计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(11): 2822−2842, 2026

Arm 机密计算架构研究进展与展望

张一鸣[1,2] 张锋巍[2] 闫守孟[3]
[1](广东技术师范大学数据科学与工程学院 广东河源 517583)
[2](南方科技大学计算机科学与工程系 广东深圳 518055)
[3](蚂蚁集团 北京 100081)
(yimingz@gpnu.edu.cn)

Arm Confidential Compute Architecture: Research Progress and Future Directions

Zhang Yiming[1,2], Zhang Fengwei[2], and Yan Shoumeng[3]
[1](*School of Data Science and Engineering*, *Guangdong Polytechnic Normal University*, *Heyuan*, *Guangdong* 517583)
[2](*Department of Computer Science and Engineering*, *Southern University of Science and Technology*, *Shenzhen*, *Guangdong* 518055)
[3](*Ant Group*, *Beijing* 100081)

Abstract With the rapid development of cloud computing, big data, and artificial intelligence, data security and privacy protection have become pressing challenges. In 2021, Arm proposed confidential compute architecture (CCA), which introduces hardware-level isolation mechanisms to provide a trusted execution environment for sensitive computing tasks, ensuring the confidentiality and integrity of both data and code during execution. Although initial progress has been made in CCA-based studies on user-level confidential execution, device isolation, and system-level security enhancement, the field as a whole remains in its early stages. Current research remains fragmented, with limited systematic analysis of architectural development and underlying technical frameworks. Moreover, some studies repeatedly reimplement fundamental mechanisms without generalizing common design, while others focus primarily on functional implementation with limited analysis of potential attack surfaces and long-term risks. To address these gaps, we systematically analyze the key mechanisms of CCA in confidential computing and review representative research advances in recent years. Specifically, we classify design strategies and implementation approaches in areas such as execution environment extension, device isolation, application adaptation, and security enhancement. Building on typical system case studies, we further summarize recurring technical patterns in CCA research, including root world isolation, multi-granule memory protection tables, and the coordination of stage-2 page tables with the system memory management unit. Finally, we discuss the major challenges and emerging trends faced by CCA, providing insights to guide future research and innovation in Arm-based confidential computing.

Key words Arm confidential compute architecture; trusted execution environment; xPU confidential computing; hardware-assisted security; system security

摘 要 随着云计算、大数据和人工智能的快速发展,数据安全与隐私保护已成为亟待解决的核心问题。2021 年 Arm 提出机密计算架构(confidential compute architecture, CCA),通过硬件级隔离机制为敏感计算

收稿日期: 2025-12-05; **修回日期**: 2026-06-09
基金项目: 国家自然科学基金项目(U2541211, 62372218, U24A6009);广东省普通高校青年创新人才项目(2025KQNCX039);广东技术师范大学校级科研项目人才专项项目(2026SDKYA080)
This work was supported by the National Natural Science Foundation of China (U2541211, 62372218, U24A6009), the Guangdong University Young Innovative Talents Program Project (2025KQNCX039), and the Talent Special Research Project of Guangdong Polytechnic Normal University (2026SDKYA080).
通信作者: 张锋巍(zhangfw@sustech.edu.cn)

任务提供可信执行环境，保障数据与代码在使用过程中的安全性。尽管在用户态机密执行、设备隔离与系统安全增强等基于 CCA 的研究中已有初步成果，但整体来看，该领域仍处于起步阶段，现有研究较为分散，缺乏对架构发展与技术体系的系统性梳理。同时，部分研究重复构建底层关键机制，却未能总结技术共性。另一些工作则更关注功能实现，而对潜在攻击面及未来风险的分析不足。针对上述问题，对 CCA 在机密计算中的关键技术机制与研究进展进行系统分析，并对近年来具有代表性的成果进行分类归纳，涵盖执行环境扩展、设备隔离、应用适配与安全增强等方面的设计思路与实现路径。在此基础上，结合典型系统案例，凝练 CCA 研究中常用的技术路径，包含根世界隔离、多颗粒表内存隔离机制、2 阶段页表与系统内存管理单元协同等关键技术。最后，进一步探讨当前 CCA 面临的主要挑战与研究趋势，为 Arm 机密计算领域的研究与创新提供参考。

关键词 Arm 机密计算架构；可信执行环境；xPU 机密计算；硬件辅助安全；系统安全

中图法分类号 TP309

DOI: 10.7544/issn1000-1239.202550886 **CSTR:** 32373.14.issn1000-1239.202550886

在数字化浪潮的推动下，数据正日益成为关键的生产要素与价值载体[1]。然而，数据安全问题也随之加剧，尤其在传统计算架构中，操作系统与虚拟机管理程序等特权软件对应用程序的代码与数据拥有完全控制权。一旦这些软件存在漏洞或遭受攻击，极有可能导致敏感信息泄露或关键算法外泄，严重威胁系统安全。

机密计算（confidential computing）[2]作为一种新兴的系统安全技术，通过在硬件层面构建可信执行环境（trusted execution environment, TEE）[3]，实现对运行中数据和代码的隔离与保护[4-6]，已成为当前安全研究的热点方向之一。在众多 TEE 实现路径中，Arm 公司于 2021 年推出的机密计算架构（confidential compute architecture, CCA）[7]，为 Arm 架构生态提供了新的安全计算框架，标志着其在云计算等数据安全与隐私保护场景中的关键布局。CCA 的核心理念是引入领域世界（realm world）与根世界（root world）两个新型隔离环境，扩展传统 Arm 架构中安全世界（secure world）与非安全世界（normal world）之间的设计。领域世界承载机密虚拟机计算任务，并受到由根世界控制的硬件隔离保护，即使特权系统软件被攻破，攻击者也无法访问其内部资源。这一设计可提升系统整体安全性，为构建云端与终端的机密计算应用场景提供了隔离保障。

近年来，围绕 CCA 的研究逐渐增加，涵盖机密计算环境扩展、外设安全与安全应用等多个关键方向，显著丰富了其生态体系与应用边界。例如，Shelter[8]在 CCA 架构中扩展用户态的机密计算环境，RContainer[9]则进一步拓展了容器级别的隔离能力。CAGE[10]，ACAI[11]，PORTAL[12]等研究探索了在 GPU，PCIe 等设备侧实现机密访问路径，解决异构机密计算环境的隔离挑战。SCRUTINIZER[13]，HitchHiker[14]等工作基于 CCA 硬件特性提供安全自省和日志保护等能力，展示了 CCA 在实际安全应用方面的潜力。

尽管国内外研究已在用户态机密执行、设备可信访问和系统安全增强等方面取得初步成果，但从整体来看，CCA 研究仍处于早期阶段，现有成果比较分散，缺乏系统性的分析对其发展脉络、技术体系和架构演化进行梳理，导致对 CCA 研究全貌尚不清晰。许多工作构建相似的底层关键技术（如多颗粒表保护机制、根世界隔离等），但尚未总结其在设计与应用中的通用原理，限制了对 CCA 系统共性机制的深入了解和创新。此外，现有研究大多聚焦于功能实现，较少探讨 CCA 威胁模型和潜在攻击面等问题，缺乏对未来研究风险的系统预警与启示。目前虽有文献对 TEE 进行了综述研究[15]，但针对 CCA 及其相关工作的系统化分析尚属空白。

基于上述研究现状，本文旨在系统梳理 CCA 的技术架构与研究进展，分析当前研究已解决的问题，凝练其共性技术机制，并展望未来工作在安全保障方面的能力提升。具体而言，本文的主要工作包括：1）从架构机制角度出发，研究 CCA 的核心设计理念与执行模型，并分别与其他代表性机密计算架构（如 AMD SEV-SNP[16]，Intel TDX[17]）以及传统的 Arm TrustZone 架构[18]在关键技术上进行对比；2）对近年来具有代表性的 CCA 研究成果进行分析，涵盖其在执行环境扩展、设备隔离、应用适配与安全增强等方面的设计策略与实现路径；3）提炼典型系统中反复出现的共性机制，总结当前 CCA 生态中常见的技术架构设计，并分析 CCA 的潜在攻击面和威胁来源；4）结合

已有研究,展望 CCA 面临的挑战与未来需求,揭示当前存在的技术瓶颈和发展方向。

1 Arm CCA 背景介绍

机密计算依赖底层处理器提供的硬件隔离能力,因此对处理器架构提出了支持 TEE 的要求。具体而言,机密计算依赖内存加密、页表隔离等硬件模块提供物理与虚拟内存保护,并结合安全启动链、硬件信任根、设备隔离以及可信设备访问等机制,共同构建 TEE。在软件层面,需要 TEE 操作系统、机密虚拟机管理器、安全镜像构建工具、安全固件以及支持远程证明的服务,以保证上层工作负载在可信边界内安全运行。整体技术栈从硬件、固件、虚拟化层到操作系统与服务端基础设施协同构成机密计算的实现。因此,根据机密计算的上述要求,本节系统介绍 CCA 的关键技术机制,内容分为 CCA 的架构设计与核心组件、支持机密计算的内存隔离与设备访问机制,以及安全启动与远程证明。最后与 SEV-SNP, TDX, TrustZone 进行关键技术的对比。

1.1 CCA 机制与核心组件

Arm 在 2021 年推出 CCA 并构建于 Armv9.2-A 架构的领域管理扩展(realm management extension, RME)[19] 硬件特性之上,引入了新的执行环境与隔离机制,为机密计算提供更精细的安全边界。如图 1 所示,CCA 将系统划分为 4 个执行环境,分别为非安全世界、安全世界、领域世界与根世界。其中:

1)非安全世界运行传统操作系统和应用程序;

2)安全世界延续 Arm TrustZone 模型,处理设备厂商的安全服务;

3)领域世界是 CCA 专为第三方用户机密计算设计的 TEE;

4)根世界具有最高特权,与其他世界隔离,用于系统初始化、切换控制及安全管理。

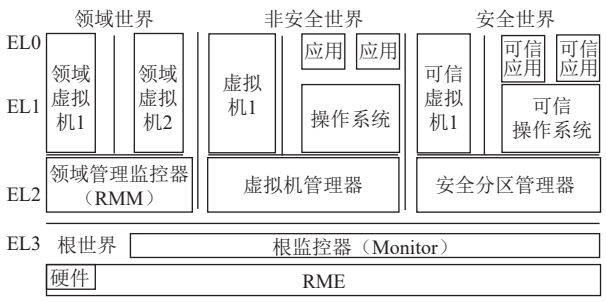


Fig. 1 Arm CCA architecture

图 1 Arm CCA 架构

在 Arm 架构的特权模式方面,CCA 沿用了传统的 4 级异常(exception level, EL)模型[20],将处理器划分为 EL0~EL3 这 4 个级别,但重新定义了其用途:

1)EL3 专属根世界,运行根监控器(Monitor)[21]。根世界作为系统中有最高权限的执行环境,负责管理其他世界间的切换和访问控制。

2)领域世界运行在 EL0~EL2。EL0 和 EL1 运行领域虚拟机,从而使第三方用户程序无需修改即可运行在机密虚拟机环境中,保证了系统兼容性。EL2 则运行领域管理监控器(realm management monitor, RMM)[22] 用于领域虚拟机的生命周期管理、上下文切换控制以及领域内存与执行状态的安全管理。

3)非安全世界同样使用 EL0~EL2 执行传统虚拟化栈。

4)安全世界继续运行 TrustZone 相关组件。

RME 是支撑 CCA 的核心硬件机制,通过一组硬件特性提供了页粒度级别的内存隔离能力。其核心是施加颗粒保护检查(granule protection check, GPC)[23] 机制,每次内存访问都需通过颗粒保护表(granule protection table, GPT)[24] 的验证,以确定访问是否合法。该机制支持内存颗粒(最小粒度为 4 KB)在多个世界属性之间的动态迁移,即配置内存颗粒所属于的世界环境(如是否属于领域世界),从而确保非法访问被硬件强制隔离。在 CCA 架构中,处理器运行被划分为 normal, secure, realm, root 这 4 种安全状态,不同安全状态的处理器只能访问具有相应属性的物理地址空间。颗粒保护表可配置的物理地址空间属性与各安全状态的访问权限如表 1 所示。

Table 1 Access Permissions to Physical Address Space in Different Security States of CCA[19]

表 1 CCA 不同安全状态的物理地址空间访问权限[19]

物理地址空间	normal	secure	realm	root
非安全世界	√	√	√	√
安全世界	×	√	×	√
领域世界	×	×	√	√
根世界	×	×	×	√

RMM 是 CCA 构建领域虚拟机的关键软件组件,运行于 EL2,采用解耦式虚拟化架构,为领域虚拟机提供安全服务,包括元数据管理和内存映射等核心功能。而非安全世界的虚拟机管理器(hypervisor)负责领域虚拟机的调度与内存资源分配,通过领域管理接口(realm management interfaces, RMIs)[25] 为虚拟机运行提供功能。虽然虚拟机管理器可以管理领域

