SegTEE：面向小型端侧设备的可信执行环境系统

杜冬冬[1)] 杨璧丞[1)] 余 炀[1)] 夏虞斌[1)] 丁佐华[2)]
赵永望[3)] 张 磊[4)] 臧斌宇[1)] 陈海波[1)]
[1)](上海交通大学电子信息与电气工程学院软件学院 上海 200240)
[2)](浙江理工大学信息学院 杭州 310018)
[3)](浙江大学网络空间安全学院 杭州 310007)
[4)](南湖实验室 浙江 嘉兴 314001)

摘 要 面向万物互联、智联计算的边端场景，如何在小型端侧设备上保护用户隐私，隔离关键代码和数据，成为一个亟待突破的重要问题。现有系统通常依赖于可信执行环境，通过基于处理器的硬件扩展，保护安全敏感应用的机密性和完整性。然而，现有端侧可信执行环境系统主要面向静态、固定的安全场景，难以满足万物互联所带来的动态复杂的安全要求。具体来说，包含四个关键挑战。首先，动态复杂的安全需求会在可信执行环境中带来不可忽视的“资源税”，导致其难以部署在小型端侧设备中。其次，在内存安全方面，现有端侧设备往往只提供简单的段隔离机制（如 ARM MPU 和 RISC-V PMP），难以支持多层多域的复杂隔离需求。再次，在 I/O 安全方面，现有系统通过静态划分或主机代理的方式，前者难以适应动态变化的安全应用场景，后者存在严重性能开销和安全隐患。最后，在可扩展性方面，端侧设备依赖的段隔离机制能够降低硬件资源开销，但是仅能支持十分有限的隔离域，无法满足万物互联场景下较多的隔离域需求。为了系统性地突破并解决上述挑战，本文提出 SegTEE，一个面向万物互联小型端侧设备的可信执行环境系统。和传统可信执行环境方案相比，SegTEE 围绕段隔离机制设计了全系统的隔离和保护，支持同特权态隔离域间隔离和跨特权态的段隔离。具体来说，SegTEE 首先提出嵌套段隔离机制，在硬件层面支持 TEE-Seg 段保护机制和 OS-Seg 段保护机制，其中 TEE-Seg 能够实现隔离域间隔离，而 OS-Seg 则提供用户态和特权态操作系统间的隔离性保障。基于 TEE-Seg 和 OS-Seg 的嵌套段隔离机制，SegTEE 引入了段滑动窗口设计，能够在有限数量（例如 16 个）的段寄存器基础上，实现上百个隔离域，有效支撑万物互联的复杂场景。SegTEE还引入了基于段的内存裁剪机制，有效降低资源税，并且设计了基于段隔离的 I/O 动态保护方案。实验结果显示，SegTEE 能够基于本文提出的设计，相比前沿 RISC-V 可信执行环境（蓬莱-PMP），隔离域数量提升了 14 倍，降低了 54% 的内存资源占用，并且可以在运行时达到相当的性能表现。

关键词 操作系统；可信执行环境；RISC-V
中图法分类号 TP316 **DOI 号** 10.11897/SP.J.1016.2025.00188

SegTEE: Trusted Execution Environment for Lightweight Edge Devices

DU Dong-Dong[1)] YANG Bi-Cheng[1)] YU Yang[1)] XIA Yu-Bin[1)] DING Zuo-Hua[2)]
ZHAO Yong-Wang[3)] ZHANG Lei[4)] ZANG Bin-Yu[1)] CHEN Hai-Bo[1)]
[1)](School of Software, SEIEE, Shanghai Jiao Tong University, Shanghai 200240)
[2)](School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018)
[3)](School of Cyber Science and Technology, Zhejiang University, Hangzhou 310007)
[4)](Nanhu Laboratory, Jiaxing, Zhejiang 314001)

Abstract In the context of interconnected IoT and intelligent computing at the edge, safeguarding user privacy and isolating critical code and data on lightweight edge devices has become an urgent and

收稿日期：2024-01-17；在线发布日期：2024-09-24。本课题得到国家自然科学基金重点项目（62132014）、国家杰出青年科学基金项目（61925206）和国家重点研发计划（2022YFB4501500，2022YFB4501502）资助。**杜冬冬**，博士，助理研究员，硕士生导师，中国计算机学会（CCF）专业会员，主要研究领域为操作系统和体系结构。E-mail：dd_nirvana@sjtu.edu.cn. **杨璧丞**，博士研究生，主要研究方向为操作系统与 RISC-V 体系结构。**余 炀**（通信作者），博士，助理研究员，主要研究领域为语言虚拟机和系统安全。E-mail：yu_y@sjtu.edu.cn。**夏虞斌**，博士，教授，博士生导师，中国计算机学会（CCF）高级会员，主要研究领域为操作系统、系统架构、云计算、系统安全等。**丁佐华**，博士，教授，主要研究领域为软件工程、计算机理论。**赵永望**，博士，教授，主要研究领域为操作系统安全、形式化验证、编程语言原理等。**张 磊**，博士，主要研究领域为系统安全。**臧斌宇**，博士，教授，主要研究领域为操作系统。**陈海波**，博士，教授，主要研究领域为操作系统。

critical issue. Existing systems often rely on trusted execution environments (TEEs) that protect the confidentiality and integrity of security-sensitive applications through hardware extensions embedded in processors. However, applying TEEs effectively on lightweight edge devices in the IoT realm remains a challenge. Specifically, four key problems and challenges exist. Firstly, the use of trusted execution environment with dynamic security requirements imposes a non-negligible resource tax, making it difficult to deploy on lightweight edge devices. Secondly, existing edge devices often only provide simple segmentation mechanisms (such as ARM MPU and RISC-V PMP), which are insufficient to support complex multi-level and multi-domain isolation requirements that are necessary for nowadays IoT applications. Moreover, in term of I/O protection, existing methods usually adopt static partition or host-proxy approach that cannot support dynamic security requirements and may incur high costs. Lastly, while segmentation mechanisms can adapt well to resource-constrained demands, they are unable to meet the growing number of isolation domain requirements in the IoT context. To systematically overcome these challenges, this paper proposes SegTEE: a trusted execution environment system tailored for small edge devices in the IoT landscape. Compared to traditional TEE solutions, SegTEE introduces a comprehensive system-level isolation and protection design based on segmentation mechanisms. It supports both isolation between privilege levels and isolation between applications with the same privilege level, utilizing segment-based methods. Specifically, SegTEE first introduces a nested segmentation mechanism that provides TEE-Seg and OS-Seg segment protections at the hardware level. TEE-Seg enables isolation between different domains, while OS-Seg ensures isolation between user-mode and the privilege-mode operating system. By using the nested segmentation mechanism of TEE-Seg and OS-Seg, SegTEE further incorporates a sliding window design, enabling the implementation of multiple isolation domains on a limited number (e.g., 16) of segment registers. Specifically, SegTEE allows one TEE-Seg memory region to be shared by multiple domains (or enclaves). Besides, SegTEE also introduces a segment-based memory trimming mechanism to effectively reduce the resource tax. Moreover, SegTEE utilizes a segment-based dynamic I/O isolation method to protect I/O devices without significant hardware resources. Overall, the design of SegTEE can effectively support the complexities of the interconnected IoT scenarios. Experimental results demonstrate that SegTEE, based on the proposed design, achieves performance comparable to cutting-edge RISC-V trusted execution environments (e.g., Penglai-PMP), while significantly increasing the number of isolation domains by 14x and reducing memory resource consumption by 54%.

Keywords　operating system; trusted execution environment; RISC-V

1　引　言

越来越多的小型边端设备已经具备了智能算力，边端场景正逐渐形成万物互联、智联计算的形态[1-5]。例如，OpenHarmony① 操作系统能够部署在摄像头、音响、空调等端侧设备内，并基于其软总线的能力，实现跨设备的协同和计算任务的流转。

然而，万物智联的计算场景同时对安全性提出了更高的要求和更大的挑战。相比数据中心、云平台等场景的服务器设备通常部署在封闭集中的机房内，边端设备部署在更加开放的环境中，面临更加多元且复杂的安全风险。

为了有效保护敏感的数据和用户隐私，传统端侧设备通过引入可信执行环境[6-22]，如 ARM TrustZone[5,8-10]、Intel SGX[6]等，结合硬件机制为安全敏感的应用程序提供一个强隔离的执行环境，并保障

① OpenAtom OpenHarmony。https://www.openharmony.cn/2024,08,30

该隔离域的机密性和完整性。相比其它系统安全方案,可信执行环境的可信基更小(通常仅包括硬件CPU),并且能够结合硬件特性实现极高的安全保障。这些优势使得可信执行环境已经成为当前最主流的移动端安全技术之一。例如,Android 默认要求设备支持 ARM TrustZone[23],并且基于 TrustZone 支持人脸识别、指纹验证、支付等重要场景。

然而,现有可信执行环境系统在端侧,主要支持资源相对较多的"大型"移动端设备,如手机、笔记本等。与之相对的,万物智联终端场景通常是资源受限的小型设备。分析发现,当前可信执行环境系统设计无法直接应用于小型端侧设备,具体来说,存在以下关键的问题与挑战。

首先,小型端侧设备可信执行环境面临资源税挑战。可信执行环境正从早期的硬件固化实现演变到软硬件协同设计,从而应对可能涌现的攻击和支持新的特性。然而,软硬件协同可信执行环境系统在硬件层面仅实现基本的安全原语,由软件可信基实现完整的可信执行环境抽象和保护。例如,ARM TrustZone[23]在硬件层面仅提供了安全世界和非安全世界硬件隔离域,具体的安全能力、跨域协同等机制依赖于运行在最高特权态(EL3)的监控器和可信执行环境操作系统(TEE-OS)实现。RISC-V 可信执行环境蓬莱[15]和 Keystone[17]通过运行在最高特权态(机器态)的安全监控器,来实现可信执行环境原语。然而,软硬件协同的方式会导致较为显著的计算和内存资源开销,即资源税。ARM TrustZone 和 Keystone 的 TEE-OS 需要上百 KB 甚至 MB 级别的内存资源占用,这在小型端侧设备上是较难承受的。

其次,小型端侧设备缺乏支持多层级多域隔离的能力。当前可信执行环境通常结合多种隔离机制,包括内存段隔离、页表(或 MMU)、缓存划分等,来实现灵活多样的隔离效果。例如,ARM TrustZone 最底层,依赖于一组段寄存器,将内存段划分为可信和不可信区域,其中仅安全世界可以访问可信内存区域。在该粗粒度内存划分之上,TrustZone 结合内部的 TEE-OS 和页表等机制,实现安全应用(Trusted Application,TA)之间的细粒度隔离和保护。与之相对的,当前小型端侧设备由于硬件设计和资源限制,通常没有任何内存隔离(例如,内核和应用运行在同一个地址空间)或仅支持简单的段隔离(如 ARM MPU),难以满足机密计算场景下,内核和应用间隔离,应用和应用间隔离,隔离域之间隔离等多种隔离需求。

再次,端侧设备缺乏灵活动态的 I/O 保护方案。当前端侧可信执行环境系统通常采用静态划分的机制,例如 TrustZone,来隔离可信和不可信 I/O 设备。在万物互联场景下,安全应用对 I/O 设备的需求可能动态变化,静态的方案不仅会导致设备资源浪费,同时也难以满足应用需求。

最后,小型端侧设备难以满足万物智联场景对隔离域高可扩展的需求。相比传统的端侧可信执行环境下安全应用的类型和数量相对固定,万物智联场景下设备间需要和多种类型和不定数量的远端设备进行连接和协同,这要求可信执行环境能够支持较高数量(数十甚至上百)的隔离域。例如,在智能家居场景下,摄像头可能会同时连接音响、手机、手表、空调等等设备,并且对于每一个连接建立对应的隔离域和程序来处理敏感信息的流转,并协同计算。当前主流的实现较高可扩展性的解决方案是引入较为复杂的硬件设计[5,24-28]或引入可信执行环境操作系统[15,17-18,29-32]在单个隔离域内进行二次隔离。这两个方案会导致显著的硬件或软件设计及资源开销,在小型端侧设备上难以应用。

为了解决上述挑战,填补万物智联场景下小型设备可信执行环境系统的空缺,本文提出了 SegTEE,一个面向万物互联等复杂场景的小型端侧设备可信执行环境。相比现有可信执行环境系统,SegTEE 完全基于段隔离机制,实现全系统的隔离与保护,并且能够在资源、性能和安全性上实现显著的突破。

具体来说,SegTEE 包含四部分技术点。首先,SegTEE 提出了嵌套段隔离保护机制。相比传统的扁平串行化的段隔离,SegTEE 中引入了两层嵌套的内存段保护,分别为 TEE-Seg 和 OS-Seg。两层段隔离分别由可信执行环境的软件可信基(SegTEE 中称为安全监控器)和操作系统内核进行管理,并在程序运行时同时进行校验。SegTEE 的分层设计,能够在保证极少的硬件改动和开销的前提下,实现接近富设备下的隔离灵活性,满足多种隔离需求。

其次,为了满足万物智联场景对多隔离域的需求,SegTEE 提出基于段保护的滑动窗口高可扩展隔离域机制。该设计能够在有限的段隔离寄存器(如 16 组)的基础上,支持上百个隔离域,相比前沿端侧可信执行环境系统(如蓬莱-PMP 和 Keystone),实

现最高 14 倍的提升。

再次，SegTEE 提出了基于段隔离机制的动态 I/O 保护方案，通过引入特权态驱动管理域，结合 RISC-V IOPMP 硬件能力，支持安全应用直接访问 I/O 设备和 I/O 设备的动态重分配，并且在驱动管理域内通过动态映射内存、最小特权等设计，进一步增强安全性。

