Dep-TEE: Decoupled Memory Protection for Secure and Scalable Inter-enclave Communication on RISC-V

Shangjie Pan[1,2,3], Xuanyao Peng[1,2], Zeyuan Man[4,5], Xiquan Zhao[3], Dongrong Zhang[3], Bicheng Yang[6], Dong Du[6], Hang Lu[1,2,3,5], Yubin Xia[6], Xiaowei Li[1,2,3]
[1]SKLP, Institute of Computing Technology, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences, [3]Zhongguancun Laboratory
[4]ShanghaiTech University, [5]Beijing Institute of Open Source Chip, [6]Shanghai Jiao Tong University

ABSTRACT

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. This paper introduces a novel TEE system, Dep-TEE, 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, and enhances security by safeguarding against attacks on page tables. We have built a prototype of Dep-TEE based on FPGA, incorporating hardware extensions and software support. The evaluation demonstrates that Dep-TEE significantly surpasses existing TEE solutions, achieving three orders of magnitude lower communication latency and 10x greater scalability while maintaining robust security guarantees.

ACM Reference Format:
Shangjie Pan, Xuanyao Peng, Zeyuan Man, Xiquan Zhao, Dongrong Zhang, Bicheng Yang, Dong Du, Hang Lu, Yubin Xia and Xiaowei Li. 2025. Dep-TEE: Decoupled Memory Protection for Secure and Scalable Inter-enclave Communication on RISC-V. In *30th Asia and South Pacific Design Automation Conference (ASPDAC '25), January 20–23, 2025, Tokyo, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3658617.3697763

1 INTRODUCTION

Trusted Execution Environments (TEEs) serve as secure computing environments, offering a hardware/software co-design protection solution to safeguard sensitive data and applications in cloud environments [3, 12, 13]. Enclaves, as crucial security containers within TEEs, protect critical parts of applications from malicious attacks and software vulnerabilities.

Xiquan Zhao, Hang Lu and Xiaowei Li are corresponding authors. This work was supported in part by the National Natural Science Foundation of China under Grant 62172387; in part by the Youth Innovation Promotion Association of Chinese Academy of Sciences (CAS) under Grant 2021098.

ASPDAC '25, January 20–23, 2025, Tokyo, Japan

ACM ISBN 979-8-4007-0635-6/25/01
https://doi.org/10.1145/3658617.3697763

In the cloud environment, enclaves often collaborate to manage distributed computing tasks or share sensitive data. Consequently, secure and efficient inter-enclave communication is essential for completing various complex operations. For instance, multiple hospitals might share patient data to train a machine-learning model for disease diagnosis, enhancing the model's performance [25]. Consider a scenario where a service provider rents cloud computing machines with Intel SGX [16] to provide dedicated enclave processes for each client [1]. However, the limited secure memory (PRM) in SGX leads to significant overhead due to multiple copying and encryption/decryption operations during inter-enclave communication. While Elasticlave [26], based on the Keystone framework [15], supports inter-enclave communication, its scalability is hindered by hardware limitations, making it unsuitable for cloud environments. Therefore, improving the scalability and performance of inter-enclave communication while ensuring data security is of significance.

We analyzed existing TEE communication mechanisms, categorizing them into copy+encryption, remapping, and shared memory. Theoretical and experimental analyses show that the shared memory mechanism offers the best performance but introduces scalability and security issues. Existing TEE systems fail to meet the demands of scalability, security, and high performance for inter-enclave communication. To address these challenges, we propose a novel TEE communication system called Dep-TEE, which utilizes a hardware-level Supervisor Physical Memory Protection (SPMP) mechanism [19] and shared memory. Our main contributions are summarized as follows:

- We design a novel TEE communication system named Dep-TEE. Dep-TEE utilizes the shared memory mechanism to achieve inter-enclave communication. It exploits the SPMP mechanism to decouple permission protection from enclaves' page tables, achieving high security and scalability while not compromising shared memory's high performance.
- We implement a prototype of Dep-TEE on the Penglai-based platform and integrate it into the high-performant open-source RISC-V processor core, Xiangshan Nanhu Core. After introducing the SPMP mechanism, Dep-TEE shows negligible overhead in CPU-intensive RV8 benchmarks, demonstrating high-performance efficiency. The implementation on the Xilinx VU19P FPGA shows that Dep-TEE requires minimal extra hardware resources, utilizing only 0.2% of LUTs and 0.04% of FFs.
- System evaluation shows that Dep-TEE can securely and efficiently support communication among at least 100 pairs of enclaves. Compared to the Penglai-PMP system, Dep-TEE

Table 1: 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. Dep-TEE uniquely offers scalable, high-performance, and secure inter-enclave communication.

System			Inter-Enclave Communication				
Name	**Arch**	**Isolation**	**Mechanism**	**Scalability**	**Performance**	**Security**	**Versatility**
SGX[16]	Intel	Table-based	Copy+encryption	Unrestricted	↓	✓	✗
TDX[13]	Intel	Table-based	Shared memory	Unrestricted	→	R. & T.	✗
SEV[3]	AMD	Table-based	Shared memory	Unrestricted	→	R. & T.	✗
CCA[4]	ARM	Table-based	Shared memory	Unrestricted	→	R. & T.	✗
Komodo[10]	ARM	Table-based	Remapping	Unrestricted	→	R.	✗
Sanctuary[5]	ARM	Table-based	Shared memory	Unrestricted	↑	R. & T.	✗
Sanctum[7]	RISC-V	Segment-based	Copy+encryption	DRAM Regions	↓	R.	✗
Elasticlave[26]	RISC-V	Segment-based	Shared memory	(PMPs-2)/2	↑	✓	✗
Penglai-TVM[9]	RISC-V	Table-based	Remapping	Unrestricted	→	R.	✗
Penglai-PMP[18]	RISC-V	Segment-based	Copy+encryption	PMPs	↓	✓	✗
Dep-TEE	RISC-V	Decoupled	Shared memory	Unrestricted	↑	✓	✓

shows performance improvements of one to three orders of magnitude across various data transfer sizes.

We organize this paper as follows. First, Section 2 introduces the communication mechanisms in existing TEE systems and their limitations. Section 3 overviews the Dep-TEE framework and details the design. Section 4 details the Dep-TEE implementation. Section 5 presents the system evaluation and secure analysis. Finally, Section 6 concludes this work.

2 BACKGROUND AND MOTIVATION

2.1 Inter-Enclave Communication Mechanisms

Inter-process communication (IPC) is crucial in modern OSes for better modularity and is increasingly important for TEE systems. It facilitates functions decomposition and collaborative work, supports complex application scenarios, and enables multitasking [1, 12, 25]. A secure, efficient, and scalable communication mechanism is crucial in cloud computing, but current inter-enclave communication schemes fall short.

Existing inter-enclave communication mechanisms include copy+encryption, remapping, and shared memory, as summarized in Table 1. TEE systems like Intel SGX [16], RISC-V Sanctum [7], and Penglai-PMP [18] use the copy+encryption mechanism, where data is encrypted, 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. ARM Komodo [10] and RISC-V Penglai-TVM [9] adopt the remapping mechanism, where enclaves unmap and remap physical memory for communication. This mechanism is inefficient due to its coarse granularity (4KB pages) and TLB shootdown issues. Intel TDX [13], AMD SEV [3], and ARM CCA [4] 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 [8]. High-concurrency demands in cloud environments amplify these security risks. Elasticlave [26], based on Keystone [15] and RISC-V PMP [22], improves security by isolating shared memory but struggles with high-concurrency communication due to resource constraints.

2.2 Inter-Enclave Communication Patterns

2.2.1 Bilateral Communication Patterns. We apply the communication mechanisms summarized in §2.1 to three representative bilateral data sharing patterns in real-world scenarios as proposed in [26], shown in Figure 1.

Pattern 1: Producer-Consumer. In this pattern, the producer enclave transfers communication data to the consumer enclave (Figure 1(a)). Common application's sub-steps often use this pattern, such as batch processing in web frameworks [11, 21].

