计 算 机 研 究 与 发 展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202221016
61(1): 243−260, 2024

可信执行环境: 现状与展望

张锋巍　周　雷　张一鸣　任明德　邓韵杰
(南方科技大学斯发基斯可信自主系统研究院　广东深圳　518055)
(南方科技大学计算机科学与工程系　广东深圳　518055)
(zhangfw@sustech.edu.cn)

Trusted Execution Environment: State-of-the-Art and Future Directions

Zhang Fengwei, Zhou Lei, Zhang Yiming, Ren Mingde, and Deng Yunjie
(*Research Institute of Trustworthy Autonomous Systems*, *Southern University of Science and Technology*, *Shenzhen*, *Guangdong* 518055)
(*Department of Computer Science and Engineering*, *Southern University of Science and Technology*, *Shenzhen*, *Guangdong* 518055)

Abstract　Trusted execution environment (TEE) technologies are widely developed in the current computer systems along with the user's serious concerns about privacy protection, secure computing, etc. in network services. Generally, TEEs provide an isolated execution environment for the managers and users for privacy and confidential computing even if the underlying operating systems are compromised. To build the TEEs, the device manufacturers like Intel, Arm update the hardware foundation by adding the external processor mode, memory control, cryptography engine, etc. In addition, they provide corresponding interfaces in the system following the application requirements. Except that, researchers further design compatible TEE models for various goals with the above hardware or firmware assistance. We comprehensively analyze the technical characteristics of TEE technology in mainstream system architecture (including x86, Arm, RISC-V, heterogeneous computing unit), including infrastructure and hardware facilities design, software interface definition, security boundary, etc., and explore the feasible application scenarios of TEE technology. At the same time, we analyze the challenges of current TEE technologies and discuss the limitations and the security risks, e.g., side-channel attacks. Finally, we summarize the advantages and disadvantages of various TEE technologies from the aspects of security and functionality, and consider the future development of TEE.

Key words　trusted execution environment (TEE); OS architecture; memory isolation; processor mode; security attestation

摘　要　当前在云服务、移动社交网络下用户普遍追求隐私保护、安全计算,从而推动了隐私计算、机密计算等领域的快速发展. 可信执行环境(trusted execution environment, TEE)作为机密计算服务中重要的技术基础已经广泛部署到各类计算平台中. 目前,以 Intel, Arm 等为代表的设备制造商采用软硬件隔离机制,推出了多类实用 TEE 技术并不断迭代更新,从功能上更加方便设备管理者、普通用户使用安全服务. 研究人员则根据不同的系统架构和应用需求,优化 TEE 模型,扩大可信应用领域并提升其工作效率. 全面分析主流系统架构(包括 x86、Arm、RISC-V、异构计算单元)中 TEE 技术发展路线、技术特点包括基础硬件设施设计、软件接口定义、安全边界等,挖掘 TEE 技术可行的应用场景. 同时,分析各类 TEE 技术面临的挑战,探讨 TEE 技术局限性以及自身面临的安全风险如侧信道攻击等. 在此基础上,从安全性、功能性等方面总结各类 TEE 技术优缺点,并提出 TEE 技术未来的发展思路.

收稿日期: 2022-12-16; **修回日期**: 2023-05-30
基金项目: 国家自然科学基金项目(62372218, 62002151); 深圳市科技计划资助项目 (SGDX20201103095408029)
This work was supported by the National Natural Science Foundation of China (62372218, 62002151) and the Shenzhen Science and Technology Program (SGDX20201103095408029).
通信作者: 周雷(zhoul6@sustech.edu.cn)

关键词 可信执行环境;操作系统架构;内存隔离;处理器模式;安全性验证

中图法分类号 TP391

随着各行业领域信息化的普及,尤其是以云计算为代表的网络计算服务的大规模应用,使得越来越多的应用程序跨系统平台和网络进行计算并生成海量数据,在推动大数据业务快速发展的同时也导致人们对于设备和数据安全的关注度不断上升.如何保证计算和数据的机密性成为网络空间安全的重要基础.因此针对用户安全计算服务的需求,研究人员提出了可信执行环境[1](trusted execution environment, TEE)技术,用于保护非可信平台中应用和数据可信执行.TEE通常采用隔离部分软硬件资源的方法构建安全区域,确保在其中运行的程序和数据保密性和完整性不受外部干扰.

具体而言,一个可靠的TEE需要提供4方面的安全保障[2]:1)数据隔离.一个可信应用使用的数据不能被其他应用访问、修改,包括可信应用的数据对外部操作系统隔离以及多个可信应用之间的数据隔离.2)计算隔离.可信应用的计算资源不能被其他应用观测和拦截,同时需要清理可信应用执行后的痕迹,并防御来自侧信道的攻击.3)通信控制.非可信应用和可信应用、多个可信应用之间的会话和数据交互不能破坏隔离性.4)错误隔离.非可信区域的安全漏洞不能扩散到可信应用中.针对上述4个安全需求,研究者可以通过密码学方法实现基本的功能,如安全多方计算[3]和同态计算[4]等,但这些算法实现依赖于大量复杂计算,即使可以通过GPU等加速器提升计算效率[5],但仍然会因为存在性能瓶颈而难以大规模应用.因此实用的TEE设计思路是基于硬件隔离基础,通过数据加解密和特权指令执行实现其安全服务.

1 TEE背景介绍

本文将以x86、Arm、RISC-V、异构计算单元架构为代表,分析TEE的发展路线和技术特点.如图1展示了这些架构中TEE技术发展的关键节点.早在20世纪末,Intel和AMD等x86架构处理器中开始设计了系统管理模式(system management mode, SMM)[6].区别于处理器保护模式和实模式,系统执行在SMM中具备独立的内存资源和高级权限,因此被设计为隔离的执行环境,用于部分安全敏感的服务如电源管理、设备检测等.早期的TEE技术主要用于服务提供商开发和部署管理程序,如Intel TXT[7], ME[8],甚至在2002年就提出的TrustZone[9].在后续的10年里,用户难以接触且并不了解设备中TEE的存在.面向用户的可信应用开始于2013年苹果智能手机的Touch ID服务,将用户指纹信息部署在安全区域,防止外部

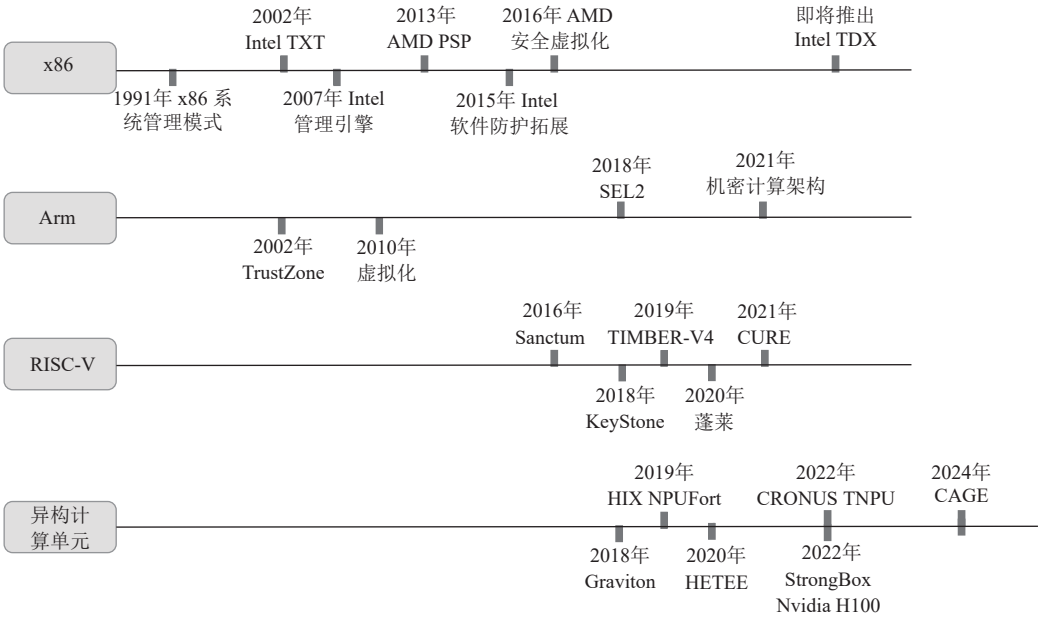

Fig. 1 The development of TEE technology in different architectures

图1 不同架构中TEE技术的发展

攻击篡改. 此后面向用户程序开发的 Intel SGX(software guard extensions)和 SEV 进一步发展了可信计算环境的应用领域.

当前主流计算机系统架构包括 x86, Arm 等均在其设备平台中部署 TEE 技术, 代表性设计包括软件保护拓展 Intel SGX [10] 和 Arm TrustZone, 其中 SGX 通过内存隔离和加密, 以及拓展的指令集在用户空间提供 TEE, 实现不同程序间的隔离运行, 保证用户关键代码和数据的机密性和完整性不受恶意软件的破坏. 而 TrustZone 则是通过设计处理器不同的处理模式, 结合隔离内存等软硬件资源, 将系统划分为安全和非安全 2 个世界. 因此服务提供商通过在安全世界中部署独立于用户操作系统的可信应用, 为用户提供身份认证和授权服务. 其他系统框架如 RISC-V 等在开放应用领域如物联网设备中广泛应用, 其应用系统也因数据采集和前端数据处理的安全性需求设计对应的 TEE, 相关研究包括 Keystone[11], PENGLAI[12] 等. 由于目前的计算单元偏向于应用适配的发展趋势, 尤其是在大数据处理环境中, 以 GPU 为代表的异构计算单元承担了绝大多数的计算任务, 因此对于这些计算模块同样需要考虑计算安全问题. 研究人员提出了以 StrongBox[13], Graviton[14] 等为代表的可信计算技术, 为大数据任务提供安全执行环境. 总的来说, 现有的 TEE 技术具有 3 个明显特点:

1)面向用户设计. TEE 的发展历程中从为管理端提供特权服务到为用户提供机密计算环境, 其应用环境有巨大的改变. 目前在商业服务中部署或将推出的 AMD SEV[15-16]、Intel SGX 和可信域拓展、Arm 机密计算架构等充分考虑在云环境中安全任务计算的需求.

2)硬件辅助支持设计. 从基于 x86 到 RISC-V 架构的平台均引入了新硬件特性, 在图 2 中展示了部分技术特征, 包括内存隔离、特权指令集和硬件加解密模块, 用于安全系统设计.

3)共享资源. 虽然在设计 TEE 的过程中引入了新的加解密引擎、协处理器等. 但 TEE 仍然复用主处理器和内存单元, 与不可信系统的数据交互保持便捷性.

因此基于上述 3 个原则, 平台供应商和研究人员在现有的系统中引入了协处理器、硬件密码算法和内存隔离机制的硬件特性, 同时设计远程和本地证明方法, 保证用户在不依赖操作系统的安全性背景下获取可信服务. 然而现有的 TEE 技术均存在不同程度的缺陷:

1)安全攻击. 由于 TEE 技术和操作系统并非完全的隔离, 如共用处理器、内存、缓存等, 导致 TEE 容易面临来自共享资源的侧信道攻击.

2)性能影响. TEE 中数据保护普遍采用内存加密的方式防止数据泄露, 即使通过硬件电路支持加解密操作, 其数据访问效率仍然低于明文计算.

3)服务便捷性. 早期 TEE 中部署的简化操作系统所提供的服务有限且由服务商定制, 难以部署大型应用. TEE 面向用户设计后, 用户可以在应用层实现关键代码的安全执行, 但对于系统资源的调用仍然需要非可信系统的支持. 目前在推出安全虚拟化的 TEE 技术后, 用户安全应用领域将进一步发展.

本文通过研究现有的 TEE 技术, 如图 2 所示, 将重点分析现有 TEE 技术的优缺点. 由于应用场景和安全需求的不同, 各类系统中 TEE 设计框架和技术实现原理存在较大差异. 而在网络计算环境中, 用户通常跨平台和系统进行机密计算, 因此分析各类 TEE 的技术特点, 有助于用户构建统一完整的可信服务环境. 本文将分别从 x86、Arm、RISC-V 和异构计算单元 4 类架构分析, 重点介绍其部署的 TEE 的技术原理、面临的挑战和未来的发展, 为 TEE 发展提供新的思路.

