KAIST CySecLab

SSLab

PIM-ORAM: Towards Oblivious RAM Primitives in Commodity Processing-In-Memory

Byeongsu Woo (KAIST), Kha Dinh Duy (Sungkyunkwan University), Youngkwang Han (New York University Abu Dhabi), Brent Byunghoon Kang* (KAIST), Hojoon Lee* (Sungkyunkwan University)

Annual Computer Security Applications Conference (ACSAC) 2025

Introduction

Side-channel attack in Trusted Execution Environment (TEE)

- **TEE provides secured and isolated execution environment (e.g., Intel SGX's enclave)**
- **Application's runtime memory area is encrypted**
- **Suffers side-channel attacks [1, 2, 3]**
  - Memory side-channel attack can extract protected data in enclave

Oblivious RAM (ORAM)

- **ORAM has been discussed as a solution against the memory side-channel attacks**
  - Obfuscates memory access pattern
- **However, ORAM suffers high overhead because of additional memory access**

[1] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. 2017. Software Grand Exposure: SGX Cache Attacks Are Practical. In 11th USENIX Workshop on Offensive Technologies (WOOT 17). USENIX Association, Vancouver, BC.
[2] Ahmad Moghimi, Gorka Irazoqui, and Thomas Eisenbarth. 2017. CacheZoom: How SGX Amplifies The Power of Cache Attacks. CoRR abs/1703.06986 (2017). arXiv:1703.06986
[3] Yuval Yarom and Katrina Falkner. 2014. {FLUSH+ RELOAD}: A High Resolution, Low Noise, L3 Cache {Side-Channel} Attack. In 23rd USENIX security symposium (USENIX security 14). 719–732.

Introduction

Processing-In-Memory (PIM)

- **A new paradigm introducing computation power into memory**
- **Reducing data transfer between CPU and memory**

PIM-ORAM

- **Reducing data transfer between CPU and memory incurred by ORAM**
- **Accelerating ORAM primitives with commodity PIM**
  - Leveraging PIM's parallelization capability
  - Exploring the new direction of PIM-accelerated ORAM
  - Evaluating the prototype on **a real commodity PIM hardware**

Motivation

Related Works

- **Smart memory-based works tried to obfuscate the memory access pattern**
  - Invisimem [4], Obfusmem [5], SecureDIMM [6]
  - Emulated on ideal smart memory device capability [4, 6]:
    - Server-grade processors (1.6 ~ 2.5GHz) with multi-level caches are used as processing units
- **There is no related work using commodity real-hardware PIM**

PIM-ORAM

- **Utilizes real-world hardware PIM to provide most realistic results**
- **Explores how the current hardware PIM's design might better support secure and oblivious computation**

[4] Aga, Shaizeen, and Satish Narayanasamy. "Invisimem: Smart memory defenses for memory bus side channel." ACM SIGARCH Computer Architecture News 45.2 (2017): 94-106.[12] Yuval Yarom and Katrina Falkner. 2014. {FLUSH+ RELOAD}: A High Resolution, Low Noise, L3 Cache {Side-Channel} Attack. In 23rd USENIX security symposium (USENIX security 14). 719–732.
[5] Awad, Amro, et al. "Obfusmem: A low-overhead access obfuscation for trusted memories." Proceedings of the 44th Annual International Symposium on Computer Architecture. 2017.
[6] Shafiee, Ali, et al. "Secure DIMM: Moving ORAM primitives closer to memory." 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2018.

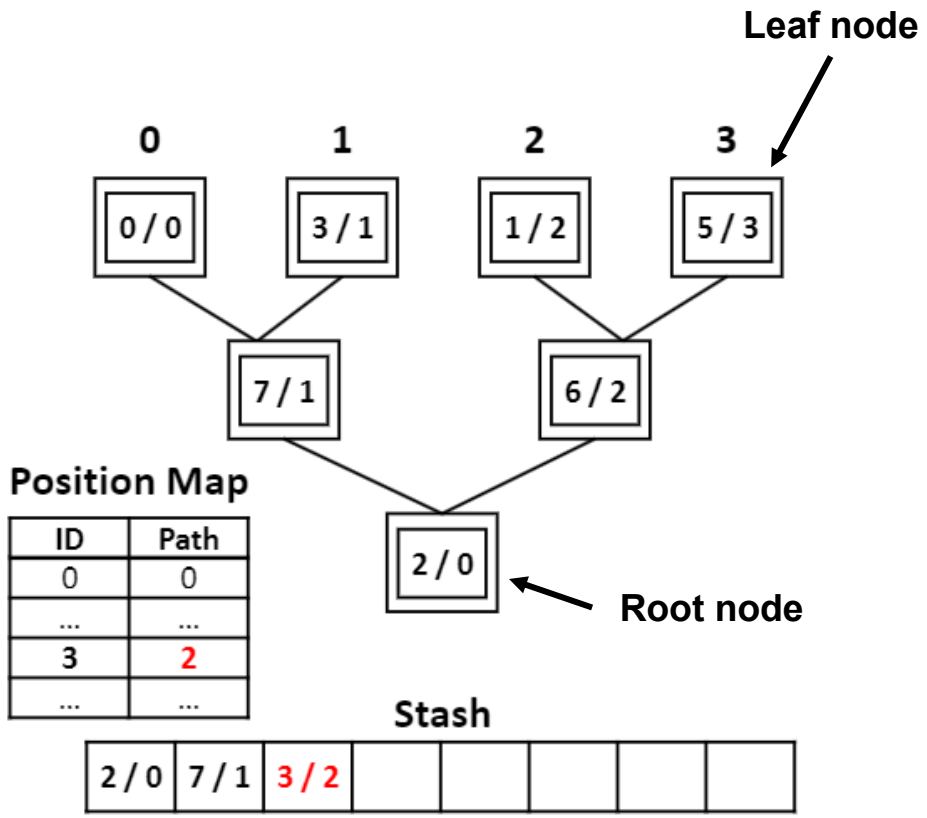
Oblivious RAM (ORAM)

