵌入式领域

对消费电子产品的要求，如便携式游戏机和家庭电影播放器，正在与移动市场的要求趋同。越来越多的有线和无线连接、更大的用户数据存储、可编程内容的动态下载以及更高价值服务的处理，都表明需要高性能和强大的安全环境。

安全攻击不限于具有用户可扩展软件栈的开放系统。在汽车市场中，大多数系统都是封闭的或嵌入很深的，但里程表欺诈仍然很普遍，里程表读数被回滚以抬高二手车的价格。美国运输部报告称，仅这一欺诈行为每年就让美国消费者在膨胀的汽车价格上损失数亿美元。

在这些嵌入式系统中通常遇到的安全特征是那些验证固件更新是可信的，以及那些确保调试机制不能被恶意使用的特征。使用 ARM TrustZone 技术构建的设计良好的系统可以满足这些要求。

1.3.2 安全问题的经济价值

几乎每个安全设计都使用风险分析来证明其合理性，风险分析平衡了攻击成功的可能性、攻击成功时的业务成本以及防止攻击的成本。在某些情况下，风险分析将决定攻击的可能性太低，不值得防御。在其他情况下，与受保护资产的价值相比，适当的防御成本太高。然而，在大多数情况下，某种形式的资产保护是正当的。接下来的几章将展示 ARM TrustZone 技术有可能降低实施安全解决方案的成本；该架构允许降低系统复杂性，从而降低设计成本、开发成本和测试成本。这使得系统设计能够保护比以往更多的资产，即使仅仅是出于经济原因。

许多可能的攻击者也可以被认为是出于经济原因。行业中最大的损失源自职业黑客而非最终用户实施的攻击，职业黑客的主要动机是经济利益。他们在一个黑客上的投资不会超过他们合理预期的财务回报。

破坏阶级的攻击

许多设备攻击是由职业罪犯、学术安全研究人员和在家工作的爱好者推动的。无论是为了经济利益、学术声望，还是仅仅为了好玩，这些组织都可以集中大量时间来分析系统，并开发旨在破坏系统的复杂攻击。

这些攻击最追求的结果是阶级分裂；一种很容易重现的攻击，可以用来破坏整整一代或一类设备。属于这一类别的最广泛发布的攻击是针对消费娱乐设备部署的攻击，例如破坏游戏控制台上的软件限制和 DVD 电影上的内容保护方案的攻击。

在许多设备硬件受到攻击的破解场景中，攻击者的第一次调查研究可能会花费大量资金。这笔钱用于资助获得工具，如电子显微镜和晶体管激励激光器，用于硅水平的分析。这种攻击的目的是发现安全弱点，随后可以在多种设备上利用这些弱点，而无需付出巨大的代价。如果可攻击设备的数量足够多，这些破坏类别的攻击可以极大地改变经济论点的平衡，使之有利于攻击者。

实证经济学

安全环境的好处不仅限于技术方面。研究表明，增加消费者对易于使用和安全的支付系统的信心可以将消费者支出提高 20%，同时还可以更快地吸收新的收入流和不同的商业模式。

设备制造商越来越意识到消费者的安全问题，设备的安全特性正成为竞争差异化的一个问题。优先选择具有良好安全审查和有用安全功能的设备，而不是不太明确的设备。对于部署在公司企业环境中的设备来说尤其如此，在这种环境中，移动设备可以连接到内部网络。将机密电子邮件和日历约会同步到智能手机的功能非常有用，但正如《华盛顿邮报》所言

报告显示，如果设备丢失或被盗，所有者公司将面临很高的风险。许多公司现在限制对公司网络的访问，降低了智能设备的价值，或者要求设备本身具有额外的安全机制。

1.3.3 设备是如何被攻击的？

设计中下一个最重要的因素是用来执行攻击的机制。执行攻击的不同机制被称为攻击媒介，这些机制在本文中分为三类——黑客攻击、shack 攻击和实验室攻击。

黑客攻击

黑客攻击是指黑客只能执行软件攻击。黑客攻击的例子包括通过物理或无线连接下载到设备的病毒和恶意软件。

在许多成功的黑客攻击案例中，设备用户无意中批准了软件的安装，然后执行攻击。这要么是因为恶意软件伪装成用户确实想要安装的软件，要么是因为用户不理解操作环境显示的警告消息。

在《保护 Java 安全》一书中，有一节总结了典型用户在安全性和理想功能之间做出选择时的决策能力:

"如果在跳舞的猪和安全之间做出选择，用户每次都会选择跳舞的猪."

棚屋攻击

shack 攻击是一种低预算的硬件攻击，使用的设备可以在大街上从 Radio Shack 等商店购买。在这些情况下，攻击者可以物理访问设备，但没有足够的设备或专业知识来攻击集成电路封装。

攻击者可以尝试使用 JTAG 调试、边界扫描 I/O 和内置自检工具连接到设备。他们可以使用逻辑探针和网络分析仪来窥探总线线路、引脚和系统信号，从而被动地监控系统。攻击者还可以执行简单的主动硬件攻击，例如强制引脚和总线处于高电压或低电压，重新编程存储设备，以及用恶意替代物替换硬件组件。

实验室攻击

实验室攻击媒介是最全面和最具入侵性的。如果攻击者可以访问实验室设备，如电子显微镜，他们就可以对设备进行无限制的逆向工程。必须假设攻击者可以对设计的任何敏感部分(包括逻辑和存储器)进行晶体管级细节的逆向工程。

攻击者可以对设计进行逆向工程，将微观逻辑探针附着在硅金属层上，并使用激光或其他技术干扰正在运行的电路。攻击者还可以监控模拟信号，如设备电源使用和电磁辐射，以执行加密密钥分析等攻击。

在大多数情况下，考虑到每个设备都可能被攻破的经验法则，设备不应该尝试直接防御实验室攻击，而是应该采取措施在设备被攻破时限制损害，从而使实验室攻击变得不经济。使用每个设备独有的秘密是对单个设备进行逆向工程不能为攻击者提供有用信息的一个例子；他们知道自己已经拥有的设备的秘密，但不知道该类别中任何其他设备的秘密。

注意

TrustZone 技术旨在提供安全组件和 SoC 基础设施其余部分之间的硬件强制逻辑隔离。

实验室攻击不在 TrustZone 技术提供的保护范围内，但如果某些资产需要防止物理攻击，使用 TrustZone 的 SoC 可以与 ARM SecurCore 智能卡结合使用。

1.3.4 谁攻击设备？

一旦设计人员确定了资产和可能的攻击，确定可能的攻击者就很重要。不同的攻击者可以部署不同类型的攻击，某些资产只会吸引某些攻击者。这种分析有助于合理化每项资产需要防范的攻击。

该分析还应包括明确信任谁可以访问存储在设备上的资产的描述。这可能会突出安全模型中的弱点。已经有许多公开的案例，其中消费者数据被维护或维修技术人员从设备中窃取，并随后在互联网上公布。

远程攻击者

传统观点认为，攻击者是我们在桌面工作站上看到的恶意软件的分发者:病毒和其他恶意软件。这些攻击者无法实际接触到他们攻击的设备，尽管他们可能接触到类似的设备来进行攻击，并依靠利用软件漏洞和用户错误来获得对敏感资源的访问权。

嵌入式设备日益增加的软件复杂性意味着统计上有更多的漏洞可供攻击者利用。安装第三方代码(包括执行基于浏览器的内容)的能力使恶意代码更容易进入设备进行攻击。

安全专家

技术能力最强的攻击者是犯罪团伙、安全专家和以攻击设备为乐的用户。这些组能够执行实验室攻击，如所述 page 1-10.

这类攻击者通常试图发现可在大量设备上重现的类破坏攻击。这种二次攻击可能由另一个攻击者部署。

值得信赖的开发人员

一个经常不被考虑的攻击是由一个被认为是可信任的人执行的，或者至少是协助的。公司员工因窃取与设计相关的秘密信息或故意破坏内部计算机网络而成为新闻焦点。在许多情况下，在攻击被发现的前一天，这些员工会被认为是明确信任的。

用经济学的观点来证明抵御这类攻击的合理性，贿赂某人获得设计信息通常比逆向工程一块硅片更便宜，也更快。

尽管这种类型的攻击很难防范，但是可以使用业务流程中的一些防御措施。可以使用限制访问敏感材料的流程，以及审核谁访问了这些材料。

设备所有者

我们将在本文中讨论的针对设备的最后一组攻击者是设备所有者本身。这组攻击者的典型目标是获得对服务和内容的免费访问。一般来说，设备所有者有动力执行

攻击，但技术不胜任。一个技术专家会开发出一种攻击，并在网上公布其细节；设备所有者所要做的就是按照食谱制作并复制它。

有趣的是，这类设备的拥有者面临比平常更高的攻击风险。在他们试图打破现有的保护措施时，他们错误地从黑客托管的网站上下载预先打包的攻击软件；这些网站通常包含病毒和其他恶意软件，用户会无意中将其安装到他们的系统中。

第二章
系统安全

在介绍 ARM TrustZone 硬件架构之前，本章概述了嵌入式系统中的一些现有安全选项。

本章包括以下几节:

2.1 系统安全

嵌入式设备的系统设计非常复杂，包括多个独立的处理器内核、DMA 引擎等二级总线主设备以及大量的存储器和外设总线从设备。除了这些功能组件之外，通常还有一个并行系统基础设施，它提供侵入式和非侵入式调试功能，以及组件边界扫描和内建自测(BIST)设施。

平台中的每一个子系统都必须以这样的方式设计和集成，即它与安全解决方案一起工作，而不是独立于安全需求开发每个子系统。如果设备的威胁模型表明它需要防范 shack 攻击，那么只保护系统的功能部分是没有意义的。不受限制地访问调试端口的攻击者可以绕过许多可能存在的功能保护。

本节旨在研究市场上已有的一些安全架构，以及它们的优缺点。

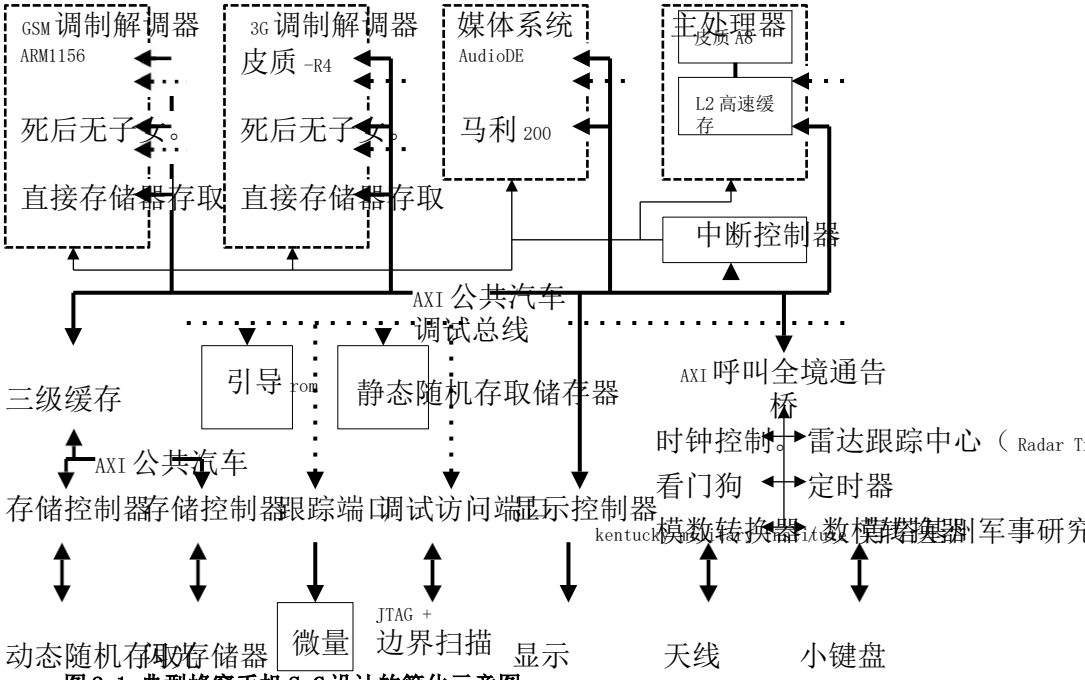


图 2-1:典型蜂窝手机 SoC 设计的简化示意图

2.1.1 外部硬件安全模块

嵌入式应用的经典安全解决方案是在主 SoC 之外加入专用硬件安全模块或可信元件。例如移动手机中的 SIM 卡或机顶盒中的条件访问智能卡。

优势

外部安全设备将资产封装在物理设备中，旨在提供强大的安全性。

使用完全独立的设计和制造流程允许使用能够提供高水平的防篡改性和物理安全性的技术和硅工艺。

智能卡制造和个性化过程经常通过批准的评估方案得到正式认证。这使得它们适用于需要高度安全性保证的用例，例如信用卡和借记卡支付方案。

不足之处

用于智能卡的硬件技术和工艺对于标准 SoC 设计来说是不切实际的，因为它们会大大增加设计过程的工作量，并损害器件的面积、功率效率和性能。

提供物理安全性的制造方法也迫使较低的处理器性能，在 5-20 Mhz 的范围内，以及设备内少量的 RAM。这些设计特征将可以在智能卡上部署的可能用例限制为相对静态的程序，这些程序可以从非易失性存储器运行，并且不需要跨越安全边界的连续高带宽流量。

智能卡的另一个问题是它们仅提供处理和安全存储功能；他们不能直接访问父设备的任何用户界面。智能卡依赖于在其安全边界之外运行的软件来提供这些设施，因此不能保护所有感兴趣的资产。例如，用户输入的个人身份号码(PIN)必须由智能卡之外的不太安全的软件来管理，这使得智能卡容易受到攻击。

智能卡的商业劣势可能是比技术障碍更大的障碍。在主 SoC 旁边提供智能卡是昂贵的，因此对于大多数资产来说是不经济的，因为它们不够有价值。

2.1.2 内部硬件安全模块

位于 SoC 内的硬件安全模块牺牲了智能卡提供的硬件防护，以降低成本和方便作为系统的集成部分。

集成模块的精确形式各不相同，但有两种主要形式。第一个是管理加密操作和密钥存储的硬件模块。第二个是通用处理引擎，它与主处理器放在一起，使用定制的硬件逻辑来防止对敏感资源的未授权访问。

优势

与智能卡相比，这些系统的主要优势在于显著降低了成本并提高了性能。

在安全性方面，为安全子系统提供专用通用处理器的系统可与 TrustZone 硬件解决方案相媲美。

不足之处

加密模块的实现具有与智能卡相似的问题，因为它们具有受限的周界，因此只能保护加密密钥材料。密码术是一种工具，其本身并不是目的；不可避免的是，受加密方法保护的资源将需要在加密模块之外使用，在那里它们不再受加密模块的保护，并且可能受到攻击。

与 TrustZone 系统相比，提供专用于安全子系统的第二通用处理器的设计具有较小的缺点。

一个问题是设计需要一个单独的物理安全处理器。该处理器通常不如主应用处理器强大，并且还消耗大量的硅面积。此外，两个处理器之间的通信需要将任何数据刷新到一致的存储器，通常是外部存储器。这种操作非常耗时，并且会消耗大量的能量。

另一个问题是，任何资源分离都需要在 SoC 内的专有硬件扩展中实现。这需要大量的设计和测试工作，并且迁移或扩展系统变得更加困难。

这里介绍的加密模块和二级处理器解决方案通常试图保护系统的功能。与 SoC 中可用的调试和测试机制相关联的系统安全风险通常被忽略，并且提供了易受攻击的接口。为了避免这个问题，必须完全禁用调试，这使得很难在现场诊断软件问题。

