LayerTEE: Decoupled Memory Protection for Scalable Multi-Layer Communication on RISC-V

Shangjie Pan, Yinghao Yang, Xuanyao Peng, Xiqun Zhao, Dong Du, Hang Lu, Yubin Xia, *Member, IEEE*, Xiaowei Li, *Senior Member, IEEE*

***Abstract*—The Trusted Execution Environment (TEE) has been widely implemented by modern hardware vendors to protect security and privacy-sensitive applications and data, such as Intel SGX/TDX, ARM TrustZone, AMD SEV, and RISC-V Penglai. However, existing TEE systems face challenges in balancing memory isolation among security, performance, and scalability requirements. Segment-based memory isolation mechanisms, like RISC-V PMP, struggle to scale effectively to the large number of segments needed for confidential cloud and data center environments. On the other hand, table-based isolation methods, such as page tables, combine address translation with memory protection, leading to inefficient cross-enclave communication and potential security vulnerabilities like Rowhammer attacks.**

This paper introduces a novel TEE system, LayerTEE, which decouples memory protection (to segments) from address translation (to page tables). This design improves communication performance by dynamically adjusting memory protection capabilities, without sacrificing application compatibility. LayerTEE enhances enclave security and scalability by designing a multi-layer segment-based isolation mechanism. We have built a prototype of LayerTEE based on FPGA, incorporating hardware extensions and software support. The evaluation demonstrates that LayerTEE significantly surpasses existing TEE solutions, achieving three orders of magnitude lower communication latency and 10x greater scalability while maintaining robust security guarantees.

***Index Terms*—Trusted execution environment (TEE), memory isolation, communication, RISC-V**

I. Introduction

With the rapid growth of cloud computing, traditional software applications are increasingly migrating to the cloud, presenting new challenges for data security and privacy protection [3], [8], [9], [10], [31]. Trusted execution environments (TEEs) provide a secure computing environment, offering a hardware/software co-design protection solution to safeguard sensitive data and applications in cloud environments [15], [21], [25], [30], [33], [48]. Enclaves, essential security containers within TEEs, protect critical application parts from malicious attacks and software vulnerabilities.

In the cloud environment, enclaves often collaborate to manage distributed computing tasks or share sensitive data. Secure and efficient inter-enclave communication is essential for completing complex operations. For example, multiple hospitals might share patient data to train a machine-learning model for disease diagnosis, thereby enhancing the model's performance [51]. Collaboration among multiple cloud service providers has led to the development of an advanced platform that enables flexible and efficient communication between applications [35], [39]. This platform supports interactions both within a single cloud provider and across multiple providers, promoting flexibility and cooperation [16], [29], [37], [46], [49].

Studies have shown that utilizing multiple independent enclaves to map tasks with untrusted components to distributed enclaves requires costly communication channels through untrusted memory areas, which can introduce potential vulnerabilities [30]. Cloud service providers rent cloud computing machines with Intel SGX [36] to offer dedicated enclave processes for each client [1]. However, the limited secure memory (PRM) [19] in SGX results in significant overhead due to multiple copying and encryption/decryption operations during inter-enclave communication. While Elasticlave [52], based on the Keystone framework [33], supports inter-enclave communication, its scalability is constrained by hardware limitations, making it unsuitable for cloud environments. Therefore, improving the scalability and performance of inter-enclave communication while ensuring data security is crucial.

Current TEE communication mechanisms face several limitations. We analyze these mechanisms and classify them into copy+encryption, remapping, and shared memory. While secure, the copy+encryption mechanism introduces significant performance overhead due to repeated data copy and encryption/decryption processes. Though more efficient, the remapping mechanism can suffer security vulnerabilities due to improper memory isolation. The shared memory mechanism offers high performance but compromises scalability and poses potential security risks, as shared memory regions can become targets for various attacks. These limitations highlight the need for a more robust solution that effectively balances performance, scalability and security.

To address these challenges, we propose a novel TEE communication system, termed LayerTEE. This system leverages the RISC-V physical memory protection (PMP) [7] and supervisor-mode physical memory protection (SPMP) mechanisms [27], combined with the shared memory-based communication mechanism. Specifically, PMP registers isolate a designated memory block known as the TEE Region. Within the TEE Region, a pages table mechanism ensures the isolation of individual enclaves. LayerTEE uses the SPMP mechanism to decouple page table permission protection, enhancing security and scalability. The shared memory, isolated via SPMP registers, facilitates secure inter-enclave communication, effectively mitigating attacks targeting the page table. Additionally, LayerTEE incorporates a series of communication interfaces integrated

Corresponding author is Hang Lu. This work was supported in part by the National Natural Science Foundation of China under Grant 62172387; in part by the CCF-Phytium Fund 2023.

TABLE I
A COMPARISON OF TEE COMMUNICATION MECHANISMS. *Isolation* MEANS THE ISOLATION MECHANISM AMONG ENCLAVES. *Unrestricted* MEANS THE NUMBER OF ENCLAVES IS UNRESTRICTED, THOUGH SYSTEM PERFORMANCE MAY DECLINE IF SECURE MEMORY IS INSUFFICIENT.PERFORMANCE IS INDICATED BY ARROWS: ↑ (HIGH), ↓ (LOW), AND → (MEDIUM). *R.* AND *T.* DENOTE THE SECURITY RISKS FROM ROWHAMMER AND TOCTTOU ATTACKS. LAYERTEE UNIQUELY OFFERS SCALABLE, HIGH-PERFORMANCE, AND SECURE INTER-ENCLAVE COMMUNICATION.

System				**Inter-Enclave Communication**			
Name	**Arch**	**Isolation**	**Scalability**	**Mechanism**	**Performance**	**Security**	**Inter-TEE Region**
SGX [19], [36]	Intel	Table-based	Unrestricted	Copy+encryption	↓	✓	✗
Data-Enclave [51]	Intel	Table-based	Unrestricted	Shared memory	↑	R. & T.	✗
TDX [31]	Intel	Table-based	Unrestricted	Shared memory	→	R. & T.	✗
SEV [3]	AMD	Table-based	Unrestricted	Shared memory	→	R. & T.	✗
CCA [8]	ARM	Table-based	Unrestricted	Shared memory	→	R. & T.	✗
Komodo [26]	ARM	Table-based	Unrestricted	Remapping	→	R.	✗
Sanctuary [14]	ARM	Table-based	Unrestricted	Shared memory	↑	R. & T.	✗
Sanctum [20]	RISC-V	Segment-based	DRAM Regions	Copy+encryption	↓	R.	✗
Elasticlave [52]	RISC-V	Segment-based	(PMPs-2)/2	Shared memory	↑	✓	✗
Penglai-TVM [25]	RISC-V	Table-based	Unrestricted	Remapping	→	R.	✗
Penglai-PMP [24]	RISC-V	Segment-based	PMPs	Copy+encryption	↓	✓	✗
LayerTEE	RISC-V	Decoupled	Unrestricted	Shared memory	↑	✓	✓

with the SPMP mechanism to secure the communication process, thereby defending against time-of-check to time-of-use (TOCTTOU) attacks [22]. Our main contributions are summarized as follows:

- We design a novel TEE communication system named LayerTEE. LayerTEE uses the shared memory mechanism for inter-enclave communication and leverages the SPMP mechanism to decouple permission protection from the enclaves' page tables, achieving high security and scalability.
- We propose a multi-layer communication pattern, distinguishing between intra-TEE Region and inter-TEE Region communications. The former is suitable for efficient data transfer between "closer" enclaves, whereas the latter secures data exchange across "more distant" enclaves using shared memory Region and extra interfaces.
- We implement a prototype of LayerTEE on the Penglai-based platform and integrate it into the high-performant open-source RISC-V processor core, Xiangshan Nanhu Core. Evaluation on the Xilinx VU19P FPGA shows that LayerTEE requires minimal additional hardware resources, using only 0.2% more LUTs and 0.04% more FFs. It securely and efficiently supports communication among at least 100 pairs of enclaves, demonstrating performance improvements of one to three orders of magnitude over the Penglai-PMP system across various data transfer sizes.

We organize this paper as follows. Section II introduces the communication mechanisms in existing TEE systems and their limitations. Section III overviews the LayerTEE framework. Section IV details the design. Section V details the LayerTEE implementation. Section VI presents the system evaluation and security analysis. Section VII concludes this work.

II. MOTIVATION

A. *Inter-Enclave Communication Mechanisms*

Inter-process communication (IPC) is crucial in modern OSes for better modularity and is increasingly important for TEE systems [34], [38], [41], [43]. IPC facilitates functions decomposition and collaborative work, supports complex application scenarios, and enables multitasking [1], [30], [51]. A secure, efficient, and scalable communication system is crucial in cloud computing; however, current inter-enclave communication schemes fall short of meeting these requirements.

