34TH USENIX SECURITY SYMPOSIUM

Enabling Low-Cost Secure Computing on Untrusted In-Memory Architectures

Authors: **Sahar Ghoflsaz Ghinani**, Jingyao Zhang, Elaheh Sadredini

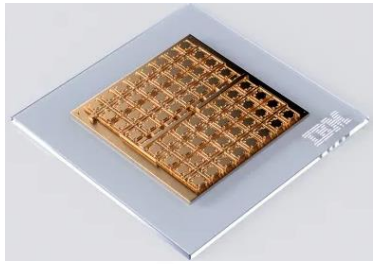
August, 2025

UC RIVERSIDE

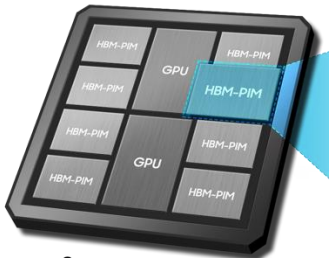
UCR Marlan and Rosemary Bourns College of Engineering

Real-World PIMs

- Machine Learning Accelerators



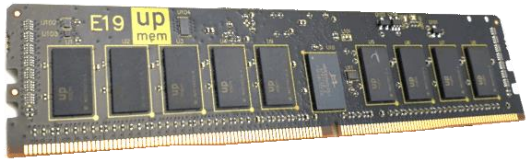
IBM HERMES



Samsung HBM–PIM

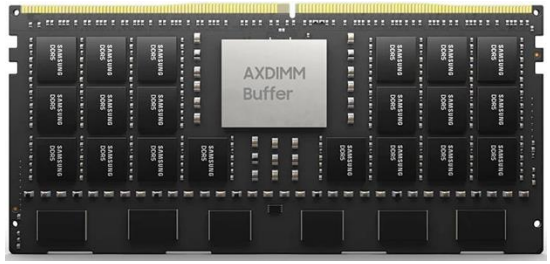
- Highly-Parallel Data Processing Engines