2.1.3 软件虚拟化

虚拟化是一种软件安全机制，其中高度可信的管理层(称为管理程序)在通用处理器的特权模式下运行。虚拟机管理程序使用内存管理单元(MMU)将运行于其上的多个独立软件平台分隔开来，并将每个平台置于由虚拟机管理程序软件控制的虚拟机中。

某些虚拟机管理程序的紧凑特性意味着，如果可以对它们进行彻底的测试，在它控制的一个虚拟机中运行的软件可以确信它不会受到在其他虚拟机中运行的软件的攻击。

注意

本节描述半虚拟化；这是嵌入式市场中常见的虚拟化类型。也存在具有用于管理程序的特殊处理器模式的处理器架构。这些体系结构允许虚拟机管理程序托管完整的操作系统，然后该操作系统可以细分虚拟机管理程序使用 MMU 给予它的内存空间。

优势

任何带有 MMU 的处理器都可以用来实现虚拟化解决方案，一些常见的富操作系统已经被移植到其上运行。实施虚拟机管理程序也不需要额外的硬件。

安全敏感的应用可以被移植到运行在管理程序之上的安全环境中，但是在丰富操作环境的视野之外。管理程序可以提供通信机制，该机制可用于允许软件虚拟机进行通信。

不足之处

虚拟化技术提供的隔离仅限于实现虚拟机管理程序的处理器。系统中的任何其他总线主控器，如 DMA 引擎和图形处理单元(GPU)，都可以绕过管理程序提供的保护，因此也必须由管理程序管理，以实施所需的安全策略。

这很难在不损害系统性能的情况下实现；在不降低图形性能的情况下验证可编程 GPU 的复杂输入超出了大多数虚拟化解决方案的范围。

虚拟化忽略了与硬件攻击相关的安全问题，例如使用调试或测试基础设施的威胁。为了保护虚拟化系统免受这种攻击，您需要完全禁用调试和测试可见性，这使得软件开发和缺陷诊断非常困难。

2.2 TrustZone 硬件安全性

与上一节讨论的安全系统相关联的问题之所以存在，是因为设计只能保护系统的受限部分中的一些资产，或者是因为安全解决方案忽略了攻击问题空间的大部分。

在许多情况下，安全设计的限制性意味着保护了错误的资产。密码硬件块主要被设计成保护密钥，密钥无疑是有价值的资产，但是如果当解密的内容在密码硬件之外的媒体播放器中被消费时，攻击者能够重复地窃取解密的内容，则密钥的保护对于内容权利所有者来说没有什么安慰。

对当今部署的真实设备的攻击多种多样，但可以证明涵盖了本章前面概述的所有攻击模式。任何只试图应对其中一个威胁领域的安全解决方案都忽略了问题的一半以上，将无法抵御现实世界中存在的许多攻击。这可能会使许多解决方案达到的安全级别低于它们想要提供的级别，从而意味着许多资产得不到充分的保护。

2.2.1 全系统安全性

ARM 在嵌入式领域实现可信计算的方法基于可信平台的概念；在整个系统设计中扩展安全基础设施的硬件架构。TrustZone 架构不是保护专用硬件模块中的资产，而是使系统的任何部分都变得安全，从而实现包括功能单元和调试基础设施的端到端安全解决方案。

通过适当使用构建在 TrustZone 架构之上的安全协议，例如安全引导和认证调试启用，许多可能的黑客攻击威胁都可以构建某种形式的对策。如果这些防御措施可以与降低实验室攻击相关风险的方法结合使用，例如，通过让每个设备使用统计上唯一的秘密，一个非常强大的解决方案就开始出现了。

中详细介绍了 TrustZone 硬件架构 Chapter 3。

第三章
TrustZone 硬件架构

本章详细介绍了 TrustZone 硬件架构及其对系统结构、处理器和调试基础设施的影响。

本章包括以下几节:

3.1 概观

TrustZone 硬件架构旨在提供一种安全框架，使设备能够应对它将面临的许多特定威胁。TrustZone 技术不是提供固定的通用安全解决方案，而是提供基础设施基础，允许 SoC 设计人员从一系列组件中进行选择，以在安全环境中实现特定功能。

该架构的主要安全目标实际上相当简单；能够构建一个可编程的环境，保护几乎所有资产的机密性和完整性免受特定攻击。具有这些特征的平台可用于构建各种各样的安全解决方案，这些解决方案用传统方法是不划算的。

系统的安全性是通过划分 SoC 的所有硬件和软件资源来实现的，以便它们存在于两个世界中的一个——安全子系统的安全世界，以及其他一切的正常世界。支持 TrustZone 的 AMBA3 AXI 总线结构中的硬件逻辑可确保正常世界组件无法访问安全世界资源，从而在两者之间构建强大的安全边界。将敏感资源置于安全环境中，并实施在安全处理器内核上运行的强大软件的设计，可以保护几乎任何资产免受许多可能的攻击，包括那些通常难以保护的攻击，如使用键盘或触摸屏。

TrustZone 硬件架构的第二个方面是已经在一些 ARM 处理器内核中实现的扩展。这些附加功能使单个物理处理器内核能够以时间分片的方式安全高效地执行来自正常世界和安全世界的代码。这消除了对专用安全处理器内核的需求，从而节省了芯片面积和功耗，并允许高性能安全软件在正常的操作环境中运行。

TrustZone 硬件架构的最后一个方面是安全感知调试基础架构，它可以控制对安全环境调试的访问，而不会影响正常环境的调试可见性。

这三个方面将在本章的以下章节中详细讨论。

注意

《ARM 架构参考手册》和许多硬件组件技术参考手册使用安全和非安全这两个术语，它们相当于安全世界和普通世界。

当仅提及硬件时，本文将使用安全和非安全命名约定，以避免混淆。术语安全世界和正常世界将用于描述形成每个执行环境的硬件和软件的组合。

3.2 系统结构

系统 IP 的架构变化提供了将硬件资源分成两个世界的机制。本节介绍这些更改的影响以及已经修改的组件。

3.2.1 AMBA3 AXI 系统总线

扩展总线设计最重要的特点是在主系统总线上为每个读写通道增加了一个额外的控制信号。这些位被称为非安全位或 NS 位，在公共 AMBA3 高级可扩展接口(AXI)总线协议规范中定义。

- AWPROT[1]:写事务 – 低表示安全，高表示不安全。
- ARPROT[1]:读取事务 – 低表示安全，高表示不安全。

所有总线主机在处理新事务时都会设置这些信号，总线或从机解码逻辑必须解释这些信号，以确保不违反所需的安全隔离。所有不安全的主机必须在硬件中将它们的 NS 位设置为高，这使得它们不可能访问安全的从机。用于访问的地址解码将不匹配任何安全从设备，并且事务将失败。

如果一个不安全的主设备试图访问一个安全的从设备，那么操作是无声地失败还是产生一个错误是由实现定义的。根据硬件外设设计和总线配置，从机或总线可能会产生错误，因此可能会出现 SLVERR(从机错误)或 DECERR(解码错误)。

3.2.2 AMBA3 APB 外设总线

TrustZone 架构最有用的功能之一是保护外设，如中断控制器、定时器和用户 I/O 设备。这使得安全环境可以扩展，从而可以解决一些更广泛的安全问题，这些问题不仅仅需要安全的数据处理环境。安全中断控制器和定时器支持不间断的安全任务来监控系统，安全时钟源支持强大的 DRM，安全键盘外设支持安全输入用户密码。

AMBA3 规范包括一个称为高级外设总线(APB)的低门数低带宽外设总线，它使用 AXI 到 APB 桥连接到系统总线。APB 总线不传送等效的 NS 位。这确保了现有的 AMBA2 APB 外设与实施 TrustZone 技术的系统兼容。AXI 到 APB 桥硬件负责管理 APB 外围设备的安全性；网桥必须拒绝不适当的安全设置的事务，并且不得将这些请求转发给外围设备。

3.2.3 内存混淆

将 NS 位添加到总线事务和系统中的任何高速缓存标签，可以被视为提供第 33 个地址位。有一个用于安全事务的 32 位物理地址空间和一个用于非安全事务的 32 位物理地址空间。

与任何地址空间一样，包括没有 TrustZone 技术的地址空间，必须注意确保 33 位地址空间的使用方式能够使数据在其存储的所有位置保持一致，否则可能会导致数据损坏。

考虑这样一种情况，安全世界主设备想要访问被缓存的非安全从设备。设计可以实现以下选择之一:

- 主设备对从设备进行不安全的访问。
- 主设备对从设备进行安全访问，而不安全的从设备接受安全交易。从设备将这些访问视为不安全的。

在第二种设计中，硬件必须支持地址空间别名。在这个别名存储器系统中，相同的存储器位置在地址映射中表现为两个不同的位置，一个是安全的，一个是不安全的。因此，缓存中可能同时存在多个表示相同数据的值。对于可修改的数据，这种混淆会导致一致性问题；如果数据的一个副本被修改，而另一个副本存在于缓存中，您将拥有数据的版本，但两个版本将会不同。系统设计者必须意识到潜在的数据一致性问题，并且必须采取措施来避免它们。

3.3 处理器架构

最显著的架构变化适用于实现架构安全扩展的 ARM 处理器。目前这些是:

- ARM1176JZ(F)-S 处理器
- cortex-A8 处理器
- Cortex-A9 处理器
- Cortex-A9 MPCore 处理器

这些设计中的每个物理处理器内核都提供了两个虚拟内核，一个被认为是不安全的，另一个被认为是安全的，并提供了一种在它们之间进行可靠的上下文切换的机制，称为监控模式。在主系统总线上发送的 NS 位的值是从执行指令或数据访问的虚拟内核的身份中间接得出的。这使得能够将虚拟处理器简单地集成到系统安全机制中；不安全的虚拟处理器只能访问不安全的系统资源，但是安全的虚拟处理器可以看到所有的资源。

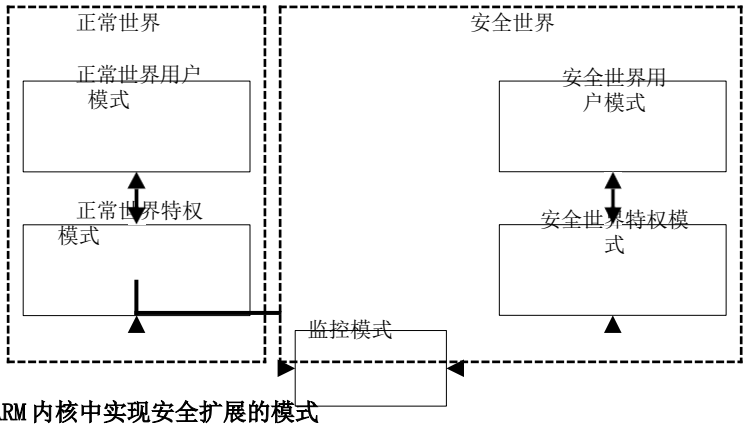


图 3-1:ARM 内核中实现安全扩展的模式

3.3.1 切换世界

这两个虚拟处理器以时间分片的方式执行，在更改当前运行的虚拟处理器时，通过一种称为监控模式的新核心模式进行上下文切换。

物理处理器从正常状态进入监控模式的机制受到严格控制，并且都被视为监控模式软件的例外。要监控的条目可以由执行专用指令(安全监控调用(SMC)指令)的软件触发，或者由硬件异常机制的子集触发。IRQ、FIQ、外部数据中止和外部预取中止异常都可以配置为使处理器切换到监控模式。

在监控模式下执行的软件是由实现定义的，但是它通常保存当前世界的状态并恢复被切换到的世界的状态。然后，它执行异常返回，以在恢复的世界中重新开始处理。

除非处理器处于监控模式，否则系统控制协处理器 CP15 的安全配置寄存器 (SCR)中的 NS 位表示处理器的执行环境。在监控模式下，无论 SCR NS 位的值是多少，处理器总是在安全环境中执行，但是如果 SCR NS 位设为 1，则 CP15 寄存器组上的操作将访问正常环境副本。

注意

如果安全世界软件在处理器不处于监控模式时将 SCR NS 位设置为 1，处理器将立即切换到正常世界运行。这将使不太可信的软件能够看到仍在流水线中的指令的执行，以及处理器寄存器中保存的任何数据。如果指令或寄存器中的数据是敏感的，这可能会导致安全违规。因此，建议只有监控模式软件才能直接修改 NS 位。

普通软件不能访问 SCR 的内容。

3.3.2 保护一级存储系统

内核外部的内存基础设施将系统分为两个世界，内核内部也需要进行类似的分区，以便将一级(L1)内存系统组件中使用和存储的数据分开。

存储器管理单元

ARM 应用程序配置文件处理器中 L1 存储器系统的主要组件是存储器管理单元 (MMU),它能够将处理器上运行的软件看到的虚拟地址空间映射到处理器外部的物理地址空间。使用软件控制的转换表来管理地址转换，该转换表详细说明了哪个虚拟地址对应于每个物理地址，以及关于存储器访问的一些其他属性，例如可缓存性和访问权限。

在具有 MMU 但没有安全扩展的 ARM 内核中，例如 ARM926EJ-S 处理器，在任何时刻都有一个虚拟地址到物理地址的映射。特权模式代码通常重写进程上下文切换上的一组新的表，或将硬件指向这组新的表，以提供多个独立的虚拟存储器空间。在 TrustZone 处理器中，硬件

提供两个虚拟 MMU，每个虚拟处理器一个。这使得每个世界都有一组本地的转换表，使它们能够独立控制虚拟地址到物理地址的映射。

ARMv6 和 ARMv7 L1 转换表描述符格式包括一个 NS 字段，安全虚拟处理器使用该字段来确定在访问与该表描述符相关的物理存储器位置时要使用的 NS 位的值。不安全的虚拟处理器硬件忽略该字段，并且总是在 NS=1 的情况下进行存储器访问。这种设计使得安全虚拟处理器能够访问安全或非安全存储器。

为了实现世界之间的有效上下文切换，ARM 处理器实现可以用执行遍历的世界的身份来标记翻译旁视缓冲器(TLB)中的条目，TLB 缓存翻译表遍历的结果。这允许非安全条目和安全条目在 TLB 中共存，从而实现更快的全球交换，因为无需刷新 TLB 条目。

注意

ARM 架构并不强制要求在 TLB 标签中存在所有世界的身份；它是实现定义的。一些处理器硬件可能刷新世界交换上的一些或所有 TLB 条目。

隐藏所

在高速缓存中支持两种安全状态的数据是任何高性能设计的期望特征。这消除了在世界之间切换时对缓存刷新的需要，并使高性能软件能够跨越世界边界进行通信。为了实现这一点，L1 以及适用的二级和二级以上的处理器高速缓存被扩展为具有额外的标记位，该标记位记录访问存储器的事务的安全状态。

关于安全状态，高速缓存的内容是动态的。任何未锁定的高速缓存行都可以被驱逐，以便为新数据腾出空间，而不管其安全状态如何。安全线路加载可以驱逐非安全线路，而非安全线路加载可以驱逐安全线路。