- **Algorithms making each data access sequence indistinguishable**
  - Data access sequence $\overrightarrow{y_1}$ and $\overrightarrow{y_2}$ must be indistinguishable

$$\vec{y} = ((op_1, id_1, data_1), \ldots, (op_n, id_n, data_n))$$

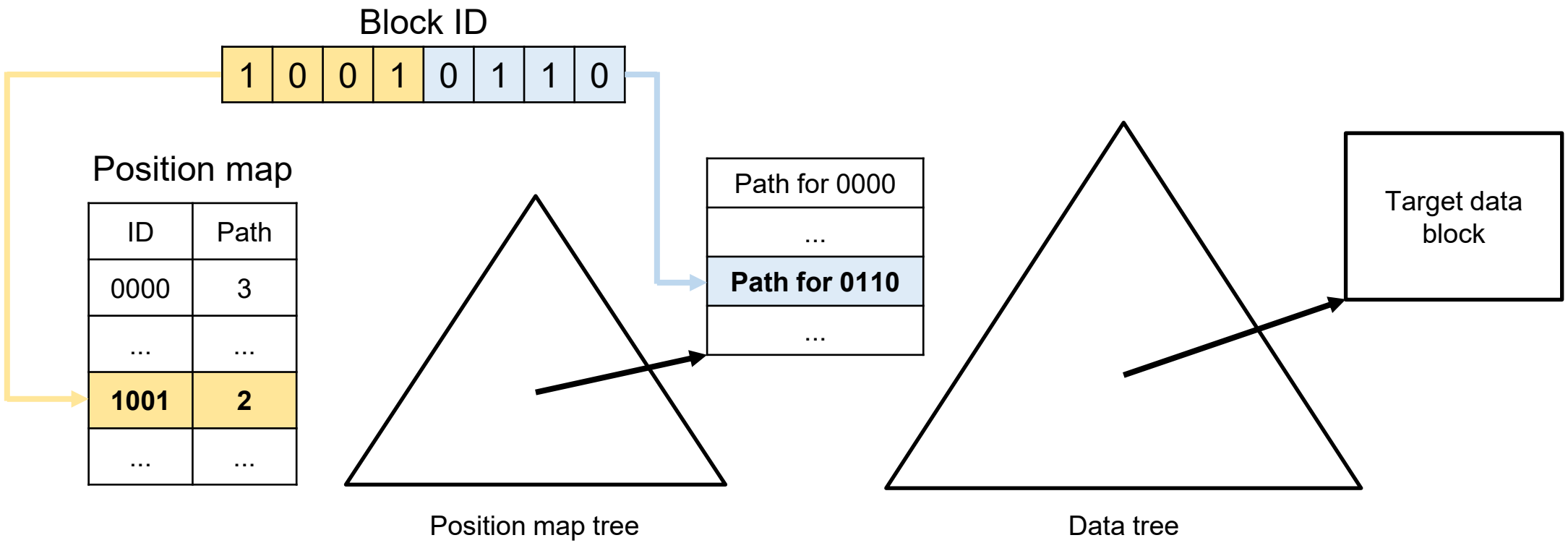
Path ORAM

- **One of the well-known and simplest ORAM algorithm**
  - Tree based ORAM
  - Path: a path from tree's root node to a leaf node
- **Component in untrusted part**
  - ORAM tree: data storage. Contents are encrypted.
- **Components in trusted part**
  - Stash: a temporal cache storing fetched data blocks from ORAM tree
  - Position map: a map containing the mapping information between block ID and path