GSI Gemini APU



UPMEM PIM–DRAM

- Reconfigurable Hardware Substrates

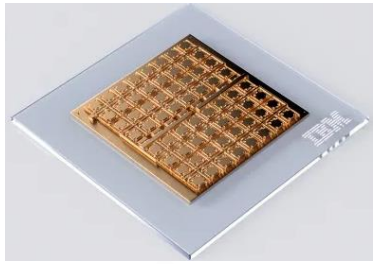


Samsung AxDIMM

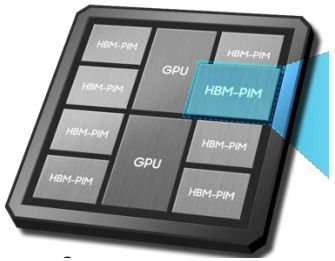
Samsung SmartSSD

Real-World PIMs

- Machine Learning Accelerators



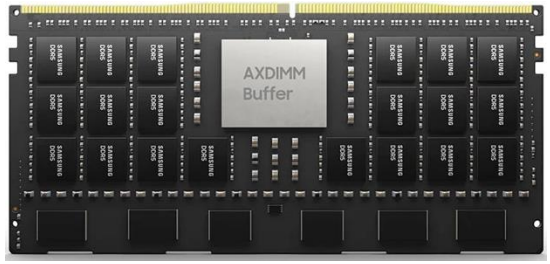
IBM HERMES



Samsung HBM–PIM

Security has become the first-class citizen

- Reconfigurable Hardware Substrates



Samsung AxDIMM

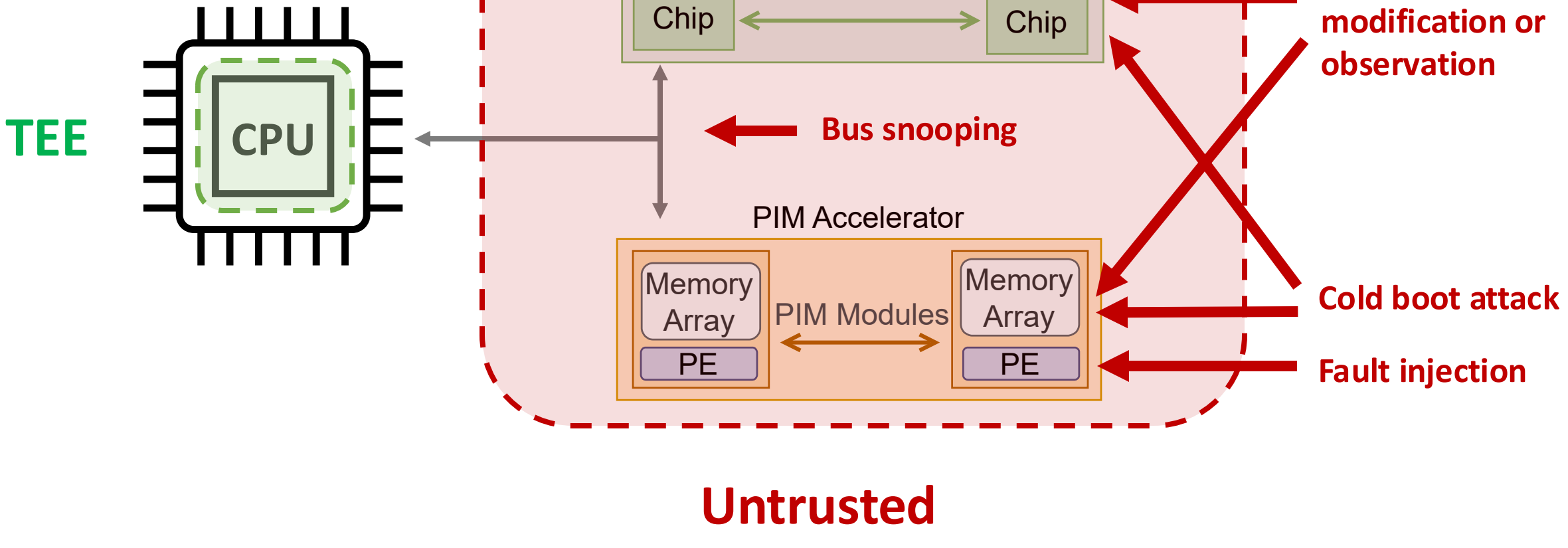
Samsung SmartSSD

TEE based Threat Model

Trusted Execution Environment

Intel SGX,

ARM Trust Zone, …

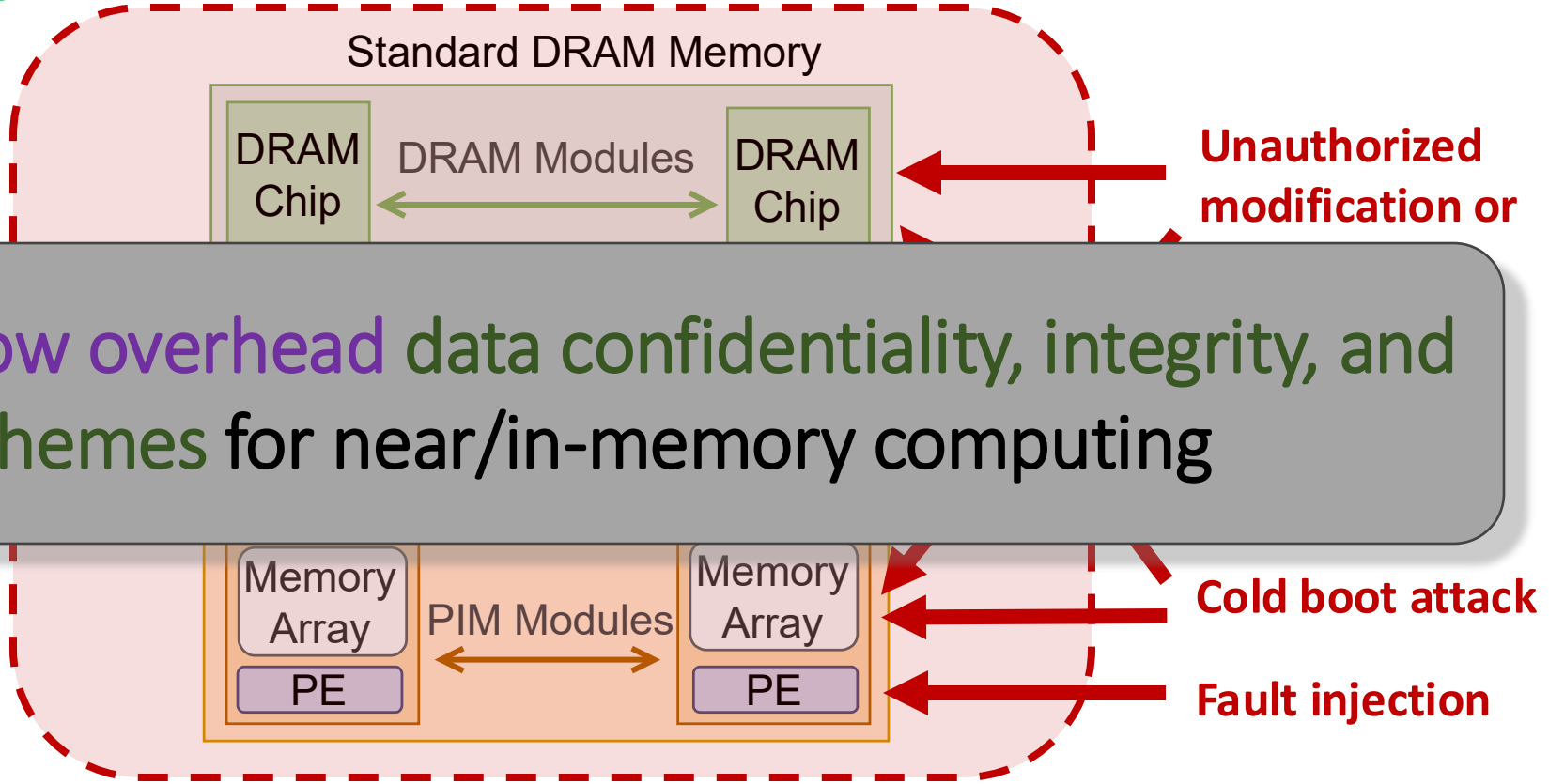


TEE based Threat Model

Trusted Execution Environment

Intel SGX,

ARM Trust Zone, …



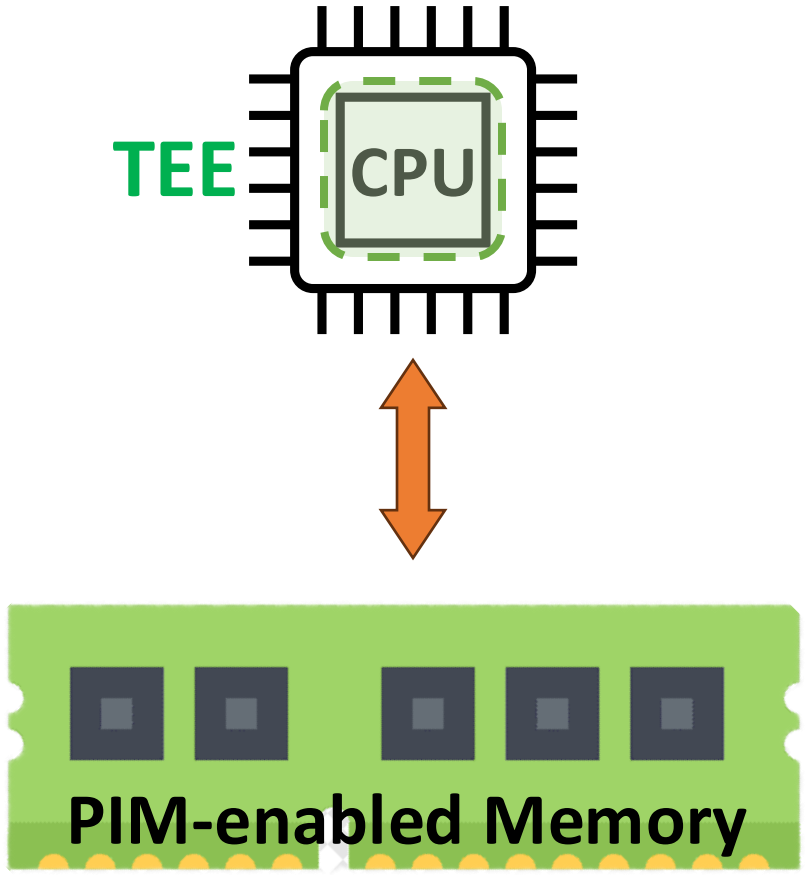
We need to provide low overhead data confidentiality, integrity, and verification schemes for near/in-memory computing

Untrusted

Data Confidentiality for PIM

Since we want to accelerate the computation using PIM, We can not use conventional memory encryption schemes.

- Homomorphic Encryption (HE)