例子

将上述所有概念放在一起，Figure 3-2 显示了理论 ARM 处理器的 L1 内存系统在访问内存系统时如何处理与安全扩展相关的状态。

1. 核心处理逻辑尝试数据加载、数据存储或指令预取。硬件将虚拟地址(VA)和当前世界(非安全表标识符，或 NSTID)传递给 TLB，使其能够执行地址转换。

2. TLB 加载物理地址(PA)以及与它所传递的 VA 和 NSTID 相关联的 NS 位，执行页表遍历，并在必要时在 NSTID=1 时强制 NS=1。然后，TLB 将此信息传递给高速缓存，以执行实际的数据或指令访问。

3. 高速缓存尝试将来自 TLB 的 PA 和 NS 位与现有高速缓存行的标签进行匹配。如果成功，它将从该高速缓存行返回数据，否则它将从外部存储器系统加载该高速缓存行。

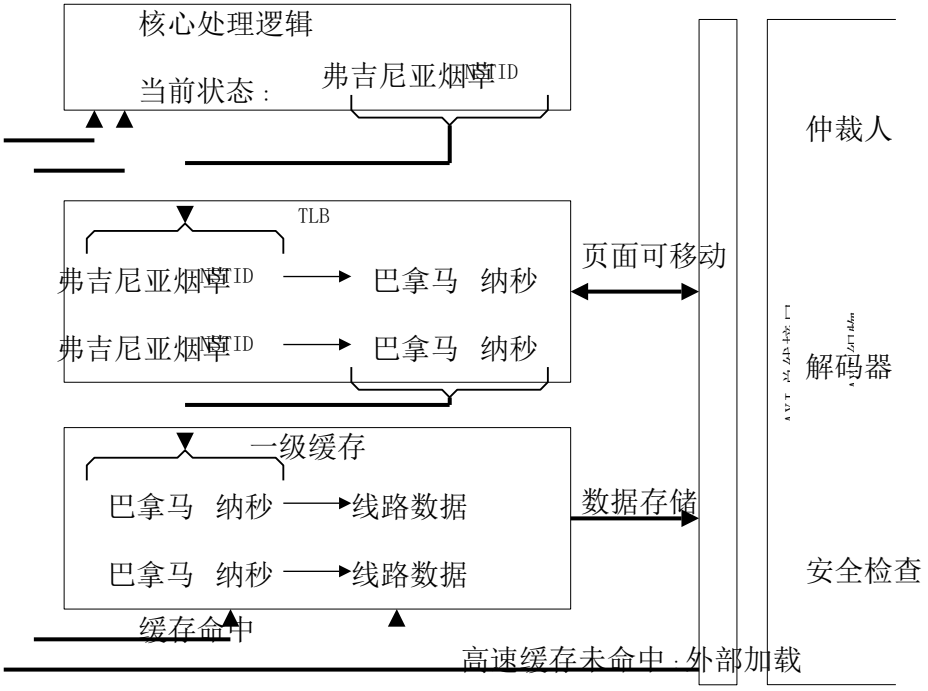


图 3-2:理论 ARM 内核的一级内存系统

在媒体应用中，加密的音频内容被加载到正常世界媒体播放器中，并在安全世界中被解密，安全世界软件可以在安全世界转换表中映射包含属于媒体播放器的数据的非安全存储器。这允许安全世界直接访问包含需要解密的音频内容的非安全高速缓存行；这种类型的内存被称为共享内存。因此，普通世界的应用程序可以通过缓存层次结构中的任何级别将数据传递给安全世界中的同伴任务。与需要将高速缓存的数据从高速缓存中刷新出来并存入外部存储器的解决方案相比，这实现了高性能的系统。

紧密结合的记忆

ARM1176JZ(F)-S 处理器支持紧耦合存储器(TCM)，这是一种高性能 SRAM 模块，与 L1 高速缓存在存储器层次结构中处于同一级别。根据合成时配置的 TCM 的总大小，每个指令和数据接口上最多有两个 TCM 块。软件可以将 TCM 的每个块配置为仅进行安全访问，或者仅进行不安全访问，并对每个块的基地址进行独立控制。

加速器一致性端口

一些 ARM 处理器，如 ARM Cortex-A9 MPCore 处理器，包括一个可选的加速器相干端口(ACP)。ACP 是处理器上的一个 AXI 从接口，它使连接到它的任何外设主机能够访问与处理器一致的物理存储器映射。这允许外部外设访问位于 ARM 处理器高速缓存层次内的数据。这项技术减少了需要从 ARM 处理器缓存中清除数据和/或使数据失效的用例数量，提高了必须与外部外设(如 DMA 控制器)密切共享数据的软件的性能。

ARM 处理器存储器系统代表使用 ACP 的外部主机进行的存储器访问的安全状态将与访问 ACP 的总线事务的安全状态相同。在 ACP 接口读取的 ARPROT[1]信号的值将用于读取，AWPROT[1]信号的值将用于写入。

AXI 从机可以区分 ARM 处理器代表内部处理单元进行的事务和代表 ACP 事务进行的事务。这取决于处理器的微体系结构，并不一定在所有实现中都可用。Cortex-A9 MPCore 处理器根据存储器访问的发起者设置以下 AXI 信号:

- ARIDMx[2]:读取事务 – 如果发起者是 ARM 处理器，则为低电平；如果是 ACP，则为高电平。
- AWIDMx[2]:写事务—如果发起者是 ARM 处理器，则为低电平；如果是 ACP，则为高电平。

安全敏感从设备的地址解码逻辑可以使用该信号来确定访问是否源自信任的主设备。例如，这种技术可以用来对访问 ACP 的其他主机隐藏 ARM 处理器的部分物理存储器映射。

注意

ACP 可以像其他 AXI 奴隶一样被安全访问。

3.3.3 安全中断

能够将 IRQ 和 FIQ 直接捕获到监控器，而无需任何代码干预，因此可以为安全中断源创建灵活的中断模型。一旦执行到达监视器，可信软件可以相应地路由中断请求。当与安全感知中断控制器结合时，这允许设计提供不能被正常世界软件操纵的安全中断源。

ARM 推荐的模型是使用 IRQ 作为正常世界中断源，使用 FIQ 作为安全世界源。IRQ 是大多数操作环境中最常用的中断源，因此使用 FIQ 作为安全中断意味着对现有软件的修改最少。如果当中断发生时，处理器正在运行正确的虚拟内核，则没有切换到监视器，并且中断在当前环境中被本地处理。当中断发生时，如果内核在另一个世界中，则硬件捕获到监视器，监视器软件引起上下文切换并跳转到恢复的世界，此时中断被接受。

注意

建议监控器总是在中断被屏蔽的情况下执行。

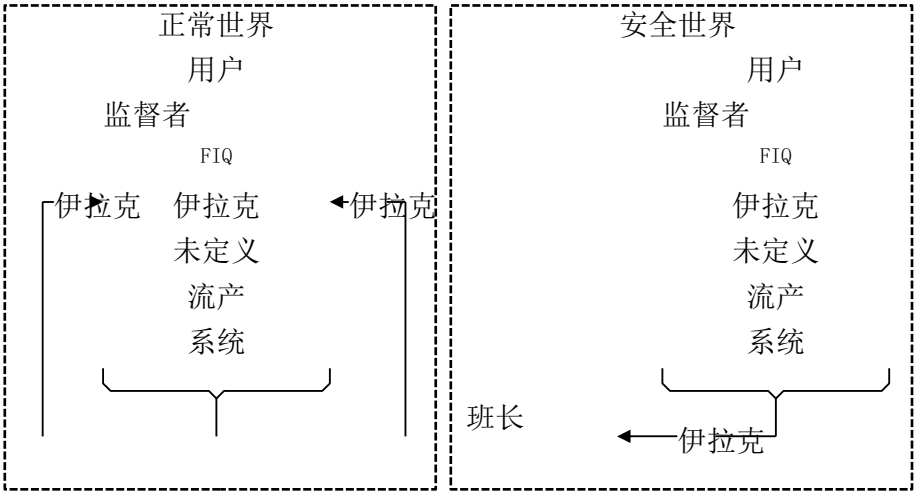


图 3-3:IRQ 配置为非安全中断的设计中一种可能的 IRQ 路由

为了防止恶意的正常世界软件屏蔽敏感的安全世界中断，处理器硬件在 CP15 中包括一个配置寄存器，可用于防止任何正常世界软件修改 CPSR 中的 F (FIQ 屏蔽)和 A(外部中止屏蔽)位。该控制寄存器只能由 Secure world 软件访问。请注意，没有选项可以阻止正常环境屏蔽 IRQ 中断。

处理器异常向量表

为了提供上述异常行为，支持 TrustZone 的处理器实现了三组异常向量表。其中一个表用于正常世界，一个用于安全世界，另一个用于监控模式。

复位时安全世界表的基址与 VINITHI 处理器输入信号的设置一致；如果未置位，0x00000000 如果置位，0xFFFF0000。其他表的基址未定义，应在使用前由软件设置。

与前几代 ARM 处理器不同，每个表的位置可以在运行时移动。这可以通过对 CP15 中适当的矢量基址寄存器(VBAR)进行编程来实现。

注意

通过设置 CP15 控制寄存器中的 V 位，可以在运行时启用或禁用高矢量的使用。如果 V 位被置位，处理器异常总是从 0xFFFF0000 开始的表中获取，而不管 VBAR 中存储的值。存储 V 位的值，从而能够独立配置安全世界和正常世界向量表。

V 位仅适用于相应的安全世界和正常世界异常表；监控异常表总是位于监控向量基址寄存器中指定的存储器地址。

3.3.4 安全处理器配置

为了能够在虚拟 CPU 上独立执行代码，硬件严格管理 CP15 中的配置选项。被认为是敏感的或全局应用于核心的配置选项只能由安全世界软件编写，尽管大多数配置选项可以由普通世界软件读取。

不是全局敏感的设置，因此可以在每个世界的本地应用，通常存储在硬件中。这使得每个世界可以独立控制影响其实现的设置。

只有安全环境可以修改的一些全局配置选项对正常环境的实施有一些影响，特别是在访问一些低级硬件功能时，如缓存锁定或系统协处理器。然而，遗留软件通常只需要很少的修改或者不需要修改就可以在正常环境中执行。

3.3.5 具有安全扩展的多处理器系统

ARM 架构支持在一个集群中包含一到四个处理器的多处理器设计。集群中的处理器可以配置为以对称多处理(SMP)模式或非对称模式执行
多处理模式。

当处理器在 SMP 模式下执行时，集群的窥探控制单元(SCU)将透明地将 SMP 处理器间共享的数据保持在 L1 数据缓存中。当处理器在 AMP 模式下执行时，如果需要，执行软件必须手动维护存储器一致性。

1. 只有 MMU 支持的存储器类型的子集通过 SCU 保持一致。如需详细资讯，请参阅适当的处理器技术参考手册。

这些多处理器系统可以实现 ARM 安全扩展，为集群中的每个处理器提供本章前面描述的程序员模型功能。目前实现多处理器特性和安全特性的 ARM 处理器是 Cortex-A9 MPCore 处理器

注意

多处理器系统通常包括加速器一致性端口，该端口允许外部总线主控访问与处理器集群相同的物理存储器视图。看见 page 3-10 了解更多详情。

每个处理器两个世界

多处理器群集中的每个处理器都有一个正常世界和一个安全世界。这为四处理器集群提供了总共八个虚拟处理器，每个虚拟处理器都可以独立控制其 MMU 配置。

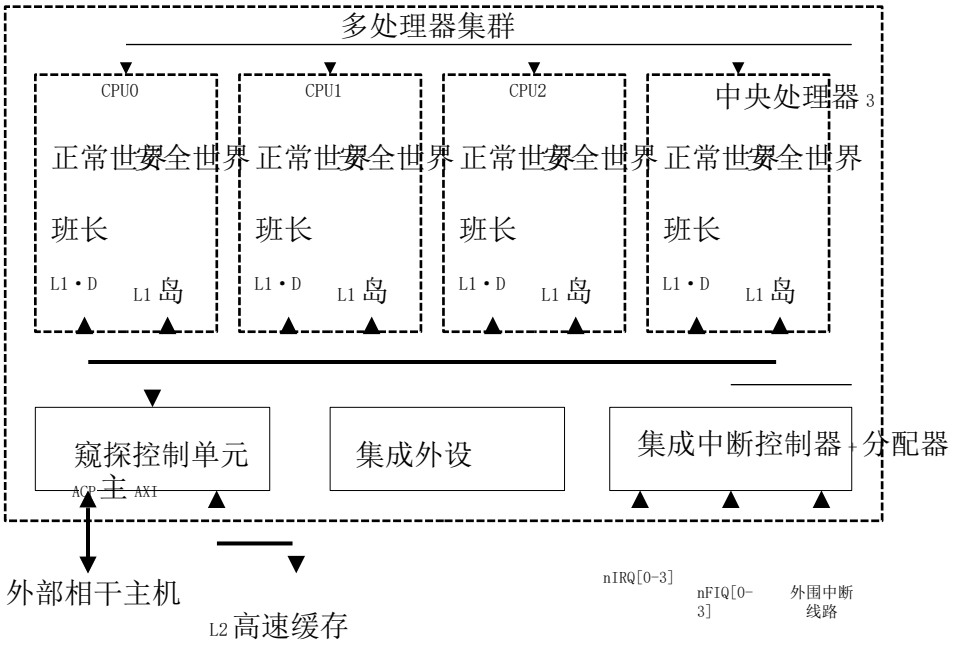


图 3-4 ARM 多处理器集群

集群中任意数量的处理器可以在任意时间点处于安全世界中，并且处理器可以独立于集群中的其他处理器在安全世界之间转换。特定软件实现可以选择限制安全世界软件的并发执行，以降低与复杂软件设计相关联的安全风险。

注意

中讨论了多重处理对安全世界软件设计的潜在影响 *Secure software and multiprocessor systems on page 5-13.*

如中所述 *Caches section on page 3-8,* 群集中的每个高速缓存行存储它包含的数据的安全状态，作为其标签的一部分。当集群中的处理器在 SMP 模式下执行时，这允许在 L1 处理器数据高速缓存中同时存储安全和非安全数据。一致性硬件在执行一致性操作时使用整个高速缓存标签，允许它同时保持安全和非安全数据的一致性。

窥探控制单元配置

SCU 包括许多配置寄存器，用于确定 SCU 本身的配置、群集中每个 ARM 处理器的配置以及处理器本地定时器对非安全存储器事务的可访问性。

- SCU 访问控制寄存器决定了群集中的哪些处理器可以对 SCU 的配置寄存器重新编程。
- SCU 安全访问控制寄存器确定非安全访问是否可以重新编程 SCU 配置寄存器或访问处理器本地定时器。

中断处理

Cortex A-profile 多处理器系统包括一个集成的中断控制器，其技术与 PrimeCell 通用中断控制器(PL390)相同，详见 page 4-6. 该中断控制器提供了一种灵活的中断模型，能够在多处理器集群中分配优先中断，如果接收到较高优先级的中断，则中断已经在执行的较低优先级的中断处理程序。

在也实现安全扩展的多处理器系统中，该中断控制器是信任区感知的。这允许它管理安全和非安全中断，并防止非安全存储器访问读取或修改安全中断的配置。

通过对中断安全寄存器中的适当位进行编程，可以将集成中断控制器管理的中断配置为安全中断。一旦中断变得安全，任何不安全的访问都不能修改其配置。

由集成中断控制器管理的所有中断都分配了一个优先级，以确定是否允许它们中断 ARM 处理器正在处理的异常。硬件确保较低优先级的中断会等到较高优先级的中断被清除后再发送给处理器。优先级空间被划分，以确保安全中断总是可以被配置为具有比非安全中断更高的优先级。为安全世界分配高优先级中断可用于防止非安全世界使用中断对安全世界进行拒绝服务攻击。