最后，SegTEE 中针对资源税，提出了一系列创新的内存裁剪优化设计。首先，SegTEE 基于局部性(Locality)对资源税进行优化，仅在内存中保留当前使用的数据和代码；其次，SegTEE 设计了可定制化的安全监控器，能够根据部署设备对功能和安全特性的需求，定制化编译，实现定制化的资源裁剪和优化。这一系列优化能够实现相比前沿系统(蓬莱-PMP) 54%的优化。

本文在 FPGA 和 Qemu 上实现了 SegTEE 所需硬件扩展和机制，并结合真实应用场景对 SegTEE 的性能、安全性、可扩展性等多个维度进行了评测分析。结果显示，SegTEE 能够在有限的资源下，有效支撑小型端侧设备的安全需求，并带来显著的性能和可扩展性的提升。

本文的主要贡献如下：

(1) 观察到当前可信执行环境系统设计难以满足万物智联场景下，小型端侧设备的资源、性能和安全需求，并揭示其挑战和主要原因。

(2) 提出了面向万物智联小型端侧设备的信执行环境系统，SegTEE。SegTEE 能够有效解决当前可信执行环境系统在小型端侧设备上的挑战。

(3) 基于 FPGA 和 Qemu 的原型系统实现，并将 SegTEE 与前沿可信执行环境蓬莱-PMP 通过大量实验进行了对比。结果表明，SegTEE 能够有效减少内存资源占用(54%)，提供更多的隔离域 (14x 提升)，并且能够实现较低的隔离域管理操作和运行性能的开销。

本文第 2 节介绍背景并讨论现有方案的不足；第 3 节介绍 SegTEE 系统的设计；第 4 节介绍系统实现；第 5 节进行实验，并对结果进行分析；第 6 节总结全文。

2　背景与相关工作

2.1　背　景

可信执行环境(Trusted Execution Environment，TEE)是基于硬件安全特性保护应用的数据和代码的机密性及隐私性的技术[33-48]。可信执行环境的威胁模型十分强，通常认为特权软件如操作系统和虚拟机监控器是不可信的[49-56]，其可信基往往只包含硬件或者可形式化验证的安全固件。现有的大部分指令集都包含了可信执行环境的支持，如 Intel SGX[6]、ARM TrustZone[5,8-10]以及 RISC-V 开源可信执行环境系统蓬莱[15]和 Keystone[17]。由于其极小的可信基和较强的安全保障，目前可信执行环境已经被广泛应用于端、边、云各个场景[57-59]，是保障用户敏感数据和隐私的重要技术。

早期的可信执行环境，如 Intel SGX，通常完全由硬件(或硬件微码)实现复杂的安全能力和隔离域抽象。这导致它们的实现较为固化，无法适应动态变化的外部环境，面对新型的攻击时较难有效应对[60-63]。软硬件协同的可信执行环境[15,17,29]近来得到了广泛关注和研究，如 RISC-V 下的蓬莱[15]和 Keystone[17]等，其架构如图 1 所示。

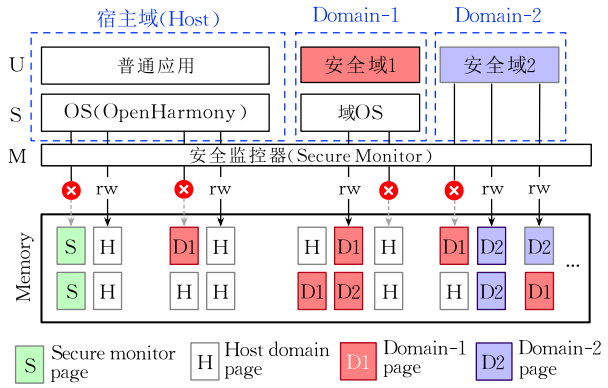

图 1　RISC-V 软硬件协同可信执行环境

图 1 中，左侧的 M、S、U 分别表示 RISC-V 指令集中的机器态(Machine Mode)、特权态(Supervisor Mode)和用户态(User Mode)。其中特权态和用户态分别运行操作系统和用户态程序。机器态有着比特权态更高的权限，通常用来运行安全监控器(Secure Monitor)。安全监控器是一层可形式化验证的软件层，负责基于硬件机制提供可信执行环境的抽象，如隔离域(图中的 Domain)的创建、销毁和隔离保障等。在一个 RISC-V 可信执行环境中，可信基只包括硬件和机器态的安全监控器。

万物互联的边端场景。随着传感器技术、5G 无线通信技术、云计算和大数据分析等技术的不断发展，万物互联的实现变得更加可行。在这种场景下，各种设备和系统能够实时收集和交换数据，从而实现更高效的资源管理、智能化的决策和更优质的用

户体验。典型应用包括智能家居、智慧城市、智能交通和工业自动化等。随着万物智联设备的发展，系统安全变得越发重要。为了保护用户隐私、机密模型等关键敏感数据，万物互联系统往往会通过软件或硬件手段来隔离各个应用。

万物互联场景对隔离数量的要求有着显著的增加。主要从两方面体现。首先，万物互联场景下端车设备上要运行的应用数量本身增加。例如，华为基于鸿蒙操作系统(HarmonyOS/OpenHarmony)研发的智能手表Huawei Watch系列。该智能手表基于LiteOS的嵌入式操作系统内核，并且同时可以部署的应用高达30多个，包括大量的安全敏感应用，如心率应用、血压应用等。另一个场景是无人飞行器(Unmanned Aerial Vehicle，UAV)，Minion[64]基于3DR IRIS+系统构建了UAV上的端侧原型系统，并且需要支持49个进程的隔离场景。随着可穿戴等端侧设备的智能化，在小型设备上运行多个应用并且对其进行隔离将会变得越来越重要。

此外，另一个导致隔离域数量需求显著增多的场景来自于设备互联。OpenHarmony操作系统目前正在推进分布式可信执行环境(Distributed TEE)能力的提升①。端侧设备之间的可信执行环境能力将被联通，不同的安全应用和数据将在可信执行环境之间流转。对于不同的连接，往往需要维护单独的隔离域来保护其安全性。目前，分布式可信执行环境的一个典型的应用场景就是智能手表和手机、智能音响、智慧屏等设备之间的协作互联。

此外，传统的小型端侧设备(如摄像头)对于基于TEE的隔离需求仍然较弱。然而，随着万物互联的需求增加，我们观察到越来越多的设备开始和人直接交互，例如智能手表、智能眼镜。这些智能可穿戴设备对于安全的需求极大增加。例如智能手表上的一个心脏健康监测应用，需要保护自己的数据不被第三方不可信的应用获取，否则可能会导致用户的重要隐私信息泄露。这样的趋势更加增强了在端侧小型设备上，实现较多隔离域(数十到上百)的TEE系统的必要性。

2.2 相关工作

可信执行环境已经在多个领域得到了广泛的应用，并且在各个主流架构中都得到了较好的支持。表1对当前代表性的可信执行环境进行了系统性的分析。

表1 相关可信执行环境系统

系统		隔离的资源税		性能	安全			年份	是否满足需求
名称	架构	硬件资源	软件资源	隔离域数量	防御页表旁路攻击	防御缓存旁路攻击	支持特权态隔离		
SGX[24]	x86(Intel)	页表、段隔离	低	无限制	×	×	√	2015	×
Scalable SGX	x86(Intel)	页表、段隔离	低	无限制	×	×	√	2021	×
TDX[25]	x86(Intel)	页表、二级页表、段隔离	高	64	√	×	√	2020	×
SEV[26]	x86(AMD)	页表、二级页表、段隔离	高	16/509	×	×	√	2016	×
SEV-ES[27]	x86(AMD)	页表、二级页表、段隔离	高	16/509	×	×	√	2017	×
SEV-SNP[28]	x86(AMD)	页表、二级页表、段隔离	高	16/509	×	×	√	2020	×
TrustZone[5]	ARM(TZ)	页表、段隔离	高	无限制	√	×	√	2002	×
Komodo[29]	ARM(TZ)	页表、段隔离	高	无限制	√	×	√	2017	×
Sanctuary[30]	ARM(TZ)	页表、段隔离	较低	无限制	√	√	√	2019	×
Shelter[36]	ARM(CCA)	GPC表	较高	无限制	√	√	√	2023	×
Sanctum[31]	RISC-V	双页表、PMP段隔离	高	DRAM区域数	√	√	√	2016	×
TIMBER-V[32]	RISC-V	PMP段隔离、Tagged Memory	高	无限制	√	×	√	2019	×
Keystone[17]	RISC-V	页表、PMP段隔离	高	PMP数量	√	√	√	2020	×
CURE[18]	RISC-V	页表、PMP段隔离	高	13	√	√	√	2021	×
Penglai-TVM[15]	RISC-V	页表、PMP段隔离	高	无限制	√	√	√	2021	×
Penglai-PMP	RISC-V	页表、PMP段隔离	高	PMP数量	√	√	√	2019	×
SegTEE	**RISC-V**	**段隔离**	**较低**	**>128**	√	√	√	**2023**	√

注：表中隔离域数量指标中的“无限制”表示该系统支持的隔离域数量不受设计限制，但是具体上限数量仍由硬件资源等限制。“是否满足需求”表示该系统能否在满足小型端侧设备的资源需求(包括内存资源和硬件能力)的前提下，支持较多的隔离域。架构中，ARM(TZ)表示基于TrustZone的系统，而CCA表示基于ARM CCA的系统。

Intel SGX主要面向桌面端[65]和云端场景[66-69]，基于页表、硬件段隔离(PRM等机制)、默克尔树等硬件机制，可以保护隔离域应用(SGX中称为Enclave)的内存机密性和完整性[70]。然而，SGX的安全内存必须存放在CPU预留的连续区域(PRM)中，导致安全内存大小有限制，一般仅能支持128 MB或256 MB。最近，Intel发布了可扩展的SGX版本，将安全内存大小扩展到TB级别，但降低了内存完整

① OpenHarmony distributed TEE service. https://gitee.com/openharmony-sig/tee_distributed_tee_service，2024，08，30

性保护，并仍然需要静态保留安全内存区域。SGX和可扩展的SGX都需要较为复杂的硬件安全特性支持，难以应用在边端小型设备中。

当前可信执行环境系统的一个重要演进趋势是支持机密虚拟机抽象，然而该抽象存在较大的软硬件开销，更进一步地限制了其在端侧的应用。例如，AMD SEV[26]及其扩展系统（如SEV-ES和SEV-SNP[28]）可以保护虚拟机（VM）不被特权软件攻击，且支持较为灵活的虚机内存分配。Intel TDX旨在将安全VM与其他软件隔离开来，依赖可信特权软件TDX模块（TDX Module）来管理和保护机密虚拟机。目前TDX主要提供基本的完整性保护，并不能防御基于硬件的内存重放攻击。SEV和TDX一方面会引入较大的硬件开销，需要硬件层面支持虚拟化等关键特性，另一方面还依赖于虚拟机（如虚机内核）等复杂的软件栈，这导致该类抽象在资源受限的边端场景是较难得到应用的。此外，机密虚拟机的数量同样受到硬件限制，例如SEV由于标识长度，通常支持16个或509个实例，而TDX则由于MKMTE中秘钥数量限制只能支持64个实例。ARM同样提出了CCA可信执行环境扩展[71]，能够支持机密虚拟机形态，但是同样会带来不可忽视的资源税。Shelter[36]基于ARM CCA，提出了用户态隔离方案，能够有限避免额外的软件开销（来自于虚拟机管理等），并且支持轻量级应用。然而，Shelter[36]所需要的软硬件资源相比轻量端侧应用的场景仍然过高。

目前终端场景使用最多的可信执行环境系统是ARM TrustZone[5]。基于TrustZone的可信执行环境系统，例如Komodo[29]和Sanctuary[30]，对安全世界中的安全应用（TA）数量和内存大小通常没有限制，但是安全应用能使用的内存等资源通常是在启动阶段静态划分的，存在性能和资源利用率的权衡，且没有加密或完整性保证。

近年来，研究者在开源指令集RISC-V架构下，探索了多种面向从端侧到云侧的可信执行环境系统。RISC-V可信执行环境，如蓬莱[15]和Keystone[17]，通常为软硬件协同的设计。它们引入了一层可形式化验证的系统软件层，安全监控器（Secure Monitor），由安全监控器负责基于硬件机制提供可信执行环境的抽象，如隔离域的创建、运行、销毁等。而硬件只需要提供最基本的安全机制，如物理内存隔离机制RISC-V PMP[72]。Keystone利用RISC-V的物理内存保护（PMP）机制实现Enclave内存的隔离，通过一组成对的寄存器指示物理内存区域和访问权限。因此，Keystone中的内存区域数量受到PMP寄存器数量的限制，最多16个。为了防御物理攻击，Keystone利用片上计算，但受限的片上RAM导致开销较高。Sanctum[31]是一个早期经典的RISC-V可信执行环境设计，影响了后续的大量系统。Sanctum中提出了双页表的隔离方式以及基于DRAM的缓存旁路攻击防御，前者在小型设备中通常无法支持，而后者对硬件的修改较大。此外，Sanctum的Enclave数量也受到DRAM隔离区域数量的限制。CURE[18]通过在硬件里面引入了隔离域ID（Enclave ID）的设计，能够支持定制化的隔离域，甚至可以在机器态划分出一个隔离环境。然而CURE对硬件的改动较大，需要修改CPU核以及内存控制器，且其目前只能支持13个隔离域，无法满足边端场景万物智联要求。TIMBER-V[32]针对边缘终端设备进行了设计和优化。TIMBER-V引入了标签内存的技术，能够支持几乎任意数量的隔离域，并且针对跨域通信进行了优化。然而，标签内存本身会带来较大的性能开销（平均25.2%），这会对应用造成较大的影响。此处，我们将Sanctum、CURE、TIMBER-V等工作仍分类为RISC-V是因为它们的基础指令集是RISC-V，且所提硬件扩展能够兼容原有指令集扩展，这同样是RISC-V开源开放的特定所支持的。