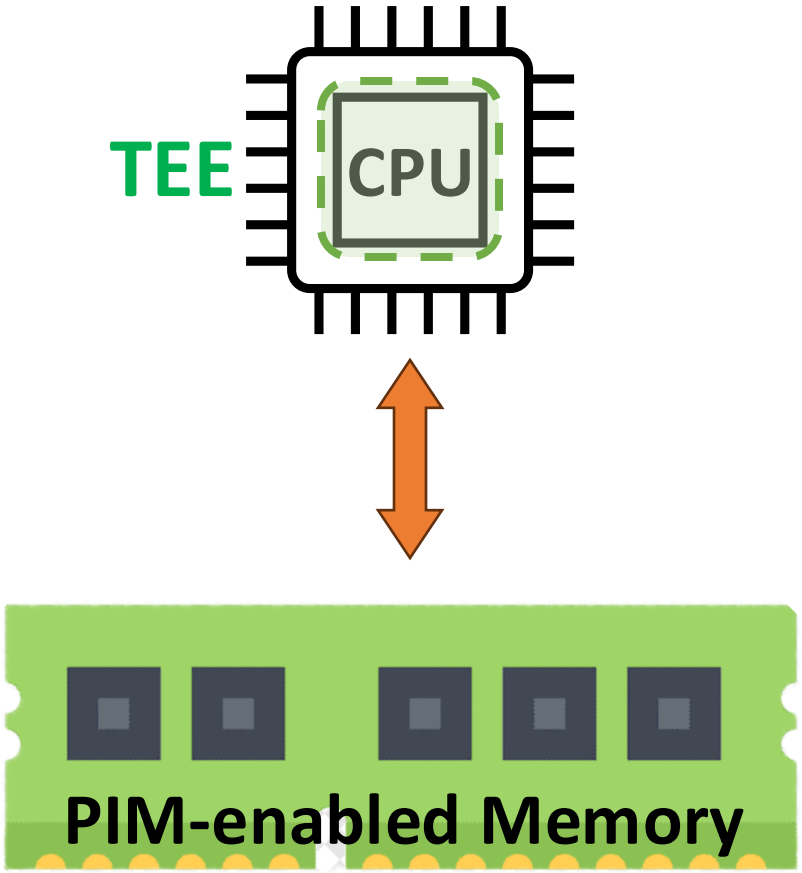
Not efficient!

- Significant overhead
- Cannot outperform the TEE computation.

Data Confidentiality for PIM

Since we want to accelerate the computation using PIM, We can not use conventional memory encryption schemes.

- Homomorphic Encryption (HE)
- Include PIM into the TEE



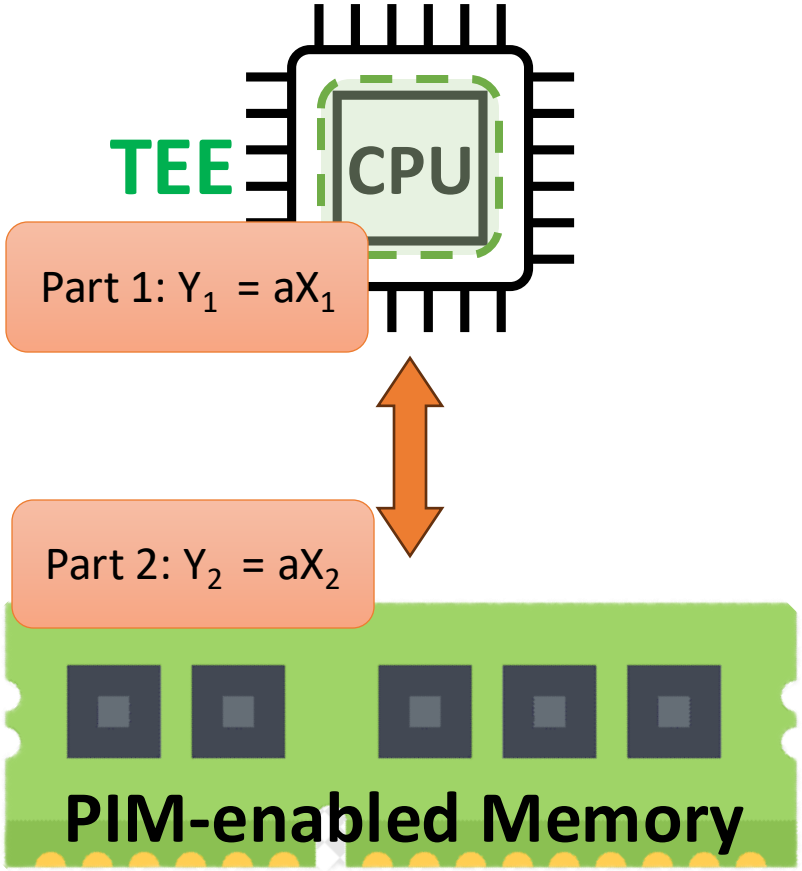
Significant overhead!
It requires to trust many components.

Data Confidentiality for PIM

Since we want to accelerate the computation using PIM, We can not use conventional memory encryption schemes.

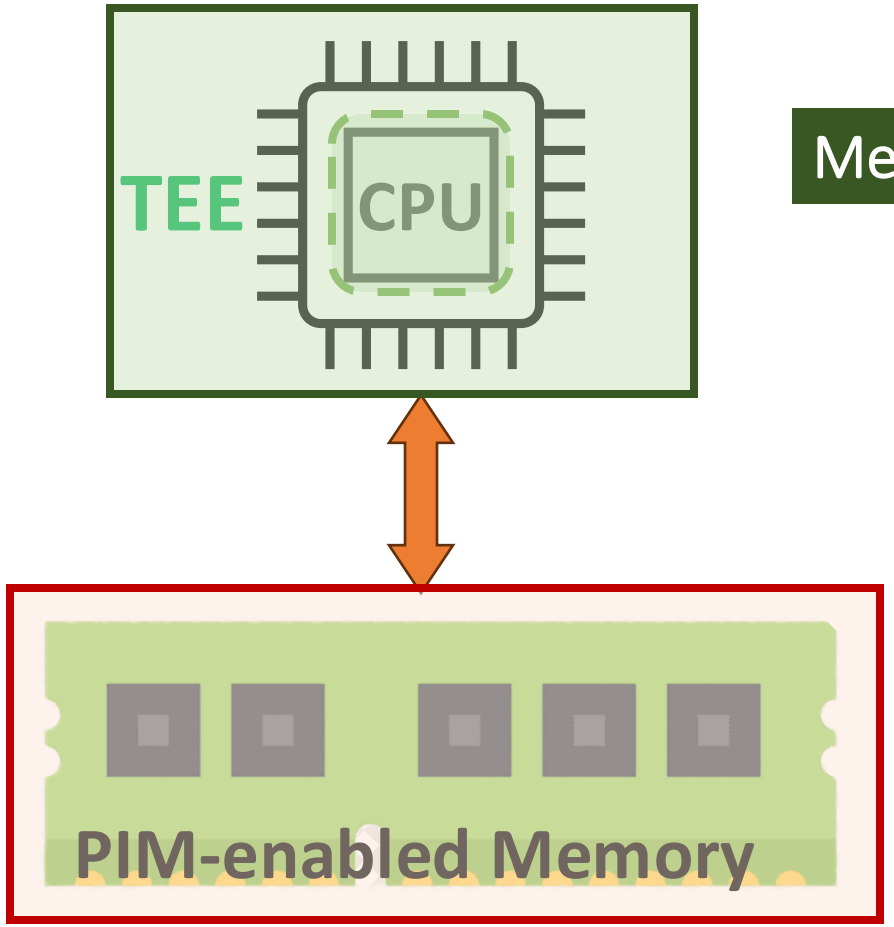
- Homomorphic Encryption (HE)
- Include PIM into the TEE
- Secure Multi Party Computing (MPC)

Partitioning the computation between PIM and TEE to balance the workload



Challenge: Workload Balancing

Compute-Intensive
Portion on **CPU** and
Memory-Intensive
Portion on **PIM** Unit



