面向RISC-V平台的安全高效固件可信平台模块设计与实现

王 杰 王 鹃*

(空天信息安全与可信计算教育部重点实验室 武汉 430072)

(武汉大学国家网络安全学院 武汉 430072)

摘 要：可信平台模块(TPM)作为提升系统安全性的核心技术，能够提供基于硬件的密钥管理、可信启动和远程认证等安全功能。然而，当前 RISC-V平台普遍缺乏TPM支持，限制了其在嵌入式和云计算场景中的安全能力。为解决这一问题，该文设计并实现了RfTPM——一种面向RISC-V平台的固件可信平台模块(fTPM)架构，无需额外硬件单元或安全扩展即可提供等效的安全功能。针对执行隔离、可信启动、高效通信和安全时钟等关键挑战，在RfTPM中，该文提出了创新解决方案，包括：基于RISC-V物理内存保护(PMP)机制的内存隔离以及结合DRAM物理不可克隆函数(PUF)与Flash锁定的静态数据保护、基于延迟度量扩展的可信启动机制、基于动态权限交换页的高效通信机制以及基于RISC-V硬件计时器的细粒度安全时钟。该文构建了RfTPM的原型系统，对其进行了安全性分析并在Genesys2 FPGA平台模拟的Rocket Core上进行了性能测试。实验结果表明，RfTPM在保证安全性的同时在大多数TPM命令处理中有比较显著的性能优势。

关键词：可信平台模块；固件可信平台模块；RISC-V；可信启动

中图分类号：TN918; TP309 **文献标识码**：A **文章编号**：1009-5896(2025)07-2385-11

DOI: 10.11999/JEIT241112 **CSTR**: 32379.14.JEIT241112

1 引言

当前，可信计算[1–3]作为一种被广泛接受的安全技术，在嵌入式设备、云计算等领域发挥着重要作用。可信平台模块 (Trusted Platform Module, TPM)[4–6]作为可信计算的核心技术，能够为平台提供关键的安全功能，包括密钥管理、数据密封、安全存储、可信启动和远程认证等。TPM技术如今已广泛应用于个人计算机、嵌入式设备和云计算等领域，提供磁盘加密、身份认证、固件防回滚保护以及可信云服务等功能。常见应用包括BitLocker, Windows Hello和Linux内核完整性测量架构等，显著提升了系统的整体安全性和可靠性。

近年来，固件TPM (firmware TPM, fTPM)[7–9]引起了学术界和工业界的广泛关注。fTPM是一种软件形式的TPM，无需依赖专用硬件即可提供与硬件TPM一致的安全功能接口。它通常运行在受保护的隔离环境中，如可信执行环境 (Trusted Execution Environment, TEE)[10,11]、Intel ME (Management Engine) 或AMD PSP (Platform Security Processor)。考虑到fTPM的便利性和可用性，目前主流平台 (包括Intel, AMD和Arm) 均已推出各自的fTPM实现。

收稿日期：2024-12-17；改回日期：2025-05-27；网络出版：2025-06-13

*通信作者： 王鹃 jwang@whu.edu.cn

基金项目：国家科技重大专项(2024ZD0803000)

Foundation Item: The National Science and Technology Major Project (2024ZD0803000)

然而，当前RISC-V 平台仍缺乏对TPM的支持。作为一个新兴的指令集架构(ISA)，RISC-V近年来发展迅速，已从嵌入式领域逐步扩展到高性能计算场景[12,13]。但无论面向嵌入式还是云计算环境设计的RISC-V平台仍然普遍缺乏对TPM的支持。一种直接的解决方案是为RISC-V SoC增加硬件TPM芯片。但这种方案不仅增加了硬件成本，还需要对SoC进行适配，从而进一步提高了设计复杂性和成本。同时，现有的fTPM方案则主要面向x86和Arm架构，并通常依赖于特定的硬件安全机制，比如独立的安全芯片、TEE或重放保护内存块(Replay Protected Memory Block, RPMB) 导致难以直接在RISC-V平台应用。鉴于以上限制，本文提出了一个关键研究问题：能否为 RISC-V平台设计一种安全、高效且实用的fTPM架构？

在RISC-V平台上设计和实现fTPM面临诸多挑战，主要包括：如何保护fTPM内存中的代码和数据，并维持其持久化数据的安全性？fTPM通常在平台启动中初始化，在这种情况下如何维持TPM的可信启动功能的完整性？如何实现fTPM 与上层TPM应用之间的安全高效通信？如何设计安全时钟以满足TPM的需求？

为了解决上述挑战，本文设计并实现了首个面向RISC-V平台的fTPM架构——RfTPM。在不增加额外硬件成本的情况下，RfTPM 能够为缺乏硬件TPM支持的RISC-V平台提供等效的安全功能，包括隔离执行、安全存储、可信启动和远程认证等。具体设计包括：

(1) 基于PMP的内存隔离以及基于受保护的DRAM PUF (Physically Unclonable Functions)[14,15]和Flash锁定的静态数据保护：通过配置PMP，有效防御特权软件对fTPM内存中代码和数据的侵害。同时，设计了基于受保护的DRAM PUF的根密钥构建算法，并利用派生密钥对fTPM的静态数据进行加密。此外，通过PMP实现Flash锁定，有效防御特权软件发起的fTPM静态数据回滚攻击。

(2) 基于延迟度量扩展的可信启动：为应对RfTPM架构中fTPM延迟启动的特性带来的问题，将fTPM划分为验证启动以及度量启动两个阶段，并通过在验证启动阶段结束后引入延迟度量扩展机制保持fTPM安全启动功能的完整性。

(3) 基于动态权限交换页的安全高效通信：为了进一步增强RfTPM中TPM命令通信效率，为RfTPM引入了跨特权层的TPM信息共享交换页来实现零拷贝的通信，并通过动态配置PMP权限保证该交换页在层间共享时的安全性。

(4) 基于硬件的安全时钟：利用 RISC-V平台的底层硬件计时器为RfTPM构建了细粒度、安全的时钟机制，保证攻击者无法通过操纵时钟来绕过fTPM内部的安全机制。

本文在 RISC-V平台上实现了RfTPM的原型系统，并将其集成为OpenSBI[16]的一个安全扩展模块。fTPM作为固件的一部分，能够向上层应用提供TPM服务。与此同时，本文为RfTPM构建了专用的内核驱动并对上层TPM软件栈(TPM Software Stack, TSS) 进行了适配。使用Genesys2 FPGA开发板模拟了Rocket Core[17]，并在其上测试了RfTPM的性能。实验结果显示，与硬件TPM相比，RfTPM在大多数命令处理上具有显著性能优势。例如，在创建RSA密钥时，RfTPM性能提升约7倍。并且在TPM常用命令的测试中，RfTPM的性能甚至优于未受安全保护的TPM模拟器。为促进社区交流和深入研究，本文已开源部分核心代码，详见 https://github.com/wchuanmu/RfTPM。