HexFive[73]是一个闭源的面向IoT场景的可信执行环境系统，其使用类似微内核的思想来解决操作系统和安全监控器争抢的问题。然而，HexFive仍然使用传统的RISC-V PMP黑白名单的方式来管理隔离域，导致其隔离域数量较为受限。

2.3　动　机

虽然当前RISC-V可信执行环境系统在面对桌面和服务器等场景中，能够实现较好的性能和可扩展性，但是仍然缺乏一个面向万物互联场景下小型端侧设备的可信执行环境系统。本文观察到，万物互联对可信执行环境系统带来了以下需求及挑战。

（1）需求1。多层多域隔离。万物互联场景下，可信执行环境所面临的一个重大的需求变化是：需要支持动态的安全应用的部署、迁移、协同的能力。然而，传统小型设备通常仅支持简单的物理内存划分保护机制，一方面，难以支持多层隔离，另一方面，也难以支持较多的隔离域。

首先，当前RISC-V设备厂商通常会将操作系统直接部署在机器态而非特权态。一个主要的原因是因为物联设备通常不具备页表等地址翻译的内存管理单元。此时，如果将操作系统部署在特权态中，那么用户态和特权态之间是不存在物理内存上的隔离保障的，这使得一个恶意应用能够直接读取到操作系统和其他应用的数据。而在机器态中，RISC-V设计了物理内存保护（Physical Memory Protection，

PMP)机制,其允许硬件来配置物理内存的权限。然而,当在小型设备上同时支持操作系统和可信执行环境时,当前的物理内存隔离机制难以支持,必须要引入页表等,从而导致较高的硬件资源开销。

其次,当前 RISC-V 可信执行环境系统使用 RISC-V PMP 机制来提供不同域之间的物理内存隔离。然而,由于 PMP 所能划分出来的隔离域和其寄存器个数直接相关(最多 16 组寄存器),导致安全监控器能够提供的隔离域十分有限,从而限制了计算场景的应用部署。现有方案中,蓬莱[15]通过利用 RISC-V TVM 特性结合虚拟内存,能够实现几乎任意个数的隔离域。然而物联设备中通常无法支持虚拟内存。Keystone[17]通过引入安全特权态运行时(即 TEEOS),能够在一个隔离域内通过软件方案再次创建多个安全应用。然而,在物联设备中,引入 TEEOS 一方面会导致较大的资源开销,另一方面也需要特权态和用户态之间有页表之外的隔离机制,而目前 RISC-V 还没有类似的方案。

(2)需求 2。动态 I/O 保护。传统可信支持环境采用静态划分的机制,例如 TrustZone,来隔离可信和不可信 I/O 设备。在万物互联场景下,由于安全应用存在动态迁移和部署的需求,例如,一个摄像头相关的安全应用可能会被临时部署在 TEE 中,此时需要有能力动态地将该摄像头设备从不可信环境中切换到可信环境。然而,支持动态的 I/O 设备对于小型设备而言十分复杂:动态 I/O 设备对于系统的安全性提出了更高的挑战,如何在 I/O 设备切换前后,保证安全应用正确使用,是较为复杂的;另一方面,动态 I/O 设备通常需要在硬件层面引入专用的单元来配置设备的能力及权限(例如,能够通过 DMA 访问的内存区域),传统的 IOMMU 等方式会在这个过程中引入较大的开销。整体来看,当前仍然缺少一种,既能够支持动态灵活的 I/O 保护,同时适应端侧小设备的方案。

(3)需求 3。轻量资源占用。万物互联对可信执行环境在动态、灵活、安全、性能等方面的要求,使得传统的硬件固化方案较难满足要求(无法支持特性升级)。软硬件协同的方案虽然带来更高的灵活性,但是会引入软件可信基和安全监控器,从而带来额外的内存资源开销。以蓬莱为例,目前的 MMU 和 MCU 版本需要的资源如表 2 所示。可见,蓬莱的多个版本目前至少需要 512 KB 以上的内存资源,这些内存资源的相当一部分是安全监控器引入的。Keystone 需要的内存和存储资源和蓬莱 PMP 版本相当。与之对比,目前的物联设备(如 MCU)的资源可能会非常小,通常在数百 KB 的量级,1 MB 以上的内存资源的设备仍然不多。可见,当前的 RISC-V 可信执行环境中安全监控器所需要的资源会给设备带来极大的资源压力,甚至在某些设备上无法部署。

表 2 蓬莱资源需求

系统	内存资源	存储资源
蓬莱-PMP	16 MB	N/A
蓬莱-MCU	512 KB(Instruction),256 KB(Data)	512 KB

整体来看,当前系统在云端和桌面端能够支持较高的灵活性和动态,但是无法满足端侧小设备的性能和资源的限制。而现有端侧可信执行环境方案,如 TrustZone,仅能提供优先的动态配置能力,难以适应万物互联所带来的新的需求。本文提出了一个新的 RISC-V 架构下的可信执行环境系统——SegTEE。其能够在不降低隔离保障的情况下兼容现有的操作系统,实现>128 的隔离域,并且将软件可信基的内存资源占用降低到<69 KB。

3 SegTEE:基于段隔离的轻量可信执行环境系统设计

本文提出 SegTEE,这是一个面向万物互联等复杂场景的小型端侧设备可信执行环境。相比现有可信执行环境系统,SegTEE 完全基于段隔离机制,实现全系统的隔离与保护,并且能够在资源、性能和安全性上实现显著的突破。

3.1 系统架构

SegTEE 的系统架构如图 2 所示,主要包括 4 个关键部分:安全监控器、安全处理器硬件扩展、可信执行环境内核驱动和用户态库。图 2 中将 SegTEE 中主要的可信基部分标为黄色。

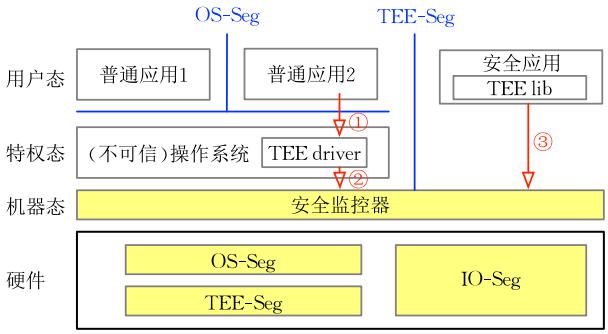

图 2 SegTEE 架构

安全监控器(或 Secure Monitor)是 SegTEE 系统的软件可信基,通常运行在最高特权态,例如 RISC-V 指令集下的机器态,拥有软件层面的最高权限。安全监控器负责管理硬件安全特性,并提供可信执行环

境抽象(图 2 中箭头②和③)。安全监控器会将上层软件隔离成不同的隔离域，如图 2 中普通应用所对应的宿主域和安全应用所在的一个隔离域。一些可信环境系统中也使用飞地(Enclave)来表示隔离环境，本文中将统一使用隔离域(Domain)的表述。

安全处理器硬件扩展提供一系列基础安全能力。和传统基于页表、嵌套页表等机制进行隔离的可信执行环境不同，SegTEE 主要使用段隔离机制。其中，TEE-Seg 和 OS-Seg 两层段隔离机制构成了 SegTEE 中关键的嵌套段隔离。TEE-Seg 能够提供隔离域之间的物理内存隔离，而 OS-Seg 则用于在特权态操作系统和用户态之间实现段隔离保护机制，从而避免对页表的依赖导致的高昂的硬件成本和开销。在内存隔离之外，IO-Seg 是基于段隔离的外设保护机制，能够防御恶意外部设备所发起的攻击。在原型系统实现中，本文基于 RISC-V PMP 来实现 TEE-Seg，并且提出了 S-mode PMP 硬件隔离机制，来实现 OS-Seg，最后基于 RISC-V IOPMP 实现了 IO-Seg 机制。除了内存和 I/O 隔离，SegTEE 同样可以支持可选的加解密和完整性保护。

可信执行环境内核驱动(图 2 中的 TEE driver)是操作系统的一个扩展，其将安全监控器提供的接口进行封装，并以系统调用等方式暴露给普通用户态程序(图 2 中箭头①)来使用可信执行环境。如普通用户态程序可以调用内核驱动提供接口来创建一个新的安全应用，该调用会被驱动转化为对安全监控器的调用，从而最终在安全监控器中完成安全应用所属隔离域的创建操作。

可信执行环境用户态库(图2中 TEE lib)是 SegTEE 封装的安全应用编程抽象，能够在不破坏安全应用隔离性的情况下，尽可能简化其编程复杂性，兼容现有应用。开发者可以基于用户态库开发安全应用。

SegTEE 安全监控器提供的编程接口如表 3 所示，主要分为宿主侧接口和隔离域侧接口。其中，宿主侧接口提供了对于一个隔离域的创建、运行、销毁、验证等操作；而隔离域接口则提供了退出(完成执行)和外部调用接口。

表 3　SegTEE 提供的主要接口

接口分类	具体方法	描述
宿主侧接口	CREATE_ENCLAVE	通过给定的参数创建一个隔离域。
	ATTEST_ENCLAVE	获取一个隔离域的认证(Attestation)报告。
	RUN_ENCLAVE	运行一个已经创建好的隔离域。
	DESTROY_ENCLAVE	销毁一个隔离域。
隔离域侧接口	EXIT_ENCLAVE	隔离域退出并返回到 Host 侧。
	ENCLAVE_OCALL	隔离域通过 OCALL 调用 Host 内核中的相关函数和服务。

下面将介绍 SegTEE 是如何解决端侧小型设备对可信执行环境带来的技术挑战。

3.2　威胁模型

SegTEE 系统的可信基仅包含硬件(CPU、内存等)和安全监控器。外部 I/O 设备和其他软件(包括操作系统)属于不可信部分，可能被攻击者攻破。具体来说，本系统主要考虑以下两类攻击：

(1)特权软件攻击。攻击者可以完全掌控不可信的操作系统，并且部署恶意的隔离域应用。

(2)物理攻击。攻击者可以拦截通过操作系统等方式，控制外部 I/O 设备，并且发起对可信执行环境的攻击。

旁路攻击中，SegTEE 可以抵御受控信道攻击(Controlled channel attack)。其他基于缓存或 TLB 等硬件的旁路攻击不在本论文的讨论范围内，可以通过结合如缓存划分、染色等机制进行防御。

SegTEE 不考虑来自于不可信软件或硬件发起的拒绝服务攻击(Denial-of-Service attack)。

该威胁模型继承了当前主流的 RISC-V 可信执行环境(蓬莱和 Keystone 等)的威胁模型。主要的一个变化在于对于内存的假设是可信的，这是由于端侧设备上通常通过较为紧密地封装方式来连接 CPU 和内存，内存在物理上被攻击的概率较小。此外，SegTEE 也支持通过整合加密内存和内存完整性(如默克尔树)的方法来进一步增强对于内存的安全保护。不过这两个技术由于其复杂性，在当前的小型端侧设备中使用仍然有限。

3.3　嵌套段隔离保护机制

3.3.1　核心设计

为了突破传统扁平化段隔离所带来的一系列问题和挑战，SegTEE 中创新性地提出了嵌套的段隔离保护机制。具体来说，SegTEE 中引入了两层段的内存保护，分别为 TEE-Seg 和 OS-Seg。两层段隔离分别由机器态的安全监控器和操作系统内核进行管理，并在代码执行时同时进行校验。

SegTEE 的嵌套隔离保护机制至少带来了以下几点收益：

首先，能实现多特权态(不少于 2)的隔离与保护。传统扁平式的段隔离机制，通常只能支持两个特权态的保护，一个特权态拥有所有权限，另一个特权态的访存行为被段隔离机制所配置的权限所约束。而可信执行环境中，为了支持灵活性，通常需要同时存在安全监控器和操作系统内核等多个特权态，导致传统设计难以在支持可信执行环境的系统中应用。

嵌套化的段隔离机制能够将隔离段分布在不同的特权态，将扁平化的检查变成分层检查，从而在实现内存空间隔离的同时，实现多个权限态之间的保护。

其次，支持特权隔离域。由于SegTEE中将隔离域的段（TEE-Seg）和操作系统的段（OS-Seg）进行了解耦，因此能够在安全监控器层，通过 TEE-Seg 划分隔离域，从而支持带特权态的隔离域。这能够支持使用驱动访问设备的安全应用场景，如数字版权保护等。

最后，有助于平衡安全性和可扩展性。由于段隔离机制的资源限制，通常其所能支持的隔离域数量较为有限，并且难以通过软件方法进行扩展（可能存在安全风险）。嵌套隔离机制在解耦两层段隔离后，能够在其中一层进行可扩展隔离域优化，并且不会影响到另一层保护，从而灵活地平衡安全性和可扩展性。该优势会在后续介绍基于段机制的高可扩展技术时展开描述。

3.3.2 OS-Seg 与 TEE-Seg 具体设计

在系统原型中，SegTEE 使用 RISC-V 架构下的 PMP 段隔离机制作为 TEE-Seg，并提出了特权态段轻量段隔离机制 sPMP（s-mode Physical Memory Protection），作为 OS-Seg。本节将重点介绍 OS-Seg 的 sPMP 具体设计。本文后续会讨论如何将 TEE-Seg 与 OS-Seg 的设计扩展到其他指令集架构中。值得注意的是，sPMP 的硬件已经被 RISC-V 国际基金会社区采纳并正在推动中。

sPMP 采用类似现有 RISC-V PMP 的段保护机制，如图 3(a)所示。sPMP 包含多组段寄存器，每组寄存器代表一段连续的物理内存及其权限。具体而言，一组 sPMP 包含两个元素：地址范围（图中的 Addr）和配置信息（图中的 Config）。以图中 sPMP-0 为例，物理基地址（Base PA）和编码的长度信息（size）可以表示一段连续物理内存的地址范围。配置信息中则记录着当前用户态程序对该内存区域的访问权限（perm）。