虚拟机，但无法访问其内部内容。RMM 通过 2 级地址转换机制（stage-2 页表）确保各领域虚拟机之间的隔离，从而精确控制每个领域虚拟机的内存视图。

根监控器是运行于根世界的核心安全组件，具有最高特权且其他世界的程序都无法访问根世界内存。根监控器负责系统启动、世界切换、转发 RMI 及管理颗粒保护表，是实现 CCA 运行时访问控制的关键。

1.2 运行时访问控制机制

CCA 的运行时访问控制机制是其安全性的核心保障。如图 2 所示，其通过 2 阶段地址转换与颗粒保护检查相结合实现领域虚拟机的内存隔离。第 1 级页表由操作系统管理，实现虚拟地址到中间物理地址的映射；第 2 级页表由 RMM 管理，映射至最终物理地址；最后由颗粒保护检查对该物理地址是否允许当前执行环境访问进行验证。其中颗粒保护检查作为 RME 的一部分，扩展了内存管理单元（MMU），以阻止非法访问。在将虚拟地址转换为物理地址时，MMU 中的颗粒保护检查会基于颗粒保护表检查当前安全状态是否有权访问目标物理地址空间。如果在颗粒保护检查的过程中产生非法访问，则会触发颗粒保护检查错误（granule protection check fault, GPF）[26]。

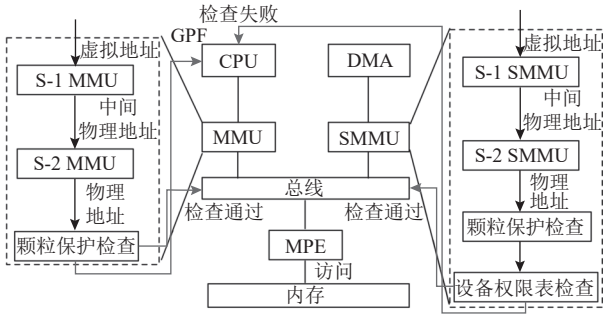


Fig. 2 Isolation mechanism in the CCA

图 2 CCA 中的隔离机制

颗粒保护表是只能由根世界访问的内存结构。如图 3 所示，颗粒保护表采用 2 级表的查找机制，不同层级的表项具有不同的描述符。在颗粒保护表中，颗粒保护信息（granule protection information, GPI）[27]用于标识内存颗粒所属的世界属性。颗粒保护检查通过确认当前处理器安全状态是否具备与颗粒保护信息匹配的访问权限，从而阻止非法访问。这种检查对软件完全透明，确保了高效性和安全性。

此外，CCA 引入动态 TrustZone 技术[28]，通过颗粒保护表实现内存的世界属性动态迁移，解决了传统 TrustZone 对安全世界内存区域的数量限制。例如，可通过根世界配置颗粒保护表将非安全世界的内存

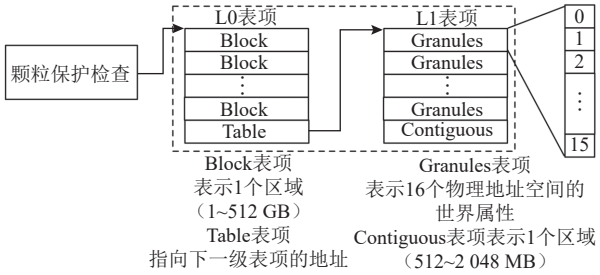


Fig. 3 GPT structure of CCA

图 3 CCA 颗粒保护表结构

转至安全世界，从而提升内存资源利用率与配置灵活性。

1.3 可信设备访问机制

在机密计算环境中实现与系统外设（如 FPGA, GPU, PCIe 外设和智能网卡等加速器）之间的可信访问是 CCA 提供的重要安全能力之一。当前主流的 Arm GPU 与 CPU 采用统一内存架构[29]，使这些 GPU 能够直接使用主机内存作为显存，支持通过 DMA 实现高效数据传输，从而提高主机系统与 GPU 之间数据传输和通信的效率。

然而，设备访问绕过了 CPU 的 MMU，因此需要由系统内存管理单元（SMMU）提供访问控制。SMMU 与 CPU 的 MMU 类似，支持 1 级与 2 级地址转换，并由高特权软件通过 MMIO 方式配置转换策略。目前，包括 Arm Mali GPU[30], Qualcomm Adreno GPU[31], NVIDIA Tegra X1[32] 等主流设备均通过物理接口连接 SMMU，实现对系统内存的访问权限控制。CCA 在传统 SMMU 机制基础上，进一步引入 RME 设备分配（RME device assignment, RME-DA）机制[33]，可在领域虚拟机直接管理设备。如图 2 右边部分所示，CCA 通过增强 SMMU 的访问控制功能，引入和 CPU 端相同的颗粒保护检查，确保设备访问遵循颗粒保护表中的隔离规则。RME-DA 在 SMMU 中扩展了一种称为设备权限表（device permission table, DPT）[34] 的结构。DPT 记录了领域虚拟机与设备的绑定关系。当直接内存访问（DMA）设备发起内存访问时，SMMU 可通过查询 DPT 来验证目标地址是否位于关联的领域虚拟机内存范围内。SMMU 中新增一组颗粒保护检查相关寄存器[35] 并且仅可由根世界访问，用于配置颗粒保护表地址、检查策略、错误处理逻辑与快表（TLB）清理等关键信息，从而实现设备访问与机密计算对象的绑定。

为进一步强化内存访问保护，CCA 还引入内存保护引擎（memory protection engine, MPE）[36]，允许每个领域虚拟机和 RMM 使用独立的加密密钥或参数，

从而在内存隔离基础上引入内存加密保护,增强系统抵御物理攻击的防御能力。

在中断管理方面,CCA 为防止中断泄露领域虚拟机切换时的上下文状态,引入了安全中断转发机制[37]:所有发送至领域虚拟机的处理器核心的物理中断处理必须先进入 RMM,由 RMM 保存领域虚拟机上下文后,再切换至非安全世界的虚拟机管理器程序进行中断虚拟化。虚拟机管理器程序不可直接访问虚拟中断控制器的系统寄存器,而是需通过调用 RMM 接口传递配置信息。RMM 会依据中断控制器规范验证配置信息的有效性,并在安全上下文中进行设置。

1.4 信任根与远程证明

CCA 提供完整的信任链构建能力,其安全性由信任根(root of trust, RoT)与远程证明(remote attestation)共同支撑。信任根是平台中必须被信任的最小软硬件组件集合。在 CCA 平台中,信任根通常由隔离可信子系统实现,用于提供初始度量、平台生命周期管理、身份认证以及远程证明等核心安全服务。Arm 把这些组件称为硬件施加安全(hardware enforced security, HES)[34]。通过 HES, CCA 将信任根功能从处理器中分离,并交由隔离可信子系统负责执行,从而构建平台安全基础。

在安全启动阶段,HES 负责收集并组织 CCA 平台的关键度量信息,形成平台声明(platform claims)。具体来说,系统从安全启动器[38]出发,逐级验证 Monitor、RMM 等关键组件的完整性与可信性,确保启动路径未被篡改。在此过程中,各阶段组件的度量值(如固件镜像哈希)会被逐级记录,并安全传递至 HES 进行存储,从而形成平台的可信启动状态[39-40]。HES 可持续度量平台信息及平台生命周期状态,为后续远程证明请求提供服务。

在运行阶段,RMM 负责领域虚拟机的运行状态度量,并响应领域虚拟机发起的远程证明请求。CCA 的远程证明由平台证明(platform attestation)和领域证明(realm attestation)两个层级组成。当领域虚拟机请求远程证明时,RMM 会向 HES 请求获取平台证明报告。HES 在运行时维护平台的度量信息,根据 RMM 的请求收集平台状态,并使用 CCA 平台证明密钥(CCA platform attestation key, CPAK)对平台状态进行签名,生成平台证明报告。CPAK 的派生方式由具体平台实现决定。RMM 负责生成针对具体领域虚拟机的领域证明报告。在初始化阶段,RMM 从 HES 获取领域证明密钥对(realm attestation key, RAK),其中私钥用于签名领域虚拟机的运行状态,而公钥哈希会被纳入平台证明中,从而建立平台证明与领域证明之间的信任绑定。当领域虚拟机请求远程证明时,RMM 会收集该领域虚拟机的初始内存镜像度量、虚拟 CPU 初始状态以及运行时动态度量等内容,并使用 RAK 私钥生成领域证明报告。最后,RMM 将平台证明报告和领域证明报告发送给远程的可信验证者进行安全检查。

1.5 对比其他代表性机密计算技术路线

本节将 CCA 与部分代表性的机密计算技术路线进行对比。当前云计算平台的主流机密计算架构包括 SEV-SNP 和 TDX,而 TrustZone 已在移动设备和嵌入式系统中得到广泛应用。基于这些现状,本节首先讨论 CCA 与 SEV-SNP, TDX 在机密虚拟机隔离技术的差异,随后讨论 CCA 与 TrustZone 这 2 代 Arm 安全架构的区别。表 2 总结了代表性机密计算架构在关键技术的对比情况。

1.5.1 CCA 与 x86 机密计算架构

SEV-SNP 和 TDX 是当前 x86 平台上具有代表性的机密计算架构,和 CCA 的设计目标相似,都以机密虚拟机作为基本隔离单位,并在不可信的操作系统和虚拟机管理器环境下,为虚拟机提供机密性与完整性保护,使敏感工作负载能够在隔离环境中安全运行。

在安全隔离技术上,SEV-SNP 是 AMD 在 SEV 基础上的增强版本。SEV 通过为每个虚拟机分配独立的内存加密密钥来保护虚拟机内存的机密性,而 SNP 在此基础上引入反向映射表(reverse map table, RMP),用于记录物理页的归属关系,可对虚拟机管理器的页表映射进行完整性检查,从而防止虚拟机管理器非法重映射或重用内存页。Intel TDX 则引入信任域(trust domain, TD)为机密虚拟机提供运行环境。为了实现 TD 的安全隔离,TDX 架构中增加一种新的 CPU 安全仲裁模式(secure-arbitration mode, SEAM)和一个经过安全认证的服务模块 TDX module。TDX module 运行在加密保护的 SEAM 内存中,承担部分原本由虚拟机管理器执行的功能,主要负责管理机密虚拟机的内存和寄存器状态,并通过安全扩展页表(secure EPT)实现隔离。可以发现,TDX 与 CCA 在体系结构上具有一定相似性。例如,TDX module 与 CCA RMM 在功能上均承担机密虚拟机管理的角色,同时 2 种架构都利用 2 阶段页表机制实现机密虚拟机之间的内存隔离。SEV-SNP 的核心安全逻辑主要依赖 PSP(platform secure processor)固件完成,TDX 依赖运行在安全环

Table 2 Comparison of CCA and Representative Confidential Computing Architectures

表 2 CCA 与代表性机密计算架构的对比

对比维度	Arm CCA	AMD SEV-SNP	Intel TDX	Arm TrustZone
体系结构	Arm	x86	x86	Arm
隔离粒度	虚拟机	虚拟机	虚拟机	进程/OS
硬件隔离扩展	RME	SEV	TDX module	TZASC
内存隔离管理	GPT，Stage-2 页表	RMP	secure EPT	TZASC
内存加密硬件	MPE	SME	TME-MK	无
安全内存大小	最大系统可用内存	最大系统可用内存	最大系统可用内存	16~64 MB
远程证明	支持	支持	支持	无统一机制
可信计算基	Monitor，RMM	PSP 固件	TDX module	Monitor，secure OS
物理攻击威胁模型	不考虑	不考虑	不考虑	不考虑
侧信道攻击威胁模型	不考虑	不考虑	不考虑	不考虑
应用兼容性	无需修改应用	无需修改应用	无需修改应用	需大量修改应用
应用场景	云计算/桌面端/嵌入式/移动端	云计算/桌面端	云计算/桌面端	嵌入式/移动端

境中的 TDX module，而 CCA 则依靠 RMM 和 Monitor 协同管理虚拟机上下文和安全状态。这 3 种机密计算架构均支持硬件级内存加密，例如 SEV-SNP 通过 SME（secure memory encryption）实现虚拟机内存加密，TDX 采用 TME-MK（total memory encryption-multi-key）为不同信任域提供独立密钥，而 CCA 则使用 MPE 提供内存加密支持。此外，这些机密计算架构在设计上均支持远程证明机制，并保持良好的软件兼容性，使现有操作系统和应用程序通常无需进行修改即可运行。

在安全威胁模型中，这些机密计算架构通常不考虑物理攻击和微架构侧信道攻击，而主要关注系统软件层面的威胁，即假设操作系统和虚拟机管理器是不可信的。3.2 节将进一步分析 CCA 的安全威胁和潜在攻击面。

1.5.2 CCA 与 TrustZone 安全架构

接下来将对比 CCA 与 TrustZone 2 代 Arm 安全架构的关键技术。如表 3 所示，在内存隔离方面，TrustZone 依赖 TZASC（TrustZone address space controller）与系统总线中的安全信号来区分安全世界和非安全世界的访问，其配置由安全世界中 S-EL1/2 的特权软件完成。然而，TrustZone 不支持 S-EL1/2 与 EL3 之间的软件隔离，攻击者一旦劫持安全世界的操作系统，将影响 EL3 代码和数据的安全。相比之下，CCA 通过颗粒保护表记录每个内存页面的物理地址空间归属，并在 MMU 的地址转换过程中进行颗粒保护检查。颗粒保护表仅能由 EL3 软件配置，并且 CCA 将 EL3 的代码和数据从安全世界变为根世界，与其他世界保持隔离，从而降低了最高特权软件受到安全世界软件影响的风险。