Latency tests (Figure 2) using varying record sizes show that the copy+encryption mechanism has the highest overhead due to extensive copying and encryption/decryption. The remapping mechanism has moderate overhead from re-mapping and un-mapping operations, triggering TLB shootdown issues. The shared memory mechanism has the lowest overhead, with consistent instruction level and read/write operations regardless of data size.

Pattern 2: Client-Server. The client and the server enclaves may simultaneously read from and write to shared data (Figure 1(b)). The copy+encryption mechanism involves four copying and two encryption/decryption processes, whereas the shared memory mechanism requires only two mapping operations.

Pattern 3: Proxy. A proxy enclave processes data from a source enclave and forwards it to a destination enclave (Figure 1(c)) like a caching proxy in web services (e.g., Nginx [17]).

2.2.2 Multilateral Communication Patterns. We propose two multilateral communication patterns: multi-bilateral communication and single-producer multi-consumer.

Pattern 4: Multi-Bilateral Communication. In cloud environments, numerous enclaves must operate concurrently, using bilateral communication patterns for multitasking. TEE systems must support this concurrency while ensuring high performance and security.

Pattern 5: Single-Producer Multi-Consumer. 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 pattern enables multiple clients to query a shared database without revealing personal health details.

2.3 Challenges

Existing TEE systems still fall short of satisfying the inter-enclave communication versatility, i.e., scalability, high performance, and high security. We use the shared memory mechanism to achieve the versatility goal. Specifically, we should overcome three challenges.

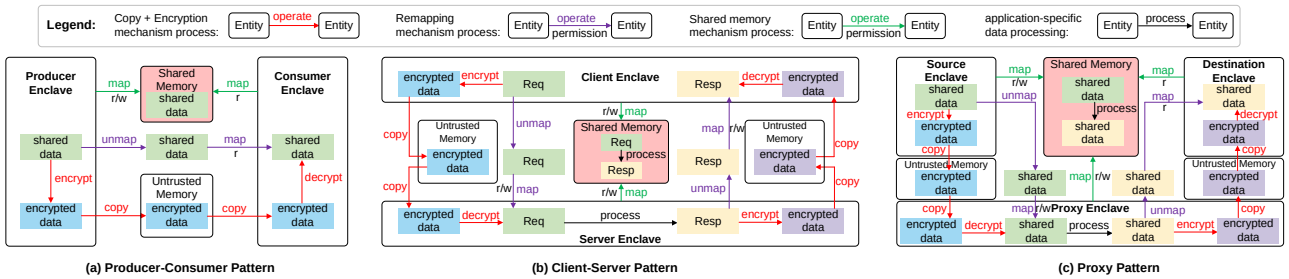


Figure 1: In-depth analysis of implementation processes for three distinct inter-enclave communication mechanisms across various bilateral communication patterns.

Challenge-1: Scalable communication over shared memory. Despite using shared memory for inter-enclave communication, Elasticlave's scalability is limited by its use of the Keystone framework and RISC-V PMP for memory isolation. The number of PMP registers limits Elasticlave's concurrency capability for enclaves.

Challenge-2: Performance over shared memory. Intel TDX, AMD SEV and ARM CCA employ page tables for memory isolation within virtual machines, enabling high-concurrency inter-enclave communication through the shared memory mechanism. However, this mechanism is vulnerable to TOCTTOU attacks, which are hard to mitigate. Additionally, these TEE systems require multiple memory accesses to check the permission table, reducing the performance.

Challenge-3: Security over shared memory. Many TEE systems including RISC-V Penglai-TVM, use page tables for isolation to support concurrent enclaves. While page tables serve for address translation and permission protection, they are vulnerable to targeted attacks like Rowhammer. Existing security mechanisms for page tables often incur substantial performance costs.

Our key insight to resolve the above challenges is **decoupling protection (based on segment) from sharing (based on page tables)**. First, the decoupling can achieve high security without compromising performance as Dep-TEE only relies on register-based segments for security protection, which can defend against attackers targeting page tables (e.g., Rowhammer). Second, we further introduce a layered segment design that can achieve scalable segments compared with existing single-layer designs.