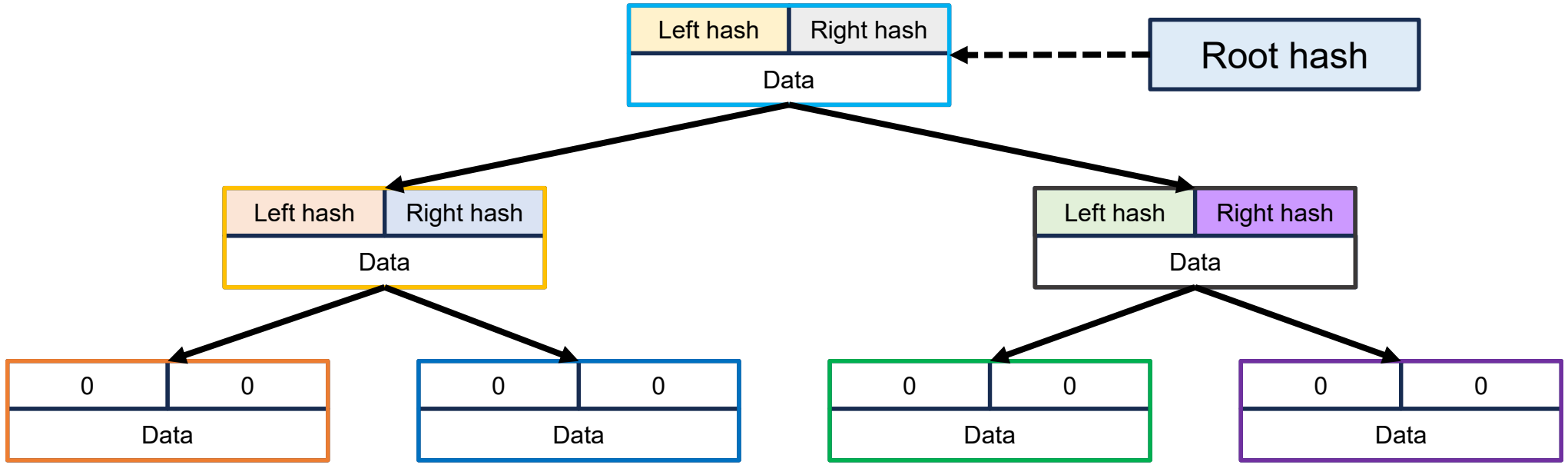
Path ORAM Extension: Recursive ORAM

- **ORAM can be applied recursively to prevent position map from bloating**



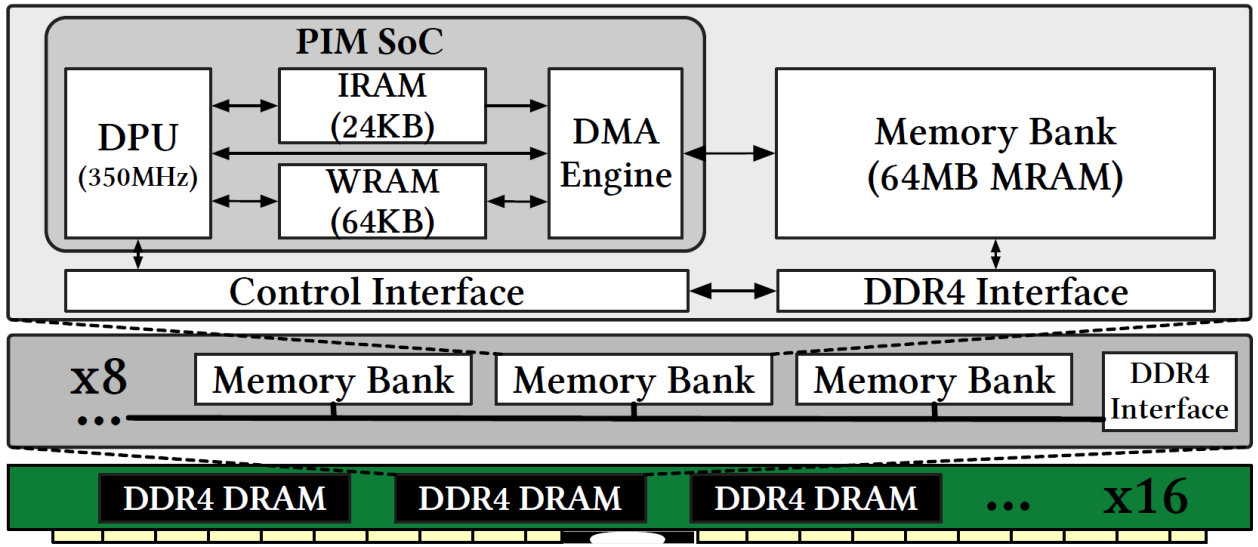
Path ORAM Extension: Merkle Tree

- Merkle tree scheme can be applied to guarantee integrity of the ORAM tree



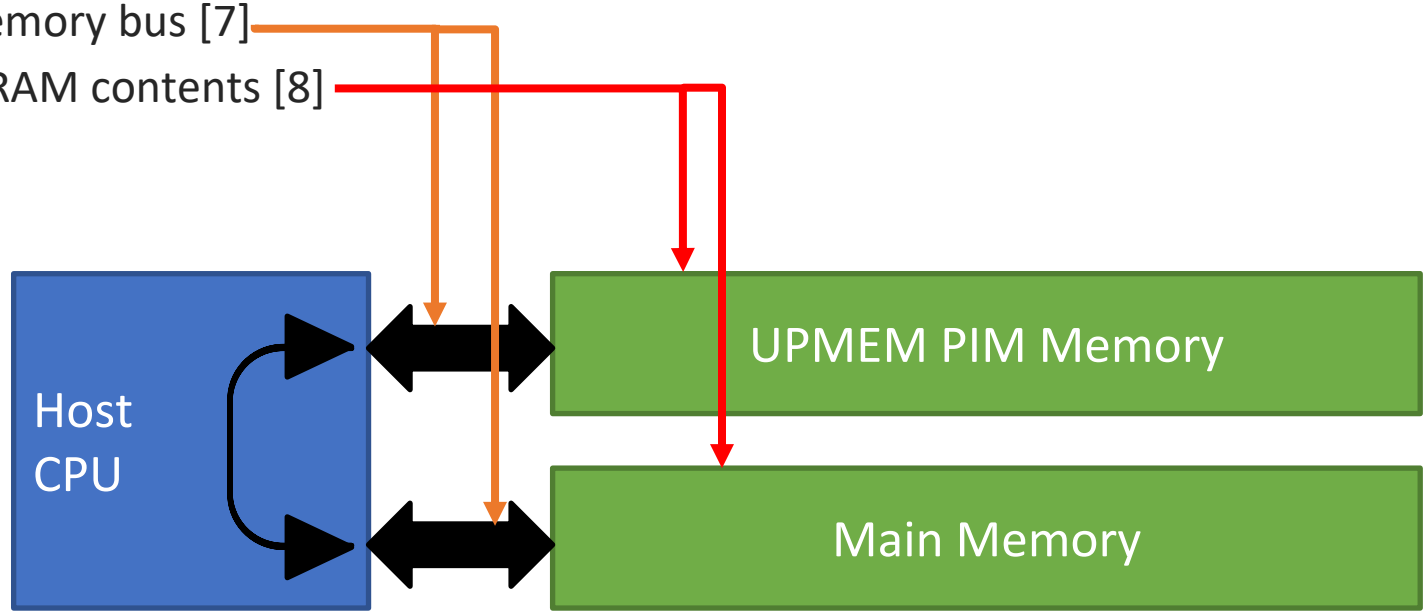
UPMEM

- **Currently only available real-hardware commodity PIM**
- **Each memory bank has below components**
  - **DPU**: DRAM Processing Unit
  - **IRAM**: SRAM for storing PIM program (24KB)
  - **WRAM**: DPU's scratchpad SRAM (64KB)
  - **MRAM**: DRAM for a DPU bank (64MB)
  - **DMA engine**: managing the data transfer between memories
- **Memory banks operate in parallel**



Threat model

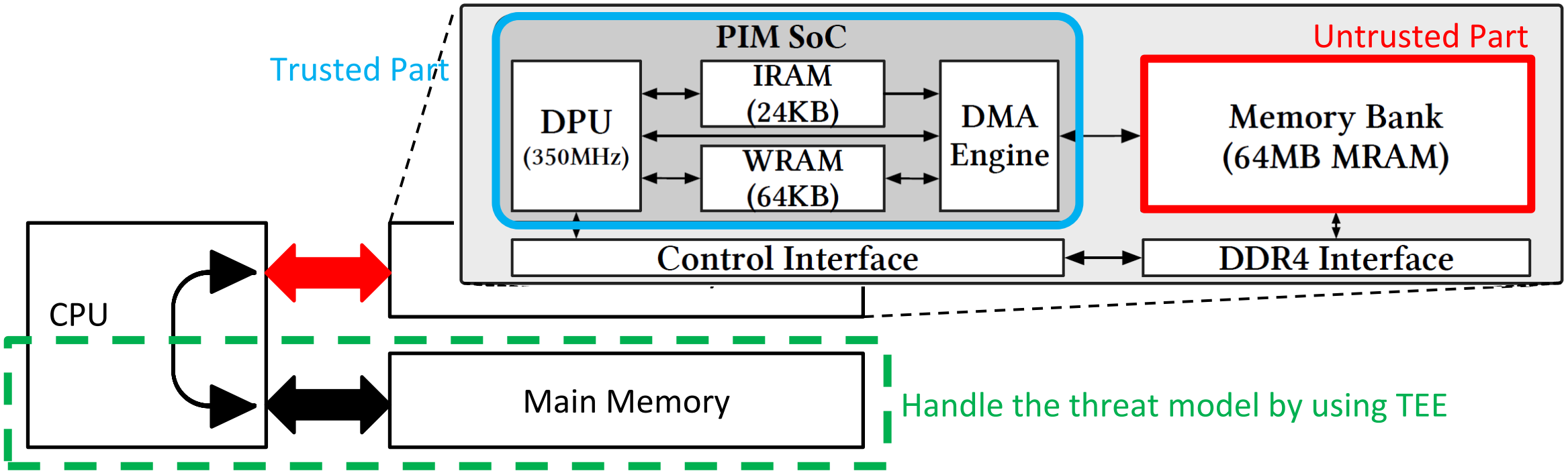
- **A strong adversary model that is conventional in TEE model**
  - Full control over system software
- **Physical attack against memory bus and memory**
  - Snooping memory bus [7]
  - Extracting DRAM contents [8]