Memory Bandwidth Limit

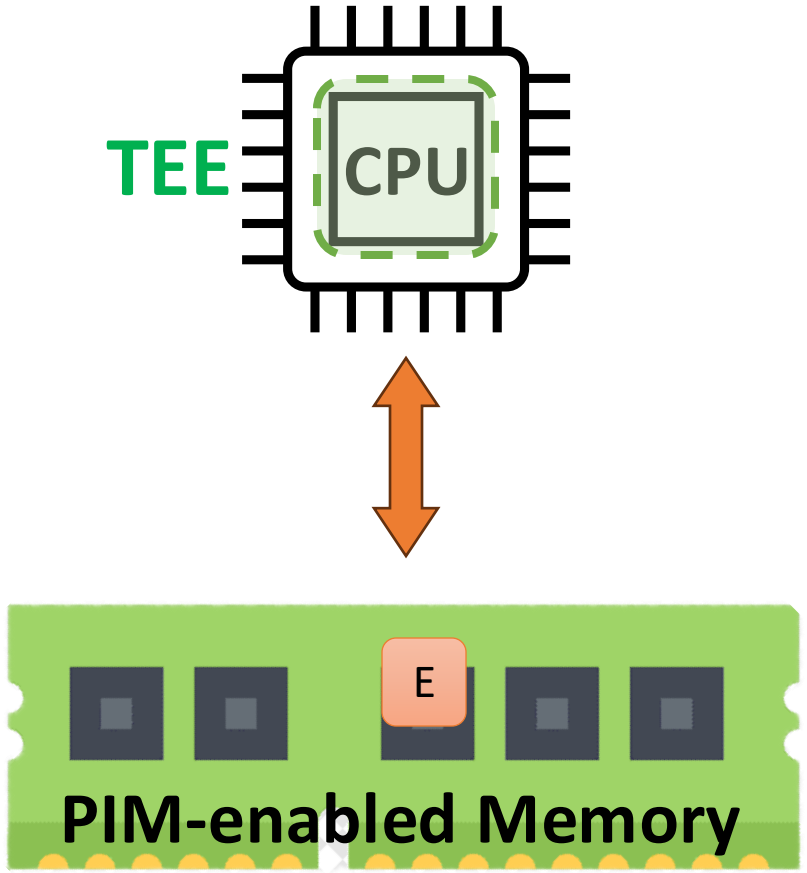
Compute Limit

Prior Work -- SecNDP (Xiong et al., HPCA '23)

- Outsourcing Linear functions to an untrusted PIM
- OTPs are generated on the fly

Precomputation

E = X – R
R = Encrypt(counter)

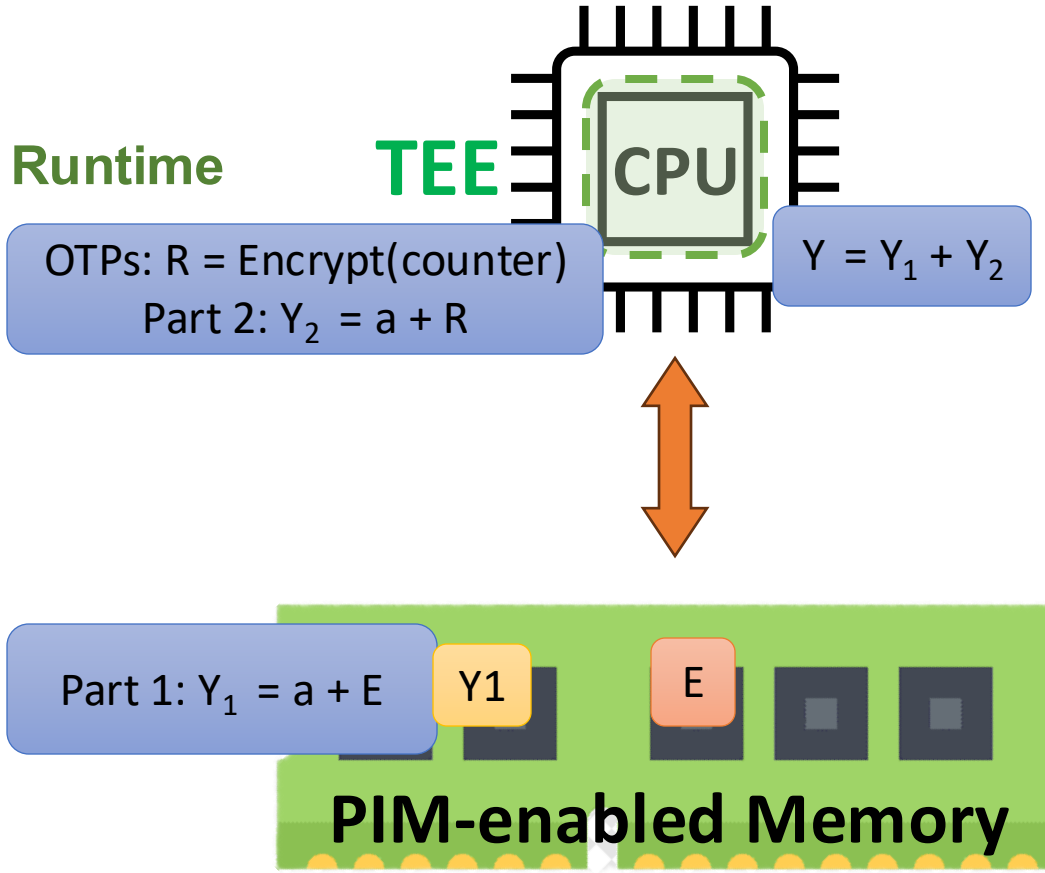


Prior Work -- SecNDP (Xiong et al., HPCA '23)

- Outsourcing Linear functions to an untrusted PIM
- OTPs are generated on the fly

Precomputation

E = X – R
R = Encrypt(counter)



Prior Work -- SecNDP (Xiong et al., HPCA '23)