Existing inter-enclave communication mechanisms include copy+encryption, remapping, and shared memory, as summarized in Table I. TEE systems like Intel SGX [36], RISC-V Sanctum [20], and Penglai-PMP [24] use the copy+encryption mechanism, where data is encrypted and transmitted to untrusted memory by sender enclave and then decrypted by the receiver enclave. This mechanism ensures data confidentiality and integrity but incurs high overhead due to the encryption/decryption and copying processes. ARM Komodo [26] and RISC-V Penglai-TVM [25] adopt the remapping mechanism, where enclaves unmap and remap physical memory for communication. This mechanism can defend against TOCTTOU attacks but is not ideal in efficiency due to its coarse communication granularity (4KB page units) and significant overhead (e.g., TLB shootdown issues [6]). Intel TDX [31] AMD SEV [3], and ARM CCA [8] map shared memory via page tables in a virtual machine, avoiding performance loss from copying and remapping but leaving page tables vulnerable to attacks like Rowhammer [2] and TOCTTOU [22]. Moreover, their isolation is based on permission tables, which introduces additional memory references, resulting in significant performance overhead [23]. Data-Enclave [51] maps the physical memory of one enclave as shared memory to the communicating enclave, also potentially susceptible to Rowhammer and TOCTTOU attacks. High-concurrency demands in cloud environments amplify these security risks. Elasticlave [52], based on Keystone [33] and RISC-V PMP [7], improves security by isolating shared memory but struggles with high-concurrency communication due to resource constraints. Additionally, existing TEE communication systems universally lack support for inter-enclave communication across different TEE Regions, limiting their scalability and flexibility

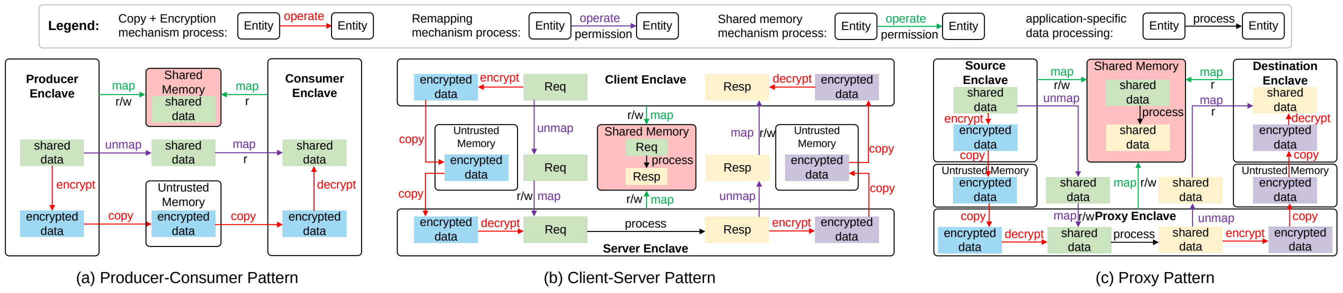

Legend:
Copy + Encryption mechanism process: Entity operate Entity
Remapping mechanism process: Entity operate permission Entity
Shared memory mechanism process: Entity operate permission Entity
application-specific data processing: Entity process Entity
Producer Enclave
Shared Memory
Consumer Enclave
shared data
encrypted data
Untrusted Memory
map
r/w
unmap
encrypt
copy
decrypt
Client Enclave
Server Enclave
Req
Resp
process
Source Enclave
Destination Enclave
Proxy Enclave
(a) Producer-Consumer Pattern
(b) Client-Server Pattern
(c) Proxy Pattern

Fig. 1. In-depth analysis of implementation processes for three distinct inter-enclave communication mechanisms across various bilateral communication patterns.

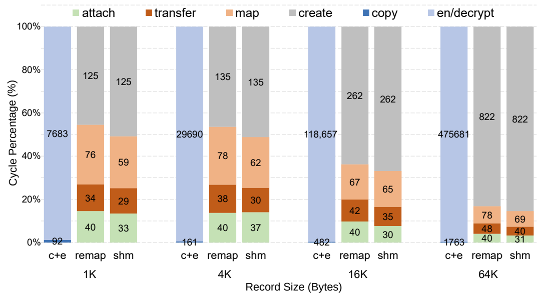


Fig. 2. Breakdowns of three communication mechanisms within the producer-consumer pattern. Data labels are proportionally scaled to reflect actual measurements and are expressed in cycles.

in cloud computing environments.

B. Inter-Enclave Communication Patterns

1) Bilateral Communication Patterns: We apply the communication mechanisms discussed in §II-A to three representative bilateral data sharing patterns in real-world scenarios as outlined in [52], as shown in Fig. 1.

Pattern 1: Producer-Consumer. In this pattern, the producer enclave transfers data to the consumer enclave (Fig. 1(a)). This method is useful for signalling the completion of sub-tasks in larger processes, such as batch processing scripts in web frameworks [28], [45].

To analyze the performance overhead and underlying causes of different communication mechanisms, we conducted latency tests within the producer-consumer pattern using three distinct communication mechanisms across various record sizes. The results (Fig. 2) indicate that the copy+encryption mechanism incurs the highest overhead due to extensive copying and the encryption/decryption process. The remapping mechanism shows moderate overhead from remapping and unmapping operations, which causes TLB shootdown issues. The shared memory mechanism has the lowest overhead, with minimal instruction level and read/write costs, which remain consistent regardless of data size.

Pattern 2: Client-Server. In this pattern, the client and server enclaves concurrently read and write shared data for exchange, as shown in Fig. 1(b). Implementing this pattern with the copy+encryption mechanism involves four data copy operations and two pairs of encryption/decryption processes. The remapping mechanism requires two unmapping and two mapping operations on the shared data memory. Conversely, the shared memory mechanism necessitates only two mapping operations on the shared data memory.

Pattern 3: Proxy. In this pattern, a proxy enclave processes data from a source enclave and then forwards it to a destination enclave, as shown in Fig. 1(c). For instance, a caching proxy in a web service can store responses to frequent requests and modify incoming requests (e.g., Nginx [40]). Implementing this pattern with the copy+encryption mechanism requires four data copying operations and two pairs of encryption and decryption processes. Utilizing the remapping mechanism involves two unmapping and two mapping operations on the shared data memory. Alternatively, employing the shared memory mechanism necessitates only three mapping operations on the shared data memory.

2) Multilateral Communication Patterns: In a cloud environment, multiple concurrent enclaves work together to execute complex tasks, often involving bilateral communication patterns. In these situations, the TEE system must support the simultaneous operation of many enclaves while maintaining both performance and security. We propose two multilateral communication patterns to address these needs: multi-bilateral communication and single-producer multi-consumer.

Pattern 4: Multi-Bilateral Communication. In this pattern, multiple pairs of enclaves engage in one-to-one communications simultaneously. This pattern encompasses various bilateral communication scenarios. For example, consider a financial services application where multiple producer enclaves generate financial data and transmit it to corresponding consumer enclaves for analysis.

Pattern 5: Single-Producer Multi-Consumer. In this pattern, a producer enclave processes data and transmits it to multiple consumer enclaves. For instance, a healthcare firm can provide personalized recommendations based on customer-provided information, such as medical histories [1]. This enables multiple clients to query a shared database without disclosing personal health details.

By supporting multilateral communication patterns, the TEE system can effectively manage complex interactions among enclaves. Using the copy+encryption mechanism to achieve multilateral communication patterns involves numerous copying and encryption/decryption steps, significantly reducing communication performance. Similarly, remapping mechanisms require multiple unmapping and mapping page tables, increasing the chances of TLB shootdowns and decreasing efficiency. In contrast, shared memory mechanisms enhance performance by reducing these overheads and enabling more efficient data exchange among enclaves.

3) Multi-layer Communication Pattern: In cloud computing environments, multiple service providers require separate regions to run their tenants' sensitive applications independently. This isolation enhances data security, optimizes resource management, and improves overall system security [16]. In this scenario, multiple enclaves often require communication across different physical and logical regions to perform complex tasks. Traditional communication patterns may not efficiently support these cross-region interactions, leading to performance bottlenecks and security challenges. To address these issues, we propose a multi-layer communication pattern. This pattern can initially support multiple TEE Regions, allowing different cloud providers to run their sensitive applications securely. Secondly, it facilitates inter-enclave communication both within a single TEE Region and between different TEE Regions. By leveraging the multi-layer communication pattern, we can improve enclave interactions' flexibility, performance and security across different cloud environments.

C. Challenges

Challenge-1: Scalability and security over shared memory. Using shared memory for inter-enclave communication, although it provides optimal performance, poses significant challenges related to scalability and security. Hardware resources such as RISC-V PMP can isolate each enclave and shared memory to ensure security. However, the limited number of PMP registers (typically only 16) restricts the number of enclaves and shared memory blocks that can be supported, thus affecting the scalability of inter-enclave communication. To improve scalability, page tables can isolate enclaves and shared memory blocks, and map shared memory to the enclaves needing communication. However, this approach is susceptible to TOCTTOU attacks, where attackers exploit state changes in shared memory data between the check and use phases. Additionally, attackers might use techniques like Rowhammer to tamper with page tables, potentially mapping shared memory to malicious entities, leading to data leakage or tampering. Balancing performance, scalability, and security is the primary challenge in designing efficient and secure inter-enclave communication mechanisms.