[7] Dayeol Lee, Dongha Jung, Ian T. Fang, Chia che Tsai, and Raluca Ada Popa. 2020. An Off-Chip Attack on Hardware Enclaves via the Memory Bus. In 29th USENIX Security Symposium (USENIX Security 20). USENIX Association, 487–504.
[8] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. 2008. Lest We Remember: Cold Boot Attacks on Encryption Keys. In Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA, Paul C. van Oorschot (Ed.). USENIX Association, 45–60.

Assumption

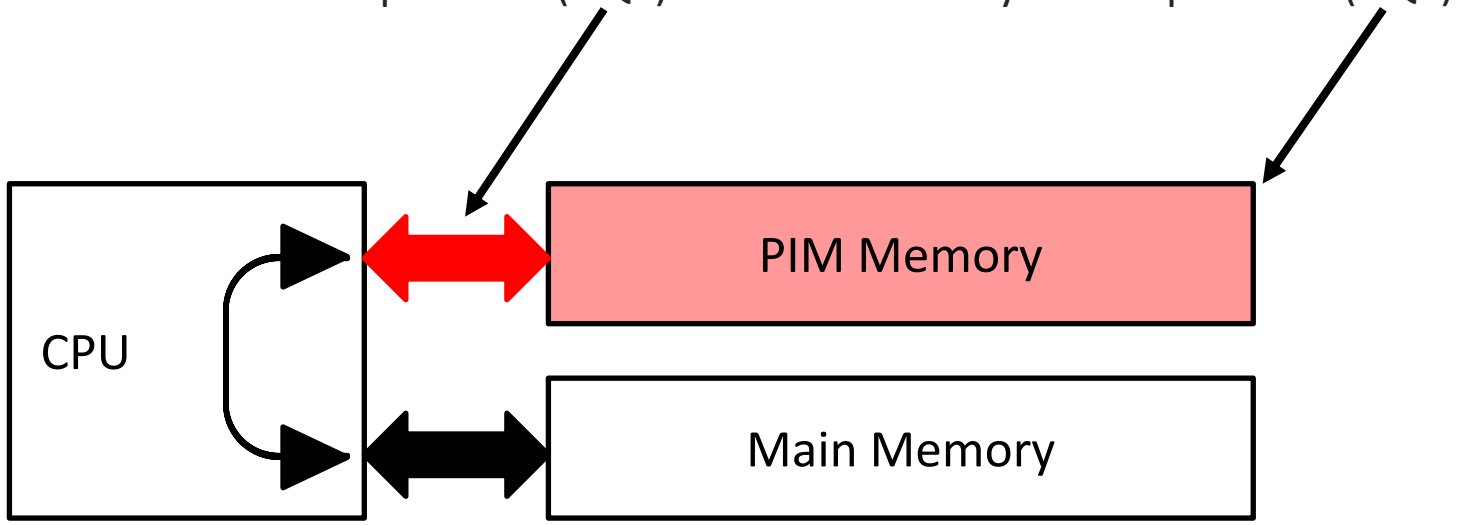
- **The CPU has the capability of establishing Trusted Execution Environment**
- **The side-channel against the host is out of scope**
  - There are previous works suggesting mitigations [9, 10]
  - We'll focus on PIM and the memory bus between the host and the PIM
- **In the PIM, we trust PIM SoC but not trust MRAM**



[9] Ashay Rane, Calvin Lin, and Mohit Tiwari. 2015. Raccoon: Closing Digital Side-Channels through Obfuscated Execution. In 24th USENIX Security Symposium (USENIX Security 15). USENIX Association, Washington, D.C., 431–446.
[10] Sajin Sasy, Sergey Gorbunov, and Christopher W Fletcher. 2018. ZeroTrace: Oblivious Memory Primitives from Intel SGX.. In NDSS.

Security requirements

- **Securing the communication channel between the CPU and the PIM**
  - Encrypting the communication channel
- **Maintaining ORAM's obliviousness guarantees**
  - Removing the correlation between the plaintext and the ciphertext
  - Retaining obliviousness in bus traffic patterns (**RQ1**) and PIM memory access patterns (**RQ2**)

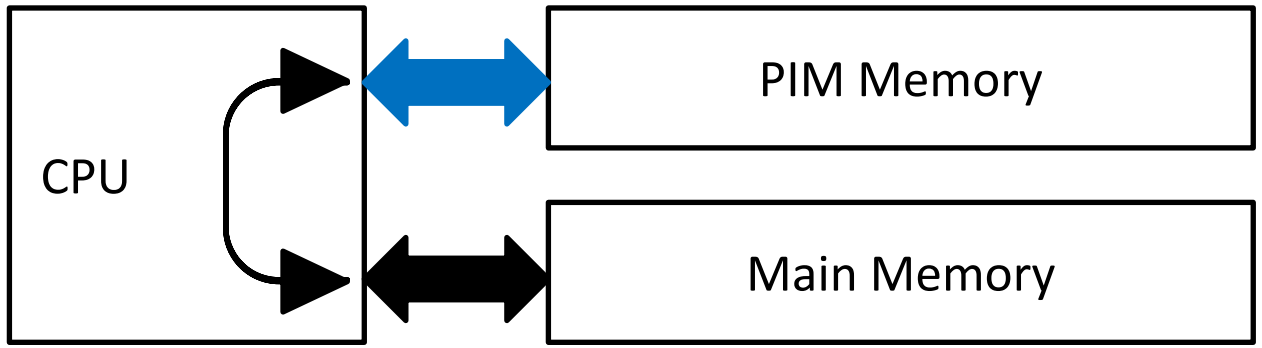


PIM confidential compute model

- The existing commodity PIM-based PIM confidential compute model
- Introducing two simulated component into the design