2 基于 x86 架构的 TEE

基于 x86 架构的设备主要面向家用、商用服务领域, 其优点在于高性能以及强兼容性. 目前主要以 Intel, AMD 厂商为代表推动 x86 架构的发展. 因此基于 x86 的计算平台以 Intel 软件防护拓展和 AMD 安全加密虚拟化为代表为用户机密计算服务提供 TEE.

2.1 x86 架构的 TEE 技术发展

随着 x86 架构日益成为个人、企业或者数据中心服务器处理器的通用架构, 面向 x86 架构平台的安全性需求设计成为高优先级的目标. 追溯到 1991 年, 在 AMD 设备中部署系统管理模式, 然后从 2005 年开始, Intel 和 AMD 分别推出了可信计算技术(trusted execution technology, TXT)[7] 和 SKINIT[16], 采用 TPM 芯片、特殊的指令集和验证算法, 构建从硬件到系统启动过程的可信证明链[52]. 其中 Intel 的 TEE 发展路线较为明确, 从 TXT 到目前的 SGX[10], 以及后续的 TDX[29], 设计了从系统底层到用户自定义开发的 TEE. 构建 x86 可信计算环境的软硬件技术呈现多样化发展趋势, 但各技术之间交叉融合, 本节将从系统架构硬件层到应用层之间介绍各类 x86 架构 TEE 技

可信技术类型	技术	独立芯片	芯片类型	独立存储	子系统	处理器模式	功能	风险 (部分)	相关研究 (部分)
可信平台技术	TPM1.0 TPM2.0	○	厂商定制	○	●	保护模式	支持平台验证、磁盘加密等	CVE-2017-16837 CVE-2018-6622	fTPM[17] vTPM[18]
基于协处理器的可信子系统	Intel ME	○	Intel Quark 处理器	○	○	协处理器模式	支持AMT、Boot Guard等技术	Ring -3 rootkit CVE-2017-5689	Nighthawk[19] Zero-touch-provisioning[20]
	AMD PSP	○	Arm Cortex 处理器	○	○	协处理器模式	提供密钥、验证SPI ROM等	Backdoor	psptrace[21] PSPreverse[15]
基于处理器模式的可信系统	x86 SMM	◐	主处理器	○	◐	系统管理模式	电源管理、系统硬件控制、OEM订制管理代码等	缓存攻击 SMM exploit等	SICE[22] KShot[23] MALT[24]
基于内存加密的TEE	Intel SGX	◐	内存加解密引擎	◐	●	保护模式 飞地模式	用户层可信应用	Spectre Meltdown 侧信道攻击	SMILE[25] Occlum[26]
	AMD SEV	◐	内存加解密引擎	◐	●	保护模式&虚拟机特权模式	用户可信虚拟机	侧信道攻击	CIPHERLEAKS[27] CrossLine[28]
	Intel TDX	◐	内存加解密引擎	◐	●	保护模式 安全仲裁模式	用户可信虚拟机	未知（处于开发阶段）	TDX[29] TDX-tools[30] TDX-module[31]

(a) 基于x86架构的TEE技术

文献项目名称	隔离机制	安全世界	特权级别	功能
vTZ[32]	TZASC+NS.EL2	NW	VM	虚拟化TrustZone硬件
OSP[33]			VM	提供可信应用运行环境
PrivateZone[34]			VM	提供可信应用运行环境
SANCTUARY[35]	TZASC+单核		App	指定核心运行NS.EL0飞地
TrustICE[36]			App	单核运行NS.EL0飞地
TEEv[37]	TZASC+软件隔离	SW	VM	在S.EL1提供轻量级hypervisor
ProOS[38]			VM	在EL3支持hypervisor
SecTEE[39]			OS	支持侧信道保护的TEE OS
ReZone[40]	TZASC+额外硬件		OS	在S.EL1创建多个区间
Hafnium[41]	TZASC+S.EL2		VM	S.EL2虚拟化多个TEE OS
TwinVisor[42]			VM	在S.EL2隔离VM
CCA[43]	RME+RMM	RW	VM	在R.EL2隔离VM

(b) 基于Arm架构的TEE技术

文献项目名称	隔离机制	支持安全I/O	Enclave特权等级	Enclave数量受限	缓减缓存侧信道攻击	硬件修改
Sanctum[44]	DRAM区域	○	U	DRAM区域数量	●	●
TIMBER-V[45]	内存标签	○	U	无	●	●
Keystone[11]	PMP	○	U+S	PMP数量	◐	○
CURE[46]	缓存标签	●	U, S, U+S, M	TileLink协议限制	●	●
PENGLAI[12]	修改MMU	○	U	无	○	●

(c) 基于RISC-V架构的TEE技术

文献项目名称	系统架构	异构计算单元	内存架构	保护方法	硬件修改
HIX[46]	Intel x86	NVIDIA GPU	独立内存	SGX Enclave技术	○
Graviton[14]	Intel x86	NVIDIA GPU	独立内存	修改GPU指令处理器	○
HETEE[47]	Intel x86	NVIDIA GPU	独立内存	增加额外FPGA硬件	○
H100[48]	x86, Arm	NVIDIA GPU	独立内存	GPU内置保护硬件	●
CRONUS[49]	Arm	通用加速器	独立内存	TrustZone技术	○
NPUFort[50]	x86, Arm	DNN加速器	独立内存	加速器内置保护硬件	●
TNPU[51]	Intel x86	NPU	共享内存	SGX Enclave技术	●
StrongBox[13]	Arm	Arm Mali GPU	共享内存	TrustZone技术	○

(d) 基于异构计算单元的TEE技术

○是 ◐混合 ●否

Fig. 2 TEE technology evolution on x86, Arm, RISC-V and heterogeneous computing unit architectures

图 2 基于 x86、Arm、RISC-V 和异构计算单元架构的 TEE 技术演进

术的发展路线.

2.1.1 可信平台技术

在处理器层面，x86 架构首先引入可信平台技术，x86 是基于可信平台模块[53-54](trusted platform module, TPM)的发展. 目前设备中普遍部署 TPM2.0 版本，TPM 作为一种集成到芯片组中的微控制器，主要用于密钥生成、存储和证书验证，同时保存系统各层次模块的测量值. TPM 基于硬件的密码学特点可以保证存储数据免受外部恶意软件的攻击，因此是广泛应用于系统启动和应用部署的安全策略的基础.

2.1.2 基于协处理器的可信子系统

由于 TPM 作为系统子模块由主系统调用，因此是与主系统耦合性强、可扩展性弱以及缺乏系统运行行的安全策略. 针对上述问题，Intel 和 AMD 分别推出了 Intel 管理引擎(management engine, ME)[8] 和 AMD 平台安全处理器(platform security processor, PSP)[16] 子系统，采用协处理器和隔离技术，构建管理端 TEE. 以 ME 为例，Intel 在主处理器外引入 Quark 芯片和 MINIX 操作系统，作为单独的子系统运行在用户设备中. ME 子系统拥有独立的物理地址和 IP 地址，通过直接连接网卡访问远程资源，因此管理端可以通过 ME 子系统远程透明地管理设备开关机、主系统启动验证、设备温度、电压等物理特征检测，甚至对主系统运行时状态监控. 由于 ME 子系统拥有独立的计算能力和内存空间，因此在 Intel 架构安全体系中作为基础可信域存在，如主系统其他可信模块的根密钥、证书等均可保存在 ME 子系统中，并对证书验证、时钟同步等操作提供基本的 TEE.

2.1.3 基于处理器模式的可信系统

针对协处理器和主处理器之间的隔离性，ME 和 PSP 子系统只能管控平台设备的物理属性，因此子系统难以同步获取和验证主系统状态，也不能操控主系统的正常运行. 而基于主 CPU 的不同执行模式构建的 TEE 则能有效控制主系统. Intel 和 AMD 平台利用系统管理模式 SMM 部署了可信服务代码. SMM 构建环境与 Arm TrustZone 类似，基于 SMM 计算模式以及内存区域的隔离性，设备制造商(OEM)可以在设备出厂前部署管理需求的任务，如 CPU 电源管理、启动代码块验证、TPM 数据清理等一系列高权限的处理模块. 研究人员也可以拓展 SMM 处理程序接口，用于检测内核程序完整性，调试用户程序的执行流等.

2.1.4 基于内存加密的用户 TEE

基于协处理器、隔离执行模式的 x86 TEE 均面向管理者服务，不提供开源服务. 因此用户难以通过上述方法开发用户可信应用程序. 而面对用户机密计算任务的广泛需求，Intel 和 AMD 分别提出 Intel SGX 和 AMD SEV[55] 技术. Intel SGX 为用户提供应用层 TEE，用户可以将机密计算程序片段部署在 SGX 飞地(Enclave)执行，但对于内核函数的调用则需要切换至非可信系统中执行. AMD 为用户提供系统级的 TEE，通过内存加密和虚拟化技术，直接构建面向用户的安全虚拟机系统. AMD SEV 相比于 SGX 飞地在使用场景上有极大的拓展，且虚拟化和加密性能降低在可接受范围内，尤其适用于云环境下对租户提供安全计算平台的需求. 因此后续 Intel 也将推出可信区域拓展技术(trust domain extensions, TDX)[28]，实现虚拟化系统的可信隔离.

2.1.5 x86 TEE 的主要设计思路

x86 TEE 是基于物理隔离方法和加密验证技术构建的. 物理隔离方法包括协处理器、隔离执行模式、物理内存隔离，如图 3 所示. 而访问权限控制和加密方法则复杂多样，包括 Diffie-Hellman 密钥交换、HMAC 可信链验证、ECC 密钥分配等通用密码算法. 协处理器单元的引入极大地提升 TEE 中加解密和特殊指令集计算的效率. 在 x86 设备中各 TEE 技术并非独立存在，如 Intel ME, AMD PSP 作为协处理系统，与主系统最大限度地隔离，因此可以作为可信第三方验证主系统启动的可信性，包括验证 x86 SMM 固件的完整性，为 SGX 提供可信证书和时间同步功能等. x86 SMM 通过增加硬件设施保存和恢复其他模式运行时的处理器状态，从而快速正确地处理系统切换. 内存区域 SMRAM 的隔离通过特殊寄存器设置访问权限和范围，有利于 OEM 厂商的个性化定制. Intel SGX, TDX 和 AMD SEV 由于面向用户设计，因此除了底层硬件实现，服务提供商设计了大量的用户接口用于自定义开发，其软硬件设计的复杂性大

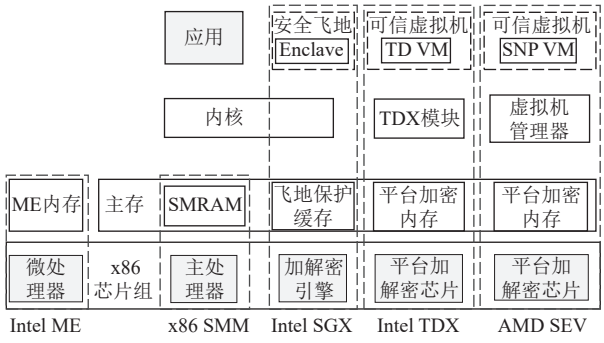

Fig. 3 x86 architecture based TEE

图 3 基于 x86 架构平台的 TEE

幅度提升.

2.2 x86 TEE 面临的安全风险与挑战

TEE 面临的主要安全风险来自于 2 个方面: 内部软硬件设计漏洞和外部侧信道攻击. 软硬件设计漏洞问题包括控制逻辑代码缺陷、接口设计不严谨等, 例如可信区域执行代码不严格遵守约定的安全可信代码库和不可信代码库之间的接口调用, 当指令在可信区域和非可信区域之间切换时, CPU 状态标志不能及时清除, 堆栈指针无法恢复, 指针和数组的范围缺少检查以及寄存器数据泄露等. 这些来自于可信系统固件、驱动或者接口自身的漏洞将严重影响可信程序的正确执行. 另外攻击者基于 TEE 自身的漏洞结合用户代码的缺陷, 在 TEE 中执行非可信程序主动暴露机密数据, 如 SmashEx[56], AsyncShock[57] 等在飞地内实现 ROP 攻击, 导致飞地控制流错误和数据泄露. 具体地, 我们从 3 个方面分析 TEE 面临的具体安全攻击.