Challenge-2: Secure and efficient data exchange across the TEE Regions. Multi-TEE Region systems allow developers to deploy sensitive applications in separate regions, enhancing flexibility and security. Hardware resources like RISC-V PMP can partition memory into multiple TEE Regions to achieve physical isolation. However, maintaining secure isolation among multiple enclaves within a single TEE Region remains challenging. Collaboration between enclaves across different TEE Regions enables efficient handling of complex tasks. Traditional copy+encryption mechanisms ensure secure inter-TEE Region communication but have significant performance overhead. Shared memory is a more efficient communication method, but allocating it within a TEE Region for a specific enclave requires granting access permissions to other enclaves, potentially compromising their security. Thus, the main challenges in inter-TEE Region communication are the secure creation of shared memory and the design of communication interfaces. In summary, designing a flexible and efficient communication mechanism while ensuring secure isolation is a critical challenge for multi-TEE Region systems.

The key insights to resolve the above challenges are **decoupling protection (based on segment) from sharing (based on page tables)** and introducing a layered segment-based isolation design with secure interfaces for multi-layer communication. First, the decoupling can achieve high security without compromising performance. The system only relies on register-based segments for security protection, which can defend against attackers targeting page tables (e.g., Rowhammer). Second, we introduce a layered segment-based isolation design that can achieve scalable segments compared to existing single-layer designs. Finally, we design a set of secure interfaces, combined with isolation mechanisms, to implement the TEE multi-layer communication pattern.

III. Architecture Overview

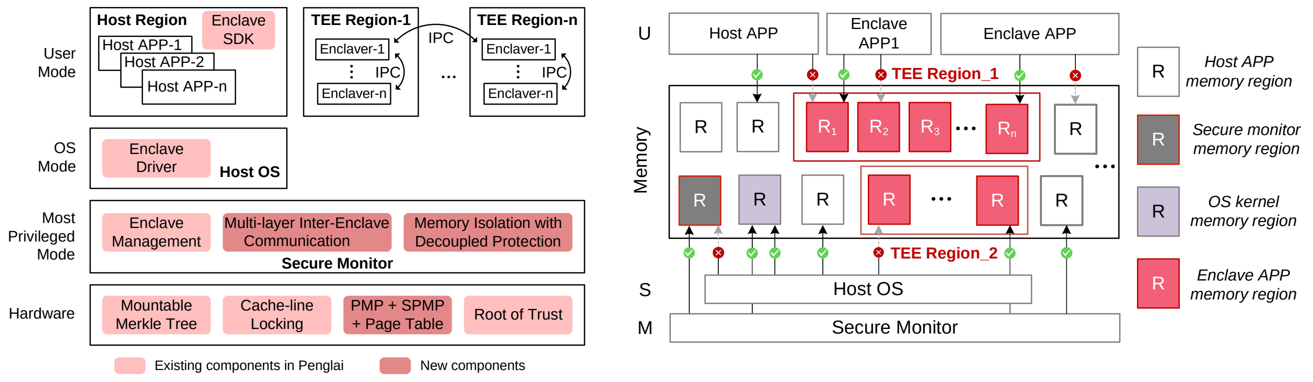
A. Architecture

We highlight two key aspects of LayerTEE abstraction: (1) memory isolation with decoupled protection and (2) multi-layer inter-enclave communication. These abstractions are integrated into a secure monitor operating in the highest privilege mode, such as machine mode in RISC-V, as illustrated in Fig. 3(a). The secure monitor comprehensively manages all enclaves within the system. It provides a robust set of APIs, enabling users to seamlessly deploy, manage, and interact with enclaves. Additionally, the secure monitor handles the lifecycle of enclaves, including their creation, destruction, and context switching, ensuring that the system remains safe and efficient during operation. During system startup, the secure monitor is loaded and authenticated by the boot ROM to ensure its integrity and trustworthiness.

We utilize the RISC-V PMP mechanism to isolate secure physical memory regions, referred to as TEE Regions. Additionally, we introduce the RISC-V SPMP [27] mechanism to decouple permission protection from the enclaves' page tables within a TEE Region, thereby defending against attacks targeting the page tables. As shown in Fig. 3(b), the secure monitor operates in machine mode (M-mode) and can access all memory. The OS runs in supervisor mode (S-mode) and is prohibited from accessing the secure monitor and TEE Regions memory. Host applications and enclaves run in user mode (U-mode) and can only access their own memory. When enclaves are deployed within the same TEE Region, they are physically closer, and their interactions are classified as "intra-TEE Region communication". If they belong to different TEE Regions, they are more distant, and their interactions are defined as "inter-TEE Region communication".

This layered approach ensures the confidentiality of the data within the TEE Regions.

We introduce several interfaces to facilitate inter-enclave communication within and across different TEE Regions. Combined with the SPMP mechanism, these interfaces enable efficient and secure inter-enclave communication. This



(a) System Architecture overview. (b) Memory access permission for different entities.

Fig. 3. The system architecture and memory access control of LayerTEE. (a) LayerTEE provides versatile inter-enclave communication and decoupled protection over SOTA TEE systems like Penglai. (b) The secure monitor (M-mode) has full access. The OS (S-mode) is restricted from the secure monitor and TEE Regions. Host apps and enclaves (U-mode) are confined to their own regions.

mitigates TOCTTOU attacks by protecting the communication process from timing vulnerabilities through the shared memory mechanism. These interfaces enhance the TEE system's flexibility and security, supporting scalable and secure multi-enclave applications.

B. Threat Model

The trusted computing base (TCB) of LayerTEE includes the hardware CPU and the secure monitor. We assume that attackers possess extensive knowledge of the OS, hardware components, and network configuration. They are also capable of creating and executing malicious enclaves within the system. Using these vulnerabilities or malicious enclaves, they could intercept and manipulate messages between the CPU and other hardware components, potentially leading to unauthorized data access and system manipulation. Furthermore, attackers might employ specific techniques such as TOCTTOU attacks and attacks targeting page tables, like Rowhammer, to extract sensitive information. These attacks can compromise the integrity and confidentiality of the system's data by exploiting the race condition time window and memory vulnerabilities. The side-channel attacks (e.g., timing or power analysis attacks) pertain to distinct security dimensions and require specialized defense strategies, which fall outside the scope of this work's focus on communication security mechanisms. Concerning denial-of-service (DoS) attacks, our threat model inherently relies on the assumption that the host OS adheres to standard resource management protocols, as memory reclamation and similar operations ultimately depend on host OS privileges. The definition of threat model aligns with established conventions in TEE research [25], [33], [36], [52].

IV. LayerTEE Design

This section primarily examines how the secure monitor leverages hardware extensions alongside the proposed communication interfaces to achieve the design goals.

A. Isolation with Decoupled Protection

RISC-V PMP Background. RISC-V's physical memory protection (PMP) mechanism divides physical memory into distinct regions, each with specific access permissions (read, write, execute) [24]. These regions are managed using the `pmpaddr` and `pmpcfg` registers, which are control and status registers (CSRs) and can only be modified in M-mode. Each memory region is defined by a continuous address range, specified by two address registers or a single address register with an embedded size. When memory access is requested from S-mode or U-mode, the hardware checks if the target address falls within a defined PMP region and verifies the corresponding permissions. M-mode configures PMP regions to be non-readable, non-writable, and non-executable to isolate its memory from S/U-mode. Additional regions are set up for the operating system and user applications, allowing appropriate access while maintaining M-mode isolation. PMP enforces a strict separation between M-mode and S/U-mode, enhancing system security and integrity.

Scalability Challenges. RISC-V-based TEEs like Keystone and Penglai-PMP use the PMP mechanism to isolate enclaves. In M-mode, PMP includes 16 sets of address and configuration registers. These registers define physical memory regions' size, location, and access permissions. Memory access is granted only if PMP checks pass; otherwise, access is denied. Each enclave's isolation requires a dedicated set of PMP registers, but the limited number of PMP registers restricts the concurrent enclaves. This limitation poses a scalability challenge for RISC-V-based TEEs in cloud environments that require handling many tasks simultaneously.

Table-based Memory Isolation. Within a TEE Region isolated by PMP registers, a page table mechanism is employed to isolate enclaves further, as shown in Fig. 4. Each enclave has its own page tables stored in dedicated memory, which the secure monitor configures to map to its specific memory space. This approach supports the simultaneous creation of multiple enclaves within the same TEE Region. In the Sv39 page table structure (3-level page table), hardware memory

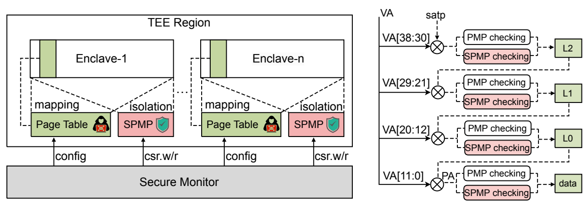


Fig. 4. Isolation and access checks after decoupling permission protection from page tables in the TEE Region.

access requires four references: three for the page table pages and one for the data pages. Each access involves a PMP check to verify permission. However, this approach is vulnerable to attacks targeting page tables like Rowhammer, which can leak sensitive data. To counter this, we introduce the RISC-V SPMP mechanism [27] to decouple permission protection from page tables, enhancing the isolation and protection of enclaves within the TEE Region.