Table 3 Comparison of Security Key Technologies for CCA and TrustZone

表 3 CCA 与 TrustZone 的安全关键技术对比

技术维度	关键安全技术对比	CCA	TrustZone
内存隔离	处理器访问控制	✓	✓
	DMA 访问控制	✓	✓
	EL1/2 与 EL3 隔离	✓	✗
	管理内存隔离的层级	R.EL2/EL3	S.EL1/2
内存加密	硬件加密机制	✓	✗
设备隔离	处理器访问控制	MMU GPC	TZASC
	DMA 访问控制	SMMU GPC	TZPC
	管理设备隔离的层级	R.EL2/EL3	S.EL1/2
远程证明	硬件远程证明	✓	✗

在内存加密方面，CCA 通过 MPE 硬件对内存中的代码和数据进行加密保护。相比之下，TrustZone 并未提供内置的内存加密功能。尽管可以通过软件实现加密，但通常会带来显著的性能开销，并且需要修改现有软件。CCA 的设计要求除非安全世界外，其他 3 个世界必须启用加密，并且每个世界使用独立的加密密钥。然而，当对内存页面的物理地址进行转换时，MPE 不会自动清除原内存内容，因此仍需由 Monitor 清理内存以确保安全。

在设备隔离方面，2 种架构均支持对设备的访问控制，但实现机制存在差异。TrustZone 通过 TZPC（TrustZone protection controller）实现外设 DMA 访问

隔离。TZPC 负责配置外设访问权限，系统总线根据访问安全状态检查权限。当 DMA 访问的安全状态与权限不匹配时，会阻止非法访问。TZPC 配置通常由安全世界操作系统完成。相比之下，CCA 利用 MMU 中的颗粒保护检查阻止处理器对外设 MMIO 的非法访问。同时，DMA 设备的非法访问可通过 SMMU 中的颗粒保护检查加以限制。尽管两者都支持外设隔离，但 CCA 的安全性更高。TZPC 可以由 S-EL1 软件修改，而颗粒保护检查只能由 Monitor 配置，从而降低了攻击者通过劫持安全世界操作系统绕过隔离的风险。

在远程证明能力方面，TrustZone 本身未提供统一机制，相关功能通常由设备厂商在固件或 TEE 软件中实现。而 CCA 在设计上支持基于 HES 的远程证明，远程验证者通过验证平台证明与领域证明，能够同时确认底层平台和领域虚拟机的可信性。

总体来看，CCA 在大多数安全维度上相较于 TrustZone 提供了更强的保障。TrustZone 更适合嵌入式设备和移动终端，主要关注设备级安全和单一可信执行环境。CCA 在此基础上进行了扩展，不仅支持多个机密虚拟机环境隔离、硬件内存加密、设备访问控制和远程证明，还将 EL3 的根世界与其他世界隔离，确保最高特权代码和数据免受低特权世界的影响。

1.6 总 结

本节着重讨论了 CCA 关键技术的实现机制，重点对其系统架构中的硬件扩展和核心软件构成与其他机密计算架构的区别进行了详细说明。后续章节将进一步调研基于 CCA 系统的研究工作现状，并总结其在解决系统安全问题方面的主要分类情况。

2 CCA 的相关研究工作现状分析

自 Arm 推出 CCA 以来，其主要应用场景集中在云计算与人工智能领域。在云计算领域，CCA 借助硬件支持的机密虚拟机技术，旨在保护云端第三方租户数据的运行时安全，为敏感工作负载提供安全的基础设施支撑。在人工智能领域，相关研究主要探索如何使机密虚拟机能够安全高效地调用外部 GPU，FPGA 等专用加速器，以解决大模型等场景下数据隐私保护与算力释放之间的矛盾。此外，CCA 也为智能汽车、终端设备等需要本地化处理敏感数据的场景提供隐私保护支持。

然而，CCA 的硬件落地和实际使用仍处于发展阶段。因此，学术界围绕 CCA 的机密计算环境扩展、应用探索以及安全模型验证等方面展开了一系列研究。与此同时，产业界也在积极推动 CCA 软件生态的构建与完善。本节将系统梳理当前学术界在 CCA 相关方向的研究进展，归纳产业界在软件生态建设方面的主要成果，探讨其技术实现与行业应用趋势。为便于分析与讨论，本文将现有基于 CCA 的研究工作划分为 5 类，分别是机密计算环境扩展、面向设备机密计算扩展、基于 CCA 的应用、针对 CCA 安全增强和其他 CCA 相关研究，具体如表 4 所示。

2.1 机密计算环境扩展

CCA 为云平台和终端设备提供了硬件级的安全隔离能力，但其原生设计主要围绕领域虚拟机展开。近年来，研究者们致力于突破这一限制，将机密计算环境扩展到更细粒度的执行单元，从而构建多样化的可信计算生态。这些工作不再局限于传统的虚拟机级别隔离，而是将机密计算能力从虚拟机层级向上延伸至用户态、容器乃至进程内部，形成了多层次的隔离架构，在保障系统安全性的同时兼顾性能与灵活性。

Shelter[8] 系统扩展原生 CCA 以领域虚拟机为单位的隔离机制，提出了一种面向用户态的 TEE 构建方法。如图 4 所示，Shelter 在 CCA 原生方案的领域虚拟机级别之外进一步扩展应用级隔离(SApp)。该系统仅信任 EL3 层的 Monitor 且不依赖虚拟化技术，从而减小可信计算基并提升了性能。其核心设计在于提出多颗粒保护表内存隔离机制(见 3.1.2 节)，结合 CCA 的颗粒保护检查，实现对用户态内存的隔离。通过在不同处理器核心配置颗粒保护表项，Shelter 让用户态进程与高特权软件(如 OS、虚拟机管理器和 RMM)实现安全隔离。此外，Shelter 将非安全关键操作(如内存分配与系统调用)委托给非安全世界的宿主操作系统，进一步降低引入 Monitor 的代码量，同时维持与宿主操作系统的安全隔离。这一扩展架构不仅补充了 CCA 的机密计算环境层次，也为后续研究打开新的方向。

RContainer[9] 在 Shelter 基础上进一步探索面向容器的机密计算环境。该系统聚焦于提升用户态容器在非可信操作系统内核运行环境下的安全性，特别是针对 Iago 攻击等控制流劫持手段的防御能力。为此，RContainer 设计了一种轻量级可信 miniOS，用于处理内存管理、控制流保护以及系统调用等关键功能。该 miniOS 通过多颗粒保护表实现与非可信操作系统的隔离，并借助 shim 机制为每个容器维护专属

Table 4 Classification and Summary of CCA-Related Research

表 4 CCA 相关研究工作分类与总结

分类	项目名称	系统架构	功能原型平台	性能测试平台	可信计算基	隔离机制	功能
机密计算环境扩展	Shelter[8]	CCA	FVP	Juno R2	Monitor	root+Multi-GPT	扩展用户态机密计算
	RContainer[9]	CCA	FVP	RK3399 Firefly	Monitor+MiniOS	root+ Multi-GPT	扩展机密容器
	NanoZone[41]	CCA	FVP	RK3399 4B	Monitor+modules	root+ Multi-GPT +POE	扩展进程内隔离
	Aster[42]	CCA	QEMU		Monitor+RMM	Multi-GPT+S2PT	扩展安卓端机密计算
面向设备机密计算扩展	CAGE[10]	CCA	FVP	Juno R2	Monitor+RMM	root+Multi-GPT+ Protected SMMU GPC	扩展集成 GPU 机密计算
	ACAI[11]	CCA	FVP	Zynq UltraScale+	Monitor+RMM	root+Multi-GPT+ Protected SMMU GPC	扩展 PCIe CPU 机密计算
	PORTAL[12]	CCA	FVP	OrangePi 5 Plus	Monitor+RMM+ Trusted VM	root+Multi-GPT+ Protected SMMU GPC	扩展高性能的设备可信 I/O
	DEVLORE[43]	CCA	FVP		Monitor+RMM	root+Multi-GPT+ Protected SMMU GPC	扩展具备安全中断的集成外设机密计算
基于 CCA 应用	SCRUTINIZER[13]	CCA	FVP	Juno R2	Monitor+Agent	root+ Multi-GPT	针对 TrustZone 安全世界的可信自省
	HitchHiker[14]	CCA	FVP	Juno R2	Monitor+Daemon	root+ Multi-GPT	针对系统审计日志的安全保护
	FortifyPatch[44]	CCA	FVP	Raspberry Pi 3B	Monitor	root+ Multi-GPT	针对虚拟机管理器的热补丁安全
	BarriCCAde[45]	CCA	QEMU		Monitor+RMM	realm VM	针对闭源设备驱动的安全保护
	GuaranTEE[46]	CCA	FVP		Monitor+RMM	realm VM	针对机器学习安全保护
针对 CCA 安全增强	Des. & Ver[47]	CCA	FVP	Neoverse N1	Monitor+RMM	realm VM	针对 RMM 形式化验证
	Sharing[48]	CCA		AmpereOne	Monitor+RMM	realm VM +核心划分	抵御瞬态执行攻击的核间隔离防御架构
	CPC[49]	CCA	FVP	SEV SNP	Monitor+RMM+Modules	realm VM	面向机密虚拟机的灵活安全的维护框架
	CubeVisor[50]	CCA	FVP	RK3399 Rock 4B	Monitor	root+ Multi-GPT	多实例化 RMM 的机密虚拟机框架
其他	virtCCA[51]	TrustZone	Arm Server	Kunpeng 920	Monitor+RMM	TZASC+S.EL2	基于 TrustZone 的 CCA
	TwinVisor[52]	TrustZone	FVP	HiSilicon Kirin 990	Monitor+SVisor	TZASC+S.EL2	基于 TrustZone 的机密虚拟机系统
	TZ&CCA[53]	CCA	FVP	Juno R2			TrustZone 与 CCA 对比

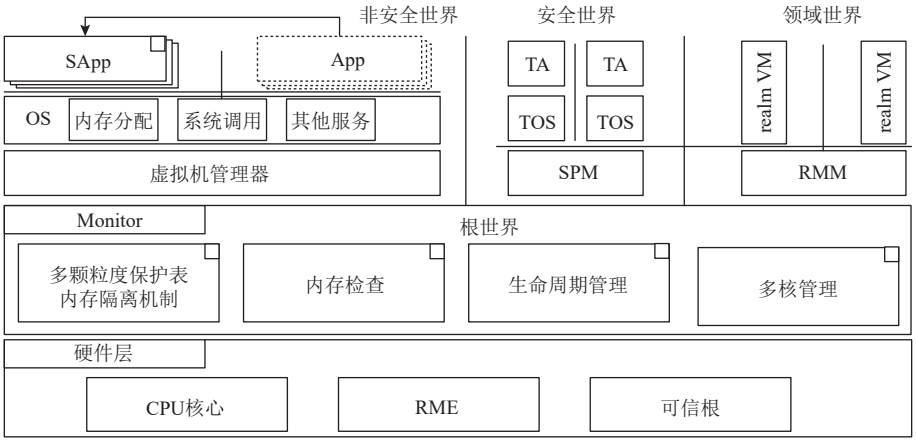


Fig. 4 User-space confidential computing architecture of Shelter

图 4 Shelter 用户态机密计算架构

的物理隔离空间，从用户态到内核态都进行了隔离。相较于 Shelter，RContainer 继承多颗粒保护表机制的设计优势，并且进一步增强了内核空间的安全边界。

NanoZone[41] 则在上述用户态与容器级机密环境的基础上，进一步深入至进程内部，构建面向进程内可信隔离的机密计算框架，以支持更细粒度的安全需求。该系统结合了 Armv8.9-A 中新增的权限覆盖扩展（privileged overlay extension，POE）硬件特性，该机制理念上类似于 Intel MPK，允许通过修改 POE 寄存器快速切换内存访问视图。NanoZone 借此可在无需上下文切换的情况下，实现进程内多个内存域之间的高效隔离。此外，NanoZone 与多颗粒保护表机制协同设计，结合代码指针完整性检查，防止攻击者在域切换过程中利用漏洞实现越权访问。

Aster[42]将 CCA 的机密计算能力引入移动终端中的安卓系统,尝试在移动设备操作系统层面提供基于 CCA 的应用隔离机制。该系统通过构建安全沙箱运行安全敏感应用,并借助类似 Shelter 的多颗粒表保护机制实现隔离保护,同时保持与现有安卓框架的兼容性。Aster 展示了 CCA 在移动平台上的适用能力。