（1）sPMP 寄存器。sPMP 机制引入了两类新的寄存器，spmpaddr 寄存器和 spmpcfg 寄存器。在 RISC-V 32 位下，一个 spmpaddr 寄存器能够记录一个 sPMP 段的地址信息，而一个 spmpcfg 寄存器能够记录 4 个段的配置信息。spmpaddr 如图 3（b）所示，在 RISC-V 32 位下，能够保存最多 34 位的物理地址信息（其中最后两位要求为 0）。spmpcfg 如图 3（c）所示，一个 spmpcfg 能够保存 4 个段的配置信息，如图中的 spmpcfg0 会记录前 4 个 sPMP 的配置信息（即 spmp0cfg 到 spmp3cfg）。每个段的配置信息为 8 位，如图 3（d）所示，包含 6 项内容：

① L 域。表示当前的 sPMP 段是否处于锁定（Locking）状态，如果该域为 1，则对应的地址和配置信息无法被特权态修改（只能由机器态软件对该段进行重置）。

② U 域。表示当前的 sPMP 段是否属于用户态内存段，如果该域为 1，则操作系统会对 sPMP 的访问进行限制（具体限制将在下文 sPMP 安全增强特性展开介绍）。

③ A 域。表示当前的 sPMP 段的长度信息的编码方法。和 RISC-V PMP 相同，sPMP 可以使用两类编码方式：其一是使用前一个 sPMP 段寄存器的地址信息作为当前段的起始地址，当前段的地址信息作为终止地址，来表示一段物理内存区域；其二是使用地址信息末尾 0 的个数来编码长度信息（要求长度为 2 的幂次方，且地址和长度对齐）。

④ R、W、X 域。表示当前的 sPMP 段的访问权限信息，三个域分别对应可读可写可执行权限，当对应的域为 1 时，用户态应用可以对该内存区域执行对应的操作，比如若 R 为 1，则用户态可以读取该内存段数据。

目前，sPMP 和 RISC-V PMP 相同，支持最多 16 个段，因此硬件最多需要实现 20 个寄存器，包括 spmpaddr0 到 spmpaddr15 和 spmpcfg0 到 spmpcfg3。需要强调的是，在不超过 16 个的限制下，具体实现

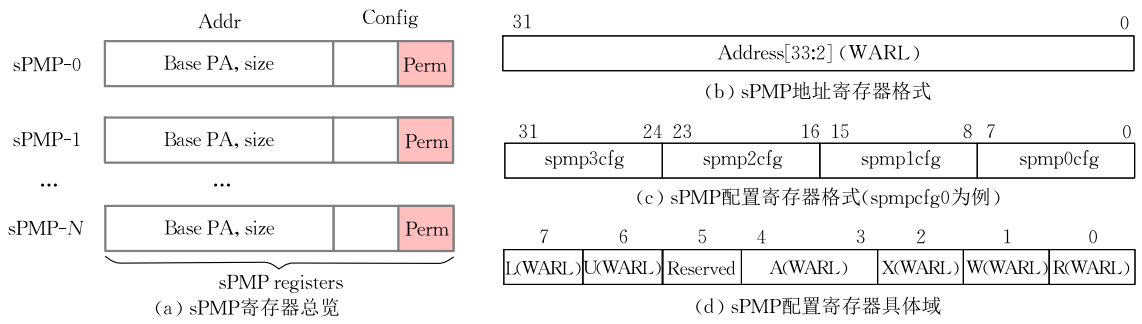

图 3 sPMP 设计及寄存器

多少个 sPMP 段取决于硬件实现时的资源要求。

(2) sPMP 安全增强特性。在 SegTEE 设计中，除了传统的读写执行权限外，还为 OS-Seg 引入了额外的安全特性。例如，sPMP 通过 L 和 U 域对操作系统权限进行了额外限制。

默认情况下，sPMP 段的权限信息仅作用于用户态程序，而当 L 域被设置上时，该权限信息将同时被作用于特权态软件。如操作系统可以将一段只读的敏感数据配置在一个独立的 sPMP 段中，将其权限设置为只读(权限信息中只有 R 域为 1)，且锁定该 sPMP 段(将 L 域设定为 1)。通过这种方式，操作系统自身同样将受到该权限的限制，只能读取而无法修改该敏感数据，且锁定的 sPMP 段无法再被操作系统自己修改。这种方式能够在操作系统遭遇攻击时，尽可能避免攻击者对数据的破坏和窃取。

此外，为了避免操作系统错误地读取、修改用户态的数据，sPMP 提供了 U 域。操作系统可以通过设置 U 域，来显式地告诉硬件该段为用户态数据。当 RISC-V 状态寄存器中的 SUM(Supervisor User Memory access)域为 0 时，操作系统访问所有标记为用户态段的内存将导致异常。而当 SUM 域为 1 时，操作系统对用户态段的访问是允许的。U 域为万物互联场景下的小型设备操作系统使能了 SMAP (Supervisor Mode Access Prevention)的安全特性，能够减少操作系统被攻击的可能性。此外，不论 SUM 是什么状态，sPMP 始终不允许操作系统执行任意用户态代码，即 SMEP(Supervisor Memory Execute Protection)安全特性同样能够通过 U 域得到支持。值得注意的是，当前 sPMP 要求 L 和 U 不能同时置为 1，即操作系统不应该锁定一个用户态 sPMP 段。

(3) 权限检查。当用户态程序访问内存时，其会首先经过 OS-Seg(即 sPMP)进行检查，然后经过 TEE-Seg(即 PMP)检查，只有两者同时通过，一次访存操作才能成功。在 sPMP 检查时，硬件会从 0 号 sPMP 段开始进行地址匹配，直到遇到第一个能够完整覆盖访存操作区域的 sPMP 段，读取该段配置中的权限信息对该访问操作进行验证。即，对于多个不同的 sPMP 段，本文采用了静态优先级的方式，即越低的段编号其优先级最高(sPMP-0 有着最高的优先级)。当所有的 sPMP 段都无法匹配一个用户态访存时，该访存操作失败，即默认情况下用户态没有访存权限。

操作系统(特权态软件)的访问同样会经过上述流程。当匹配到的第一个 sPMP 段中，既没有配置 L 域也没有配置 U 域时，该次访问操作成功。如果没有任何一个段匹配访问的内存地址，该访问操作同样成功，即默认情况下特权态拥有所有访问权限。而如果匹配到的段中 L 为 1，则按照权限进行检查；如果 U 为 1 且 SUM 为 0，则触发异常。安全监控器(机器态软件)的访问不受 sPMP 的检查。

(4) OS-Seg 在操作系统中的应用。以 sPMP 为例，OS-Seg 为操作系统提供了灵活的访问权限的管理。操作系统首先需要预留几个 sPMP 段来对自身安全敏感的区域进行保护，如将系统代码部分设置为可读可执行但不可写，并且锁定该 sPMP 段，从而避免攻击者恶意修改操作系统代码。剩下的 sPMP 段，操作系统需要通过白名单的方式，分配给应用使用。如每个应用可以访问不同的物理内存段，操作系统需要在上下文切换时，根据目标应用能够访问到的区域，切换对应的 sPMP 寄存器。

(5) 基于 sPMP 的 OS-Seg 机制安全性分析。OS-Seg 属于特权态(RISC-V Supervisor Mode)寄存器和特性，可以由操作系统修改和配置。值得说明的是，这并不会破坏 SegTEE 的威胁模型(操作系统不可信)。在操作系统和安全应用之间切换时，一定会经过安全监控器，安全监控器会根据安全应用的上下文来恢复 OS-Seg(即 sPMP 寄存器)。本质上，SegTEE 将安全保护和资源管理进行了分离，操作系统只负责具体的分配 sPMP 寄存器等操作，而运行中操作系统无法篡改安全应用的寄存器。该设计和现有的如 Intel SGX、RISC-V 蓬莱等可信执行环境的设计思路是类似的。

(6) 贡献与创新点。现有系统通常选择使用：①段隔离物理内存，结合页表虚拟内存的映射与隔离；或者②仅使用段隔离物理内存。方案①对资源有较大需求，难以适应万物互联场景下的小型设备，方案②仅能提供扁平化的隔离能力，无法满足多层特权态的应用需求。SegTEE 中创新性地提出嵌套的段隔离硬件机制，在使用较少的资源占用下，实现多特权态分层隔离的能力，有效支撑小型设备安全需求。

3.4　基于滑动窗口的高可扩展隔离域

SegTEE 基于段隔离进行隔离域和特权态间的隔离。然而，段隔离机制带来的一个重大的挑战是：十分有限的隔离域数量。如 3.2 节介绍，段隔离机

制(如 OS-Seg 的 sPMP 设计)依赖于 CPU 内部的寄存器来记录内存区域及对应的权限,而 CPU 片上资源有限,难以在硬件层面支持较多的隔离域。目前 RISC-V PMP 机制和本文提出的 sPMP 机制,都仅能支持最多 16 个不同的内存段。在万物互联场景下,需要支持较多的隔离域数量来实现和多设备间的安全协同,现有硬件隔离域数量无法满足要求。

下面,将具体介绍现有的可信执行环境如何基于段隔离机制进行隔离域保护,并且分析其固有的可扩展性(即隔离域数量)的限制,并随后给出 SegTEE 中基于滑动窗口的高可扩展隔离域方案。

3.4.1 传统基于段隔离机制的隔离域保护

本节以 RISC-V PMP 作为 TEE-Seg 案例,介绍传统基于段隔离机制的隔离域保护设计。

RISC-V PMP 和本文提出的 sPMP 类似,通过地址信息指定物理内存区域,通过配置信息记录其权限。当前,安全监控器通常采用一套黑白名单的机制实现基于 PMP 的内存隔离,如图 4 所示。图中显示当前的系统中包含 3 个隔离域:Host、Domain-1 和 Domain-2。物理内存会被划分给这三个不同的隔离域以及安全监控器。

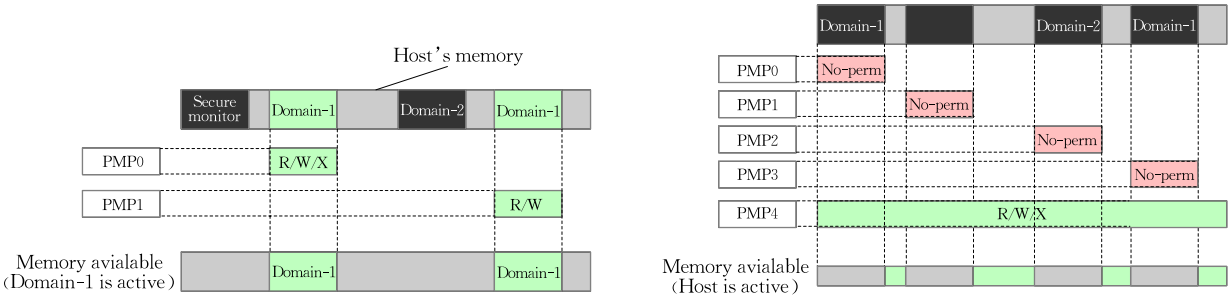

图 4 物理内存隔离的黑白名单机制

由于非 Host 的隔离域通常是单个应用,其使用的物理内存段较为有限,当前可信执行环境系统(如 RISC-V 蓬莱和 Keystone)使用白名单的管理方式,即通过 PMP 段寄存器,显式地将隔离域允许访问的内存进行授权。如图 4(a)所示,Domain-1 包含两个物理内存段。当其开始运行时,安全监控器需要配置两个 PMP 段寄存器,分别授权 Domain-1 对内存段的访问权限。RISC-V PMP 硬件会保证,对于未匹配的区域,当前的隔离域默认没有访问权限。

与之对应的,Host 域需要动态划分资源给其他隔离域,因此其物理内存段通常较为离散。当前可信执行环境系统采用黑名单的方式实现 Host 域的物理内存隔离,即通过 PMP 段寄存器显式地将不可访问的内存段设置为没有权限,如图 4(b)所示。此时,安全监控器会配置前 4 个 PMP 段寄存器,将安全监控器和其他隔离域对应的物理内存段设置为无权限(No-perm),并且在最后一个 PMP 段寄存器将整个地址空间的权限设置为可读可写可执行。由于 PMP 的段存在优先级(编号越低优先级越高),当 Host 域访问的地址匹配到前面 4 个物理内存段时,将由于无权限而触发异常;而当 Host 访问内存没有匹配到其他隔离域和安全监控器内存时,则访问成功(由 PMP-4 保证)。

上述机制虽然能够准确地实现不同隔离域之间的内存隔离,但是存在可扩展性的问题。基于黑名单的 Host 域内存隔离中,除了安全监控器和 Host 域的 PMP 段寄存器(最后一个),剩余的段寄存器仅(最多)14 个。

这意味着即使每一个非 Host 隔离域只使用一个物理内存段,当前系统也仅能支持 14 个隔离域,这显然是无法适用于万物互联端侧场景的。而一旦硬件本身实现更少的段寄存器(如 8 个或 4 个),隔离域的可扩展问题将会变得更为严重。