Attestation module

- For establishing a secure channel between the host and the PIM
- Protecting the loaded PIM program and PIM-ORAM commands

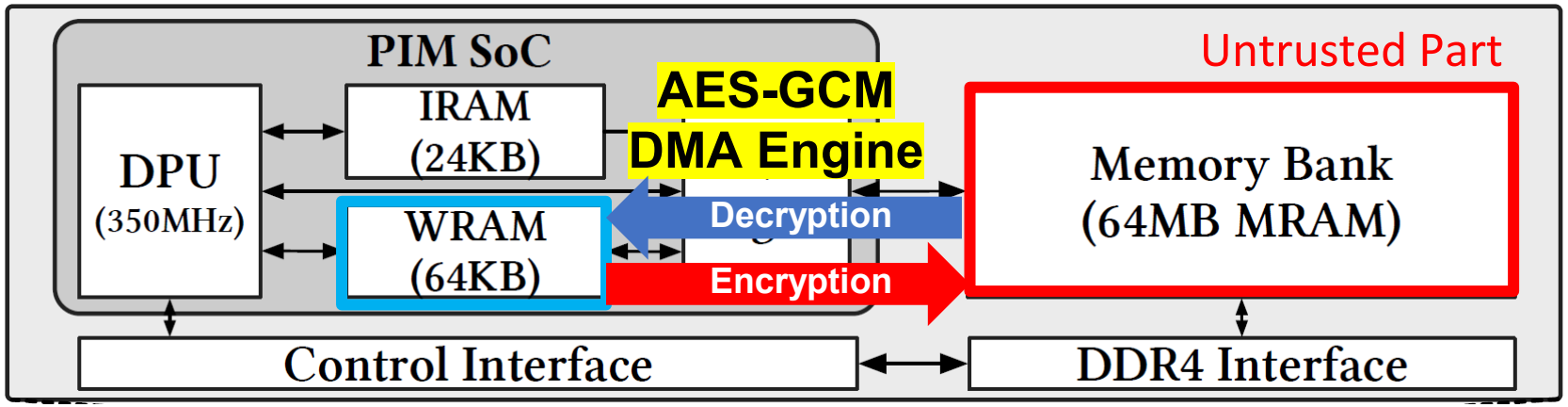


PIM confidential compute model

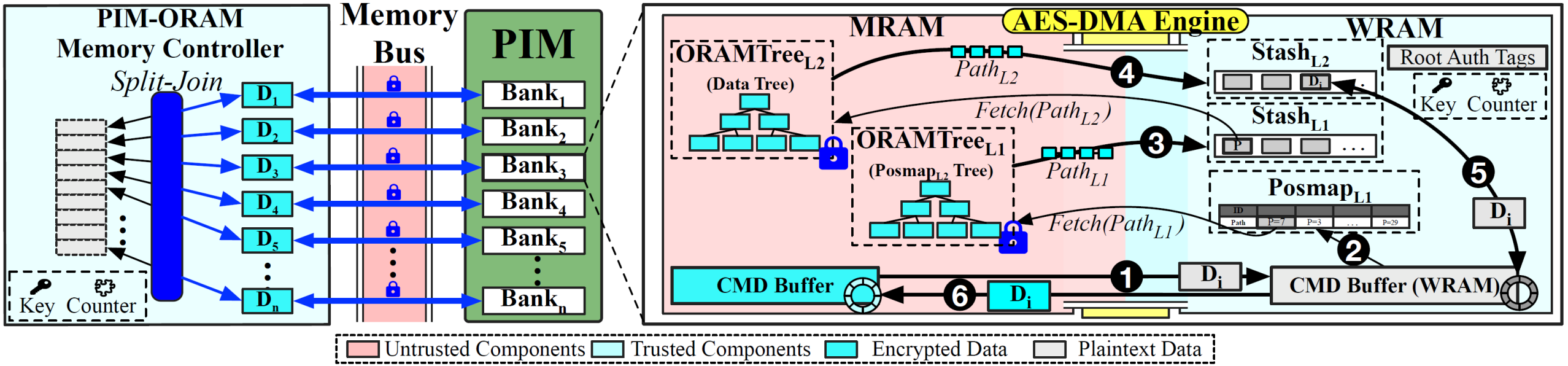
- The existing commodity PIM-based PIM confidential compute model
- Introducing two simulated hardware into the design

AES-capable DMA engine

- It is inefficient to make DPU encrypt data due to the DPU's limited computation power (350MHz)
- Encrypts all data when data moves from WRAM to MRAM
- Decrypts all data when data moves from MRAM to WRAM



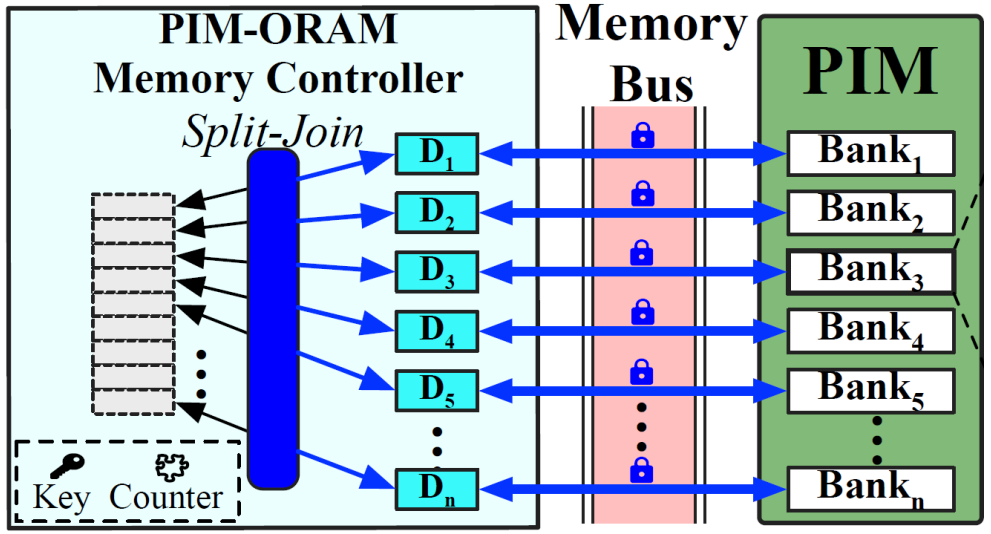
Design overview



Host-side design

PIM-ORAM memory controller (POMC)

