Responses to Reviewers' Comments for Manuscript T-IFS-22131-2025

Coffer: An Efficient and Scalable TEE on RISC-V

Addressed Comments for Publication to

IEEE Transactions on Information Forensics and Security

by

Author(s)

Dear Dr. Jason,

Please find enclosed the revised version of our previous submission entitled "COFFER: An Efficient and Scalable TEE on RISC-V" with manuscript number T-IFS-22131-2025. We would like to thank you and the reviewers for the valuable comments which help improving the quality of our manuscript. In this revision, we have carefully addressed the reviewers' comments. A summary of main modifications and a detailed point-by-point response to the comments from Reviewers 1 to 3 (following the reviewers' order in the decision letter) are given below.

Sincerely,

Author(s)

Note: To enhance the legibility of this response letter, all the editor's and reviewers' comments are typeset in boxes. Rephrased or added sentences are typeset in color. The respective parts in the manuscript are highlighted to indicate changes.

Contents

Response to the Associate Editor

Summary Comment

Based on the enclosed set of reviews, your manuscript requires a MAJOR REVISION (RQ).

Response 0.0: We sincerely appreciate your handling of the review process and the constructive feedback from all three reviewers.

We are grateful for the reviewers' recognition of COFFER's contributions to RISC-V confidential computing, particularly the LPMP framework for overcoming hardware scalability limitations and the EModules design for balancing enclave autonomy with minimal TCB. The reviewers have provided valuable suggestions that have significantly strengthened our work.

According to the reviewers' comments, we have carefully addressed all concerns and suggestions through the following major revisions:

1. We add texts.

These revisions have substantially improved the clarity, completeness, and rigor of our manuscript, addressing all reviewer concerns while maintaining focus on COFFER's core contributions to scalable and efficient TEEs on commodity RISC-V platforms.

Concluding Response to the Editor. We sincerely thank you and the reviewers for the valuable feedback on our manuscript. The review process has been exceptionally constructive, and the detailed comments have allowed us to significantly improve the quality, clarity, and completeness of our work. We believe the revised manuscript now provides a more comprehensive treatment of COFFER's design, security analysis, and practical deployment considerations. We hope that these revisions adequately address all concerns and demonstrate COFFER's significant contribution to the RISC-V confidential computing community.

I. Revision Summary

Based on the reviewers' insightful comments, we improve the paper in four major aspects: xx, xx, xx, and xx.

A. Scalable and Efficient Memory Isolation

Symmetric host/enclave treatment. In practice, the enclave memory pool is reserved at boot time, while actual allocations from this pool are fully dynamic at runtime. When enclaves are created, they request memory from the pool through the Memory Manager, which updates an ownership table to track which memory chunks belong to which enclave. When enclaves terminate, their memory is returned to the pool for reuse. The flexible boundary ensures that if the host OS experiences memory pressure, it can allocate from the edge of the reserved enclave pool; if that boundary region is occupied by enclaves, the Memory Manager migrates enclave memory to other regions within the pool to make boundary memory available to the host OS.

Virtualizing PMP entries. The LPMP Controller maintains the LPMP list in a most recently used order: when a LPMP entry is accessed, it is moved to the head of the list and loaded into a hardware PMP register. Entries at the tail of the list, those accessed least recently, are implicitly evicted from hardware PMP registers when new entries are loaded. This effectively implements an *Least Recently Used* replacement policy for what gets evicted, leveraging temporal locality in enclave memory access patterns. Furthermore, we have evaluated some other replacement policies, and the results **TODO: Figure Ref** shows that the MRU policy gets the best overall performance.

B. TCB

All *Enclave Module*s (EModules) within a single enclave execute in the same S-mode protection domain and are not shared across enclaves; each enclave loads its own private instances according to its permission table.

EModule ecosystem security. Beyond signature verification, Coffer supports practical lifecycle hardening for EModules. The platform owner controls signing keys and can adopt stronger key management. Attestation mechanisms include module measurements, enabling relying parties to verify the exact module set and versions. Minimal module sizes facilitate auditing; permission tables let enclaves include only necessary modules, reducing exposure. These mechanisms help contain supply-chain risks and enable patching and policy-based control as the ecosystem matures.

Security–performance tradeoffs. Coffer trades trap-and-emulate overhead for fine-grained memory isolation. The instruction/data split and optional TLB-enhancement recover most of the lost performance while preserving isolation via exclusive ownership and freshness principles. The autonomous design increases enclave-side TCB (e.g., `EMod_Manager`) to reduce dependence on an untrusted OS, eliminating Iago and controlled-channel attack surfaces.

C. Evaluation

Scalability. We conduct all performance evaluations on the HiFive Unmatched board with four cores (SiFive U74) and 16GB DRAM. For the number scalability test (Figure 6a), we launch up to 2,048 concurrent enclaves distributed across all four cores, each running the `sha512` benchmark from the RV8 suite.

Comparison. A direct quantitative performance comparison for workloads with highly fragmented memory would require substantial modifications to baseline systems and is highly duplicated to our works. Existing software-based TEEs like Keystone cannot support enclaves with such fragmented memory layouts. Our feature-based comparison and worst-case fragmentation experiments in Section VII-B demonstrate that COFFER is the only evaluated system capable of executing workloads under extreme memory fragmentation (achieving $\leq$3% overhead), representing a qualitative breakthrough from "architecturally infeasible" to "feasible and efficient."

D. Protection against Different Attacks

Side-channel attacks. While Section III excludes side-channel attacks from our threat model in general, we note that COFFER's architectural design provides defense against certain side-channel subsets as a derivative advantage. Previous works have demonstrated various side-channel attacks on commodity RISC-V devices. Defending against cache-based side-channel attacks requires hardware modification and remains orthogonal to our study. However, COFFER provides significant mitigation against TLB-based and page-table-based side-channel attacks as a natural consequence of its design: each enclave manages page tables independently, and the SM enforces the principle of exclusive ownership of TLB through full TLB flushes on domain switches.

Response to Reviewer 1

Summary Comment

Thanks for submitting the work to TIFS. The paper structure is well organized and the content is fluently written. Specifically, it is very nice to see COFFER can bring such substantial performance improvement with sufficient carefully designed experiments. Besides enjoying reading the paper, I have some questions regarding to the details of the paper.

Response 1.0: Thank you for your positive feedback and insightful questions.