3 DEP-TEE DESIGN

3.1 Architecture

We highlight two key aspects of the Dep-TEE abstraction: (1) memory isolation with decoupled protection and (2) versatile inter-enclave communication. We integrate these abstractions into a secure monitor operating in the highest privilege mode (e.g., machine mode in RISC-V), as shown in Figure 3. The secure monitor manages all enclaves and provides APIs for users to deploy enclaves. It is loaded and verified by the boot ROM during system startup.

We introduce the RISC-V SPMP (Supervisor Physical Memory Protection) mechanism [19] to assist the secure monitor in implementing the Dep-TEE's abstractions. We use the SPMP mechanism to decouple the permission protection from the page tables, defending against specific attacks targeting the page table. Additionally, we leverage SPMP registers to isolate shared memory for versatile inter-enclave communication.

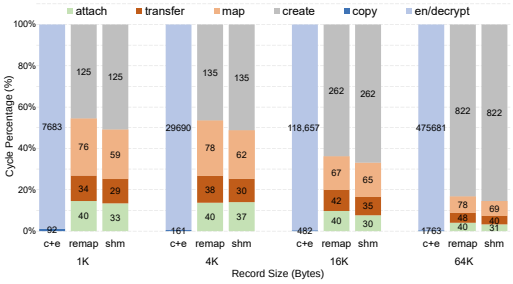


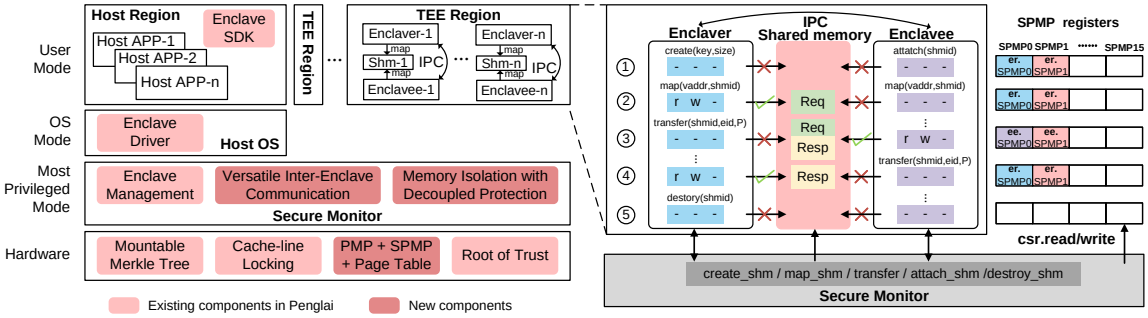
Figure 2: Breakdowns of three communication mechanisms in the producer-consumer pattern. Data labels in the diagram are scaled proportionally to actual data, measured in cycles.

Threat Model. The trusted computing base (TCB) of Dep-TEE includes the hardware CPU and the secure monitor. We assume attackers can access detailed information about OS, hardware components, and network configuration, and can create and execute malicious enclaves. Attackers may exploit OS vulnerabilities or use malicious enclaves to gain unauthorized access and manipulate messages between the CPU and hardware. Moreover, attackers could leverage TOCTTOU attacks and targeting page tables attacks (e.g. Rowhammer) to extract sensitive data. Side-channel and denial-of-service (DoS) attacks are not considered in this work.

3.2 Isolation with Decoupled Protection

Scalability Challenges. RISC-V-based TEEs like Keystone and Penglai-PMP use Physical Memory Protection (PMP) to isolate enclaves. PMP has 16 sets of address and configuration registers in Machine Mode (M-mode) that define physical memory's 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, we use a page table mechanism to isolate among enclaves, as shown in Figure 4. Each enclave's page tables, stored in its dedicated memory, are configured by the secure monitor to map to its memory space. This approach supports the simultaneous creation of multiple enclaves within the same TEE region. In the SV39 page table structure, hardware memory access requires four references: three for the page table pages and one for the data page. Each access involves a PMP check to verify permission. However, this approach is vulnerable to attacks targeting page tables like



(a) Architecture of Dep-TEE. Dep-TEE provides versatile inter-enclave communication and decoupled protection over SOTA TEE systems like Penglai.