- Software-level memory controller on the host
- Managing the execution of PIM and the communication channel

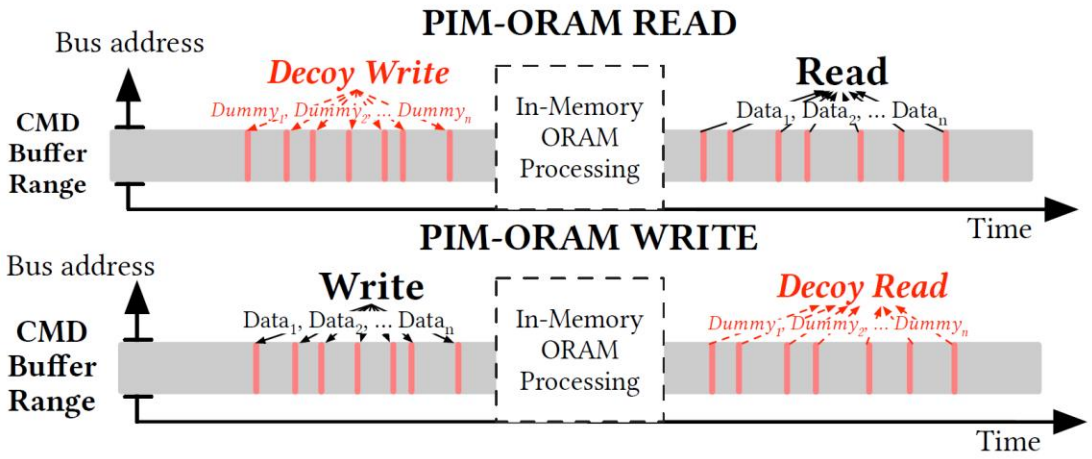


Split-data ORAM

- Each PIM manages its own ORAM independently
- Splitting data block into data pieces and processing them in parallel

Split-Join process

- The process splitting and joining data block in POMC
- Dummy data can be split or joined to retain obliviousness in memory bus



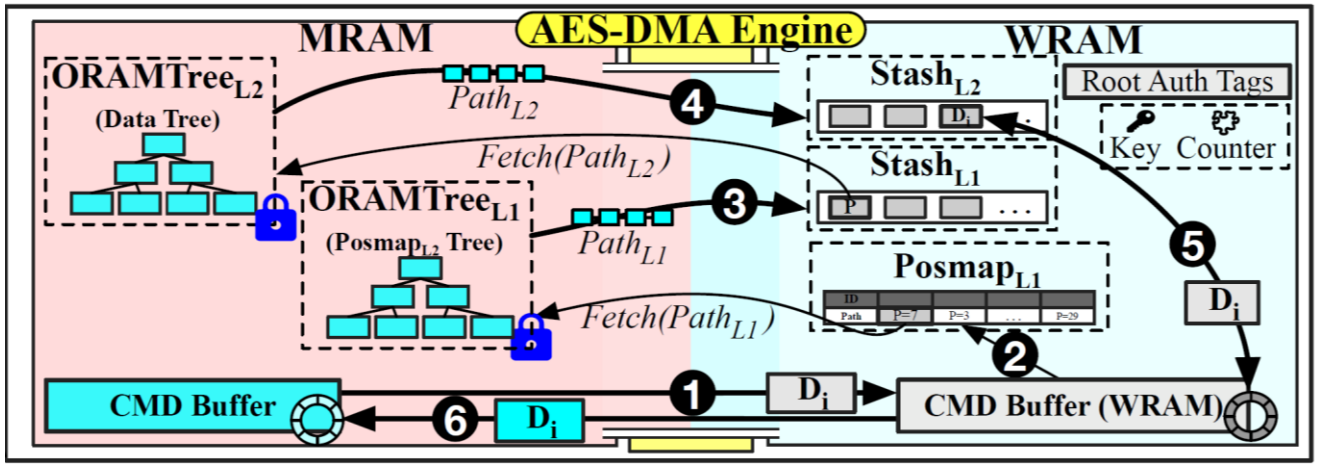
PIM-side design

Components in MRAM

- **Command buffer**
  - The buffer temporally contains encrypted data transferred from or to the host
  - Contains command (Initialize, Read, Write), block ID, and data piece (or dummy piece)
- **Position map tree ($ORAMTree_{pos}$)**
- **Data tree ($ORAMTree_{data}$)**