We have carefully addressed all questions point by point as follows.

Comment 1.1: Memory Pool Management

First, in fig 3, you say the enclaves allocate memory from a reserved memory pool. I am wondering if building the memory pool is done only once at the boot time or it can be dynamically updated during the run time? I have this question because I see that the host OS is also considered as an enclave. Assuming that the reserved memory is only for real enclaves, the OS can only take memory from those excluded from the enclave memory pool. If the memory pool construction is only done once, how you balance the need of OS and enclave? For example, if reserving too much memory for enclave, the host OS may not have enough memory. Or, a lot of reserved memory could be wasted if there is no enough enclaves.

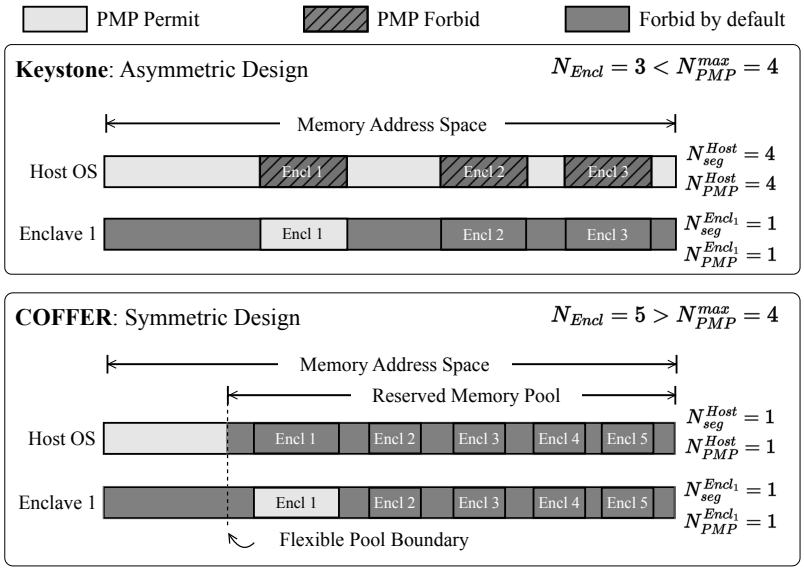

Figure 3: Comparison of asymmetric and symmetric approaches for enclave isolation.

Response 1.1:

Thank you for this excellent question about memory pool management. We clarify the design as follows:

At boot time, the enclave memory pool is defined through kernel parameters (e.g., `movablecore=` in Linux), reserving a physical region for potential enclave use. Allocation from this pool is fully dynamic, with enclaves requesting only the memory they require via a certain interface `__ecall_ebi_mem_alloc` and returning it to the pool when destroyed. The Security Monitor's Memory Manager tracks allocations through a memory ownership table and supports a flexible boundary between enclave and OS memory. This allows the OS to tap into unused pool memory when needed, and if the boundary region is occupied by enclaves, the manager can migrate enclave data to other pool areas. This design ensures reserved memory is never wasted and enables dynamic balancing of OS and enclave demands.

In response, we have expanded §A. to the paper's Section IV-B to provide a more comprehensive overview.

Comment 1.2: LPMP Replacement Policy (Exp WIP)

Second, I am a bit confused about the replacement policy used by LPMP. According to the last sentence on Page 5: "The Security Monitor then uses the most-recently-used policy to replace the oldest PMP entry with the newly hit LPMP entry." What do you mean by oldest? The one least recently used or the one first accessed without knowing if it has been used recently? I currently did not see why they are related to MRU policy. Another question is that is it possible to use any other policy? Will they largely affect the performance? Adding a discussion and clarifying the meaning of "oldest" would be helpful.

Response 1.2:

Thank you for catching this confusing terminology. We apologize for the unclear phrasing.

COFFER uses an MRU list management policy for LPMP entries. When an LPMP entry is accessed (hit), it is moved to the head of the LPMP list. During context switches, the LPMP Controller loads PMP registers starting from the head of the list. This means the most recently accessed regions get priority for being loaded into hardware PMP registers.

By "oldest," we meant the PMP entry corresponding to the LPMP entry at the tail of the PMP registers—that is, the entry that was accessed *least recently*. When all PMP entries in the PMP registers are occupied and a new LPMP entry is accessed, the entry at the tail of the PMP registers (least recently used) is effectively evicted from the hardware PMP registers to make room for the newly accessed entry.

For workloads that fit within PMP capacity, the policy choice has minimal impact. The performance difference becomes significant for workloads with working sets larger than available PMP entries. Theoretically, MRU has the best balance of the performance and locality. To be more clear, we experimentally evaluated several policies during development. FIFO is simpler to implement but showed ∼x% more LPMP faults in memory-intensive workloads. Random replacement is even simpler but increased LPMP faults by ∼x%. We have added this experiments in Section III-B to clarify following §A.

Comment 1.3: EModule Trust Model and Security

Third, it seems that Emodules that are signed are considered as trusted? Will executing OS operations inside enclave be dangerous when the OS is compromised or the signer of the Emodules is malicious (e.g., supply-chain attack)?

Response 1.3:

Thank you for this important security consideration. Here are the rationales behind our trust model and the security mechanisms that protect Emodules.

Yes, signed Emodules are trusted components of the TCB. This design choice follows the standard practice in TEE systems where certain platform-specific components must be trusted to provide meaningful functionality. Intel SGX trusts Intel-signed quoting enclaves and architectural enclaves, AMD SEV-SNP trusts AMD-signed firmware components, ARM TrustZone trusts vendor-signed trusted applications, and Keystone trusts the Security Monitor and runtime. For COFFER, Emodules enable rich OS functionality within enclaves while maintaining isolation from the untrusted host OS. The platform owner controls the signing key and determines which Emodules are authorized, establishing a clear chain of trust.

Besides, The signing mechanism provides strong security guarantees. As described in Section IV-A, when the `Emod_Manager` requests an Emodule from the host OS, it cryptographically verifies the digital signature contained in the Emodule image before loading. Only Emodules signed with the platform's private key can pass verification and execute. Even with a completely compromised OS attempting to provide malicious Emodules, the signature verification will fail, preventing unauthorized code from executing within enclaves. Furthermore, Emodules remain isolated in S-mode within enclave memory protected by PMP, preventing the OS from directly calling, manipulating, or injecting code into loaded Emodules.