(b) Inter-enclave communication process based on the client-server pattern. er.SPMP and ee.SPMP mean the SPMP registers context of enclaver and enclavee, respectively.

Figure 3: Overview of the Dep-TEE architecture with a detailed depiction of the inter-enclave communication process.

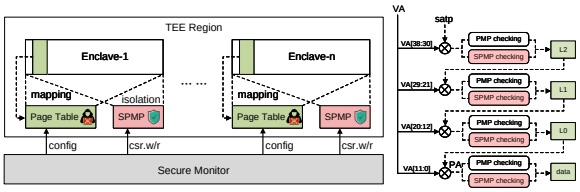


Figure 4: Isolation and access checks after decoupling permission protection from page tables in the TEE region.

Rowhammer, which can leak sensitive data. To counter this, we introduce the RISC-V SPMP mechanism 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 block of physical memory, establishes 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 the enclave creation, the secure monitor writes the configuration information into the SPMP context based on the allocated physical address, size, and access permissions. When the enclave is executed, the secure monitor uses the 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 based on 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.

By using segment-based SPMP registers, we decouple permission protection from page tables, establishing a secure and scalable isolation mechanism for enclaves. As shown in Figure 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.

3.3 Versatile Inter-Enclave Communication

Communication Interface. In our system, the enclave initiating communication is the *enclaver*, and the enclave receiving communication is the *enclavee*. Table 2 outlines the seven instructions for versatile inter-enclave communication within Dep-TEE. 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. Figure 3(b) shows the communication flow between two enclaves in the TEE region using a client-server pattern and SPMP registers' changes at each stage.

Stage 1: The *enclaver* issuing the `create` instruction to request the shared memory. The secure monitor allocates the memory and 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.

Stage 5: When communication concludes, the *enclavee* uses the `detach` instruction to disassociate from the shared memory, and

Table 2: Inter-enclave communication instructions in Dep-TEE. 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

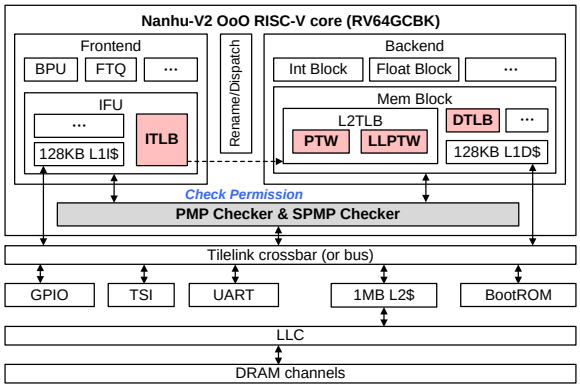


Figure 5: Dep-TEE implementation based on Xiangshan Nanhu Core. The red box highlights the areas most closely associated with the PMP and SPMP checker.

the *enclaver* may 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 §2.2 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, enabling a 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 un-mappings and re-mappings needed in a remapping mechanism and enhances security against TOCTTOU attacks and targeting page table attacks.

4 IMPLEMENTATION

Hardware. We extend the SOTA RISC-V high-performance processor Xiangshan Nanhu-V2 processor (an 11-level superscalar out-of-order core) [24] by adding 16 SPMP CSR registers and introducing an SPMP Checker module, as shown in the Figure 5. During memory access, the MMU translates virtual addresses into physical addresses, performing parallel PMP and SPMP checks. Only if both checks pass memory access is allowed; otherwise, it is prohibited. These modifications do not alter the core's pipelines.

Software. We implement Dep-TEE on the Penglai Enclave (PMP version, v0.2 release [18]), 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 page table proposed by Penglai-TVM or other new hardware features.