RISC-V SPMP Mechanism. SPMP is a segment-based design similar to RISC-V PMP. It supports 16 entries, each consisting of an address register and a configuration register. The SPMP entries can define the range of physical memory regions managed, with the range limited by the A field in the configuration register and the address register. The access permissions for these regions are determined by the X (execute), W (write), and R (read) fields in the configuration register. The S field of the configuration marks a rule as S-mode-only when set and U-mode-only when unset.

TEE Region Memory Management. When creating an enclave within a TEE Region, the secure monitor allocates a contiguous physical memory region, sets up the page tables, and assigns metadata containing the enclave's information. This metadata, stored in an isolated memory area managed by the secure monitor, includes the enclave's unique identifier (eid), starting physical address, memory size, SPMP register context, and more. During enclave creation, the secure monitor configures the SPMP context based on the allocated physical address, size, and access permissions. When the enclave is executed, the secure monitor uses this SPMP context to write into the SPMP address and configuration registers via the `CSR.write` instruction, ensuring the correct memory access permissions. When switching to another enclave, the secure monitor retrieves the target enclave's SPMP context using its eid and loads it into the SPMP registers. This dynamic adjustment of SPMP register configurations ensures strict isolation among different enclaves. Thus, within the TEE Region, the management and switching of the SPMP register context effectively facilitate the physical memory isolation of multiple enclaves.

Using segment-based SPMP registers, we decouple permission protection from page tables, establishing a secure and scalable enclave isolation mechanism. As shown in Fig. 4, the system performs an SPMP check alongside each PMP check, allowing memory access only if both checks pass. This approach enhances security and efficiency by storing permission settings directly in registers and conducting checks internally within the CPU. Additionally, LayerTEE can create multiple TEE Regions, each capable of hosting unrestricted enclaves. Enclaves that communicate frequently ("close enclaves") can be created within the same TEE Region, while those with less frequent communication ("distant enclaves") can be created in different TEE Regions. This approach highlights LayerTEE's scalability, security, and adaptability.

TABLE II
INTER-ENCLAVE COMMUNICATION INSTRUCTIONS IN LAYERTEE. P MEANS ACCESS PERMISSION TO ANOTHER ENCLAVE.

Instructions	Semantics
shmid = create(key, size)	create a shared memory
err = map(vaddr, shmid)	map vaddr range to a shared memory
err = transfer(shmid, eid, P)	transfer P to another enclave
err = attach(shmid)	associate with a shared memory
err = detach(shmid)	disassociate with a shared memory
err = share(shmid)	share a shared memory with read-only
err = destroy(shmid)	destroy a shared memory

B. Inter-Enclave Communication within the TEE Region

Communication Interface. In our system, the enclave initiating communication is the *enclaver*, and the enclave receiving communication is the *enclavee*. Table II outlines the seven instructions for versatile inter-enclave communication within LayerTEE. In this scheme, the *key* is predefined by the user-defined communication schemes and includes two parts: *shm_key* and *enclave_key*. The *shm_key* is used to connect *enclavee* to shared memory, while the *enclave_key* ensures the *enclavee* is correctly identified when the *enclaver* transfers shared memory permissions. This paper details the design and implementation of this inter-enclave communication mechanism through a client-server communication flow.

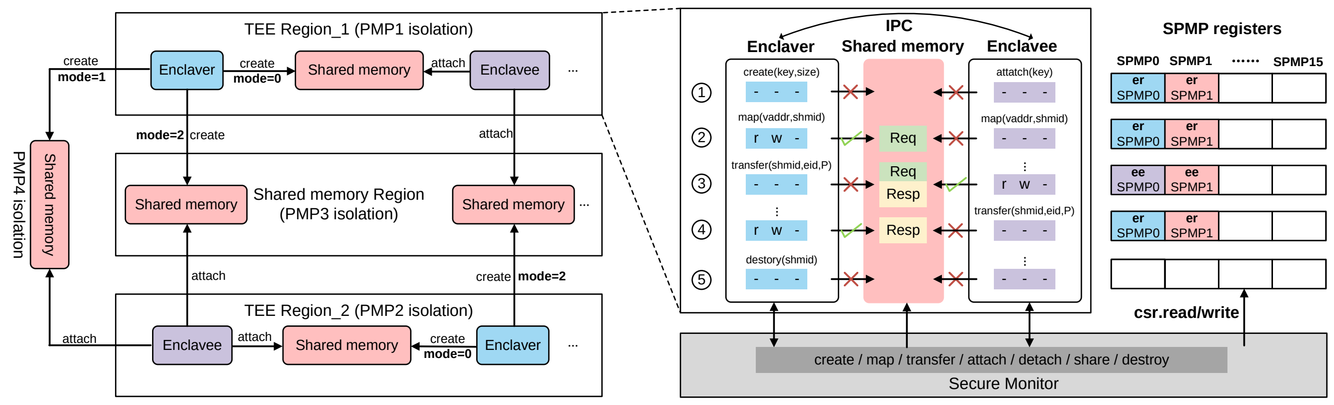
Communication Flow. A predefined *key* is bound to the eid during the creation of each enclave requiring communication. For enclaves communicating via the same shared memory, their *shm_keys* are identical, while *enclave_keys* are unique. Fig. 5(b) shows the communication flow between two enclaves in the TEE Region using a client-server pattern and the changes in SPMP registers at each stage.

Stage ①: The *enclaver* issues the `create` instruction to request the shared memory. The secure monitor allocates the memory, binds its shmid with the *shm_key* and the *enclaver's* eid, and updates the SPMP registers and the *enclaver's* SPMP context. Upon successfully creating shared memory, the *enclavee* obtains the shmid via *shm_key* and associates the memory using the `attach` instruction.

Stage ②: Both enclaves execute the `map` instruction, mapping the shared memory into their virtual address spaces. The *enclaver* has read and write access, but the *enclavee* cannot access the memory because the SPMP registers store the information of the *enclaver*.

Stage ③: After writing data, *enclaver* transfers access permissions of the shared memory to the *enclavee* using the `transfer` instruction. The secure monitor updates the SPMP registers, allowing the *enclavee* to read the request (req) and provide a response (resp).

Stage ④: After sending its resp, the *enclavee* transfers access permissions back to the *enclaver* using the `transfer` instruction.



(a) Inter-enclave communication with multi-layer

(b) Inter-enclave communication within a TEE Region

Fig. 5. LayerTEE communication system. (a) "Close" enclaves within the same TEE Region can use mode0 to create shared memory for communication, while "distant" enclaves in different regions can use mode1 or mode2. (b) The inter-enclave communication process follows a client-server pattern. The terms er.SPMP and ee.SPMP refer to the SPMP register contexts of the enclaver and enclavee.

TABLE III
UPDATED INTER-ENCLAVE COMMUNICATION INSTRUCTIONS IN LAYERTEE. THE USER SELECTS THE MODE AND CK VALUE.

Instructions	Semantics
shmid = create(mode, key, size)	create a shared memory
err = destroy(shmid, ck)	destroy a shared memory and determine whether to check ShMRegion

Stage ⑤: When communication concludes, the *enclavee* uses the `detach` instruction to disassociate from the shared memory, and the *enclaver* destroy the shared memory using the `destroy` instruction. The secure monitor clears the SPMP configurations and memory mappings, releasing resources.

The bilateral and multi-bilateral communication patterns in §II-B all follow this fundamental flow. By using the `share` instruction, the *enclaver* can set the shared memory to read-only, allowing the *enclavees* executing the `attach` instruction to read data, thus enabling the single-producer multiple-consumer communication pattern.

Ownership Transfer. In this scheme, the `transfer` instruction triggers a trap into the M-mode secure monitor, which reconfigures the SPMP registers to facilitate shared memory ownership transfer among enclaves. This approach avoids the overhead of multiple unmapping and remapping operations needed in a remapping mechanism and enhances security against TOCTTOU attacks and targeting page table attacks.

C. Inter-TEE Region Communication

To support multi-layer communication in a multi-cloud environment, we design inter-enclave communication across TEE Regions in the LayerTEE system.

The intra-TEE Region communication scheme described in §IV-B does not meet the requirements for inter-TEE Region communication, as the communicating enclaves are located in different TEE Regions. Creating shared memory within one TEE Region and keeping its PMP permissions open during communication can pose a security risk to other enclaves within the same region. While the traditional copy+encryption method can facilitate inter-TEE Region communication, it incurs significant performance overhead. Therefore, we use the shared memory mechanism to address the needs of inter-TEE Region communication, as shown in Fig. 5(a).

We updated certain communication instructions from Table II, detailed in Table III. When an *enclaver* initiates communication, the appropriate mode is selected based on the scenario to invoke the `create` instruction for shared memory creation.

Mode 0 is designed for communication between enclaves within the same TEE Region. In this mode, shared memory is created within the same TEE Region and isolated using SPMP registers.

