计算机研究与发展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(10): 2497−2511, 2026

本期推荐

ResiCVM: 基于客内特权隔离的云集群容灾备份架构

马鸿展[1,2] 曲若非[1,2] 沈牧言[1,2] 贾梦涵[1,2] 冯登国[2] 秦 宇[2] 冯 伟[2]
[1](中国科学院大学 北京 100049)
[2](中国科学院软件研究所 北京 100190)
(mahongzhan21@mails.ucas.ac.cn)

ResiCVM: A Cloud Cluster Disaster Recovery and Backup Architecture Based on Intra-Guest Privilege Isolation

Ma Hongzhan[1,2], Qu Ruofei[1,2], Shen Muyan[1,2], Jia Menghan[1,2], Feng Dengguo[2], Qin Yu[2], and Feng Wei[2]
[1] (*Unicersity of Chinese Academy of Sciences*, *Beijing* 100049)
[2] (*Institute of Software*, *Chinese Academy of Sciences*, *Beijing* 100190)

Abstract The memory encryption mechanism in confidential computing environments renders the internal state of confidential virtual machines (CVMs) invisible to the host. While ensuring internal data security, this poses severe technical challenges for traditional hypervisor-based disaster recovery and backup solutions. Meanwhile, existing solutions relying on TEE secure processors suffer from severe performance bottlenecks, which heavily restricts the deployment of cloud confidential computing clusters and degrades their security resilience. We present ResiCVM (resilient confidential virtual machine (VM)), a resilient disaster recovery and backup framework based on intra-VM privilege isolation. ResiCVM shifts the state extraction and cryptographic protection logic from the legacy TEE secure processor to a high-privilege domain within the VM, strictly excluding the guest operating system from the trusted computing base (TCB). Additionally, we design an end-to-end cryptographic authentication protocol to ensure secure communication between the VM and the security management center during the state backup process. We implement a prototype of ResiCVM on the AMD EPYC 9745 platform. Experimental evaluations show that ResiCVM achieves a backup encapsulation and storage throughput 62.7 times higher than that of TEE secure processor-based schemes. Crucially, during concurrent multi-VM maintenance tasks, ResiCVM maintains stable execution efficiency, whereas TEE-based solutions suffer proportional performance degradation. Requiring zero host-side modifications, ResiCVM significantly improves disaster recovery performance and provides robust technical support for the secure operations and security resilience of confidential computing clusters.

Key words cloud computing; resilient confidential computing; confidential virtual machines (CVM); security resilience; virtual machine maintenance

摘 要 机密计算环境的内存加密机制使得宿主机对机密虚拟机(confidential virtual machine, CVM)的内部状态不可见,这在保证了CVM内部数据安全的同时对传统基于虚拟机监视器的容灾备份方案带来了严重的技术挑战。而使用TEE安全处理器的解决方案存在着严重的性能问题。这严重制约了云机密计算集群的部署,削弱了云机密计算节点的安全弹性。提出了ResiCVM(resilient confidential virtual machine),一个基于虚拟机客内特权级隔离的弹性容灾备份框架。通过虚拟机内部特权级隔离机制,

收稿日期: 2026-06-06; **修回日期**: 2026-07-08
基金项目: 国家重点研发计划项目(2024YFE0211100)
This work was supported by the National Key Research and Development Program of China (2024YFE0211100).
通信作者: 秦宇(qinyu@iscas.ac.cn)

ResiCVM 将客户虚拟机的状态提取与密码学保护逻辑,从原有 TEE 安全处理器上转移至虚拟机内部的高特权域中执行,同时将客户操作系统排除在可信计算基(TCB)之外。此外,为了保证虚拟机和安全管理中心之间通讯的安全性,设计了一套端到端的密码学认证通信协议,确保了虚拟机状态备份流程的安全。在 AMD EPYC 9745 平台上实现了 ResiCVM 的原型系统,实验评估表明,ResiCVM 对虚拟机状态数据的备份封装存储吞吐量达到了基于 TEE 安全处理器方案的 62.7 倍。并且在多虚拟机并发的维护任务中,ResiCVM 的执行效率并未出现显著下降;而基于 TEE 安全处理器的方案的性能则随着并发数的增加呈等比下降。ResiCVM 无需对宿主机端系统做任何修改,提高了机密计算集群的灾备性能,为机密计算集群的安全运维提供了安全弹性技术支撑。

关键词 云计算;弹性机密计算;机密虚拟机;安全弹性;虚拟机维护

中图法分类号 TP311.13; TP309

DOI: 10.7544/issn1000-1239.202660436 **CSTR:** 32373.14.issn1000-1239.202660436

随着云计算与边缘计算技术快速发展,云端计算节点已逐步取代传统的本地计算节点,被广泛用于处理高度敏感的数据。在现代云原生架构中,为了最大化资源利用率,云服务提供商(cloud service provider, CSP)通常会在同一台物理宿主机上高密度地部署多个计算节点,形成规模庞大的计算集群。然而,在云环境中,CSP 通常不在用户数据的绝对信任域内。恶意的共驻租户(co-resident tenants)、被攻破的宿主机乃至 CSP 本身,均可能对用户数据进行窃取与篡改。为应对这一挑战,机密计算(confidential computing)技术应运而生。通过依托硬件支持的可信执行环境(trusted execution environment, TEE),机密计算节点能够将虚拟机(virtual machine, VM)隔离在受保护的内存区域中,从而有效保障“使用中数据”的机密性与完整性。

尽管 TEE 提供了强大的单点安全保障,但其严格的硬件隔离机制也引入了新的系统级挑战:阻断了底层宿主机对上层节点内存与执行状态的直接访问,这使得虚拟机的内部状态对于基于宿主机操作系统的虚拟机监视器(hypervisor)处于视野盲区之中,进而导致依赖于宿主机虚拟机监视器的传统集群维护与灾备手段全面失效[1-2]。在多节点共驻的集群环境下,这种缺陷尤为致命:一旦宿主机发生硬件故障或遭遇系统级攻击,其上承载的所有机密节点均缺乏有效的状态留存与回滚手段,面临集体宕机与关键数据丢失的风险。这导致现有的机密计算集群在遭遇突发威胁时,表现出极脆弱的安全弹性(security resilience)。

为了突破上述瓶颈并赋予机密计算集群应对突发异常的能力,本文提出了一种面向海量机密计算节点的安全状态备份与恢复架构:ResiCVM(resilient confidential virtual machines)。该架构由一个可信的安全中心(security center)进行集群统一管理,在不修改底层宿主机硬件或固件的前提下,引入了高效的“安全快照(secure snapshot)”机制。通过在节点内部构建特权隔离的安全维护模块突破宿主机上虚拟机监视器对虚拟机内部状态的盲区,以对客户操作系统(Guest OS)透明的方式提取 CVM 内部的运行状态。这一设计使得当物理宿主机发生故障或集群遭遇攻击时,安全中心能够基于受保护的可信快照,实现单个节点乃至整个集群在安全环境下的快速重构与业务接管,从而为高密度部署的机密计算集群注入了强大的安全弹性。

此外,在集群规模的灾备协同中,通信通道与存储协议的安全性同样至关重要。因为机密虚拟机(confidential virtual machine, CVM)的备份要存储在不可信的 CSP 提供的存储上,同时 CSP 或其他攻击者可能会劫持通信信道。为此,我们设计了一套端到端的安全协同通信协议。通过整合远程证明(remote attestation)与轻量级密钥交换机制,系统能够确保并发快照数据在不可信网络传输过程或不可信集群存储中的机密性、完整性与新鲜性,并有效防御伪造与中间人(MITM)攻击,实现了安全中心与海量共驻节点之间的双向身份认证与信任锚定。

我们在 AMD EPYC 9745 平台上实现了 ResiCVM 的原型系统。实验结果表明,得益于将快照逻辑放入虚拟机内部透明执行与高效的中心化管理,该机制在单机多节点高并发备份、跨宿主机异常恢复等集群级压力测试中,展现出极低的性能损耗与卓越的扩展性,显著优于现有的底层硬件依赖型方案,数据封装的速度达到了基线的 62.7 倍。同时,安全分析证明其能在系统级敌手(SysAdv)的威胁模型下,

维持极高的防御基线。

综上所述，本文的主要贡献如下：

1）提出了一种由安全中心统一管理的机密计算集群安全架构，在不扩大硬件可信计算基（trusted computing base, TCB）的前提下，打破了机密计算环境下的 CVM 内部状态的盲区，有效解决了物理机多节点共驻场景下的集群容灾备份问题；

2）设计并实现了面向机密计算节点的安全快照机制，支持对节点状态的高并发提取与跨节点可信验证，使机密集群具备从重大异常中快速恢复的能力；

3）构建了保障集群大规模协同的安全通信协议，并通过全面评估证明了本架构在提升机密计算集群安全弹性（核心测量点）与保持高性能运行方面的双重优势。

1 相关工作

本节主要对 CVM 及其维护以及弹性机密计算这一概念进行介绍，并回顾了 CVM 维护这一问题的国内外研究现状。

1.1 弹性机密计算

目前应用的 TEE 面临着诸多安全威胁，主要包括各类物理攻击如总线窃听攻击[3]、错误注入攻击[4-5]，以及微架构与侧信道攻击，如瞬态执行攻击[6]、分支预测攻击[7]或受控信道攻击[8]。同时，机密计算技术在设计之初便不能应对拒绝服务攻击[9]（denial-of-service attack, DoS）。对于云上的 TEE，保护其完全不受攻击或者异常情况影响是不可能的。