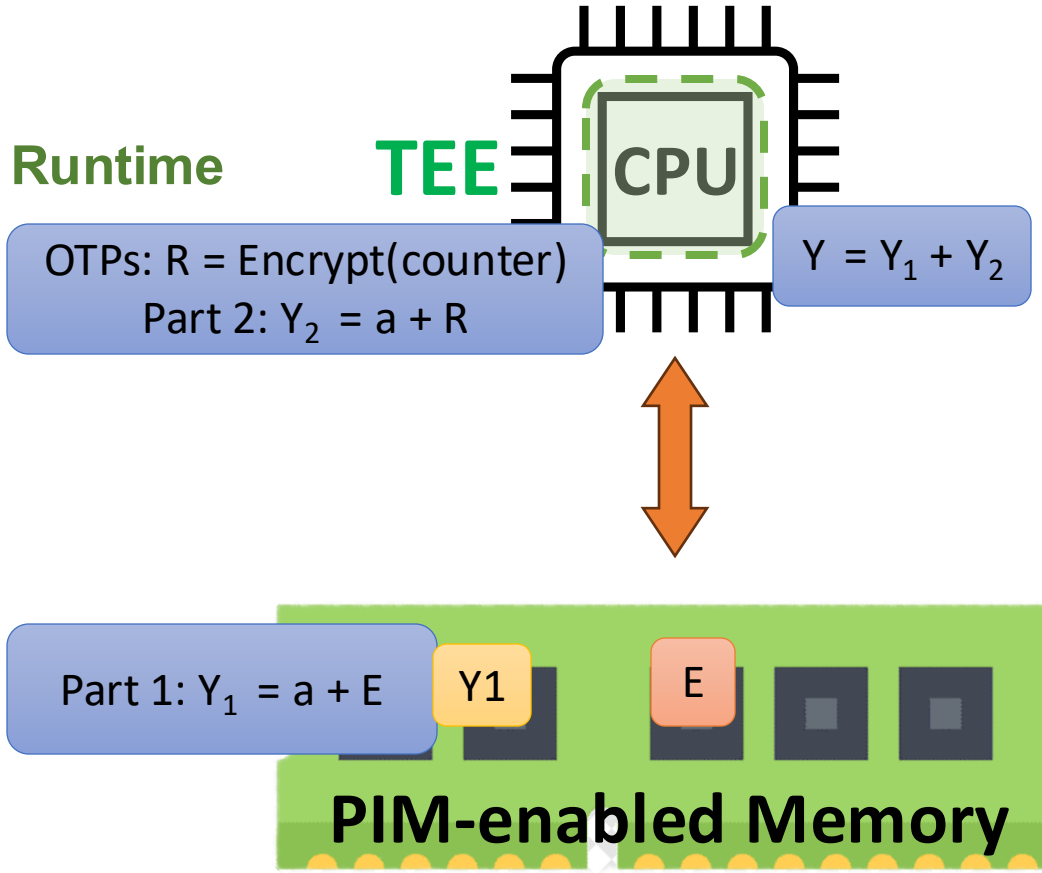
- Outsourcing Linear functions to an untrusted PIM
- OTPs are generated on the fly

Precomputation

$E = X - R$
$R = Encrypt(counter)$

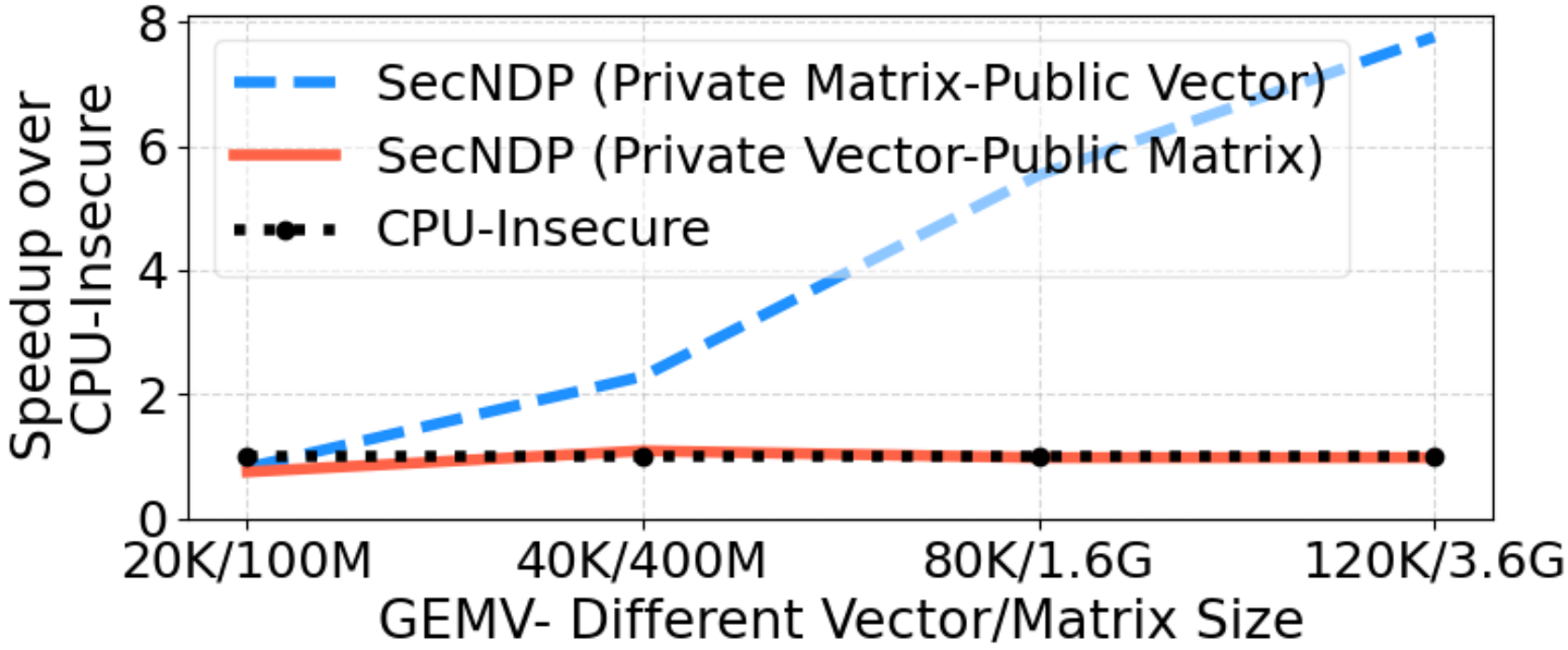
Challenges:

- Computation bottleneck when public data is larger than private data
- No support for non-linear operation
- Evaluation is based on simulation not real hardware



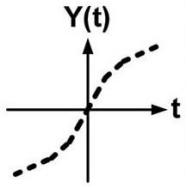
Observation: Public Data Transfer Can become bottleneck

- User data is private
- Non-user data is public



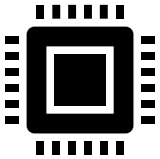
Our Proposed Method for Secure PIM

Compute-Intensive Portion on CPU and Bandwidth-Intensive Portion on PIM Unit + Precomputation