2.2.1 针对隔离机制的攻击

TEE 基础设施的隔离并非完全封闭. 首先, 可信域和非可信域之间存在交互通道, 并提供了可信域接口用于信号传递和数据传输. 其次, 系统预留调试接口用于厂商对可信系统的开发、测试和维护. 再次, 可信域框架引入额外的攻击面, 包括系统切换后的状态保存恢复的堆栈空间、时针同步机制等. 最后, 基于硬件和固件组成的可信基础存在设计缺陷. 因此, 当隔离机制在上述问题中存在漏洞, 将会导致 TEE 的暴露, 如 2017 年因 Intel 直联接口技术漏洞导致 ME 系统运行时指令控制流被拦截[58], 其攻击方式同样能针对 SMM 技术并造成严重后果. 而在 SGX CROSSTALK 攻击[59] 中, 飞地运行时使用的隐藏共享缓存设施被暴露, 因此导致飞地内机密数据可能被泄露. SGX-Step[60] 则是因为允许单步配置 APIC 计时器、中断和跟踪页表项, 从而泄露飞地控制流信息. 而对于 AMD SEV, 由于安全虚拟机对 I/O 操作缺乏保护, 因此攻击者能够利用非安全的 I/O 实施任意的安全内存加解密. 此外可以通过修改 NPT 入口地址的特殊字节, 产生并利用 SEV 的内存页错误来跟踪 SEV 内存访问, 从而达到侧信道攻击效果[27].

2.2.2 基于共享资源的侧信道攻击

由于平台资源的高度集成化, 计算机系统各组件通过紧耦合设计实现. 因此内存、处理器等通常在非可信系统和可信系统中共享使用. 2009 年 Intel ME[8] 系统因内存地址重映射漏洞从而泄露了 ME 的外部内存, 其作为系统主内存中隐藏的区域, 可通过重新映射加入到主系统内存页表中, 因此攻击者可以利用该内存区域并部署任意的代码在 ME 子系统中运行. 处理器中缓存即便是在引入了 TEE 技术后, 仍然没有为可信区域和非可信区域提供缓存隔离机制. 因此基于缓存的攻击能有可能导致可信执行区域的机密信息泄露. 目前基于缓存的攻击能碰撞出 SMM, SGX, SEV 的可信程序机密数据, 其共同点是遍历缓存读写、刷新并定位可暴露的可信内存数据[61-63]. 例如 2010 年, 针对于 SMM cache 攻击[6], 导致 SMRAM 内存段暴露给非可信应用, 使得攻击者可以绕过内存访问控制权限, 在 SMM 可信处理单元中部署调用非可信区域代码段. 而针对 SGX 的缓存攻击则包括 CacheOut[64], CacheZoom[65] 等一系列攻击方法, CacheZoom 攻击从非可信区域检测是否存在可信区域内存数据存留在相同的缓存片段上, 有则暴露并利用. 文献 [66] 使用 Prime+Probe 方法攻击 Intel 的验证飞地, 暴露飞地的 EPID 信息. 文献 [67] 在 AMD SEV 虚拟化研究中, 提出可以通过 TLB 缓存投毒的方式, 暴露安全虚拟机加密内存.

2.2.3 基于硬件设计漏洞的攻击

指令预测执行极大地优化了系统执行效率, 但对于大部分 Intel 和 AMD TEE 设计引入了新的安全漏洞, 幽灵攻击 Spectre [68] 的核心是预测执行指令会导致 CPU 状态或 CPU 缓存发生可测量的变化. 因此攻击者可以通过状态信息变化暴露预测指令涉及的数据, 这些数据可能来自可信区域. SgxPectre[69] 验证了 SGX 面临分支预测执行攻击的风险, 利用 SgxPectre 攻击从 Intel 签名引用飞地中提取秘密密封密钥和认证密钥. Koruyeh 等人[70] 提出了 SpectreRSB 攻击, 该攻击交替使用返回堆栈缓冲区以推测预测执行帧(函数)的返回地址. 熔断攻击 Meltdown[71] 则利用了现代 CPU 的无序执行特性. 与 Spectre 不同的是, Meltdown 没有利用推测执行功能, 而是在 CPU 发出故障并回滚这些指令的结果之前, 通过无序执行的指令访问未经授权的内存. 基于文献 [69−71] 所述的方法, Foreshadow 攻击[72] 通过 mprotect 功能调整页面访问权限, 使得攻击者可以通过页面故障和预测执行攻击, 从缓存中获取飞地内部数据. 类似的攻击同样可以运行在 AMD SEV 和 Intel SMM 机制中 [73-74].

另外针对电源管理漏洞的攻击同样会导致 TEE 安全边界破坏, 如 Murdock 等人[75] 提出的 Plundervolt 攻击. 为了破坏飞地计算, Plundervolt 攻击滥用 x86 CPU 上动态电压缩放的特权接口, 使得攻击者可以在 SGX 飞地的计算过程中将故障注入到被攻击的飞

地控制流. 相关工作还包括 VoltPillager[76], 由于从物理特性上破坏了内存的完整性, 从而导致 TEE 隔离机制破坏, 影响整个可信区域的安全性.

2.3 预测 x86 TEE 未来发展

针对 x86 TEE 技术, 我们从功能、架构和安全方面探讨未来的发展方向.

1)功能方面. x86 TEE 从早期面向设备供应商定制的安全服务(如 ME, PSP, SMM 等), 到目前的面向普通用户的 TEE (SGX, SEV, TDX), 其设计目的在逐步变化. 在未来发展中, 面向管理端和用户端的 TEE 方案将同步发展, 如目前在 Intel 设施中同时提供了 ME 和 SGX 服务. 值得注意的是, 由于不同 TEE 技术在面向不同用户时所暴露的接口、访问权限等均不同, 因此安全等级有所区别. 例如, 到目前为止 ME 仅产生 3 个严重漏洞, 而 SGX 面临大量的攻击. Intel 设计过程中, SGX 的可信根如时针信息等保存在 ME 中, 另外还包括 SMM 模块的固件验证值. 因此, x86 TEE 将进一步推动管理端 TEE 和用户端 TEE 设计, 协同构建系统可信执行区域.

2)架构方面. 管理端 TEE 以协处理系统设计为主, 独立于主操作系统, 面向平台远程管理、能耗分析、主系统资源验证等提供可信环境, 进一步提升可信服务计算性能并减少对主系统的干扰. 用户端 TEE 则基于处理器异构模式、内存隔离、数据加解密引擎设计面向用户层的计算环境. 对于目前 CPU 集成大小核机制和独立的安全芯片, 管理端 TEE 能构建功能丰富的可信系统. 对于用户端, TEE 将面向大型安全应用、丰富的资源调度和远程安全连接等功能需求, 因此基于虚拟化的 TEE 将进一步推动发展, 如 AMD 进一步完善 SEV-SNP, Intel 将要推出 TDX 等.

3)安全方面. 提升对侧信道攻击的防御能力, 减少系统被攻击的接口数量. 例如针对管理端 TEE, 限制调试模式的访问权限, 降低共享资源的使用, 严格验证寄存器使用规则, 最大限度地降低攻击者从用户端对管理端 TEE 的访问权限. 而在用户端 TEE 方面, 需要检查已有指令集处理模式对 TEE 功能的影响, 如预测执行和乱序执行等, 验证可信域与非可信域之间交互接口的有效性, 以及分析处理器异常、中断等对于主系统内可信区域程序运行的影响.

3 基于 Arm 架构的 TEE

Arm 在智能手机和物联网领域有着至关重要的作用. 在数据中心、云端服务器和车联网领域, Arm 也在快速发展. 针对安全应用需求, Arm 侧重于设计通用的 TEE 硬件基础. 本节将介绍 Arm 架构 TEE 发展与现有研究的设计思路、面临的风险挑战以及未来 Arm TEE 的展望.

3.1 Arm TEE 发展

3.1.1 TrustZone 技术

在 2002 年左右, Arm 开始发展 TEE 技术. 为了增强移动设备运行的安全性, Arm 公司在 Armv6 架构中引入了 TrustZone. 2011 年 11 月, Arm 发布了新一代处理器架构 Armv8, 这是 Arm 公司的首款支持 64 位指令集的处理器架构. 同时, 随着 Armv8-M 架构的公布, 为了满足市场对嵌入式安全解决方案的需求, Arm Cortex-M 微处理器系列也引入了 TrustZone 技术.

TrustZone 是一种嵌入到 Arm 处理器中的技术, 已在数十亿的移动终端和嵌入设备中运行. 供应商和原始设备制造商(OEM)依靠 TrustZone 部署 TEE, 保护名为可信应用(TA)的敏感程序的执行. 一些可信应用实现了操作系统的内核级服务, 例如, 用于用户认证或文件磁盘加密[9]. 其他可信应用提供各类用户功能, 例如, 数字媒介解码器[77] 或在线银行服务[78]. TEE 本身包括一个可信的软件栈, 为托管可信应用提供接口和运行环境支持, 例如高通公司的 QSEE[79] 和开源的 OP-TEE[80].

TrustZone 技术提供了 2 种执行环境, 即正常世界和安全世界. 正常世界运行一个丰富的软件栈, 称为富执行环境(REE), 由完整的操作系统(如 Linux)和应用程序组成, 它通常被认为是不可信任的. 安全世界运行一个较小的软件栈, 由可信操作系统和可信应用组成. TrustZone 执行系统范围内的世界间隔离, 并在富执行环境调用可信应用的服务时为世界切换提供受控的入口点. TrustZone 启用了 2 种处理器安全状态: 正常状态(NS)和安全状态(S). 在 Arm 架构中, 有一种基于权限划分的防御机制, 称为异常级别(EL0~EL3). 一般来说, 处于异常级别 EL0 和 EL1 的处理器核心运行在这 2 种安全状态中的任何一种中, 例如, 在 NS.EL1 中执行一个不受信任的操作系统, 在 S.EL1 中执行一个受信任的操作系统. 从 Armv-8.4 开始, EL2 可以在安全状态下使用, 因为安全分区管理器(SPM)得到了支持. EL3 始终处于安全世界, 并运行一个安全监控器, 在改变安全状态方面发挥作用. 典型的基于 TrustZone 的系统使用资源分区的静态策略, 只允许安全世界内存驻留在几个固定的内存区域. 例如, TrustZone 地址空间控制器, TZASC (TZC-400)[81] 支持配置多达 8 个不同的内存区域.

3.1.2 CCA 架构设计

2021 年上半年, 在 Armv8 发布 10 年后, Armv9 架构面世. 同时, Arm 公司发布全新的 Arm 机密计算架构(confidential compute architecture, CCA)的初步技术规格. CCA 的关键目标之一就是为第三方提供机密计算的能力. 此外, 随着 Arm 逐渐布局服务器端处理器, CCA 旨在改变行业在应用程序中构建可信计算环境信任模型的处理方式, 随时随地保护云端用户的数据与代码的机密性和一致性. 不同于 TrustZone, CCA 在硬件上直接支持内存加密能力, 并且 TrustZone 内的特权软件也无法访问 CCA 中的数据.

如图 4 所示, Arm CCA 在一个名为领域世界的新隔离环境中进行计算. CCA 旨在保留现有的系统软件(如虚拟机管理器)来管理领域虚拟机的硬件资源, 同时防止软件和其他硬件组件观察或修改领域虚拟机的内容. 为了管理领域虚拟机的执行, CCA 引入了一个名为领域管理监控器(RMM)的软件组件[82]. 在领域状态下, 运行在异常级别 EL2 的领域管理监控器也使用现有的虚拟机管理技术, 如用阶段-2 转换表来隔离领域虚拟机. 领域管理扩展[79]是 CCA 的硬件组件, 扩展了 TrustZone 中引入的隔离模型. 与 TrustZone 相比, 领域管理扩展将异常级别 EL3 扩展到新的根安全状态, 监控器运行在根世界. 根世界防止从任何其他世界访问 EL3 内存. 当处理器执行内存访问时, 领域管理扩展通过颗粒保护检查(granule protection check, GPC)[83]确定该访问是否被允许. 领域管理扩展阻止非法访问, 并返回一个访问故障(granule protection fault, GPF). CCA 维护一个颗粒保护表(granule protection table, GPT), 作为内存中的结构, 它规定了每个细粒度的物理内存(例如 4KB)所属的安全世界, 以便与领域管理扩展配合. CCA 支持通过更新颗粒保护表动态地将一块物理内存转换为一个新的安全地址空间(如正常世界、安全世界、领域世界).