弹性机密计算这一概念由冯登国院士等人[10]提出，是为了在诸多威胁下尽可能地保障 TEE 的可用性，减少隐私泄露。弹性机密计算将网络弹性（cyber resiliency）[11]这一概念引入到了机密计算中，使得机密计算环境在面对各种攻击与漏洞时仍能保持安全与可靠性。一个弹性机密计算系统应具备防篡改（tamper-resistance）、可证明性（attestablity）、机密互联（confidential interconnection）、可恢复性（recoverability）四种能力。

在上述 4 种能力中，可恢复性与可证明性是机密计算集群在遭受攻击或发生异常后，快速重构信任边界并恢复业务的核心支柱。然而，目前的 TEE 在生产实践中仍缺乏一种实用、高效且可证明的系统状态备份与回滚机制。为了真正实现机密计算的安全弹性，支持系统在异常后的快速恢复，弹性 TEE 架构必须具备 3 个维度的能力：

1）弹性 TEE 必须依托稳固且严格隔离的硬件 RoT。该 RoT 通过物理与密码学双重保护，确保核心代码与数据在运行期间免遭受任何篡改或破坏。

2）系统需具备灵活可配置的控制权接管机制。一旦检测到异常，该机制可强制将控制权安全转移至高特权可信域，以便执行全面的状态检查与灾备回滚。

3）弹性 TEE 必须实现覆盖底层硬件与上层软件的全链路远程可证明性。安全管理中心借此可实时验证系统的完整性，并在证明异常时精准触发恢复操作，从而将安全机制对正常业务的干扰降至最低。

尽管上述机制指明了弹性 TEE 的发展方向，其高效落地仍具挑战。传统方案倾向将状态提取与安全证明逻辑完全固化于底层硬件。但在 CVM 架构下，过度依赖算力受限的底层硬件不仅牺牲了管理灵活性，更直接引发了阻碍系统灾备弹性的性能瓶颈。

1.2 CVM 及其维护

CVM 是机密计算领域的核心技术，与传统的进程级 TEE（如 Intel SGX[12]）不同，CVM 将安全保护边界由某一进程拓展至整个虚拟机实例。当前主流的生产级架构（如 AMD SEV-SNP[9]、Intel TDX[13] 与 ARM CCA[14]）均通过硬件级内存加密与严格的访问控制来保障数据安全。CVMs 的内存加密密钥由集成的安全协处理器进行管理，每个 CVM 在进行内存访问时使用内存控制器中加载的唯一临时密钥对内存进行加解密。这种机制确保了诸如 Hypervisor 或 CSP 等不可信实体只能观测到内存的密文，且几乎不带来性能损耗。此外，现代 CVM 在退出至 Host 时会对虚拟机控制块（VMCB）进行加密，以防止宿主机篡改寄存器状态。

相比于进程级 TEE，CVM 支持在 TEE 中运行未经修改的完整 Guest OS，这使得未经修改的原生软件能够直接被迁移到 TEE 中。但是这不可避免地带来了 TCB 的急剧膨胀，引入了诸如特权级攻击等严重的软件层漏洞[15-16]，以及 Linux 内核本身具有的大量漏洞[17-18]。为了应对这一问题，在 CVM 中进行二次隔离，在 CVM 内部构建一个具有较小 TCB 的安全服务模块来为安全敏感业务提供最高等级的安全保障，成为了目前的研究思路[19]。这种核心安全功能的特权隔离机制确保了即使 Guest OS 被完全攻破，系统底层的安全底线依然坚固。例如，AMD SEV-SNP 引入的虚拟机特权级（VMPL）、Intel TDX 支持的分区飞地（partitioning enclaves）以及 ARM CCA 提供的机密领域（realms）权限管控，都是该机制在工业界前沿

实践中的典型代表。不同 TEE 架构的特性对比如表 1 所示。

Table 1 Comparison of Different TEE Architecture Features

表 1 不同 TEE 架构特性对比

架构	保护粒度	原生软件支持	内部隔离
SGX	进程级	×	×
TrustZone	系统级	×	×
SEV-SNP	虚拟机	√	VMPL
TDX	虚拟机	√	TD 分区
CCA	Realm	√	RMM

CVM 面对的另一个问题就是维护手段的缺乏。在云原生集群中, 虚拟机的实时迁移与状态快照等维护任务对于负载均衡与灾难恢复至关重要, 是保障集群安全弹性的基础[20]。针对此问题, 现有研究多倾向于对 TEE 底层固件进行侵入式修改[21-22], 或者增加新的可信硬件[23-24]。这不仅限制了异构集群的部署灵活性, 而且在面对单机多节点的密集型部署时, 会导致平台的 TCB 急剧膨胀, 反而放大了集群的攻击面与管理开销。另一方面, 依靠 CVM 安全处理器的原生数据封装/维护方案则面临着严重的效率问题, 比如 AMD SEV-SNP 虚拟机依靠嵌入式 AMD 安全处理器(AMD secure processor, ASP)的原生方案。该方案依赖 ASP 对 CVM 的内存进行加解密, 因为 ASP 的性能问题, 其相比与传统维护方案有着极严重的性能下降问题[25]。我们在 AMD EPYC 9745 平台上的评估表明, 原生基于 ASP 的 CVM 迁移速度仅为标准 KVM 的 1/500, 其密码学页面封装的吞吐量上限仅约为 1.9 MBps, 如图 1 所示。更为严重的是, 由于 ASP 采用单线程串行执行机制, 在处理物理机上多个 CVM 实例并发维护任务时, 会引发灾难性的性能骤降。这种极端的低效性完全无法满足生产级云集群应对突发灾备的需求。各方案的横向对比如表 2 所示。其中, “可信数据面依赖”表示快照或迁移过程中是否需要由安全处理器、TEE 固件或第三方可信组件直接参与大规模状态数据的处理。

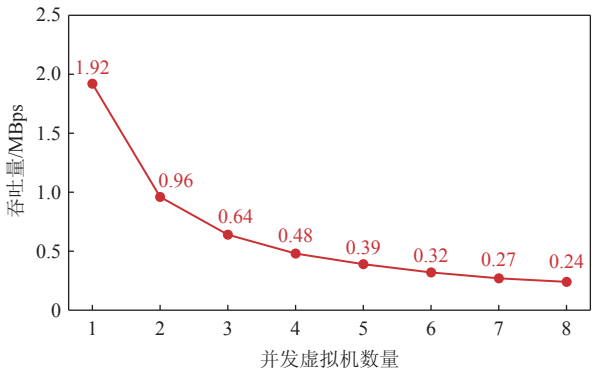


Fig. 1 Performance degradation trend of concurrent snapshots under the legacy ASP architecture

图 1 传统 ASP 架构下并发快照的性能衰减趋势

Table 2 Comparison of Maintenance Mechanisms of ResiCVM and Existing CVM

表 2 ResiCVM 与已有 CVM 维护机制的对比

维护方案	硬件/固件修改	可信数据面依赖	TCB 大小	维护效率
ASP	×	√	小	低
VirTEE[7]	√	√	小	高
TEEnder[6]	×	√	大	高
Guest OS	×	×	大	高
ResiCVM	×	×	小	高

由表 2 可以看出, 现有 CVM 维护方案通常难以同时满足“无需平台修改”“小 TCB”和“高维护效率”3 个目标。因此, 探索一种无需修改底层硬件、保持较小 TCB 且支持高并发的高效灾备架构, 已成为实现机密计算集群安全弹性的迫切需求。

2 威胁模型

为了严格评估所提出框架的安全性, 本节概述了敌手假设、TCB 的范围, 以及 ResiCVM 需要满足的核心安全目标。

2.1 威胁模型

遵循以往基于 TEE 的虚拟化研究中确立的安全模型[26-28], 我们假设存在一个 SysAdv, 其完全控制了宿主机的软件栈, 包括 Hypervisor 和 CSP 的管理基础设施。与这些既定的威胁模型一致, 我们假设敌手无法破坏 TEE 平台底层的硬件安全机制。我们进一步假设 CVM 实例与安全管理中心之间的外部通信通道是不可信的。根据该领域的标准网络威胁模型, 敌手能够窃听、拦截或恶意篡改网络上传输的任何数据。具体而言, SysAdv 可以访问并篡改驻留在磁盘、网络或非私有内存区域中的数据, 这使得敌手能够发起以下攻击。

1)身份伪造与中间人攻击: 敌手可能尝试伪造来自安全管理中心的维护指令, 或伪装成合法的 CVM 实例, 以劫持数据备份与恢复过程;

2)重放与回滚攻击: 我们假设敌手对磁盘和网络 I/O 具有完全的控制权。因此, 他们可能会在恢复阶段提供过时的快照, 或重放备份数据流, 试图对虚

拟机状态发起“分叉攻击(forking attack)”；

3)其他威胁：与机密计算领域内的标准安全假设一致，DoS 攻击、侧信道攻击(例如时序或缓存访问模式分析)以及物理攻击超出了本文的讨论范围。尽管这些攻击对 TEE 构成了重大威胁，但它们通常需要通过正交的硬件级或微架构级缓解措施来解决。

ResiCVM 的 TCB 包括以下内容：1)TEE 平台硬件及其固件，本文假设其安全且未经篡改；2)在 CVM 内部高特权隔离域中执行的 ResiCVM 核心代码；3)安全管理中心及其私钥。

除上述组件外，宿主机软件栈、Hypervisor、CSP 管理基础设施、Guest OS 以及外部通信通道均不属于 ResiCVM 的 TCB。系统采用标准密码学原语，其中认证加密算法采用 AES-GCM，散列算法采用 SHA-512，签名算法采用 Ed25519，并假设上述原语满足标准安全性要求，包括认证加密的机密性与完整性、数字签名的不可伪造性以及哈希函数的抗碰撞性。本文假设 CVM 与安全管理中心之间不存在安全的通信通道。