2 背景

2.1 TPM

TPM是由可信计算组 (Trusted Computing Group, TCG) 定义的国际安全规范，旨在为计算平台提供信任根支持。TPM包含存储、度量和报告3大信任根，用于支持数据密封、可信启动、安全存储以及远程认证等功能。目前，TPM的实现形式主要包括以下几种。

(1) 硬件TPM (Discrete TPM, dTPM)：基于独立的计算和存储单元设计，通常具备专门的硬件安全模块(如防篡改保护、电磁屏蔽、物理入侵检测等)，能够有效抵御物理攻击和软件层面的恶意篡改。因此，dTPM具有最高的安全级别，并广泛应用于企业级服务器、可信计算环境和高安全性设备。然而，由于TPM采用独立硬件，TPM命令的处理速度受限于硬件资源，整体性能相对较低。此外，dTPM设计固化在硬件中，一旦发现安全漏洞，修复通常需要更换芯片或通过有限的固件更新进行缓解，成本较高，灵活性较差。

(2) 软件TPM (Software TPM, SWTPM)：软件实现的TPM，例如ibmswtpm2[18]，主要用于开发和测试环境。SWTPM通过软件模拟TPM的功能，提供标准化的TPM接口，支持应用层进行TPM相关操作。然而，由于其完全运行在普通软件环境中，缺乏独立的硬件隔离，攻击者可以通过获取主机权限来访问或篡改TPM状态，因此安全性较低。在实际应用中，SWTPM适用于TPM相关软件开发、功能验证环境，但不适用于对安全性要求较高的场景，如可信计算或机密计算环境。

(3) 固件TPM (Firmware TPM, fTPM)：运行于受保护的隔离执行环境(如 Arm TrustZone)，通过固件实现TPM功能，避免了额外的硬件成本。fTPM依赖于CPU内部的安全隔离机制，与硬件TPM相比，虽然在物理安全性上略逊一筹，但能够提供较强的软件隔离保护，使得攻击者难以通过普通操作系统访问TPM相关数据。此外，fTPM具备更高的可扩展性，支持安全补丁更新，可用于嵌入式系统、移动设备和轻量级可信计算场景。

(4) 虚拟化TPM (Virtual TPM, vTPM)：主要用于云计算和虚拟化环境，为虚拟机提供TPM相关功能。vTPM运行在虚拟化管理程序(Hypervisor)或专用安全环境中，支持多个虚拟机共享TPM资源，从而提升多租户环境下的可信计算能力。然而，传统vTPM由于依赖于软件实现，可能受到管理程序层面的攻击，导致隔离性不足。近年来，研究重点聚焦于如何利用安全技术(如TEE)提升vTPM的隔离性，以降低对Hypervisor的信任依赖，提高安全防护能力[19,20]。

当前，TPM在多个安全领域中发挥着核心作用，广泛应用于可信计算、设备认证、安全启动和远程证明等关键场景。在工业互联网、云计算和物联网等环境中，TPM作为系统的信任根，能够确保设备身份的可信性，并提供安全存储与密钥管理等功能。例如，在内生安全体系中，TPM可用于设备身份认证和动态信任评估，并可结合区块链与零信任机制，进一步增强工业控制系统的安全防护

能力[21,22]。此外，TPM还可应用于边缘计算与联邦学习方案，保障边缘设备间数据交互的可信性，从而提升协同计算的整体安全性[23,24]。

2.2 RISC-V

RISC-V是一种开源的精简指令集架构，凭借其开源特性、模块化设计、简洁性以及高度的可扩展性，受到了工业界和学术界的广泛关注。自2010年由加州大学伯克利分校首次发布以来，RISC-V已从最初的研究原型发展成为具有全球影响力的开放计算架构，在多个领域取得了显著进展。RISC-V的核心理念是提供一个灵活且高效的基础指令集，并通过可选扩展模块支持不同场景的优化需求。这种模块化的设计使研究者和开发者能够根据应用需求，设计适配于通用计算、嵌入式系统、高性能计算或人工智能等领域的专用处理器。同时，开源的特性降低了技术准入门槛，激发了大量创新，包括硬件加速器[25,26]、面向物联网的低功耗设计[27]、TEE[28]以及安全性增强[29]等领域的研究与实践。

2.3 PMP机制

PMP是RISC-V特权架构提供的一种物理内存隔离机制，它允许机器模式中的软件控制低特权模式软件的内存访问权限。每个RISC-V硬件线程都有一组PMP条目，能够控制多个内存区域的地址、大小和权限。PMP主要包括两类寄存器：pmpaddr和pmpcfg。pmpaddr寄存器用于配置物理地址的隔离边界，而pmpcfg寄存器则用于控制地址匹配模式和权限。RISC-V平台通常提供16组PMP条目，因此最多可以创建16个隔离区域。此外，RISC-V还为PMP设计了优先级机制：具有最低PMP索引的条目决定隔离地址域的权限。

3　架构总览

3.1 设计目标

(1) 安全性：RfTPM需要保证TPM命令的执行不受特权软件的干扰和破坏，并且其非易失性静态数据也应受到保护，保持机密性、完整性和新鲜性。

(2) 高效性：RfTPM利用CPU核心执行TPM命令，具备较高的执行性能。RfTPM中引入的安全机制应保持较小的开销保证不会对TPM命令的执行性能造成较大的影响。

(3) 实用性：RfTPM应仅依赖RISC-V平台的标准安全机制而不能依赖特定的安全扩展以最大化RfTPM的实际可用性。

3.2 架构概述

图1展示了RfTPM的总体架构。该架构主要包括硬件层和软件层。硬件层要求使用支持PMP机制的RISC-V CPU核。此外，RfTPM运行的SoC平台需要配备闪存(Flash) 或BootROM，以存储fTPM的非易失性数据(Non-Volatile Random Access Memory, NVRAM) 。eFuse (Electronic Fuse)是一种一次性可编程电子组件，可以为设备保存根启动密钥。RfTPM可以基于硬件出厂时在eFuse中固化的根启动密钥来派生与fTPM相关的密钥，比如NVRAM的加密密钥。eFuse为可选组件，如果平台不支持eFuse, RfTPM支持通过PUF构建根密钥。软件层根据RISC-V的特权级别分为3种模式：用户模式、监管者模式和机器模式。TPM应用及TSS套件运行于用户模式；fTPM驱动作为内核模块运行于监管者模式；而fTPM则运行于机器模式，拥有最高特权。