3.4.2 基于滑动窗口的物理内存隔离

本文观察到,导致当前隔离域可扩展性问题的根本原因在于段寄存器(如 PMP)和隔离域内存段的一一映射关系。如果一个段寄存器能够对应多个隔离域的内存段(如 4 个),那么就可以通过有限的段寄存器实现更多的隔离域。

更进一步,本文观察到同一个物理核在同一时间只会运行一个隔离域,这意味着安全监控器是有可能采用分时复用的方式,来复用同一个 TEE-Seg 段寄存器的。为了实现这点,安全监控器还需要能够将这些分时复用的物理内存段进行拼接,这是由于

在切换到 Host 域时，安全监控器仍然只能使用有限的 TEE-Seg 段寄存器来限制 Host 域的访存。比如，当前安全监控器通过分时复用的方式给非 Host 域分配了 100 个物理内存段。当安全监控器需要进入 Host 域执行时，需要通过（最多）14 个 TEE-Seg 段寄存器，去限制 Host 域对这 100 个内存段的访存，这是一件较为有挑战的任务。

为了解决上述问题，Seg-TEE 中提出了基于滑动窗口的物理内存隔离设计，如图 5 所示。其核心思想是使用一个 TEE-Seg 段寄存器去保护多组（不同的）隔离域的物理内存段。为了保证安全监控器仍然能够高效地限制 Host 域对这些内存段的访问，TEE-Seg 要求这些内存段在物理内存层面连续。如图 5(a)所示，PMP-0 对应了三个物理内存段，分别为 Domain-1、Domain-2 和 Domain-3。当 Domain-1 正在运行时，安全监控器会滑动 PMP-0，将其范围缩小到 Domain-1 对应的区域，从而只允许 Domain-1 访问自己的内存。当安全监控器切换到 Domain-2 时（图 5(b)），安全监控器会滑动 PMP-0 将其地址范围移动到 Domain-2 对应的物理内存段的范围，从而允许 Domain-2 访问自己的内存。而当切换到 Host 域时，安全监控器会将所有的 PMP 都展开为其保护的所有内存段的范围总和。如图 5(c)所示，安全监控器会配置 PMP-1，使其同时将三个物理内存段的权限设置为 No-perm，从而限制 Host 域对其的访问。

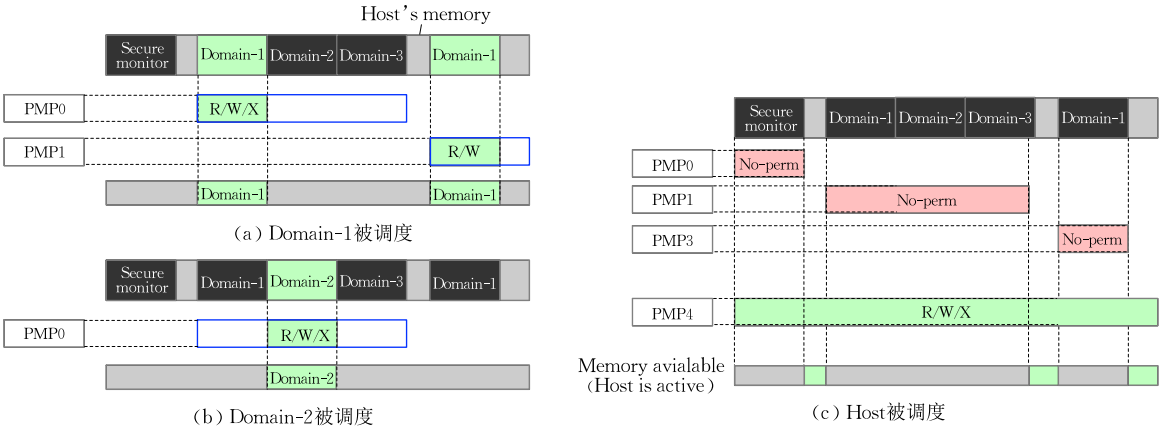

图 5　基于滑动窗口的高可扩展物理内存隔离

通过滑动窗口的设计，上述案例中一个 TEE-Seg 寄存器能够用来管理三个隔离域的三个不同的物理内存段，并且没有安全性的损失。滑动窗口设计能够有效提升 TEE-Seg 在万物互联端侧场景下的隔离域的可扩展性。

贡献与创新点：基于段的物理内存隔离是当前 TEE 系统一个主流实现，然而，现有系统均无法突破段隔离所带来的局限的隔离域数量的挑战，一个关键原因在于，对于段隔离机制本身分时复用，虽然存在可行性，但是极易引入安全风险，现有主流 TEE 系统，如 TrustZone、蓬莱、Keystone 等，均保持底层物理内存隔离机制的相对静态性。本文结合 OS-Seg 的嵌套隔离，提出了滑动窗口的设计，该设计通过在大的隔离域范围内动态滑动（只缩小范围，而不增大范围），从而能够在不影响安全性的前提下，突破段隔离的隔离域数量限制，实验显示最多能达到 100 个以上的隔离域数量。

3.5　资源税与内存裁剪优化

可信执行环境在小型端侧设备上面临着灵活性与资源税的权衡。一种可信执行环境的实现方式是将安全特性和能力全部固化到硬件上，这能够最大限度地提升可信执行环境性能，并且避免额外的资源占用（如内存和 CPU 等）。然而，硬件固化实现的方式会导致灵活性的问题，难以实时更新可信执行环境能力，无法抵御可能涌现的攻击和支持新的特性。由于万物互联场景下设备间的协同和交互变多，灵活性的缺点尤为严重。与之对应的是软硬协同的方式（也是本文 SegTEE 所采用的方式），在硬件层面仅实现基本的安全原语，由软件可信基（安全监控器）实现完整的可信执行环境抽象和保护。而该方式会导致较为显著的资源开销，也即资源税。

为了在达到灵活性的前提下突破资源税挑战，SegTEE 提出了以下两点创新技术。

3.5.1　基于局部性（Locality）的资源税优化

可信执行环境所产生的资源税，主要来自于其软件可信基，即 SegTEE 中的机器态软件和安全监控器。从功能角度来看，安全监控器的资源税主要

用于两方面:首先,负责设备启动和环境初始化的代码和数据(如图 6(a)所示)。这部分资源主要用于完成硬件状态的初始化、安全策略的初始化等。启动加载阶段的最后一步通常是加载 Host 域操作系统镜像,从而进入 Host 域中完成后续的操作和流程。其次,负责运行时可信执行环境调用的代码和数据。以 RISC-V 指令集架构为例,运行时的资源用于支持 RISC-V 机器态固件按照特定协议(RISC-V SBI 协议),并为上层软件(操作系统和应用程序)提供服务,允许上层软件通过 SBI 调用,来陷入机器态执行特定操作。

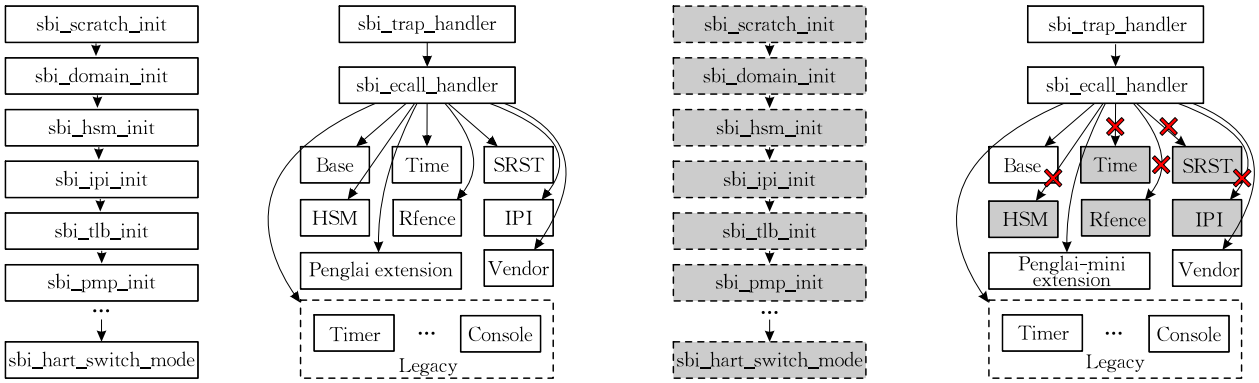

图 6 内存裁剪

本文观察到,安全监控器的资源具有明显的局部性(Locality)特征,在不同的运行阶段,通常只会依赖特定的资源。例如,图 7 显示了安全监控器在系统运行声明周期的内存资源占用的表现。在初始化阶段执行所需要的大量的启动加载代码和对应的数据,占据相当比重的内存资源,但是在运行时阶段几乎不会被使用。

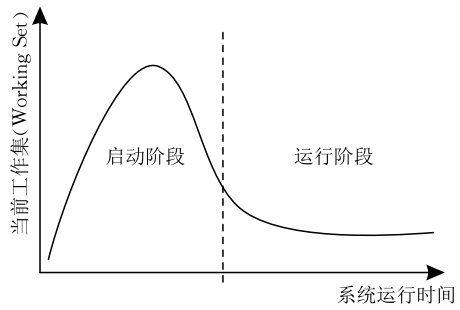

图 7 TEE 系统安全监控器对资源的“局部性”表现

基于上述观察,本文在 SegTEE 中提出了第一个资源税优化方法:基于局部性(Locality)的内存资源税优化。

具体来说,SegTEE 借鉴 Linux 内核中的思路,引入了初始化宏(__init)的设计。SegTEE 在安全监控器中提供了三个特殊的宏,用来标记安全监控器中的代码或数据为初始化代码和数据。其中,__init 用来标记一个初始化函数,而__initdata 和__initconst 用来标记初始化相关数据。SegTEE 安全监控器中,会将这些代码和数据放置在专门的区域。当安全监控器完成启动流程时,会释放这些初始化相关的内存资源,如图 6(c)所示,灰色部分表示对应的代码内存被释放。

3.5.2 可定制安全监控器

运行时代码和数据对内存资源的占用同样较大。本文观察到,即使应用了上节中的优化,安全监控器在运行时阶段仍然存在部分资源(代码和数据)从不被使用的情况。经过分析,发现是由于万物互联端侧场景的设备情况较为复杂,存在差异化的硬件能力和特性。例如,硬件可能是单核或者多核的情况。而安全监控器作为一个支撑通用的端侧设备的软件层,需要尽可能考虑多样复杂的情况,这就导致部分代码和数据在特定设备上不需要的情况。例如,传统的机器态能力,状态刷新(rfence),在端侧设备上通常是不需要的。

现有的其他可信执行环境系统,如蓬莱和 Keystone,通常面临的是资源较为充足的设备,因此只需要考虑尽可能通用的情况即可。而这样的设计在万物互联端侧场景会导致显著的资源税开销。

为此,SegTEE 提出第二个资源税优化方法:可定制安全监控器。

具体来说,SegTEE 中,安全监控器的运行时特性被分为了两类:核心特性(Core)和扩展特性(Extensions)。设备开发和部署人员可以根据目标

设备的资源和功能/安全需求，对 SegTEE 的安全监控器进行定制化编译，仅加载必要能力，如图 6(d)所示。目前，SegTEE 的原型实现中，基于 RISC-V OpenSBI，将多个扩展进行了解耦和拆分，并将核心的安全特性(TEE Extension)作为必选之一。

贡献与创新点：软件 TCB 所带来的灵活性与其资源占用的问题是小型设备可信执行系统上长期存在的、较难平衡的挑战。现有系统通常通过尽可能裁剪 TEE 软件可信基，来最小化资源开销，但是该方案仍存在局限性。本文所提出的 SegTEE 系统，将局部性的概念引入到资源占用中，并提出了基于局部性的资源优化，能够有效解决资源税的问题。

3.5.3　其他资源税优化方法

除了 SegTEE 所提出的新的两个资源税优化方法，原型系统还继承了当前主流的 RISC-V 可信执行环境系统，蓬莱 TEE 在资源上的优化。其中，较为重要的特性是动态内存分配。安全监控器需要内存资源来管理隔离域，随着隔离域数量的上升所需要的资源会增加。一个简单的处理方式是设定一个隔离域上限并提前根据该上限预留对应的资源给安全监控器。然而，这种方式会导致即使很少的隔离域(甚至只有 Host 域)时安全监控器仍然占用相当一部分的资源。蓬莱选择动态分配内存的方式来优化内存占用，当创建新的隔离域时，安全监控器会从 Host 域中获取对应的内存来维护隔离域的元数据。

3.6　I/O 设备隔离

万物互联小型设备上存在种类较多、情况复杂的 I/O 外部设备。这些设备通常需要处理和用户直接相关的敏感信息，例如，用户的地理位置，照片，音频和视频等。当前存在两类主要的隔离 I/O 设备隔离和保护的方式：(1)安全应用不直接和设备交互，而是使用主机代理。这种方式下，安全应用和设备的所有交互都将暴露在不可信的外部环境中，通过加解密等软件方法来对数据进行保护。该机制在云计算和桌面端的 TEE 系统中被广泛应用，例如 Intel SGX 等。然而，这种方式难以适应端侧应用场景，即，I/O 设备会直接获取到敏感数据，需要安全地传递给安全应用；(2)基于硬件能力，对 I/O 设备进行静态划分。例如，ARM TrustZone 中，会通过硬件层面的 S/NS 状态位，来表示设备的所属，不可信环境的软件，包括特权软件，也无法访问到安全世界中的设备，而安全环境中应用可以通过 TEE-OS 来管理和使用安全硬件。然而，这种方式存在设备划分静态，TEE-OS 成为公共可信基等问题。