Supports both linear and non-linear functions

By switching between arithmetic secret sharing and Yao's sharing



Implemented on real-world hardware

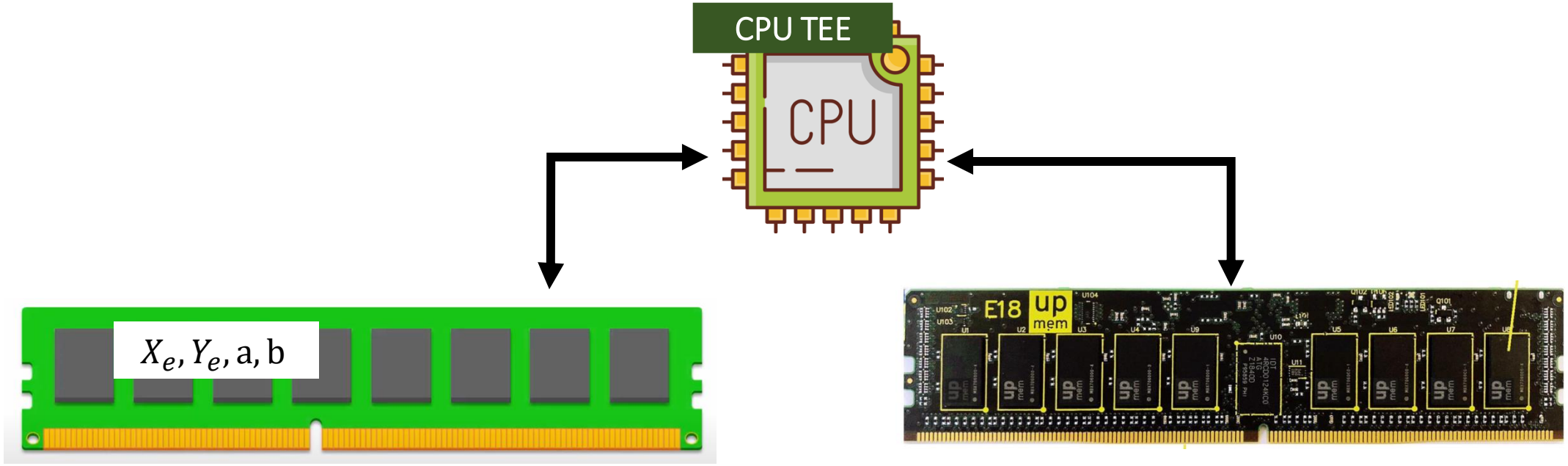
UPMEM, the first publicly available PIM.

- Our proposed method enables low overhead data confidentiality, integrity, and verification schemes for both linear and non-linear operations for real-world PIMs

High-Level Security Scheme: Linear Operations

Data Encryption

$$Z = aX + bY \qquad X \text{ and } Y \text{ are secrets}$$



High-Level Security Scheme: Linear Operations

Data Encryption

Generate Plaintext

$$X_P = Decrypt\ (X_e)$$
$$Y_P = Decrypt\ (Y_e)$$

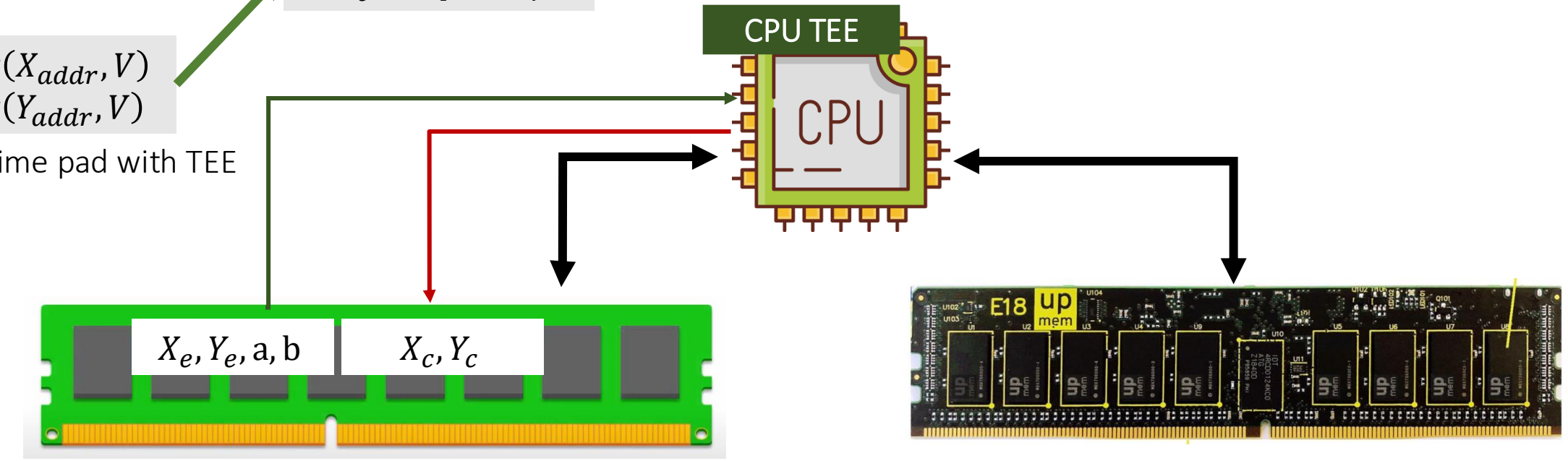
Ciphertext

$$X_c = X_P - X_r$$
$$Y_c = Y_P - Y_r$$

$$X_r = Encr(X_{addr}, V)$$
$$Y_r = Encr(Y_{addr}, V)$$

Generate one-time pad with TEE

$$Z = aX + bY \qquad X \text{ and } Y \text{ are secrets}$$



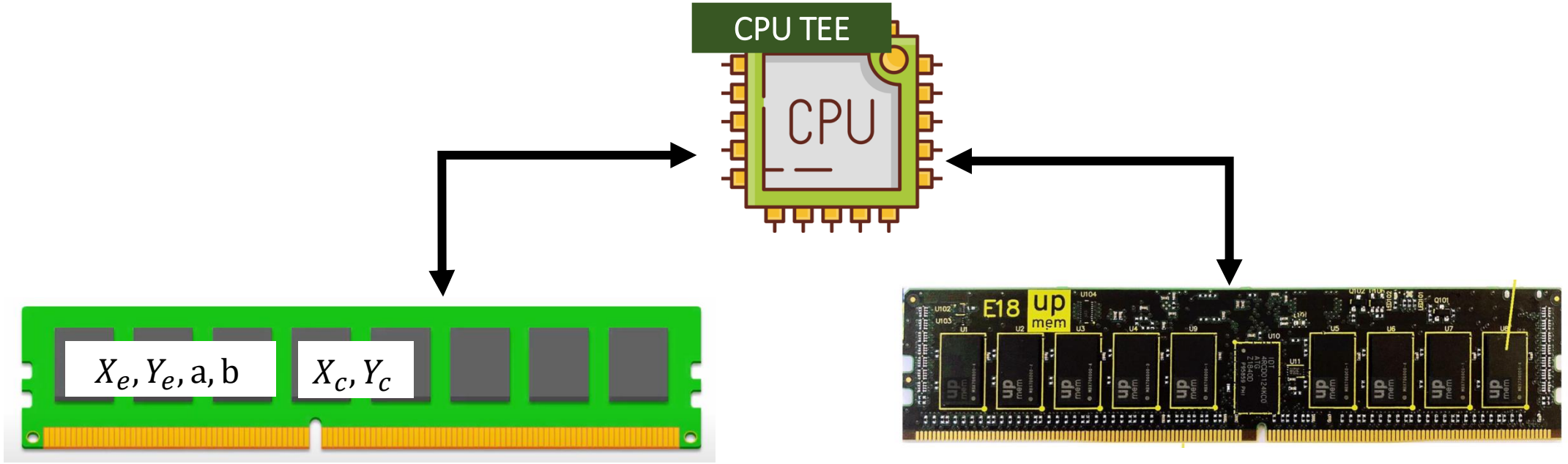
High-Level Security Scheme: Linear Operations