在RfTPM系统中，fTPM作为固件工作，并向上层提供TPM服务。fTPM包括多个模块，按照功能可以划分为功能模块、安全模块和接口处理模块。功能模块包括TPM命令执行引擎、密钥管理模块、密码引擎和PCR (Platform Configuration Register)等。安全模块则包括隔离执行模块和身份认证模块。隔离执行模块利用动态配置的PMP实现fTPM内存隔离，并保护Flash的MMIO接口，从而确保fTPM的静态数据和动态数据的全面防护。接口模块包括ECALL (Environment Call)处理模块和SPI (Serial Peripheral Interface)驱动模块。ECALL处理模块解析TPM命令并将其路由至相关处理单元，而SPI驱动则确保fTPM能够读取和写入Flash，以保存和恢复fTPM的NVRAM数据。

在RfTPM中，一条TPM命令的整体处理过程如下。

(1) TPM应用根据所需功能构建TPM命令，并写入指定的内存地址。随后，将请求发送到fTPM驱动。

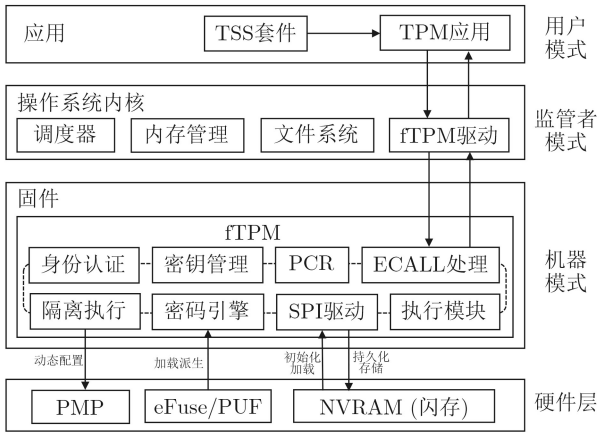

图 1 RfTPM系统架构

(2) fTPM驱动接收到命令后，会进行初步的安全性检查。由于TPM是单线程的，fTPM驱动会根据当前fTPM的状态判断是否转发命令。如果fTPM处于空闲状态，驱动会通过sbi_ecall转发该TPM命令。

(3) fTPM中的ECALL处理模块接收到请求后，会将TPM命令所在的内存页权限关闭。之后，fTPM会对TPM命令进行安全检查，解析TPM命令并将其发送给相应的处理单元。对于需要持久化存储的命令 (如创建持久密钥、授权认证等)，fTPM会将数据加密后通过文件接口保存至NVRAM。

(4) 完成命令处理后，fTPM将执行结果写入指定的内存页，开启该内存页读写权限，并将执行权交还给fTPM驱动。fTPM驱动最终会返回到TPM应用，完成命令执行。

3.3 威胁模型

在RfTPM中，假设操作系统是不可信的，攻击者可以控制操作系统的各个方面，包括CPU调度、内存管理和设备控制等功能。信任RISC-V硬件，并假设其硬件实现不存在漏洞，保证PMP机制和特权级别等安全机制无法被绕过。此外，仅信任运行在机器模式下的fTPM，并假设fTPM的实现没有漏洞。

需要注意的是，物理攻击，如冷启动攻击、Rowhammer攻击和瞬态执行不在本文考虑范围内。尽管RfTPM会在上下文切换时刷新缓存，以避免同一核心上的缓存信息泄露，但针对核间共享缓存发起的侧信道攻击，目前也未纳入考虑。对于此类缓存攻击，需要额外的硬件单元，如L2缓存控制器，来进行防御。

4 架构设计

4.1 内存隔离与NVRAM安全防护

TPM作为平台的可信根，其自身安全性至关重要。为了确保平台的整体安全性，TPM必须保证运行期内存中代码和数据的安全性。除此之外，确保TPM静态持久性数据的安全性同样重要。TPM内存中包含了TPM的命令处理逻辑以及一些动态易失性数据，例如运行时的一些配置和状态、会话HMAC (Hash-based Message Authentication Code)、临时计数器等。而TPM的持久性静态数据存储在NVRAM中，包括持久性的密钥——EK (Endorsement Key)，永久性的授权值、NV存储计数器等。与硬件TPM利用硬件来保存运行期代码和数据以及NVRAM中的数据不同，fTPM作为软件形态的TPM，必须与宿主机共享硬件资源。在这种情况下，如何有效防御特权攻击者对fTPM内存代码、数据以及静态持久性数据的攻击是一个挑战。

为解决该挑战，RfTPM同时考虑了对fTPM动态数据和静态数据的保护。如图2所示，RfTPM利用基于PMP完成对fTPM内存数据的防护，并通过设计基于受保护的DRAM PUF和Flash锁定机制来保护NVRAM中的静态数据。

4.1.1 基于PMP的内存数据防护

为了保证fTPM运行期内存的安全性，首先构建了一种基于PMP的内存数据防护机制。fTPM启动时会被加载到位于0x80000000到0x80200000的2MB地址空间。如果平台支持16个PMP寄存器，则在fTPM启动时，RfTPM会首先配置最低权限的PMP寄存器，即pmpaddr15和pmpcfg15，以打开整个物理内存的权限。之后，fTPM会配置最高权限的PMP寄存器pmpaddr0和pmpcfg0，从而关闭fTPM所在的内存区域，以防止特权操作系统的攻击。经过以上配置，特权操作系统无法访问fTPM所在的内存空间，而fTPM可以读取或写入操作系统所在的内存地址空间。这种配置不仅保证了内存安全性，还简化了fTPM对TPM命令的处理。在支持IOPMP的平台上，RfTPM可以利用IOPMP机

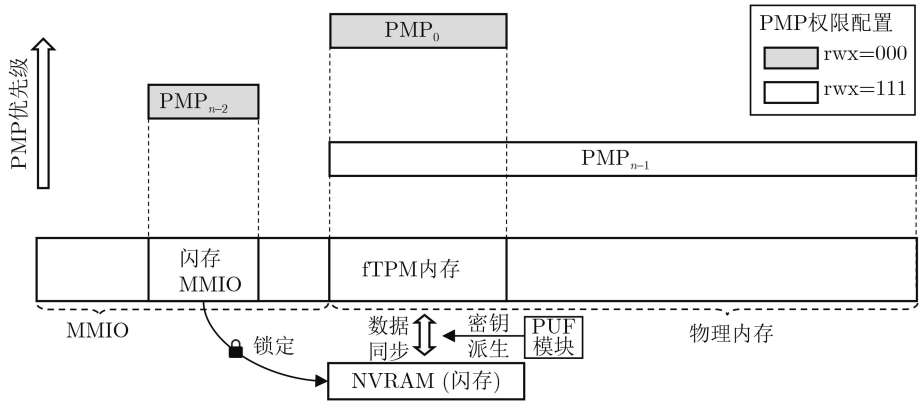

图 2 RfTPM内存与静态数据防护

制来防御DMA攻击，从而进一步提升fTPM内存的隔离性。

4.1.2 基于受保护的DRAM PUF和Flash锁定的NVRAM数据防护

在一些基于TEE的fTPM方案设计中，或者利用非可信主机来实现fTPM非易失性数据的存储，或者要求平台具备一些安全特性，比如RPMB。这些方案不仅丧失了灵活性，也带来了不小的安全风险。在RfTPM中，本文采用了多种防护机制来保护静态数据的安全性。静态数据除了保证机密性外，还需要保证完整性和新鲜性。