Mode 1 is intended for temporary communication between enclaves in different TEE Regions. In this mode, the secure monitor allocates a memory block of the specified *size* in untrusted memory, isolates it using PMP registers, and synchronizes the address information into the SPMP registers context of the *enclaver*. During communication, the PMP permissions for this memory remain open to ensure access checks pass smoothly. When other enclaves or untrusted applications run, the PMP permissions are closed to prevent unauthorized access. Subsequent communication flows, such as the *enclavee* invoking the `attach` instruction to associate the shared memory or the *enclaver* invoking the `transfer` instruction to transfer shared memory permissions to the *enclavee*, follow the same process as intra-TEE Region communication. Once communication is complete, the `destroy` instruction is invoked to release the shared memory and free resources.

Mode 2 is designed for communication among many enclaves across multiple TEE Regions. In this mode, the secure monitor first allocates a large fixed-size memory block in untrusted memory, isolates it using PMP registers, and designates it as a Shared Memory Region (ShMRegion). The size of this region is determined by the LayerTEE system based on the available memory capacity. If sufficient contiguous physical memory is unavailable, inter-enclave communication defaults to mode 1. Next, the secure monitor allocates a physical memory

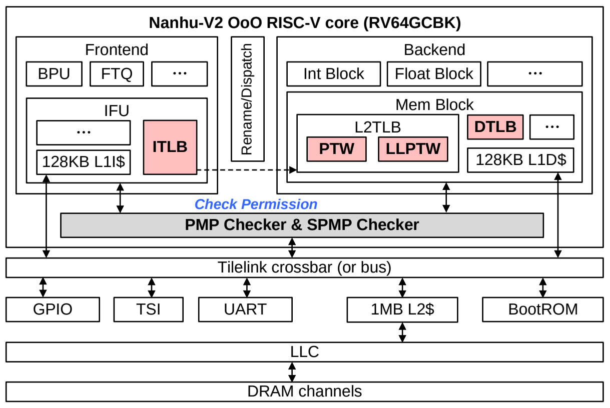


Fig. 6. LayerTEE implementation based on Xiangshan Nanhu Core. The red box highlights the areas most closely associated with the PMP and SPMP checker.

block of the specified *size* within the ShMRegion and configures the address information into the SPMP registers context of the *enclaver*. The subsequent communication process follows the same steps as previously described. During communication, the PMP permissions for the ShMRegion remain open, and the ownership of the shared memory is transferred by invoking the `transfer` instruction to switch the SPMP registers context. When non-communicating enclaves or untrusted applications are running, the PMP permissions for the ShMRegion are closed. Upon completion of communication, the *enclaver* invokes the `destroy` instruction to release the shared memory. When the `destroy` instruction is invoked, the secure monitor decides whether to check for remaining shared memory within the ShMRegion based on the value of ck (ck=0 means no check, ck=1 means check). If none exists, the region's memory is released, and the data is cleared.

In summary, the three communication modes designed in the LayerTEE system effectively support both intra-TEE Region and inter-TEE Region communication across various scenarios. By utilizing the shared memory mechanism, the LayerTEE system ensures security and significantly enhances communication performance.

V. Implementation

Hardware. We extend the SOTA RISC-V high-performance processor, Xiangshan Nanhu-V2 (an 11-level superscalar out-of-order core) [50] by adding 16 SPMP CSR registers and introducing an SPMP Checker module, as shown in the Fig. 6. During memory access, the MMU translates virtual addresses into physical addresses, performing parallel PMP and SPMP checks. Memory access is allowed only if both checks pass; otherwise, it is prohibited. These modifications do not alter the core's pipelines.

Software. We implement LayerTEE on the Penglai Enclave (PMP version, v0.2 release [24]), an advanced RISC-V platform open-source TEE. Penglai Enclave provides Linux drivers, an SDK, and authentication mechanisms for creating, running, relaying, and destroying enclaves. We extend the secure monitor to support scalable and efficient inter-enclave communication and TEE Region memory management without using a guarded

TABLE IV
Simulation configurations.

	Parameter	Value / Description
Xiangshan Core Nanhu V2 Architecture	Processor	OoO RISC-V CPU@2GHz
	Front-end	6-way decoder, 64-entry fetch target queue, 48-entry instruction buffer, 256-entry micro branch target buffer, 2048-entry fetch target buffer, 16K-entry TAGE-SC, RAS, ITTAGE
	Execute	6-way rename/dispatch, 256-entry ROB, 192 int/fp physical registers, ALU, MUL/DIV, JUMP/CSR/I2F, LD, STA, STD, FMAC, FMISC
	LSU	80-entry load queue, 64-entry store queue
	L1 Cache	128KB 8-way I-cache/D-cache
	L2 Cache	1MB 8-way non-inclusive
	LLC	6MB 8-way non-inclusive
	L1 I/D TLB	40-entry ITLB (32-entry normalpage, 8-entry superpage) full-associative, 136-entry DTLB
	L2 TLB	2048 entries
Memory	8GB DDR4 KVR26S19S8/8	
OS	Buildroot, Linux 5.10, OpenSBI 0.9	

page table proposed by Penglai-TVM or other new hardware features.

Methodology. We evaluate LayerTEE on the Xilinx Virtex UltraScale+ VU19P (XCVU19P) FPGA [4], which simulates a Nanhu Core System on Chip (SoC) running at 2GHz. The detailed hardware configurations are provided in Table IV. Our microbenchmark tests compare the communication performance of LayerTEE with several other systems: PL-Copy (based on the original PMP version), PL-Remapping (an enhanced Penglai-PMP version with a remapping communication mechanism), PL-PMPTable (an improved Penglai-PMP version incorporating the PMP-Table mechanism), and PL-PT-Copy (an extension of the Penglai-PMP version with a page table mechanism and copy+encryption communication mechanism). We compared LayerTEE with Penglai-PMP and Host-with-SPMP, a non-TEE environment equipped with SPMP hardware for application-level benchmarks. Further details can be found in §VI.

Note: This paper extends our previous work on Dep-TEE [42] by introducing LayerTEE, an enhanced system for inter-enclave communication. LayerTEE improves the shared memory mechanism by establishing the ShMRegion, designed explicitly for inter-enclave communication across different TEE Regions, thereby facilitating inter-TEE Region communication. Within a single TEE Region, the inter-enclave communication mechanism in LayerTEE remains identical to that of Dep-TEE. As a result, the communication performance across the five pattern regions is unchanged. Consequently, a quantitative comparison between LayerTEE and Dep-TEE is unnecessary, as LayerTEE inherently reflects Dep-TEE's performance within a TEE Region. Importantly, LayerTEE does not introduce new hardware extensions, ensuring the hardware overhead remains consistent with Dep-TEE.

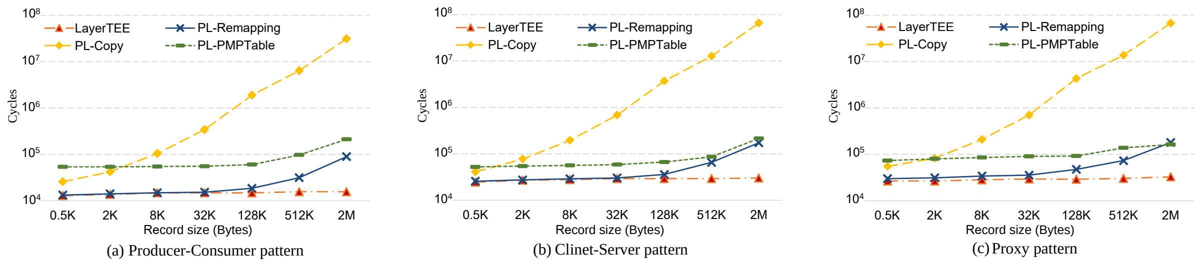


Fig. 7. Performance comparison of three bilateral patterns: producer-consumer, client-server, and proxy. LayerTEE is compared against the baselines of copy, remapping, and shared memory mechanisms used in CVM, all implemented on the Penglai system.

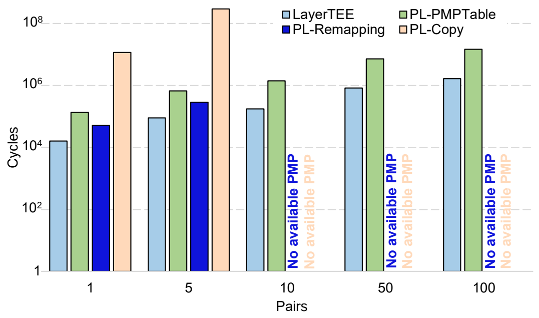


Fig. 8. Performance of multi-pair producer-consumer communication. Here, "pair" refers to two communicating enclaves, e.g., a producer and consumer enclave. "No available PMP" indicates that the number of PMP registers is insufficient to support inter-enclave communication.

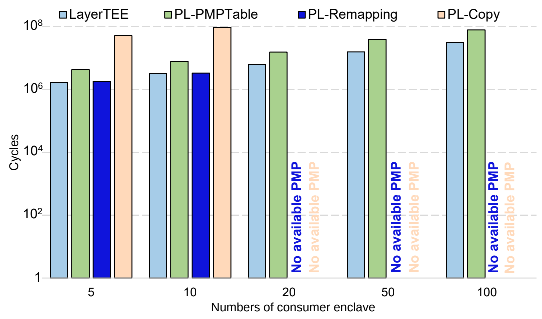


Fig. 9. Performance of single-producer multi-consumer communication.