Nevertheless, if the signing key infrastructure is compromised and a malicious Emodule is signed, that Emodule could compromise enclaves that load it. However, this scenario represents a supply-chain attack on the TCB itself, which is outside our threat model. Such attacks would undermine the foundation of all TEE security guarantees, as no TEE system can provide protection if its root of trust is compromised. Platform owners can mitigate this risk by maintaining strict control over the signing key infrastructure and conducting thorough audits of Emodule code before signing.

In response, we have added the clarification §B. to the paper's Section V.

Comment 1.4: TLB-based Side Channel Attacks

Finally, why COFFER is resilient to TLB-based side channel attacks? Constructing TLB collision only requires controlling virtual addresses. Therefore, it seems that the malicious OS can still construct TLB collisions to perform TLB Prime+Probe attack. Can you clarify this? Adding a short discussion in the paper would be helpful.

Response 1.4:

Thank you for this important observation. The TLB Prime+Probe attack is a variation

of the Prime+Probe side-channel method that uses the TLB instead of CPU caches to learn about another process's memory accesses. It leverages shared microarchitectural state to leak information across security boundaries. However, this attack requires shared TLB entries between the attacker and victim, which COFFER's design aims to prevent.

As described in §IV-B, COFFER enforces security principles for TLB management. First, regarding exclusive ownership of TLB, at any moment all TLB entries belong to a single entity (either the host OS or a specific enclave), and upon every context switch into or out of an enclave, the LPMP Controller performs a complete TLB flush using `sfence.vma` to prevent the OS from directly observing enclave TLB entries or vice versa. Second, COFFER provides a freshness guarantee where the entire TLB is flushed when an enclave's memory mappings change (e.g., during memory allocation) to prevent stale entries from being exploited. Third, each enclave manages its own page tables independently (Section IV-A), with the OS and enclaves operating in different virtual address spaces, reducing opportunities for controlled TLB collisions.

Just like reviewer 2's comment 4, we have clarified this as a derivative advantage §D. in paper's Section VI-A.

Concluding Response. Thank you for your valuable comments and detailed questions on our manuscript. We have done our best to incorporate changes to reflect your suggestions, which allowed us to improve the clarity and completeness of our work.

Response to Reviewer 2

Reviewer 2

Summary Comment

This paper presents COFFER, a scalable and efficient software-based Trusted Execution Environment (TEE) for standard RISC-V platforms. The authors ingeniously address the core scalability bottleneck in RISC-V TEEs—the limited number of enclaves and support for fragmented memory—by introducing the innovative LPMP framework, which virtualizes the constrained PMP hardware resources. Furthermore, the proposed EModules design offers a promising solution for striking a balance between enclave autonomy and a minimal Trusted Computing Base (TCB). The work is well-motivated, with a precise problem definition, elegant design, solid engineering implementation, and a comprehensive evaluation. The performance data, particularly the support for over 2,000 concurrent enclaves and the low overhead under worst-case memory fragmentation, is highly impressive. COFFER presents significant practical value and an immediate contribution to the RISC-V confidential computing community. I support the acceptance of this paper after it incorporates the following minor revisions.

Sincerely thanks for the thoughtful and constructive feedback. We are grateful for the specific suggestions to further strengthen our work and address them point-by-point below.

Comment 2.1: Comparison with Related Work

Regarding Comparison with Related Work (Future Work Outlook)
The paper effectively employs a feature-based comparison (Table VI) to position COFFER's contributions clearly against prior art, which is very useful. A natural next step to further solidify the performance claims—specifically, to vividly illustrate the scalability breakthrough when the number of enclaves exceeds the limits of standard PMP—would be a direct, quantitative performance comparison. For instance, plotting the aggregate throughput or latency against increasing enclave counts for both COFFER and a baseline like Keystone on the same hardware would powerfully demonstrate the transition from "architecturally infeasible" to "feasible and efficient."
I understand that conducting such a comprehensive performance benchmarking against other TEEs can be non-trivial, potentially involving significant porting and setup efforts that lie beyond the scope of a revision. Therefore, I suggest the authors briefly acknowledge this perspective in their Conclusion or Future Work section. Discussing how such a direct performance/scalability comparison would complement their feature-based analysis would valuably guide future research in the community.

Response 2.1:

Thank you for this insightful suggestion regarding quantitative performance comparison. We appreciate the opportunity to clarify the nature of COFFER's scalability breakthrough and our evaluation methodology.

We believe there may be a slight misunderstanding regarding what aspect of performance would exceed PMP limits. The key insight is that COFFER's LPMP design decouples the number of concurrent enclaves from PMP hardware constraints. Each enclave maintains its own private LPMP list, which is loaded into the standard PMP configuration during context switches. Therefore, the number of concurrent enclaves does not directly impact individual enclave performance—whether running 10 or 2,000 enclaves, each enclave experiences comparable performance when scheduled. As demonstrated in Figure 6a of Section VII-A, the overhead remains stable (within 5%) even when scaling from a single enclave to 2,048 concurrent enclaves.

The performance challenge that *does* relate to exceeding PMP capacity is **memory fragmentation within a single enclave**. When an enclave's physical memory becomes highly fragmented, the number of LPMP regions required to describe its memory layout can exceed what the limited PMP registers (typically 16 entries) can accommodate. This is where LPMP's virtualization capability becomes critical.

To quantitatively demonstrate COFFER's performance under this challenging scenario, we conducted worst-case memory fragmentation experiments in Section VII-B. Specifically, we forced the allocator to allocate physically non-adjacent pages for each enclave, ensuring maximum fragmentation where nearly every page requires a separate LPMP region. Under this worst-case condition:

- For memory-intensive workloads (`stress-ng` vm workers), enclaves without any LPMP optimization could not complete execution in reasonable time (exceeding 1 hour vs. baseline of <10 seconds).
- With instruction/data split optimization alone, performance improved significantly, demonstrating the effectiveness of separating instruction and data memory regions.
- For compute-intensive workloads (RV8 benchmarks with large memory consumption such as `qsort`, `aes`, and `norx`), TLB-enhancement optimization provided greater benefits by leveraging TLB caching of PMP results.
- With both optimizations enabled, COFFER achieved $\leq$3% overhead even under worst-case memory fragmentation, demonstrating near-native performance.