3.1.3 Arm TEE 架构设计

第 1 类设计方案是利用虚拟化提供正常世界的 TEE. 著名的例子包括 vTZ[32], OSP[33], PrivateZone[34], 它们利用正常世界的虚拟化扩展在正常世界创建隔离环境. vTZ 虚拟化了整个 TrustZone 的硬件使其能够在正常世界中执行可信操作系统, 而 OSP 和 PrivateZone 则为每个可信应用提供一个自定义的运行环境. 第 2 类系统如 Sanctuary[35]和 TrustICE[36]使用 TZASC 提供正常世界的用户级别飞地, 它们依靠监控器和可信的操作系统对 TZASC 进行动态编程来与 REE 隔离. 文献 [84] 通过对富运行环境操作系统底层接口进行修改、隔离, 将敏感应用运行在正常世界中. 第 3 类系统是创建软件构建的安全世界隔离环境. 例如, TEEv[37]和 PrOS[38]旨在支持没有 S.EL2 的 Arm 平台上的多个 TEE 软件栈. 这些方案依赖于在 S.EL1(TEEv)或 EL3(PrOS)中运行的管理程序, 并对在 S.EL1 中运行的可信操作系统进行二进制检测以保证同特权隔离. SecTEE [39]除了提供 TEE, 还能防御基于侧信道的攻击. ReZone [40]旨在创建硬件施加的安全世界飞地. 通过依靠 PPC/ACU 硬件, ReZone 创建了安全世界的隔离区域, 以有效限制 S.EL1 的权限. Arm 在 Armv8.4-A 之后引入了 S.EL2. 通过新的安全权限, Hafnium[41]在安全世界中执行独立的安全分区管理器, 以虚拟化多个可信的操作系统. TwinVisor [42]利用 S.EL2 将安全保护与资源管理解耦, 并在正常世界中重用 hypervisor.

截至目前, 由于 CCA 的生态系统处于早期阶段, 开发者还无法使用它. 领域管理监控器的可用代码还未公开发布, 领域管理监控器实现领域虚拟机的细节有待进一步的分析.

3.2 Arm TEE 面临的安全风险与挑战

尽管 TEE 已经被广泛用于保护移动设备免受恶意软件的侵害[85-87], 不幸的是, 在过去几年中仍然有

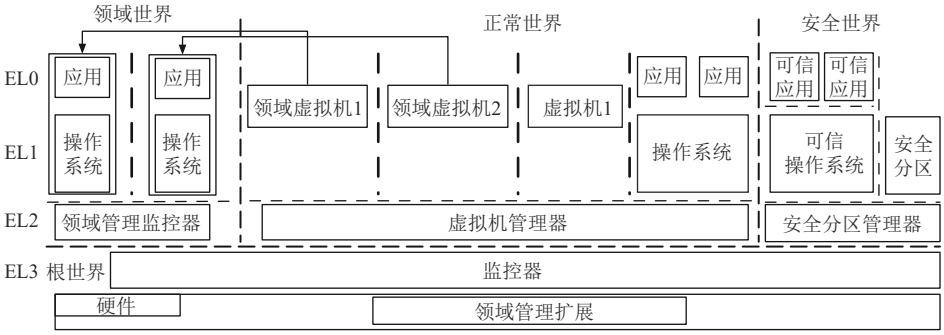

Fig. 4 Arm CCA TEE architecture

图 4 Arm CCA TEE 架构

一些商用 TrustZone 的 TEE 出现了漏洞. 我们观察到主要有 3 个方面的安全风险.

1)Arm TEE 系统长期以来一直存在严重的软件实现错误[84]. 许多漏洞在可信应用程序和负责管理 TEE 运行时的可信操作系统内核中被发现. 这些错误涉及典型的输入验证错误, 如缓冲区溢出. 正如多个攻击漏洞所显示的那样, 这些错误可以被用来劫持安卓的 Linux 内核, 或者完全破坏如高通[85-86]、Trustonic[87-88] 或华为[89] 设备的 TEE 内核.

2)TrustZone 的 TEE OS 缺少一些保护机制, 这为利用脆弱的 TA 提供了便利. 例如, 现代操作系统中常见的内存保护, 如 ASLR 或页面保护, 在大多数 TEE 系统中几乎没有或实现得不好的[88]. TEE 系统也倾向于暴露出一个大的攻击面, 包括可以被 TA 调用的危险的 TEE OS 系统调用. 例如, 在高通公司的 TEE 上, 任何 TA 都可以映射到主机操作系统的内存区域. 因此, 通过劫持脆弱的 TA, 例如利用缓冲区溢出, 攻击者可以很容易地控制 Android[86].

3)大多数 TrustZone 系统在架构和微架构层面忽略了重要的硬件属性, 这可能会损害 TEE 的安全性. 一些漏洞是由于微架构的侧信道(例如在缓存中)导致可信硬件组件的意外行为造成的[90-94]. 还有一些威胁是由可以利用的硬件组件造成的, 例如通过嵌入现代 SoC 的可重构硬件 FPGA, 从 TEE 限制的内存中泄露敏感数据[95-96]; 没有正确处理缓存属性也可能导致系统从缓存中暴露数据, 例如基于缓存中间件的攻击[97].

3.3 Arm TEE 的展望

针对 Arm TEE 技术, 我们分别从应用和安全方面来探讨其未来的发展趋势.

1)应用方面. 随着 TEE 技术的发展, 基于移动终端的 TEE 应用已经较为成熟[98-100], 具体应用场景包括指纹/人脸识别、手机钱包等涉及隐私计算的服务. 2013 年, 苹果公司推出的基于 TEE 的指纹识别 TouchID, 可被视为 TEE 技术在手机端的一次具体应用.

Arm 服务器端的 TEE 应用还处于起步阶段, 根据 Arm 的路线图, CCA 的领域管理扩展需要等到 Armv9.2 才能作为一个可选硬件出现. 尽管如此, Arm 公司为 CCA 的发展已经确立了一条清晰的路线, 同时与开源社区也有了密切合作. 领域管理监控器将由 Arm 公司与操作系统厂商微软及行业合作, 推进与领域管理监控器及其扩展功能交互的标准接口, 为领域虚拟机提供架构. 此外, Arm 与 trustedfirmware.org 等开源项目合作, 提供 CCA 监控器固件的标准实现, 并为机密计算创建新项目, 如 Project Veraison[101] 将为构建 CCA 证明(attestation)提供开源软件. 最后, Arm 还提出了动态 TrustZone 技术: 由领域管理扩展成的 TrustZone 扩展功能, 可以移除 TrustZone 对静态分区内存的需求, 并使得 TrustZone 可应用于占据大容量动态内存的应用程序.

2)安全方面. 目前的商用 TEE 中实施的防御机制严重落后于商用主流操作系统中所包含的和研究界所提出的最先进的防御机制. 因此 TEE 应该考虑增加部分架构防御机制, 如安全通道、内存加密, 以及现代操作系统的内存保护.

此外, 商业化 TEE 系统大多是用 C 语言编写的, 可以编译出高效率的代码, 但不提供内存安全. 然而, 许多实现错误是由程序员引入的内存违规错误造成的. 研究人员已经探索了使用类型安全的编程语言来实现 TEE 软件的特定组件的想法. 例如 RustZone[102] 是 OP-TEE 的一个扩展, 其可信应用是用 Rust 编程语言实现的. 鉴于 Rust 提供内存和线程安全, RustZone 可以帮助防止验证错误和一些可信应用软件的并发错误. 另外, 实现错误往往是由于软件的预期需求和实际实现之间的不匹配而出现的. 软件验证由模型检查、符号执行和形式化验证等技术组成[103], 旨在通过确保实现完全满足所有设想的需求来防止这种错误匹配. 因此, 它有可能帮助防止可能存在的 TEE 实现错误. 我们相信, 通过采用最新的防御措施, 商业化的 TEE 可以变得更加安全, 并且能够抵御许多普遍存在的漏洞.

4 基于 RISC-V 架构的 TEE

RISC-V 是新兴的开源指令集架构, 因为其高度可定制性而受到学术界、嵌入式设备厂商等的青睐. 课题组或厂商可以使用 RISC-V 来自行设计 TEE 架构, 从而摆脱对传统厂商 TEE 设计的限制和依赖. 本节将介绍 RISC-V 架构的相关背景、RISC-V 架构下现有的 TEE 设计、现有 TEE 设计中存在的问题以及未来 RISC-V TEE 可能的发展方向.

4.1 RISC-V 架构

RISC-V 架构是一个开源的指令集架构[104], 该项目于 2010 年在加州大学伯克利分校启动. RISC-V 指令集架构具有高度可定制性, 硬件厂商可以根据自己的需求选择支持不同的拓展模块. 厂商必须选择 RV32I, RV64I, RV128I 这 3 种整数计算模块中的一个作为基础模块. 可选的拓展模块包括 M-Extension(整

数乘法与除法)、A-Extension(原子指令)、F-Extension(单精度浮点数计算)等[105]. 由于 RISC-V 开源的特性，厂商也可以设计自定义的拓展模块来满足其额外的对功能性、安全性等方面的需求.

类似于 Arm 架构的异常等级(exception level)设计，RISC-V 中也有不同的特权等级：用户态 U-Mode(user mode)、内核态 S-Mode(supervisor mode)和机器态 M-Mode(machine mode). 如果设备支持 H-Extension，则在 S-Mode 和 M-Mode 之间存在一个 H-Mode(hypervisor mode). 对于嵌入式设备，厂商也可以选择只支持 U-Mode 和 M-Mode 这 2 种特权等级.

TEE 使用硬件来保护运行在其中的程序，而 RISC-V 的高度可定制性使得自行设计 TEE 硬件单元成为可能，因此使用 RISC-V 指令集架构设计 TEE 已经在学术界引起关注. RISC-V 官方社区提供的特权指令规范文件中提供了一个基础的硬件隔离单元设计：物理内存保护(physical memory protection, PMP). 用户可以通过设置一些特定的 M-Mode 控制状态寄存器(control and status register, CSR)来配置 PMP. CPU 中的每个核上都各自有一组 PMP CSR，用来标识若干段连续的物理地址范围，以及该核对每个范围的访问权限.

具体来说，PMP CSR 包含若干组 pmpcfg 和 pmpaddr 寄存器. 其中 pmpcfg 用于配置地址配对模式及访问权限，pmpaddr 用于标记对应的地址范围. 核上的 pmpcfg 和 pmpaddr 寄存器数量决定了硬件能够提供的隔离区域数量上限. PMP 可选的地址配对模式有 TOR(top of range)，NA4(naturally aligned four-byte region)，NAPOT(naturally aligned power-of-two region)这 3 种. 其中 TOR 模式下，2 个 pmpaddr 寄存器用于标识一段连续的物理地址区域；NA4 和 NAPOT 模式仅需要 1 个 pmpaddr 寄存器即可标识一段连续的物理地址区域，但前提是该区域大小是 2 的幂次，并且起始地址对齐至该大小. 除了官方提供的 PMP 设计之外，厂商也可以使用自己设计的硬件隔离单元.

4.2 RISC-V 架构 TEE 设计及存在的问题

现有的 RISC-V 架构的 TEE 设计侧重点包括 TEE 的整体架构、可信 I/O、规模化运行以及对缓存侧信道攻击的防御. 本节将分别进行讨论，并在最后讨论现有 RISC-V TEE 中存在的问题.

4.2.1 TEE 的整体架构

在第 2 节和第 3 节中我们已经介绍了 Intel SGX 和 Arm TrustZone 的设计. 目前部分研究者也在 RISC-V 架构下研究构造类似架构和编程模型的 TEE：如 Costan 等人[44]的Sanctum 和Weiser 等人[45]的TIMBER-V. 同时，部分工作抛开主流平台中如 SGX 以及 Trustzone 的 TEE 设计思路，研究全新的 TEE 架构，如 Lee 等人[11]的 Keystone、Bahmani 等人[46]的 CURE 等. 其中 Keystone 是一个基于软件的 TEE 的设计，仅用到了 PMP 作为硬件隔离单元，它给每一个创建的 Enclave 分配一组 PMP CSR，从而实现飞地与宿主机之间、飞地彼此之间的相互隔离.