需要特别说明的是，Guest Kernel 中的代理模块也不属于 ResiCVM 的 TCB。该模块仅作为不可信的控制消息转发组件和特权级切换触发组件存在，其行为可以被敌手任意控制，包括丢弃、延迟、篡改、重放或伪造控制消息。ResiCVM 不依赖该代理模块的正确性来保证安全性，所有控制指令的合法性验证、密钥管理、远程证明、快照完整性校验和恢复提交均由 CVM 内部高特权隔离域中的 ResiCVM 核心模块完成。代理模块被攻破后最多导致拒绝服务、维护流程延迟或无效请求触发，拒绝服务类影响不属于本文安全目标范围，可通过正交的可用性保障机制进行缓解。

2.2 安全目标

ResiCVM 的设计旨在实现以下安全目标：

1)机密性与完整性(G1)。在整个维护生命周期内，绝不能泄露 CVM 内部的任何敏感数据。任何对快照或恢复数据未经授权的篡改，都必须能被 CVM 或安全管理中心通过密码学校验严格检测出。

2)状态新鲜度(G2)。系统必须能有效防御重放与回滚攻击，确保 CVM 不会被恢复至废弃的历史状态，或由过时的快照进行初始化。

3)快照与恢复的原子性(G3)。ResiCVM 必须确保状态恢复过程具备原子性特征，以防止虚拟机分叉(即克隆体的并发执行)，并允许客户机在遭遇非法指令或操作被中断时能够安全地中止流程。

4)双向身份认证(G4)。系统必须在 CVM 实例与安全管理中心之间强制执行健壮的双向身份认证机制，以彻底杜绝未经授权的指令注入或身份伪造行为。

3 协议设计

本节将详细介绍 ResiCVM 所采用的密码学协议，以确保 CVM 灾备份与恢复操作的安全性。第 3.1 节描述了内部密钥层级结构以及将 ResiCVM 身份锚定在底层硬件中的远程证明机制。第 3.2 节阐述了 ResiCVM 与安全中心之间执行的密钥交换过程，以建立用于数据传输的安全会话。

3.1 密钥设计与远程证明

既往研究表明，网络栈中存在着大量漏洞并被多种攻击方式所威胁[29]。在 TEE 中部署网络协议栈不仅会使得 TCB 显著膨胀，更会增加隐私信息泄露的风险。因此，ResiCVM 没有搭载网络模块。作为替代，ResiCVM 利用 Guest OS 或宿主机中的网络模块与外界进行通信。需要强调的是，ResiCVM 并不信任这些处于不可信环境中的网络代理，攻击者可以完全控制这些网络代理对快照提取或状态恢复过程发起攻击。为了缓解这一威胁，我们提出了一种锚定在硬件信任根(root of trust, RoT)上的健壮密钥层级结构与通信协议。图 2 展示了这些密钥之间的

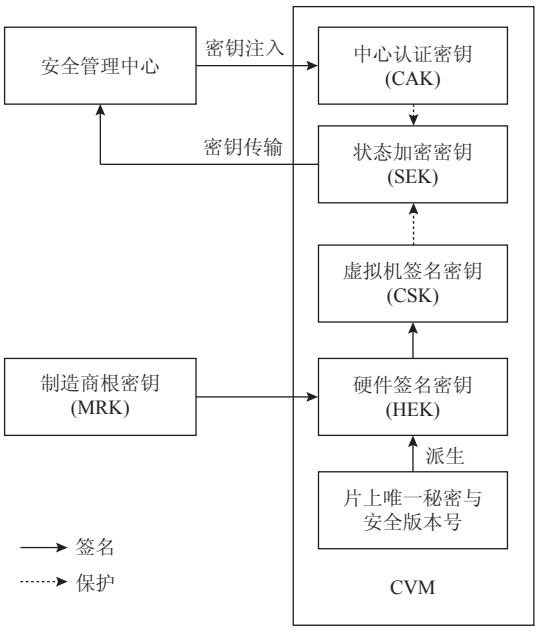


Fig. 2 Chain of trust and key derivation model in ResiCVM architecture

图 2 ResiCVM 架构下的信任链与密钥派生模型

依赖关系。

1)硬件信任根。我们利用 TEE 的底层硬件作为整个信任模型的基础。在平台启动时,底层可信硬件被初始化并加载安全固件。该固件会利用芯片在制造阶段硬编码的唯一物理 ID 与当前的安全版本号(TCB version),派生出唯一的硬件背书密钥(hardware endorsement key, HEK)私钥。HEK 作为硬件级远程证明报告的签名密钥,其对应的公钥证书由可信的硬件厂商或者可信的第三方证书签发者进行签名,可连同完整的证书链一起从可信的密钥管理服务器获取,用于验证底层硬件的真实性。

2)CVM 签名密钥(CVM signing key, CSK)。CSK 是每次 ResiCVM 模块初始化时,随机生成的一对临时非对称密钥对。此密钥会在 ResiCVM 模块初始化时重新生成,在虚拟机关机后失效。为了向安全管理中心证明某个特定的 CSK 确实属于一个合法的 ResiCVM 实例,底层的 HEK 会包含对 CSK 公钥的远程证明。

3)报告进行硬件签名。这种密码学绑定有效防止了敌手伪造实例身份。该证明报告在标准的 TEE 硬件报告基础上,进行了如下的特定的约束与扩展。

①特权级(privilege level)断言:严格限制该报告必须由运行在最高特权级(如完全隔离域内)的 ResiCVM 模块触发生成,确保其不是来自不可信 Guest OS 中恶意程序的伪造。

②身份绑定:将 CSK 公钥嵌入报告的用户自定义数据字段(User Data)中,实现当前维护会话身份与底层硬件信任根的绑定。

③防重放随机数(Nonce):包含由安全管理中心或交互方提供的随机挑战码,同样嵌入 User Data 字段,用于抵御重放攻击。

④度量值(Measurement):代表 ResiCVM 二进制代码的密码学哈希值。安全管理中心或目标节点仅允许与度量值匹配(即代码未被恶意篡改)的合法实例进行快照或恢复交互。

⑤硬件签名(Signature):由 HEK 生成的硬件级签名,可通过硬件厂商的证书链进行合法性校验以验证报告的真实性。

⑥状态加密密钥(state encryption key, SEK):在建立上述信任基础之后,ResiCVM 在每次发起快照或恢复会话之初,都会生成一个 SEK。SEK 是一个对称加密密钥,专门用于保护传输与静态存储过程中虚拟机内部状态数据的机密性与完整性。通过为每个独立的容灾恢复会话生成全新的 SEK, ResiCVM 保证了不同容灾维护会话之间的密钥独立性,避免单次会话密钥泄露影响其他历史或后续会话,并确保该过程能够有效抵御针对备份数据流的重放攻击或未经授权的历史状态回滚。关于 SEK 的具体密钥交换协议将在第 3.2 节中详细阐述。

3.2 密钥交换协议

在 CVM 启动后, ResiCVM 模块会在 Guest OS 启动之前完成初始化。在初始化过程中, ResiCVM 模块会生成一对非对称密钥 CSK, 此 CSK 在每次运行时保持不变,但每次初始化时生成的 CSK 是随机的。

CVM 的灾备工作流由安全管理中心发起,包含“快照提取”与“状态恢复”2 种场景。每次快照提取流程 ResiCVM 模块都会生成一个新的 SEK,并将此 SEK 传输给安全管理中心。之后 ResiCVM 将使用此 SEK 加密自身状态并将加密后的快照储存在 Host 的物理存储中。而在状态恢复流程中, ResiCVM 模块则需要从安全管理中心安全地获取对应的历史 SEK,以便解密并恢复宿主机中的快照数据。

CVM 与安全管理中心之间的密钥交换可以分为 2 个不同的过程:远程认证过程和密钥交换过程。远程认证过程负责 CVM 和安全管理中心确认双方身份,密钥交换过程则为 ResiCVM 将 SEK 传输给安全管理中心。需要强调的是,该协议假定虚拟机的可用性:虽然恶意虚拟机管理程序可能会丢弃数据包或者直接关闭信道以干扰快照或恢复过程,但这种拒绝服务攻击不会损害客户机数据的机密性或完整性,因此不在我们的安全考虑范围之内。该密钥交换协议的安全性分析见第 6.2 节。

3.2.1 双向身份认证(阶段 1)

此阶段的主要目标是确保安全管理中心与 CVM 能够相互验证彼此的合法身份。具体而言,需要确认快照指令确实来源于受信任的管理中心,同时管理中心也需确认对端是一个运行在 VMPL 0 特权级且未被篡改的合法 ResiCVM 实例。该阶段的流程在图 3 中给出,具体流程如下。

1)安全管理中心发起随机数挑战。安全管理中心在发起快照流程时,首先生成一个 32 位的随机数 N_{sc},并使用中心认证密钥(CAK)私钥对该挑战请求进行数字签名,随后将其发送给目标 CVM。该机制旨在防止敌手(如恶意的宿主机平台)重放过时的操作指令。