We agree that a direct performance comparison showing aggregate throughput across increasing enclave counts would be valuable. However, such comparison faces a fundamental challenge: existing systems like Keystone are architecturally limited by PMP hardware constraints (typically 8-16 enclaves maximum). To conduct a fair comparison beyond this threshold would require either: (1) substantial architectural modifications to Keystone to incorporate LPMP-like virtualization (essentially reimplementing our core contribution), or (2) comparing against a hypothetical extended version that doesn't currently exist. As you astutely notes, this would involve "significant porting and setup efforts that lie beyond the scope of a revision."

In response, we have added a discussion in the §C. of paper's Section VII explaining that direct performance comparison at scale requires substantial architectural modifications to baseline systems, and how our feature-based comparison demonstrates the qualitative breakthrough enabled by LPMP.

Comment 2.2: Multi-core Configuration Clarification

Regarding Clarification of Multi-core Configuration (Suggestion)
The design claims support for multi-core concurrent execution. To provide clarity, I suggest that the authors briefly state the multi-core configuration used in the relevant experiments (e.g., specify how many processor cores were utilized for the concurrent enclave test in Figure 6a). This simple clarification will help readers better understand the system's behavior on real hardware.

Response 2.2:

Thanks for this helpful suggestion. We have clarified the multi-core configuration used in our experiments.

All performance evaluations in Section VII were conducted on the HiFive Unmatched board, which features four cores (SiFive U74 cores) and 16GB of DRAM. As stated in Section IV-A, Coffer enclaves support multi-threading and can concurrently execute on all system cores. For the concurrent enclave test in Figure 6a (number scalability), we launched up to 2,048 concurrent enclaves across all four cores of the HiFive Unmatched board.

In response, we have added the clarification §C. to paper's Section VII-A.

Comment 2.3: EModules Internal Security Model

Regarding Clarification of the EModules Internal Security Model (Suggestion)
The EModules design is a significant contribution. To precisely define the enclave's TCB and prevent reader confusion, I recommend that the authors explicitly state the isolation relationship between different EModules (e.g., whether they reside in the same protection domain) in the Security Analysis (§VI) or Design (§IV-C) sections. Clarifying this key design choice will greatly enhance the paper's rigor and clarity.

Response 2.3:

Thank you for this important suggestion to clarify the EModules internal security model. This clarification will indeed help readers better understand the enclave TCB composition.

To explicitly address the isolation relationship between different EModules: all EModules within a single enclave reside in the same protection domain. Specifically, all EModules execute in Supervisor Mode (S-Mode) and share the same enclave memory space, which is isolated from the host OS and other enclaves. The EApp runs in User Mode (U-Mode) within the same enclave, while the Security Monitor runs in Machine Mode (M-Mode).

Importantly, while EModules within an enclave share the same S-Mode protection domain, EModules are not shared between different enclaves. As stated in Section VI-A: "Coffer enclaves do not have shared resources, including EModules, physical memory, TLB entries, and page tables." Each enclave has its own private instances of EModules loaded according to its permission table, ensuring isolation between enclaves.

The rationale for this design is that EModules provide trusted OS services to the EApp within the enclave. Sharing the same protection domain enables efficient function calls between EModules and reduces context switch overhead, while the permission-based

dynamic loading mechanism allows users to customize the TCB by including only necessary EModules.

Just like reviewer 1's comment 3, we have added the clarification §B. to Section VI.

Comment 2.4: Side-Channel Discussion Refinement

Regarding Refinement of the Side-Channel Discussion (Suggestion)
The threat model appropriately excludes microarchitectural side-channel attacks requiring hardware modifications, and the paper insightfully notes that its design can mitigate TLB and page-table-based channels. To make the logic more rigorous, I suggest the authors briefly clarify the scope of side-channels discussed, explicitly stating that the channels excluded in §III and those discussed in §VI.A represent different subsets, with the latter being a "derivative advantage" of the architecture.

Response 2.4:

Thank you for this insightful observation. We agree that clarifying this distinction will strengthen the logic of our security discussion. We have refined the side-channel discussion §D. in paper's Section VI-A to clarify the scope.

Concluding Response. Thank you for your valuable comments and detailed questions on our manuscript. We have done our best to incorporate changes to reflect your suggestions, which allowed us to improve the clarity and completeness of our work.

Response to Reviewer 3

Reviewer 3

Summary Comment

This paper presents COFFER, a scalable and efficient Trusted Execution Environment (TEE) for commodity RISC-V platforms. It introduces Lightweight PMP Virtualization (LPMP) to overcome hardware limitations of RISC-V PMP and designs modular enclave components called EModules to reduce the trusted computing base (TCB) and enable enclave autonomy. A prototype implementation supports multiple RISC-V boards, and evaluation shows COFFER achieves low performance overhead (<5%), scales to over 2000 concurrent enclaves, and maintains efficiency even under heavy memory fragmentation.
* Strengths
1.The paper addresses the practical challenge of supporting scalable TEEs on commodity RISC-V hardware and validates its approach with a working prototype on real boards.
2.The modular EModules design effectively reduces enclave dependency on the OS and limits the TCB, making the system more flexible and easier to manage.
3.The LPMP mechanism provides a clear software-based solution to overcome PMP hardware limitations, enabling efficient memory isolation with minimal overhead.

We sincerely thank the reviewer for the comprehensive evaluation and recognition of COFFER's contributions to scalable TEEs on commodity RISC-V platforms. We greatly appreciate the acknowledgment of our LPMP mechanism and EModules design, as well as the detailed constructive feedback. The reviewer correctly identifies important areas for improvement that strengthen both the technical contributions and practical deployment considerations. We address each concern systematically below.

Comment 3.1: I/O Security Model

Incomplete I/O Security Model. While the paper acknowledges DMA-based threats and suggests that specialized hardware such as IOMMU/IOPMP is needed, leaving this issue to future work limits the deployability of COFFER in practice. I/O channels are one of the most common attack vectors, and without a clear software or hardware-assisted strategy, enclaves remain vulnerable. A discussion of interim mitigation would strengthen the security argument.

Response 3.1:

Thank you for this important concern about I/O security. The reviewer correctly identifies that I/O channels represent a significant attack surface, and we appreciate the opportunity to provide a more comprehensive treatment of this critical aspect.