Table 3: 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	

Methodology. We evaluate Dep-TEE on the Xilinx Virtex UltraScale+ VU19P (XCVU19P) FPGA which emulates a Nanhu Core SoC operating at 2GHz. Detailed hardware configurations are in Table 3. Microbenchmark tests compare Dep-TEE with Penglai-Copy (based on the original PMP) and Penglai-Remapping (PMP version with a remapping communication mechanism). Application-level benchmarks compare Dep-TEE with Penglai-PMP.

5 EVALUATION

5.1 Microbenchmarks

To evaluate Dep-TEE and the Penglai baseline (PL-Copy and PL-Remapping), we develop a benchmark encompassing five communication patterns based on three sharing mechanisms. Our research focuses on data transmission performance, excluding data processing procedures. Specifically, we analyze data transfer in Dep-TEE and Penglai-Remapping and data copying in Penglai-Copy, omitting encryption or decryption operations.

Performance of Bilateral Communication Patterns. Figure 6 shows the latency results for producer-consumer, client-server, and proxy patterns. Dep-TEE achieves a 7x increase in performance over Penglai-Copy for 8KB of data, and a 2000x increase for 2MB of data. Compared to Penglai-Remapping, Dep-TEE shows significant performance advantages for data sizes over 128KB, achieving a 10x increase for 2MB of data. This is because the overhead of unmapping and mapping operations in the remapping mechanism becomes extremely large as data size increases.

Performance of Multilateral Communication Patterns. Figure 7(a) shows the performance of various systems under multi-bilateral communication patterns using the producer-consumer pattern. Dep-TEE significantly outperforms Penglai-Copy for transferring 1MB of data, achieving up to 3000x acceleration. Penglai-Copy and Penglai-Remapping fail with more than ten pairs due to PMP register limitations. Elasticlave faces the same constraints. In contrast, Dep-TEE supports over 100 pairs, demonstrating its potential for large-scale multitasking in cloud environments.

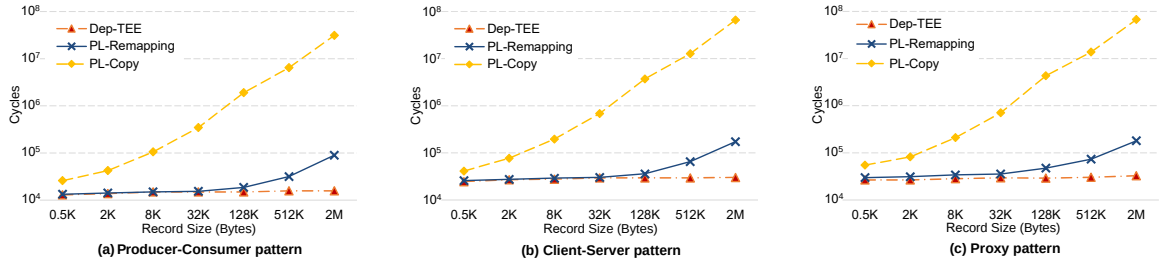
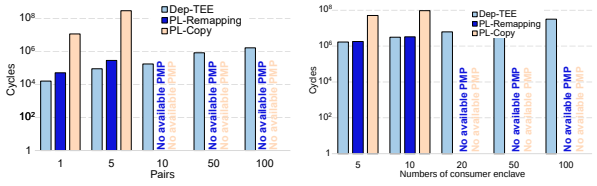
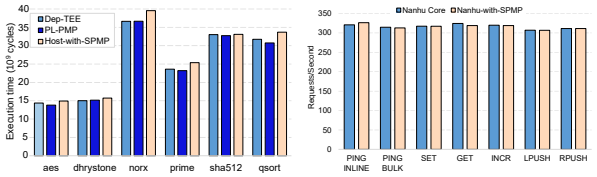


Figure 6: Performance of three bilateral patterns.



(a) multi-pair producer-consumer. (b) single-producer multi-consumer.

Figure 7: Performance of multilateral communication.



(a) RV8 benchmark suite. (b) Redis benchmark suite.

Figure 8: Performance of benchmarks.