不同于 SGX 只提供用户层服务，Keystone 中的飞地包含 S-Mode 的特权级别. 另外，不同于 Trustzone 安全/非安全世界的双重隔离，Keystone 支持多个不同的隔离域，类似于 Arm CCA. CURE 进一步扩大飞地应用领域，同时支持仅运行在 U-Mode 的飞地、包含 S-Mode 运行时的飞地和运行在 M-mode 下的深层飞地. 其中深层飞地用于将部分关键的 M-Mode 代码从其他代码中隔离开，使得 TCB 中不必包含全部的 M-Mode 代码.

图 5 展示了部分现有的 RISC-V TEE 的架构设计. 其中最右侧为宿主系统架构，左侧 5 款 TEE 设计中，Sanctum 与 SGX 类似，飞地所需要的特权功能委托宿主内核进行处理；TIMBER-V 则在 S-Mode 中运行一个可信操作系统来处理这些特权功能；Keystone 则兼顾了这 2 种模式：部分特权功能在一个微小的运行时中处理，其余的运行时委托给宿主操作系统

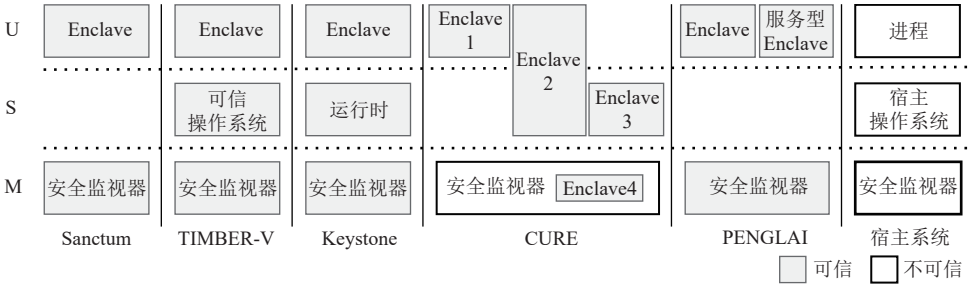

Fig. 5 TEE architecture on RISC-V platform

图 5 RISC-V 平台 TEE 架构

内核；CURE 中支持多种不同特权等级的飞地，可以运行普通的飞地应用（Enclave 1）、不依赖宿主功能的飞地（Enclave 2）、提供系统功能的特权飞地（Enclave 3）以及用于缩小可信基的飞地（Enclave 4）；PENGLAI 使用的是类似微内核的思路，由一些服务型飞地提供类似文件系统等功能，供飞地应用使用.

4.2.2 安全 I/O

由于 SGX 中的飞地不包含 S-Mode 的处理模块，因此飞地中所有 I/O 请求都必须交给宿主系统处理，这会带来额外的安全隐患. 为了解决 I/O 过程中可能的信息泄露，目前有部分工作使用自定义的硬件来辅助实现安全的 I/O，包括上文提到的 CURE 和 Nasahl 等人[106] 的 HECTOR-V，这 2 部分工作通过修改 I/O 总线设施，实现“飞地-设备绑定”机制，即在同一时刻 1 个设备只能被 1 个飞地应用访问. 但是上述机制并未考虑如何对有状态的外设进行绑定和解除绑定.

4.2.3 规模化运行飞地

随着云服务中无服务器计算 Serverless 的兴起，越来越多的厂商开始使用 Serverless 架构部署自己的服务. Serverless 架构中，往往使用单个容器或虚拟机来启动单个的应用. 这种“单应用系统环境”跟飞地有很多相似之处，因此可以考虑使用飞地来保护 Serverless 架构下提供服务的应用. 此前在 x86 架构下已有类似的工作使用 SGX 保护容器运行，如 Arnautov 等人[107] 的 SCONE. 使用飞地来主持 Serverless 服务要求架构可以规模化地启动飞地以及飞地自身需要灵活、可拓展，Feng 等人[12] 的 PENGLAI 使得这成为可能. 在 PENGLAI 之前的 TEE 并不适合主持 Serverless 服务，因为飞地的数量受到限制（如 Keystone 和 CURE 中飞地数量受到硬件单元数量限制）或飞地本身大小受到限制（如 SGX 中飞地内存最高可以拓展到 256 MB）. PENGLAI 使用了自定义的 MMU，从而支持大范围地开启飞地，最高支持 1 000 个以上的飞地同时运行. 此外，PENGLAI 还使用可挂载的 Merkle 树（mountable Merkle tree）来大范围保证内存完整性.

4.2.4 针对缓存侧信道攻击的防御

侧信道攻击使用时间、功率等非常规的数值作为信息源来泄露关键信息，它是 TEE 的重要安全隐患之一. 其中基于缓存的侧信道攻击是侧信道攻击中的一个大类，如 Yarom 等人[61] 提出的 FLUSH+RELOAD 攻击. 这类攻击存在的根本原因是并没有对缓存进行隔离设计. 缓存作为一个所有运行环境共享的用于优化的硬件单元，难免会在彼此之间泄露信息. 但是对于已经出厂的设备，这类漏洞往往难以根治，而运行在其中的飞地的安全性也难以得到保证. 因为 RISC-V 的硬件可定制性，开发者可以自行设计缓存的结构，从而防御部分已知的基于缓存的侧信道漏洞. 如此前提到的 Sanctum 和 CURE，使用了缓存标签、缓存分区等方案实现缓存的隔离机制，从而可以防御基于缓存的侧信道攻击.

4.2.5 现有 RISC-V TEE 设计中存在的问题

目前 RISC-V 的 TEE 中还存在碎片化、Iago 攻击和侧信道攻击等问题. RISC-V 自身的硬件可定制性带来了便利的同时也不可避免地带来了碎片化的问题. 每个厂商或高校课题组设计出不同的硬件来实现自己的目标，但硬件彼此之间并不兼容，也没有统一的编程模型，因此难以推广，目前更多停留在学术界的尝试中. 与 SGX 中的飞地一样，RISC-V 下的 TEE 也可能会存在 Iago 攻击. 如 Bulck 等人[108] 证明了 Keystone 中存在 Iago 攻击，这是因为 Keystone 在运行时将一部分系统调用委托给了宿主内核. 其他借助了宿主内核系统调用的 TEE 也面临同样的风险. 最后，RISC-V TEE 也面临层出不穷的侧信道攻击. 目前已发现针对 RISC-V 高性能处理器 BOOM 的基于乱序执行的侧信道攻击[109].

4.3 RISC-V TEE 的未来发展

总体来看，RISC-V 下的 TEE 设计还在蓬勃发展的阶段. 除了此前提到的各类学术界工作，RISC-V 社区本身、一些硬件厂商也都在推进自己的 TEE 设计[110-112]. RISC-V 官方社区推进的 TEE 设计也有助于解决设计碎片化的问题. 此外，未来 RISC-V 的 TEE 设计还可以会聚焦 3 个方向：侧信道攻击的防御、基于 RISC-V 协处理器的 TEE 和针对边缘计算场景的 TEE.

首先，因为侧信道攻击大多与硬件相关，所以很难从软件上根治侧信道相关的漏洞. 要根治这一类漏洞，必须进行硬件修改. RISC-V 在这方面有得天独厚的优势.

其次，现在也有很多厂商选择在片上系统 SoC 中加入一个 RISC-V 核心作为协处理器，辅助完成一些高度定制化的任务. 这些协处理器也可以用于给任意一个架构的 SoC 增加 TEE 的功能. 目前业界已经有这个方面的尝试[113].

最后，RISC-V 目前备受青睐的领域是嵌入式设备等边缘计算场景. 这些场景下设备的数据采集、数据处理也可能会导致隐私信息的泄露，而这些设备往往成本相对较低，功能相对较少，难以直接移植使用现有的 TEE 设计. 因此，专门针对嵌入式设备的

TEE 设计也可能是未来的一个研究方向. 此前提到 TIMBER-V 也是该方向的一个尝试.

5 异构计算单元的 TEE

5.1 异构计算单元 TEE 的现状

计算应用的蓬勃发展对计算性能提出了很高的要求, 尤其是要求系统能够拥有快速处理大规模数据的能力. 因此, 当今的计算机系统开始使用各种异构计算单元来加速计算过程, 其中最为广泛使用的是 GPU, 除此之外还有 DPU, NPU 等各类专用加速器. 这些异构计算单元通常是独立于 CPU 之外的硬件外设, 拥有极高的数据吞吐量和强大的计算能力, 目前已经广泛应用于数据分析[114]、机器学习[115]、深度神经网络[116] 等大规模计算任务的加速. 如今, 不管是边缘端的设备, 还是用于高性能计算的服务器, 都开始配备异构计算单元以满足用户日益增长的计算需求.

在异构计算单元被广泛应用于计算机系统中的同时, 异构计算所面临的安全性问题也需要受到重视. 对于大多数异构计算单元来说, 其设计仅考虑了计算的高效性, 而没有对安全性进行考量. 目前, 许多异构计算单元的使用高度依赖于安装在操作系统中的驱动程序, 计算单元中的上下文创建、数据传输、计算命令提交等操作都需要借助驱动程序来与相关硬件进行交互. 然而, 这种交互过程会受到外部环境的威胁, 攻击者可以利用操作系统层面的许多漏洞来取得特权, 从而干预驱动程序与异构单元的交互, 进而破坏异构计算单元中的上下文隔离或直接访问其内存中的计算数据. 考虑异构计算单元需要服务于各种涉及隐私信息(如人脸信息、目标图像等)的计算应用, 解决异构计算所面临的威胁迫在眉睫.

近年来, 学术界提出了许多针对异构计算单元的 TEE 构建方案, 其中大部分工作以构建 GPU TEE 为主. Volos 等人[14] 提出了 Graviton, 对 GPU 上的指令处理器进行修改以对 GPU 内存数据提供安全保护, 并禁止操作系统直接与 GPU 硬件进行交互, 所有交互行为都必须通过在 GPU 上的指令处理器的检查. Jang 等人[117] 设计了 HIX 系统, 将 GPU 驱动程序部署于利用 Intel CPU 提供的 SGX 飞地中, 并在已有的 SGX 技术基础上增加部分新的硬件指令, 实现对 GPU 数据和计算的保护. Zhu 等人[47] 提出了 HETEE 系统, 该系统将 GPU 资源抽象成计算节点, 通过结合一个可信 CPU 和额外的 FGPA 硬件来对每个计算节点提供保护和安全认证操作, 为人工智能模型的训练和推理提供了隔离的执行环境, 并保证了相关数据的机密性. 随着 GPU TEE 逐步受到重视, 工业界也开始着手解决 GPU 计算所面临的安全性问题. 著名的 GPU 生产厂商 NVIDIA 于 2022 年发布了 H100 GPU[48], 这是全球第一个具有机密计算功能的 GPU 产品. NVIDIA 通过在 GPU 内建提供安全功能的硬件设施, 针对人工智能相关的机密数据以及模型数据进行保护, 并能够确保多方协作的安全性, 在确保 GPU 计算机密性的同时也能提供优秀的计算性能.