Data Encryption → **Pre-computation**

$$X_r = Encr(X_{addr}, V)$$
$$Y_r = Encr(Y_{addr}, V)$$

→ $Z_r = \mathrm{a}X_r + \mathrm{b}Y_r$

Generate one-time pad with TEE



High-Level Security Scheme: Linear Operations

Data Encryption → **Pre-computation**

$$X_r = Encr(X_{addr}, V)$$
$$Y_r = Encr(Y_{addr}, V)$$

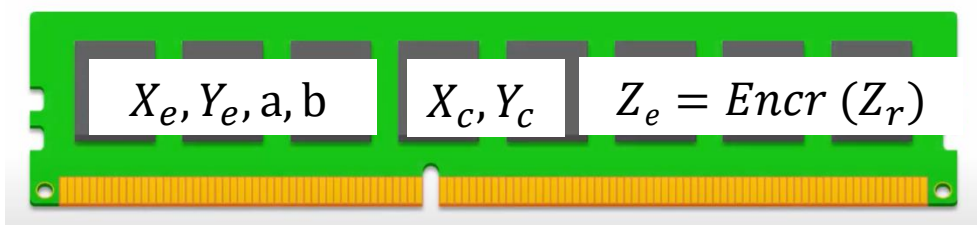
Generate one-time pad with TEE

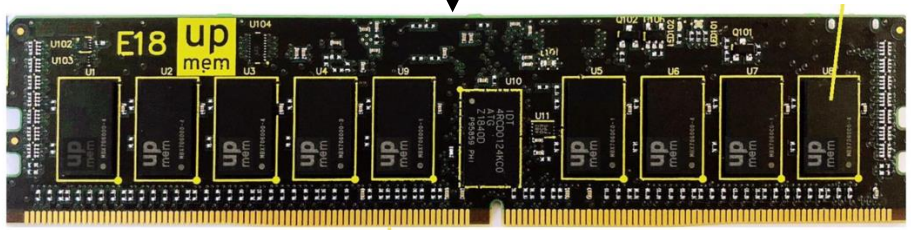