综上,这些研究工作共同推动了 CCA 向多层次、多粒度、跨生态的方向演进,为未来安全 CCA 架构的设计提供了参考。

2.2 面向设备的机密计算扩展

随着机密计算技术的不断发展,研究者逐渐将关注点从传统的 CPU 侧隔离扩展至设备侧的机密计算支持。尤其在人工智能与高性能计算等场景中,GPU 等异构加速器已成为关键计算资源,其处理的数据往往具有高度敏感性。因此,将机密计算机制从 CPU 拓展至设备侧,已成为提升系统整体安全的重要路径。在 CCA 刚推出的初期阶段,多个研究陆续展开,旨在围绕 CCA 构建异构机密计算环境与安全高效的设备 I/O 机制,增强领域世界与设备交互之间的安全性。

CAGE[10]是该方向的代表性工作之一,聚焦于统一集成 GPU 场景下机密计算能力的拓展。Arm 主流架构支持统一内存访问,而集成 GPU 由于与 CPU 共享物理内存资源,面临跨安全域的数据泄露风险。CAGE 是首个针对集成 GPU 设计的 CCA 扩展方案,解决了原生 CCA 缺乏安全 GPU 支持的问题。如图 5 所示,该工作借助多颗粒表内存隔离设计,并引入影子任务机制管理 GPU 任务,确保敏感计算仅在受控的机密路径中执行。同时,CAGE 将颗粒保护机制与 SMMU 的硬件访问控制相结合,实现对 CPU 与 GPU 之间的数据隔离和完整性保护。在不依赖额外硬件修改的前提下,该方案引入了较低的计算性能开销,在安全与性能之间实现了平衡。

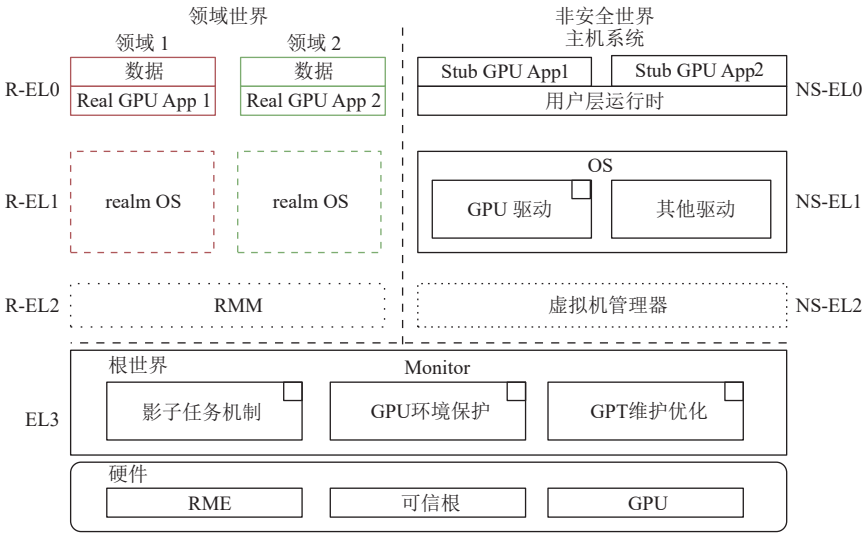


Fig. 5 GPU confidential computing architecture in CAGE

图 5 CAGE 中的 GPU 机密计算架构

ACAI[11]将研究视角进一步拓展至通过 PCIe 挂载的异构加速器(包括 GPU, FPGA 等),提出将这类设备纳入 CCA 框架下的可信执行范围。鉴于当前 CCA 尚未原生支持此类设备,RME-DA 还停留在概念阶段,未有具体的硬件支持,ACAI 通过扩展 Monitor 与 RMM 组件,将设备侧访问映射为领域世界的隔离区域,使加速器能够像访问领域虚拟机内存一样进行受控访问。该方案同样结合多颗粒保护表隔离机制,通过设计针对 SMMU 的设备页表保护,实现了更灵活的设备可信访问控制。

PORTAL[12]旨在为 CCA 构建一条高效安全的设备 I/O 路径。不同于依赖内存加密保障 I/O 路径安全的传统方法,PORTAL 以访问权限隔离为核心设计思想,结合多颗粒表保护机制进行细粒度的访问控制。其关键设计在于动态切换内存访问权限,在主机与设备之间建立接口,为领域虚拟机和外设构建一条独占的明文内存路径。通过专属的领域页表管理结构(见 3.1.3 节),PORTAL 允许设备安全访问预配置的物理内存页,并与非可信组件实现隔离。

DEVLORE[43]则进一步推进了对集成异构设备的机密计算支持。与上述工作类似,DEVLORE 通过颗粒保护机制和设备页表隔离,使集成设备能够直

接接入 CCA 的机密环境，并将设备的 DMA 操作纳入可信内存访问控制之中。该方案还特别优化了设备中断路径，引入委托再校验策略：在中断传递路径中，首先由虚拟机管理程序处理中断请求，随后由根世界的 Monitor 对中断处理逻辑进行完整性验证，从而防止中断路径被恶意利用而破坏敏感任务的执行。

总体来看，面向设备的机密计算扩展是完善 CCA 生态体系中的关键一环。从集成 GPU 到 PCIe 外设，从安全高效的 I/O 到嵌入式计算设备，研究者不断探索如何将领域世界的安全保障能力延伸至各类设备与通信路径之中。

2.3 基于 CCA 的应用

随着 CCA 硬件特性逐渐受到研究者关注，其应用已不再局限于传统的 TEE 构建，而是被越来越多地用于提升现有系统与应用场景的安全性。这类研究不仅利用 CCA 提供的硬件隔离能力（如颗粒保护检查等）强化系统防护，还针对特定应用需求（如系统审计日志、驱动隔离等）设计了定制化解决方案。总体来看，现有应用研究可归纳为 2 类：一是对传统安全机制的增强，例如借助 CCA 提升 TrustZone 安全世界的安全性；二是面向新兴场景的保护，如确保机器学习模型数据的安全。

SCRUTINIZER[13] 是此类工作中对 CCA 应用的代表工作之一。该系统聚焦于提升对 TrustZone 安全世界的可信取证能力。传统 TrustZone 依赖安全与非安全世界的环境划分，但安全世界缺乏运行时自省机制，一旦被攻破（如利用漏洞或加载恶意固件），整个安全世界执行环境将失去可信性。SCRUTINIZER 基于 CCA 的根世界构建自省能力，是首个面向受损 TrustZone 环境的安全自省框架。如图 6 所示，系统通过设计一个位于安全世界内部的轻量代理，并通过双颗粒保护表实现其与安全世界的隔离，从而能够以较低开销获取安全世界的内存快照，同时减少引入根世界的代码规模。该机制为安全世界的运行时分析与恶意代码检测提供了一种高效且可信的系统自省方案，弥补了现有 TrustZone 系统在安全世界受损情况下自省能力不足的问题。

HitchHiker[14] 聚焦于系统审计日志的安全保护问题。现有操作系统的审计架构（如 Linux audit）通常运行在非安全世界，易被攻击者篡改或删除。为应对这一安全问题，HitchHiker 提出一种基于 CCA 硬件隔离的安全设计，利用颗粒保护表实现动态权限切换审计日志的内存区域，从而在无需加密和内存拷贝的情况下保护系统日志。该系统可实时收集和保护多种审计数据，不仅保障审计数据完整性，同时维持较高的性能。

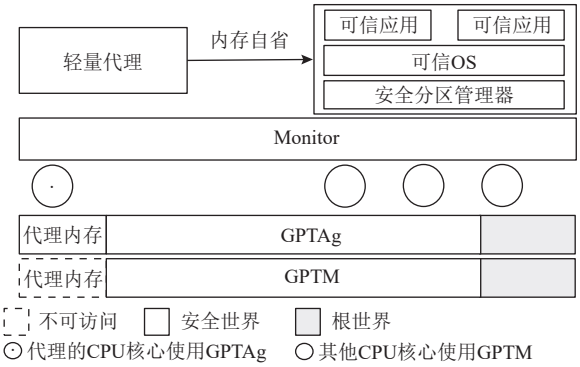


Fig. 6 Trusted introspection architecture of SCRUTINIZER for TrustZone secure world

图 6 针对 TrustZone 安全世界的 SCRUTINIZER 可信自省架构

FortifyPatch[44] 则致力于解决非安全世界下虚拟机管理程序的在线热补丁安全问题。传统热补丁机制往往缺乏安全保护与完整性验证，容易被攻击者绕过。为此，FortifyPatch 提出一种抗篡改的补丁框架，结合 CCA 的颗粒保护表与 TrustZone 原本的硬件机制，构建可信补丁部署系统。系统通过性能监控单元定期执行完整性检查，并隔离补丁代码的访问路径，从而在保障打补丁效率的同时，提升虚拟机管理程序热补丁的整体安全性和稳定性。

BarriCCAde[45] 探索将 CCA 机制应用于驱动模块的隔离保护。其设计将闭源设备驱动运行在领域虚拟机中，避免操作系统或虚拟机管理程序对其直接访问，从而实现最小信任基础下的驱动安全。通过该机制，可阻止漏洞经由驱动影响系统安全，并通过硬件隔离能力增强设备访问的安全性，降低驱动攻击面。

GuaranTEE[46] 则关注人工智能场景下的执行安全，提出一种融合 CCA 的可信人工智能执行框架。该系统利用 CCA 提供的机密计算环境，将机器学习模型及推理过程放置到隔离的领域世界中运行，并与非领域世界的高性能计算过程解耦，实现模型安全与执行性能的兼顾。

综上所述，上述基于 CCA 的应用研究挖掘了该架构在多场景、多层级系统安全构建中的潜力。从安全世界的自省、审计日志的保护，到虚拟机热补丁机制与人工智能推理任务的防护，研究者借助 CCA 的隔离能力与良好兼容性的虚拟化支持，为复杂系统构建了安全保障。未来，随着 CCA 硬件部署的加

速，预计其在数据安全保护领域中的应用潜力将得到进一步释放。

2.4 针对 CCA 的安全增强

尽管 CCA 通过硬件隔离提供了坚实的安全基础，但其复杂的软件栈、新兴的侧信道攻击等仍可能带来潜在风险。近年来，研究者从形式化验证、架构加固与运行时可信检测等角度对 CCA 的安全性进行了系统性增强。这些研究提升了 CCA 核心组件的可信性(如 RMM 的形式化证明)，还针对瞬态执行攻击、虚拟机维护等场景设计了专用防御机制，从而进一步巩固了机密计算环境的安全。

在此方向上的早期代表性工作是 Des. & Ver[47]。该研究对 CCA 关键固件 RMM 进行形式化建模与验证，首次将形式化方法引入 CCA 软件栈中。该研究采用 Coq 构建了理想-真实机模型，将 RMM 的 C 语言与汇编代码的安全性归约至一个具有限定资源分配权限的可信固件，并通过定理证明工具验证了其行为一致性与安全属性。该工作推动了 RMM 规范与开源实现的安全审计流程，为 CCA 软件生态提供安全支撑，提升了 RMM 的可信度。

Sharing[48] 针对瞬态执行攻击对 CCA 机密虚拟机的威胁提出了一种基于处理器核心间隔离的软件防御架构。尽管 CCA 提供内存与寄存器隔离，共享核心资源仍可能被攻击者利用进行侧信道攻击。为此，Sharing 通过修改固件调度逻辑，实现机密虚拟机对物理核心的独占运行，并将中断或主机任务分离至不同的核心，避免任何微架构共享，从源头阻断多种瞬态执行攻击路径。该研究表明，在具备隔离保障的 CCA 环境下，微架构攻击仍是不可忽视的风险，而基于处理器核心的物理隔离机制为应对此类威胁提供了一种可行的防御方案。

CPC[49] 提出了针对机密虚拟机的灵活且安全的维护框架。传统虚拟机维护操作(如热迁移、快照)需要宿主虚拟机管理器访问客户机内存，这与 CCA 的隔离原则冲突。该研究提出机密过程调用机制，允许在机密虚拟机内部署受 RMM 保护的安全模块，通过 RMM 管理的 2 阶段页表动态划分出可维护内存区域。这些区域仅对特定维护操作模块开放，且由 RMM 严格审计其访问合法性，保证维护过程始终在可信范畴内执行，防止攻击者劫持或篡改。

CubeVisor[50] 则从架构层面重构了 CCA 的软件栈，通过多实例化 RMM 提升整体安全性。受 Shelter 的多颗粒保护表启发，CubeVisor 将单 RMM 拆分为多个轻量级实例，每个实例仅管理 1 个领域虚拟机及其关联资源。这种设计缩小了机密计算实例的安全边界(如一个实例被攻破不影响其他 RMM 和领域虚拟机)，可实现互不信任的多租户场景下的隔离。该机制不同于传统虚拟机管理器单一的信任模型，支持领域虚拟机管理器的并行隔离，进一步增强 CCA 的安全。

这些研究从设计验证、微架构安全、运行时机制到系统级多实例扩展等不同维度出发，围绕 CCA 的安全保障能力进行了系统性增强。它们共同表明，CCA 不仅是一个提供静态隔离的硬件架构，更是在多种攻击威胁和动态运维场景下，能够不断适应与演进的可信基础平台。通过这些工作，CCA 正在从安全特性支撑逐步转向安全能力塑造，为构建下一代可信计算体系提供基础与多样化路径。