对于机密性需求，如果平台没有eFuse, RfTPM支持基于受保护的DRAM PUF来构建安全根密钥，并能够基于该根密钥派生加密密钥来实现对fTPM的NVRAM的加密。DRAM PUF利用了访问DRAM时产生的错误模式。通过一些方式，比如减小定时参数可以引导DRAM的一些单元表现为强1或强0。算法1简述了密钥构建过程。首先，该算法会选取受保护的DRAM内存域作为PUF的构建内存区。为了确保密钥的稳定性，算法通过反复读取DRAM中的物理响应，并根据多个读取结果来统计每个位的稳定性。具体而言，若某一位在多次读取中保持一致 (如大于设定的阈值)，则该位被认为是稳定的，并用于生成最终的密钥。通过重复读取和稳定性判断，算法能够从DRAM响应中提取出具有足够强度的稳定位，从而形成一个高质量的密钥。

算法 1　基于受保护 DRAM PUF 的密钥生成算法

输入：受保护的 DRAM 区域；重读次数repeat_count；位稳定性的阈值threshold
输出：生成的稳定密钥Key

```
Key ← ∅
response_set ← ∅
for i in range(0, repeat_count):
    response ← ∅
    初始化受保护 DRAM 为全“0”或全“1”
    读取受保护 DRAM 状态并记录为 response
    response_set.add(response)
end for
bit_statistics ← 统计每个位在 response_set 中为“1”的频率
for idx in bit_statistics:
    if bit_statistics[idx] ≥ threshold:
        Key.add(1)
    else:
        Key.add(0)
    end if
end for
return Key
```

该算法的优势在于无需依赖外部密钥存储，同时由于PUF所利用的内存域是受到保护的，特权攻击者无法通过控制内存来重新生成该密钥，保证了密钥自身的安全。此外，通过对多个响应的统计与比较，算法能够提高密钥生成的稳定性，确保密钥的可靠性和安全性。算法可以通过进一步引入模糊提取、辅助数据与纠错码技术来提高稳定性和可用性。

尽管加密能够保障fTPM的NVRAM的机密性，但如果特权操作系统能够控制Flash硬件，仍可能发起多种攻击，破坏数据的完整性和新鲜性，例如回滚攻击。为了解决这一问题，本文基于PMP进一步构建了针对Flash的锁定机制以防止这种回滚攻击。在fTPM启动过程中，系统会利用PMP锁定NVRAM所使用的Flash的MMIO地址空间，从而剥夺特权攻击者对该Flash的访问权限。通过这种方式，该Flash尽可以被fTPM访问而无法被非可信的操作系统访问，从而保证完整性和新鲜性。

4.2 基于延迟度量扩展的可信启动

可信启动是TPM的核心安全功能。在硬件TPM中，TPM在系统开始启动前就已经就绪。因此，BIOS可以度量BootLoader (BL)，并将度量结果通过TPM提供的扩展操作扩展到PCR中。之后，BL也可以利用相同的操作流程去度量内核，从而形成一个完整的信任链。在这个过程中，TPM作为存储信任根，存储了启动软件的度量值。

在此过程中，可以发现TPM作为存储信任根需要满足两个要求：一是其度量结果需要受到保护，不能被攻击者破坏；二是TPM需要在系统启动之前就已经处于就绪状态，从而能够满足启动过程中的PCR扩展请求。对于第一个条件，RfTPM可以通过之前章节所述的NVRAM保护机制来实现。而对于第2个挑战，本文发现RfTPM仍然难以满足。这是因为fTPM作为固件的一部分，其启动发生在零阶段BL和一阶段BL之后。这就导致在对一阶段BL进行哈希扩展操作时，fTPM尚未建立，因此无法满足前序的PCR扩展请求。

为了解决该问题，本文在RfTPM中设计了基于延迟度量扩展的可信启动机制。如图3所示，将RfTPM的系统启动阶段划分为验证启动和度量启动两个阶段。在验证启动阶段，与硬件TPM一样，将零阶段BL中的启动代码作为静态度量信任根。RfTPM同样要求前一阶段的启动软件对后一阶段的启动软件进行度量。然而，与硬件TPM不同的

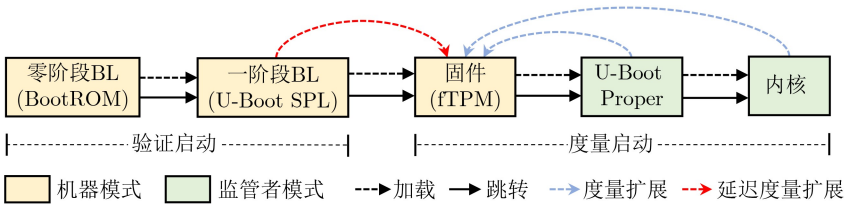

图 3 基于延迟度量扩展的可信启动

是，RfTPM要求前一阶段启动软件根据度量值是否符合预期来决定是否跳转到后一阶段，而不是将哈希扩展到TPM的PCR中。由于在验证启动阶段，fTPM尚未建立，因此对后一阶段启动软件的度量值会在验证成功后直接传递到后一阶段。进入度量启动阶段后，fTPM会将前序阶段传递来的度量值扩展到其PCR中。通过这种方式，虽然RfTPM的fTPM启动模式与硬件TPM有所不同，但仍能保持与硬件TPM一致的可信启动功能。

4.3 基于动态权限页的高效通信

在硬件TPM中，TPM设备通常通过固定的MMIO或PIO接口进行通信。在执行TPM命令时，数据需要从用户态拷贝到内核态的驱动中，再由内核态驱动传递到硬件TPM。这种多次拷贝的数据传输方式导致TPM通信效率相对较低。与此相比，fTPM作为软件形态的TPM，为设计全新的通信机制提供了可能。

然而，为fTPM设计新的通信机制同样面临着安全性和性能方面的挑战。在性能方面，如何在跨越特权级时避免不必要的数据拷贝，是一个关键问题。而在安全性方面，还需额外考虑如何防御TOCTOU (time-of-check-to-time-of-use)攻击。例如，TPM命令的处理通常需要先进行安全检查，在安全检查通过后才会执行命令。然而，在fTPM解析命令期间，另一个核心可能会修改缓冲区中的命令。这种情况会使得TPM命令的安全检查通过但仍然暴露在安全风险之下。