2)CVM 进行远程证明。收到请求后, ResiCVM 模块提取被编码在虚拟机初始镜像中或者由启动

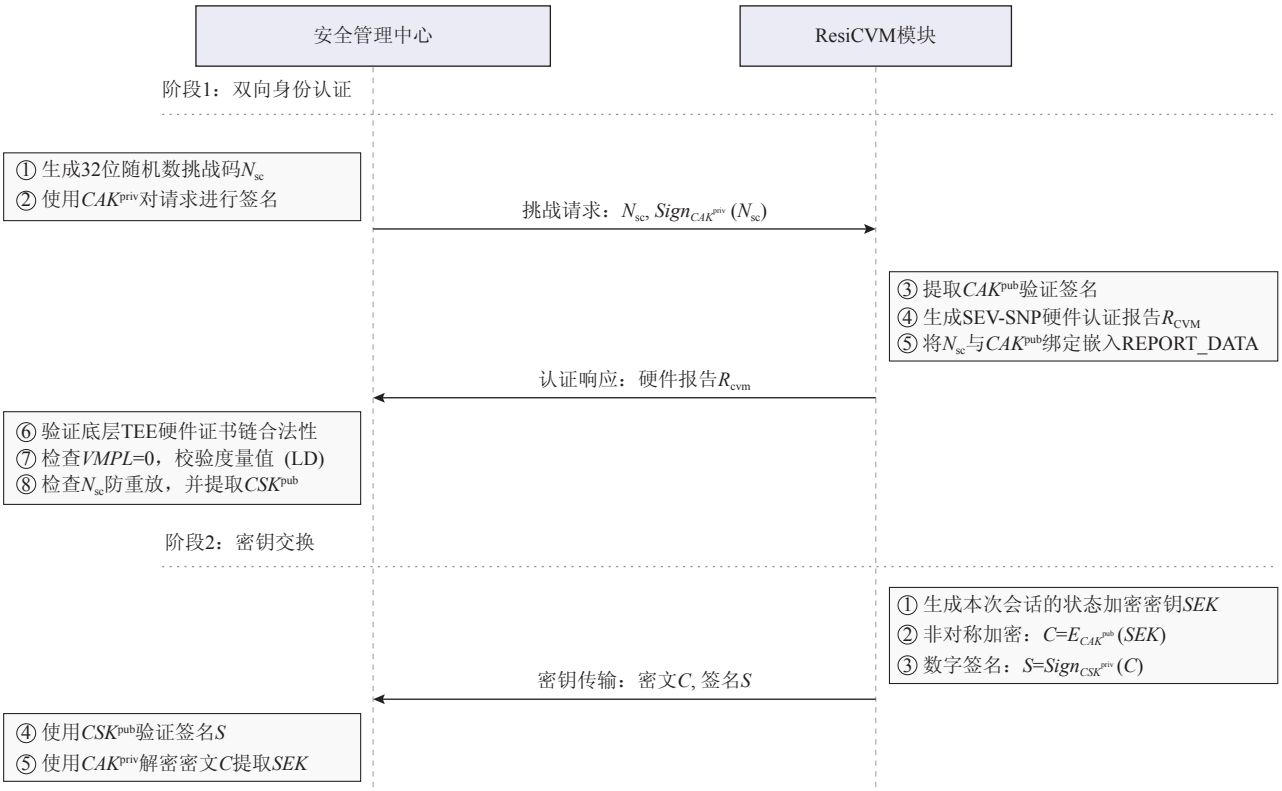


Fig. 3 Mutual identity attestation and key exchange protocol between ResiCVM and the security management center

图 3 ResiCVM 与安全管理中心之间的双向身份认证与密钥交换协议

参数注入到虚拟机中的 CAK 公钥，对接收到的请求签名进行验证。一旦确认安全管理中心的身份合法，ResiCVM 会调用底层的 SEV-SNP 硬件指令生成一份认证报告R_{CVM}。在此过程中，ResiCVM 会将接收到的挑战码N_{sc}以及自身在初始化阶段生成的非对称密钥公钥CSK^{pub}绑定并嵌入到该报告的 REPORT_DATA 字段中，随后将R_{CVM}发送回安全管理中心。

3）安全管理中心远程验证。安全管理中心收到R_{CVM}后，利用从中提取的唯一芯片标识 CHIP_ID 与 TCB 元数据拉取 TEE 制造商或证明服务中心的证书链以验证底层硬件签名的合法性。若硬件签名验证失败，管理中心将立即中止快照会话。在签名验证通过后，安全管理中心会严格执行以下 3 项关键的安全检查：特权级检查，确保硬件报告是由高特权域的 ResiCVM 模块发出的，而非来自已被攻破的 Guest OS 或其他低特权级应用（步骤⑥）；完整性检查，校验报告中的启动度量值（LD）是否与标准 ResiCVM 镜像度量值完全一致，以此保证运行时的软件版本未遭到篡改（步骤⑦）；新鲜性检查，检查硬件报告中提取出的数据位是否与步骤①中生成的随机数N_{sc}完全匹配，从而防御针对状态报告的重放攻击（步骤⑧）。

3.2.2 密钥交换（阶段 2）

经过阶段 1 的验证，安全管理中心已经成功获取并确认了可信 CVM 的 CSK 公钥CSK^{pub}。在此建立的互信基础上，双方开始进行状态加密密钥SEK的安全交换。

1）CVM 生成SEK。ResiCVM 模块在其高特权级环境中，生成一个全新的、专门用于本次快照会话的对称加密密钥SEK。

2）加密与签名。为了确保SEK在网络传输中的机密性与可认证性，ResiCVM 首先使用预先硬编码在镜像中的CAK^{pub}对SEK进行非对称加密，生成密文 $E_{CAK^{pub}}(SEK)$。随后，ResiCVM 利用自身持有的CSK^{priv}对该密文进行数字签名，并将附带签名的数据包传输给安全管理中心。

3）验证与解密。安全管理中心收到数据包后，首先利用在阶段 1 中获取的CSK^{pub}验证该签名。若签名验证通过，即可证明该数据包未被篡改，且确实来源于合法的 ResiCVM 实例。最后，安全管理中心使用其高度保密的 CAK 私钥CAK^{priv}对有效载荷进行解密，从而安全地提取出SEK。引入 CSK 的必要性在于在大多数 TEE 架构中，硬件签名密钥 HEK 会被严格管理，为避免泄露，大多数时候只能用于签名

特定的证明报告;客户软件不能使用 HEK 任意的签名或加密数据。通过将CSK^{pub}嵌入到硬件签名的报告中,我们将临时CSK与 CVM 的硬件支持的身份验证进行绑定。这使得 ResiCVM 可以使用CSK进行灵活的签名和加密操作(例如保护 SEK交换),同时保持对中间人攻击和选择明文攻击的抵御能力。

4 系统实现

我们基于 AMD EPYC 9745 平台实现了 ResiCVM 的原型系统。Guest 与 Host 的 Linux 内核版本均为 6.12。本节会对 ResiCVM 主要组件的实现和 ResiCVM 的工作流进行说明。

4.1 系统结构

ResiCVM 的系统结构如图 4 所示,主要分为虚拟机监视器、运行时组件、Guest OS 模块 3 个部分,下面逐一对其进行说明。

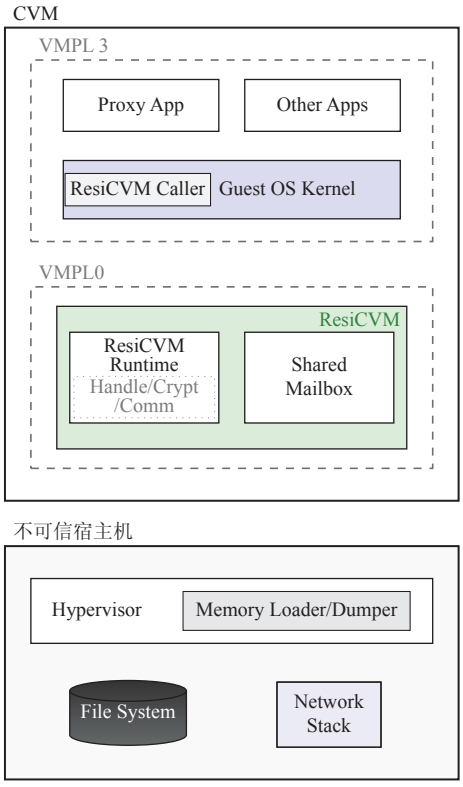


Fig. 4 System architecture of ResiCVM

图 4 ResiCVM 系统结构

1)虚拟机监视器。在传统的基于 Hypervisor 的管理范式中,虚拟机快照或灾备操作通常通过 Hypervisor 发起,并由 KVM ioctl 调用执行。遵循这一既往范式,我们在 QEMU 中集成了 2 个核心功能模块:状态导出器(state dumper)与状态加载器(state loader)。为了协调宿主机与客户机内部的交互,我们实现了一个基于共享信箱(Shared Mailbox)机制的通信桥梁,用于持续监听请求并将其分发至对应的内存管理模块。具体而言,宿主机通过修改共享内存中的状态位来触发交互;同时,状态导出器和加载器仅负责处理与持久化存储 ResiCVM 生成的加密数据,在整个流程中严格保持与虚拟机明文状态的物理隔离。

此外,安全管理中心可以选择在制作镜像时直接将 CAK 的公钥硬编码到虚拟机镜像中,或者选择使用 QEMU 命令行在启动时进行注入,相关修改也在 QEMU 中进行了实现,对 QEMU 的上述修改共包含约 600 行代码。

2)运行时组件。ResiCVM 的核心组件运行在虚拟机内部的高特权级区域(VMPL 0)中。我们在该层实现了约 1 600 行代码,并将其划分为以下 4 个功能组件。