集成中断控制器可以支持本章前面描述的模型，使其控制的安全中断生成 FIQ 异常
它控制产生 IRQ 异常的非安全中断。在这种情况下，所有中断都由集成中断控制器管理，不可能从外部中断控制器直接产生中断。集成中断控制器还支持许多传统配置，这些配置会导致 FIQ 和/或 IRQ 异常由外部中断触发器产生，从而完全绕过集成中断控制器。

可以独立配置 FIQ 和 IRQ 异常的传统中断生成。

- 如果仅针对 FIQ 异常启用传统模式，则集成控制器会将其控制的安全和非安全中断路由至 IRQ 异常向量。
- 如果仅针对 IRQ 异常启用传统模式，则集成控制器将无法为非安全中断生成异常，安全中断将被路由至 FIQ 异常向量。
- 如果 FIQ 和 IRQ 异常都启用了传统模式，则集成中断控制器将被完全旁路。

注意

如果在设计中使用，处理器集群的传统中断输入信号通常由一个或多个外部中断控制器产生。可以使用与 TrustZone 系统中任何其他外部 AXI 或 APB 从设备相同的方法来保护这些外部设备。

3.4 调试架构

与安全扩展集成的系统基础结构的最后一部分是调试条款。ARM 提供的调试解决方案分为两部分:处理器调试组件和系统调试组件。

3.4.1 处理器调试控制

在引入安全扩展之前，ARM 处理器已经包括单个调试控制信号，该信号全局启用或禁用调试器对处理器的访问。在这些设计中，在丰富的操作环境中部署安全敏感软件意味着必须全局禁用调试，否则您将面临来自简单硬件攻击的巨大威胁。

随着操作环境变得越来越大和越来越复杂，这开始变得远非理想的情况，特别是当有一个希望为丰富的操作环境开发软件的大型开发人员社区时。这些人需要调试他们的应用程序，通常是在设备到达现场并且安全软件已经启用之后。

TrustZone 调试扩展将调试访问控制分为以下各个方面的独立可配置视图:

- 安全特权入侵(JTAG)调试
- 安全特权非侵入式(跟踪)调试
- 安全用户侵入式调试
- 安全用户非侵入式调试

安全特权调试访问由内核的两个输入信号 SPIDEN(侵入式)和 SPNIDEN(非侵入式)控制。安全用户模式调试访问由安全特权访问专用 CP15 寄存器中的两位 SUIDEN(侵入式)和 SUNIDEN(非侵入式)控制。这些设置使 TrustZone 处理器能够在部署设备后控制调试可见性。例如，有可能提供完全正常世界调试可见性，同时还防止所有安全世界调试。

注意

ARM 处理器还包括全局调试使能输入信号:DBGEN 和 NIDEN，后者位于实施 ARMv7 架构的内核上。这些信号可用于禁用内核的所有调试可见性，包括正常的全局调试。

- arm V6:DBGEN – 全局侵入式和非侵入式调试使能。
- arm V7:DBGEN – 全局侵入式调试使能。
- arm V7:NIDEN – 全局非侵入式调试使能。

多处理器调试控制

在实现多处理器扩展的ARM处理器中，如 *Multiprocessor systems with the Security Extensions section on page 3-13,* 集群中的每个处理器都有独立的DBGEN、NIDEN、SPIDEN和SPNIDEN输入信号。这允许以可能方式的子集来调试集群中的处理器子集。

注意

系统设计者必须意识到，SMP数据一致性硬件可能允许启用侵入式调试的处理器修改被禁用侵入式调试的处理器使用的数据。

技术性能分析

为了支持代码的低水平基准测试，ARMv6和ARMv7应用级处理器在CP15中包含一个性能监控器。该硬件单元可用于对代码执行进行计时，并对运行时可能发生的处理器事件进行计数，例如高速缓存行驱逐。

为了防止性能监视器被用来攻击Secure world软件，可以使用secure CP15配置选项来防止正常环境和用户模式访问这些计数器。

3.4.2 系统调试控制

ARM系统调试解决方案是ARM CoreSight片上调试和跟踪技术。它为整个SoC提供调试和跟踪解决方案，支持多个处理器和其他系统组件的调试。CoreSight调试基础设施既可以通过设备外工具访问，也可以通过设备内组件访问。

可从片上系统硬件和软件访问的CoreSight基础设施部分被实施为APB外设。为了减少所需的组件数量，CoreSight外围设备被设计为不使用由AXI到APB桥提供的标准的每外围设备保护机制；CoreSight组件应该可以访问非安全内存事务。

作为AXI-APB桥提供的保护的替代方案，CoreSight组件包括许多控制信号，用于启用或禁用安全调试。这些信号被称为CoreSight认证接口，包括SPIDEN、SPNIDEN和DBGEN信号，其作用类似于针对处理器内核描述的同名信号。

如果外部调试硬件或目标 Normal world 软件在 SPIDEN 被解除断言时试图在安全地址上设置系统断点，CoreSight 硬件将无法创建断点。对于仪器解决方案，如果 SPNIDEN 未置位，安全跟踪信息将被外设丢弃。

注意

调试安全架构的结果是，当 SPIDEN 或 SPNIDEN 被断言时，正常世界软件可能能够直接影响或监视系统中的安全世界执行。

因此，只有当设备位于可信环境中时，才应启用安全调试。

第四章
TrustZone 硬件库

本章概述了 ARM 提供的支持 TrustZone 的 IP。本章包括以下几节:

4.1 系统 IP

本节概述了 ARM 提供的一些系统 IP，这些 IP 内置了对安全扩展的支持。

注意

如果您使用适当的 AXI 到 APB 桥，任何 APB 外围设备都可以使用安全扩展来保护。

4.1.1 PrimeCell 高性能矩阵- PL301

实现系统级隔离的主要元件是符合 AMBA3 AXI 标准的总线矩阵，它将所有系统元件连接在一起。ARM 在 PrimeCell 高性能矩阵产品 PL301 中提供了这一特性。

为了满足现代 SoC 基础设施的要求，AXI 总线发电机还配备了一系列支持组件。用于时序隔离的寄存器片、用于将总线宽度缩小到低带宽 SoC 区域的宽度缩放缩小器以及用于链接时钟域的同步或异步桥均可用:

- PrimeCell 基础架构 AMBA3 AXI 寄存器片- BP130
- prime cell infra structure AMBA 3 AXI 缩减版- BP131
- prime cell infra structure AMBA 3 AXI 至 AXI 桥梁- BP132-4

PrimeCell 基础设施 AMBA3 AXI 至 APB 桥- BP135

在典型的 ARM 系统中，大多数外设都连接到 APB 总线。APB 是一种比 AXI 总线更简单、功耗更低的总线。APB 协议不携带与总线事务的信任区安全状态相关的位。这使得在 AMBA3 APB 总线上使用现有的外设设计成为可能，并将管理安全状态的责任放在 AXI 到 APB 桥上，该桥提供高速 AXI 域和低功率 APB 域之间的接口。

每个 AXI 到 APB 桥都提供一个 AXI 从接口，并且可以调解其本地 APB 总线上多达 16 个外围设备的访问。该桥包含地址解码逻辑，根据输入的 AXI 事务产生 APB 外设选择。桥包括一个单独的 TZPCDECPROT 输入信号，用于总线上的每个外设。该信号用于确定外围设备被配置为安全还是不安全；桥将拒绝不安全的交易以保护外围设备地址范围。

这些桥输入信号可以在合成时固定连接，也可以通过可信外设(如 TrustZone 保护控制器(TZPC))进行动态控制，以便在运行时动态切换安全状态。

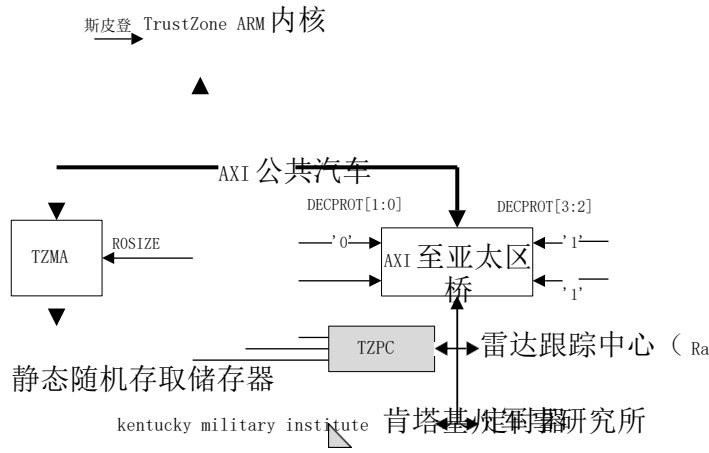


图 4-1:使用 TZPC 控制 SoC 上的安全信号

Figure 4-1 显示了控制 4 个外设的 AXI 到 APB 桥。TZPC 被配置为总是安全的，定时器和实时时钟(RTC)总是安全的
非安全，键盘和鼠标接口(KMI)在软件控制下具有可编程的安全状态。安全世界软件可以在运行时对 TZPC 编程，以改变 AXI 到 APB 桥的信号输入，从而将 KMI 从安全切换到非安全，反之亦然。

注意

图中的阴影块表示安全外围设备，带阴影角的块表示可以从安全切换到非安全的外围设备。

这种设计允许外围设备在大多数时间存在于正常世界，但允许它们在短时间内临时切换到安全世界。KMI 外围就是一个很好的例子；它通常用作通用的普通键盘，但也可能在短时间内转换为安全输入设备，以便在安全的环境中输入用户密码。如该图所示，TZPC 的增加允许动态控制 SoC 上的其它信号，例如 ARM 内核的 SPIDEN 调试控制输入。

prime cell infra structure AMB a3 AXI 至 AMBA 2 AHB 大桥- BP136-7

AXI 到 AHB 和 AHB 到 AXI 的网桥允许实现 AMBA3 和 AMBA2 规范的两个子系统连接在一起。每个通信方向都需要一个单独的网桥；AXI-AHB 桥允许 AXI 事务在 AHB 总线上主控，AHB-AXI 桥允许 AHB 事务在 AXI 总线上主控。

AHB 总线不提供任何机制来承载通过它的内存事务的安全状态，因此所有的安全实现都必须在桥本身中管理。

- AXI-AHB 桥允许整个 AHB 从域变得安全或者不安全。
- AHB-AXI 桥允许整个 AHB 主域变得安全或不安全。

如果设计需要安全和非安全 AHB 主设备和从设备的混合，建议不要将安全组件和非安全组件放在同一个网桥后面。

——— **注意** ———————

AHB 域内的安全性完全由 AHB 配置管理，不在安全扩展的范围之内。

———————————

4.1.2 PrimeCell 二级高速缓存控制器- PL310

随着处理器时钟频率的上升，越来越需要在内核和外部存储器系统之间包括二级高速缓存。这减少了由于访问外部存储器的延迟而导致的处理器流水线停顿次数，在某些应用中实现了显著更快的性能，并且通常降低了功耗。TrustZone 系统中的每个缓存都需要用其包含的数据的安全状态来标记每个缓存行，从而实现来自两个世界的数据的并发存储。

ARM1176JZ(F)-S 处理器和 Cortex-A9 处理器可以利用单独的二级缓存控制器，该控制器必须实现适当的安全状态标记。ARM 高性能 PrimeCell 二级缓存控制器可用于此目的。

——— **注意** ———————

Cortex-A8 处理器包括一个集成的 L2 高速缓存控制器。

———————————

4.1.3 PrimeCell DMA 控制器- PL330

使用处理器在系统中移动数据可能是对功率和处理器执行时间的低效使用。因此，许多系统都包括直接内存访问控制器(DMAC),它是在物理内存系统中移动数据的专用引擎。

PrimeCell DMA 控制器是一个多通道 AXI 引擎，具有微编码的工作描述，能够传输复杂的结构。DMAC 可以同时支持安全和非安全通道，每个通道都有独立的中断事件，并由专用 APB 接口控制。试图对进出安全存储器的 DMA 传输进行编程的非安全事务将导致 DMA 传输失败。

4.1.4 PrimeCell 信任区地址空间控制器- PL380

TrustZone 地址空间控制器(TZASC)是一个 AXI 组件，它将其从属地址范围划分为多个内存区域。TZASC 可以由安全软件编程以将这些区域配置为安全或不安全的，并且将拒绝不安全的交易到被配置为安全的区域。

当综合设计时，存储器区域的数量和 TZASC AXI 接口的总线宽度是可配置的。

使用 TZASC 的主要原因是将单个 AXI 从设备(如片外 DRAM)划分为多个安全域。片外 RAM 是一个很好的例子，因为由于额外的引脚排列、印刷电路板面积和存储器本身的成本，存储器设备具有与其相关的显著成本。因此，希望系统对单个外部存储器进行分区，使得它可以包含安全区域和非安全区域；这通常比放置两个较小的存储器设备更便宜。

ARM AXI 动态内存控制器(DMC)系列是一系列高性能控制器，内部不支持创建安全和非安全分区。为了创建安全分区，可以在 DMC 和需要访问它的片上主机之间放置一个 TZASC。TZASC 设计用于动态内存，允许突发访问通过它，对内存延迟的影响最小。

注意

TZASC 只能用于对内存映射设备进行分区；特别是，它不能用于对基于数据块的设备(如 NAND 闪存)进行分区。

4.1.5 prime cell infra structure AMB a3 AXI trust zone 内存适配器- BP141

TrustZone 内存适配器(TZMA)使设计人员能够保护片上静态内存(如 ROM 或 SRAM)中的区域。与为每个世界提供单独的专用存储器相比，放置单个大存储器设备并将其划分为安全和非安全区域通常更便宜。TZMA 允许将高达 2MB 的单个静态存储器划分为两个区域，其中下半部分是安全的，上半部分是非安全的。

安全区域和非安全区域之间的分区位置总是 4KB 的倍数，并通过 TZMA 的 ROSIZE 输入信号来控制。这些信号可以通过将信号连接到 TZPC 外设的 TZPCROSIZE 输出来动态配置，也可以在合成时绑定，以固定方式对存储器进行分区。

TZMA 不能用于对动态存储器或需要一个以上安全区域的存储器进行分区；对于这些设计，必须使用 TZASC。

4.1.6 PrimeCell 通用中断控制器- PL390

为了支持安全和非安全中断的健壮管理，底层中断控制器必须防止正常世界修改安全世界中断源的配置。这意味着单个中断控制器必须支持使用内部分区的 TrustZone 技术，或者必须在系统中放置两个中断控制器。

通用中断控制器(GIC)是支持安全和非安全优先级中断源的单一硬件设备。GIC 硬件将阻止普通软件修改被配置为安全源的中断线路的配置的尝试。此外，不安全的软件只能配置优先级范围下半部分的中断，以防止拒绝服务攻击。

注意

ARM Cortex-A9 MPCore 集成了自己的中断控制器，与 GIC 的编程器型号相同，因此不需要外部中断控制器。有关集成中断控制器的更多信息，请参见 *Interrupt handling on page 3-15*。

4.1.7 PrimeCell 基础设施 AMBA3 信任区保护控制器- BP147

如“AXI 至 APB 桥”部分所述，信任区保护控制器(TZPC)是一个可配置的信号控制模块，可以放置在 APB 总线上，为 SoC 上的其他组件提供控制信号。TZPC 包括三个