为了解决以上问题，在RfTPM中，本文设计了一种基于动态权限交换页的高效通信机制，旨在确保TPM命令和数据传输的安全性和高效性。该动态权限交换页可以用于实现TPM应用、fTPM驱动与fTPM硬件之间的零拷贝数据传输。同时，通过动态权限配置，可以确保数据处理的安全性。具体而言，在fTPM驱动加载时，它会申请相应的物理内存页作为动态权限交换页，并向fTPM注册。当用户态TPM应用构建TPM命令时，会向fTPM驱动申请内存，驱动将动态权限页映射到用户态TPM的虚拟地址空间中。在fTPM执行时，它会利用PMP关闭动态权限页的权限，从而防止特权操作系统修改其中的数据。当TPM命令执行完成并返回fTPM驱动之前，驱动会再次打开动态权限页的权限，以便上层TPM应用可以获取fTPM的执行结果。

4.4 安全时钟

TPM的时钟在多种安全操作中具有重要意义，例如防止重放攻击、防止回滚攻击、管理密钥生命周期以及实施定时策略等。而若时钟受到非可信端控制，将严重威胁TPM的安全性。

在RfTPM中，由于fTPM运行在最高特权级M模式，选择利用RISC-V平台的计时器来设计安全时钟。mtime是RISC-V平台上的一个64位递增计数器，会以固定频率递增，可用于跟踪系统自上电或复位以来的时间。RISC-V平台通过mtime以及相关的mtimecmp来实现时间中断。RfTPM通过将相关时间接口替换为mtime，形成了一个安全时钟。此外，RfTPM会在适当时机将时钟写入安全存储中，例如在TPM关闭或关键安全操作完成后(以记录安全事件)。在RfTPM启动时，它会将安全存储中的安全时钟恢复，从而保证时钟的安全性。通过以上方式，无论在运行期间，还是在重置或断电期间，RfTPM的时钟都不依赖于外部的非可信软件，从而确保其安全性。

5 系统实现

5.1 fTPM

本文基于OpenSBI构建了fTPM固件，并将ibmswtpm2裁剪后链接到OpenSBI。由于ibmswtpm2利用了OpenSSL实现密码算法，因此本文SL中的crypto库，构建了一个面向fTPM的、可在裸机环境下使用的cryptomini库。然而，由于裸机环境不支持系统调用，OpenSSL中的相关内存操作无法满足。为此，本文静态划分了32 kB的内存空间作为堆，并对堆内存进行了管理，以满足OpenSSL的内存请求。

为了保护NVRAM数据，本文在fTPM中引入SPI驱动，并在此基础上集成FATFS，实现了一个小型文件系统。本文拦截并代理了fTPM中原有的文件操作函数，将其引导至FATFS。由于FATFS部分接口的语义与libc标准的文件接口有所不同，本文码进行了兼容性修改。

为了确保NVRAM机密性，本文在fTPM中实现了AES加解密算法。在fTPM启动时，系统会通过文件系统及SPI驱动读取Flash中的NVRAM文件并加载到内存中。然后，fTPM会使用派生的NVRAM密钥对数据进行解密。解密完成后，fTPM会验证解密后的NVRAM数据是否合法，防止其被篡改。如果验证通过，fTPM将使用解密后的NVRAM数据恢复内存内容。在fTPM成功启动后，如果TPM命令涉及持久化操作，fTPM将使用NVRAM加密密钥对NVRAM数据进行加密，并将其写入Flash中。

5.2 fTPM驱动和TSS

与硬件TPM不同，RfTPM中的fTPM向上层提供的是ECALL接口形式的TPM接口。因此，现有的TPM驱动无法兼容。为此，本文了一个面向RfTPM的驱动，该驱动会注册一个/dev/ftpm设备。当用户态应用操作该接口时，系统态的fTPM驱动会解析相关命令，并通过sbi_ecall调用将命令数据转发给fTPM进行处理。同样的，在ibmtss中，本文也构建了一个新的接口来进行适配新的fTPM驱动。

6　安全性分析

6.1 启动安全性

RfTPM采用的基于延迟度量扩展的安全启动机制能够保证平台启动的安全性和可验证性。在RfTPM的验证启动阶段，前一级启动软件通过对后一级启动软件实施先度量、后验证、再跳转的启动策略来防御攻击者对启动软件的攻击和篡改。如果攻击者篡改了验证启动阶段的任意软件中的代码或数据，则启动不会成功。如果fTPM能够进入初始化阶段，则证明fTPM本身以及其前序的启动软件是受信任的。在fTPM初始化成功并完成对前序启动软件的延迟度量扩展之后，RfTPM会进入度量启动阶段。在该阶段，RfTPM会遵循TPM的安全启动规范。前一级的启动软件会对后一级别启动软件实施先度量、后PCR扩展、再跳转的启动策略。在系统成功启动后，远程认证方可以基于TPM提供的远程认证机制验证平台启动的安全性。由于RfTPM采用了延迟度量扩展机制，所以其PCR会反应包含验证启动阶段和度量启动阶段在内的启动软件的安全性。远程认证方可以据此验证平台从上电到启动完成过程中的启动软件是否被攻击或篡改。

6.2 执行安全性

RfTPM的执行安全性需要同时保证TPM命令执行以及其处理数据的安全性。在fTPM接收到TPM命令请求后，其首先需要解析TPM命令并对其进行安全检查。为了防御TOCTOU攻击，fTPM首先会通过配置PMP关闭数据传递页的权限，防止攻击者在TPM命令安全检查通过后再通过篡改TPM命令来发起攻击。而在TPM命令的执行过程中，由于TPM命令处理的相关代码和数据都受到PMP的保护，特权攻击者无法对TPM命令处理时的所占用的内存或CPU状态进行篡改。通过以上方式，RfTPM保证了TPM命令执行的安全性。

6.3 NVRAM静态数据安全性

RfTPM基于受保护的DRAM PUF构建加密密钥以及Flash隔离技术来保证NVRAM的机密性、完整性和新鲜性。由于NVRAM的加密密钥是基于受保护DRAM内存区的PUF构建的，特权攻击者无法访问这段受保护的内存空间，所以攻击者无法恢复该加密密钥来破坏NVRAM的机密性。进一步，RfTPM通过配置PMP保护Flash的MMIO来锁定该Flash，使其专用于fTPM存储NVRAM静态数据。特权攻击者无法访问NVRAM或发起对NVRAM的回滚攻击，保证了其新鲜性和完整性。

7　测试与评估

7.1 实验设置

本文利用Genesys2 FPGA开发板模拟了Rocket Core。并在其上运行Linux系统，并测试RfTPM的性能。由于模拟的核心频率只有100 MHz，比真实RISC-V硬件核心要低，所以本文结果进行了归一化处理。在对SWTPM和RfTPM的测试中，直接获取执行周期数，并假设运行频率为1 GHz，从而得到相应的执行时间。后续对NVRAM启动以及TPM命令处理中均为执行50次求平均值。