①通信模块(Comm)。为了确保系统在不同环境下的广泛兼容性,我们的设计避免了对宿主机内核或原生 KVM 模块的修改。与其引入会产生显著性能开销的全新 VMEXIT 与 Hypercall,我们在 CVM 与宿主机之间通过预分配的共享内存区域建立了一个轻量级的通信通道。这种设计不仅提升了 ResiCVM 框架的跨平台可移植性,还大幅降低了频繁触发 VMEXIT 导致的上下文切换开销,从而显著提升了灾备任务的整体运行效率。在初始化阶段,ResiCVM 会分配一个专门的共享内存页。该内存页的物理地址在 CVM 的整个生命周期内保持静态不变,使得宿主机侧 QEMU 能够通过轮询共享信箱中的状态字段与 ResiCVM 进行同步。该确定性的同步通信模型降低了异步系统中的竞争条件,在保障操作原子性的同时,大幅简化了系统安全状态机的实现。

②密码学模块(Crypt)。为了支持安全的密钥交换与状态页面的加密,我们在 ResiCVM 内集成了一个密码学模块。该模块提供了数字签名、验证、加密/解密以及密钥派生等基础原语,还实现了针对硬件签名报告的获取、组装与验证接口,以支持端到端的远程证明过程。该模块是 ResiCVM 密钥交换、指令认证和快照完整性保护的核心安全组件。

③业务处理模块(Handle)。业务处理模块负责管理对虚拟机内存的处理流程,并严格保障快照与恢复等生命周期操作的原子性。此外,对于 AMD SEV-SNP 架构,提取 CVM 的全量状态需要将客户机的物理地址空间映射到 VMPL 0 的内核中。如果采

用标准的 4 KB 页面进行映射，会导致页表项激增，在处理大容量内存实例时极易引发页表溢出。为了缓解这一问题，我们实现了对 2 MB 大页（Huge Page）的支持。这不仅降低了映射粒度、减轻了 TLB（转换后备缓冲区）的压力，还确保了整个大容量状态快照过程的稳定性。

④内核模块。我们在客户操作系统内核（Guest OS kernel）内部开发了一个轻量级的代理模块（约 300 行代码）。该代理模块仅负责监听来自安全管理中心的快照或恢复控制消息，并将其作为不可信输入转发至 VMPL 0 中的 ResiCVM 核心模块。所有控制消息的合法性、完整性和新鲜性均由 ResiCVM 核心模块独立验证；代理模块不保存 CSK、SEK 等敏感密钥，也不能访问 VMPL 0 私有内存。因此，代理模块仅位于不可信控制路径上，不扩大 ResiCVM 的 TCB。

此外，该代理模块实现了触发虚拟机内部特权级切换的必要逻辑，用于将控制流从 Guest OS 内核转移至 ResiCVM 核心模块。我们还在 CPU 核心处于高特权级状态时暂停 Guest OS 内核对该 CPU 的调度，以防止 CPU 状态被抢占和时钟状态不一致等问题。

4.2 关键数据结构与核心接口

为进一步说明 ResiCVM 原型系统的核心实现细节，本节对系统中的关键数据结构和主要接口进行说明。ResiCVM 的实现主要围绕一个核心数据结构——内存头（MemoryHeader）展开。MemoryHeader 处于 CVM 和 Host 的 Shared Mailbox 中。表 3 给出了 MemoryHeader 的主要字段及其作用。该结构体以 4 KB 页大小对齐，使其能够作为一个独立的硬件页在私有页与共享页之间进行状态转换。其所在的共享页在系统启动阶段完成分配和初始化，而其中与具体快照块相关的字段会在每个数据块处理前由 ResiCVM 重新填充和更新。

Table 3 Description of Key Fields in MemoryHeader

表 3 MemoryHeader 的关键字段说明

字段类别	字段	作用
标识字段	magic_num version	标识合法结构体和协议版本
控制字段	flag is_finish	flag 表示当前块处理状态，is_finish 表示当前块是否为最终块
会话字段	session_id, block_order, total_blocks	标识会话、当前数据块序号和预期块总数
内存映射字段	num_pages, data_paddr, src_paddr	描述当前数据块的内存映射信息
认证加密字段	nonce tag	AES-GCM 使用的初始化向量和认证标签
密钥与身份字段	pub_key	当前实例绑定的 CSK 公钥
哈希链字段	header_hash, final_data_hash, final_header_hash	数据块和元数据的累积摘要，用于完整性校验。
签名字段	signature_data, signature_ header	保存对最终摘要的签名认证信息，用于验证快照数据来源

需要强调的是，MemoryHeader 中不保存虚拟机明文状态，也不保存 SEK、CSK 私钥等敏感密钥。QEMU 只能观测到控制字段、密文块描述信息和认证元数据，而不能访问 VMPL 0 私有内存中的密钥和明文页面。

围绕 MemoryHeader，ResiCVM 在 QEMU 和 Guest OS 内核模块中实现了若干核心接口。表 4 给出了这些接口的主要功能。需要说明的是，QEMU 侧接口仅负责对指定客户机物理地址（guest physical address, GPA）对应的共享缓冲区进行读写和持久化管理，不能直接访问 CVM 的明文私有内存。明文状态的提取、加密封装、完整性校验和恢复提交均由 VMPL 0 中的 ResiCVM 核心模块完成。

Table 4 Core Interfaces Description of ResiCVM

表 4 ResiCVM 核心接口说明

模块	接口/功能	说明
QEMU	数据转储接口	根据内存映射字段，读取指定 GPA 对应的共享密文缓冲区，并将其写入持久化存储。
QEMU	数据写入接口	从持久化存储读取加密块，并写入共享缓冲区。
QEMU	监听接口	轮询 flag 位
Guest OS 内核模块	设备接口	挂载用于接收维护指令的字符设备
Guest OS 内核模块	调度暂停接口	暂停当前 CPU 上 Guest Kernel 的调度
Guest OS 内核模块	VMPL 切换接口	将当前 CPU 的执行流转移至 VMPL 0 中的 ResiCVM 核心模块

4.3 快照与恢复的运行工作流

CVM 的容灾恢复工作流分为 4 个主要阶段：1）双向身份认证与密钥交换；2）内存页封装与归档；3）内存页解封与加载；4）恢复执行。快照生成包含前 2 个阶段，而状态恢复则包含第 1 阶段以及后 2 个阶段。由于第 1 节已经详细描述了双向身份认证与密钥交换的密码学协议，本节将重点介绍状态封装和解封的具体实现。

1）内存页封装

在密钥协商流程结束后，ResiCVM 通过 MemoryHeader 中的状态字段通知 QEMU 侧监听接口启动快照流程，并在高特权级内存中分配一块连续的共享缓冲区。ResiCVM 会将客户机内存划分为离散的数据块进行处理，数据块的大小受限于可用的共享内存容量。在我们的原型实现中，为了在稳定性与封装吞吐量之间取得最佳平衡，我们将数据块大小设置为 4 MB（包含 2 个 2 MB 的大页）。

在处理每个数据块的伊始，ResiCVM 会定位并

填充 MemoryHeader 结构。承载 MemoryHeader 的内存页最初被配置为客户机私有页,以确保在初始化期间宿主机无法访问。随后,ResiCVM 以 2 MB 的粒度遍历 Guest OS 的内存空间,使用 SEK 对页面进行加密。生成的密文随后被提交至预先分配的共享内存区域中。为了保护数据完整性,ResiCVM 采用了一种双哈希链机制。为每个加密块生成的认证标签会被迭代更新到 data_hasher 中,而 MemoryHeader 本身的累积哈希值则记录在 header_hash 字段中,并同步更新到 header_hasher 中。在完成头部更新后,ResiCVM 会将 MemoryHeader 的页面属性翻转为“共享”,并通知宿主机将该内存块存储至物理存储中。当所有内存页都处理完毕后,最后一个数据块的 MemoryHeader 将包含 2 个哈希值的最终计算值。ResiCVM 会使用 CSK 的私钥对这些哈希值进行签名,以保证其真实性与可认证性。最后,系统会设置一个终止标志,以通知宿主机封装序列已完成。

为了缓解重放攻击并确保 AES 加密的密码学健壮性,我们在每个 MemoryHeader 中引入了唯一的会话 ID(session ID)和单调递增的块索引(block index)。关于该机制的正式安全分析将在第 6 节中提供。

2)内存页解封与恢复

内存页解封是封装的逆过程,在容灾恢复阶段被调用。该过程使用宿主机物理存储提供的加密快照数据块来恢复 CVM 的内存状态。这一过程确保只有经过认证、保持新鲜度且未被篡改的数据才会被写回客户机的私有内存中。

在每个恢复周期的开始,宿主机将一个加密的内存块及其对应的 MemoryHeader 放入共享内存区域。ResiCVM 会对这些数据块进行迭代处理。对于每个数据块,它使用已配置的 SEK 通过 AES-GCM 算法解密有效载荷,并更新内部的 data_hasher 和 header_hasher。同时,ResiCVM 会严格校验会话 ID 和块索引,以保持正确的处理顺序,并防止块级别的重放或重排攻击。

为了优化性能,ResiCVM 避免了为每个独立数据块验证数字签名所带来的高昂开销。相反,密码学签名的验证被推迟到整个快照数据处理完毕之后进行。在接收并解密最后一个数据块后,ResiCVM 会使用 CSK 公钥执行一次单一的签名验证。这一仅有一次的检查验证了 2 个哈希器的最终值,从而整体上确保了整个恢复内存状态的完整性与真实性。