Current I/O Security Mechanisms: COFFER already implements several I/O security mechanisms that provide practical protection in current deployments. For memory-mapped I/O (MMIO) peripheral devices, COFFER adds special LPMP entries to protect I/O operations, ensuring that only authorized enclaves or the host OS can access specific

peripheral devices. Additionally, COFFER implements a secure bulk I/O mechanism through memory ownership transfer, where the sender allocates a memory region, writes the data, and then requests the Security Monitor to transfer ownership to the receiver. Throughout this process, the principle of exclusive ownership is enforced—preventing concurrent access and ensuring data integrity.

Secure Message Channels: COFFER provides secure message channels for communication between enclaves and the host OS through the `SBI_EXT_EBI_LISTEN_MESSAGE` and `SBI_EXT_EBI_SEND_MESSAGE` interfaces. These channels allow controlled data exchange while maintaining enclave isolation. The Security Monitor mediates all message transfers, ensuring that only authorized communication occurs and that message buffers respect enclave memory boundaries.

DMA Attack Mitigation Strategies: While complete DMA protection requires specialized hardware (IOMMU/IOPMP), COFFER provides several interim mitigation strategies. COFFER can be deployed on platforms where DMA-capable devices are limited or can be controlled through administrative policies, reducing the DMA attack surface. COFFER's flexible boundary design ensures that enclave memory pools are allocated from specific physical memory regions that can be administratively configured to avoid certain DMA-capable peripherals. The autonomous enclave design further reduces attack surface by minimizing the need for extensive I/O operations, as EModules handle most system functionality within the TEE boundary.

Hardware Integration Roadmap: COFFER is designed with forward compatibility for emerging RISC-V security extensions. Once IOPMP becomes available, COFFER can integrate it by extending the LPMP framework to manage IOPMP entries alongside PMP entries, providing unified memory and I/O protection. Similarly, COFFER's Security Monitor can be extended to configure IOMMU page tables for enclave-specific I/O memory mappings, ensuring that DMA operations respect enclave boundaries. COFFER's modular design allows these hardware features to be added incrementally without requiring fundamental architectural changes.

Practical Deployment Considerations: For current deployments, COFFER addresses practical I/O security through multiple layers. COFFER's threat model explicitly acknowledges that DMA protection requires additional hardware, allowing users to make informed deployment decisions based on their specific threat landscape. The combination of memory isolation, secure message channels, and memory ownership transfer provides a foundation for secure I/O even without complete DMA protection. Applications can implement application-level encryption for sensitive data transmitted through these channels, providing additional protection layers.

We have expanded the I/O security discussion in Section VI (Security Analysis) to provide more comprehensive guidance:

> **I/O security.** For MMIO peripheral devices, COFFER can add special LPMP entries for the host/enclaves to protect the I/O operations. However, there are also peripheral devices with DMA. Preventing attacks from such peripheral devices requires additional specialized hardware such as IOMMU or IOPMP. IOPMP is still under active development stage, and several RISC-V vendors have announced

plans to implement IOPMP in future SoCs. IOMMU has already been ratified in non-ISA specifications, but currently only a few commodity RISC-V SoCs support it. While complete DMA protection requires specialized hardware (IOMMU/IOPMP), COFFER provides several interim mitigation strategies. COFFER can be deployed on platforms where DMA-capable devices are limited or can be controlled through administrative policies, reducing the DMA attack surface. COFFER's flexible boundary design ensures that enclave memory pools are allocated from specific physical memory regions that can be administratively configured to avoid certain DMA-capable peripherals. It is non-trivial to retrofit IOMMU/IOPMP into COFFER without significant architectural changes. COFFER is designed with forward compatibility for emerging RISC-V security extensions.

This comprehensive approach provides practical I/O security for current RISC-V platforms while positioning COFFER for enhanced protection as specialized hardware becomes available.

Comment 3.2: LPMP Scalability and Sharing

Scalability and Sharing Limitations. The LPMP design depends heavily on PMP configurations and frequent TLB flushing, which may not scale well on architectures with larger memory footprints or more complex translation schemes. The paper does not analyze the performance cost under such scenarios. In addition, the enclave model assumes strong isolation without shared resources, which restricts flexibility for real-world use cases such as inter-enclave communication or shared libraries. Exploring controlled sharing policies could improve practicality.

Response 3.2:

Thank you for this insightful question about LPMP scalability limitations. The reviewer correctly identifies important considerations regarding the scalability of our trap-and-emulate approach, and we appreciate the opportunity to provide a more thorough analysis of LPMP's theoretical and practical scalability bounds.

Theoretical Scalability Analysis: LPMP's trap-and-emulate approach has well-defined theoretical scalability limits. The primary bottleneck occurs when the working set of memory regions exceeds the available PMP entries ($N_{seg}^{active} > N_{PMP}^{max}$), triggering frequent LPMP traps. In the worst case, each memory access to a non-cached LPMP entry incurs a trap overhead of approximately 1000-2000 cycles on RISC-V platforms. However, COFFER's instruction/data split optimization significantly reduces this overhead by reserving dedicated PMP entries for instruction memory, which typically exhibits high locality. Our TLB-enhancement optimization further improves scalability by effectively extending the number of active PMP entries through TLB caching, particularly beneficial for workloads accessing larger memory footprints with reasonable spatial locality.

Memory Footprint Scalability: COFFER has been evaluated with enclaves up to 2GB memory size with less than 7% performance overhead. For larger memory footprints, LPMP's performance depends on memory access patterns rather than absolute memory size. Sequential access patterns benefit significantly from TLB-enhancement, where each

TLB entry can cache PMP results for 2MB regions (matching RISC-V SV39 mega-page size). Random access patterns rely more heavily on instruction/data split optimization. The key insight is that LPMP scales with the active working set of memory regions, not total enclave memory size. Large enclaves with localized memory access maintain near-native performance, while those with scattered access patterns across many regions experience higher overhead.

TLB Flushing Impact and Mitigation: The reviewer's concern about frequent TLB flushing is valid but mitigated by COFFER's selective TLB management. COFFER implements two TLB flushing strategies: (1) full TLB flush during execution environment context switches to ensure exclusive TLB ownership, and (2) selective TLB entry invalidation during LPMP traps when TLB-enhancement is enabled. The selective approach only invalidates the specific TLB entry causing the trap, preserving other cached PMP results. This dramatically reduces TLB flush overhead compared to naive approaches that would flush the entire TLB on every LPMP configuration change. On multi-core systems, TLB flush costs are further amortized since each core maintains its own TLB state.