Figure 7(b) shows the performance in a single-producer multi-consumer pattern, including data read/write latency. For 1MB data transfers, Dep-TEE outperforms Penglai-Copy and is slightly better than Penglai-Remapping. This is because the remapping mechanism doesn't perform multiple re-mappings and un-mappings in this pattern, avoiding overhead issues such as TLB shootdown. Dep-TEE manages large volumes of data by updating the value of SPMP registers. Penglai-Copy, Penglai-Remapping and Elasticlave do not support more than 20 consumer enclaves due to PMP register limitations. In contrast, Dep-TEE supports at least 100 consumer enclaves, making it suitable for cloud computing applications.

5.2 Benchmark suites

RV8 Benchmark Suite. We used the RV8 benchmark suite to evaluate compute-intensive workloads in environments with physical memory isolation. As shown in Figure 8(a), the introduction of the SPMP mechanism in Dep-TEE results in negligible performance overhead for CPU-intensive applications compared to Penglai-PMP.

Redis Benchmark Suite. Redis, a widely used in-memory data store, evaluates the SPMP mechanism's impact on memory-intensive applications. Redis-benchmark simulates multiple client connections to a Redis server, measuring the average number of requests per second. Figure 8(b) shows that the SPMP mechanism does not introduce significant overhead for the Nanhu V2 Core.

5.3 Hardware Costs

The Vivado [23] resource utilization report (Table 4) shows the impact of the SPMP mechanism on the FPGA. After implementing SPMP on the Xiangshan Nanhu V2 Core, we observed minimal extra hardware cost: 0.2% Look-Up Tables (LUTs), 0.04% Flip-Flops (FFs), and negligible other resources.

Table 4: 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%

5.4 Security Analysis

Dep-TEE'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 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. Dep-TEE mitigates this threat by using the SPMP mechanism to decouple permission protection from page tables in a TEE region. This isolation ensures that even if a Rowhammer attack modifies the page tables, it prevents attackers from bypassing SPMP checks and unauthorized access to enclave data. Attackers might still corrupt data within the enclave but cannot extract sensitive information. Other defences against Rowhammer, like guard rows [14], counter-based methods [20], and ECC [6], complement our approach and are orthogonal to our research.

Defending against TOCTTOU Attacks. TOCTTOU attacks exploit the window between the time of checking a resource and the time of using it, potentially allowing malicious enclaves to manipulate shared memory. Dep-TEE uses the SPMP mechanism to transfer shared memory ownership with `transfer` instruction, ensuring only one enclave can write to the shared memory at a time. This prevents TOCTTOU attacks, maintaining secure and consistent access to shared memory.

6 CONCLUSION

This paper presents Dep-TEE, a new TEE design that achieves scalable, high-performant, and secure inter-enclave communication with decoupled protection. Our evaluation shows that Dep-TEE significantly optimizes inter-enclave communication numbers, enhances communication performance and requires minimal extra hardware resources. In the future, we plan to extend Dep-TEE to support communication between enclaves in different TEE regions. Additionally, we aim to enable secure communication between the TEE and the Rich Execution Environment (REE).

REFERENCES

[1] Adil Ahmad, Juhee Kim, Jaebaek Seo, Insik Shin, Pedro Fonseca, and Byoungyoung Lee. 2021. Chancel: efficient multi-client isolation under adversarial programs. In *NDSS*.

[2] Barbara Aichinger. 2015. Ddr memory errors caused by row hammer. In *2015 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 1–5.

[3] 2019. Amd secure encrypted virtualization. https://developer.amd.com/sev/. Referenced 2024. (2019).

[4] 2024. Arm cca. https://www.arm.com/architecture/security-features/arm-confidential-compute-architecture. Referenced 2024. (2024).

[5] Ferdinand Brasser, David Gens, Patrick Jauernig, Ahmad-Reza Sadeghi, and Emmanuel Stapf. 2019. Sanctuary: arming trustzone with user-space enclaves. In *NDSS*.