2.5 其他 CCA 相关研究

除了上述聚焦于架构扩展、设备支持、应用场景和安全增强的研究外，仍有一些与 CCA 密切相关的研究工作。这些工作可分为 2 类：一是系统的向前兼容实现、CCA 与 TrustZone 的对比研究等；二是产业界的开源生态构建，涵盖从官方参考实现到终端设备部署场景的支持。这些研究完善了 CCA 生态，为学术界和产业界提供了宝贵的实践经验。

为了在尚不具备 CCA 硬件支持的 Arm 平台上运行基于 CCA 的软件栈，virtCCA[51] 提出一种基于 TrustZone 的 CCA 软件栈兼容层。由于非 CCA 设备无法支持领域管理扩展，virtCCA 利用 TrustZone 的 secure EL2 特权模式模拟 RMM 功能，并扩展 KVM 以支持在安全世界中运行机密虚拟机。研究人员实现了与 CCA API 兼容的软件与固件栈，并通过 MMU 的 2 级页表代替颗粒保护表机制。

在 CCA 架构发布之前，TwinVisor[52] 已提出类似 CCA 理念的机密虚拟机架构。该系统基于 TrustZone 的 secure EL2 引入双虚拟机管理器的设计，将传统虚拟机管理器划分为安全世界与非安全世界 2 部分：安全管理器负责上下文管理与内存隔离，非安全管理器处理调度与虚拟机维护。这种结构在功能划分上与 CCA 的 RMM 架构相似，是多世界资源分离与最小可信计算基理念的早期探索。为解决 TrustZone 的安全世界区域数量有限的问题，TwinVisor 还设计基于连续物理内存的分配管理机制，并引入内存回收算法以提升管理效率。

TZ&CCA[53] 则以系统化视角对比分析了 TrustZone 与 CCA 之间的异同。该研究从灵活性、安全性与性能 3 个维度出发，对两者进行了定量与定性的评估。

研究揭示了传统 TrustZone 的安全世界机制在多租户隔离与最小信任边界等方面的局限，并指出 CCA 通过领域管理扩展在硬件粒度、虚拟化兼容性和边界管理等方面具有明显优势。

在工业界，围绕 CCA 的软件生态也日趋完善。Arm 与 Linaro 联合开发了 RME 软件栈[54]，包括 RMM 及其配套工具链，目前主要运行于 Arm 官方提供的 FVP（fixed virtual platform）[55] 模拟环境，作为参考实现用于指导厂商部署与验证自身方案。与此同时，QEMU CCA[56] 项目的推进为研究社区提供了新的模拟平台，该项目致力于将 RME 软件栈集成至 QEMU CCA 模拟器中，使研究者无需 FVP 平台即可在软件层面开展 CCA 架构的验证与实验，提升模拟性能并促进开源社区的发展。

此外，三星公司主导的 Islet 项目[57] 展示了 CCA 在终端设备中的应用前景。Islet 是一个基于 Rust 语言开发的开源机密计算平台，专为在 CCA 架构下的终端设备实现机密计算而设计。系统聚焦于端侧用户数据的隐私保护，通过本地运行领域机密计算环境，实现对敏感数据的保护。Islet 的设计与实现表明，CCA 不仅适用于云端虚拟化场景，也具备在智能手机、IoT 设备等边缘计算环境中提供隔离和高性能数据保护的潜力，进一步拓展了其实际应用的广度。

综上所述，围绕 CCA 的软件兼容性、体系结构演进与产业化路径正在逐步清晰。学术界通过模拟 RMM、改造虚拟化层等手段实现了在非 CCA 平台上的验证与过渡，工业界则通过标准软件栈、开源工具链与终端应用原型不断完善 CCA 生态。这一系列工作共同推动了 CCA 从理念探索走向实际部署，加速其在可信计算领域的全面落地。

2.6 总 结

本节对基于各种 CCA 的研究工作现状进行了分析和归纳，这些相关工作的发展脉络如图 7 所示。综合前述各类研究工作可以发现，当前 CCA 研究大体呈现出 3 个明显特点。首先，在体系结构层面，CCA 的设计主要围绕领域虚拟机这一隔离单元展开，其核心目标是在虚拟机级别提供机密计算能力。然而，随着实际应用需求的多样化，研究者开始尝试突破这一边界，将机密计算能力从虚拟机层级进一步向更细粒度的 TEE 扩展，从而在保证安全隔离的同时提升系统灵活性与资源利用效率，使 CCA 能够在不同的系统环境中发挥作用。其次，在系统边界层面，研究对象从单纯的 CPU 硬件隔离能力扩展至 GPU、PCIe 设备及 I/O 通信路径等各类计算资源，体现出当下广泛的异构机密计算系统构建需求。最后，在应用层面，研究者开始围绕各种系统安全和人工智能计算场景探索 CCA 硬件辅助安全的研究潜力，推动机密计算从底层研究逐步走向面向实际场景的安全能力构建。

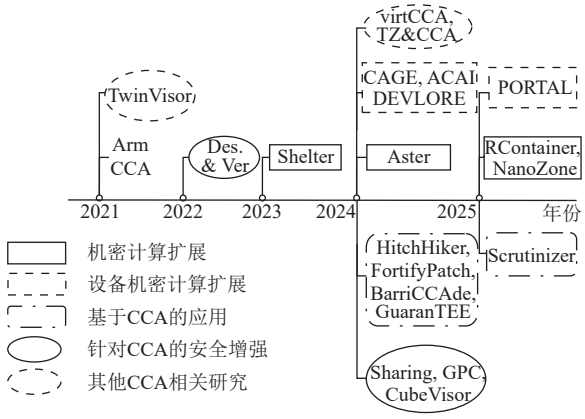


Fig. 7 The development context of CCA-related work

图 7 CCA 相关工作发展脉络

与此同时，从整体工作现状来看，现有研究虽然在 CCA 的机密计算环境构建、设备扩展以及应用探索等方面取得了丰富成果，但不同研究方向之间仍然呈现出一定的技术碎片化特征。大量工作往往针对特定问题提出局部改进，而对 CCA 体系结构中若干关键设计机制的共性规律缺乏系统梳理和清晰的全局视角。此外，在 CCA 的威胁模型假设下，系统仍然存在一定的安全风险和潜在攻击面。

在第 3 节，本文将进一步提炼已有 CCA 系统设计中的关键共性机制，并结合已有研究分析 CCA 可能面临的安全威胁。这将有助于从整体理解 CCA 研究工作的共性设计逻辑及其安全边界，为后续相关研究提供设计参考。

3 CCA 共性机制与安全威胁分析

3.1 CCA 研究工作的共性机制和设计方案

在表 4 中，本文系统归纳了基于 CCA 的相关工作。从当前研究现状来看，围绕 CCA 的工作大体可分为 4 类：1）机密计算扩展。它是在 CCA 隔离环境基础上的体系结构或系统能力拓展，例如支持用户态的轻量级可信执行环境。2）设备机密计算扩展。它增强设备侧机密计算能力，如为外设或边缘设备构建 TEE。3）基于 CCA 的应用。它利用 CCA 构建上层应用与系统服务，如可信取证、审计、在线补丁、闭源驱动以及机器学习模型保护等。4）针对 CCA

的安全增强。它通过提升固件可信性或构建新的软件架构,以应对更加多样化的安全威胁。

尽管这些系统的设计目标和应用场景各不相同,但其所采用的核心技术路径却存在潜在的共性。这表明当前研究已逐渐围绕 CCA 本身提供的硬件级隔离能力展开利用与组合创新,为多样化的机密计算需求提供解决方案。本节旨在基于第 2 节的研究工作归纳凝练这些共性机制。通过对已有代表性工作进行分析,可以将 CCA 系统中常见的关键技术路径大致归为 3 类共性机制。表 5 给出了共性技术机制和不同类别代表性工作之间的映射关系、解决的通用痛点和适用边界。下面将具体介绍这 3 种共性机制。

Table 5 Mapping Relationship Table of Common Mechanisms in CCA and Representative Works

表 5 CCA 共性机制与代表性工作的映射关系表

共性机制	代表性工作	解决的通用痛点	适用边界
基于根世界进行隔离与控制的设计方案	① Shelter, RContainer, Aster ② CAGE, ACAI, PORTAL ③ SCRUTINIZER, HitchHiker, FortifyPatch ④ CubeVisor	系统自身的安全保护; 减小系统的可信计算基; 使用 CCA 的隔离能力。	各类需要使用颗粒保护检查隔离能力的场景
基于多颗粒保护表的内存隔离机制	① Shelter, RContainer, NanoZone, Aster ② CAGE, ACAI, PORTAL, DEVLORE ③ SCRUTINIZER, HitchHiker, FortifyPatch ④ CubeVisor	灵活安全的隔离能力; 轻量化的隔离能力。	各类机密计算环境构建; 各类应用的安全保护
SMMU 与 2 阶段页表的设备访问隔离机制	② CAGE, ACAI, PORTAL, DEVLORE	安全可控的设备访问	各类机密计算的安全设备场景

注:①机密计算扩展; ②设备机密计算扩展; ③基于 CCA 应用; ④针对 CCA 安全增强。

3.1.1 基于根世界进行隔离与控制的设计方案

从表 5 映射关系可以发现,在第 2 节的 5 类相关研究中,几乎大部分工作都会在根世界扩展 EL3 Monitor 以构建各自的核心安全功能。由此可以抽象出一种在 CCA 系统设计中广泛采用的技术机制,即基于根世界进行系统隔离与安全控制。

该设计方案解决了 CCA 系统构建的 3 个通用痛点。1)系统自身的安全保护。EL3 Monitor 运行于根世界,在 CCA 架构中具有最高特权和最强隔离性,在 Monitor 中的实现功能能够保障这些系统自身关键代码与数据的安全。2)减小系统的可信计算基。部分代表性系统(例如 Shelter, FortifyPatch, CubeVisor)选择只在 Monitor 中实现关键安全功能,从而可以仅信任 Monitor 并将 RMM 排除在外,使得系统整体的可信计算基被缩小。3)使用 CCA 的隔离能力。CCA 的颗粒保护检查是实现内存与设备访问控制的核心机制,而颗粒保护表和颗粒保护检查寄存器仅根世界有权限访问与配置,因此任何需要利用 CCA 隔离能力的系统都必须通过根世界代码实现相关管理逻辑。

从适用边界来看,基于根世界进行隔离与控制的设计方案适合所有需要利用 CCA 颗粒保护检查隔离能力的系统场景。通过分析代表性工作(例如 Shelter, CAGE, ACAI, SCRUTINIZER 等)的开源代码发现,在 EL3 中扩展 Monitor 逻辑主要有以下 3 种思路。首先,在 Monitor 中扩展 SMC(secure monitor call)接口,为上层程序提供 API 调用入口。大部分系统都选择在对应的 SMC 处理函数中实现具体的系统功能。此外,根据各自系统对内存与设备隔离的不同需求,还会在颗粒保护表函数库中增加新的函数功能,用于实现对 MMU 或 SMMU 的颗粒保护检查控制。最后,部分工作进一步优化引入 Monitor 的代码规模。由于系统功能需要集中在根世界实现,这会增加 Monitor 的代码量并扩大潜在攻击面。因此,部分工作提出了 Monitor 可信计算基优化设计。例如,RContainer 实现了一个可信 miniOS 来承担容器操作系统的功能,只在 Monitor 维护尽可能少的安全功能。SCRUTINIZER 则将大部分功能代码从 Monitor 中解耦,并迁移到安全世界的可信隔离代理中实现自省功能。

3.1.2 基于多颗粒保护表的内存隔离机制

分析第 2 节的 5 类相关工作可以发现另一个共性技术机制——基于多颗粒保护表的内存隔离。如表 5 映射关系所示,所有机密计算扩展、设备机密计算扩展以及部分 CCA 应用和安全增强工作都采用了这一机制。主要是因为这些系统都需要实现更灵活轻量的隔离能力。

基于多颗粒保护表的内存隔离机制最早在 Shelter 工作中提出,用于用户态 TEE 的构建。具体来说,CCA 的颗粒保护表主要被用于划分内存所属世界,因此无法将用户程序同时与各世界(即 normal, realm, secure)的所有高特权软件隔离。为满足更加灵活和轻量的用户态隔离需求,Shelter 提出为每个 SApp 应用进行单独的颗粒保护表配置来实现多个隔离的用

户态执行环境。如图 8 所示，在 Shelter 系统中，每个 SApp 应用都有一个颗粒保护表，由 Monitor 统一进行管理并负责不同处理器核心的颗粒保护表切换。这样，在运行 SApp 和宿主软件的不同处理器核心，对同一 SApp 物理内存将具有不同的物理地址空间访问权限，从而在 SApp 与操作系统、虚拟机管理器和 RMM 等高特权软件之间建立清晰的安全边界。