Scalability with Different RISC-V Configurations: COFFER's scalability adapts to different RISC-V memory management configurations. On platforms with more PMP entries (16/32 instead of the typical 8), LPMP naturally scales better due to reduced trap frequency. For platforms without TLB caching of PMP results, COFFER gracefully degrades to rely primarily on instruction/data split optimization. COFFER has been tested on four different RISC-V platforms (HiFive Unmatched with SiFive U74, VisionFive 2, D1 Nezha, and the XiangShan platform) with consistent scalability characteristics, demonstrating broad hardware compatibility. We observe that the XiangShan platform does not support TLB-cached PMP entries, and COFFER relies on instruction/data split optimization alone, still achieving practical performance compared to the case without any optimization.

Addressing Inter-Enclave Communication Limitations: Regarding the reviewer's concern about strong isolation limiting inter-enclave communication, COFFER provides controlled communication mechanisms while maintaining security. COFFER implements secure message channels through our predefined SBI interfaces (`SBI_EXT_EBI_LISTEN_MESSAGE` and `SBI_EXT_EBI_SEND_MESSAGE`), enabling authorized data exchange between enclaves and the host OS. For inter-enclave communication, the current design prioritizes security over flexibility by maintaining strict isolation. However, controlled sharing could be implemented through Security Monitor-mediated shared memory regions with fine-grained access control, enabling use cases like shared libraries or collaborative computing while preserving the fundamental security guarantees, which represents a promising direction for our future work.

We have expanded Section IV-C (LPMP Design) to provide more comprehensive scalability analysis:

> The theoretical scalability of LPMP depends on the active working set of memory regions rather than total enclave memory size. When the number of active memory segments exceeds available PMP entries ($N_{seg}^{active} > N_{PMP}^{max}$), trap frequency increases proportionally. However, the instruction/data split optimization reserves dedicated

PMP entries for instruction memory, significantly reducing traps for code with high spatial locality. TLB-enhancement further extends effective PMP capacity by caching results for 2MB regions, enabling COFFER to maintain near-native performance even for large memory footprints with reasonable access locality. The selective TLB invalidation strategy minimizes flush overhead by preserving cached PMP results for non-conflicting memory regions, making LPMP practical even under memory-intensive workloads.

This analysis demonstrates that while LPMP has theoretical scalability limits, the practical performance remains excellent for realistic workloads through careful optimization and adaptive hardware utilization.

Comment 3.3: EModule Ecosystem Security

Risks in the EModule Ecosystem. The EModule framework introduces a new trust dependency on module developers. Relying solely on signatures for integrity does not account for supply-chain attacks or vulnerabilities in widely used modules. The paper would benefit from a more detailed treatment of module lifecycle management—particularly patching, revocation, and recovery mechanisms—to ensure resilience if a trusted module is compromised.

Response 3.3:

Thank you for this critical concern about EModule ecosystem security. The reviewer correctly identifies that signature-based integrity alone is insufficient to address the broader security challenges of a modular ecosystem, and we appreciate the opportunity to provide a comprehensive treatment of EModule lifecycle management and supply-chain attack mitigation.

Current Security Architecture: COFFER implements a multi-layered security architecture for EModules beyond basic signature verification. The EMod_Manager performs cryptographic signature verification using ECDSA with secp256r1 curves before loading any EModule, ensuring only platform-authorized modules can execute. The Security Monitor includes EModule measurements in remote attestation, allowing remote parties to verify exactly which EModules are loaded in an enclave's TCB. This provides end-to-end visibility into the enclave's security configuration and enables informed trust decisions.

Supply-Chain Attack Mitigation: COFFER addresses supply-chain risks through several architectural and operational mechanisms. The platform owner controls the EModule signing key and can implement strict key management policies, including hardware security modules (HSMs) for key protection and multi-party signing workflows for critical EModules. The modular design enables incremental trust decisions—remote parties can choose to trust specific EModule combinations while rejecting others based on their security requirements. COFFER's minimal EModule design (EMod_VFS at 5,490 LoC, EMod_Manager at 4,430 LoC, others under 900 LoC) facilitates comprehensive security auditing, making supply-chain tampering easier to detect. The permission-based loading mechanism allows enclaves to minimize their attack surface by loading only necessary

EModules, reducing exposure to potentially compromised modules.

Lifecycle Management and Update Mechanisms: COFFER supports secure EModule lifecycle management through several mechanisms. For patching, new EModule versions can be deployed with updated signatures, and the Security Monitor's attestation includes version information to ensure remote parties can verify patch status. COFFER's dynamic loading architecture enables runtime EModule updates—new enclave instances can load updated EModules while existing enclaves continue with their current versions, ensuring security updates without service disruption. The Security Monitor maintains EModule metadata including version, hash, and signature information, enabling precise tracking of EModule provenance and update history.

Enhanced Trust and Verification Framework: Beyond signatures, COFFER enables additional verification mechanisms. The deterministic build system ensures reproducible EModule binaries, enabling community verification of EModule integrity. The Security Monitor can be extended to support policy-based EModule loading, where platform administrators define rules about acceptable EModule combinations and versions. COFFER's design enables integration with emerging technologies like code transparency frameworks and binary provenance systems, providing cryptographic proof of EModule build integrity from source to deployment.

Compartmentalization and Damage Limitation: COFFER's architecture limits the impact of compromised EModules through strong isolation boundaries. While EModules within an enclave share the S-Mode protection domain for efficiency, each enclave runs completely isolated EModule instances—a compromised EModule in one enclave cannot access or affect EModules in other enclaves. The Security Monitor maintains strict memory isolation, preventing compromised EModules from accessing host system resources or other enclave memory. COFFER's autonomous enclave design reduces the need for complex interactions between enclaves or between enclaves and the host system, limiting the attack surface and potential for privilege escalation.

We have expanded Section VI (Security Analysis) to provide comprehensive coverage of EModule ecosystem security:

> **EModule Ecosystem Security.** COFFER addresses EModule ecosystem risks through comprehensive lifecycle management beyond signature verification. The platform owner controls EModule signing keys and can further implement multi-party signing workflows and hardware security modules for enhanced key protection. The minimal EModule design (EMod_VFS at 5,490 LoC, EMod_Manager at 4,430 LoC, others under 900 LoC) facilitates comprehensive security auditing, while strong inter-enclave isolation ensures that compromised EModules cannot affect other enclaves or the host system. Integration with reproducible builds and code transparency frameworks provides additional verification layers beyond signature-based integrity.