7.2 TPM命令处理

本节在硬件TPM, SWTPM以及RfTPM上对TPM的常用命令进行了测试。其中硬件TPM采用国民科技的TPM芯片，SWTPM则指的是ibm-swtpm2模拟器。

首先，对2048位RSA密钥创建进行了测试。测试结果如图4所示，硬件TPM消耗时间最长，为17.28 s而SWTPM和RfTPM消耗的时间都比较小，分别为1.87 s和2.18 s。由于RSA密钥的生成涉及素数的生成以及大数的模幂运算，这些操作对硬件资源的需求较高。对于硬件TPM来说，由于其资源相比硬件CPU频率低、资源少，所以花费的时间较多。

SWTPM和RfTPM由于直接利用硬件CPU来生成RSA密钥，所以花费的时间要比硬件TPM少

得多，而又由于RSA密钥的生成需要涉及NVRAM的持久化操作。而在RfTPM中，对NVRAM的写入以及额外的加密操作使得其处理时间相比SWTPM要更多一点。尽管相比于没有受到安全保护的SWTPM带来了一定的开销，但RfTPM的安全优势也很明显，相比硬件TPM带来了大约7倍的性能优势。

图5展示了密封、解封、签名和验证方面的开销。在该组测试中，可以看到，RfTPM的性能比SWTPM还要好，性能提升分别为6.9%, 8.2%, 3.7%和6.8%。这主要是由于RfTPM在命令传递上的零拷贝机制更具性能优势。相比硬件TPM, RfTPM在签名上的性能优势比较明显，性能提升大约为6倍。

图6展示了RSA加解密以及AES加解密的性能。在RSA加解密上，RfTPM带来大幅的性能提升。然而，在AES加解密测试上，RfTPM花费时间相比硬件TPM要多一些。这主要是由于当前RISC-V还没有支持硬件级别的AES指令，相比于有硬件支持的硬件TPM开销要多一些。相比于SWTPM，不论在RSA和AES测试中，RfTPM性能都要更高。其中，RSA加密性能提升为8.2%，RSA解密性能提升8%，AES加密性能提升9.1%，AES解密性能提升9.2%。

在本部分测试可以看到，相比于硬件TPM, RfTPM在大部分的测试性能中都要更好，且带来的性能提升比较显著。相比于未受保护的SWTPM，在大部分测试中RfTPM性能也要更好。这充分证明了RfTPM在提高安全性的同时保证了高效性。当前假设RfTPM运行于1GHz，而随着RISC-V处理器时钟频率与IPC的提升，RfTPM的性能将会更加有优势。

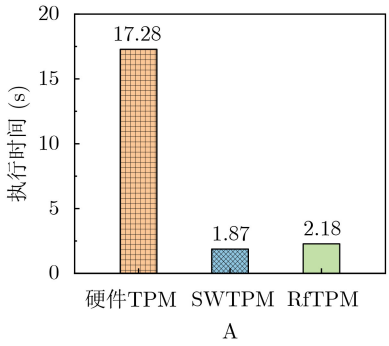

图 4 2048位RSA密钥创建

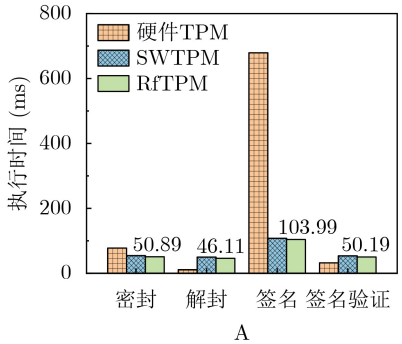

图 5 密封与签名相关TPM命令测试

7.3 NVRAM启动

在NVRAM的启动测试中，分别对RfTPM和SWTPM进行了测试。RfTPM启动时间为5.28 ms，而SWTPM启动时间仅为0.9 ms。RfTPM的开销主要是由于启动时的保护机制以及文件系统的效率两方面带来的。对于SWTPM来说，NVRAM启动的主要操作是读取NVRAM文件中的数据到内存。而对于RfTPM来说，除了读取数据外，RfTPM还额外需要对NVRAM中的数据进行解密。此外，由于RfTPM利用的针对嵌入式的FATFS文件系统，其面向的是资源受限的嵌入式场景，并不是为高性能场景设计。在读取小文件时的性能要比采用Ext-FS的SWTPM差一些。

尽管RfTPM在NVRAM启动方面带来了一些开销，但由于NVRAM的启动时间在毫秒级，且只在平台启动时进行1次，因此其并不会对RfTPM整体性能造成的影响。

7.4 内存占用

在内存占用方面，分别对SWTPM和RfTPM进行了测试。在SWTPM启动后，查看了SWTPM占用的内存空间。SWTPM占用的虚拟内存为20 864 kB，其中大部分作为堆空间。在没有运行TPM命令的情况下实际使用的物理内存为1 536 kB。对于RfTPM， fTPM与OpenSBI编译后的固件所占用的整个内存空间为956 kB。并且在编译时采用Os编译优化级别后，固件大小可以缩减到808 kB。相比之下，RfTPM比SWTPM占用的内存空间要小得

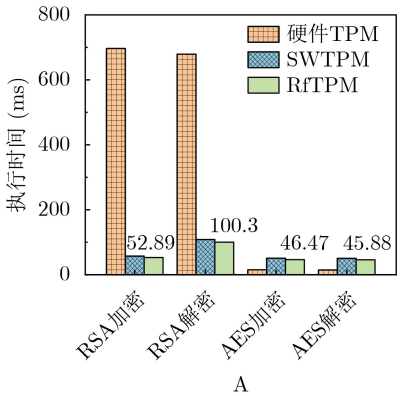

图 6 加密与解密相关TPM命令测试

多。这主要是由于在构建RfTPM时，本文删除了不必要的网络Socket以及线程等模块。除此之外，RfTPM的编译采用了Section GC机制，为每个函数和变量创建独立的Section，在编译和链接时仅引入了必要的函数，而剔除未使用的部分，从而进一步减小了对内存空间的占用。

8　思考与讨论

8.1　侧信道防御

在RfTPM中，为防止fTPM执行过程中遗留在微架构Cache中的信息被攻击者利用而发起侧信道攻击，本文在fTPM执行完成返回时会执行Cache flush操作。然而，对于具有共享L2 Cache的CPU，这种方法的防御效果有限。目前，针对这类攻击主要有两种应对方案。一是修改算法，借助fTPM的可定制化特性，可以将其密钥操作替换为抗Cache侧信道的算法。二是利用某些硬件平台提供的安全能力，例如L2控制器或安全片上内存，从硬件层面消除侧信道攻击的影响。