通用寄存器 TZPCDECPROT{2:0}，每个寄存器可以控制 SoC 内的 8 个信号。它还包括一个寄存器 TZPCR0SIZE，可用于为 TrustZone 内存适配器提供分区位置控制信号。

TZPC 的上电状态是将 TZPCDECR0T 寄存器的所有位设置为 0(这是安全的),并将 TZPCR0SIZE 寄存器设置为 0x200(这使得 TZMA 支持的整个 2MB 存储器范围是安全的)。引导代码可以放松安全设置，根据需要使组件变得不安全。

4.2 处理器 IP

有许多处理器实现了 AMBA3 AXI 存储器接口，但并不是所有的处理器都实现了相同的功能集。本节概述了这些处理器的主要特性，以及如何在 TrustZone 系统中使用它们。

本节还将介绍 ARM SecurCore 系列智能卡处理器，它可以与支持 TrustZone 的 SoC 一起使用，以提供更高级别的物理安全性。

4.2.1 ARM1176JZ(F)-S 处理器

ARM1176JZ(F)-S 处理器是首款 TrustZone 处理器。它实现了 ARMv6Z 架构，并使用 ARM Artisan Metro 标准单元库提供了一个能够在 90 纳米工艺中实现 320MHz 的 8 级单发射整数流水线。当采用这种硅工艺时，它消耗 1mm2 的芯片面积，不包括高速缓存。

与 TrustZone 系统设计相关的有趣特性:

- 可选的紧密耦合存储器(TCM)。
- 可选外部二级高速缓存控制器(PL310)。

4.2.2 Cortex-A8 处理器

Cortex-A8 处理器是 ARMv7 A 系列处理器中的第一款。它实现了一个 13 级双发行整数管道和一个额外的 10 级霓虹灯媒体 SIMD 管道。它能够使用 ARM Artisan Advantage 库在 65 纳米 LP 工艺中实现 650MHz，消耗不到 3mm2 的芯片面积，不包括 NEON 和高速缓存 ram。

与 TrustZone 系统设计相关的有趣特性:

- 没有中药。
- 集成二级高速缓存控制器。

4.2.3 Cortex-A9 处理器和 Cortex-A9 MPCore 处理器

Cortex-A9 是一款 ARMv7 A 系列处理器，有单核和 1-4 路多核两种实现方式。它实现了一个无序、多发射超标量流水线，带有一个可选的矢量浮点(VFP)或 NEON 流水线。

与 TrustZone 系统设计相关的有趣特性:

- 首款实现 ARM 安全扩展的多处理平台，在四核设计中总共提供八个虚拟处理器。
- 没有中药。
- 设计用于 PrimeCell 二级高速缓存控制器(PL310)。设计用于通用中断控制器(PL390)。
  这只是 Cortex-A9 的单处理器版本的要求；多处理器 MPCore 版本包括一个内置中断控制器。

4.2.4 ARM1156T2(F)-S 处理器

ARM1156T2(F)-S 处理器是一款带有存储器保护单元(MPU)的 ARMv6 处理器，专为嵌入式应用而设计。处理器本身不实现安全扩展，但它实现了 AMBA3 AXI 总线接口，这使它能够直接置于支持 TrustZone 的 SoC 设计中，而无需额外的逻辑或桥。

在大多数设计中，ARM1156T2(F)-S 将被静态绑定，以生成安全存储器事务或非安全存储器事务，从而使 ARM1156T2(F)-S 能够与主应用处理器一起充当非对称处理引擎，作为安全世界的一部分，或者作为正常世界的一部分。

4.2.5 皮层-R4 处理器

Cortex-R4 处理器是一款面向嵌入式应用的 ARMv7 R 系列处理器。与 ARM1156T2(F)-S 处理器类似，Cortex-R4 处理器不实现安全扩展，但它实现了 AMBA3 AXI 总线接口，使其能够直接置于支持 TrustZone 的 SoC 设计中，无需额外的逻辑或桥接器。

4.2.6 SecurCore 智能卡处理器

ARM SecurCore 系列提供了一系列专为智能卡和防篡改集成电路部署而设计的处理器 IP。对于需要比 TrustZone 系统所能提供的更高级别的物理安全性的安全应用，基于 SecurCore 的智能卡可以与 TrustZone 系统一起使用。TrustZone 系统可以提供智能卡无法提供的功能，例如安全用户界面和可以处理高性能、不太敏感的安全任务的软件。

处理器特性概述:

- SC100:支持 ARM 和 Thumb 指令集。
- SC200:支持 ARM 和 Thumb 指令集以及集成的 Jazelle 技术，用于加速 Java Card 2.x 应用程序。
- SC300:基于 Cortex-M3，SC300 支持多种技术，包括 Thumb -2 指令集，以降低内存需求并提高软件性能。

4.3 AMBA2 AHB IP 的重用

系统设计者可能需要在也使用 AMBA3 和安全扩展的设计中重用传统的 AMBA2 AHB 主模块和从模块。这些传统设备可能包括非处理器主机，如 DMA 控制器，以及复杂的子系统，包括主机，如 ARM926EJ-S 处理器。

4.3.1 AHB 大师的再利用

在这些设计中，任何 AHB 主机都必须连接到 AXI 域，通过 AHB 到 AXI 的桥传递它们的事务，如中所述 page 4-4。这些网桥基于网桥配置信号实现了 AXI NS 比特的简单映射。从 AXI 域的角度来看，这允许网桥后面的整个 AHB 域变得安全或不安全。

如果仅单个网桥之后的 AHB 主机的子集需要可用作 AXI 域中的安全主机，则需要额外的定制逻辑来生成到网桥的配置输入，该配置输入基于 AHB 主机 ID 来确定 NS 位。

注意

每个 AHB 主机只能直接寻址 4GB 的物理内存。不可能在一个 AHB 主机的存储器映射中寻址全部 8GB 的安全和非安全物理地址空间。

4.3.2 AHB 奴隶的再利用

ARM 建议使用单独的 AHB 公共汽车来容纳安全和不安全的 AHB 奴隶。这使得能够使用由 AXI-AHB 桥提供的单个 NS 比特控制信号；桥将拒绝任何具有无效安全许可的交易。

在这不可能的情况下，可以在 AXI 到 AHB 桥之前放置一个 TrustZone 地址空间控制器，以便对 ns 位执行基于地址的检查。

第五章
TrustZone 软件架构

本章着眼于利用 ARM 安全扩展的一些可能的软件架构。

本章包括以下几节:

5.1 软件概述

在 SoC 硬件中实现安全世界需要一些安全软件在其中运行，并利用存储在其中的敏感资产。

安全扩展是 ARM 架构的一个开放组件，因此任何开发人员都可以创建一个自定义的 Secure world 软件环境来满足他们的需求。本节介绍了软件架构师在设计安全世界软件堆栈时可能要考虑的一些可能性。

注意

这里列出的每个软件选项都对安全世界硬件提出了不同的要求。例如，具有独立的抢先式安全世界操作系统的设计将需要安全定时器和安全感知中断控制器。

5.1.1 保护世界处理资源

软件架构的整体结构将受到可用的安全世界处理资源的性质的严重影响。系统可以提供支持 TrustZone 的内核，如 ARM1176JZ(F)-S 处理器，也可以为安全领域提供专用处理器，如 Cortex-R4 处理器。

具有两个物理处理器的设计是一种经典的嵌入式设计，不会受到添加安全扩展的严重影响。在安全世界处理器上运行的软件必须是自包含的，并且提供其自己的本地操作环境。

预计大多数设计将选择使用支持 TrustZone 的处理器。这通常为安全领域提供了更高的软件性能，并且比专用安全处理器需要更少的芯片面积。这就是我们在本章剩余部分要关注的情况。

5.1.2 软件体系结构

支持 TrustZone 的处理器内核上的安全世界软件堆栈可以实现许多可能的软件架构。最复杂的是专用的安全世界操作系统:最简单的是放置在安全世界中的同步代码库。在这两个极端之间有许多中间选择。

安全操作系统

安全世界中的专用操作系统是一种复杂但强大的设计。它可以模拟多个独立的安全世界应用程序的并发执行，新安全应用程序的运行时下载，以及完全独立于正常世界环境的安全世界任务。

这些设计的最极端版本非常类似于在非对称多处理(AMP)配置中具有两个独立物理处理器的 SoC 中看到的软件堆栈。运行在每个虚拟处理器上的软件是一个独立的操作系统，每个世界使用硬件中断来抢占当前运行的世界并获取处理器时间。

紧密集成的设计使用通信协议将安全世界任务与请求它们的正常世界线程相关联，可以提供对称多处理(SMP)设计的许多好处。例如，在这些设计中，安全世界应用可以继承它正在协助的正常世界任务的优先级。这将为媒体应用实现某种形式的软实时响应。

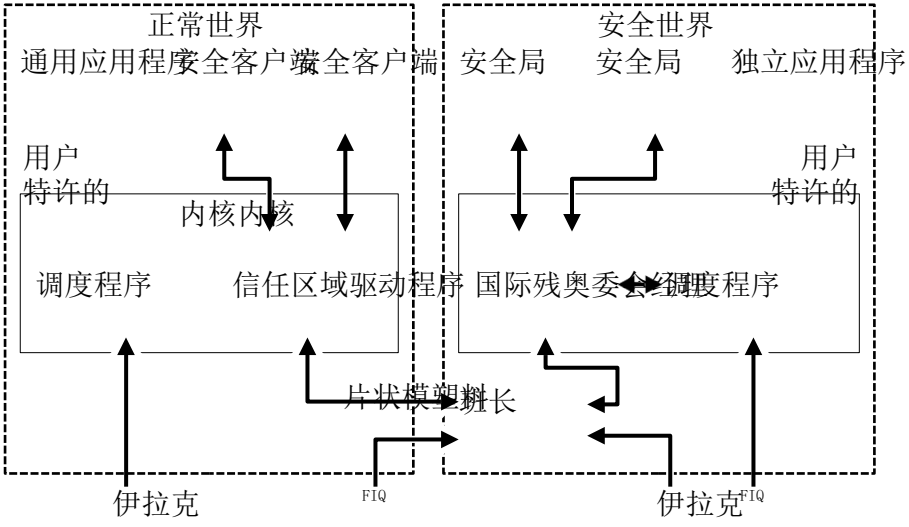


图 5-1:一个独立的安全世界操作系统的可能架构

基于操作系统原理的设计的优点之一是使用处理器 MMU 将安全世界存储器空间分成多个用户空间沙箱。假设安全世界内核软件被正确实现，来自独立利益相关者的安全任务可以同时执行

需要互相信任。内核设计可以加强安全任务之间的逻辑隔离，防止一个安全任务篡改另一个安全任务的内存空间。

同步库

许多用例不需要安全世界操作系统的复杂性。在安全世界中，一个一次可以处理一个任务的简单代码库对于许多应用程序来说已经足够了。这个代码库完全是使用普通操作系统的软件调用来调度和管理的。

这些系统中的安全世界是正常世界的奴隶，不能独立运行，但因此可以具有低得多的复杂度。

中间选项

在这两个极端之间有一系列的选择。例如，安全世界多任务操作系统可以被设计成没有专用中断源，因此可以由正常世界提供虚拟中断。如果正常世界的操作系统停止提供虚拟中断，这种设计将容易受到拒绝服务攻击，但在许多情况下，这不是一个有问题的攻击。或者，MMU 可用于静态分离同步安全世界库的不同组件。

5.2 引导安全系统

安全系统生命周期中的一个关键点是启动时。许多攻击者试图在设备断电时破坏软件，例如用已被篡改的软件映像替换闪存中的安全世界软件映像。如果系统从闪存启动映像，而没有首先检查其真实性，则系统易受攻击。

这里应用的原则之一是为所有安全世界软件和潜在的正常世界软件生成信任链，该信任链是从不能被轻易篡改的信任根建立的。这就是所谓的安全引导序列。看见 *Secure boot on page 5-6*.

5.2.1 启动顺序

启用 TrustZone 的处理器在通电时在安全环境中启动。这使得任何敏感的安全检查能够在普通软件有机会修改系统的任何方面之前运行。

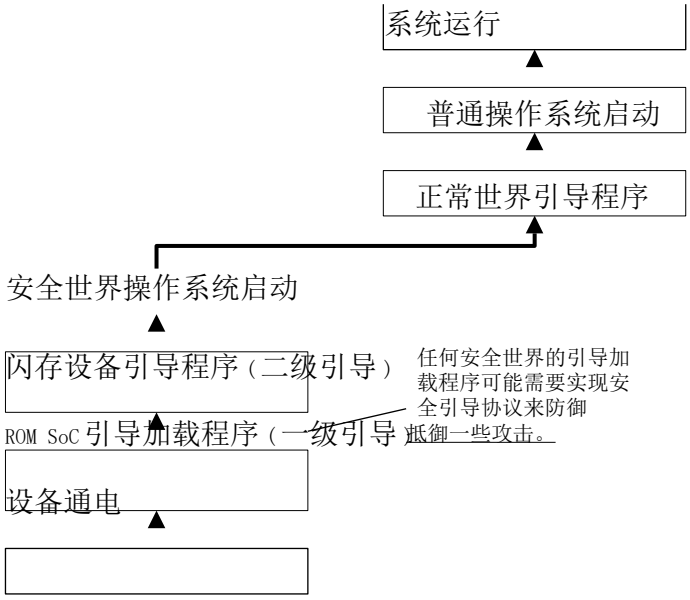


图 5-2:支持 TrustZone 的处理器的典型引导序列

在通电之后，大多数 SoC 设计将开始执行基于 ROM 的引导加载程序，该引导加载程序负责在切换到位于外部非易失性存储器(例如闪存)中的设备引导加载程序之前初始化关键外围设备(例如存储器控制器)。然后，引导序列将通过安全世界操作进行

环境初始化阶段，在将控制传递给正常的引导加载程序之前。这将进展到启动正常的世界操作系统，在这一点上，系统可以被认为正在运行。

系统控制协处理器锁定

需要额外保护的系统可以使用处理器内核的信号输入来锁定 CP15 中的一些关键安全配置选项。断言 CP15 可禁用处理器输入信号将导致某些 Secure world CP15 设置变得不可修改，即使修改是由 Secure world 特权软件尝试进行的。

预期使用 CP15SDISABLE 的设计将在引导过程中配置敏感设置，并在将控制传递给普通软件之前断言信号。

请注意，系统启动时必须将 CP15SDISABLE 设置为低电平，以启用安全世界启动代码来配置 CP15 寄存器的适当设置。用于改变信号的方法应该仅可用于安全领域，并且其他保护可能是合适的，例如使用锁存信号发生器，该发生器只能通过复位设备来复位到低状态。

5.2.2 安全启动

安全引导方案将加密检查添加到安全世界引导过程的每个阶段。此过程旨在维护所有执行的安全世界软件映像的完整性，防止任何未经授权或恶意修改的软件运行。

密码签名协议

最合理的加密协议是基于公钥签名算法的协议，如 RSA-PSS (Rivest、Shamir 和 Adleman 概率签名方案)。

在这些协议中，受信任的供应商使用他们的私钥(PrK)来生成他们想要部署的代码的签名，并将其与软件二进制文件一起推送到设备。该设备包含供应商的公钥(PuK),可用于验证二进制文件未被修改，以及它是否是由受信任的供应商提供的。