This comprehensive approach addresses the full spectrum of EModule ecosystem risks while maintaining the practical benefits of modular enclave construction and autonomous operation.

Comment 3.4: Hardware Dependency and Compatibility

Hardware Dependency of LPMP Optimizations. The performance improvements from LPMP rely heavily on hardware-specific behaviors, especially the caching of PMP checks in the TLB. This may not be present across all RISC-V implementations. Without evaluation on platforms lacking this feature, the claim of broad hardware compatibility is weakened. Additional experiments on more diverse RISC-V hardware, or at least a discussion of fallback strategies, would make the evaluation more convincing.

Response 3.4:

Thank you for this important observation about LPMP's hardware dependency. The reviewer correctly identifies that TLB caching of PMP checks is a platform-specific feature, and we appreciate the opportunity to provide a comprehensive analysis of COFFER's hardware compatibility and performance characteristics across diverse RISC-V implementations.

Hardware Diversity Analysis: COFFER has been implemented and tested on four distinct RISC-V platforms with varying hardware characteristics. The HiFive Unmatched (SiFive U74 core) and VisionFive 2 boards both support TLB-cached PMP checking with two-level TLB hierarchies (L1: 40-entry fully-associative instruction/data TLBs, L2: 512-entry direct-mapped hybrid TLB). The D1 Nezha board (Allwinner C906 core) and the XiangShan platform do not support TLB-cached PMP checking, providing direct validation of COFFER's fallback behavior. This diverse testing validates the broad hardware compatibility of our LPMP mechanism across different processor microarchitectures and TLB configurations.

Performance Without TLB Caching: On platforms lacking TLB-cached PMP checking (such as D1 Nezha and XiangShan), COFFER gracefully degrades to rely on the instruction/data split optimization. The instruction/data split reserves dedicated PMP entries for instruction memory, which typically exhibits high spatial and temporal locality. This optimization alone provides substantial performance benefits even without TLB enhancement. Our evaluation demonstrates that COFFER maintains practical performance on the D1 Nezha and the XiangShan platforms, compared to the case without any optimization of LPMP, because the instruction/data split effectively reducing the LPMP trap frequency. While TLB-enhancement provides additional performance gains on supporting platforms, it is not required for COFFER's fundamental functionality or security guarantees.

Quantitative Performance Analysis: The performance impact of TLB-enhancement varies by workload characteristics. Our evaluation under worst-case memory fragmentation demonstrates that instruction/data split optimization is particularly effective for memory-intensive workloads (such as stress-ng vm workers), while TLB-enhancement provides greater benefits for compute-intensive workloads (such as RV8 benchmarks). With both optimizations enabled, COFFER achieves less than 3% overhead even under worst-case memory fragmentation (near native performance). For general performance, COFFER achieves within 5% overhead on RV8 benchmarks and within 7% overhead for enclaves with memory sizes up to 2GB, demonstrating that the core LPMP design provides excellent

efficiency across diverse platforms.

Security-Performance Tradeoff Analysis: COFFER's TLB utilization strategy carefully balances performance and security. On platforms with TLB caching support, COFFER enforces two security principles: (1) exclusive ownership of TLB through full TLB flushes on context switches, and (2) freshness guarantee through TLB flushes when LPMP lists are modified. These principles ensure that TLB-enhancement does not compromise security while providing performance benefits. The selective TLB invalidation during LPMP traps preserves security by only maintaining TLB entries consistent with the enclave's LPMP list.

Future Hardware Compatibility: COFFER's design anticipates future RISC-V processor evolution. The modular LPMP architecture can seamlessly integrate additional hardware features as they become available, such as increased PMP entry counts (according to the RISC-V specification, it can be up to 64), tagged TLB entries for security domain isolation, or hardware-assisted PMP virtualization. COFFER's software-based approach ensures backward compatibility with existing platforms while enabling forward compatibility with enhanced hardware. The consistent behavior observed across our four test platforms (HiFive Unmatched, VisionFive 2, D1 Nezha and XiangShan) demonstrates COFFER's robust adaptation to hardware diversity.

We have expanded Section IV-C (LPMP Design) to provide more comprehensive hardware dependency discussion:

> The TLB-enhancement optimization leverages TLB caching of PMP checking results, a feature present on many RISC-V platforms including HiFive Unmatched and VisionFive 2. On platforms without this feature (such as D1 Nezha and XiangShan), COFFER gracefully degrades to rely on instruction/data split optimization alone, which still provides substantial performance benefits through dedicated PMP entries for instruction memory with high locality. COFFER has been tested on four platforms with varying TLB configurations (HiFive Unmatched, VisionFive 2, D1 Nezha, and XiangShan), demonstrating correct operation across diverse RISC-V implementations. While TLB-enhancement provides additional performance gains for certain workloads (particularly compute-intensive benchmarks), it is not required for COFFER's security guarantees or fundamental functionality. The consistent performance characteristics across these test platforms validate COFFER's broad hardware compatibility.

This comprehensive analysis demonstrates that COFFER's hardware compatibility extends beyond TLB-caching platforms, with adaptive optimization strategies ensuring efficient operation across the diverse landscape of RISC-V implementations.

Comment 3.5: Security Analysis and Trade-offs

Limited Security Analysis and Trade-offs. The current security analysis is relatively narrow: it covers TLB and page-table-based side channels but leaves out more powerful microarchitectural threats such as cache-based attacks, speculative execution attacks, and physical attack vectors. This omission leaves enclaves potentially vulnerable. Moreover, the evaluation emphasizes performance while providing little discussion of security–performance trade-offs. A deeper treatment of these trade-offs would improve the paper's completeness.

Response 3.5:

Thank you for this constructive feedback on expanding our security analysis. The reviewer correctly identifies that while COFFER's threat model appropriately scopes the security guarantees, a more comprehensive discussion of out-of-scope threats and security-performance tradeoffs would strengthen the paper's completeness. We appreciate the opportunity to provide this expanded treatment.