上述 GPU TEE 都是考虑解决服务器 GPU 的安全计算问题. 但随着边缘端计算的兴起, 边缘端设备上的 GPU 也不可避免地需要承担许多敏感计算任务, 同样有构建 GPU TEE 的需求. 然而现有的工作都是依赖于服务器 GPU 拥有独立内存的特性, 即计算数据存放于自己的独立内存中. 而边缘端设备使用的是共享内存 GPU, 其计算数据直接存放于与 CPU 共享的内存之中, 针对独立内存 GPU 的相关设计方案难以直接应用于共享内存 GPU 上. StrongBox[13] 工作填补了这一空缺. StrongBox 是为 Arm 边缘端统一内存 GPU 构建 TEE 的工作, 解决了共享内存 GPU 面临的相关安全性问题. 除此之外, 不同于先前基于 Intel x86 平台系统的工作, StrongBox 利用了 Arm 架构现有的硬件特性, 在不需要硬件修改的前提下为共享内存 GPU 构建了 TEE. 图 6 展示了 StrongBox 在 Rodinia[118] 基准中的测试结果, 实验结果表明其整体引入较低的性能开销(4.70%~15.26%). 除 StrongBox 外, 近期的 CAGE[119] 工作也为下一代 Arm 设备(即支持 Arm CCA 的设备)的统一内存 GPU 提供安全保护, 扩展了 Arm CCA 的功能. CAGE 借助了 Arm CCA 提出的内存保护机制, 将统一内存 GPU 的工作流程与 Arm CCA 的系统设计相结合, 为 Arm CCA 中新提出的领域世界提供 GPU 机密计算支持, 有助于 Arm CCA 在

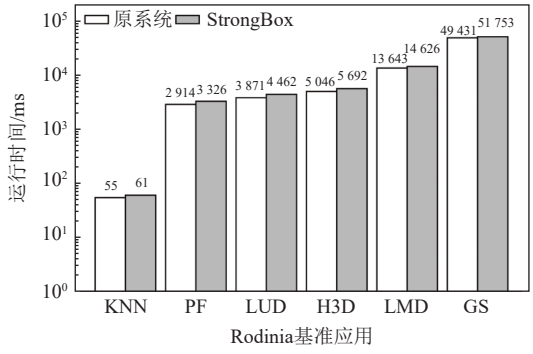

Fig. 6 Evaluation of StrongBox in the Rodinia benchmark

图 6 StrongBox 在 Rodinia 基准中的评估

机密计算和异构计算领域的推广.

除了针对 GPU 的 TEE, 目前也有一些工作考虑了为其他异构计算单元构建 TEE. CRONUS[49] 设计了一个通用的加速器 TEE, 该工作通过将加速器驱动程序部署到 Arm TrustZone 中, 并利用 Arm 提供的安全世界虚拟化特性 SEL2, 保证了服务器中多个加速器的安全隔离和错误隔离. NPUFort[50] 考虑的是保护 DNN 加速器中的模型计算安全, 通过在加速器中增加保护指令和数据的安全单元来对模型计算中的重要参数配置信息和时间信息进行隐藏以保护模型隐私. 文献 [50−51] 所述的工作针对的是独立内存架构的加速器, 而 TNPU[51] 关注的是共享内存架构 NPU 的计算安全性问题. 该工作利用了 SGX 飞地来对 NPU 的上下文和计算内存进行隔离, 并使用 SGX 提供的内存加密技术来提供更低开销的完整性保护机制.

5.2 异构计算单元 TEE 的问题及展望

目前异构计算单元的 TEE 构建尚处于发展阶段, 尚未有成熟且广泛应用的产品. 尽管 NVIDIA 已经推出了具有机密计算功能的 H100 GPU[48], 但这仍然只是 NVIDIA 的初步尝试, 距离其成为主流推广使用的 GPU 仍有一定的距离. 因此, 为当前广泛使用的异构计算单元构建 TEE, 保护敏感计算的安全性, 仍然是需要重点解决的问题. 根据现有的工作, 我们可以总结出未来构建异构计算单元工作需要关注的 3 个问题, 即安全性、兼容性以及性能开销.

1)安全性. 安全性需要确保异构计算单元的计算数据机密性和完整性, 为数据的传输构建安全通路并提供相应的内存保护. 数据的安全通路构建主要依赖于加密手段, 所有敏感数据在传输到受保护的内存之前都处于加密状态, 并且只会在受保护的内存中被解密. 明文数据只会在计算时被使用, 并且会在计算之后被加密或者安全销毁. 针对异构计算单元内存的保护需要禁止外部操作系统的恶意访问, 并确保机密计算任务内存与其他计算任务内存的隔离性, 避免攻击者利用恶意的计算任务发起攻击.

安全性还需要考量可信计算基的大小, 这是衡量 TEE 安全性的重要指标. 小的可信计算基可以降低安全漏洞的存在可能性, 也可以更容易验证安全性. 部分现有的工作选择将整个加速器的驱动程序引入 TEE 中. 然而, 许多加速器的驱动程序都存在安全漏洞[120-122], 这意味着直接引入复杂的驱动程序会导致额外的安全性问题. 因此, 相关工作可以考虑利用驱动程序完成非敏感性的操作, 辅以轻量级的安全检查程序来对敏感操作进行监视和保护, 以减少可信计算基的大小.

2)兼容性. 在兼容性方面, 针对异构计算单元的 TEE 需要保证尽可能高的兼容性, 以便能够大规模部署在现有的系统上. 然而, 许多现有的异构计算单元 TEE 的设计与实现都依赖于硬件层面的修改, 这种基于硬件修改的解决方案会降低方案的兼容性, 导致相关工作难以应用在现有的系统上. 未来的相关工作可以考虑纯软件的 TEE 设计方案, 利用系统架构现有的软硬件特性, 实现高兼容性、可广泛部署的解决方案.

3)性能开销. 在性能开销方面, 需要尽可能降低 TEE 保护方案对计算性能的影响. 根据现有工作的分析, 性能开销主要来源于构建数据安全通路时对计算数据的加解密以及完整性验证过程. 这是因为异构计算单元通常用于处理大规模计算数据, 针对这些数据的加解密操作会带来不可忽视的延迟开销. 针对降低性能开销的问题, 未来的相关工作可以设计专用的硬件加解密引擎, 以降低加解密对计算资源的占用, 减少 TEE 所带来的额外开销.

6 总　　结

TEE 技术伴随着用户隐私计算、机密计算等任务需求而得到有效地发展, 从 Arm TrustZone 等实用性 TEE 推出到目前 x86、RISC-V 和以 GPU 为代表的异构计算单元计算框架中 SGX, SEV, Keystone 等技术的发展, TEE 技术在服务器领域、个人电脑、移动终端设备等计算平台中广泛部署. 本文针对上述各类系统架构, 详细介绍了相关部署的 TEE 技术的发展历程和技术路线, 并具体分析了各类技术在应用范围内面临的挑战和安全风险. 需要重点指出的是目前基于软硬件协同的可信技术框架广泛采用硬件隔离技术基础, 降低了对不可信系统的依赖, 从而减少了在运行期间可能存在的攻击. 但是上述 TEE 技术并非绝对安全, 攻击者可能利用基于缓存溢出、页表故障、非法指令的访问甚至电源控制等软件侵入或者侧信道攻击手段, 导致 TEE 的机密性和完整性破坏. 因此在未来的发展中, TEE 技术将需要重点研究 TEE 的高效性和安全性, 尤其是针对共享资源隔离、运行时安全认证等方面提出快速处理和可信性保证的方法, 为用户在网络环境中安全任务提供一个 TEE.

作者贡献声明：张锋巍负责总体论文的撰写和修改，确立论文框架和指导文章内容；周雷负责资料调研和论文整理，撰写背景知识与基于x86构架的相关技术；张一鸣、任明德、邓韵杰分别负责撰写基于Arm、RISC-V和异构计算单元的相关技术.

参 考 文 献

[1] Dai Weiqi, Jin Hai, Zou Deqing, et al. TEE: A virtual DRTM based execution environment for secure cloud-end computing[C] //Proc of the 17th ACM Conf on Computer and Communications Security. New York: ACM, 2010: 663−665

[2] Bryan P. Bootstrapping trust in a trusted platform[C/OL] // Proc of the 3rd Conf on Hot Topics in Security. Berkeley, CA: USENIX Association, 2008[2023-05-17].https://dl.acm.org/doi/10.5555/1496671.1496680

[3] Cramer R, Damgård I B. Secure Multiparty Computation and Secret Sharing[M]. Cambridge, UK: Cambridge University Press, 2015

[4] Xu Yi, Paulet R, Bertino E, et al. Homomorphic Encryption and Applications[M]. Berlin: Springer, 2014

[5] Wang Zhiwei, Hou Rui, Li Peinan, et al. HE-Booster: An efficient polynomial arithmetic acceleration on GPUs for fully homomorphic encryption[J]. IEEE Transactions on Parallel and Distributed Systems, 2023, 34(4): 1067−1081

[6] Wojtczuk R, Joanna R. Attacking SMM memory via Intel CPU cache poisoning [EB/OL]. Invisible Things Lab, 2009 [2023-04-23]. https://invisiblethingslab.com/resources/misc09/smm_cache_fun.pdf

[7] Futral W, Greene J. Intel Trusted Execution Technology for Server Platforms: A Guide to More Secure Datacenters[M]. Berkely, CA: Apress, 2013

[8] Skochinsky I. Intel ME secrets[EB/OL]. RECON, 2014[2023-04-21].https://recon.cx/2014/slides/Recon 2014 Skochinsky.pdf

[9] Android. Android enterprise security[EB/OL]. 2020[2023-03-01]. https://www.android.com/static/2016/pdfs/enterprise/Android_Enterprise_Security_White_Paper_2019.pdf

[10] Costan V, Devadas S. Intel SGX explained[EB/OL]. IACR Cryptology ePrint Archive, (2017-02-21)[2023-05-17].https://eprint.iacr.org/2016/086.pdf

[11] Lee D, Kohlbrenner D, Shinde S, et al. Keystone: An open framework for architecting trusted execution environments[C] //Proc of the 15th European Conf on Computer Systems. New York: ACM, 2020: 1−16

[12] Feng Erhu, Lu Xu, Du Dong, et al. Scalable memory protection in the PENGLAI enclave[C] //Proc of the 15th USENIX Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2021: 271−294

[13] Deng Yunjie, Wang Chenxu, Yu Shunchang, et al. StrongBox: A GPU TEE on Arm endpoints[C] //Proc of the 29th ACM SIGSAC Conf on Computer and Communications Security (CCS'22). New York: ACM, 2022: 769–783

[14] Volos S, Vaswani K, Bruno R. Graviton: Trusted execution environments on GPUs[C] //Proc of the 13th USENIX Conf on Operating Systems Design and Implementation (OSDI'18). Berkeley, CA: USENIX Association, 2018: 681–696

[15] Buhren R, Eichner A, Werling C. Uncover, understand, own-regaining control over your AMD CPU[R/OL]. 2019[2023-04-23].https://pretalx.com/36c3/talk/10942/

[16] Lai R. AMD security and server innovation[R/OL]. 2013[2023-04-20].https://uefi.org/sites/default/files/resources/UEFI_Spec_2_8_final.pdf

[17] Raj H, Saroiu S, Wolman A, et al. fTPM: A firmware-based TPM 2.0 implementation, MSR-TR-2015−84[R]. Redmond: Microsoft Research, 2015

[18] Perez R, Sailer R, van Doorn L. vTPM: Virtualizing the trusted platform module [C] //Proc of the 15th USENIX Security Symp (USENIX Security'06). Berkeley, CA: USENIX Association, 2006: 305−320

[19] Zhou Lei, Xiao Jidong, Kevin L, et al. Nighthawk: Transparent system introspection from ring-3[C] //Proc of the 24th European Symp on Research in Computer Security (ESORICS'19). Berlin: Springer, 2019: 217–238

[20] Vassilios V. Security evaluation of Intel's active management technology[D]. Stockholm, Sweden: KTH Information and Communication Technology, 2010

[21] Christian W, Robert B. Dissecting the AMD platform security processor [EB/OL]. Chaos Communication Camp, (2019-08-24)[2023-05-17].https://www.reddit.com/r/Amd/comments/cvdy5t/dissecting_the_amd_platform_security_processor/

[22] Azab A M, Ning Peng, Zhang Xiaolan. SICE: A hardware-level strongly isolated computing environment for x86 multi-core platforms[C] //Proc of the 18th ACM Conf on Computer and Communications Security. New York: ACM, 2011: 375−388

[23] Zhou Lei, Zhang Fengwei, Liao Jinghui, et al. KShot: Live kernel patching with SMM and SGX[C] //Proc of the 50th Annual IEEE/IFIP Int Conf on Dependable Systems and Networks (DSN). Piscataway, NJ: IEEE, 2020: 1−13

[24] Zhang Fengwei, Kevin L, Angelos S, et al. Using hardware features for increased debugging transparency[C] //Proc of the 36th IEEE Symp on Security and Privacy(S&P'15). Piscataway, NJ: IEEE, 2015: 55−69

