Poster: Isolating PIM from OS Level Adversaries

Fabian van Rissenbeck[1], Amit Choudhari[2], and Christian Rossow[2]

[1] TU Dortmund University, August-Schmidt-Straße 1, 44227 Dortmund, Germany
[2] CISPA Helmholtz Center for Information Security, Stuhlsatzenhaus 5, 66123 Saarbrücken, Germany

1 Motivation and Background

Modern cloud infrastructures run data-intensive workloads in multi-tenant environments, where shared hardware resources–especially CPU caches–are vulnerable to side-channel attacks that reveal access patterns [2]. While cache isolation can mitigate these risks, it is impractical at scale [3]. Processing-in-Memory (PIM) architectures, like UPMEM's, restructure traditional von Neumann architectures by embedding small RISC-style processors directly into DRAM chips [5, 4]. A PIM architecture can be beneficial for combating some classes of side-channel attacks, because moving computation closer to data inherently reduces the amount of cache-based leakage. Current PIM systems lack critical security primitives such as secure key storage and random number generation, however some simulated PIM architectures have included such hardware extensions [1]. Additionally, we currently see no mechanism to leverage Trusted Execution Environments (TEEs) like Intel SGX and Arm TrustZone to extend their protection to PIM modules, leaving PIM fully exposed in scenarios with strong adversaries. We propose a software-based solution leveraging a trusted hypervisor and TPM to enable PIM computation within a trusted environment.

2 Problem Statement

A host system utilizes two channels for communication with UPMEM's Data Processing Units (DPUs): A shared memory between host and DPU for transferring bulk data (MRAM) and a hardware register called the Control Interface (CI). The CI exposes primitives to load new code, access DPU internal registers and read the contents of DPU local memories. A malicious operating system could easily abuse the CIs to extract sensitive data or manipulate the results of an otherwise benign compute task, even the execution of malicious DPU code can be forced using its CI. Securing the CIs is therefore critical. Furthermore, data transfers between the CPU and DPUs, stored unencrypted in MRAM, are vulnerable to interception or tampering. Securing MRAM based data transfers via encryption is non-trivial as the DPUs lack hardware-based sources of entropy, which prevents secure local key generation. Finally, DPUs lack direct intercommunication paths, forcing all traffic to pass through the potentially compromised host system (i.e. via CPU). This bottleneck severely limits scalability for workloads that depend on frequent aggregation of results across multiple DPUs.

3 Proposed Approach

We address the issue with the adversary-exposed CIs, by introducing a thin, trusted hypervisor, that unmaps the CIs and replaces them with a software-defined, Virtual Control Interface (VCI). The VCI commands are interpreted by a small program that we call the CI-switch, embedded within trusted hypervisor. To minimize the attack surface, we implement only the most essential CI functionalities in the VCI, explicitly removing capabilities such as loading new code.

Since the VCI is significantly less powerful than its hardware counterpart, we modify the communication paradigm with DPUs. Instead of deploying short-lived, single task DPU kernels, we use a persistent DPU kernel deployed at boot time by the trusted hypervisor, that implements all necessary functions for a given task. To interact with these functions, we define a half-duplex messaging protocol, carried over MRAM. As MRAM accesses need to be synchronized between DPU and host, we leverage a DPU feature that allows it to enter fault states, which the host can then resolve.

To secure the message transmission over MRAM–an adversary exposed channel–we include a compact implementation of the AES-CCM authenticated cipher mode within the DPU kernel. This ensures encryption and authentication and a replay-detection mechanism. A fresh key is derived for each client, based on an initial value provided on startup by the trusted hypervisor and with assistance from a trusted third party.

Finally, to scale up computation across multiple DPUs, we introduce a user-level router that facilitates data exchange between client and DPUs. This enables efficient broadcasting, inter-DPU communication and allowing aggregation steps to be performed directly on the DPUs by forwarding inter-DPU messages. The router software operates without the knowledge of any DPU internal secrets and cannot decrypt the messages that it forwards.

4 Experimental Validation

We evaluated our framework using a decision tree classification workload (prone to memory access pattern leaks), and compared it against a baseline PIM setup, a naive CPU implementation, and an oblivious (ORAM-based) CPU version. For large inputs ($\geq$1GiB), the secure PIM platform is 27x slower than the baseline PIM and 7$\times$ slower than the naive CPU but outperforms the ORAM-protected CPU by 18x. This performance gap is mainly due to software-based encryption and decryption, accounting for up to 95% of total DPU runtime in lightweight workloads. Additional overhead arises from single-threaded client-side encryption and inefficient data movement in the user-level router, both of which could be improved through multithreading and optimized data handling. Virtualization overhead is minimal, indicating that the main performance trade-offs originate from the secure PIM design rather than the hypervisor.

Currently, the DPU kernel cannot be dynamically replaced after boot. Data exchange relies on fault-state synchronization and a messaging model, differing from traditional host-driven PIM usage.

5 Conclusion and Future Directions

Our software-only security framework isolates control registers through virtualization and secures data in MRAM with lightweight encryption. Despite of the performance overhead, the platform significantly outperforms CPU-based oblivious models for access-pattern protection. For future work, we will investigate mechanisms to allow the safe execution of user provided code, including an attestation mechanism, to enhance platform flexibility. We further aim to reduce the trusted hypervisor's responsibilities, integration of TPMs and TEEs for enhanced hypervisor security and key provisioning.

This study demonstrates that a trusted hypervisor, combined with cryptographically hardened DPUs, can close critical security gaps in PIM systems, enabling secure, multi-tenant cloud computation without the need for hardware modifications.

References

1. Duy, K.D., Lee, H.: SE-PIM: in-memory acceleration of data-intensive confidential computing. IEEE Trans. Cloud Comput. (2023). https://doi.org/10.1109/TCC.2022.3207145, https://doi.org/10.1109/TCC.2022.3207145
2. Godfrey, M.M., Zulkernine, M.: Preventing cache-based side-channel attacks in a cloud environment. IEEE Trans. Cloud Comput. (2014). https://doi.org/10.1109/TCC.2014.2358236, https://doi.org/10.1109/TCC.2014.2358236
3. Mittal, S.: A survey of techniques for cache partitioning in multicore processors. ACM Comput. Surv. (2017). https://doi.org/10.1145/3062394, https://doi.org/10.1145/3062394
4. Mutlu, O., Ghose, S., Gómez-Luna, J., Ausavarungnirun, R.: A Modern Primer on Processing in Memory. CoRR (2020), https://arxiv.org/abs/2012.03112
5. UPMEM: Processing In-Memory: Ultra-Efficient Acceleration for Data-Intensive Applications. Tech. rep., UPMEM (2024), https://www.upmem.com/technology/