基于多颗粒保护表的内存隔离机制主要解决了以下 2 个通用痛点。1)灵活安全的隔离能力。该机制可以为每个执行对象提供独立的内存访问范围，并与其他世界隔离，就像页表一样控制访问权限。此外，因为颗粒保护表由根世界管理，其他世界的软件无法修改，可以保证隔离配置的安全性。2)轻量化的隔离能力。CCA 的机密计算环境依赖虚拟化技术构建虚拟机级隔离，而多颗粒保护表机制直接在根世界配置访问控制，仅依赖颗粒保护检查即可完成隔离。因此，该机制能够在不引入虚拟化软件的情况下实现轻量级隔离，降低系统复杂度并减少性能开销。

从适用边界来看，多颗粒保护表隔离机制逐渐成为构建最小可信执行单元(如机密容器)和设备安全访问的关键工具。例如，在机密计算环境扩展中，Shelter 和 RContainer 分别利用多颗粒保护表实现进程和容器的细粒度隔离。在设备机密计算场景中，CAGE, PORTAL 等工作利用该机制保护设备 MMIO 内存区域或构建安全高效的 I/O 通道。在应用和安全增强工作中(如 SCRUTINIZER, HitchHiker, Fortify-Patch, CubeVisor)，多颗粒保护表也被用于实现自省框架、在线补丁、审计日志以及多 RMM 实例等各类组件的隔离保护。

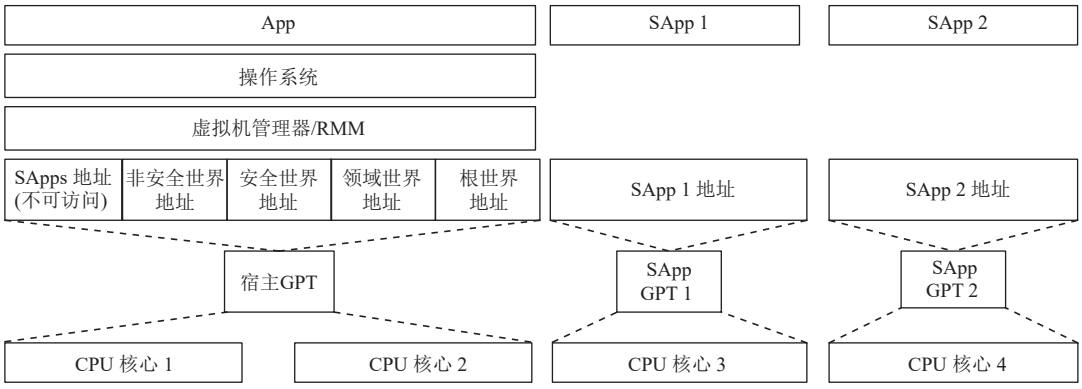


Fig. 8 Memory isolation mechanism design of multi-grain protection table in Shelter

图 8 Shelter 的多颗粒保护表内存隔离机制设计

3.1.3 SMMU 与 2 阶段页表的设备访问隔离机制

分析第 2 节关于设备机密计算扩展的相关工作，发现现有研究(如 CAGE, ACAI, PORTAL, DEVLORE)普遍通过结合 SMMU 与 2 阶段页表机制，将设备的访问范围限制在可控区域内。基于这些工作，可以进一步凝练出第 3 个共性机制：SMMU 与 2 阶段页表的设备访问隔离机制。具体而言，该机制通过 SMMU 流表与 2 阶段页表的地址转换，可以对设备发起的 DMA 访问进行重映射，从而精确控制设备能够访问的物理内存范围，实现对机密虚拟机设备访问权限的细粒度管理。此外，系统通常利用多颗粒保护表机制对关键设备访问控制策略进行保护。一方面，通过保护设备 MMIO 区域，使设备控制接口仅对相应的机密虚拟机可见，从而确保设备控制权归属于指定的机密计算环境。另一方面，对 SMMU 流表及 2 阶段页表等关键数据结构进行保护，从而避免 SMMU 控制结构被非法访问或修改。

该机制主要解决了设备访问控制中的一个通用痛点问题：在保证机密计算隔离性的前提下，如何实现安全可控的设备访问。从适用边界来看，在现有缺乏 RME-DA 的情况下，可以采用 SMMU 与 2 阶段页表相结合的设备访问隔离机制。例如，CAGE 系统展示了如何将 SMMU 机制与多颗粒保护机制协同使用，从而构建支持集成 GPU 设备的异构机密计算环境；而 ACAI 与 PORTAL 等系统则将 SMMU 相关的 2 阶段页表管理委托给 RMM 或可信虚拟机进行管理(如图 9 所示)，并通过多颗粒保护表机制保护这些 SMMU 关键结构的安全性，以实现领域世界内部的安全设备访问功能，并在保证隔离性的前提下维持较高的设备 I/O 性能。

3.1.4 总 结

综上所述，现有 CCA 相关研究在关键机制设计

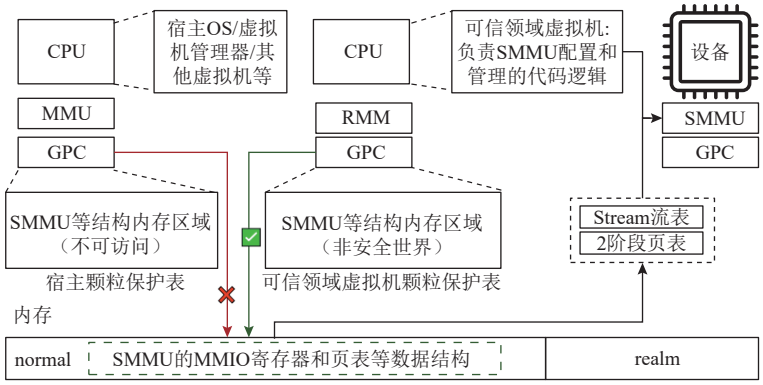


Fig. 9 Protection of PORTAL for SMMU configuration code and page table data structures

图 9 PORTAL 保护 SMMU 配置代码和页表数据结构

上呈现出较为明显的共性特征，普遍依赖由根世界负责配置的多颗粒保护表机制来构建系统隔离基础。基于此，可得到以下启发：

1）根世界隔离的基础性作用。几乎所有系统均依赖 EL3 根世界实现关键隔离，Monitor 自身的安全能力成为保障系统安全的核心。

2）多颗粒保护表的跨层级扩展能力。该机制适用于从用户态、内核态到设备的多层级保护需求，具备良好的应用扩展性。

3）硬件-软件协同设计的系统实现。研究者广泛结合 CCA 原生硬件特性（如 RME, GPC）与软件扩展能力（如在 Monitor 与 RMM 实现多颗粒保护表隔离机制、SMMU 和 2 阶段页表隔离、影子任务、可信 miniOS 等）构建安全系统，体现出软硬件协同设计的技术趋势。

通过上述系统剖析，本文进一步揭示当前 CCA 研究背后的底层设计逻辑与未来演进方向，为理解和推动该架构在可信计算中的发展提供技术参考。

3.2 CCA 研究工作的威胁模型与潜在攻击面

现有研究工作普遍沿用 CCA 官方的威胁模型假设：攻击者主要来自不可信的软件栈，包括宿主机操作系统和虚拟机管理器。在信任边界方面，现有工作通常假设 EL3 Monitor 和 RMM 是可信的组件，并且不考虑物理攻击与微架构侧信道攻击。在这一威胁模型下，CCA 通过 RME 在硬件层面隔离领域世界的内存，并通过 RMM 和 Monitor 保护虚拟机上下文切换状态，从而防止不可信软件访问机密数据。

然而，这个威胁模型假设实际上仍然存在潜在攻击面。近年来针对机密计算的安全研究表明，即使在硬件隔离机制存在的情况下，攻击者仍可能通过固件接口、I/O 交互路径、跨机密虚拟机的数据共享以及虚拟机上下文切换等多种途径攻击机密虚拟机。因此，本文进一步分析针对 CCA 系统的威胁来源，并结合现有研究提出的典型攻击案例说明 CCA 可能面临的安全风险。

3.2.1 固件接口

固件接口是机密虚拟机中的一个潜在攻击面。由于机密虚拟机在启动、配置及运行过程中需要通过接口与底层固件进行交互。攻击者可能利用接口功能漏洞绕过隔离机制，从而破坏机密虚拟机的隔离边界。

一个典型例子是 BadAML[58]，利用了 AML（Acpi machine language）代码可以被动态加载并在操作系统内核上下文执行的特性实现攻击。具体来说，ACPI（advanced configuration and power interface）通常用于设备配置。攻击者可通过不可信宿主机调用机密虚拟机的固件接口加载任意恶意 AML 二进制代码到 ACPI 表，在机密虚拟机启动后由内部操作系统解析并执行。ACPI 表的动态注入通常发生在机密虚拟机启动之后，此时远程证明过程已经通过验证，因此该攻击可绕过远程证明的完整性保护。类似攻击已在 AMD SEV-SNP 和 Intel TDX 系统中被验证。BadAML 攻击与机密计算的体系结构无关，因此 CCA 也可能受到类似攻击。

综上所述，在 CCA 系统设计中需要加强固件接口的安全防护。例如，针对 BadAML 攻击，可在 AML 解析器中限制代码的可访问内存范围，或对关键 ACPI 表进行完整性验证，以防止固件接口被利用而破坏机密虚拟机的安全。

3.2.2 I/O 交互路径

I/O 路径是机密虚拟机与外部设备交互的关键通道，也是当前机密计算的潜在攻击面。在现有 CCA 系统中，机密虚拟机的设备访问通常通过虚拟设备实现，并由宿主机虚拟机管理器将 I/O 请求转发至实

际设备进行通信。这意味着 I/O 路径经过宿主机控制的不可信软件。例如，网络驱动通常将数据包写入与宿主机共享的内存页，再由虚拟机管理器转发到物理网卡。由于该通道对宿主机完全可见，攻击者可能修改设备的输入输出数据或通过 DMA 操作篡改内存内容[11]。

为缓解这些风险，Arm 将在 CCA 后续设计中提出基于 RME-DA 的安全设备分配机制，支持将设备分配给指定的领域虚拟机并实现设备保护。同时，现有研究中的 SMMU 与 2 阶段页表的设备访问隔离机制也能将设备专门分配给某个机密虚拟机，实现与 RME-DA 相似的效果。

然而，即使在引入这些机制之后，仍然存在可用性攻击、恶意设备攻击和侧信道攻击等潜在攻击面[43]。例如，由于 RME-DA 的设备分配仍由宿主机虚拟机管理器控制，攻击者可以暂停设备分配，从而影响可用性。此外，如果与机密虚拟机绑定的设备本身是恶意的（如网卡设备固件被攻击者控制），仍可能发起非法 DMA 访问。最后，攻击者还可以通过观察 I/O 相关执行时间，对机密虚拟机运行状态进行行为推测，形成侧信道攻击。

因此，即使引入了 RME-DA 等安全设备访问机制，I/O 交互路径仍然可能成为机密虚拟机的重要攻击面，需要结合设备隔离、通信加密以及中断管理等综合防护措施来抵御攻击。

3.2.3 多机密虚拟机内容共享

多个机密虚拟机之间的数据交换通常依赖共享内存或宿主机管理的缓冲区，为保障数据传输安全，共享数据通常会进行加密。虽然 CCA 能保护机密虚拟机的内存不被非法访问，但如果这些内容共享通信路径经过未受保护的宿主机虚拟机管理器，则会引入潜在攻击面。例如，可能受到重放攻击（replay attack）或加密/解密预言攻击（encryption/decryption oracle attack）[59]。重放攻击可以记录某个共享内存页的密文，并在未来将其重新写回，使机密虚拟机在解密后误用旧数据。加密预言攻击可构造特定密文写入共享缓冲区，使机密虚拟机在处理时完成解密操作，攻击者通过网络交互可进一步获得任意明文。解密预言攻击则诱导机密虚拟机加密特定数据，攻击者通过分析密文推断原始数据信息。

这些攻击说明，即便内存受到硬件保护，多机密虚拟机的数据共享路径仍可能成为攻击入口。在 CCA 系统设计中，除了对共享数据加密外，还可考虑完整性验证和重放检测等机制，以进一步降低跨虚拟机数据共享带来的安全风险。

3.2.4 虚拟机上下文切换

宿主机虚拟机管理器可以控制机密虚拟机的上下文切换，例如触发 VMEXIT 或控制虚拟机进入和恢复执行的时机，这为攻击者观察执行行为提供了机会。在一些机密虚拟机系统中，攻击者可以利用系统调用中断实现虚拟机上下文切换的精细控制，例如 HECKLER[60] 和 WeSee[61] 攻击在 AMD SEV-SNP 与 Intel TDX 中构造任意代码执行链。然而，由于 CCA 不支持触发相似系统调用中断功能的指令，并且不允许虚拟机管理器观察机密虚拟机寄存器状态，这类攻击无法应用于 CCA。