[25] Zhou Lei, Ding Xuhua, Zhang Fengwei. SMILE: Secure memory introspection for live enclave[C] //Proc of the 43rd IEEE Symp on Security and Privacy (S&P'22). Piscataway, NJ: IEEE, 2022: 1536−1536

[26] Shen Youren, Tian Hongliang, Chen Yu, et al. Occlum: Secure and efficient multitasking inside a single enclave of Intel SGX[C] //Proc of the 25th Int Conf on Architectural Support for Programming Languages and Operating Systems (ASPLOS '20). New York: ACM, 2020: 955–970

[27] Li Mengyuan, Zhang Yinqian, Wang Huibo, et al. CIPHERLEAKS: Breaking constant-time cryptography on AMD SEV via the

ciphertext side channel[C] //Proc of the 30th USENIX Security Symp (USENIX Security'21). Berkeley, CA: USENIX Association, 2021: 717−732

[28] Li Mengyuan, Zhang Yinqian, Lin Zhiqiang. CrossLine: Breaking "Security-by-Crash" based memory isolation in AMD SEV[C] //Proc of the 28th ACM SIGSAC Conf on Computer and Communications Security (CCS'21). New York: ACM, 2021: 2937–2950

[29] Intel. Trust domain extensions (TDX)[EB/OL]. 2023[2023-03-08].https://www.intel.com/content/www/us/en/developer/articles/technical/intel-trust-domain-extensions.html

[30] Intel. TDX-TOOLS[EB/OL]. 2023[2023-04-21].https://github.com/intel/tdx-tools

[31] Intel. TDX-Module[EB/OL]. 2023[2023-04-21].https://github.com/intel/tdx-module

[32] Hua Zhichao, Gu Jinyu, Xia Yubin, et al. vTZ: Virtualizing Arm TrustZone[C]//Proc of the 26th USENIX Security Symp. Berkeley, CA: USENIX Association, 2017: 541−556

[33] Cho Y, Shin J, Kwon D, et al. Hardware-assisted on-demand hypervisor activation for efficient security critical code execution on mobile devices[C]//Proc of the 24th USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2016: 565−578

[34] Jang J, Choi C, Lee J, et al. PrivateZone: Providing a private execution environment using Arm TrustZone[J]. IEEE Transactions on Dependable and Secure Computing, 2016, 15(5): 797−810

[35] Brasser F, Gens D, Jauernig P, et al. SANCTUARY: Arm TrustZone with user-space enclaves[C/OL]//Proc of the 26th Network and Distributed System Security Symp (NDSS). Washington: ISOC, (2019-02-24)[2023-05-17].https://www.ndss-symposium.org/ndss-paper/sanctuary-arming-trustzone-with-user-space-enclaves/

[36] Sun He, Sun Kun, Wang Yuewu, et al. TrustICE: Hardware-assisted isolated computing environments on mobile devices[C]//Proc of the 45th Annual IEEE/IFIP Int Conf on Dependable Systems and Networks. Piscataway, NJ: IEEE, 2015: 367−378

[37] Li Wenhao, Xia Yubin, Lu Long, et al. TEEv: Virtualizing trusted execution environments on mobile platforms[C]//Proc of the 15th ACM SIGPLAN/SIGOPS Int Conf on Virtual Execution Environments. New York: ACM, 2019: 2−16

[38] Kwon D, Seo J, Cho Y, et al. PrOS: Light-weight privatized secure OSes in Arm TrustZone[J]. IEEE Transactions on Mobile Computing, 2019, 19(6): 1434−1447

[39] Zhao Shijun, Zhang Qianying, Qin Yu, et al. SecTEE: A software-based approach to secure enclave architecture using TEE[C]//Proc of the 26th ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2019: 1723−1740

[40] Cerdeira D, Martins J, Santos N, et al. ReZone: Disarming TrustZone with TEE privilege reduction[C]// Proc of the 31st USENIX Security Symp. Berkeley, CA: USENIX Association, 2022: 2261−2279

[41] Arm. Hafnium architecture [EB/OL]. (2019-11-14) [2023-05-17]. https://review.trustedfirmware.org/plugins/gitiles/hafnium/hafnium/+/HEAD/docs/Architecture.md

[42] Li Dingji, Mi Zeyu, Xia Yubin, et al. TwinVisor: Hardware-isolated confidential virtual machines for Arm[C] //Proc of the 28th ACM SIGOPS Symp on Operating Systems Principles. New York: ACM, 2021: 638−654

[43] ARM. Arm confidential compute architecture software stack [EB/OL]. 2022[2023-05-17].https://developer.arm.com/documentation/den0127/a/Software-components

[44] Costan V, Lebedev I, Devadas S. Sanctum: Minimal hardware extensions for strong software isolation[C] //Proc of the 25th USENIX Security Symp. Berkeley, CA: USENIX Association, 2016: 857−874

[45] Weiser S, Werner M, Brasser F, et al. TIMBER-V: Tag-isolated memory bringing fine-grained enclaves to RISC-V[C/OL] //Proc of the 26th Network and Distributed System Security Symp. Washington: ISOC, 2019[2023-05-17].https://www.ndss-symposium.org/ndss-paper/timber-v-tag-isolated-memory-bringing-fine-grained-enclaves-to-risc-v/

[46] Bahmani R, Brasser F, Dessouky G, et al. CURE: A security architecture with customizable and resilient enclaves[C] //Proc of the 30th USENIX Security Symp. Berkeley, CA: USENIX Association, 2021: 1073−1090

[47] Zhu Jianping, Hou Rui, Wang XiaoFeng, et al. Enabling rack-scale confidential computing using heterogeneous trusted execution environment[C]// Proc of the 41st IEEE Symp on Security and Privacy (SP). Piscataway, NJ: IEEE, 2020: 1450–1465

[48] NVIDIA. NVIDIA confidential computing [EB/OL]. 2022[2023-05-17].https://www.nvidia.com/en-us/data-center/solutions/confidential-computing/.

[49] Jiang Jianyu, Qi ji, Chen Xusheng, et al. CRONUS: Fault-isolated, secure and high-performance heterogeneous computing for trusted execution evironment [C] //Proc of the 55th IEEE/ACM Int Symp on Microarchitecture (MICRO). Piscataway, NJ: IEEE, 2022: 124−143

[50] Wang Xingbin, Hou Rui, Zhu Yifan, et al. NPUFort: A secure architecture of DNN accelerator against model inversion attack [C] // Proc of the 16th ACM Int Conf on Computing Frontiers. New York: ACM, 2019: 190−196

[51] Lee S, Kim J, Na S, et al. TNPU: Supporting trusted execution with tree-less integrity protection for neural processing unit [C] //Proc of the 28th IEEE Int Symp on High-Performance Computer Architecture (HPCA). Piscataway, NJ: IEEE, 2022: 229−243

[52] Kinney L. Trusted Platform Module Basics: Using TPM in Embedded Systems[M]. Washington: Newnes, 2006

[53] Proudler G, Chen Liqun, Dalton C. Trusted Computing Platforms: TPM2.0 in Context [M]. Berlin: Springer, 2014

[54] Kauer B. OSLO: Improving the security of trusted computing[C] // Proc of the 16th USENIX Security Symp. Berkeley, CA: USENIX Association, 2007: 173−191

[55] AMD. SEV-SNP, Strengthening VM isolation with integrity protection and more [EB/OL]. 2020[2023-05-17].https://www.

amd.com/system/files/TechDocs/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf

[56] Cui Jinhua, Yu Z, Shinde S, et al. SmashEx: Smashing SGX enclaves using exceptions[C] //Proc of the 28th ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2021: 779−793

[57] Weichbrodt N, Kurmus A, Pietzuch P, et al. AsyncShock: Exploiting synchronisation bugs in Intel SGX enclaves[C] //Proc of the 21st European Symp on Research in Computer Security (ESORICS). Berlin: Springer, 2016: 440−457

[58] Ermolov M, Goryachy M. How to hack a turned-off computer, or running unsigned code in Intel management engine[R/OL]. London: Black Hat Europe, 2017 [2023-04-20].https://www.blackhat.com/docs/eu-17/materials/eu-17-Goryachy-How-To-Hack-A-Turned-Off-Computer-Or-Running-Unsigned-Code-In-Intel-Management-Engine-wp.pdf

[59] Ragab H, Milburn A, Razavi K, et al. CROSSTALK: Speculative data leaks across cores are real[C] //Proc of the 42nd IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2021: 1852−1867

[60] Van Bulck J, Piessens F, Strackx R. SGX-Step: A practical attack framework for precise enclave execution control[C/OL] //Proc of the 2nd Workshop on System Software for Trusted Execution. New York: ACM, 2017[2023-05-17].https://dl.acm.org/doi/10.1145/3152701.3152706

[61] Yarom Y, Falkner K. FLUSH+RELOAD: A high resolution, low noise, L3 cache side-channel attack[C] //Proc of the 23rd USENIX Security Symp. Berkeley, CA: USENIX Association, 2014: 719–732

[62] Osvik D, Shamir A, Tromer E. Cache attacks and countermeasures: The case of AES[G/OL]//LNCS 3960: Proc of the 6th Cryptographers' Track at the RSA Conf on Topics in Cryptolog. Berlin: Springer, 2006[2023-05-17].https://dl.acm.org/doi/10.1007/11605805_1

[63] Tromer E, Osvik E, Shamir A. Efficient cache attacks on AES, and countermeasures[J]. IACR Journal of Cryptology, 2010, 23(1): 37−71

[64] Van Schaik S, Minkin M, Kwong A, et al. CacheOut: Leaking data on Intel CPUs via cache evictions[C] // Proc of the 42nd IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2021: 339−354

[65] Moghimi A, Irazoqui G, Eisenbarth T. CacheZoom: How SGX amplifies the power of cache attacks[C] // Proc of the 19th Int Conf of Cryptographic Hardware and Embedded Systems. Berlin: Springer, 2017: 69−90

[66] Dall F, De Micheli G, Eisenbarth T, et al. CacheQuote: Efficiently recovering long-term secrets of SGX EPID via cache attacks[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018(2): 171−191

[67] Li Mengyuan, Zhang Yinqian, Wang Huibo, et al. TLB poisoning attacks on AMD secure encrypted virtualization[C] //Proc of the 37th Annual Computer Security Applications Conf. New York: ACM, 2021: 609−619

[68] Kocher P, Horn J, Fogh A, et al. Spectre attacks: Exploiting speculative execution[C] //Proc of the 40th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2019: 1–19

[69] Chen Guoxing, Chen Sanchuan, Xiao Yuan, et al. SgxPectre: Stealing Intel secrets from SGX enclaves via speculative execution[C] //Proc of the 4th IEEE European Symp on Security and Privacy. Piscataway, NJ: IEEE, 2019: 142−157

[70] Koruyeh E M, Khasawneh K, Song Chengyu, et al. Spectre returns! Speculation attacks using the return stack buffer[C/OL] //Proc of the 12th USENIX Workshop on Offensive Technologies. Berkeley, CA: USENIX Association, 2018[2023-05-17].https://dl.acm.org/doi/abs/10.5555/3307423.3307426

[71] Lipp M, Schwarz M, Gruss D, et al. Meltdown: Reading kernel memory from user space[C] //Proc of the 27th USENIX Security Symp. Berkeley, CA: USENIX Association, 2018: 973−990

[72] Van Bulck J, Minkin M, Weisse O, et al. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution[C] //Proc of the 27th USENIX Security Symp. Berkeley, CA: USENIX Association, 2018: 991−1008

[73] Huo Tianlin, Meng Xiaoni, Wang Wenhao, et al. Bluethunder: A 2-level directional predictor-based side-channel attack against SGX[J]. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(1): 321−347

[74] Morbitzer M, Huber M, Horsch J, et al. SEVered: Subverting AMD's virtual machine encryption[C/OL] //Proc of the 11th European Workshop on Systems Security. New York: ACM, 2018[2023-05-17].https://doi.org/10.1145/3193111.3193112

[75] Murdock K, Oswald D, Garcia F D, et al. Plundervolt: Software-based fault injection attacks against Intel SGX[C] //Proc of the 41st IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2020: 1466−1482