Scope of Threat Model and Security Guarantees: COFFER's threat model (Section III) explicitly defines its security scope: we assume a privileged attacker with full OS kernel control who attempts to access enclave memory or manipulate binaries, while physical attacks, side-channel attacks, and DoS attacks are out of scope. This scoping is intentional and aligns with practical TEE deployment scenarios where COFFER provides strong guarantees against software-based attacks while acknowledging that certain attack classes require hardware modifications beyond commodity RISC-V platforms. We address attacks within our threat model comprehensively through PMP-based memory isolation, binary attestation, and autonomous enclave design minimizing OS dependencies.

Microarchitectural Side-Channel Attacks: The reviewer correctly notes that our current security analysis focuses on TLB and page-table-based side channels. We acknowledge that more powerful microarchitectural threats exist. Cache-based attacks (such as Prime+Probe, Flush+Reload, and Evict+Reload) exploit shared cache resources and require hardware modifications for complete mitigation—such as cache partitioning, random cache indexing, or cache line locking. These defenses are orthogonal to COFFER's software-based design and remain active research areas even for hardware-assisted TEEs like Intel SGX. Speculative execution attacks (such as Spectre and Meltdown variants) exploit transient execution and speculative loads, requiring processor microarchitecture changes including speculation barriers, bounds check bypass prevention, or architectural register file isolation. Physical attack vectors (such as cold boot attacks, bus probing, or fault injection) require physical access to hardware and are typically addressed through hardware security modules, memory encryption, and tamper-resistant packaging rather than software TEE mechanisms.

Rationale for Threat Model Scope: COFFER's threat model focuses on software-based attacks because these represent the primary threat vector in cloud and edge computing deployments where COFFER targets. Our design builds upon commodity RISC-V hardware with standard PMP support, deliberately avoiding dependency on specific optional extensions to maximize deployment compatibility. This pragmatic approach enables immediate deployment on existing RISC-V platforms while providing forward compatibility

with emerging security extensions such as IOMMU and IOPMP. Other TEE systems make similar scoping decisions: Intel SGX excludes physical attacks and certain side-channels from its threat model, AMD SEV-SNP focuses on memory encryption while delegating I/O protection to IOMMU, and ARM TrustZone leaves cache-based side-channel mitigation to software countermeasures.

Security-Performance Tradeoff Analysis: COFFER's design embodies several deliberate security-performance tradeoffs. The LPMP trap-and-emulate approach trades performance (trap overhead) for security (fine-grained memory isolation without hardware modification). Our evaluation shows this tradeoff is favorable: less than 3% overhead under worst-case fragmentation and within 5% overhead for general benchmarks. The instruction/data split optimization improves performance while maintaining security through dedicated PMP entries preventing code memory thrashing. TLB-enhancement optimization leverages TLB caching for performance while enforcing two security principles: exclusive TLB ownership (full flushes on context switches) and freshness guarantee (TLB flushes on LPMP list modifications). These principles ensure TLB utilization does not compromise isolation. The autonomous enclave design with EModules trades increased TCB size (EModules in enclave memory) for security (reduced OS attack surface by eliminating Iago attacks and controlled-channel attacks). Our TCB analysis shows this tradeoff is acceptable: the Security Monitor plus EMod_Manager total 21,473 LoC, comparable to other TEE systems.

Comparative Security Posture: COFFER's security guarantees are comparable to other software-based TEEs on RISC-V while providing unique advantages. Compared to Keystone, COFFER provides strong scalability through LPMP (PMP virtualization) enabling fine-grained memory protection, while Keystone relies on coarse-grained PMP entries limiting memory flexibility. Compared to Penglai, COFFER's autonomous enclave design aims at reducing attack surface through EModules, whereas Penglai enclaves depend on OS for system call handling creating controlled-channel vulnerabilities. COFFER's modular architecture enables security-TCB customization through dynamic loading at runtime and permission-based EModule loading, allowing enclaves to include only necessary functionality—a capability not present in monolithic TEE designs.

Deployment Recommendations and Limitations: For deployments requiring protection beyond COFFER's threat model, we recommend defense-in-depth strategies. Against cache-based side-channels, applications can employ algorithmic countermeasures such as constant-time implementations, data-oblivious algorithms, or software-based cache partitioning through careful memory access patterns. Against speculative execution attacks, applications can use speculation barriers at security-critical boundaries and avoid secret-dependent branches. Against physical attacks, deployments should employ hardware security modules for key storage, tamper-evident packaging, and secure boot chains. COFFER's design complements these countermeasures by providing the foundational memory isolation and attestation required for secure enclave execution.

We have expanded Section VI (Security Analysis) to provide comprehensive discussion of threat model scope and security-performance tradeoffs:

Threat Model Scope and Limitations. COFFER's threat model focuses on software-based attacks from privileged adversaries with full OS control, providing strong guarantees through PMP-based memory isolation, binary attestation, and autonomous enclave design. Physical attacks, microarchitectural side-channel attacks (cache-based, speculative execution), and DoS attacks are explicitly out of scope, as defending against these threats requires hardware modifications beyond commodity RISC-V platforms. This scoping aligns with practical TEE deployment scenarios and is consistent with other TEE systems (Intel SGX, AMD SEV-SNP, ARM TrustZone) that similarly exclude certain attack classes from their threat models. For deployments requiring broader protection, COFFER can provide forward compatibility with emerging security extensions such as IOMMU/IOPMP and other security extensions, and can be combined with defense-in-depth strategies including algorithmic countermeasures for side-channels, hardware security modules for physical security, and emerging RISC-V security extensions as they become available.

Security-Performance Tradeoffs. COFFER's design embodies several deliberate security-performance tradeoffs. The LPMP trap-and-emulate approach trades trap overhead for fine-grained memory isolation (less than 3% overhead under worst-case fragmentation). TLB-enhancement optimization improves performance while enforcing exclusive TLB ownership and freshness guarantee to prevent TLB-based attacks. The autonomous enclave design trades increased TCB (Security Monitor plus EMod_Manager: 21,473 LoC) for reduced OS attack surface by eliminating dependencies that enable Iago and controlled-channel attacks. These tradeoffs result in a practical TEE system with strong security guarantees against in-scope threats while maintaining efficiency comparable to native execution.

This expanded treatment acknowledges the limitations of COFFER's threat model while providing clear guidance on the security guarantees provided and strategies for addressing out-of-scope threats in practical deployments.