VI. Evaluation

A. Microbenchmarks

To evaluate the communication performance of LayerTEE, we design a comprehensive set of benchmarks that include five communication patterns utilizing three different communication mechanisms. Our research focuses on data transmission performance, intentionally excluding data processing aspects. We examined the efficiency of data transmission in LayerTEE and PL-Remapping and the data copying performance in PL-Copy without considering encryption or decryption operations. The microbenchmark tests are structured into two main parts. The first part evaluates the inter-enclave communication performance within the same TEE Region. The second part evaluates the inter-enclave communication performance across different TEE Regions. This division allows for a detailed analysis of the communication capabilities in various scenarios.

1) Performance of Intra-TEE Region: We evaluate the performance of LayerTEE and compare it with three major types of TEE communication mechanisms: PL-copy (representing TEEs that use copy+encryption mechanisms, such as SGX [36] and Sanctum [20]), PL-Remapping (representing TEEs that use remapping mechanisms, such as Komodo [26] and Penglai-TVM [25]), and PL-PMPTable (representing confidential virtual machine types of TEEs that use permission table isolation and shared memory communication, such as Intel TDX [31] and ARM CCA [8]).

Bilateral communication patterns results. Using microbenchmark tests, we evaluate the data transfer performance of three bilateral communication patterns, i.e., producer-consumer, client-server, and proxy. The analysis focuses on how data size affects transfer latency, with Fig. 7 illustrating the results for record sizes ranging from 0.5 KB to 2 MB. The results demonstrate that LayerTEE consistently delivers significant performance advantages in all three communication patterns. For instance, in the producer-consumer pattern, LayerTEE achieves a 7× performance improvement over PL-Copy for 8 KB of data, dramatically increasing to 2000× for 2 MB of data. Similarly, in the client-server and proxy patterns, LayerTEE outperforms PL-Remapping and PL-Copy by 10× and several hundred times, respectively, when handling larger data volumes. The performance decline of PL-Copy with increasing data size is primarily due to data copying overhead, which is further exacerbated when encryption and decryption operations are included. For PL-Remapping, performance drops significantly beyond 128 KB of data because frequent unmapping and remapping operations impose considerable overhead. While PL-PMPTable benefits from the shared memory mechanism, its performance remains slightly lower than LayerTEE. This is attributed to the additional overhead from multiple memory accesses required for permission checks in its permission table isolation mechanism.

Multilateral communication patterns results. In the multilateral communication patterns, we evaluate the performance of different systems in the producer-consumer pattern while transferring 1 MB of data. Evaluating performance with a larger data size is crucial to ensure efficient communication within TEEs under substantial load. Fig. 8 shows the performance of each system as the number of enclave pairs increases, ranging from 1 to 100 pairs. The results show that LayerTEE significantly outperforms PL-Copy, with performance improvements of over 3000× when transferring 1 MB of data. As the number of enclave pairs exceeds 10, both PL-Copy and PL-Remapping fail to support additional communication due to limitations in PMP register resources. Similarly, Elasticlave [52], based on the Keystone [33] design, faces scalability issues for the same reason—insufficient PMP registers. In contrast, LayerTEE can efficiently support communication among more than 100 pairs of enclaves, highlighting its strong potential for handling large-scale multitask processing. Although PL-PMPTable supports communication over 100 enclave pairs, its performance lags behind LayerTEE due to the multiple memory accesses required per permission check, resulting in significant overhead.

In the single-producer multi-consumer pattern, we analyze data transfer latency across different systems as the number of consumer enclaves varies. Fig. 9 shows the performance of each system, including data read and write latency, when transferring 1 MB of data with the number of consumer enclaves ranging from 5 to 100. This reflects common usage scenarios and evaluates the scalability of the systems under varying loads. As we can see, LayerTEE exhibits substantially superior communication performance compared to PL-Copy and PL-PMPTable, while also achieving a modest yet consistent advantage over PL-Remapping. This advantage arises from LayerTEE's dynamic SPMP register adjustment mechanism that optimizes large-scale communication data management, while other baseline systems suffer significant overhead due to their inherent limitations: PL-Copy requires redundant data duplication, PL-Remapping demands frequent page table remapping, and PL-PMPTable incurs multiple memory accesses for permission checks. It is noteworthy that PL-Copy and PL-Remapping, like Elasticlave, encounter scalability limitations beyond 20 consumer enclaves due to insufficient PMP registers. In comparison, LayerTEE supports communication with unrestricted consumer enclaves, demonstrating its exceptional capability for cloud computing environments that demand large-scale concurrent task execution.

B. Performance of Inter-TEE Region

Since current TEE architectures do not yet support the concept of TEE Regions and no existing comparison baselines are available, we designed and implemented a baseline system, PL-PT-Copy and PL-PT-Remapping. Based on Penglai-PMP, this system uses the page table mechanism to isolate each enclave within physically separated memory regions managed by PMP registers. Inter-enclave communication is achieved through a copy+encryption and Remapping mechanism. Using bilateral and multilateral communication patterns, our performance evaluation compares LayerTEE with PL-PT-Copy and PL-PT-Remapping. Enclaves requiring communication are created in different TEE Regions to simulate real-world scenarios. We analyze the performance of the three systems under various communication patterns to determine their effectiveness and efficiency.

Results. LayerTEE not only supports inter-TEE Region communication but also demonstrates significant performance advantages in both bilateral (Fig. 10 (a), (b), and (c)) and multilateral (Fig. 10 (d) and (e)) communication patterns. This performance is primarily due to LayerTEE 's use of the shared memory mechanism, which allows data exchange through direct instructions, making its performance less dependent on data size. In contrast, while PL-PT-Copy can also support inter-TEE Region communication, its copy and encryption operations introduce a significant performance overhead as data size increases. This overhead reduces efficiency, particularly in large data transfers and multi-task environments. The PL-PT-Remapping demonstrates comparable performance to LayerTEE in low-data-volume communication scenarios, but exhibits substantial performance degradation under high-data-volume conditions. This limitation is attributed to the inherent overhead of the page table remapping mechanism when processing large-scale data transfers, primarily caused by system-level maintenance operations, including TLB flush operations and page table entry reconstruction. In addition, since the enclave isolation mechanisms of these two systems rely on page tables, their security is still threatened by page table attacks. On the other hand, LayerTEE shows more pronounced advantages in handling large-scale data transfers and concurrent tasks. It provides an advanced platform that facilitates efficient communication within a single cloud provider and supports collaboration across multiple providers. This flexibility allows LayerTEE to adapt to various communication scenarios, significantly enhancing the efficiency and scalability of cross-cloud and inter-TEE Region communication. Consequently, LayerTEE offers an effective solution for complex application environments, making it a robust choice for modern cloud computing needs.

C. Benchmark suites

We use the RV8 [17] benchmark suite to evaluate the performance of compute-intensive workloads in environments with physical memory isolation. We port the benchmark suite to LayerTEE and execute it in two additional environments for comparison: Penglai-PMP, the original TEE environment, and Host-with-SPMP, a rich execution environment (REE) augmented with SPMP check extensions. Additionally, we use Redis [44], a widely used in-memory data store, to evaluate the SPMP mechanism's impact on memory-intensive applications. The Redis benchmark simulates multiple client connections to a Redis server, measuring the average number of requests per second.

Results. As shown in Fig. 11, introducing the SPMP mechanism in LayerTEE results in negligible performance overhead for CPU-intensive applications compared to Penglai-PMP. Similarly, Fig. 12 demonstrates that the SPMP mechanism does not introduce significant overhead for the Nanhu V2 Core in memory-intensive applications. This is because, during the design phase, the CPU simultaneously performs PMP and SPMP checks when the MMU translates virtual addresses to physical addresses, resulting in minimal overhead for the original PMP system.

D. Hardware Costs

The Vivado [5] resource utilization report (Table V) shows the impact of the SPMP mechanism on the FPGA. After implementing the SPMP mechanism on the Xiangshan Nanhu

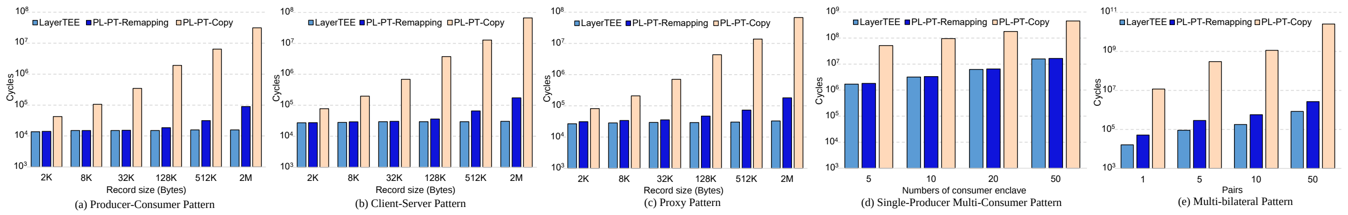


Fig. 10. Performance of Inter-TEE Region communication. Figures (a) to (c) represent bilateral communication patterns, while Figures (d) and (e) depict multilateral communication patterns. PL-PT-copy and PL-PT-Remapping refer to two extensions of the Penglai-PMP version that incorporate a page table mechanism with the copy+encryption and the remapping communication mechanism.