PuK 不需要保密，但是它需要以一种方式存储在设备中，这意味着它不能被属于攻击者的 PuK 替换。

信任链

安全引导过程实现了信任链。从一个隐式信任的组件开始，所有其他组件都可以在执行前进行身份验证。该链的所有权可以在每个阶段改变-属于设备 OEM 的 PuK 可以用于认证第一引导加载程序，但是安全世界 OS 二进制文件可以包括用于认证其加载的应用程序的第二 PuK。

除非设计能够抵御硬件 shack 攻击，否则安全引导过程的基础(称为信任的根源)必须位于片上 ROM 中。SoC ROM 是系统中唯一不能被简单的重新编程攻击轻易修改或替换的组件。

为信任根存储 PuK 可能是有问题的；将其嵌入片上 ROM 意味着所有器件使用相同的 PuK。这使得他们容易受到
如果 PrK 被盗或被成功逆向工程，则类破解攻击。片上一次性可编程(OTP)硬件，如多晶硅熔丝，可用于在器件制造期间在每个 SoC 中存储唯一值。这使得许多不同的 PuK 值能够被存储在一类设备中，降低了类破坏攻击的风险。

——— 注意 ———

OTP 存储器会消耗相当大的硅面积，因此可用的位数通常是有限的。RSA PuK 的长度超过 1024 位，通常太大而不适合可用的 OTP 存储。然而，由于 PuK 不是机密的，所以它可以存储在片外存储器中，只要 PuK 的加密散列存储在 OTP 的片上存储器中。散列比 PuK 本身要小得多(SHA256 散列为 256 位)，并且可以用于在运行时验证 PuK 的值。

片上安全世界还是片外安全世界

对 shack 攻击最简单的防御是将任何安全的世界资源执行保持在片上内存位置。如果代码和数据从未暴露在 SoC 封装之外，窥探或修改数据值将变得非常困难；对 SoC 封装的物理攻击比将逻辑探针连接到 PCB 走线或封装引脚要困难得多。

安全引导代码通常负责将代码加载到片上系统存储器中，正确安排认证顺序以避免给攻击者带来可乘之机至关重要。假设运行代码和所需的加密哈希已经在安全的片上系统存储器中，则在使用加密方法进行身份验证之前，被验证的二进制文件或 PuK 应该被复制到安全的位置。A

验证图像，然后将其复制到安全内存位置的设计会面临攻击风险。攻击者可以在检查完成和复制发生之间的短时间内修改图像。

5.3 监控模式软件

监控模式软件在设计中的作用是提供一个强大的看门人，管理安全和非安全处理器状态之间的切换。在大多数设计中，它的功能将类似于传统的操作系统上下文切换，确保处理器离开的世界的状态被安全地保存，并且处理器切换到的世界的状态被正确地恢复。

正常世界进入监控模式受到严格控制。只有通过以下异常才可能:中断、外部中止或通过 SMC 指令的显式调用。进入监控模式的安全世界稍微灵活一些，除了正常世界可用的异常机制之外，还可以通过直接写入 CPSR 来实现。

监视器是一个安全关键组件，因为它提供了两个世界之间的接口。出于鲁棒性原因，建议监控代码在中断禁用的情况下执行；编写一个可重入的监视器会增加复杂性，并且不太可能比简单的设计提供更大的好处。

5.3.1 上下文开关程序

如前所述，监视器的主要作用是切换两个世界都需要的资源。监视器保存的任何安全状态都应该保存到安全存储器的一个区域中，以便正常世界不能篡改它。

每个交换机需要保存和恢复的具体内容取决于硬件设计和用于世界间通信的软件模型。该列表通常包括:

- 所有通用 ARM 寄存器。
- 任何协处理器寄存器，如氖或 VFP。
  注意:只有在两个世界都使用协处理器时才需要。
- CP15 中任何依赖于世界的处理器配置状态。

硬件异常:IRQ、FIQ、外部中止

当处理器硬件配置为捕获异常(IRQ、FIQ 和外部中止)进行监控时，中断上下文的状态是任意的。

这意味着需要所有状态的完整上下文切换，除非一些状态可以被惰性上下文切换。看见 *Lazy context switching on page 5-10* 了解更多详情。

软件异常:SMC

在许多情况下，设计将使用 SMC 指令作为简单的跨世界产出，这导致以与上述硬件异常类似的方式进行完整的上下文切换。然而，在某些情况下，SMC 驱动的世界交换机在某些处理器寄存器中携带消息有效载荷是有益的。在这些情况下，不需要完全的上下文切换。

使用 SMC 启动的上下文切换和全局共享内存，可以在两个世界之间建立有效的软件通信协议。

惰性上下文切换

一些连接到 ARM 协处理器接口的硬件协处理器，如 ARM VFP 和 NEON 执行单元，可以支持一种称为惰性上下文切换的机制。这允许仅在必要时保存协处理器的上下文，而不是在每个操作系统上下文切换或 TrustZone world 切换时保存。与 VFP 和霓虹单元相关联的状态可能非常大，因此惰性上下文切换可以显著降低平均切换开销。

为了在 TrustZone 系统中实现惰性上下文切换，安全环境可以使用 CP15 非安全访问控制寄存器(NSACR)中的位设置来阻止对每个协处理器接口的正常环境访问。当普通世界或用户模式软件试图使用在 NSACR 中配置为安全的协处理器时，会出现未定义的指令异常。异常必须被正常世界内核捕获，并且处理程序必须向监视器发出 SMC 以请求所需的协处理器上下文切换。一旦协处理器上下文被交换，监视器就可以使用 NSACR 配置使非安全软件能够访问协处理器，并返回到正常的全局处理程序。

——— **注意** ———

许多安全世界的实现将不需要浮点或 SIMD 运算。在安全世界不使用协处理器的设计中，系统引导代码可以使所有协处理器可被非安全软件访问。在这些情况下，协处理器中没有敏感数据，并且监视器代码不需要上下文切换它们的状态。

———————————

5.3.2 中断模式-监视器要求

中概述的中断模型 *Secure interrupts on page 3-11* 在中，IRQ 被配置为正常世界中断，FIQ 被配置为安全世界中断，需要通过世界交换机上的监控软件进行一些核心配置。

该模型提出，当中断发生时，如果处理器已经在正确的世界中执行，则硬件不会陷入监控，而是直接跳到局部世界的向量表。这避免了切换到监视器的开销，并使监视器软件设计更简单。

注意

在捕获监视器异常的设计中，您将捕获到监视器模式向量表中的适当条目；然而，内核将处于监控模式，而不是各自的异常模式。

CP15 中的安全配置寄存器(SCR)包含决定是否将 IRQ、FIQ 或外部异常中断捕获到监控器硬件的设置。为了实现这里提出的模型，监控器需要修改每个世界交换机上的 SCR 的内容。当切换到正常模式时，SCR IRQ 位必须清零，SCR FIQ 位必须置位。当切换到安全世界时，SCR IRQ 位必须置位，SCR FIQ 位必须清零。

注意

这只是在 TrustZone 处理器中使用中断的许多可能模式中的一种。

5.3.3 中断延迟影响

在任何必须切换世界来处理中断的设计中，监控器成为关键路径的一部分，该路径定义了系统的最坏情况中断延迟。与通常部署在深度嵌入式系统中的 ARM R-profile 和 M-profile 处理器不同，部署到 ARM A-profile 应用处理器的软件通常不需要低中断延迟。然而，设计应考虑监控软件带来的任何额外延迟，以确保最差情况下的行为不会违反任何时序限制。

监视器增加的中断开销可能比单个世界交换机的成本还要高。例如，当中断发生时，如果处理器刚刚进入监视器，它将在处理它之前转换通过监视器，如果处理器然后需要切换回另一个世界来处理中断，它将需要执行第二次监视器转换。

在希望将中断延迟影响降至最低的设计中，监控器使用的代码和数据应放在靠近内核的快速存储器中。在使用 ARM1176JZ(F)-S 处理器的系统中，监视器可以放在 TCM 中。在不提供 TCM 的系统中，监视器可以放置在锁定的 L2 高速缓存线或 fast 中
片上 SRAM。

示例中断等待时间增加

堆叠通用寄存器并执行本节前面描述的中断模型所需的 SCR 设置重新配置的简单监控器可能会产生以下开销:

- ARM1176JZ(F)-S:
  - 监控 TCM 中的代码和数据。
  - 上限是每个开关 200 个周期，或总共 400 个周期。
  - 300MHz 下 400 个周期为 1.3us。
- 皮质 A8:
  - 监控位于锁定的 L2 缓存中的代码和数据。
  - 每个交换机的开销大约为 1200 个周期，即总共 2400 个周期。
  - 600MHz 下 2400 个周期为 4us。

注意

总中断延迟将包括由于正常世界软件、安全世界软件和系统设计方面(如外部存储器性能)而产生的开销。这些间接费用没有在这份清单的数字中标明。

相比之下，在 ARM1176JZ(F)-S 上运行的 Linux 驱动程序在没有任何 Secure world 软件的情况下，其中断延迟大约为 5000 个周期。这是由操作系统本身的开销造成的。

5.4 安全软件和多处理器系统

一些希望利用安全执行环境的设计可能基于多处理器 ARM 设计，例如 Cortex-A9 MPCore 处理器。这些系统通过在多个硬件处理器上并发执行线程来实现更高的软件性能，这使得它们非常适合处理软件媒体编解码器处理等密集型任务。

安全世界软件架构师需要做出一个重要的决定——安全世界软件将如何利用多个处理器？

大多数安全系统旨在使用简单的软件设计，因为简单意味着暴露安全漏洞的软件缺陷的风险较小。真正的多线程增加了软件的复杂性，由于它引入了时间敏感性，通常很难测试。由于这个原因，许多安全世界软件实现可能选择实现单处理器安全世界，其很少或不使用 SMP 处理器特征，即使正常世界使用完全 SMP 模式运行。

5.4.1 安全世界处理器关联性

如果选择实现一个安全的世界
多处理则设计需要能够同步 SMP 正常世界和安全世界之间的通信。

设计可能会选择将安全世界执行固定在一个特定的处理器上，这使得安全中断路由简单，但也意味着安全世界可能会使正常世界线程调度效率较低，因为它正在使用正常世界不容易负载平衡的处理器时间。在这种设计中，与安全世界通信的正常世界驱动程序通常需要使用处理器间通信将使用安全世界的请求路由到正确的处理器。此外，安全世界不使用的处理器上的监控软件必须防止正常世界引起恶意世界切换。该架构如所示 Figure 5-3 on page 5-14，其中安全世界软件仅使用 CPU0。

另一种设计可以选择让安全环境在系统中的多个处理器之间迁移，对其进行限制，使其在任何单个时间点都只能在一个处理器上执行。这使得安全世界更有效，因为它可以在与使用它的正常世界应用相同的处理器上运行，并且允许正常世界负载平衡它的调度，但是它使得安全中断到必要处理器的路由更复杂。

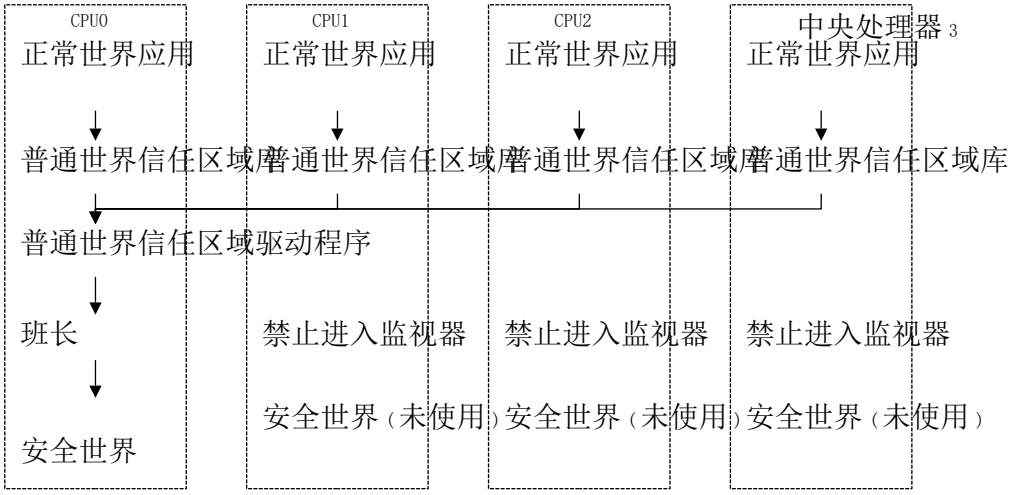


图 5-3 多处理器软件架构示例

5.4.2 安全世界中断使用

在决定安全世界将如何利用 SMP 处理器之后，开发人员必须选择如何将任何安全中断源集成到设计中。

在单个固定处理器上执行安全世界的架构中，中断路由是直接的。每个处理器的中断控制器处理器接口控制寄存器必须进行适当的配置，以使安全中断产生 FIQ 或 IRQ 异常，这取决于软件是如何利用硬件特性的。

多处理器系统中单个处理器的中断路由模型在 *Interrupt model - monitor requirements on page 5-10*。如该节所示，必须处理安全或非安全中断的处理器的监控软件需要能够将它们路由到适当的环境，以便它们可以由正确的异常处理程序处理。

在 SMP 系统中，安全世界在集群内的固定单个处理器上执行，FIQ 中断可由不运行安全世界软件的处理器上的非安全中断使用。为了防止硬件将这些中断路由到安全领域，在这些处理器上执行的安全引导加载程序或监控软件必须确保 CP15 中的安全配置寄存器被适当编程。

注意

在安全世界仅在单个处理器上执行的设计中，通常需要在其他处理器上运行的监控软件来拒绝正常世界使用 SMC 指令、中断或外部中止来切换世界的尝试。

5.5 TrustZone API

为了鼓励安全解决方案的开发，ARM 开发了一个标准化的软件 API，称为 TrustZone API (TZAPI)，它定义了一个软件接口，运行在丰富操作环境中的客户端应用程序可以使用该接口与安全环境进行交互。

API 主要是一种通信 API，使客户端能够向安全服务发送命令请求，并使客户端能够与它所连接的服务有效地交换数据。该通信接口旨在支持全局共享内存的原则，用于高性能批量数据传输。