[6] Lucian Cojocar, Kaveh Razavi, Cristiano Giuffrida, and Herbert Bos. 2019. Exploiting correcting codes: on the effectiveness of ecc memory against rowhammer attacks. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 55–71.

[7] Victor Costan, Ilia Lebedev, and Srinivas Devadas. 2016. Sanctum: minimal hardware extensions for strong software isolation. In *25th USENIX Security Symposium (USENIX Security 16)*, 857–874.

[8] Drew Dean and Alan J Hu. 2004. Fixing races for fun and profit: how to use access (2). In *USENIX security symposium*, 195–206.

[9] Erhu Feng, Xu Lu, Dong Du, Bicheng Yang, Xueqiang Jiang, Yubin Xia, Binyu Zang, and Haibo Chen. 2021. Scalable memory protection in the {penglai} enclave. In *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*, 275–294.

[10] Andrew Ferraiuolo, Andrew Baumann, Chris Hawblitzel, and Bryan Parno. 2017. Komodo: using verification to disentangle secure-enclave hardware from software. In *Proceedings of the 26th Symposium on Operating Systems Principles*, 287–305.

[11] Paul Heinlein. 1998. Fastcgi. *Linux journal*, 1998, 55es, 1–es.

[12] Tyler Hunt, Zhiting Zhu, Yuanzhong Xu, Simon Peter, and Emmett Witchel. 2018. Ryoan: a distributed sandbox for untrusted computation on secret data. *ACM Transactions on Computer Systems (TOCS)*, 35, 4, 1–32.

[13] 2020. Intel tdx. https://software.intel.com/content/www/us/en/develop/articles/intel-trust-domain-extensions.html. Referenced 2024. (2020).

[14] Radhesh Krishnan Konoth, Marco Oliverio, Andrei Tatar, Dennis Andriesse, Herbert Bos, Cristiano Giuffrida, and Kaveh Razavi. 2018. {Zebram}: comprehensive and compatible software protection against rowhammer attacks. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 697–710.

[15] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. 2020. Keystone: an open framework for architecting trusted execution environments. In *Proceedings of the Fifteenth European Conference on Computer Systems*, 1–16.

[16] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R Savagaonkar. 2013. Innovative instructions and software model for isolated execution. *Hasp@ isca*, 10, 1.

[17] 2021. Nginx docs | nginx content caching. https://docs.nginx.com/nginx/admin-guide/content-cache/content-caching/. (2021).

[18] 2024. Penglai-enclave-pmp. https://github.com/Penglai-Enclave/PenglaiEnclave-sPMP. Referenced 2024. (2024).

[19] 2024. Risc-v spmp. https://github.com/riscv/riscv-spmp/blob/main/rv-spmp-spec.pdf. Referenced 2024. (2024).

[20] Seyed Mohammad Seyedzadeh, Alex K Jones, and Rami Melhem. 2016. Counter-based tree structure for row hammering mitigation in dram. *IEEE Computer Architecture Letters*, 16, 1, 18–21.

[21] 2004. The common gateway interface (cgi) version 1.1. https://www.rfc-editor.org/rfc/rfc3875.html. (2004).

[22] 2016. The risc-v instruction set manual volume ii: privileged architecture version 1.9. technical report ucb/eecs-2016-129. eecs department, university of california, berkeley. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-129.html. (2016).

[23] 2024. Vivado design suite. https://www.xilinx.com/products/design-tools/vivado.html. Referenced 2024. (2024).

[24] 2024. Xiangshan-2 (nanhu) core. https://xiangshan-doc.readthedocs.io/zh-cn/latest/integration/overview/. Referenced 2024. (2024).

[25] Yuanchao Xu, James Pangia, Chencheng Ye, Yan Solihin, and Xipeng Shen. 2024. Data enclave: a data-centric trusted execution environment. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 218–232.

[26] Jason Zhijingcheng Yu, Shweta Shinde, Trevor E Carlson, and Prateek Saxena. 2022. Elasticlave: an efficient memory model for enclaves. In *31st USENIX Security Symposium (USENIX Security 22)*, 4111–4128.