3.6.1　特权态驱动管理域

SegTEE 中引入了特权态驱动管理域的设计，如图 8 所示。具体来说，SegTEE 支持运行一个特殊的运行在特权态(Supervisor-mode)的隔离域，叫做驱动管理域。驱动管理域和不可信的环境之间，通过 TEE-Seg(即 RISC-V PMP)进行隔离，从而保证两个特权态隔离域之间在内存上的互相隔离。和传统的特权态软件以及 TEE-OS 不同，驱动管理域仅运行必要的设备驱动，用来辅助安全应用安全地访问到设备数据。SegTEE 中，基于 OH LiteOS-M 实现了该模块。具体来说，包含以下设计：

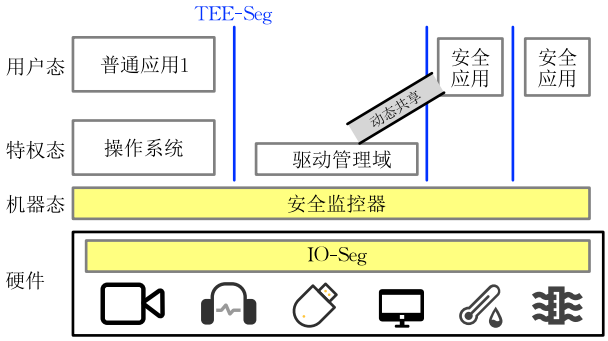

图 8　特权态驱动管理域

(1)设备动态映射。相比 TrustZone 等传统方案中静态分配设备的方案，驱动管理域支持设备动态映射。即，当一个 I/O 设备被分配给某个(或某几个)安全应用时，会先通过安全监控器，将该设备对应的 MMIO 物理内存区域从不可信的宿主环境分配给驱动管理域。从底层来看，安全监控器会通过 TEE-Seg，限制不可信环境对 MMIO 内存区域的访问，并且使能驱动管理域对其的访问。这可以带来相比静态分配方案更高的灵活性。当然，动态映射同样需要 DMA 硬件保护机制的支持，对应设计将会在下一小节进行介绍。

(2)域内细粒度权限管理。由于驱动管理域将同时支持部署和运行多个驱动程序，不同的驱动程序和设备间可能存在安全隐患。为此，SegTEE 提出域内细粒度权限管理机制，基于最小特权原则，来避免不同驱动模块之间的安全影响。具体来说，SegTEE 在运行特定驱动程序时，通过 OS-Seg(sPMP)的自我降权能力，仅打开该驱动程序所需访问的数据和代码区域。

驱动管理域内引入了 Gate 机制，当从一个驱动模块切换到下一个驱动模块时，会经过 Gate，Gate 会重新配置 OS-Seg，从而实现按需切换权限的要求。值得注意的是，该设计并非针对恶意的驱动(运

行在驱动管理器内的驱动属于可信基的部分),而是避免程序出现崩溃等问题,并增大防御纵深,尽可能减少安全风险。

(3)按需内存映射。SegTEE 同样按需共享驱动管理域和安全应用的内存。具体包括:① 只有使用设备的安全应用,会和驱动管理域进行交互,非 I/O 相关的安全应用仍运行在自己的隔离空间内,从而保障它们的高度安全;② 驱动管理域和安全应用之间并非共享所有内存,而是仅共享一个区域用于 I/O 操作,包括 DMA 读写等。驱动管理域对该区域的权限同样遵循域内细粒度权限管理的原则,由 Gate 动态切换。

(4)贡献与创新点。相比传统静态划分方案,SegTEE 所提出的特权态驱动管理域一方面能够实现灵活的设备动态分配,从而最大化设备使用;另一方面结合动态映射、最小特权等设计,有效提升 I/O 灵活保护的安全性。

3.6.2 基于 IOPMP 的 IO-Seg 设计与支持

一个恶意的 I/O 设备,可能通过 DMA 操作来窃取或者篡改安全应用的敏感数据。由于 DMA 操作不经过核内机制(如OS-Seg和TEE-Seg)的检查,需要专门的硬件设计来支撑 I/O 隔离。为此,Seg-TEE 中引入了基于段的隔离机制,使 IO-Seg 作为I/O隔离的硬件扩展。当前,IO-Seg 主要基于 RISC-V IOPMP 进行实现,并在系统层面提供了支持。

(1) IOPMP 设计。IOPMP 是 RISC-V 社区正在推进的一个基于段隔离的 IO 隔离方案,能够允许最高特权软件(RISC-V 机器态)配置设备,从而能够访问 I/O 区域。具体来说,一个主设备,在 IOPMP 中会被分配到一个 SID,用作唯一的标识。该标识用于表示主设备所能够访问的内存区域及其权限。如果一个主设备支持多通道(Channel),并且通道间存在不同的权限,那么可以通过给该设备分配多个 SID 来实现。除 SID 之外,IOPMP 还引入了内存域(Memory Domain,MD)的概念。一个 MD 包含多个内存区域及其对这些区域的访问权限。IOPMP 中,会将 SID 和一组 MD 进行绑定,从而得到一个设备能够访问的内存区域及其权限(读、写等)。

(2)IO-Seg的支持。和OS-Seg及TEE-Seg在上下文切换时按需调整和配置不同,IO 隔离仅和当前 IO 设备所属的隔离域相关。为此,SegTEE 在 IO-Seg 的支持中,引入了每个设备的上下文(Per-device Context),该上下文记录当前设备所属的隔离域(不可信或某特定安全应用),并且根据该上下文配置 IOPMP 的硬件信息。当一个设备的所属权发生变化时,例如,从不可信切换为可信,该 I/O 设备对应的 IOPMP 表项的信息会同步发生变化。只有 I/O 设备的 IOPMP 表现完成更新,安全监控器才会认为该设备可以被新的隔离域使用。

3.7 SegTEE 相比蓬莱的贡献

SegTEE 在蓬莱可信执行环境的基础上,具有以下贡献:

首先,在内存隔离机制上,相比蓬莱可信执行环境使用的 PMP 结合 HPT(基于页表的新硬件扩展),SegTEE 提出 PMP 结合 sPMP 的多层次的段隔离保护,能够以极低的硬件资源开销实现可扩展的隔离域。sPMP 是本文首次提出的关键硬件扩展,目前已经被蓬莱等系统接收使用,并且已经被 RISC-V 国际基金会接纳,成立了 sPMP 工作组(Task Group),有望成为 RISC-V 指令集标准特性之一。

其次,在硬件段隔离设计基础上,本文提出了滑动窗口的段隔离技术(蓬莱中没有该设计),能够充分发挥两层段隔离的能力,实现较多的隔离域。

再次,在优化安全监控器的内存资源上,本文提出了基于局部性的资源优化、定制化安全监控器等技术,这些技术能够有效降低可信执行环境对于端侧资源的占用,同样是相比蓬莱系统的新的创新。

最后,SegTEE 还引入了 IOPMP 的支持,能够在端侧小型设备上抵御不可信外设攻击。蓬莱系统尚未集成类似 IOPMP 的设备防御能力。

综上,SegTEE 在继承蓬莱可信执行环境架构和关键设计的基础上,重点关注在资源受限(包括硬件能力受限、软件资源受限)的场景下如何实现可扩展的隔离,面向小型端侧设备提出了一系列新的技术和创新。

4 系统实现

SegTEE 原型系统基于 RISC-V 指令集目前最前沿的开源可信执行环境系统之一,蓬莱[15],实现并扩展了本文中所提出的一系列面向万物互联端侧小型设备的能力和技术。

(1)硬件实现。本文在 Qemu(RISC-V64)和 FPGA(基于 AWS EC2 F1 实例)实现了 SegTEE 硬件特性。具体来说,Qemu 和 FPGA 上均实现了 sPMP 硬件扩展,并结合 RISC-V 的 PMP 硬件特性,构成了 SegTEE 中所依赖的嵌套段隔离(TEE-Seg 和 OS-Seg)的硬件原语。此外,Qemu 原型系统中还集成了

可信根和安全启动能力，对 TEE 的可信基进行校验。FPGA 原型系统基于开源 RISC-V 硬件实现 Rocket 开发。在 IO 隔离部分，原型系统在 FPGA 上实现了 IOPMP，并且通过微基准测试对其功能正确性进行了验证。

(2) 软件实现。相比蓬莱可信执行环境系统，SegTEE 能够运行在没有内存管理单元的环境下(基于段隔离)。如果设备支持页表，SegTEE 将会使用蓬莱的内存管理模块来兼容现有蓬莱安全应用。此外，SegTEE 对安全监控器资源进行了细致的优化，并支持基于滑动窗口的高可扩展隔离方式，以及基于 sPMP 隔离的隔离域等。

(3) 生态影响。SegTEE 支持两个主流的国产操作系统，openEuler(版本：21.09)和 OpenHarmony(版本：3.2)。SegTEE 将作为万物互联计算生态重要的一部分，补齐当前端侧小型设备安全能力的不足。

5　实验与结果分析

5.1　实验方法与环境

实验中，硬件平台方面同时使用 Qemu 和 FPGA。其中，Qemu 主要用于更加准确高效地分析资源占用、系统可扩展性(隔离域数量)、安全性等指标；而 FPGA 主要用于分析系统真实性能表现等指标。两个平台均基于 RISC-V 64 位指令集(32 位指令集在各个指标上的表现是类似的)。为了模拟小型设备，实验平台仅包含单个 CPU 和 50 MB 内存。这是正常启动基线系统蓬莱-PMP 所需要的资源。为了保证 Qemu 模拟环境下性能数据的可靠性和稳定性，测试中打开了 Qemu 的 icount 选项，即所汇报的时钟周期会和程序运行所执行的指令数相关。

实验中，使用蓬莱-PMP 作为基线版本进行对比，并通过微基准测试和应用测试，对性能表现和安全性进行分析。本文选取了 RV8 作为基准测试集，其中包含一系列典型小型设备上的典型应用，例如 dhrstone、加解密(aes)、哈希计算(sha256)、快速排序(qsort)等。在可扩展性测试等场景中，本文使用 helloworld 作为一个最简化安全应用的例子，其在被调用时将立即返回一个常数。

5.2　安全性测试与分析

(1) 物理内存隔离能力。SegTEE 基于 PMP 作为 TEE-Seg，并且引入了 sPMP 机制用于 OS-Seg。其中，PMP 已经在大量 RISC-V 可信执行环境系统中得到了验证。为此，本文构造了一系列微基准测试用来检验 sPMP 能否按照指定要求进行物理内存保护，如表 4 所示。

表 4　sPMP 安全测试(部分)

测试分组	Config：R/W/X/U/L (Region matched)	U-mode			S-mode			Passed
		Op	Expected	Actual	Op	Expected	Actual	
Perm	1/0/0/0/0	Read	√	√	Read	√	√	√
	1/0/0/0/0	Write	×	×	Write	√	√	√
	1/1/0/0/0	Exec。	×	×	Exec。	√	√	√
	1/1/1/0/0	Read	√	√	Read	√	√	√
	1/0/1/0/0	Write	×	×	Write	√	√	√
Locking	1/0/0/0/1	Write	×	×	Write	×	×	√
	1/0/1/0/0	Write	×	×	Write	×	×	√
SMAP (Sum=0)	1/0/0/1/0	Read	√	√	Read	×	×	√
	1/1/1/0/0	Write	√	√	Write	×	×	√

具体来说，测试分为三部分，首先是基础物理内存读写执行权限，其次是 sPMP 锁定特性，最后是 SMAP(避免特权态读写用户态数据)测试。每个部分包含一系列测试用例(表中仅体现部分内容)，每个测试用例对应一个特定的 sPMP 段配置，主要为配置寄存器中的 R、W、X、L、U 域，并且测试中保证配置的 sPMP 段为第一个覆盖访存区间的段寄存器。本文在用户态和特权态分别进行了特定的访存操作(表中 Op，如读、写、执行)，并对期望结果(Expected)和实际结果(Actual)进行了比较。

结果显示，sPMP 能够有效限制用户态的内存操作，并且其锁定机制和 SMAP 机制能够对特权态软件(操作系统)进行能力限制。实验中，原型系统所实现的 sPMP 能够通过所有安全测试用例。

(2) I/O 隔离能力。实验设计了两个场景分析 SegTEE 的 I/O 隔离能力。场景一(恶意特权态软件)：在该场景中，特权软件被认为是恶意的。对于构造的 Dummy 设备，将其从操作系统转移到驱动管理域，验证前后操作系统是否能够正常使用设备。由于在设备转移后，SegTEE 会重新配置 TEE-Seg，使得操作系统无法访问设备对应的 MMIO 区域，从而避免恶意特权态软件使用其没有权限的设备。场

景二(恶意设备):更进一步,恶意特权软件可能通过其有访问权限的设备(即恶意设备),来发起 DMA 操作,从而试图读写安全应用的数据。在该场景中,SegTEE 通过 IO-Seg(即 IOPMP 机制),限制设备仅能访问自己所属隔离域的内存区域,从而能够避免该攻击。

整体来看,SegTEE 能够基于更轻量的硬件机制,达到前沿可信执行环境系统蓬莱-PMP 相同的安全性保障。从威胁模型角度来看,SegTEE 能够满足本文所要求的安全能力。基于内存、CPU、I/O 的隔离,能够保障隔离域的机密性和完整性,避免被恶意软件攻击。SegTEE 当前原型系统中没有支持内存加密和默克尔树保护,这是由于端侧小型设备通常会将内存等和 SoC 进行较为紧密的结合。并且,SegTEE 可以在必要时结合内存加解密和完整性硬件来实现更强的安全保障。相比蓬莱-PMP,SegTEE 的设计和扩展不会引入额外的安全风险,并且由于降低了安全监控器的大小,能够有效降低系统的软件可信基。

5.3 隔离域可扩展性

可信执行环境的可扩展性通常对数据中心、云计算等场景十分重要,而对于传统小型端侧设备而言相对意义较小。然而,随着万物互联、协同计算的端侧生态形成,安全应用跨设备流转,分布式 TEE 计算等的出现,使得端侧场景对单机下能运行的隔离域数量有着更高的要求,传统较少的隔离域数量难以满足需求。