尽管如此，攻击者仍可能通过行为推断获得信息。例如 SEVerESt[62] 攻击通过观察 VMEXIT 的执行频率，对机密虚拟机程序的行为模式进行识别，并通过执行特征推断程序路径或版本信息。此外，还有基于性能计数器的攻击，例如 TDXdown[63] 攻击和 CounterSEVeillance[64] 攻击。这些攻击利用性能计数器中断来控制虚拟机上下文切换或者直接计数获取虚拟机执行的指令数量，并结合程序行为模型推断执行路径或密钥信息。在 CCA 中，领域世界的性能计数器机制由平台实现决定。如果性能计数器未充分考虑隔离，攻击者则可能通过宿主机或恶意机密虚拟机来利用性能计数器推断领域世界的执行行为。

总体来看，虽然 CCA 提供了内存和寄存器隔离，但机密虚拟机的上下文切换过程仍可能成为潜在的攻击面，因此需要 CCA 软件开发者针对这类攻击进行防御。

3.2.5 其他潜在攻击面

1）其他软件攻击。CCA 系统在实际部署中还可能面临其他软件的安全威胁。例如，机密虚拟机内部操作系统漏洞。由于机密虚拟机通常运行传统操作系统（Linux 或 Windows），所以仍可能受到已知漏洞的影响。已有研究开始关注机密虚拟机内部的防御，例如 Erebor[65] 提出在机密虚拟机的内部建立安全沙盒，以隔离内部操作系统和敏感任务。

2）物理攻击。尽管 CCA 威胁模型通常不考虑物理攻击，但实际部署中仍可关注。例如，电压故障注入攻击[66] 可在安全启动阶段通过注入异常电压，使启动过程中的部分校验被跳过，从而绕过安全启动的完整性验证并加载恶意固件。BadRAM[67] 攻击通过操纵内存控制器中 DRAM 地址映射的配置，使 2 个不同的物理地址映射到同一个 DRAM 单元。这样攻击者可以在自己控制的物理内存地址访问与机密

虚拟机物理地址对应的同一 DRAM 单元数据绕过基于物理地址的内存访问控制，打破机密计算的内存隔离。

3)微架构侧信道攻击。CCA 提供内存隔离，但无法消除共享硬件资源带来的微架构攻击风险，例如 Spectre 攻击[68]、Prime+Probe[69] 或 TLB Evict[70] 等微架构侧信道攻击。Sharing[48] 提出在独立物理核心上运行机密虚拟机，以减少共享微架构资源导致的信息泄露风险。

综上所述，CCA 系统仍可能面临固件接口、I/O 路径、跨虚拟机共享、上下文切换、内部操作系统漏洞、物理攻击和微架构侧信道等多种来源的威胁。因此，在部署 CCA 系统时，可能需要考虑威胁模型之外的攻击面，以降低各种潜在攻击风险。

4 CCA 研究和应用需求展望

随着 Arm CCA 作为新一代 TEE 架构在学术界和工业界日益受到广泛关注，现有研究在隔离机制、系统构建与安全增强等方面已取得一定进展。然而，从长远发展和实际应用的角度来看，当前 CCA 体系仍面临一些挑战，尤其在安全保护、架构扩展、测评标准以及产业体系构建等方面，仍存在需要探索的空间。为此，本节在前文研究进展与关键技术分析的基础上，围绕 CCA 发展过程中的典型问题，提出若干未来的可能研究发展方向，旨在为后续相关研究提供思路启发。

4.1 加强机密计算环境的安全可管控性

目前，CCA 软件栈(如 RMM 等)仍处于持续迭代与演进过程中。尽管已有研究针对 RMM 等核心组件开展了形式化验证，以保障其功能正确性与系统可信性，但随着系统功能的不断增强与集成度的提升，单纯依赖形式化方法已难以覆盖所有潜在攻击面，尤其在多模块协同、I/O 交互路径和运行时行为等方面仍存在验证盲区。

未来的 CCA 研究可进一步引入混合安全验证机制，例如结合符号执行、模糊测试、动态污点分析与运行时行为监测等技术，对领域世界内部实现实时监控与异常检测。同时，可设计轻量级的安全监控代理或安全审计日志机制，使 TEE 在不降低隔离强度的前提下，具备一定程度的自我可测性与行为可观测性。这类机制将有助于提升 CCA 系统的安全保障能力，增强其在高安全等级应用场景中的部署价值。

4.2 拓展多域协作与分布式可信执行能力

目前，CCA 的设计目标主要聚焦于单设备内部的多世界隔离模型，尚缺乏对跨节点、跨设备环境中机密计算能力扩展的系统性支持。在人工智能、隐私保护以及可信数据空间等典型应用场景中，数据往往需要在多个安全计算节点之间进行联合推理、分布式训练或可信交换，而当前 CCA 所依赖的信任根机制尚不具备满足这一需求的能力。因此，未来 CCA 可在体系结构层面引入可信身份管理、跨节点密钥协商和安全通信与通道管理等机制，以实现机密计算环境之间的安全协同与非 Arm 架构的扩展兼容。同时，还应关注机密虚拟机上下文在异构设备之间的迁移与恢复问题，例如支持领域虚拟机的快照保存、状态同步与联合调度等关键能力，构建面向多节点的可信执行框架。此外，可信启动链和远程证明机制也需进一步扩展，以支持分布式场景中的节点接入验证与信任绑定，为构建大规模可信协同计算环境提供安全能力。

4.3 突破 CCA 硬件隔离机制的精细化能力限制

CCA 的颗粒保护检查机制为构建基于世界的隔离模型和内存物理地址空间划分提供了硬件支撑。该机制以固定粒度(最小为 4 KB)作为基本隔离单位，并通过颗粒保护表在 EL3 层实现访问控制。然而，当前机制仍存在一定局限性，例如仅支持访问和拒绝访问 2 个控制粒度，缺乏对内存访问权限中读/写的细粒度区分。这一限制在处理可信数据共享、部分页共享等复杂访问场景时，可能导致实现效率较低，甚至无法满足实际需求。为此，未来可考虑研究为 CCA 架构引入对颗粒保护表的读/写/执行权限分离支持，类似于更精细化的访问权限配置页表，以提升其在多租户环境、数据共享以及动态访问控制等场景下的适配能力与实用性。

4.4 构建统一的性能测评平台与实验基准

目前，CCA 研究面临的一个突出问题是缺乏可用的真实硬件平台。由于现阶段尚无公开可用、原生支持 CCA 的硬件设备，研究人员不得不依赖模拟器(如 FVP 或 QEMU-CCA)进行功能验证，并通过将 CCA 相关代码移植至非 CCA 的 Arm 开发板上进行性能评估。这一现状带来了研究门槛高、工作重复性强以及性能评估结果缺乏一致性等问题。针对这一挑战，已有研究者提出了 OpenCCA[71] 平台，旨在构建一个统一且开源的 CCA 性能测试与实验环境。OpenCCA 通过修改 CCA 软件栈的兼容性(包括引导加载程序、固件、虚拟化管理程序与操作系统内核)，

在商用 Armv8.2 硬件上直接运行 CCA 软件栈，为当前缺乏真实 CCA 硬件平台的问题提供了一种性能测试的替代方案。未来，可进一步探索基于真实硬件的可扩展测试平台建设，并推动产业界与学术界协同制定统一的评估基准，以促进 CCA 相关研究的标准化、工程化和实际部署落地。

4.5 推动构建自主可控的机密计算技术生态

在当前复杂的国际技术竞争格局下，构建自主可控的机密计算平台体系，已成为保障国家数据安全与算力自主的核心诉求。Arm CCA 作为新一代可信执行体系结构的重要组成，其发展路径对于我国机密计算技术生态的构建具有重要的战略意义。近年来，华为鲲鹏 TEE[72]、飞腾 TEE[73]、海光 CSV[74] 等平台已在可信计算方向上开展了深入的探索，同时 RISC-V 架构下的 TEE 体系[75] 也在逐步成型。未来，以 CCA 等基础硬件架构为依托，推动从底层硬件支持、微架构设计到系统软件栈（如 RMM、Monitor、机密虚拟机框架、运行时 API）的全链路自主实现是一个发展趋势。同时，围绕 CCA 构建标准化、模块化的国产安全计算平台，并推动其在政务云、金融云、医疗数据流通等实际应用场景中的试点部署，逐步形成从芯片到平台再到应用的闭环，从而切实提升关键场景的安全可控能力。

5 结　论

针对现有零散的 Arm CCA 研究现状，本文从架构机制、研究分类和共性技术方案 3 个层面对 CCA 进行了系统化分析，既总结了其在用户态可信执行、设备隔离与系统增强等方面的技术积累，也揭示了典型系统中潜在的共性安全设计机制与技术逻辑。通过上述系统分析，本文为理解 CCA 的整体演进脉络提供更清晰的视角，可为后续研究的开展提供参考。

作者贡献声明：张一鸣确定综述主题和论文框架，负责论文主体撰写；张锋巍提供指导和整体修改意见，并负责终审论文；闫守孟提供论文修改意见，并提供未来研究建议。

参　考　文　献

[1] National Data Administration of China. Three-Year Action Plan for Data Element (2024–2026)[EB/OL]. [2025-07-21]. https://www.cac.gov.cn/2024-01/05/c_1706119078060945.htm (in Chinese)
（国家数据局. “数据要素”三年行动计划（2024—2026 年）[EB/OL]. [2025-07-21]. https://www.cac.gov.cn/2024-01/05/c_1706119078060945.htm）

[2] Global Computing Consortium. Confidential Computing White Paper (2024)[EB/OL]. [2025-07-21]. https://www.gccorg.com/article/70/239.html (in Chinese)
（全球计算联盟. 机密计算白皮书（2024）[EB/OL]. [2025-07-21]. https://www.gccorg.com/article/70/239.html）

[3] GlobalPlatform. Introduction to trusted execution environments[EB/OL]. [2025-07-21]. https://globalplatform.org/wp-content/uploads/2018/05/Introduction-to-Trusted-Execution-Environment-15May2018.pdf

[4] Li Wei, Feng Wei, Qin Yu, et al. Scheme of runtime monitoring in confidential computing with dynamic integrity measurement[J]. Journal of Computer Research and Development, 2024, 61(10): 2482−2500 (in Chinese)
（李为, 冯伟, 秦宇, 等. 基于动态完整性度量的机密计算运行时监控方案[J]. 计算机研究与发展, 2024, 61(10): 2482−2500）

[5] Du Dongdong, Yang Bicheng, Yu Yang, et al. SegTEE: Trusted execution environment for lightweight edge devices[J]. Chinese Journal of Computers, 2025, 48(1): 188−209 (in Chinese)
（杜冬冬, 杨璧丞, 余炀, 等. SegTEE: 面向小型端侧设备的可信执行环境系统[J]. 计算机学报, 2025, 48(1): 188−209）

[6] Zheng Xianyi, Li Wen, Meng Dan. Analysis and research on 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）

[7] Arm. Introducing Arm confidential compute architecture[EB/OL]. [2025-07-21]. https://developer.arm.com/documentation/den0125/latest

[8] Zhang Yiming, Hu Yuxin, Ning Zhenyu, et al. Shelter: Extending Arm CCA with isolation in user space[C]//Proc of the 32nd USENIX Security Symp. Berkeley, CA: USENIX Association, 2023: 6257−6274

[9] Zhou Qihang, Cao Wenzhuo, Jia Xiaoqi, et al. RContainer: A secure container architecture through extending Arm CCA hardware primitives[C/OL]//Proc of the 32nd Network and Distributed System Security Symp. Washington: ISOC, 2025[2025-07-21]. https://dx.doi.org/10.14722/ndss.2025.240328

[10] 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: ISOC, 2024[2025-07-21]. https://doi.org/10.14722/ndss.2024.24763

[11] Sridhara S, Bertschi A, Schlüter B, et al. ACAI: Protecting accelerator execution with Arm confidential computing architecture[C]//Proc of the 33rd USENIX Security Symp. Berkeley, CA: USENIX Association, 2024: 3423−3440

[12] Sang Fan, Lee J, Zhang Xiaokuan, et al. PORTAL: Fast and secure device access with Arm CCA for modern Arm mobile system-on-chips (SoCs)[C]//Proc of the 46th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2025: 4099−4116

[13] Zhang Yiming, Zhang Fengwei, Luo Xiapu, et al. SCRUTINIZER: Towards secure forensics on compromised TrustZone[C/OL]//Proc of the 32nd Network and Distributed System Security Symp. Washington: ISOC, 2025[2025-07-21]. https://dx. doi.org/10.14722/ndss.2025.230147

[14] Zhang Chuqi, Zeng Jun, Zhang Yiming, et al. The HitchHiker's guide to high-assurance system observability protection with efficient permission switches[C]//Proc of the 31st ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2024: 3898–3912

[15] Zhang Fengwei, Zhou Lei, Zhang Yiming, et al. Trusted execution environment: state-of-the-art and future directions[J]. Journal of Computer Research and Development, 2024, 61(1): 243−260 (in Chinese)
(张锋巍, 周雷, 张一鸣, 等. 可信执行环境: 现状与展望[J]. 计算机研究与发展, 2024, 61(1): 243−260)

[16] AMD. AMD secure encrypted virtualization (SEV) [EB/OL]. [2026-03-10]. https://www.amd.com/en/developer/sev.html

[17] Intel. Intel trust domain extension[EB/OL]. [2026-03-10]. https://www.intel.com/content/www/us/en/developer/tools/trust-domain-extensions/overview.html

[18] Arm. TrustZone for Cortex-A[EB/OL]. [2025-07-21]. https://www.arm.com/technologies/trustzone-for-cortex-a/tee-reference-documentation

[19] Arm. Learn the architecture-realm management extension guide [EB/OL]. [2025-07-21]. https://developer.arm.com/documentation/den0126/latest/

[20] Arm. Learn the architecture —AArch64 exception model[EB/OL]. [2025-07-21].https://developer.arm.com/documentation/102412/0103/Privilege-and-Exception-levels/Exception-levels

[21] Arm. Arm CCA software architecture[EB/OL]. [2025-07-21]. https://developer.arm.com/documentation/den0125/400/Arm-CCA-Software-Architecture

[22] Arm. Realm management monitor specification[EB/OL]. [2025-07-21]. https://developer.arm.com/documentation/den0137/latest/

[23] Arm. Granule protection checks[EB/OL]. [2025-07-21]. https://developer.arm.com/documentation/den0126/0101/Granule-Protection-Checks.