一旦签名验证通过,ResiCVM 会完成恢复流程,安全清除 VMPL 0 寄存器中残留的任何敏感数据,并触发特权级切换,使 Guest OS 从快照时的精确位置恢复执行。如果在最后一个数据块的签名验证失败,ResiCVM 会立即中止恢复过程并停机,严格防止已被破坏或篡改的 CVM 状态被执行。

4.4 对其他架构的适配性

ResiCVM 当前原型基于 AMD SEV-SNP 的 VMPL/SVSM 实现,但其核心机制并不绑定于某一特定硬件平台。ResiCVM 依赖的是“CVM 内部高特权服务域”这一抽象能力,即在不信任 Guest OS 的前提下,在 CVM 内部提供一个可被硬件证明、具备更高特权级且能够访问必要状态的受保护执行域。

从实现角度看,ResiCVM 可划分为平台无关逻辑和平台相关适配层 2 部分。平台无关逻辑包括快照状态机、密钥管理、状态加密、哈希链完整性校验和最终签名验证等;平台相关适配层主要包括远程证明、私有页/共享页转换、特权域进入/退出以及底层内存访问接口。因此,在迁移到其他 CVM 架构时,ResiCVM 的核心协议和快照封装逻辑可以复用,只要替换平台适配代码。

对于 Intel TDX, ResiCVM 可将 VMPL/SVSM 相关接口替换为 TDX 平台上的 TD 内部隔离域、TDREPORT/Quote 远程证明机制以及 TD 私有页/共享页转换接口。对于 ARM CCA, ResiCVM 可利用 Realm/RMM 提供的隔离、证明和内存管理能力,将维护逻辑部署为 Realm 内部受保护服务。由于 TDX 和 ARM CCA 在证明格式、特权模型和内存管理接口上与 SEV-SNP 存在差异,本文当前原型尚未实现对应版本,相关工程适配将作为后续工作。

5 实验评估

本节通过一系列基准测试,评估了 ResiCVM 的性能、效率和可扩展性。我们的评估重点在于内存状态封装的吞吐量、全量备份与恢复的总耗时,以及维护操作对在线服务性能的影响。通过将 ResiCVM 与现有的原生硬件解决方案以及标准的软件框架进行对比,我们证明了其能够克服当前 CVM 容灾恢复范式中固有的性能瓶颈。

5.1 实验环境

我们在配备 AMD EPYC 9745 处理器的服务器上实现并评估了 ResiCVM 原型系统。该平台提供 256 个逻辑核心和 128 GB 内存,并使用 4 TB HDD 用于持久化存储虚拟机快照。硬件级安全特性由 AMD SEV-SNP 架构提供支持。用于所有快照和恢复基准

测试的目标 CVM 配置为 1 个 vCPU 和 2 GB 内存。

需要说明的是，安全管理中心主要承担控制平面功能，负责发起快照/恢复指令、完成远程证明、维护 SEK 等密钥元数据，而不直接转发完整的虚拟机内存快照。快照数据平面由 CVM 内部的 ResiCVM 模块完成加密封装后，直接写入云平台提供的云平台本地或近端存储后端。因此，在同一云平台本地备份与本地恢复场景下，网络通信主要由控制指令、远程证明报告、密钥交换消息以及状态确认消息构成，其规模远小于 GB 级虚拟机状态数据，网络开销相对于内存封装、加解密和存储 I/O 可忽略不计。基于上述数据路径，本文实验主要评估 ResiCVM 引入的可信封装与恢复开销；网络传输开销属于云平台存储后端的正交因素，可通过提高存储网络带宽、采用近端副本或异步复制机制进一步优化。

5.2 安全快照

我们按照 4.3 节描述的封装工作流，评估了 ResiCVM 安全快照机制的效率。由于 AMD SEV-SNP 并未原生提供用于原子化 CVM 快照的高级直接接口，我们使用 SNP_PAGE_SWAP_OUT 接口作为具有代表性的基线。该接口通过 ASP 对 CVM 内存页进行加密并导出到持久化存储中，是官方用于内存状态提取的硬件原语。

实验结果如图 5 所示，启用 AES-NI 的 ResiCVM 实现了 120.50 MBps 的状态封装吞吐量，与原生 ASP 方案（1.92 MBps）相比，实现了 62.7 倍的加速。纯软件实现的变体达到了 4.80 MBps，这表明了即使在硬件加速不能开启的情况下，主机 CPU 对于安全处理器仍具有性能优势。至关重要的是，由于原生 ASP 方案采用单线程、顺序执行的命令处理队列，其单台虚拟机的吞吐量在并发情况下呈线性下降；而 ResiCVM 通过将密码学任务下放至每个虚拟机的专属高特权域（VMPL 0），充分利用了宿主机的多核并行能力。因此，在高密度的云环境中，对于多个 CVM 并发备份的情况下，ResiCVM 仍能保持稳定且高吞吐量的表现。

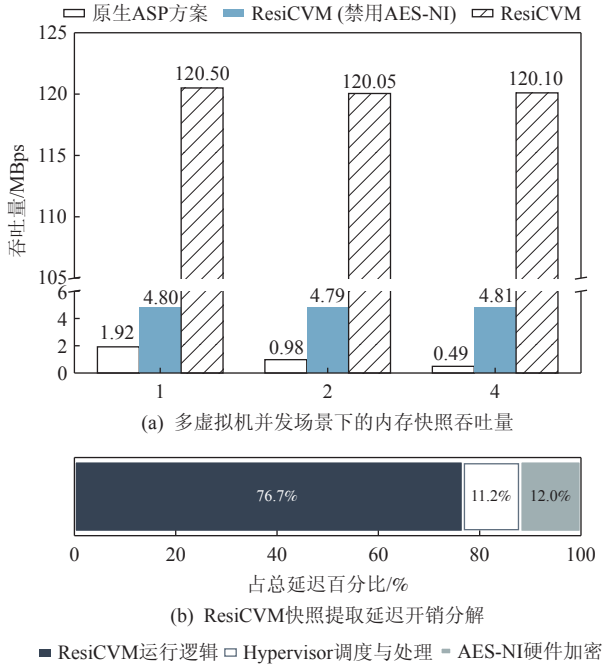


Fig. 5 Time cost of extracted by ResiCVM snapshot

图 5 ResiCVM 快照提取的时间开销

这一结果充分验证了 ResiCVM 通过特权隔离域卸载密码学计算的架构设计，成功打破了硬件安全处理器（ASP）在并发处理能力上的串行化性能瓶颈，为云原生机密计算提供了更具可扩展性的容灾备份范式。

5.3 端到端的备份与恢复

除了单次的快照提取，我们还评估了 ResiCVM 端到端的容灾恢复效率。该实验包含一个完整的周期：生成 CVM 的完整快照，将其归档到物理存储中，随后将该状态恢复到一个新初始化的 CVM 实例中。为了消除网络带来的变量，并严格聚焦于系统和密码学开销，备份和恢复均在同一台物理服务器上执行。

总耗时从安全握手启动开始计算，直到恢复后的 CVM 恢复 Guest OS 的执行为止。我们使用基于原生 ASP 的状态导出/导入序列作为基线，而标准的 KVM 快照/恢复则作为性能上限，实验结果如表 5 所示。值得注意的是，在全量备份和恢复场景中，ResiCVM 相对于基于 ASP 的方法的性能优势进一步扩大到了 119 倍，超越了仅在快照阶段观察到的 62.7 倍提升。此效率的提升归功于 ResiCVM 流水线式的内存封装与解封工作流，它有效地将密码学处理与存储 I/O 进行了重叠，这是目前串行的 ASP 执行模型所无法实现的。

此外，ResiCVM 的端到端有效备份与恢复带宽约等价于 885 MBps，低于现代云数据中心常见的 10 Gbps 甚至更高的物理链路承载能力。因此，在单实例本

Table 5 Snap and Restore Time for Different Solutions

表 5 不同方案的快照与恢复时间

性能	KVM	ASP	ResiCVM	ResiCVM (no Enc)
总耗时/s	4.43	2 203.15	18.51	16.18
带宽/MBps	462.30	0.93	110.64	126.58

地或近端恢复场景中,网络通常不是 ResiCVM 的主要瓶颈。然而,在同云平台跨宿主机恢复、跨可用区恢复或多 CVM 并发恢复场景中,网络传输开销会随快照规模和并发实例数量增加。设快照大小为S,ResiCVM 封装/恢复吞吐量为B_{r},云平台存储带宽为B_{s},网络传输带宽为B_{n},则非本地恢复场景下的端到端耗时可近似表示为

$$T_{\mathrm{remote}} \approx T_{\mathrm{attest}} + \frac{S}{\min(B_{\mathrm{r}}, B_{\mathrm{s}}, B_{\mathrm{n}})}。$$

由此可见,当B_{n}高于 ResiCVM 的封装/恢复吞吐量时,系统瓶颈主要来自可信封装、内存重映射和存储 I/O;当跨可用区或异地恢复导致B_{n}较低时,网络传输会成为主要瓶颈。该开销属于云平台存储与网络后端因素,与 ResiCVM 的安全封装机制正交,可通过近端副本、异步复制或提高存储网络带宽进一步优化。

尽管如此,与原生 KVM 的明文快照并恢复相比,ResiCVM 仍然存在 4.19 倍的性能差距。为了调查这一开销的来源,我们进行了一项额外的消融实验,即在不进行加密或解密的情况下处理内存。结果表明,即使没有密码学操作,与标准的 KVM 虚拟机相比,仍然存在显著的效率差距(3.65 倍)。我们将这一剩余开销主要归因于必须执行的可信硬件对内存的重映射操作以及 ResiCVM 对内存块的处理,这些操作是维持严格的虚拟机内安全隔离所必需的。