SegTEE 通过滑动窗口设计,能够在物联设备等小型设备上,通过有限的 RISC-V PMP 段寄存器实现较多的隔离域。为了验证该设计的真实表现,本文选用蓬莱标准的 Demo 应用为案例,来测试蓬莱-PMP 和 SegTEE 能够同时运行的并发隔离域数。为了保证隔离域并发运行,测试中将首先创建要求的所有并发隔离域,在所有隔离域成功创建(或失败)后再运行。测试程序将一直尝试启动到 128 个隔离域。测试中蓬莱-PMP 与 SegTEE 所使用的硬件一致,且都有 16 套 PMP 寄存器。

结果如图 9 所示,图中横坐标为隔离域的编号,纵坐标为每个实例最终完成计算任务的运行时间。

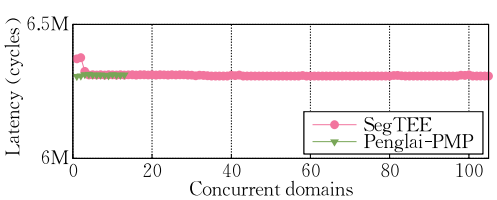

图 9 SegTEE 可扩展性测试(SegTEE 能够支持 128 个并发隔离域)

可见,蓬莱-PMP 只能支持最多 13 个隔离域,当请求更多的实例时,安全监控器会提示因为资源受限而无法创建新的隔离域。而 SegTEE 却能够成功创建到 128 个隔离域。

此外,图 9 中数据显示,SegTEE 在前几个隔离域中完成计算任务的运行时间相比蓬莱-PMP 而言较高。这是由于滑动窗口需要更加复杂的资源分配策略导致的,而当资源分配稳定后,SegTEE 和蓬莱-PMP 的表现是相近的。可见,SegTEE 在实现更多并发隔离域的情况下,其对应用的性能几乎没有影响。

为了测试当前滑动窗口的策略下,SegTEE 能够支持的最大并发数,本文将并发数量提升到 200,此时 SegTEE 能够稳定地创建至少 193 个隔离域。这样的隔离域数量对于物联设备而言在大部分情况下已经足够。在真实场景下,SegTEE 能够启动的隔离域数量一方面和隔离域的内存占用相关,另一方面也和整个系统的资源预留策略相关,系统部署者可以根据需求进行调整和配置。整体来看,相比蓬莱-PMP,SegTEE 能够达到 14 倍的隔离域数量提升。通过进一步优化滑动窗口策略,相信 SegTEE 的滑动窗口技术能够应对隔离域数量要求更高的环境。

5.4 可信执行环境性能

本节对 SegTEE 提供的可信执行环境的生命周期管理操作进行测试和分析。

对于可信执行环境而言,主要包含 4 个操作:隔离域资源分配、隔离域创建、隔离域运行以及隔离域验证。本文采用 RV8 测试集中的 6 个应用,来分别评估它们在 SegTEE 和蓬莱-PMP 下四个操作的时延,结果如图 10 所示。

(1)隔离域资源分配。在创建一个新的隔离域时,Host 域会通过安全监控器,将自己的部分内存资源等分配给新创建出来的隔离域,因此该操作同时涉及 Host 域操作系统和安全监控器两部分的逻辑。实验中,在可信执行环境内核驱动对完整的资源分配时延进行了记录。图 10(a)显示了 SegTEE 和蓬莱-PMP 的资源分配开销,可见本工作引入的扩展对于资源分配不会造成额外开销。

(2)隔离域的创建和运行。图 10(b)和(d)显示了隔离域的创建和运行开销。其中运行开销包含了 RV8 应用的运行时间。从结果可见,SegTEE 和蓬莱-PMP 在创建和运行上的性能表现是接近的,甚至在创建过程中 SegTEE 能够实现更快的性能。这是因为 SegTEE 对于安全监控器支持的 SBI 扩展进行了

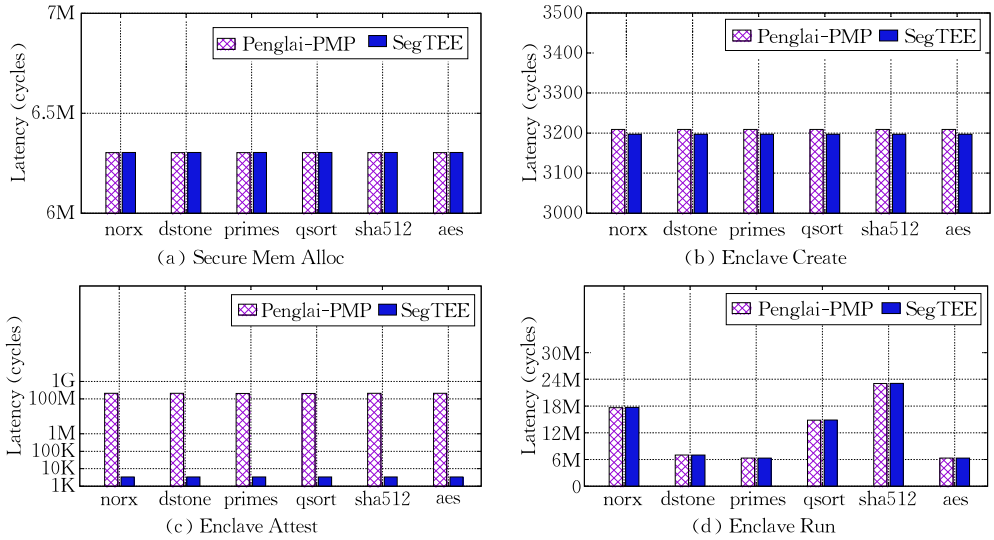

图 10　可信执行环境操作在 RV8 测试集下的性能表现

裁剪，使得安全监控器执行创建操作的路径更短。

(3)隔离域的验证。图 10(c)显示了安全监控器对隔离域的验证开销。可以看到 SegTEE 的验证开销比蓬莱-PMP 明显降低。这是因为，蓬莱-PMP 中使用了较为复杂完善的加解密和哈希库，来验证隔离域的状态，该库会引入较大的内存占用开销。SegTEE 将其替换为了更简单的算法来优化内存占用，但是会存在验证安全性降低的风险。

然而，本文观察到该设计选择在大部分物联设备中是可接受的：通常物联设备会提前将应用安装在镜像中，因此对于安全应用的验证可以放在镜像生成阶段。安全监控器只需要维护安全应用的验证结果报告，在需要时进行返回，而不需要在运行时动态进行验证。

5.5　隔离运行时开销

为了分析 SegTEE 维护的隔离域对应用性能的影响，本节以 RV8 测试集为例，分析应用在原生环境下(非隔离环境，用 Host 表示)和隔离环境下的性能表现，结果如图 11 所示。

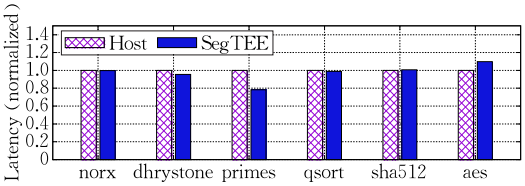

图 11　安全应用和非安全应用性能对比

由于不同应用的运行时间相差较大，图中对数据按照原生环境(Host 数据)进行了标准化。可以看到，SegTEE 在大部分情况下的性能表现和非隔离环境是相同的。其中 aes 会有 9.7%的额外开销，这是由于 SegTEE 所使用的 SDK(基于 musl libc)相比 Host 环境中的 glibc 所导致的差异。而 primes 甚至能够有 22%的性能提升。此处性能提升的主要原因是，原生应用的调度是由操作系统直接完成的，其执行环境更加复杂。而安全应用在隔离环境中运行时只由安全监控器直接管理，运行环境更加简单，并且不太容易被其他应用打断。该性能提升为 50 次测试平均值，在实验过程中可以反复复现，具备一定普适性。此外，该提升和蓬莱可信执行环境在资源较高的场景下(基于页表隔离)带来的提升同样是一致的。

5.6　内存资源占用

SegTEE 对内存资源占用进行了细致的优化。本文对蓬莱-PMP(表中 Penglai-PMP)和 SegTEE 的安全监控器的内存占用进行了统计，如表 5 所示。

表 5　安全监控器内存开销

系统	安全监控器内存开销/KB
OpenSBI	92
蓬莱-PMP	148
SegTEE	69

可见，SegTEE 能够将内存占用优化到 69 KB，相比蓬莱-PMP，SegTEE 实现了 54%的内存资源优化。

6　总　结

万物互联的计算场景对小型设备安全计算能力提出较高的要求，现有可信执行环境系统难以同时满足小型设备上高性能、高安全、低资源占用、动态可扩展的安全计算需求。本文提出 SegTEE，一个面向万物互联场景小型端侧设备的可信执行环境系

统,并围绕段隔离机制设计了全系统的隔离和保护,有效突破现有方案在小型设备上的安全局限。

参 考 文 献

[1] Mei Hong, Guo Yao. Toward ubiquitous operating systems: A software-defined perspective. Computer, 2018, 51(1): 50-56

[2] Jiang Chao, Li Yu-Feng, Cao Chen-Hong, Li Jiang-Tao. Survey of security technologies for IoT edge stream processing based on trusted execution environment. Journal of Cyber Security, 2021, 6(3): 169-186(in Chinese)
(姜超, 李玉峰, 曹晨红, 李江涛. 基于可信执行环境的物联网边缘流处理安全技术综述. 信息安全学报, 2021, 6(3): 169-186)

[3] Yang Wei-Yong, Liu Wei, Cui Heng-Zhi, et al. SG-Edge: Key technology of power Internet of Things trusted edge computing framework. Journal of Software, 2022, 33(2): 641-663(in Chinese)
(杨维永, 刘苇, 崔恒志等. SG-Edge: 电力物联网可信边缘计算框架关键技术. 软件学报, 2022, 33(2): 641-663)

[4] Liu Zhi-Juan, Gao Jun, Ding Qi-Feng, Wang Yue-Wu. Research on development of trusted execution environment technology on mobile platform. Netinfo Security, 2018, 18(2): 84-91(in Chinese)
(刘志娟, 高隽, 丁启枫, 王跃武. 移动终端 TEE 技术进展研究. 信息网络安全, 2018, 18(2): 84-91)

[5] Liu B, Li Y, Liu Y, et al. PMC: A privacy-preserving deep learning model customization framework for edge computing. Proceedings of the ACM on Interactive Mobile Wearable and Ubiquitous Technologies, 2020, 4(4): 1-25

[6] Zhao Wenjia, Lu Kangjie, Qi Yong, Qi Saiyu. MPTEE: Bringing flexible and efficient memory protection to Intel SGX//Proceedings of the 15th European Conference on Computer Systems(EuroSys'20). Heraklion, Greece, 2020, Article 18: 1-15

[7] Hua Zhichao, Gu Jinyu, Xia Yubin, et al. vTZ: Virtualizing ARM TrustZone//Proceedings of the 26th USENIX Security Symposium (USENIX Security 17). Vancouver, Canada, 2017: 541-556

[8] Iannillo A K, Rivera S, Suciu D, et al. An REE-independent approach to identify callers of TEEs in TrustZone-enabled Cortex-M devices//Proceedings of the 8th ACM on Cyber-Physical System Security Workshop(CPSS'22). Nagasaki, Japan, 2022: 85-94

[9] Khurshid A, Yalew S D, Aslam M, et al. TEE-Watchdog: Mitigating unauthorized activities within trusted execution environments in ARM-based low-power IoT devices. Security and Communication Networks, 2022, Article 8033799: 1-21

[10] Park H, Zhai S, Lu L, Lin F X. StreamBox-TZ: Secure stream analytics at the edge with TrustZone//Proceedings of the 2019 USENIX Annual Technical Conference (USENIX ATC'19). Renton, USA, 2019: 537-554

[11] Paju A, Javed M O, Nurmi J, et al. SoK: A systematic review of TEE usage for developing trusted applications//Proceedings of the 18th International Conference on Availability, Reliability and Security (ARES 2023). Benevento, Italy, 2023, Article 34: 1-15

[12] Oliveira D, Gomes T, Pinto S. uTango: An open-source TEE for the Internet of Things. arXiv preprint arXiv:2102.03625, 2021: 1-16

[13] Yuhala P, Ménétrey J, Felber P, et al. Fortress: Securing IoT peripherals with trusted execution environments. arXiv preprint arXiv:2312.02542, 2023: 1-8

[14] Wu Yuming, Liu Yutao, Liu Ruifeng, et al. Comprehensive VM protection against untrusted hypervisor through retrofitted AMD memory encryption//Proceedings of the 2018 IEEE International Symposium on High Performance Computer Architecture(HPCA). Vienna, Austria, 2018: 441-453

[15] Feng Erhu, Lu Xu, Du Dong, et al. Scalable memory protection in the Penglai enclave//Proceedings of the 15th USENIX Symposium on Operating Systems Design and Implementation(OSDI 21). Virtual, 2021: 275-294

[16] Zhao Shijun, Zhang Qianying, Qin Yu, et al. SecTEE: A software-based approach to secure enclave architecture using TEE//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security(CCS '19). London, UK, 2019: 1723-1740

[17] Lee D, Kohlbrenner D, Shinde S, et al. Keystone: An open framework for architecting trusted execution environments//Proceedings of the 15th European Conference on Computer Systems. Heraklion, Greece, 2020: 1-16

[18] Bahmani R, Brasser F, Dessouky G, et al. CURE: A security architecture with customizable and resilient enclaves//Proceedings of the 30th USENIX Security Symposium(USENIX Security 21). Virtual, 2021: 1073-1090