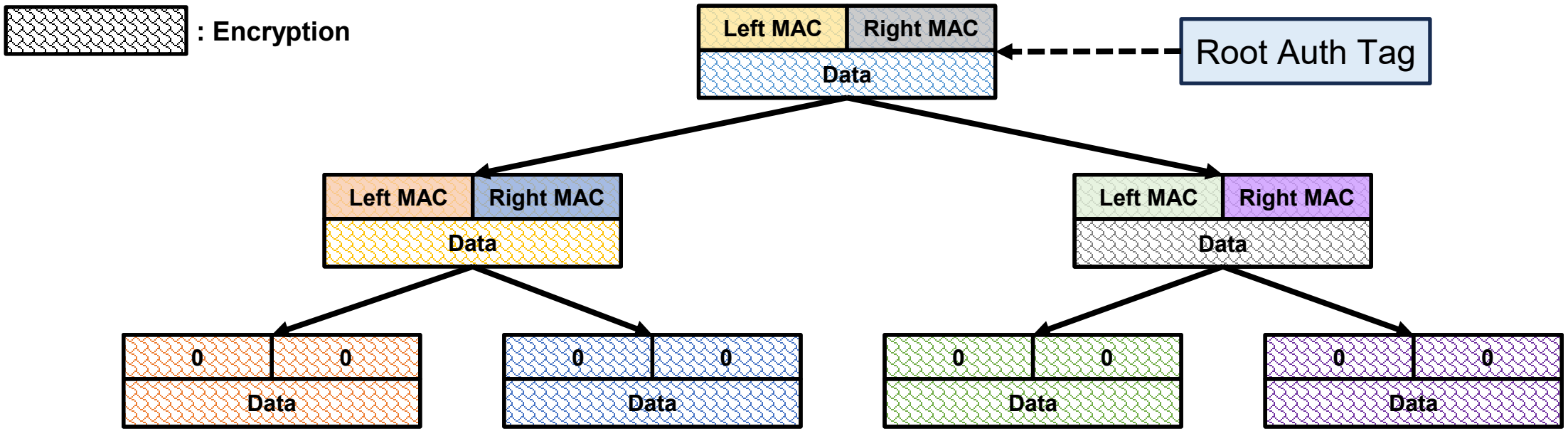
Components in WRAM

- **Decrypted command buffer ($Command\ Buffer_{WRAM}$)**
- **Position map for position map tree ($Posmap_{pos}$)**
- **Stash for position map tree ($Stash_{pos}$)**
- **Stash for data tree ($Stash_{data}$)**
- **Root Authentication Tags of ORAMTrees, Key and Counter for encryption**



Lightweight integrity verification

- **Use Authentication Tag of AES-GCM, instead of using hash function such as SHA256**
  - Prevent PIM from introducing additional hardware assumption for hash accelerator
- **Child node's Authentication Tag is stored in its parent node**
- **Authentication Tags in the nodes are fully encrypted**
- **Root node's Authentication Tag is stored in WRAM**



Evaluation Setup

- **PIM-ORAM is evaluated on UPMEM test-bed data center**
  - 2 x Intel Xeon Silver 4110 CPU (8 cores) @ 2.10GHz
    - Intel SGX simulation mode (Appendix B)
  - 40 UPMEM ranks with 2560 DPUs @ 350MHz
- **Prior work-based artificial cryptographic delay is inserted [8, 9]**
  - The calculated overhead of AES-capable DMA engine is 22.35%

[8] Shaizeen Aga and Satish Narayanasamy. 2017. InvisiMem: Smart Memory Defenses for Memory Bus Side Channel. In Proceedings of the 44th Annual International Symposium on Computer Architecture (Toronto, ON, Canada) (ISCA '17). Association for Computing Machinery, New York, NY, USA, 94–106.
[9] Kha Dinh Duy and Hojoon Lee. 2022. SE-PIM: In-Memory Acceleration of Data-Intensive Confidential Computing. IEEE Transactions on Cloud Computing (2022), 1–18.

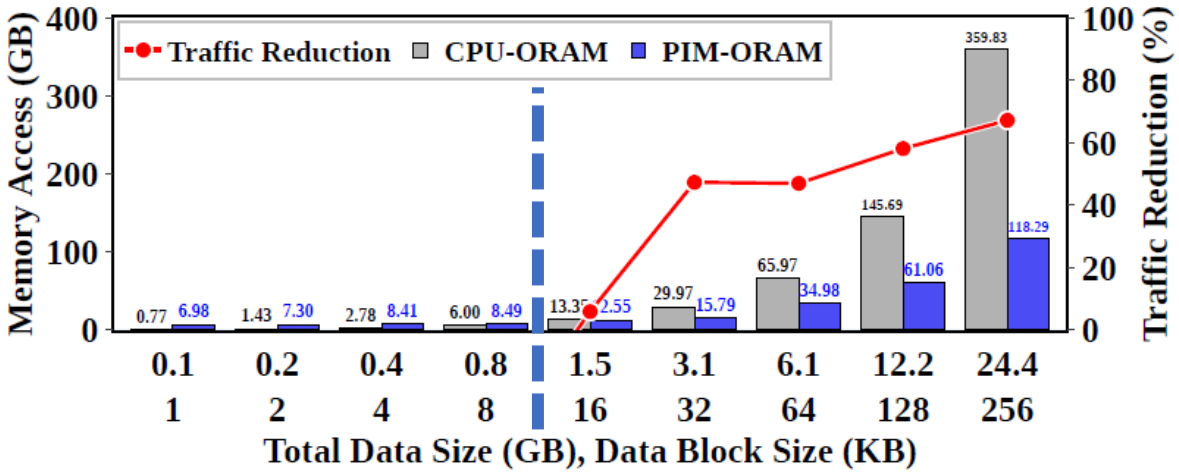
Evaluation Setup

- **CPU-ORAM: evaluation counterpart**
  - A traditional Path-ORAM implementation that only uses CPU's computation power
  - Uses Intel SGX to handle the same threat model
- **Tree Topology**
  - PIM-ORAM's allocated number of DPUs varies from 8 to 2048
  - CPU-ORAM's data block size varies from 1KB to 256KB to match with PIM-ORAM's topology

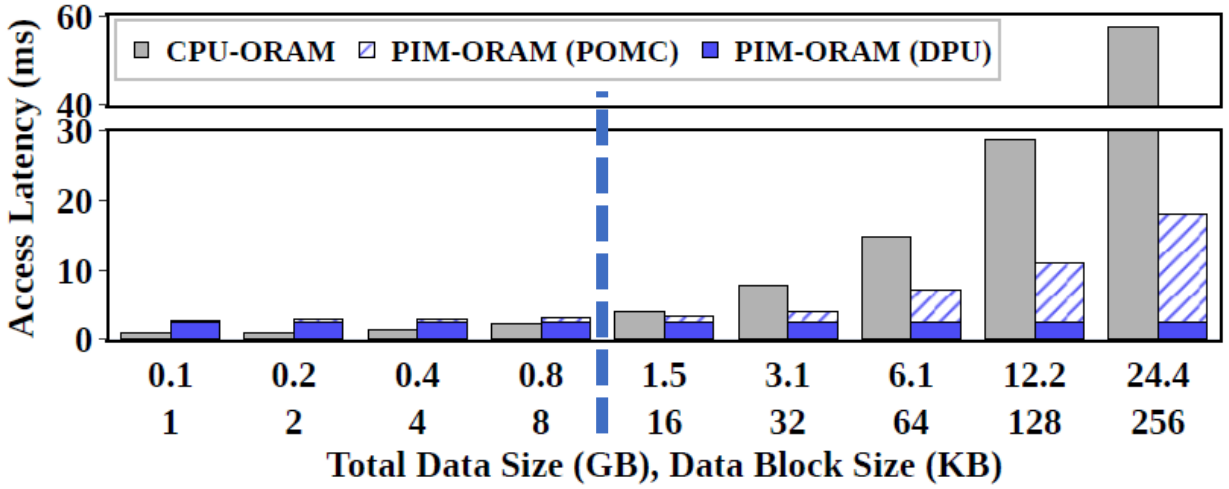
Shared Setting for CPU-ORAM & PIM-ORAM									
Data SZ (GB)	0.1	0.2	0.4	0.8	1.5	3.1	6.1	12.2	24.4
Block SZ (KB)	1	2	4	8	16	32	64	128	256
PIM-ORAM-Specific Parameters									
# DPUs Involved	8	16	32	64	128	256	512	1024	2048
# POMC Threads	1	1	2	2	4	4	8	8	8