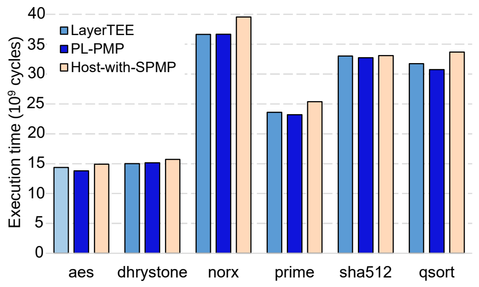


Fig. 11. Performance of RV8 benchmark suite.

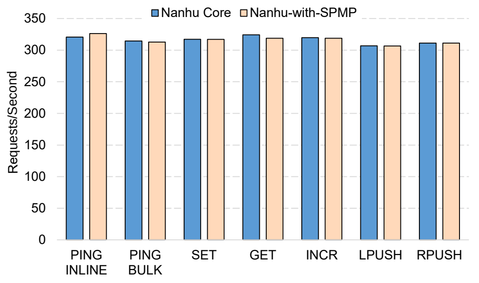


Fig. 12. Performance of Redis benchmark suite.

TABLE V
HARDWARE RESOURCE COSTS OF THE TOP MODULE IN FPGA.

Recourse	NanhuV2 Core	NanhuV2 Core-with-SPMP	Cost
LUT	1259204	1267350	0.20%
LUTRAM	68336	68332	0.00%
FF	447069	450546	0.04%
BRAM	336	336	0.00%
URAM	90	90	0.00%
DSP	3	3	0.00%

V2 Core, we observed that the hardware cost is remarkably low. Specifically, Look-Up Tables (LUTs) are used only 0.2%, Flip-Flops (FFs) are used 0.04%, and other resources in the top module are used negligibly. These results indicate that introducing the SPMP mechanism demands minimal additional hardware resources, demonstrating its efficiency in controlling hardware costs.

E. Security Analysis

LayerTEE's communication system is implemented on Penglai TEE. The Mounted Merkle Tree (MMT) ensures data and code integrity, while cache line locking prevents cache side-channel attacks. Additionally, segment-based registers for physical memory isolation can protect the enclave's privacy from untrusted OS applications.

Mitigating Rowhammer Attacks. Rowhammer attacks exploit hardware flaws to induce bit flips in memory cells, potentially compromising page tables and altering memory mappings. TEEs like Intel TDX [31] and AMD SEV [3] rely on shared memory mechanisms for inter-enclave communication. However, these TEEs map shared memory directly to an enclave's physical address space using page tables, leaving them vulnerable to Rowhammer attacks. In the LayerTEE system, relying solely on page tables to isolate enclaves and shared memory in the TEE Region introduces vulnerabilities to Rowhammer attacks. A malicious enclave could exploit Rowhammer through two attack vectors: (1) manipulating its own page table permissions to evade memory isolation, and (2) corrupting a target enclave's page tables to gain unauthorized access to sensitive data. Our proposed solution employs the SPMP mechanism to isolate enclaves and shared memory within the TEE Region. Every memory access undergoes CPU-level SPMP physical address checks, ensuring robust protection even when page tables are compromised by Rowhammer. Even if a malicious enclave successfully modifies its own page table, any attempt to access another enclave's memory region would still be intercepted by SPMP checks. While attackers might corrupt data within an enclave, they are prevented from extracting sensitive information. Our approach complements existing Rowhammer defenses, such as guard rows [12], [13], [32], counter-based methods [47], and error-correcting codes (ECC) [11], [18], which provide additional layers of protection and are orthogonal to this work.

Defending against TOCTTOU Attacks. TOCTTOU attacks exploit the gap between checking and using a resource, allowing malicious enclaves to manipulate shared memory during this window. TEE communication systems like Data-Enclave [51] reduce communication overhead by mapping shared memory to multiple enclaves simultaneously. However, this approach makes them particularly susceptible to TOCTTOU attacks. In the TEE Region without SPMP-based isolation, communicating enclaves retain concurrent access to the shared memory. When a target enclave completes access verification and initiates operations (e.g., read/write), the malicious enclave can reacquire the same memory region to disrupt these operations. This temporal inconsistency in permission views creates exploitable conditions for TOCTTOU attacks. LayerTEE mitigates this risk by using the SPMP mechanism to enforce shared memory ownership transfer through the `transfer` instruction. This ensures that only one enclave has write access to the shared memory at any time, preventing TOCTTOU attacks and ensuring secure, consistent access.

In summary, the aforementioned works use the shared memory mechanism to implement inter-enclave communication, achieving similar performance but lacking security, making them vulnerable to attacks targeting page tables and TOCTTOU attacks. LayerTEE enhances security by utilizing the SPMP mechanism for permission decoupling, maintaining high performance while improving protection against these attacks.

VII. Conclusion

This paper presents LayerTEE, a novel TEE communication system that addresses the limitations of existing TEE solutions in balancing performance, security, and scalability. Using the RISC-V PMP and SPMP mechanisms, LayerTEE decouples memory protection from address translation, enabling secure shared memory communication while mitigating vulnerabilities such as Rowhammer and TOCTTOU attacks. LayerTEE introduces a multi-layer communication design that distinguishes between intra-TEE Region and inter-TEE Region interactions. Experimental results show LayerTEE achieves significantly reduced communication latency, improved scalability, and robust security guarantees compared to existing solutions. This work highlights the potential of LayerTEE to advance TEE communication systems, offering a practical and efficient solution for emerging security-sensitive applications. Future research can explore further optimizations and extensions to enhance LayerTEE 's applicability across diverse hardware architectures and use cases.

References

[1] A. Ahmad, J. Kim, J. Seo, I. Shin, P. Fonseca, and B. Lee, "CHANCEL: Efficient Multi-client Isolation Under Adversarial Programs." in *NDSS*, 2021.

[2] B. Aichinger, "DDR memory errors caused by Row Hammer," in *2015 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2015, pp. 1–5.

[3] AMD. (2019) AMD Secure Encrypted Virtualization (SEV). [Online]. Available: https://developer.amd.com/sev/

[4] AMD. (2020) Virtex UltraScale+ VU19P. Accessed 2025. [Online]. Available: https://www.amd.com/en/products/adaptive-socs-and-fpgas/fpga/virtex-ultrascale-plus-vu19p.html

[5] AMD, "Vivado design suite," https://www.xilinx.com/products/design-tools/vivado.html, 2024, referenced 2024.

[6] N. Amit, A. Tai, and M. Wei, "Don't shoot down TLB shootdowns!" in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020, pp. 1–14.

[7] Andrew Waterman and Yunsup Lee and Rimas Avizienis and David A. Patterson and Krste Asanović, "The risc-v instruction set manual volume ii: Privileged architecture version 1.9," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-129, 2016. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-129.html

[8] ARM. (2022) Confidential Compute Architecture. [Online]. Available: https://www.arm.com/architecture/security-features/arm-confidential-compute-architecture

[9] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'keeffe, M. L. Stillwell *et al.*, "{SCONE}: Secure linux containers with intel {SGX}," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 689–703.

[10] A. Baumann, M. Peinado, and G. Hunt, "Shielding applications from an untrusted cloud with haven," *ACM Transactions on Computer Systems (TOCS)*, vol. 33, no. 3, pp. 1–26, 2015.

[11] M. V. Beigi, Y. Cao, S. Gurumurthi, C. Recchia, A. Walton, and V. Sridharan, "A systematic study of ddr4 dram faults in the field," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2023, pp. 991–1002.

[12] C. Bock, F. Brasser, D. Gens, C. Liebchen, and A.-R. Sadeghi, "Rip-rh: Preventing rowhammer-based inter-process attacks," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, 2019, pp. 561–572.

[13] F. Brasser, L. Davi, D. Gens, C. Liebchen, and A.-R. Sadeghi, "{CAn't} touch this: Software-only mitigation against rowhammer attacks targeting kernel memory," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 117–130.

[14] F. Brasser, D. Gens, P. Jauernig, A.-R. Sadeghi, and E. Stapf, "SANCTUARY: ARMing TrustZone with User-space Enclaves." in *NDSS*, 2019.

[15] S. Brenner, C. Wulf, D. Goltzsche, N. Weichbrodt, M. Lorenz, C. Fetzer, P. Pietzuch, and R. Kapitza, "Securekeeper: Confidential zookeeper using intel sgx," in *Proceedings of the 17th International Middleware Conference*, 2016, pp. 1–13.

[16] S. Chasins, A. Cheung, N. Crooks, A. Ghodsi, K. Goldberg, J. E. Gonzalez, J. M. Hellerstein, M. I. Jordan, A. D. Joseph, M. W. Mahoney *et al.*, "The sky above the clouds," *arXiv preprint arXiv:2205.07147*, 2022.