[76] Chen Zitai, Vasilakis G, Murdock K, et al. VoltPillager: Hardware-based fault injection attacks against Intel SGX enclaves using the SVID voltage scaling interface[C] //Proc of the 30th USENIX Security Symp. Berkeley, CA: USENIX Association, 2021: 699−716

[77] Lu M. TrustZone, TEE and trusted video path implementation considerations [EB/OL]. 2018[2023-04-23].https://www.arm.com/files/event/Developer_Track_6_TrustZone_TEEs_and_Trusted_Video_Path_implemention_considerations.pdf

[78] Trustonic. Trustonic application protection delivers comprehensive security for mobile financial aervices [EB/OL]. 2018[2023-04-23]. https: //www. trustonic. com/markets/financial-services/

[79] Beniamini G. QSEE privilege escalation vulnerability and exploit [EB/OL]. 2016[2024-04-23]. http://bits-please.blogspot.com/2016/05/qsee-privilege-escalation-vulnerability.html

[80] Linaro. ClearKey OP-TEE AOSP DRM plugin[EB/OL]. 2019 [2023-04-23].https://gith ub.com/linaro-mmwg/clearkeydrmplugin

[81] ARM. ARM CoreLink TZC-400 TrustZone address space controller technical reference manual, ID063014 [R/OL]. 2021[2023-04-23].https://developer.arm.com/documentation/ddi0504/latest/

[82] ARM. Arm CCA software architecture [EB/OL]. 2022[2023-04-23].https://developer.arm.com/documentation/den0125/0200/Arm-

CCA-Software-Architecture

[83] ARM. Arm realm management extension (RME)[EB/OL]. 2021[2023-04-23].https://developer.arm.com/documentation/den0129/aa

[84] Zhang Yingjun, Feng Dengguo, Qin Yu, et al. A TrustZone based application protection scheme in highly open scenarios[J]. Journal of Computer Research and Development, 2017, 54(10): 2268−2283 (in Chinese)
(张英骏, 冯登国, 秦宇 等. 基于TrustZone的开放环境中敏感应用防护方案[J]. 计算机研究与发展, 2017, 54(10): 2268−2283)

[85] Beniamini G. TrustZone kernel privilege escalation (CVE-2016-431)[EB/OL]. 2016[2023-04-23]. http://bits-please.blogspot.com/2016/06/trustzone-kernel-privilege-escalation.html

[86] Beniamini G. War of the worlds-hijacking the Linux kernel from QSEE[EB/OL]. 2016[2023-04-23].https://bits-please.blogspot.com/2016/05/war-of-worlds-hijacking-linux-kernel.html

[87] Komaromy D. Unbox your phone [EB/OL]. 2018[2023-04-23].https://medium.com/taszksec/unbox-your-phone-part-i-331bbf44c30c

[88] Gal Beniamini. Trust issues: Exploiting TrustZone TEEs[EB/OL]. 2019[2023-04-23].https://googleprojectzero.blogspot.com/2017/07/trust-issues-exploiting-trustzone-tees.html

[89] Di Shen. Attacking your trusted core exploiting TrustZone on Android [EB/OL]. BlackHat USA. 2019[2023-04-23].https://www.blackhat.com/docs/us-15/materials/us-15-Shen-Attacking-Yor-Trusted-Core-Exploiting-Trustzone-On-Android.pdf

[90] Zhang Ning, Sun He, Sun Kun, et al. CacheKit: Evading memory introspection using cache incoherence [C] // Proc of the 1st IEEE European Symp on Security and Privacy. Piscataway, NJ: IEEE, 2016: 337−352

[91] Guanciale R, Nemati H, Baumann C, et al. Cache storage channels: Alias-driven attacks and verified countermeasures [C] // Proc of the 1st IEEE European Symp on Security and Privacy. Piscataway, NJ: IEEE, 2016: 38−55

[92] Lipp M, Gruss D, Spreitzer R, et al. ARMageddon: Cache attacks on mobile devices [C] // Proc of the 25th USENIX Conf on Security Symp. Berkeley, CA: USENIX Association, 2016: 549−564

[93] Zhang Ning, Sun Kun, Shands D, et al. TruSpy: Cache side-channel information leakage from the secure world on ARM devices [J/OL]. IACR Cryptology ePrint Archive, 2016[2023-04-24].https://eprint.iacr.org/2016/980

[94] Cho H, Zhang Penghui, Kim D, et al. Prime+Count: Novel cross-world covert channels on ARM TrustZone [C] // Proc of the 34th Annual Computer Security Applications Conf (ACSAC'18). New York: ACM, 2018: 441–452

[95] Jacob N, Heyszl J, Zankl A, et al. How to break secure boot on FPGA SoCs through malicious hardware [C] // Proc of the 19th Int Conf on Cryptographic Hardware and Embedded Systems (CHES). Berlin: Springer, 2017: 425–442

[96] Benhani E, Marchand C, Aubert, A, et al. On the security evaluation of the ARM TrustZone extension in a heterogeneous SoC [C] // Proc of the 30th IEEE Int System-on-Chip Conf (SOCC). Piscataway, NJ: IEEE, 2017: 108−113

[97] Wang Jie, Sun Kun, Lei Lingguang, et al. Cache-in-the-middle (CITM) attacks: Manipulating sensitive data in isolated execution environments [C] // Proc of the 27th ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2020: 1001−1015

[98] Zhang Yingjun, Feng Dengguo, Qin Yu, et al. TrustZone-based trusted code execution scheme under strong security requirements [J]. Journal of Computer Research and Development, 2015, 52(10): 2224−2238(in Chinese)
(张英骏, 冯登国, 秦宇, 等. 基于Trustzone的强安全需求环境下可信代码执行方案 [J]. 计算机研究与发展, 2015, 52(10): 2224−2238)

[99] Zheng Xianyi, Li Wen, Meng Dan. Analysis and research of TrustZone technology[J]. Chinese Journal of Computers, 2016, 39(9): 1912−1928 (in Chinese)
(郑显义, 李文, 孟丹. TrustZone技术的分析与研究[J]. 计算机学报, 2016, 39(9): 1912−1928)

[100] Zheng Xianyi, Shi Gang, Meng Dan. Summary of research on system security isolation technology[J]. Chinese Journal of Computers, 2017, 40(5): 1057−1079 (in Chinese)
(郑显义, 史岗, 孟丹. 系统安全隔离技术研究综述[J]. 计算机学报, 2017, 40(5): 1057−1079)

[101] Birkholz H. Veraison [EB/OL]. 2023[2023-04-24].https://github.com/veraison

[102] Evenchick E. RustZone: Writing trusted applications in Rust [EB/OL]. 2019[2023-04-24].https://i.blackhat.com/eu-18/Thu-Dec-6/eu-18-EvenchickRustZone.pdf

[103] Shinde S, Wang Shengyi, Yuan Pinghai, et al. BesFS: A POSIX filesystem for enclaves with a mechanized safety proof [C] // Proc of the 29th USENIX Security Symp (USENIX Security' 20). Berkeley, CA: USENIX Association, 2020: 523−540

[104] RISC-V. RISC-V [EB/OL]. 2023[2023-04-24].https://riscv.org/

[105] RISC-V. RISC-V specification [EB/OL]. 2023[2023-04-24]. https://riscv.org/technical/specifications/

[106] Nasahl P, Schilling R, Werner M, et al. HECTOR-V: A heterogeneous CPU architecture for a secure RISC-V execution environment [C] // Proc of the 16th ACM Asia Conf on Computer and Communications Security. New York: ACM, 2021: 187−199

[107] Arnautov S, Trach B, Gregor F, et al. SCONE: Secure Linux containers with Intel SGX [C] // Proc of the 12th USENIX Symp on Operating Systems Design and Implementation (OSDI'16). Berkeley, CA: USENIX Association, 2016: 689−703

[108] Bulck V J, Oswald D, Marin E, et al. A tale of two worlds: Assessing the vulnerability of enclave shielding runtimes [C] // Proc of the 26th ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2019: 1741–1758

[109] Le A, Dao B, Suzaki K, et al. Experiment on replication of side channel attack via cache of RISC-V Berkeley out-of-order machine (BOOM) implemented on FPGA [C/OL] // Proc of the 4th Workshop on Computer Architecture Research with RISC-V

(CARRV 2020). Piscataway, NJ: IEEE, 2020[2023-04-24]. https://carrv.github.io/2020/papers/CARRV2020_paper_2_Le.pdf

[110] RISC-V. RISC-V trusted execution environment [EB/OL]. 2023[2023-04-24].https://lists.riscv.org/g/tech-tee

[111] RISC-V. ZAYA TEE [EB/OL]. 2021[2023-04-24].https://riscv.org/blog/2021/10/zaya-now-offers-trusted-execution-environment-for-risc-v-zaya/

[112] Hex-five. MultiZone [EB/OL]. 2023[2023-04-24].https://hex-five.com/multizone-security-tee-riscv/

[113] FOSDEM. Trusted RV [EB/OL]. 2021[2023-04-24].https://archive.fosdem.org/2021/schedule/event/tee_trusted_rv/

[114] Zhang Jie, Jung M. ZnG: Architecting GPU multi-processors with new flash for scalable data analysis [C] // Proc of the 47th ACM/IEEE Annual Int Symp on Computer Architecture (ISCA). Piscataway, NJ: IEEE, 2020: 1064−1075

[115] Li Peilong, Luo Yan, Zhang Ning, et al. Heterospark: A heterogeneous CPU/GPU spark platform for machine learning algorithms [C] // Proc of the 10th IEEE Int Conf on Networking, Architecture and Storage (NAS). Piscataway, NJ: IEEE, 2015: 347−348

[116] Cao Qingqing, Balasubramanian N, Balasubramanian A. MobiRNN: Efficient recurrent neural network execution on mobile GPU[C/OL] // Proc of the 1st Int Workshop on Deep Learning for Mobile Systems and Applications. New York: ACM, 2017[2023-05-17].https://doi.org/10.1145/3089801.3089804

[117] Jang I, Tang A, Kim T, et al. Heterogeneous isolated execution for commodity GPUs[C] //Proc of the 24th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2019: 455–468

[118] Che Shuai, Boyer M, Meng Jiayuan, et al. Rodinia: A benchmark suite for heterogeneous computing [C] // Proc of the 10th IEEE Int Symp on Workload Characterization (IISWC). Piscataway, NJ: IEEE, 2009: 44–54

[119] Wang Chenxu, Zhang Fengwei , Deng Yunjie, et al. CAGE: Complementing Arm CCA with GPU extensions[C/OL]// Proc of the 31st Network and Distributed System Security Symp. Washington, DC: ISOC, 2023[2024-02-26]. https://dx.doi.org/10.14722/ndss.2024.24763

[120] CVE. CVE-2020-5991 [EB/OL]. 2020[2023-04-24].https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-5991

[121] CVE. CVE-2021-1121 [EB/OL]. 2021[2023-04-24].https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-1121

[122] CVE. CVE-2022-21815 [EB/OL]. 2022[2023-04-24].https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-21815

Zhang Fengwei, born in 1986. PhD, professor, PhD supervisor. Senior member of CCF. His main research interests include system security, hardware assisted security, trusted execution environment, transparent tracing, and debugging technology.

张锋巍, 1986 年生. 博士，研究员，博士生导师. CCF 高级会员. 主要研究方向为系统安全、硬件辅助安全、可信执行环境、透明跟踪、调试技术.

Zhou Lei, born in 1988. PhD. His main research interests include x86-based system security and trusted computing technology.

周　雷, 1988 年生. 博士. 主要研究方向为 x86 架构系统安全、可信计算技术.

Zhang Yiming, born in 1994. PhD candidate. His main research interests include Arm architecture confidential computing and hardware assisted security system.

张一鸣, 1994 年生. 博士研究生. 主要研究方向为基于 Arm 架构机密计算和硬件辅助的安全系统.

Ren Mingde, born in 1999. PhD candidate. His main research interests include RISC-V architecture system security and confidential computing technology.

任明德, 1999 年生. 博士研究生. 主要研究方向为 RISC-V 架构系统安全、机密计算技术.

Deng Yunjie, born in 1998. Master. His main research interests include Arm architecture system security and confidential computing technology.

邓韵杰, 1998 年生. 硕士. 主要研究方向为 Arm 架构系统安全、机密计算技术.