5.4 虚拟机维护对服务 CVM 性能的影响

为了评估虚拟机维护操作(快照提取与状态恢复)对活跃业务的影响,我们使用运行 Apache Web 服务器(配置为 2 vCPU, 2 GB 内存)的目标 CVM 进行了一次高负载模拟测试。在此过程中,远程客户端使用 ApacheBench 工具与该 Web 服务器建立了 100 个并发连接,以模拟真实的高负载服务场景。

需要指出的是,由于 ResiCVM 利用了虚拟机自身的 vCPU 周期来执行高特权级的状态提取与解密,这不可避免地会占用原本用于处理业务的计算资源,从而在维护操作期间降低 CVM 的业务处理能力。为了量化这种影响,我们在快照与恢复过程中持续监控了 Apache 服务的实时吞吐量,结果如图 6 所示。评估显示,尽管 vCPU 卸载开销在瞬时会导致 ResiCVM 的服务带宽出现下降,但从整个维护周期来看,其对总服务流量的整体影响却远小于 ASP 方案。具体而言,ResiCVM 完成 1 次完整的快照与恢复过程所导致的累积吞吐量损失仅约 2.8 万次请求;形成鲜明对比的是,硬件 ASP 方案由于其极其缓慢的执行速度,导致服务经历了漫长的中断窗口,最终造成了约 13.6 万次请求的巨大吞吐量损失。

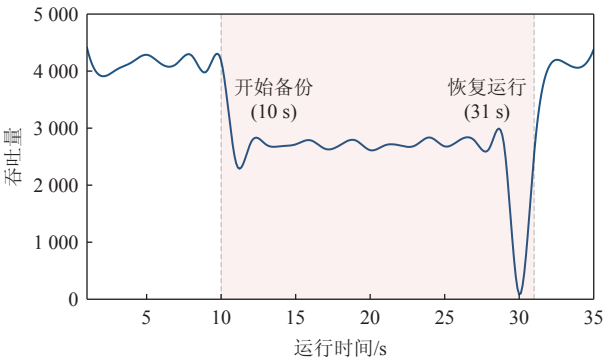


Fig. 6 Real-time service throughput during the period of snapshot and restoration of ResiCVM

图 6 ResiCVM 快照与恢复期间的实时服务吞吐量

这些发现证实,ResiCVM 大幅缩短维护窗口的架构优势,有效地压缩了性能下降的持续时间,从而在保证数据安全的同时,显著提升了机密计算系统的整体弹性与高可用性。

6 安全性分析

本节评估了 ResiCVM 抵御本威胁模型中定义的各类敌手(包括恶意宿主机、网络级攻击者以及被攻破的客户机软件)的安全弹性。通过将硬件级强制隔离与健壮的密码学协议相结合,ResiCVM 确保了 CVM 容灾备份与恢复生命周期内的机密性与完整性。

6.1 密钥的机密性

ResiCVM 协议的安全性基于一种分层的密钥结构,其中每一层的机密性均由底层硬件严格保障。作为信任基础,HEK 由底层可信硬件与安全固件管理,确保其私钥部分永远不会离开安全处理器。在此基础上,CSK 与 SEK 完全驻留于 CVM 内部高特权级的私有非共享内存中,宿主机与 Guest OS 均绝对无法访问。具体而言,在执行快照操作时,SEK 在导出前会使用 CAK 的公钥进行封装。由于对应的 CAK 私钥仅由安全管理中心持有,因此即使将封装后的 SEK 以及加密后的客户机状态数据存储在不可信的宿主机环境中,其安全性依然可以得到保证。

6.2 密钥交换协议的安全性

本节在标准密码学假设下对 ResiCVM 与安全管理中心之间的密钥交换协议进行安全性分析。设敌手$\mathcal{A}$能够完全控制 Host、Hypervisor、Guest OS、网络代理和外部通信信道,可以窃听、篡改、重放、删除或伪造任意协议消息。但$\mathcal{A}$不能破坏 TEE 硬件隔离

机制，不能伪造 HEK 生成的硬件远程证明报告，不能获取安全管理中心私钥 CAK^{priv}，也不能读取 VMPL 0 中 ResiCVM 的私有内存。

本文基于如下密码学假设：硬件远程证明报告不可伪造；数字签名算法满足选择消息攻击下的存在性不可伪造性；公钥加密算法满足密文不可区分性；安全管理中心生成的随机挑战 N_{sc} 具有足够长度，碰撞概率可忽略。

首先，协议能够保证 ResiCVM 身份不可伪造。在双向身份认证阶段，ResiCVM 将安全管理中心生成的随机挑战 N_{sc} 和自身生成的 CSK^{pub} 绑定到硬件远程证明报告 R_{CVM} 中，并由 HEK 对该报告进行硬件签名。安全管理中心只有在硬件签名有效、代码度量值匹配、特权级检查通过且 N_{sc} 与当前会话一致时，才接受其中的 CSK^{pub}。由于敌手无法伪造 HEK 签名，也无法在未运行合法 ResiCVM 代码的情况下生成有效报告，因此敌手不能将任意公钥伪装成合法 ResiCVM 的 CSK^{pub}。

其次，协议能够保证 SEK 传输的真实性与完整性。在密钥交换阶段，ResiCVM 使用 CAK^{pub} 加密 SEK，并使用 CSK^{priv} 对密文进行签名。安全管理中心使用阶段 1 中已经通过远程证明绑定的 CSK^{pub} 验证签名。若敌手篡改、替换或伪造密文，则必须生成一个能够通过 CSK^{pub} 验证的有效签名，这与数字签名不可伪造性假设矛盾。因此，安全管理中心接受的 SEK密文只能来源于合法 ResiCVM 实例。

最后，协议能够抵抗重放攻击。安全管理中心在每次认证过程中生成新的随机挑战 N_{sc}，并要求 ResiCVM 将其绑定到硬件远程证明报告中。若敌手重放历史证明报告，则报告中的 N_{sc} 与当前会话挑战不一致，验证会失败。进一步地，每次快照或恢复会话都会使用独立的会话标识和新生成的 SEK，从而避免不同维护会话之间的密钥混用。

综上，在硬件远程证明不可伪造、签名算法不可伪造、公钥加密算法安全以及 VMPL 0 私有内存受硬件隔离保护的前提下，ResiCVM 的密钥交换协议能够保证 CSK^{pub} 的身份绑定、SEK 传输的机密性、真实性、完整性以及协议消息的新鲜性。

6.3 来自宿主机的攻击

在面对系统级敌手时，ResiCVM 采用了多层防御机制来防止未经授权的数据访问或状态篡改。尽管宿主机管理着底层的物理资源，但由于底层 TEE 硬件提供的强制内存加密机制，它无法从 CVM 中提取任何敏感信息。宿主机任何试图窃取或篡改导出的快照数据的行为，都会被 AES-GCM 认证加密和哈希值认证所检测到。此外，TEE 安全固件针对 Host 对 CVM 内存的访问控制在硬件层面阻止了宿主机直接修改客户机的私有内存。

为了确保状态新鲜度并抵御回滚攻击，ResiCVM 采用了一次性密钥机制：为每一次快照或恢复操作生成唯一的、特定于当前会话的 SEK。如果宿主机提供未经安全管理中心授权的旧快照，ResiCVM 会因会话标识、快照摘要或密钥元数据不匹配而拒绝恢复。通过将这些临时会话密钥与验证了安全版本、代码度量值以及新鲜度随机数的硬件签名报告相结合，ResiCVM 确保容灾恢复过程仅能基于安全管理中心授权，且通过完整性与新鲜性校验的快照状态进行。

6.4 来自网络敌手的攻击

针对能够窃听、篡改或劫持通信通道的网络级敌手，ResiCVM 通过密码学认证握手与严格的指令验证逻辑进行防御。为了抵御密钥交换和远程证明过程中的重放攻击，ResiCVM 在双向证明阶段引入了基于随机挑战码 N_{sc} 的新鲜性机制。安全管理中心生成随机挑战码，并要求 ResiCVM 将该挑战码与自身的身份公钥、代码度量值和特权级信息共同绑定到硬件远程证明报告中。只有当报告签名有效、度量值匹配、特权级正确且挑战码与当前会话一致时，安全管理中心才接受该 ResiCVM 实例。因此，攻击者无法通过重放历史证明报告伪装成合法实例。

此外，为了保护安全管理中心与 ResiCVM 模块之间信息流的安全。我们实现了一种基于序列号的纵深防御机制，以防止未经授权的指令调用。安全管理中心发出的每一个容灾维护请求，在被签名与加密之前，都会拼接一个唯一的、单调递增的序列号。ResiCVM 模块在接收到指令后，会校验其签名并严格检查该序列号未曾被处理过。该机制有效防止了攻击者重复触发维护操作或非法状态分叉。由于会话密钥与指令均以密码学方式锚定至硬件身份并由安全管理中心进行认证，任何中间人介入或重放尝试在计算复杂度上都是不可行的。

7 结　　论

本文提出了 ResiCVM，一种基于虚拟机内部特权级隔离的面向机密计算集群的实用、安全且高效的容灾备份与状态快照框架。ResiCVM 成功将状态提取与密码学逻辑从算力僵化的底层硬件上转移到

虚拟机内部的高特权级模块中。实验结果表明,在将 Guest OS 严格排除在 TCB 之外的前提下,ResiCVM 相比于传统的基于底层可信硬件的方案,实现了 62.7 倍的快照吞吐量提升,并将总备份与恢复时间缩短至 1/119。此外,在多节点并发场景下,ResiCVM 保持了高效稳定的执行能力,显著缓解了机密计算集群在灾备过程中的性能瓶颈,为系统赋予了强大的安全弹性。