[19] Xia Y, Liu Y, Chen H. Architecture support for guest-transparent VM protection from untrusted hypervisor and physical attacks//Proceedings of the High Performance Computer Architecture(HPCA 2013). Shenzhen, China, 2013: 245-257

[20] Cai Z, Ren B, Ma R, et al. GUARDIAN: A hardware-assisted distributed framework to enhance deep learning security. IEEE Transactions on Computational Social Systems, 2023, 10(6): 3012-3020

[21] Koshiba A, Gust F, Pritzi J, et al. Trusted heterogeneous disaggregated architectures//Proceedings of the 14th ACM SIGOPS Asia-Pacific Workshop on Systems(APSys'23). Seoul, Republic of Korea, 2023: 72-79

[22] Ning Zhenyu, Zhang Fengwei, Shi Weisong. A study of using TEE on edge computing. Journal of Computer Research and Development, 2019, 56(7): 1441-1453

[23] Alves T. TrustZone: Integrated hardware and software security. ARM, White Paper, 2004

[24] Intel. Intel@ 64 and IA-32 architectures software developer's manual. California, USA: Intel Corporation, Technical Report: 085, 2024

[25] Intel. Intel trusted domain extensions whitepaper. California, USA: Intel Corporation, Technical report: 1.0, 2022

[26] AMD. SEV secure nested paging firmware ABl specification. California, USA: Advanced Micro Devices (AMD), Technical report: 1.56, 2024

[27] Kaplan D. Protecting VM register state with SEV-ES. White Paper, 2017: 13

[28] Larry Dewey, Diego Gonzalez Villalobos. SEV-SNP platform attestation using VirTEE/SEV. California, USA: Advanced Micro Devices (AMD), Technical report: 1.2, 2023

[29] Ferraiuolo A, Baumann A, Hawblitzel C, Parno B. Komodo: Using verification to disentangle secure-enclave hardware from software//Proceedings of the 26th Symposium on Operating Systems Principles. Shanghai, China, 2017: 287-305

[30] Brasser F, Gens D, Jauernig P, et al. SANCTUARY: ARMing TrustZone with user-space enclaves//Proceedings of the 2019 Network and Distributed System Security Symposium. San Diego, USA, 2019

[31] Costan V, Lebedev I, Devadas S. Sanctum: Minimal hardware extensions for strong software isolation//Proceedings of the 25th USENIX Security Symposium (USENIX Security 16). Austin, USA, 2016: 857-874

[32] Weiser S, Werner M, Brasser F, et al. TIMBER-V: Tag-isolated memory bringing fine-grained enclaves to RISC-V//Proceedings of the 2019 Network and Distributed System Security Symposium. San Diego, USA, 2019

[33] Zhu J, Hou R, Wang X, et al. Enabling rack-scale confidential computing using heterogeneous trusted execution environment//Proceedings of the IEEE Symposium on Security and Privacy (SP). San Francisco, USA, 2020: 1450-1465

[34] Lee U, Park C. SofTEE: Software-based trusted execution environment for user applications. IEEE Access, 2020, 8: 121874-121888

[35] Will N C, Maziero C A. Intel software guard extensions applications: A survey. ACM Computing Surveys, 2023, 55(14s): 1-38

[36] Zhang Yiming, Hu Yuxin, Ning Zhenyu, et al. SHELTER: Extending arm CCA with isolation in user space//Proceedings of the 32nd USENIX Conference on Security Symposium (SEC'23). Anaheim, USA, 2023: 6257-6274

[37] Chakraborti A, Suciu D, Sion R. Wink: Deniable secure messaging//Proceedings of the 32nd USENIX Conference on Security Symposium (SEC'23). Anaheim, USA, 2023: 1271-1288

[38] Islam M S, Zamani M, Kim C H, et al. Confidential execution of deep learning inference at the untrusted edge with ARM TrustZone//Proceedings of the 13th ACM Conference on Data and Application Security and Privacy (CODASPY'23). Charlotte, USA, 2023: 153-164

[39] Jang J, Kang B B. 3rdParTEE: Securing third-party IoT services using the trusted execution environment. IEEE Internet of Things Journal, 2022, 9(17): 15814-15826

[40] Wang J, Li A, Li H, et al. RT-TEE: Real-time system availability for cyber-physical systems using ARM TrustZone//Proceedings of the IEEE Symposium on Security and Privacy (SP). San Francisco, USA, 2022: 352-369

[41] Gu J Y, Li H, Xia Y B, et al. Unified enclave abstraction and secure enclave migration on heterogeneous security architectures. Journal of Computer Science and Technology, 2022, 37: 468-486

[42] Sun L, Wang S, Wu H, et al. LEAP: TrustZone based developer-friendly TEE for intelligent mobile apps. IEEE Transactions on Mobile Computing, 2023, 22(12): 7138-7155

[43] Schrammel D, Waser M, Lamster L, et al. SPEAR-V: Secure and practical enclave architecture for RISC-V//Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security (ASIA CCS'23). Melbourne, Australia, 2023: 457-468

[44] Zhao S, Li M, Zhangyz Y, Lin Z. vSGX: Virtualizing SGX enclaves on AMD SEV//Proceedings of the IEEE Symposium on Security and Privacy (SP). San Francisco, USA, 2022: 321-336

[45] Antonino P, Derek A, Wołoszyn W A. Flexible remote attestation of Pre-SNP SEV VMs using SGX enclaves. IEEE Access, 2023, 11: 90839-90856

[46] Zhao X, Li M, Feng E, Xia Y. Towards a secure joint cloud with confidential computing//Proceedings of the IEEE International Conference on Joint Cloud Computing (JCC). Fremont, USA, 2022: 79-88

[47] Sartakov V A, O'Keeffe D, Eyers D, et al. Spons & shields: Practical isolation for trusted execution//Proceedings of the 17th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE 2021). Virtual, USA, 2021: 186-200

[48] Vo V, Lai S, Yuan X, et al. Towards efficient and strong backward private searchable encryption with secure enclaves//Proceedings of the International Conference on Applied Cryptography and Network Security. Kamakura, Japan, 2021: 50-75

[49] Dyachkov L, Orenbach M, Silberstein M. Fuzzing libraryOSes for Iago vulnerabilities//Proceedings of the 16th ACM International Conference on Systems and Storage (SYSTOR'23). Haifa, Israel, 2023: 151

[50] Patil R, Modi C. An exhaustive survey on security concerns and solutions at different components of virtualization. ACM Computing Survey, 2023, 52(1): 1-38

[51] Gonçalves C F, Antunes N, Vieira M. Intrusion injection for virtualized systems: Concepts and approach//Proceedings of the 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Porto, Portugal, 2023: 417-430

[52] Wang J, Mahmoody P, Brasser F, et al. VirTEE: A full backward-compatible TEE with native live migration and secure I/O//Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC'22). San Francisco, USA, 2022: 241-246

[53] Chen Z, Qiu H, Ding X. DScope: To reliably and securely acquire live data from kernel-compromised ARM devices//Proceedings of the 28th European Symposium on Research in Computer Security (ESORICS). The Hague, The Netherlands, 2023: 271-289

[54] Zhou Q, Jia X, Jiang N. Protecting virtual machines against untrusted hypervisor on ARM64 cloud platform//Proceedings of the IEEE International Conference on Communications. Seoul, Republic of Korea, 2022: 5451-5456

[55] Tang Y, Qin Z, Lin Z, et al. vTrust: Remotely executing mobile apps transparently with local untrusted OS. IEEE Transactions on Computers, 2022, 71(12): 3349-3360

[56] Hua Z, Yu Y, Gu J, et al. TZ-Container: Protecting container from untrusted OS with ARM TrustZone. Science China Information Sciences, 2021, 64: 192101

[57] Hunt T, Zhu Z, Xu Y, et al. Ryoan: A distributed sandbox for untrusted computation on secret data. ACM Transactions on Computer Systems (TOCS), 2018, 35(4): 1-32

[58] Li Mingyu, Xia Yubin, Chen Haibo. Confidential serverless made efficient with plug-in enclaves//Proceedings of the 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). Virtual, 2021: 306-318

[59] Priebe C, Vaswani K, Costa M. EnclaveDB: A secure database using SGX//Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP). San Francisco, USA, 2018: 264-278

[60] Briongos S, Karame G, Soriente C, Wilde A. No forking way: Detecting cloning attacks on Intel SGX applications//Proceedings of the 39th Annual Computer Security Applications Conference (ACSAC'23). Austin, USA, 2023: 744-758

[61] Chen Sanchuan, Lin Zhiqiang, Zhang Yinqian. Controlled data races in enclaves: Attacks and detection//Proceedings of the 32nd USENIX Security Symposium. Anaheim, USA, 2023: 4069-4086

[62] Sridhara S, Bertschi A, Schlüter B, Shinde S. SIGY: Breaking Intel SGX enclaves with malicious exceptions & signals. arXiv preprint arXiv:2404.13998, 2024: 1-15

[63] Constable S, van Bulck J, Cheng X, et al. AEX-Notify: Thwarting precise single-stepping attacks through interrupt awareness for Intel SGX enclaves//Proceedings of the 32nd USENIX Security Symposium. Anaheim, USA, 2023: 4051-4068

[64] Kim C H, Kim T, Choi H, et al. Securing real-time microcontroller systems through customized memory view switching//Proceedings of the Network and Distributed System Security Symposium. San Diego, USA, 2018: 1-15

[65] Tsai C-C, Porter D E, Vij M. Graphene-SGX: A practical library OS for unmodified applications on SGX//Proceedings of the 2017 USENIX Annual Technical Conference (USENIX ATC 17). Santa Clara, USA, 2017: 645-658

[66] Brito C, Ferreira P, Portela B, et al. SOTERIA: Preserving privacy in distributed machine learning//Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (SAC'23). Tallinn, Estonia, 2023: 135-142

[67] Niu Jianyu, Peng Wei, Zhang Xiaokuan, Zhang Yinqian. NARRATOR: Secure and practical state continuity for trusted execution in the cloud//Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS'22). Los Angeles, USA, 2022: 2385-2399

[68] Howard H, Alder F, Ashton E, et al. Confidential consortium framework: Secure multiparty applications with confidentiality, integrity, and high availability. Proceedings of the VLDB Endowment, 2023, 17(2): 225-240

[69] Koshiba A, Yan Ying, Guo Zhongxin, et al. TEE-KV: Secure immutable key-value store for trusted execution environments//Proceedings of the ACM Symposium on Cloud Computing (SoCC'18). Carlsbad, USA, 2018: 535

[70] Rogers B, Chhabra S, Prvulovic M, Solihin Y. Using address independent seed encryption and bonsai merkle trees to make secure processors OS-and performance-friendly//Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007). Chicago, USA, 2007: 183-196

[71] Arm Limitd. Arm CCA Security Model 1.0. Cambridge, England: Arm Limited, Technical report: A.a, 2021

[72] Waterman A, Asanović K. The RISC-V instruction set manual. Volume II: Privileged architecture, Document Version v1.10. 2017

[73] Garlati C, Pinto S. A clean slate approach to Linux security RISC-V enclaves//Proceedings of the Embedded World Conference. Nuremberg, Germany, 2020

DU Dong-Dong, Ph. D., assistant professor. His main research interests are operating system and architecture.

YANG Bi-Cheng, Ph. D. candidate. His main research interests are operating system and RISC-V architecture.

YU Yang, Ph. D., assistant professor. His main research interests are language virtual machine and system security.

XIA Yu-Bin, Ph. D., professor. His main research interests are operating system, system architecture, cloud computing, and system security.

DING Zuo-Hua, Ph. D., professor. His main research interests are software engineering, computer theory.

ZHAO Yong-Wang, Ph. D., professor. His main research interests are operating system security, formal verification, and programming language.

ZHANG Lei, Ph. D. His main research interest focus on system security.

ZANG Bin-Yu, Ph. D., professor. His main research interest focus on operating system.

CHEN Hai-Bo, Ph. D., professor. His main research interest focus on operating system.

Background

In the context of edge scenarios progressively moving towards distributed and intelligent systems, the protection of user privacy and the isolation of critical code and data on small-scale edge devices have become crucial challenges that need immediate attention. Existing systems typically rely on Trusted Execution Environments (TEEs) that ensure the confidentiality and integrity of security-sensitive applications through processor-based hardware extensions. For instance, ARM TrustZone has been widely adopted in mobile and other scenarios. Intel, AMD, and others have also introduced solutions like SGX and SEV. However, current solutions struggle to meet the requirements of software and hardware-constrained resources on edge devices, making them difficult to adapt.

This paper addresses this issue by proposing SegTEE, which leverages segment isolation mechanisms to balance the demands of security and complex application requirements. Specifically, SegTEE breaks through the limitations of traditional segment isolation protection in terms of the number of isolation domains by introducing a nested segment isolation mechanism that divides segment protection into two layers. By combining sliding window technology, SegTEE significantly increases the number of isolation domains without compromising security or introducing excessive hardware resources.

Moreover, SegTEE introduces minimal segment protection registers at the hardware level and significantly reduces resource requirements at the software level through techniques like dynamic customization and trimming. Additionally, SegTEE optimizes the security capabilities of edge systems by incorporating I/O mechanisms such as IOPMP for enhanced security.

Experimental results demonstrate that SegTEE effectively balances resource, application, and security requirements, offering significant assistance in enhancing the security capabilities of small-scale edge devices.

This work was supported in part by the Key Program of the National Natural Science Foundation of China (No. 62132014), the National Science Fund for Distinguished Young Scholars of China (No. 61925206) and National Key R&D Program of China (Nos. 2022YFB4501500 and 2022YFB4501502).