API 的次要特征允许正常世界的客户端应用程序使用安全服务来认证它们自己，查询已安装服务的属性，并且允许可信的正常世界代码执行新的安全服务的运行时下载。

注意

通信 API 支持同步和(可选)异步调用约定，以支持嵌入式设备上所有常见的标准操作系统。

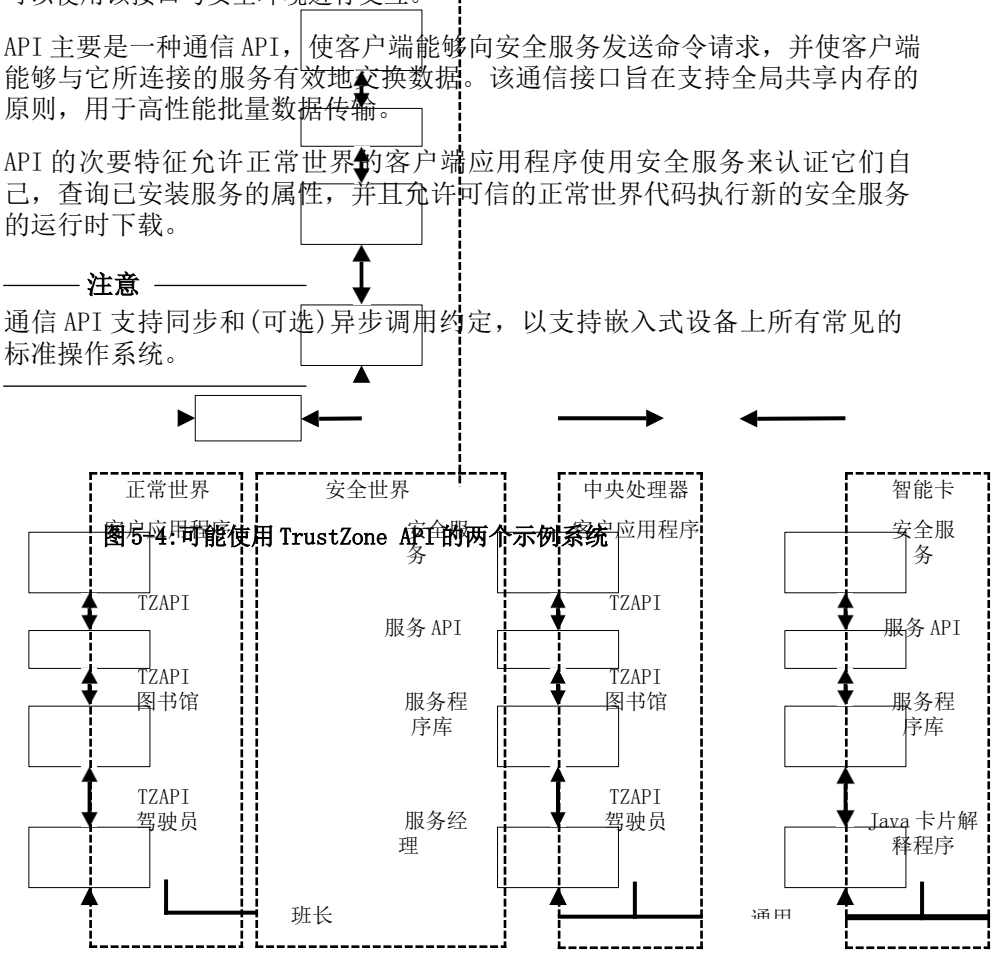


图 5-4: 可能使用 TrustZone API 的两个示例系统

尽管 TrustZone API 的目标是使用支持 TrustZone 的处理器的系统，并试图利用可用的硬件功能，如全局共享内存，但它被设计为可移植到几乎任何安全环境的实现中。使用在智能卡上运行的可扩展安全框架的系统也是 TrustZone API 实现的合适候选。在这些非信任区系统中，TZAPI 全局共享内存构造的实现可能需要一个副本，但是它们应该仍然是可移植的。

5.5.1 API 可用性

ARM 认识到缺乏软件开发的通用标准阻碍了安全软件生态系统的发展，因此发布了 TrustZone API 作为公共规范，任何软件开发人员都可以免费下载和使用，作为其底层安全解决方案的接口。

TrustZone API 可以从 ARM 网站免费下载。看见 http://www.arm.com/TrustZone 了解更多详情。

第六章
TrustZone 系统设计

本章采用了一些可能的安全用例，并为一个假想的移动设备 Gadget2008 创建了一个 SoC 设计。

本章包含以下几节:

6.1 Gadget2008 产品设计简介

Gadget2008 产品旨在提供一种便携式蜂窝手机，具有功能丰富的操作系统，能够执行用户下载的应用程序。

为了支持便携式媒体的当前趋势，手机将支持音频和视频播放。与主要内容提供商的许可证交易需要在软件环境中添加 DRM 内容保护方案。

为了提供额外的收入，将开发配套的内容供应服务 GadgetStore。GadgetStore 将允许用户使用标准的银行设施为下载的内容付费。该设计应旨在降低内容供应服务的卡交易成本，并将消费者遭受欺诈的风险降至最低。

6.2 示例使用案例

如本文档前面章节所述，创建具有必要安全属性的系统需要分析需要防御的资产、平台预计会遇到的攻击者类型以及设计需要提供应对措施的攻击细节。

本节概述了几个用例，本章稍后将使用这些用例来演示如何应用 TrustZone 技术来保护相关的敏感资产。

6.2.1 内容管理

移动设备中许多当前的安全使用案例都与内容管理有关。数字版权管理(DRM)代理的作用是基于用户已经购买的权利来实施对媒体内容的访问。权利的选择是多样的；他们可以根据截止日期、播放次数、回放分钟数和播放设备等来限制用户。当用户选择播放受 DRM 保护的文件时，代理必须确保用户的权限仍然有效，如果有效，则解密该文件并将其传递给媒体播放器。

使用 TrustZone 技术的系统集成

在支持信任区的系统中集成 DRM 代理在架构上相对简单。Gadget2008 中的设计将负责权限存储、权限验证和内容解密的 DRM 代理组件置于安全世界中。该设计将把编解码器和媒体播放器放在正常的环境中。尽管这将解密的内容暴露在一定程度的风险中，但是可以使用诸如反病毒的机制来检测对正常世界操作系统和媒体播放器的未授权改变，此时安全世界可以停止向正常世界发送内容。

将整个软件编解码器堆栈置于安全环境中是可能的，但由于典型软件实现的大小和复杂性，会引发其他安全问题。从统计数据来看，较大的软件比简单紧凑的软件环境有更多的安全漏洞，并可能威胁到更敏感资产的安全。

注意

使用外部安全世界硬件块，例如 DSP 或专用加速器，可以用来解码内容文件，而不会将它们暴露给正常世界。诸如主 ID 过滤之类的附加硬件保护可以用于将复杂的编解码器硬件与它不需要访问的安全世界资产隔离。

资产

下表列出了与 DRM 方案相关的资产:

表 6-1:示例 DRM 方案中的资产

资产	安全性	描述
权限数据	真实性	权限数据包含关于用户权限的信息；例如，剩余的播放次数。 必须能够在运行时检查权限对象的真实性。
权利检查代码	真实性	检查权限对象的代码必须是可信的，并且从安全的位置执行。如果攻击者可以修改用于权限检查的代码以总是返回 valid，那么 DRM 可以被轻易绕过。
设备密钥	保密性、完整性	设备密钥是根秘密，当从内容存储器传输时，用于解密权限数据。 设备密钥必须保密。如果攻击者能够窃取设备密钥，他们就能够解密所有权限对象以获得所有内容密钥。这将允许攻击者解密所有存储的内容文件。
内容密钥	机密	内容密钥是存储在权利对象中的秘密密钥，用于在回放之前解密特定的内容。 如果攻击者能够窃取内容密钥，他们就可以解密与之相关的单个内容。
内容数据	机密	内容数据是传递给媒体播放器的解密内容。 系统完整性监控和 Normal world 防病毒可用于降低被盗风险。

攻击者

与 DRM 方案相关联的主要攻击者是设备所有者；他们希望获得免费的内容和服务。

针对桌面个人计算机上的内容保护方案的大多数攻击是软件黑客攻击，因此我们必须防御恶意软件。用户还可以实际接触设备，这意味着简单的硬件攻击，如重新编程攻击，必须构成风险分析的一部分。

6.2.2 移动支付

许多嵌入式设备现在存储着大量用户数据，包括同步电子邮件、移动银行详情和移动支付凭证等敏感信息。这些用户数据可以得到保护，需要输入密码才能使用，但是一旦解锁，就容易受到底层软件环境中任何弱点的攻击。

对于许多使用用户数据的应用程序来说，将数据存储、数据操作甚至密码输入迁移到安全的环境中是有意义的。尽管所有这些用例都有微妙的不同资产，但它们都有相似的安全需求。出于这个例子的目的，Gadget2008 将使用移动支付，这比大多数其他用例有更严格的要求。

概观

有两种常见的支付模式:持卡人在场和持卡人不在场。持卡人出示支付需要某种形式的认证，以显示用户和他们的卡都存在，通常是用户签名或芯片和 PIN PIN 输入。持卡人不在场交易，例如在网上购物时使用的交易，不能验证卡，因此具有更高的欺诈程度。持卡人不出席交易可能会使商家花费更多的交易费用，并且如果卡交易被证明是欺诈性的，还可能承担对商家收取退款罚款的重大风险。

为了满足降低交易成本和欺诈风险的 Gadget2008 设计概要，希望设备能够支持远程持卡人出席交易。为此，设计必须能够证明用户和他们的卡都存在。

注意

任何新支付方案的部署都需要发卡机构和卡组织的批准。这个示例只是试图展示支持 TrustZone 的系统的一些可能性。

使用 TrustZone 技术的系统集成

支持 TrustZone 的 SoC 提供了一个平台，允许部署可访问可信键盘和可信显示器的可信软件。使用这项技术，支付应用程序可以安全地显示远程购买的详细信息，用户可以在可信的键盘上输入密码，以批准屏幕上显示的交易。

为了提高安全性，可以将通行码和交易信息传递给智能卡，例如芯片密码银行卡。运行在银行卡上的软件可以验证用户密码的正确性，并且如果密码正确，则生成对远程交易的签名批准。安全用户密码和只能来自用户银行卡的交易批准的组合，允许设备实现比持卡人不在场交易高得多的安全级别，并开始接近现有持卡人在场交易机制所实现的级别。与 Gadget2008 手机相比，智能卡可以提供更高水平的防篡改能力，使其适合存储根支付凭证，但需要依靠手机来提供安全的用户界面。

手机硬件不具备 FIPS 140-2 认证的安全性
芯片和 PIN 终端，需要一定程度的防篡改能力，但用户不太可能对自己的设备进行物理修改来窃取自己的银行密码。

——— **注意** ———————
支付基础设施涉及的安全世界硬件和软件可能需要经过安全评估过程，以满足支付方案的要求。
————————————

资产

下表列出了与付款方案相关的资产:

表 6-2:示例付款方案中的资产

资产	安全性	描述
交易信息	真实性	当用户批准远程交易时，重要的是他们确信他们批准的是正确的信息。让用户认为他们已经同意购买 10 美元的书，但后来却发现他们为一块假表支付了 1000 美元，这将严重损害消费者对支付方案的信心。 这就要求用户实际认可的信息，也就是他们在设备显示屏上看到的信息，必须来自可信的来源。
用户和卡信息	机密	用户的卡的详细信息，例如他们的卡号和发行日期，以及其他用户的详细信息，例如他们的家庭地址，对于试图执行持卡人不在场交易的攻击者来说都是有价值的。 这些资产必须得到保护，因为持卡人不在场欺诈是当今市场上增长最快的金融欺诈之一。
用户密码	机密	如果攻击者也可以窃取手机和相应的银行卡，那么用户的密码就是有价值的。密码的值不能存储在手机上，否则它可能被攻击者恢复，然后攻击者可以执行欺诈性的持卡人出示交易。
根支付凭证	机密	用于批准交易的根加密秘密必须保密，否则攻击者可以批准任意的欺诈性交易。 这些根秘密的安全性对于整个支付方案的安全性至关重要，并且在本例中，将依赖于银行级智能卡提供的安全性。

攻击者

如果支付方案的安全性可以被破坏，那么可以实施的欺诈就值很多钱。因此，支付方案吸引了大量的攻击者，从攻击硬件的人，到试图窃取用户信息的电子邮件诈骗者。大多数针对这些设备的攻击都是远程的，这意味着
纯软件黑客攻击，但被盗设备和送去维修的设备也可能受到物理攻击。

6.3 Gadget2008 规格

既然有了用例的概要，以及所涉及的安全资产，就有可能开始设计一个满足需求的平台。

6.3.1 通用规范

有许多需求并不特定于上一节中描述的用例。本节涵盖了这些通用要求。

安全启动

如中所述 *Booting a secure system on page 5-5* 攻击移动设备的常见方法之一是用修改后的副本对设备固件进行重新编程。需要一个安全的引导实现来防止这种情况。

Gadget2008 将包括:

- 位于片上 ROM 中的安全引导代码，
- 256 位 OTP fuse，它可以包含 Secure world 软件开发人员拥有的 RSA 公钥的 SHA256 散列，
- 和位于片上加密加速器中的统计上唯一的密钥。

安全引导调查表明，用于图像解密和签名验证的足够的软件加密可以放入大约 4KB 的 ROM 中。此外，还需要 8KB 的片上 RAM 来包含任何动态引导状态，否则安全引导秘密很容易受到外部存储器接口上的窥探攻击。

使用 OTP fuse 存储 RSA 公钥的加密哈希意味着我们可以分发具有各种根密钥的设备，从而降低成功的类破解攻击的风险。

在片上系统加密硬件中包含只有安全软件才能使用的统计唯一秘密，意味着机密数据可以加密并绑定到设备。恢复密钥的唯一机制是执行
加密的旁道分析，这是硬件加速器设计应该防止的，或者拆除硅以恢复密钥。考虑到 Gadget2008 包含的资产的价值，这可能太昂贵了，所以这是一个很好的安全对策。

多任务安全世界软件

Gadget2008 的设计需要安全世界中的多个软件:一个用于管理从 GadgetStore 购买的内容的 DRM 代理，一个安全的 GadgetStore 购物后端，以及一个由用户的发卡方提供的支付应用程序。

在现实世界的场景中，提供这种软件的利益相关者不太可能真正足够信任彼此而不存在某种形式的隔离。由于这个原因，许多安全世界软件栈将需要包括所有利益相关者信任的特权内核组件，并且这个内核必须在运行的安全服务之间提供健壮的隔离。

Gadget2008 将包括:

- 使用 MMU 分离用户任务的安全世界操作系统。

保护调试通道

Gadget2008 的调试方案将把生产设备分成两批。第一批将是一个小的开发运行，将允许全面的调试访问，包括安全世界的访问。这些设备将仅被运送到可信方，并且 OTP 熔丝将被设置为已知的无效密钥。第二批设备将启用正常世界调试，但使用保险丝永久禁用所有安全世界调试。

其他系统方面

本文没有涵盖 SoC 设计的许多方面。这些领域包括多时钟域、多电源域和省电状态的使用，例如系统休眠，在休眠状态下 SoC 的电源被移除。

所有这些都可能对系统的安全性产生影响，这取决于用例以及所涉及的资产。例如，电源控制器可能能够关闭执行后台安全检查的 SoC 区域，而时钟频率的变化可能会将加密算法暴露给旁路分析。任何设计都必须审查并考虑任何系统设计方面的可能影响，以及在给定所需安全级别的情况下，它们如何影响整个系统。

电源管理尤其是安全世界软件可能需要提供安全协议实现的一个领域。许多现代手机 SoC 积极省电，关闭部分 SoC，同时仍给用户设备完全活跃的印象。在这些设计中，系统状态保存在外部 DRAM 中，然后进入状态保持低功耗模式，然后关闭 SoC。上电时，电源控制器通知设备

从休眠或热启动中返回，设备从外部 DRAM 恢复运行状态。如果这种系统中的安全世界使用片上存储器来存储其资源，则在安全性分析中必须考虑休眠和热启动周期。使用片上存储器的主要原因是防止窥探攻击，因此这意味着存储在外部存储器中的休眠状态的内容必须加密。如果设计还与重放攻击有关，在重放攻击中，攻击者试图从旧的休眠状态恢复，则可能需要包含一个
加密状态下的非易失性计数值。

6.3.2 内容管理规范

内容管理方案有以下要求:

软件视频解码

Gadget2008 性能最强的任务是处理用户可以从 GadgetStore 购买的视频文件的软件解码。媒体编解码器算法相对来说是处理器密集型的，因此设计将使用单处理器 Cortex-A9 处理器来实现 NEON 媒体处理引擎。

总之，该设备将包括:

- ARM Cortex-A9 处理器。
- AMBA3 AXI 高性能总线矩阵(PL301)。

注意

支持 TrustZone 的处理器所需的总处理器带宽是正常世界处理要求和安全世界处理要求的总和。在这种情况下，必须有足够的处理器频率和存储器带宽来处理媒体编解码器和 DRM 解密任务的并发执行。

软实时性能

任何涉及流媒体的用例都需要来自媒体处理中涉及的任何组件的软实时响应，包括 DRM 代理。如果内容在需要时没有被提供给媒体编解码器，则输出的质量将会降低，并且最终用户的体验将会很差。

为了支持这些要求，安全世界软件将是多任务的
抢占式操作系统，使用安全中断源来抢占处理器上运行的其他任务。这使得更高优先级的 DRM 任务能够被有效地

预定的，并且应该确保编解码器不会用完要解码的内容。此外，与视频回放相关的数据速率被判断为需要系统中的一级高速缓存。

Gadget2008 将包括:

- 一个 ARM 双定时器模块(SP804)。
- PrimeCell 通用中断控制器(PL390)。
- 一个 PrimeCell 二级高速缓存控制器(PL310)。

非易失计数器

内容管理系统易受的一种攻击是重放攻击，在重放攻击中，攻击者用包含过期权限对象的旧版本替换包含过期权限对象的文件系统映像。即使加密文件系统映像本身也不是一种保护；攻击者可以简单地交换加密图像，甚至不需要破坏加密保护。

将非易失性计数器添加到设计中允许将硬件强制版本计数嵌入到存储的数据中。围绕该计数器值构建由强加密措施实施的安全存储协议可以使系统能够识别和防止重放攻击。

Gadget2008 将包括:

- 32 位片上非易失性计数器。

安全实时时钟源

许多内容方案实现某种形式的基于时间的权利发布，因此要求平台支持不能被普通软件修改的实时时钟(RTC)。

将设备上的 RTC 与由后端内容服务器管理的更可信的时间同步，使得设备上的 RTC 能够针对微小的时钟漂移进行校正。该设计还将把最后接收的服务器时间存储在由非易失性计数器保护的文件中，以便它还可以防止重放攻击。

Gadget2008 将包括:

- 一个安全的世界实时时钟。

内存要求

许多内容管理方案都很复杂，需要大量的安全代码和数据来实现完整的代理。包含所有代码和数据的足够片上内存的成本会令人望而却步，因此 Gadget2008 将使用 TrustZone 地址空间控制器来划分单个片外 DRAM。

Gadget2008 将包括:

- TrustZone 地址空间控制器(PL380)

6.3.3 移动支付规范

移动支付对设备有以下要求:

支付子系统

对支付系统的安全要求通常比对内容管理代理的要求高得多。为了提供针对硬件攻击的额外保护，希望安全世界代码和与支付相关的数据以及将支付与其他安全世界服务分开的安全内核代码和数据位于片上 RAM 中而不是片外 RAM 中。

初步调查表明，运行一个轻量级安全世界内核(带有少量安全服务)所需的代码和数据可以放入一个 96KB 的片上 RAM 中。使用 MMU，安全世界内核可以将更大的片外存储器用于不太重要的用例，如内容管理。Gadget2008 设计将包括一个 128KB 的 SRAM，并使用 TrustZone 内存适配器来确保底部 96KB 的安全。这剩下 32KB 可用于普通软件。

为了在设计中实现一定的灵活性，我们将使用 TrustZone 保护控制器向 TrustZone 内存适配器提供输入，使分区大小在启动时处于最佳位置。

Gadget2008 将包括:

- 128KB 的片上 SRAM，
- PrimeCell 基础设施 AMBA3 TrustZone 存储器适配器(BP141)。
- PrimeCell 基础设施 AMBA3 TrustZone 保护控制器(BP147)。

安全键盘

Gadget2008 设计中的外部键盘控制器使用 PS/2 协议与主 SoC 通信，因此我们将使用标准 APB 外设来处理这一问题。该设计将使用 TrustZone 保护控制器来设置 AXI 到 APB 桥内外设的安全设置，使小键盘能够在运行时进出安全世界。

Gadget2008 将包括:

- PrimeCell PS/2 键盘/鼠标接口(PL050)

安全显示

Gadget2008 设计使用简单的帧缓冲图形，其中显示控制器每帧将帧缓冲存储器的内容复制到外部 LCD 接口。

为了支持安全显示器，硬件设计将包括两个帧缓冲器，一个安全，一个不安全。显示控制器将读取每个帧的两个帧缓冲器的内容，并将安全显示器的可见部分覆盖在非安全显示器的顶部。这使得显示具有受保护的完整性，这是支付交易批准屏幕所需的重要安全属性。

Gadget2008 将包括:

- 自定义显示控制器

近场通信接口

为了经济地实现类似于持卡人出示交易的安全级别，手机必须能够与银行卡通信。这将通过在设备中包括近场通信(NFC)读卡器接口来实现，允许它与实现无源 NFC 卡接口的银行卡通信。

支持我们支付方案的银行和信用卡公司需要在他们的卡上部署一个应用程序。安全世界软件服务可以与该智能卡应用通信，以批准支付交易。安全世界服务将管理对键盘和安全显示器的访问，但银行卡应用程序负责使用用户的密码来实际批准金融交易。

6.3.4 将硬件组装在一起

将本节讨论的所有特性放在一起，我们可以制作出以下 SoC 设计:

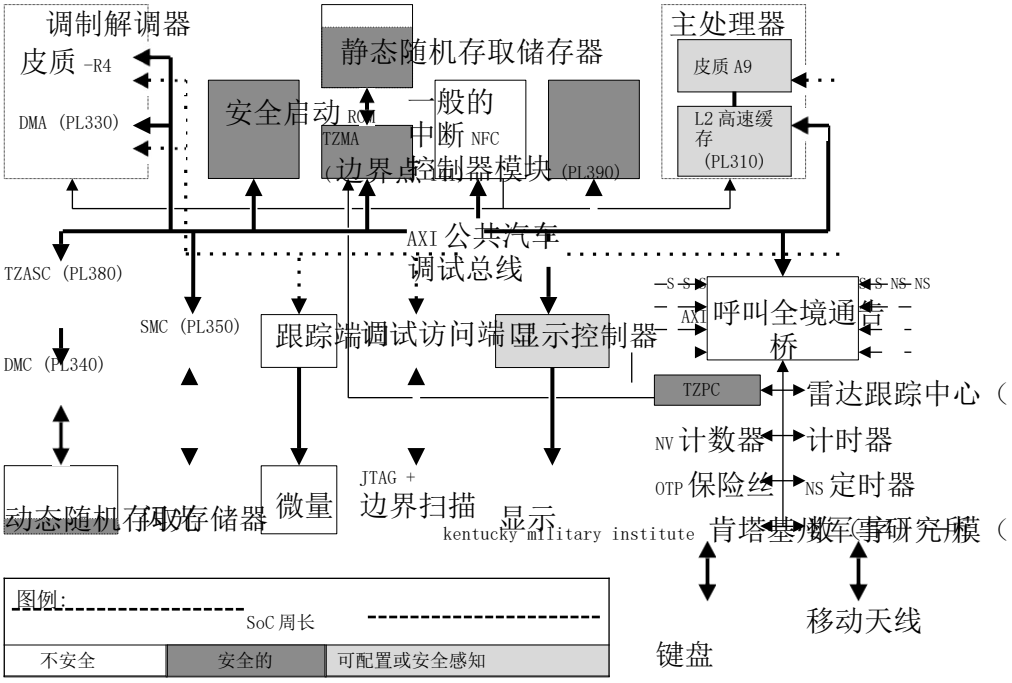


图 6-1:gadget 2008 SoC 设计

注意

请注意，此设计仅显示了与系统安全性相关或重叠的方面。大多数普通外设和所有系统外设，如电源控制器，都没有显示。

将软件组装在一起

软件体系结构可以类似地概括出来，如 Figure 6-2 onpage 6-16。在这个设计中，安全世界是围绕微内核实现的
原则，并将关键系统组件(如设备驱动程序)推到独特的用户空间任务中。

安全世界中安装了三个独立的安全服务，分别用于 DRM、GadgetStore 和移动支付用例。这些服务负责管理与每个用例相关的所有敏感资产，并通过安全设备驱动程序利用硬件来充分保护它们。

普通世界使用 TrustZone API 与安全世界通信，并实现一个驱动程序组件，该组件管理来自普通世界客户端应用程序(如媒体播放器和 GadgetStore 前端)的并发通信会话。

最后，安全世界能够使用 NFC 接口与运行在银行卡上的支付应用程序进行通信。这与设备上的支付服务协同工作，以批准金融交易。

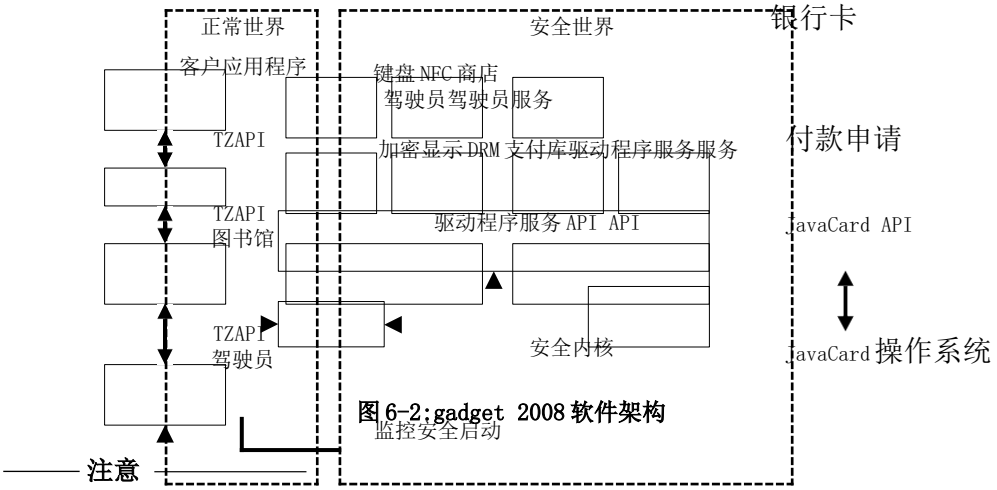


图 6-2:gadget 2008 软件架构

注意

为了允许普通软件利用设备的外部通信接口，例如 NFC 或 GPRS，设计者可以选择使较低级别的通信协议栈在普通环境中运行。

可以使用加密方法在安全世界软件和通信链路另一端的设备之间创建安全隧道，而不会将通信内容暴露给不太可信的正常世界。

第七章
设计清单

本章提供了一些清单，在使用 TrustZone 技术设计 SoC 时，或者在审查安全设施的设计时，可以使用这些清单。

本章包含以下几节:

7.1 用例清单

该清单涵盖了设计阶段的第一部分；识别用例、资产和威胁。

- SoC 有哪些使用案例？
- 每个用例中存在的资产是什么？
- 资产需要什么安全属性，例如机密性或完整性？
- 每个资产的预期攻击者是谁？
- 每个攻击者能够使用哪些攻击方法？
- 谁是安全利益相关者，他们信任谁？

7.2 硬件设计清单

该清单涵盖了设计阶段的第二部分；确定基本硬件要求:

- 资产存储和处理在系统的什么位置，相应的硬件组件是如何保护的？

  资产的位置是否与系统上实施的物理硬件隔离屏障(如 SoC 封装)充分匹配？

- 属于每个利益相关者的资产是否受到保护，免受其他利益相关者的攻击？
- 任何总线从设备、网桥或自定义逻辑的安全配置(包括主过滤)是否正确？
- 任何 APB 外围设备的安全配置是否正确，包括较大组件(如 AXI 内存控制器)的 APB 配置端口？
- 设计中使用的秘密值是否足够独特以防止类中断攻击？

  是否应考虑使用一次性可编程资源(如金属层保险丝)来安装每个设备的标识？

- 是否应该将特定资产保留在 SoC 上以增强对 shack 攻击的防护？

  这种分析应该包括不太有形的资产，例如对片上系统信号的控制，如处理器调试使能。如果从 SoC 外部控制调试启用，攻击者将很容易操纵 SoC 引脚输入来启用调试。

  是否有足够的片上内存来存储运行所需的软件代码和数据？8KB 的 SRAM 足够用于安全引导和监控，但如果在安全环境中需要类似操作系统的功能来分隔多个并发任务或利益相关者，则应考虑 96KB。

- 设计是固定功能，还是配置的动态控制有用？

  如果对硬件块安全性的动态控制是有用的，那么考虑使用 TZMA、TZPC 或 TZASC 来允许安全配置的引导时间或运行时间配置。

- 如何保护侵入式调试和内置自测基础设施免受恶意使用？
- 如何保护非侵入式跟踪调试免受恶意使用？

- 需要能够在现场重新启用调试吗？这种机制是如何保护的？
- 系统组件，如时钟控制器和电源控制器，是否得到适当的保护？
- **如果使用 ARM 处理器的 CP15SDISABLE 信号，是如何保护的？**

7.2.1 多处理器设计

核对表的这一部分提出了一些附加问题，这些问题仅与利用 ARM 内核实现多处理器扩展的设计相关。

- 集群中每个处理器的调试控制信号是否安全？
- 如果使用 ARM 处理器的集群范围 CFGSDISABLE 信号，如何保护它？
- 给定预期的软件使用模型，安全中断能被路由到正确的处理器吗？

  给定预期的使用模式，传统安全 nFIQ 中断(如果有)是否被路由到正确的处理器？

7.3 软件设计清单

该清单涵盖了设计阶段的第三部分、软件要求以及它们可能对硬件设计产生的影响。

- 在安全世界中，设计现在或将来是否需要一种以上的服务？

  如果不是，那么单线程库是一个合适的设计，否则考虑一个服务任务之间 MMU 分离的安全世界操作系统。

- 安全世界的任何部分都需要处理软实时任务吗，比如媒体处理？

  如果是，那么考虑让安全世界调度抢占。这就要求设计中包含一个用于安全领域的硬件中断源。

- 安全世界或正常世界的任何部分对中断延迟敏感吗？

  如果是，则考虑将监控代码和数据放入 TCM(如果可用)、锁定的高速缓存行或快速片上 SRAM 中。

- 安全世界或正常世界的任何部分需要硬实时行为吗？

  如果是的话，那么这两个世界的设计以及它们的交互都需要重新审核，以确保任何禁用中断的路径都有足够的时间限制。

- 监视器软件是否正确地对通过它的每个可能路径的所有适当状态进行上下文切换？

- 只有监控软件修改 CP15 SCR 中的 NS 位吗？

- 安全世界关心其执行的直接或间接的正常世界可见性吗？

  如果是，那么考虑模糊中断计时，禁止对性能计数器的非安全访问，并在 world switch 上对关键地址范围执行选择性高速缓存维护。

7.3.1 多处理器设计

核对表的这一部分提出了一些附加问题，这些问题仅与利用 ARM 内核实现多处理器扩展的设计相关。

- 软件是否需要利用 SMP 集群，或者是否应该只在单个处理器上运行，以便从更简单的软件设计中获益？

- 如果软件是 SMP，或者是单线程的，但是根据处理器的使用情况进行迁移，那么中断是如何路由的，以确保它们能够被可靠地获取？
- 如果软件是单线程的，并且只在集群中的特定处理器上运行，那么在其他处理器上运行的监控软件是否能正确拒绝非安全软件执行全局切换的尝试？

  这应该包括尝试使用 SMC 指令进行切换、中断和外部中止。