8.2　安全熵源

安全的熵源对TPM中密码算法的安全强度也会造成影响。尽管RfTPM可以通过PUF在启动时获取高熵随机数，但在fTPM启动后，PUF难以持续为系统提供充足且安全的高熵随机数。同时，纯软件方案也难以构建高安全性的熵源。本文认为，RfTPM的安全熵源需要依赖硬件支持。当硬件平台引入硬件随机数生成器后，RfTPM可以轻松将其集成为一个安全的熵源，进一步提升整体安全性。

9　结束语

本文提出并实现了一种面向RISC-V平台的固件TPM架构——RfTPM，旨在为缺乏硬件TPM支持的RISC-V平台提供等效的安全功能。通过引入创新的设计，包括基于PMP的执行隔离、延迟度量扩展的可信启动机制、动态权限交换页的通信机制以及细粒度的安全时钟机制，RfTPM实现了对TPM核心功能的高效支持，并在不增加硬件成本的情况下，提供了安全存储、可信启动、远程认证等关键安全特性。本文实现了RfTPM原型系统并对其进行了详尽的性能测试，实验结果显示RfTPM在保证安全性的同时具有较高的执行效率。

参 考 文 献

[1] CHALLENER D, YODER K, CATHERMAN R, *et al*. A Practical Guide to Trusted Computing[M]. Upper Saddle River: IBM Press, 2007.

[2] 张焕国, 罗捷, 金刚, 等. 可信计算研究进展[J]. 武汉大学学报: 理学版, 2006, 52(5): 513–518. doi: 10.3321/j.issn:1671-8836.2006.05.001.
ZHANG Huanguo, LUO Jie, JIN Gang, *et al*. Development of trusted computing research[J]. *Journal of Wuhan University: Natural Science Edition*, 2006, 52(5): 513–518. doi: 10.3321/j.issn:1671-8836.2006.05.001.

[3] 沈昌祥, 张焕国, 王怀民, 等. 可信计算的研究与发展[J]. 中国科学: 信息科学, 2010, 40(2): 139–166. doi: 10.1360/zf2010-40-2-139.
SHEN Changxiang, ZHANG Huanguo, WANG Huaimin, *et al*. Research on trusted computing and its development[J]. *Science China Information Sciences*, 2010, 53(3): 405–433. doi: 10.1007/s11432-010-0069-x.

[4] 张焕国, 李晶, 潘丹铃, 等. 嵌入式系统可信平台模块研究[J]. 计算机研究与发展, 2011, 48(7): 1269–1278.
ZHANG Huanguo, LI Jing, PAN Danling, *et al*. Trusted platform module in embedded system[J]. *Journal of Computer Research and Development*, 2011, 48(7): 1269–1278.

[5] ARTHUR W, CHALLENER D, and GOLDMAN K. A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security[M]. Berkeley: Springer, 2015. doi: 10.1007/978-1-4302-6584-9.

[6] SVENDA P, DUFKA A, BROZ M, *et al*. TPMScan: A wide-scale study of security-relevant properties of TPM 2.0 chips[J]. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024, 2024(2): 714–734. doi: 10.46586/tches.v2024.i2.714-734.

[7] RAJ H, SAROIU S, WOLMAN A, *et al*. fTPM: A software-only implementation of a TPM chip[C]. 25th USENIX Security Symposium, Washington, USA, 2016: 841–856.

[8] Intel Corporation. Intel® core™ processors[EB/OL]. https://www.intel.com/content/www/us/en/support/articles/000094205/processors/intel-core-processors.html, 2024.

[9] JACOB H N, WERLING C, BUHREN R, *et al*. faulTPM: Exposing AMD fTPMs' Deepest Secrets[C]. 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P), Delft, Netherlands, 2023: 1128–1142. doi: 10.1109/EuroSP57164.2023.00069.

[10] COSTAN V and DEVADAS S. Intel SGX explained[EB/OL]. https://eprint.iacr.org/2016/086, 2016.

[11] AMD. AMD SEV-SNP: Strengthening VM isolation with integrity protection and more[R]. 2020: 1450–1465.

[12] CUI Enfang, LI Tianzheng, and WEI Qian. RISC-V instruction set architecture extensions: A survey[J]. *IEEE Access*, 2023, 11: 24696–24711. doi: 10.1109/ACCESS.2023.3246491.

[13] LI Tianzheng, CUI Enfang, WU Yuting, *et al*. TeleVM: A lightweight virtual machine for RISC-V architecture[J]. *IEEE Computer Architecture Letters*, 2024, 23(1): 121–124. doi: 10.1109/LCA.2024.3394835.

[14] KIM J S, PATEL M, HASSAN H, *et al*. The DRAM latency PUF: Quickly evaluating physical unclonable functions by exploiting the latency-reliability tradeoff in modern commodity DRAM devices[C]. 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA), Vienna, Austria, 2018: 194–207. doi: 10.1109/HPCA.2018.00026.

[15] TEHRANIPOOR F, KARIMIAN N, XIAO Kan, *et al*. DRAM based intrinsic physical unclonable functions for system level security[C]. Proceedings of the 25th edition on Great Lakes Symposium on VLSI, Pittsburgh, USA, 2015: 15–20. doi: 10.1145/2742060.2742069.

[16] RISC-V Software Source. OpenSBI: RISC-V open source supervisor binary interface[EB/OL]. https://github.com/riscv-software-src/opensbi, 2024.

[17] Chipsalliance. Rocket chip generator[EB/OL]. https://github.com/chipsalliance/rocket-chip, 2024.

[18] Kgoldman. IBM software TPM 2.0[EB/OL]. https://github.com/kgoldman/ibmswtpm2, 2024.

[19] WANG Juan, WANG Jie, FAN Chengyang, *et al*. SvTPM: SGX-based virtual trusted platform modules for cloud computing[J]. *IEEE Transactions on Cloud Computing*, 2023, 11(3): 2936–2953. doi: 10.1109/TCC.2023.3243891.

[20] NARAYANAN V, CARVALHO C, RUOCCO A, *et al*. Remote attestation of confidential VMs using ephemeral vTPMs[C]. The 39th Annual Computer Security Applications Conference, Austin, USA, 2023: 732–743. doi: 10.1145/3627106.3627112.

[21] WU Jiangxing. Cyberspace endogenous security and safety problems[M]. WU Jiangxing. Cyber Resilience System Engineering Empowered by Endogenous Security and Safety. Singapore: Springer, 2024: 1–73. doi: 10.1007/978-981-97-0116-2_1.

[22] CHEN Hongsong, HAN Xintong, and ZHANG Yiying. Endogenous security formal definition, innovation mechanisms, and experiment research in industrial Internet[J]. *Tsinghua Science and Technology*, 2024, 29(2): 492–505. doi: 10.26599/TST.2023.9010034.