[24] Trusted Firmware. Granule protection tables in TF-A[EB/OL]. [2025-07-21]. https://www.trustedfirmware.org/docs/tfa_tech_forum_2021_10_21_gpt.pdf

[25] Arm. Realm management interface[EB/OL]. [2025-07-21]. https://developer.arm.com/documentation/den0127/latest/Realm-communication-APIs/Realm-Management-Interface

[26] Arm. Arm CCA hardware architecture[EB/OL]. [2025-07-21]. https://developer.arm.com/documentation/den0125/400/Arm-CCA-Hardware-Architecture

[27] Arm. Granule protection table walk trace[EB/OL]. [2025-07-21]. https://developer.arm.com/documentation/100964/1129/Plug-ins-for-Fast-Models/TarmacTrace/Granule-protection-table-walk-trace

[28] Arm. Introducing Arm's dynamic TrustZone technology[EB/OL]. [2025-07-21]. https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/introducing-arms-dynamic-trustzone-technology

[29] Arm. The Mali GPU memory model[EB/OL]. [2025-07-21]. https://developer.arm.com/documentation/dui0538/f/opencl-concepts/the-mali-gpu-memory-model

[30] Arm. Mali graphics processing units (GPUs)[EB/OL]. [2025-07-21]. https://developer.arm.com/ip-products/graphics-and-multimedia/mali-gpus

[31] Qualcomm. Adreno graphics processing units[EB/OL]. [2025-07-21]. https://developer.qualcomm.com/software/adreno-gpu-sdk/gpu/

[32] NVIDIA. Tegra X1[EB/OL]. [2025-07-21]. https://developer.nvidia.com/content/tegra-x1/

[33] Arm. Arm realm management extension (RME) system architecture[EB/OL]. [2025-07-21]. https://developer.arm.com/documentation/den0129/latest/

[34] Arm. Arm system architecture specifications[EB/OL]. [2025-07-21]. https://developer.arm.com/documentation/den0125/0300/Device-Assignment--DA--and-Memory-Encryption-Contexts--MEC-

[35] Arm. Initializing granule protection checks[EB/OL]. [2025-07-21]. https://developer.arm.com/documentation/102754/r1p1/Software-initialization-examples/Initializing-the-SMMU/Initializing-Granule-Protection-Checks

[36] Arm. Memory encryption contexts (MEC)[EB/OL]. [2025-07-21]. https://developer.arm.com/documentation/den0125/0300/Device-Assignment--DA--and-Memory-Encryption-Contexts--MEC-

[37] Arm. Arm CCA extensions[EB/OL]. [2025-07-21]. https://developer.arm.com/documentation/den0125/400/Arm-CCA-Extensions

[38] Arm. Boot flow for Arm CCA[EB/OL]. [2025-07-21]. https://developer.arm.com/documentation/den0127/300/Arm-CCA-reference-software-stack/TF-A-for-Arm-CCA/Boot-flow-for-Arm-CCA

[39] Trusted Firmware. Chain of trust (CoT) for confidential compute architecture(CCA)[EB/OL].[2025-07-21].https://neoverse-reference-design.docs.arm.com/en/latest/shared/cot_cca.html

[40] Arm. Arm's confidential compute architecture reference attestation token[EB/OL]. [2026-03-02]. https://www.ietf.org/archive/id/draft-ffm-rats-cca-token-03.html

[41] Liu Shiqi, Gao Yongpeng, Zhang Mingyang, et al. NanoZone: Scalable, efficient, and secure memory protection for Arm CCA[J]. arXiv preprint, arXiv: 2506.07034, 2025

[42] Kuhne M, Sridhara S, Bertschi A, et al. Aster: Fixing the Android TEE ecosystem with Arm CCA[J]. arXiv preprint, arXiv: 2407.16694, 2024

[43] Bertschi A, Sridhara S, Groschupp F, et al. DEVLORE: Extending Arm CCA to integrated devices a journey beyond memory to interrupt isolation[J]. arXiv preprint, arXiv: 2408.05835, 2024

[44] Ye Zhenyu, Zhou Lei, Zhang Fengwei, et al. FortifyPatch: Towards tamper-resistant live patching in Linux-based hypervisor[C]//Proc of the 33rd ACM SIGSOFT Int Symp on Software Testing and Analysis. New York: ACM, 2024: 38–50

[45] Schulze M, Lindenmeier C, Röckl J. BarriCCAde: Isolating closed-source drivers with ARM CCA[C]//Proc of the 45th IEEE European Symp on Security and Privacy Workshops. Piscataway, NJ: IEEE, 2024: 245–251

[46] Siby S, Abdollahi S, Maheri M, et al. GuaranTEE: Towards attestable and private ML with CCA[C]//Proc of the 4th Workshop on Machine Learning and Systems. New York: ACM, 2024: 1–9

[47] Li Xupeng, Li Xuheng, Dall C, et al. Design and verification of the Arm confidential compute architecture[C]//Proc of the 16th USENIX Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2022: 465–484

[48] Castes C, Baumann A. Sharing is leaking: Blocking transient-execution attacks with core-gapped confidential VMs[C]//Proc of the 29th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2024: 267–281

[49] Chen Jiahao, Mi Zeyu, Xia Yubin, et al. CPC: Flexible, secure, and efficient CVM maintenance with confidential procedure calls[C]// Proc of the 21st USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2024: 1065−1082

[50] Chen Jiayun, Zhou Qihang, Yan Xiaolong, et al. CubeVisor: A multi-realm architecture design for running VM with ARM CCA[C]//Proc of the 40th Annual Computer Security Applications Conf. Piscataway, NJ: IEEE, 2024: 1–13

[51] Xu Xiangyi, Wang Wenhao, Wu Yongzheng, et al. virtCCA: Virtualized Arm confidential compute architecture with TrustZone[J]. arXiv preprint, arXiv: 2306.11011, 2023

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

[53] Huang Haoyang, Zhang Fengwei, Yan Shoumeng, et al. SoK: A comparison study of Arm TrustZone and CCA[C]//Proc of the 1st Int Symp on Secure and Private Execution Environment Design. Piscataway, NJ: IEEE, 2024: 107–118

[54] Arm. Build and run the Arm CCA stack on an Arm FVP[EB/OL]. [2025-07-21]. https://learn.arm.com/learning-paths/servers-and-cloud-computing/rme-cca-basics/rme-cca-fvp/

[55] Arm. FVP_Base_RevC−2xAEMvA reference stacks[EB/OL]. [2025-07-21]. https://gitlab.arm.com/aemfvp-a/docs/aemfvp-a-docs/-/blob/main/user-guide.rst

[56] Linaro. Building an RME stack for QEMU[EB/OL]. [2025-07-21]. https://linaro.atlassian.net/wiki/spaces/QEMU/pages/29051027459/Building+an+RME+stack+for+QEMU

[57] Samsung. Islet project[CP/OL]. [2025-07-21]. https://github.com/islet-project/islet

[58] Takekoshi S, Mori M, Fukai T, et al. BadAML: Exploiting legacy firmware interfaces to compromise confidential virtual machines[C]// Proc of the 32nd ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2025: 4469–4483

[59] Li Mengyuan, Zhang Yinqian, Lin Zhiqiang, et al. Exploiting unprotected I/O operations in AMD's secure encrypted virtualization[C]//Proc of the 28th USENIX Security Symp. Berkeley, CA: USENIX Association, 2019: 1257–1272

[60] Schlüter B, Sridhara S, Kuhne M, et al. HECKLER: Breaking confidential VMs with malicious interrupts[C]//Proc of the 33rd USENIX Security Symp. Berkeley, CA: USENIX Association, 2024: 3459–3476

[61] Schlüter B, Sridhara S, Bertschi A, et al. WeSee: Using malicious #VC interrupts to break AMD SEV-SNP[C]//Proc of the 45th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2024: 4220–4238

[62] Werner J, Mason J, Antonakakis M, et al. The SEVerESt Of Them All: Inference attacks against secure virtual enclaves[C]//Proc of the 14th ACM Asia Conf on Computer and Communications Security. New York: ACM, 2019: 73–85

[63] Wilke L, Sieck F, Eisenbarth T, et al. TDXdown: Single-stepping and instruction counting attacks against Intel TDX[C]//Proc of the 31st ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2024: 79–93

[64] Gast S, Weissteiner H, Schröder R, et al. CounterSEVeillance: Performance-counter attacks on AMD SEV-SNP[C/OL]//Proc of the 32nd Network and Distributed System Security Symp. Washington: ISOC, 2025[2026-03-13]. https://dx. doi.org/10.14722/ndss.2025.241038

[65] Zhang Chuqi, Priolkar R, Jiang Yuancheng, et al. Erebor: A drop-in sandbox solution for private data processing in untrusted confidential virtual machines[C]//Proc of the 20th European Conf on Computer Systems. New York: ACM, 2025: 1210–1228

[66] Buhren R, Jacob H, Krachenfels T, et al. One glitch to rule them all: Fault injection attacks against AMD's secure encrypted virtualization[C]//Proc of the 28th ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2021: 2875–2889

[67] De J, Wilke L, Oswald D, et al. BadRAM: Practical memory aliasing attacks on trusted execution environments[C]//Proc of the 46th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2025: 4117–4135

[68] Kocher P, Horn J, Fogh D, 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] Osvik A, Shamir A, Tromer E. Cache attacks and countermeasures: The case of AES[C]//Proc of the 6th Cryptographers' Track at the RSA Conf. Berlin: Springer, 2006: 1–20

[70] Gras B, Razavi K, Bos H, et al. Translation leak-aside buffer: Defeating cache side-channel protections with TLB attacks[C]//Proc of the 27th USENIX Security Symp. Berkeley, CA: USENIX Association, 2018: 955–972

[71] Bertschi A, Shinde S. OpenCCA: An open framework to enable Arm CCA research[J]. arXiv preprint, arXiv: 2506.05129, 2025

[72] Kunpeng. Confidential computing TEE suite[EB/OL]. [2025-07-21]. https://www.hikunpeng.com/document/detail/zh/kunpengcctrustzone/tee/twp/kunpengtee_19_0003.html (in Chinese)
(鲲鹏. 机密计算 TEE 套件[EB/OL]. [2025-07-21]. https://www.hikunpeng.com/document/detail/zh/kunpengcctrustzone/tee/twp/kunpengtee_19_0003.html)

[73] Phytium. Phytium Tianyi trusted execution environment[EB/OL]. [2025-07-21]. https://www.phytium.com.cn/developer/8/ (in Chinese)
(飞腾. 飞腾天衣可信执行环境[EB/OL]. [2025-07-21]. https://www.phytium.com.cn/developer/8/)

[74] Hygon. Cloud-native confidential computing[EB/OL]. [2025-07-21]. https://openanolis.cn/assets/static/confidentialComputing.pdf (in Chinese)
(海光. 云原生机密计算[EB/OL]. [2025-07-21]. https://openanolis.cn/assets/static/confidentialComputing.pdf)
[75] RISC-V. TEE SoC based on RISC-V[EB/OL]. [2025-07-21]. https://riscv.org/blog/2023/02/tee-soc-based-on-risc-v/



Zhang Yiming, born in 1994. PhD, lecturer. Member of CCF. His main research interests include confidential computing, trusted execution environments, and system security.
张一鸣, 1994 年生。博士，讲师。CCF 会员。主要研究方向为机密计算、可信执行环境、系统安全。



Zhang Fengwei, born in 1986. PhD, professor, PhD supervisor. Distinguished member of CCF. His main research interests include system security and trusted execution environments.
张锋巍, 1986 年生。博士，研究员，博士生导师。CCF 杰出会员。主要研究方向为系统安全、可信执行环境。



Yan Shoumeng, born in 1978. PhD, professor. Distinguished member of CCF. His main research interests include OS and confidential computing accelerators.
闫守孟, 1978 年生。博士，研究员。CCF 杰出会员。主要研究方向为操作系统、机密计算加速器。