Accumulated system bus traffic usage, Average access latency

- Measured during 10,000 ORAM accesses



Accumulated System Bus Traffic Usage

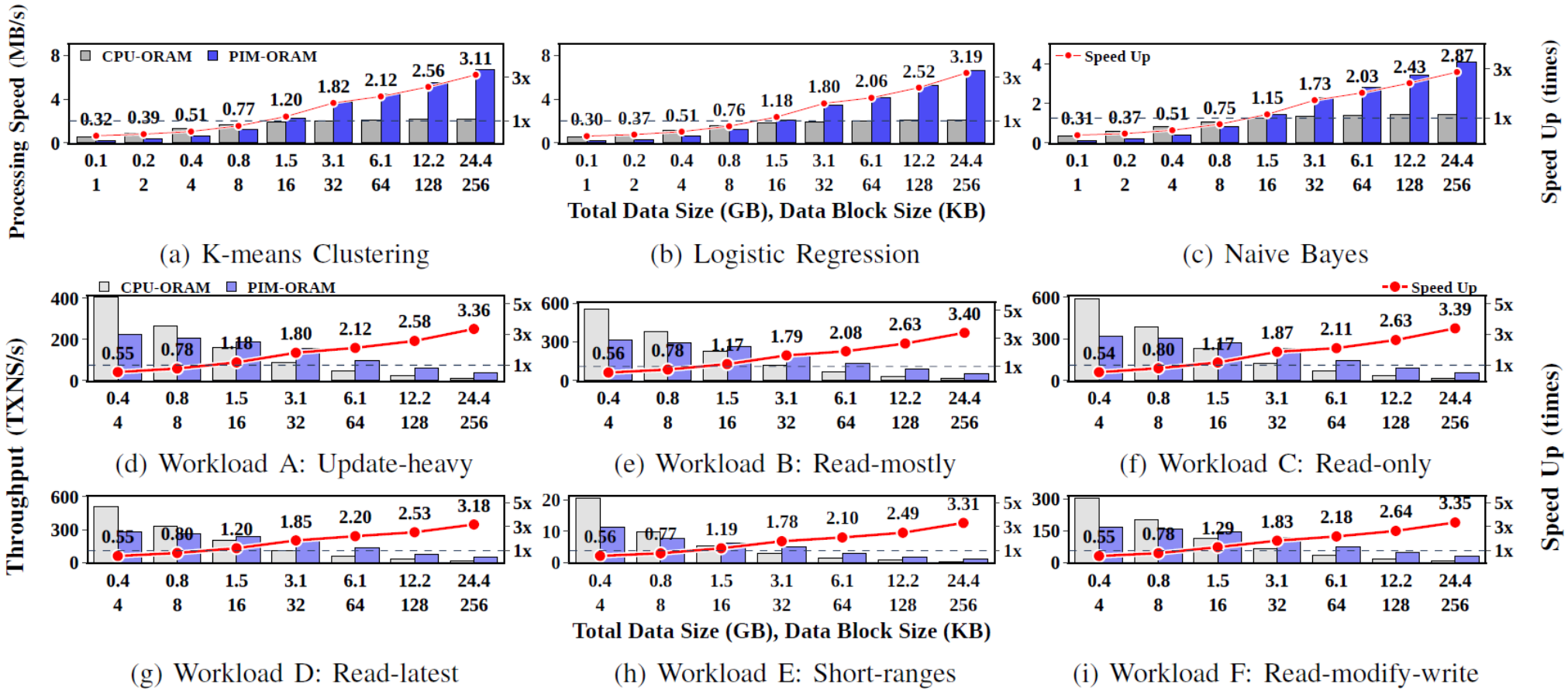


Average Access Latency

Real-world workload

Memory-intensive workloads

- Machine learning algorithms, Redis in-memory database



Conclusion

- Presented PIM-ORAM that accelerates ORAM primitives using a currently available commodity PIM device
- PIM-ORAM explores how the current hardware PIM's design might better support secure and oblivious computation
- PIM-ORAM introduces a design for adapting ORAM to the currently available PIM while retaining the original security guarantee of ORAM
- Even though PIM-ORAM shows a lower base performance than CPU-ORAM, PIM-ORAM scales better than CPU-ORAM
- PIM-ORAM can be applied to real-world applications such as K-means clustering and Redis

Please refer to our paper for more information

- Optimization, Security analysis, Etc.

KAIST CySecLab

SSLab

Q&A

Comparison between PIM-ORAM and ZeroTrace

- ZeroTrace: a representative SGX-based Path-ORAM implementation
- As ZeroTrace's Path-ORAM implementation differs from PIM-ORAM's implementation, we implement our CPU-ORAM that shares the same ORAM implementation with PIM-ORAM

Total Data Size (GB)	0.1	0.2	0.4	0.8	1.5	3.1	6.1	12.2	24.4
Data Block Size (KB)	1	2	4	8	16	32	64	128	256
ZeroTrace	7.99	14.60	27.81	54.34	107.27	213.40	**N/A**	**N/A**	**N/A**
PIM-ORAM	2.84	2.92	2.98	3.10	3.48	4.14	7.05	11.09	18.06
└*POMC*	0.43	0.51	0.57	0.69	1.07	1.73	4.64	8.68	15.65
└*DPU*	2.41	2.41	2.41	2.41	2.41	2.41	2.41	2.41	2.41

Comparison between SGX SIM mode and HW mode

- 10000 CPU-ORAM accesses on SGX simulation mode vs. on SGX hardware mode

Block Size (KB)	1	2	4	8	16	32	64	128	256
$TPUT_{SGXSim}$ / $TPUT_{SGXHW}$	2.43	2.21	1.96	1.62	1.36	1.20	1.11	1.10	1.04