[17] M. Clark, "rv8-bench: RISC-V Benchmark Suite," 2023, accessed: 2023-10-01. [Online]. Available: https://github.com/michaeljclark/rv8-bench/

[18] L. Cojocar, K. Razavi, C. Giuffrida, and H. Bos, "Exploiting correcting codes: On the effectiveness of ecc memory against rowhammer attacks," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 55–71.

[19] V. Costan, "Intel SGX explained," *IACR Cryptol, EPrint Arch*, 2016.

[20] V. Costan, I. Lebedev, and S. Devadas, "Sanctum: Minimal hardware extensions for strong software isolation," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 857–874.

[21] A. Dave, C. Leung, R. A. Popa, J. E. Gonzalez, and I. Stoica, "Oblivious coopetitive analytics using hardware enclaves," in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020, pp. 1–17.

[22] D. Dean and A. J. Hu, "Fixing races for fun and profit: how to use access (2)." in *USENIX security symposium*, 2004, pp. 195–206.

[23] D. Du, B. Yang, Y. Xia, and H. Chen, "Accelerating Extra Dimensional Page Walks for Confidential Computing," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, 2023, pp. 654–669.

[24] P. Enclave. (2022) Penglai-Enclave-sPMP. [Online]. Available: https://github.com/Penglai-Enclave/Penglai-Enclave-sPMP

[25] E. Feng, X. Lu, D. Du, B. Yang, X. Jiang, Y. Xia, B. Zang, and H. Chen, "Scalable memory protection in the {PENGLAI} enclave," in *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*, 2021, pp. 275–294.

[26] A. Ferraiuolo, A. Baumann, C. Hawblitzel, and B. Parno, "Komodo: Using verification to disentangle secure-enclave hardware from software," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 287–305.

[27] R.-V. T. Group. (2022) RISC-V S-Mode Physical Memory Protection Unit (SPMP). [Online]. Available: https://github.com/riscv-admin/spmp

[28] P. Heinlein, "FastCGI," *Linux journal*, vol. 1998, no. 55es, pp. 1–es, 1998.

[29] M. E. Hossain, M. F. Kabir, A. Al Noman, N. Akter, and Z. Hossain, "Enhancing data privacy and security in multi cloud environments," *BULLET: Jurnal Multidisiplin Ilmu*, vol. 1, no. 05, pp. 967–975, 2022.

[30] T. Hunt, Z. Zhu, Y. Xu, S. Peter, and E. Witchel, "Ryoan: A distributed sandbox for untrusted computation on secret data," *ACM Transactions on Computer Systems (TOCS)*, vol. 35, no. 4, pp. 1–32, 2018.

[31] Intel. (2020) Intel TDX. [Online]. Available: https://software.intel.com/content/www/us/en/develop/articles/intel-trust-domain-extensions.html

[32] R. K. Konoth, M. Oliverio, A. Tatar, D. Andriesse, H. Bos, C. Giuffrida, and K. Razavi, "{ZebRAM}: Comprehensive and compatible software protection against rowhammer attacks," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 697–710.

[33] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, and D. Song, "Keystone: An open framework for architecting trusted execution environments," in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020, pp. 1–16.

[34] M. Li, Y. Xia, and H. Chen, "Confidential serverless made efficient with plug-in enclaves," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2021, pp. 306–318.

[35] S. M. Marium, L. A. Thebo, M. H. Memon *et al.*, "Time efficient data migration among clouds," *arXiv preprint arXiv:1810.04609*, 2018.

[36] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, “Innovative instructions and software model for isolated execution.” *Hasp@ isca*, vol. 10, no. 1, 2013.
[37] K. J. Merseedi and S. R. Zeebaree, “The cloud architectures for distributed multi-cloud computing: a review of hybrid and federated cloud environment,” *The Indonesian Journal of Computer Science*, vol. 13, no. 2, 2024.
[38] F. Mo, H. Haddadi, K. Katevas, E. Marin, D. Perino, and N. Kourtellis, “PPFL: Privacy-preserving federated learning with trusted execution environments,” in *Proceedings of the 19th annual international conference on mobile systems, applications, and services*, 2021, pp. 94–108.
[39] M. Muhil, U. H. Krishna, R. K. Kumar, and E. M. Anita, “Securing multi-cloud using secret sharing algorithm,” *Procedia Computer Science*, vol. 50, pp. 421–426, 2015.
[40] NGINX Docs. (2021) NGINX Content Caching. [Online]. Available: https://docs.nginx.com/nginx/admin-guide/content-cache/content-caching/
[41] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa, “Oblivious {Multi-Party} machine learning on trusted processors,” in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 619–636.
[42] S. Pan, X. Peng, Z. Man, X. Zhao, D. Zhang, B. Yang, D. Du, H. Lu, Y. Xia, and X. Li, “Dep-TEE: Decoupled Memory Protection for Secure and Scalable Inter-enclave Communication on RISC-V,” in *2025 30th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2025.
[43] J. Park, N. Kang, T. Kim, Y. Kwon, and J. Huh, “Nested enclave: Supporting fine-grained hierarchical isolation with sgx,” in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, pp. 776–789.
[44] Redis. (2024) Redis. Referenced Apr 2024. [Online]. Available: https://redis.io/
[45] D. Robinson and K. Coar, “The common gateway interface (CGI) version 1.1,” Tech. Rep., 2004.
[46] D. Saxena, R. Gupta, and A. K. Singh, “A survey and comparative study on multi-cloud architectures: emerging issues and challenges for cloud federation,” *arXiv preprint arXiv:2108.12831*, 2021.
[47] S. M. Seyedzadeh, A. K. Jones, and R. Melhem, “Counter-based tree structure for row hammering mitigation in dram,” *IEEE Computer Architecture Letters*, vol. 16, no. 1, pp. 18–21, 2016.
[48] C.-C. Tsai, D. E. Porter, and M. Vij, “{Graphene-SGX}: A practical library {OS} for unmodified applications on {SGX},” in *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, 2017, pp. 645–658.
[49] G. Viswanath and P. V. Krishna, “Hybrid encryption framework for securing big data storage in multi-cloud environment,” *Evolutionary intelligence*, vol. 14, no. 2, pp. 691–698, 2021.
[50] Xiangshan. (2024) Xiangshan-2 (nanhu) core. Referenced 2024. [Online]. Available: https://xiangshan-doc.readthedocs.io/zh-cn/latest/integration/overview/
[51] Y. Xu, J. Pangia, C. Ye, Y. Solihin, and X. Shen, “Data Enclave: A Data-Centric Trusted Execution Environment,” in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2024, pp. 218–232.
[52] J. Z. Yu, S. Shinde, T. E. Carlson, and P. Saxena, “Elasticlave: An efficient memory model for enclaves,” in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 4111–4128.

Shangjie Pan received the B.Eng. and M.Eng. degrees in computer science from the Hefei University of Technology, Hefei, China, in 2019 and 2022, respectively. He is currently pursuing a Doctoral degree with the Institute of Computing Technology, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing, China.

His research interests include hardware security, system security, and trusted execution environments.

Yinghao Yang is currently pursuing the Master's degree with the Institute of Computing Technology, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing, China.

His research interests include fully holomorphic encryption acceleration, FPGA accelerator design, and fully homomorphic processor design.

Xuanyao Peng is currently pursuing the Master degree with the Institute of Computing Technology, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing, China.

His research interests include hardware security, system security, and trusted execution environments.

Xiquan Zhao graduated from the University Of Science And Technology Of China in 2008, and received the Ph.D. degree in computer science from University of Chinse Academy of Sciences, China, in 2016.

His research interests include system security and trusted execution environments.

Dong Du received the PhD degree in Shanghai Jiao Tong University, China, in 2022. He is currently an assistant professor at the Institute of Parallel and Distributed Systems, Shanghai Jiao Tong University, China.

His research interests include systems, architecture, system security (TEE), serverless computing and HW/SW co-design.

Hang Lu received the B.S. and M.S. degrees in electronic information engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2008 and 2011, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 2015.

He is currently an Associate Professor and a Master Tutor with the ICT, CAS, University of Chinese Academy of Sciences, Beijing. He is also a Research Scientist with Shanghai Innovation Center for Processor Technologies, Beijing. His research interests include fully homomorphic encryption acceleration, FPGA accelerator design, fully homomorphic processor design, and trusted execution environment.

Yubin Xia (Member, IEEE) received the PhD degree in computer science from Peking University, China, in 2010. He is currently a full professor at the Institute of Parallel and Distributed Systems, Shanghai Jiao Tong University, China. He is a member of ACM.

His research interests include system software, computer architecture, and system security.

Xiaowei Li (Senior Member, IEEE) received the B.Eng. and M.Eng. degrees in computer science from the Hefei University of Technology, Hefei, China, in 1985 and 1988, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 1991.

He was an Associate Professor with the Department of Computer Science and Technology, Peking University, Beijing, from 1991 to 2000. In 2000, he joined ICT, CAS, as a Professor, where he is currently the Deputy Director of the State Key Laboratory of Computer Architecture. He has co-authored over 280 papers in journals and international conferences and holds 60 patents and 30 software copyrights. His current research interests include VLSI testing, design for testability, design verification, dependable computing, and wireless sensor networks.