尽管取得了上述显著的进展,目前的 ResiCVM 原型系统仍存在一些局限性,有待在未来的工作中解决。首先,与原生的 KVM 备份机制相比,ResiCVM 仍然存在 4.19 倍的性能差距。这一开销主要来源于 ResiCVM 模块内部的内存复制与内存状态转换。其次,由于 ResiCVM 依赖客户机的 vCPU 来卸载和执行安全任务,这不可避免地引入了底层资源争抢,导致在执行快照的窗口期内,客户机正常运行业务的实时吞吐量会出现短暂的下降。最后,受限于当前实验平台硬件版本的约束,我们对增量快照机制的评估目前依赖于基于访存轨迹的脏页追踪模拟,而非原生硬件指令执行[30]。

展望未来,我们计划在 2 个极具实用价值的方向上对 ResiCVM 进行深度扩展。首先,为了最大化处理吞吐量,我们计划在 ResiCVM 内部实现多线程并行机制,将页面数据封装与密码学工作负载高效分发至多个 vCPU 上进行并行处理。其次,我们将在实验平台支持后,为 ResiCVM 实现增量快照与服务器负载平衡机制。

作者贡献声明:马鸿展负责方案整体设计并撰写论文;曲若非对方案进行了改进;沈牧言、贾梦涵负责部分实验;冯登国提供了论文的思路;秦宇、冯伟对论文提出了指导意见并修改了论文。

参 考 文 献

[1] AMDESE (AMD SECURE ENCRYPTED VIRTUALIZATION). QEMU at sev_live_migration_v4 branch[EB/OL]. 2025[2025-12-04]. https://github.com/AMDESE/qemu/tree/sev_live_migration_v4

[2] Guerreiro J, Moura R, Silva J N. TEEnder: SGX enclave migration using HSMs[J]. Computers & Security, 2020, 96: 101874

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

[4] Kenjar Z, Frassetto T, Gens D, et al. V0LTpwn: Attacking x86 processor integrity from software[C]//Proc of the 29th USENIX Security Symp (USENIX Security 20). Berkeley, CA: USENIX Association, 2020: 1445−1461

[5] Van Schaik S, Milburn A, Österlund S, et al. RIDL: Rogue in-flight data load[C]//Proc of 2019 IEEE Symp on Security and Privacy (SP). Piscataway, NJ: IEEE, 2019: 88−105

[6] Kocher P, Horn J, Fogh A, et al. Spectre attacks: Exploiting speculative execution[J]. Communications of the ACM, 2020, 63(7): 93−101

[7] Lipp M, Schwarz M, Gruss D, et al. Meltdown[J]. arXiv preprint, arXiv: 1801.01207, 2018

[8] Schwarz M, Weiser S, Gruss D. Practical enclave malware with Intel SGX[C]//Proc of the 16th Int Conf on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2019). Berlin: Springer, 2019: 177−196

[9] Advanced Micro Devices. AMD SEV-SNP:Strengthening VM isolation with integrity protection and more[EB/OL]. 2020[2020-01]. https://docs.amd.com/v/u/en-US/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more

[10] Feng Dengguo, Qin Yu. Confidential computing: Progress and prospects[J]. Communications of China Computer Federation, 2024, 20(12): 38−46 (in Chinese)
(冯登国, 秦宇. 机密计算: 进展与展望[J]. 中国计算机学会通讯, 2024, 20(12): 38−46)

[11] Ross R, Pillitteri V, Graubart R, et al. Developing cyber resilient systems: A systems security engineering approach[R]. Gaithersburg, MD: National Institute of Standards and Technology, 2019

[12] McKeen F, Alexandrovich I, Anati I, et al. Intel® Software Guard Extensions (Intel® SGX) Support for Dynamic Memory Management Inside an Enclave[M]//Proc of the Hardware and Architectural Support for Security and Privacy 2016. New York: ACM, 2016: 1−9

[13] INTEL CORPORATION. Intel trust domain extensions (Intel TDX) overview[EB/OL]. 2025[2025-12-04]. https://www.intel.com/content/www/us/en/developer/tools/trust-domain-extensions/overview.html.

[14] ARM LIMITED. Arm confidential compute architecture (CCA)[EB/OL]. 2025[2025-12-04]. https://www.arm.com/architecture/security-features/arm-confidential-compute-architecture.

[15] Wilke L, Wichelmann J, Rabich A, et al. Sev-step: A single-stepping framework for amd-sev[J]. arXiv preprint, arXiv: 2307.14757, 2023

[16] Schlüter B, Sridhara S, Kuhne M, et al. HECKLER: Breaking Confidential VMs with Malicious Interrupts[C]//Proc of the 33rd USENIX Security Symp (USENIX Security 24). Berkeley, CA: USENIX Association, 2024: 3459−3476

[17] Guo Ziyi, Le D K, Lin Zhenpeng, et al. Take a step further: Understanding page spray in Linux kernel exploitation[C]//Proc of the 33rd USENIX Security Symp (USENIX Security 24). Berkeley, CA: USENIX Association, 2024: 1189−1206

[18] Zeng K, Chen Y, Cho H, et al. Playing for K (H) eaps: Understanding and improving Linux kernel exploit reliability[C]//Proc of the 31st USENIX Security Symp (USENIX Security 22). Berkeley, CA: USENIX Association, 2022: 71−88

[19] Ziqiao Zhou, Anjali, Weiteng Chen, et al. VeriSMo: A verified security module for confidential VMs[C]//Proc of the 18th USENIX Symp on Operating Systems Design and Implementation (OSDI 24). Berkeley, CA: USENIX Association, 2024: 599−614

[20] Jamshidi P, Ahmad A, Pahl C. Cloud migration research: A systematic

review[J]. IEEE Transactions on Cloud Computing, 2013, 1(2): 142−157

[21] Wang J, Mahmoody P, Brasser F, et al. VirTEE: A full backward-compatible TEE with native live migration and secure I/O[C]//Proc of the 59th ACM/IEEE Design Automation Conf. New York: ACM, 2022: 241−246

[22] Yang Fangjie, Li Jianlin, Xue Kaiwen, et al. Designing and implementing live migration support for ACM-based confidential VMs[C]//Proc of the 15th ACM SIGOPS Asia-Pacific Workshop on Systems. New York: ACM, 2024: 53−59

[23] Bai Yunkai, Li Peinan, Huang Yubiao, et al. Hypertee: A decoupled tee architecture with secure enclave management[C]//Proc of the 57th IEEE/ACM Int Symp on Microarchitecture (MICRO). Piscataway, NJ: IEEE, 2024: 105−120

[24] Mao Jingkai, Chang Xiaolin, Li Lin, et al. Towards verifiable trust proof for trusted confidential virtual machines[J]. IEEE Transactions on Network Science and Engineering, 2026, 13: 552-567

[25] Kapil D, Mishra S K, Gupta V K. A performance perspective of live migration of virtual machine in cloud data center with future directions[J]. International Journal of Wireless and Microwave Technologies, 2022, 12(4): 48−56

[26] 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 (USENIX Security 21). Berkeley, CA: USENIX Association, 2021: 1073−1090

[27] Misono M, Stavrakakis D, Santos N, et al. Confidential VMS explained: An empirical analysis of amd sev-snp and Intel TDX[J]. Proceedings of the ACM on Measurement and Analysis of Computing Systems, 2024, 8(3): 1−42

[28] Priebe C, Vaswani K, Costa M. EnclaveDB: A secure database using SGX[C]//Proc of 2018 IEEE Symp on Security and Privacy (SP). Piscataway, NJ: IEEE, 2018: 264−278

[29] Pawar M V, Anuradha J. Network security and types of attacks in network[J]. Procedia Computer Science, 2015, 48: 503−506

[30] Advanced Micro Devices, Inc. AMD64 RMP Dirty[R/OL]. Advanced Micro Devices, Inc., 2026. https://docs.amd.com/api/khub/documents/dSDYo9rxWwNDSJkmMJCxYQ/content.



Ma Hongzhan, born in 2000. PhD candidate. His main research interests include confidential computing and confidential machine leaning.

马鸿展, 2000 年生。博士研究生。主要研究方向为机密计算、机密机器学习。



Qu Ruofei, born in 2002. Master candidate. Her main research interests include confidential computing and system security.

曲若非, 2001 年生。硕士研究生。主要研究方向为机密计算、系统安全。



Shen Muyan, born in 2000. PhD candidate. His main research interests include confidential computing and system security.

沈牧言, 2000 年生。博士研究生。主要研究方向为机密计算、系统安全。



Jia Menghan, born in 2002. Master candidate. Her main research interest includes confidential computing.

贾梦涵, 2002 年生。硕士研究生。主要研究方向为机密计算。



Feng Dengguo, born in 1965. PhD supervisor, academician of CAS. Senior member of CCF. His main research interests include confidential computing, cryptography, and trusted computing.

冯登国, 1965 年生。博士生导师。中国科学院院士。CCF 高级会员。主要研究方向为机密计算、密码学、可信计算。



Qin Yu, born in 1979. PhD, senior engineer. His main research interests include confidential computing, trusted computing, and system security.

秦　宇, 1979 年生。博士，正高级工程师。主要研究方向为机密计算、可信计算、系统安全。



Feng Wei, born in 1986. PhD, associate professor. His main research interests include system security, confidential computing, and trusted computing.

冯　伟, 1986 年生。博士，副研究员。主要研究方向为系统安全、机密计算、可信计算。