[23] GUO Jinnan, PIETZUCH P, PAVERD A, *et al*. Trustworthy AI using confidential federated learning[J]. *Communications of the ACM*, 2024, 67(9): 48–53. doi: 10.1145/3677390.

[24] CHEN Hongsong, TAO Zimei, WANG Zhiheng, *et al*. Merkle multi-branch hash tree-based dynamic data integrity auditing for B5G network cloud storage[J]. *Journal of Information Security and Applications*, 2025, 89: 103981. doi: 10.1016/j.jisa.2025.103981.

[25] FRITZMANN T, SIGL G, and SEPÚLVEDA J. RISQ-V: Tightly coupled RISC-V accelerators for post-quantum cryptography[J]. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020, 2020(4): 239–280. doi: 10.13154/tches.v2020.i4.239-280.

[26] DE CASTELNAU J. Software optimization for a RISC-V accelerator: A case study[EB/OL]. https://infoscience.epfl.ch/server/api/core/bitstreams/472275ca-4a0a-4f5b-831d-1082a77f98f2/content, 2024.

[27] SCHIAVONE P D, CONTI F, ROSSI D, *et al*. Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for internet-of-things applications[C]. 2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), Thessaloniki, Greece, 2017: 1–8. doi: 10.1109/PATMOS.2017.8106976.

[28] LEE D, KOHLBRENNER D, SHINDE S, *et al*. Keystone: An open framework for architecting trusted execution environments[C]. The Fifteenth European Conference on Computer Systems, Heraklion, Greece, 2020: 38. doi: 10.1145/3342195.3387532.

[29] LIU Chang, WU Yanjun, WU Jingzheng, *et al*. A buffer overflow detection and defense method based on RISC-V instruction set extension[J]. *Cybersecurity*, 2023, 6(1): 45. doi: 10.1186/s42400-023-00164-x.

王 杰：男，博士生，研究方向为可信计算和机密计算.

王 鹃：女，教授，博士生导师，研究方向为系统安全与AI安全.

责任编辑：马秀强

The Design and Implementation of a Secure and Efficient Firmware Trusted Platform Module for RISC-V Platforms

WANG Jie WANG Juan

(*Key Laboratory of Aerospace Information Security and Trusted Computing*, *Ministry of Education*, *Wuhan* 430072, *China*)

(*School of Cyber Science and Engineering*, *Wuhan University*, *Wuhan* 430072, *China*)

Abstract:

Objective The Trusted Platform Module (TPM) is a critical technology in modern secure computing systems, providing hardware-based key management, trusted boot, and remote attestation to safeguard sensitive

operations in embedded and cloud environments. However, current RISC-V platforms lack native TPM support, presenting a security challenge as these systems are increasingly deployed in diverse application scenarios. To address this limitation, RfTPM—a firmware-based TPM (fTPM) architecture—has been developed to deliver the same security functionality as conventional hardware TPMs without requiring additional hardware components or specialized security extensions. This solution provides an immediate, cost-effective means to secure RISC-V systems while contributing to the advancement of trusted computing on emerging processor architectures.

Methods The development of RfTPM incorporates several innovative techniques to overcome the challenges of implementing TPM functionalities in firmware. The design utilizes the RISC-V Physical Memory Protection (PMP) mechanism to enforce strict memory isolation, ensuring that fTPM code and data are inaccessible to unauthorized processes. A novel static data protection strategy is introduced, combining a DRAM-based Physically Unclonable Function (PUF) with Flash locking to secure the generation and storage of cryptographic root keys, preventing rollback attacks on persistent fTPM data. To secure the boot process, RfTPM employs a delay measurement extension mechanism, which divides the boot sequence into two phases: a verification phase where each boot stage is measured and authenticated before control is transferred, and a subsequent measurement phase that continuously validates system integrity according to TPM standards. The architecture also features a dynamic permission exchange page, enabling zero-copy communication across different privilege levels by dynamically configuring PMP permissions, reducing data transfer overhead. Additionally, a fine-grained secure clock is established using the native RISC-V hardware timer to counter timing-based attacks. The solution is prototyped as a secure extension module within OpenSBI, integrated with a dedicated kernel driver and an adapted TPM Software Stack (TSS), and evaluated on a Genesys2 FPGA board simulating a Rocket Core running Linux.

Results and Discussions Comprehensive experimental evaluations demonstrate that RfTPM meets stringent security requirements while offering significant performance benefits over both traditional hardware TPMs and conventional software TPM implementations. In a benchmark involving 2048-bit RSA key generation (Fig. 4), the hardware TPM required approximately 17.28 seconds to complete the operation, whereas the RfTPM implementation achieved the same task in just 2.18 seconds, representing a 7 times improvement. Further tests evaluating sealing, unsealing, signing, and verification commands (Fig. 5) reveal performance enhancements ranging from 3.7% to 8.2%, primarily due to the efficiency of the zero-copy communication mechanism. Additional evaluations of cryptographic operations show that RfTPM improved RSA encryption and decryption by 8.2% and 8.0%, respectively, and AES encryption and decryption by 9.1% and 9.2% (Fig. 6). Although the NVRAM startup process in RfTPM incurs minor overhead—measured at 5.28 milliseconds compared to 0.9 milliseconds for conventional software TPMs—this delay is negligible, as NVRAM initialization occurs only once during system boot and does not impact overall runtime performance. Memory footprint analysis further reveals that while conventional software TPMs may consume approximately 1 536 kB of physical memory, the combined footprint of the fTPM and OpenSBI firmware in RfTPM is only 956 kB, which can be reduced to 808 kB through compiler optimizations. These results collectively confirm that RfTPM not only provides robust defense against various security threats, including TOCTOU and rollback attacks, but also enhances operational efficiency, making it an optimal solution for secure computing on RISC-V platforms.

Conclusions In summary, RfTPM represents the first comprehensive firmware-based TPM architecture specifically tailored for RISC-V platforms, effectively addressing critical challenges such as secure execution, trusted boot integrity, efficient inter-layer communication, and precise timekeeping without incurring additional hardware costs. By integrating advanced techniques—including PMP-based memory isolation, DRAM PUF-enhanced static data protection, a dual-phase boot process with delay measurement extension, dynamic permission exchange for zero-copy communication, and a hardware-based secure clock—RfTPM delivers robust security functionality that matches or exceeds that of traditional hardware TPMs. Experimental results confirm that RfTPM upholds rigorous security standards while offering substantial performance and resource utilization advantages over both hardware TPMs and existing software TPMs. The open-sourcing of core components further fosters community collaboration and provides a platform for future research focused on refining trusted computing solutions for emerging architectures like RISC-V. Future work may explore additional hardware optimizations, such as native AES instruction support, and further enhancements to file system performance to increase the efficiency and robustness of fTPM implementations.

Key words: Trusted Platform Module (TPM); Firmware Trusted Platform Module (fTPM); RISC-V; Secure Boot