$$Z_r = \mathrm{a}X_r + \mathrm{b}Y_r$$

$$Z_e = Encr\ (Z_r)$$

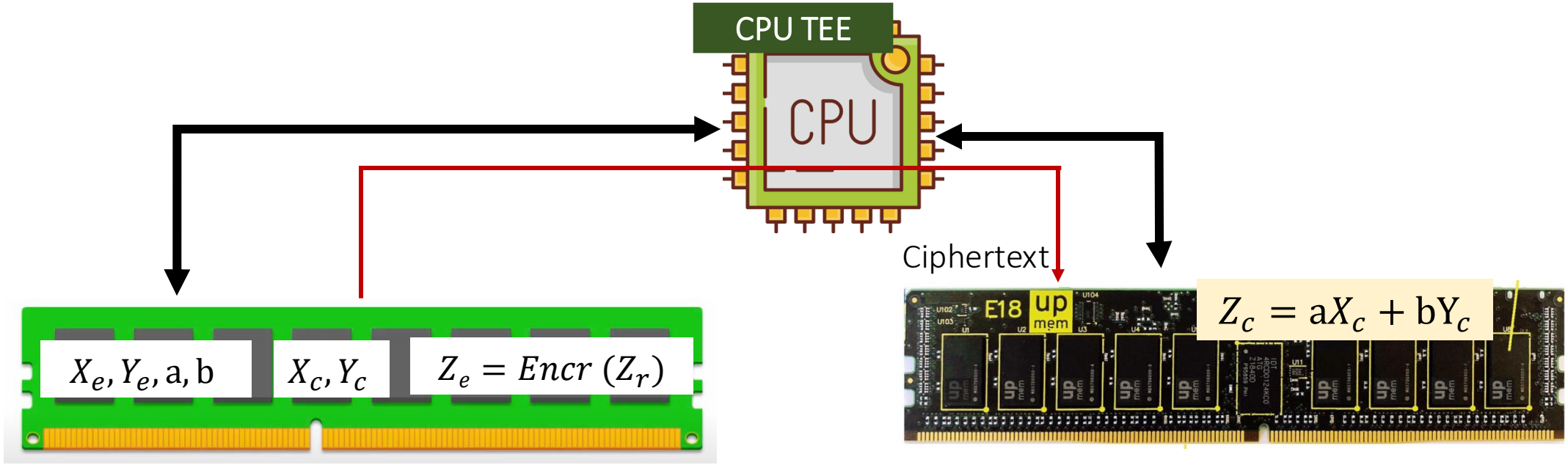
CPU TEE

CPU





High-Level Security Scheme: Linear Operations

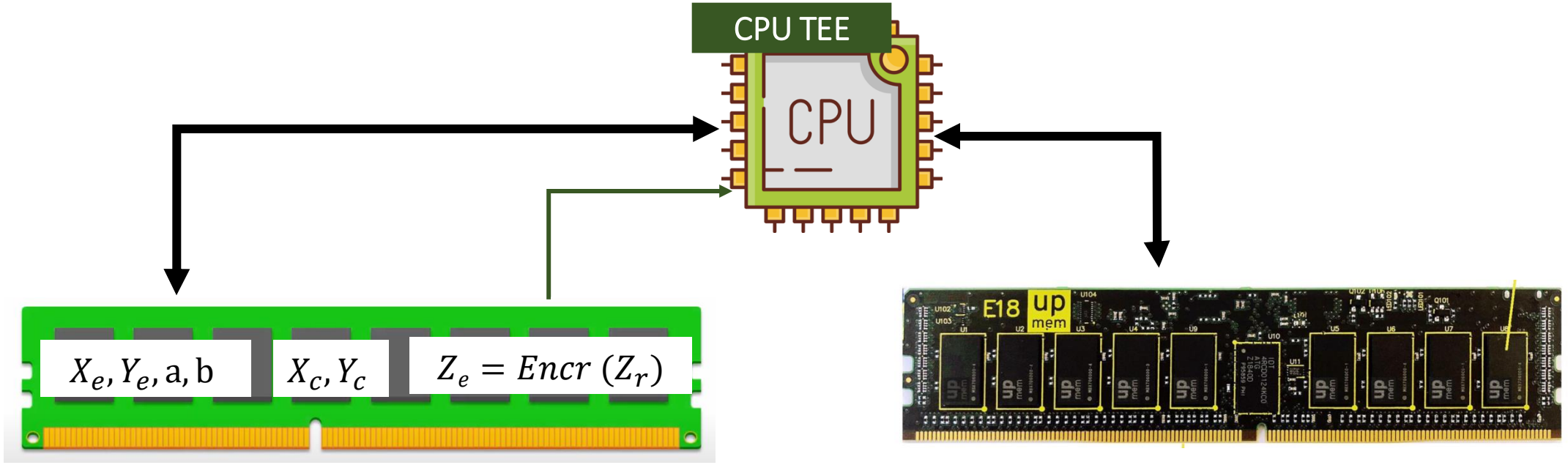


High-Level Security Scheme: Linear Operations

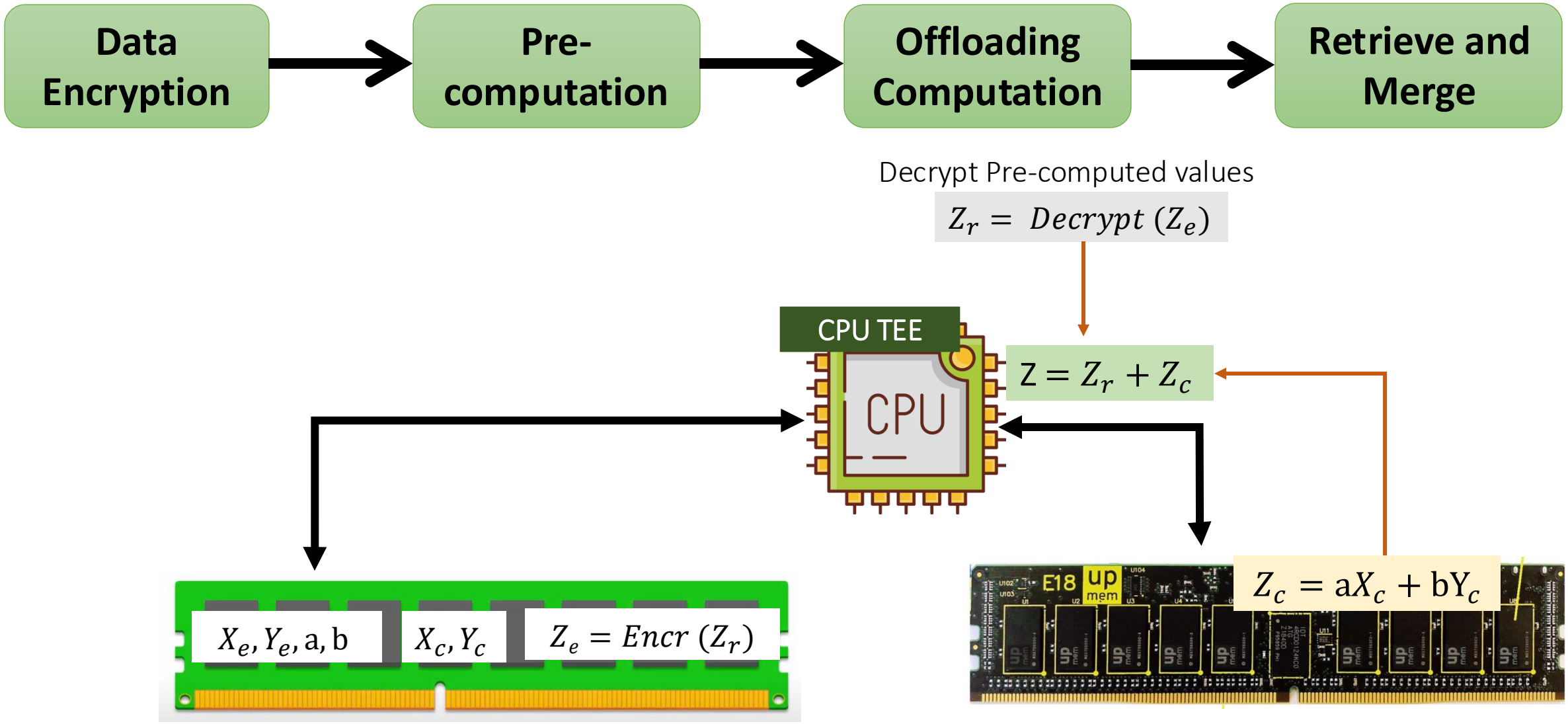
Data Encryption → **Pre-computation** → **Offloading Computation**

Decrypt Pre-computed values

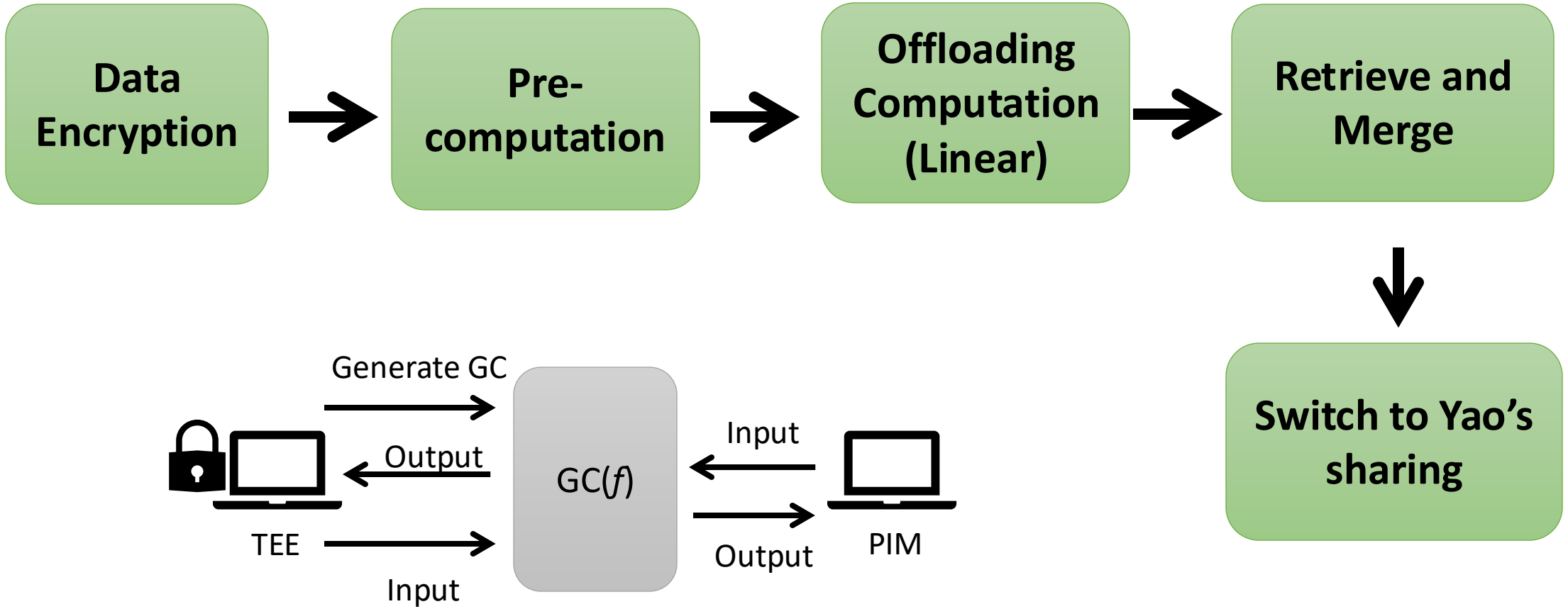
$$Z_r = Decrypt\ (Z_e)$$



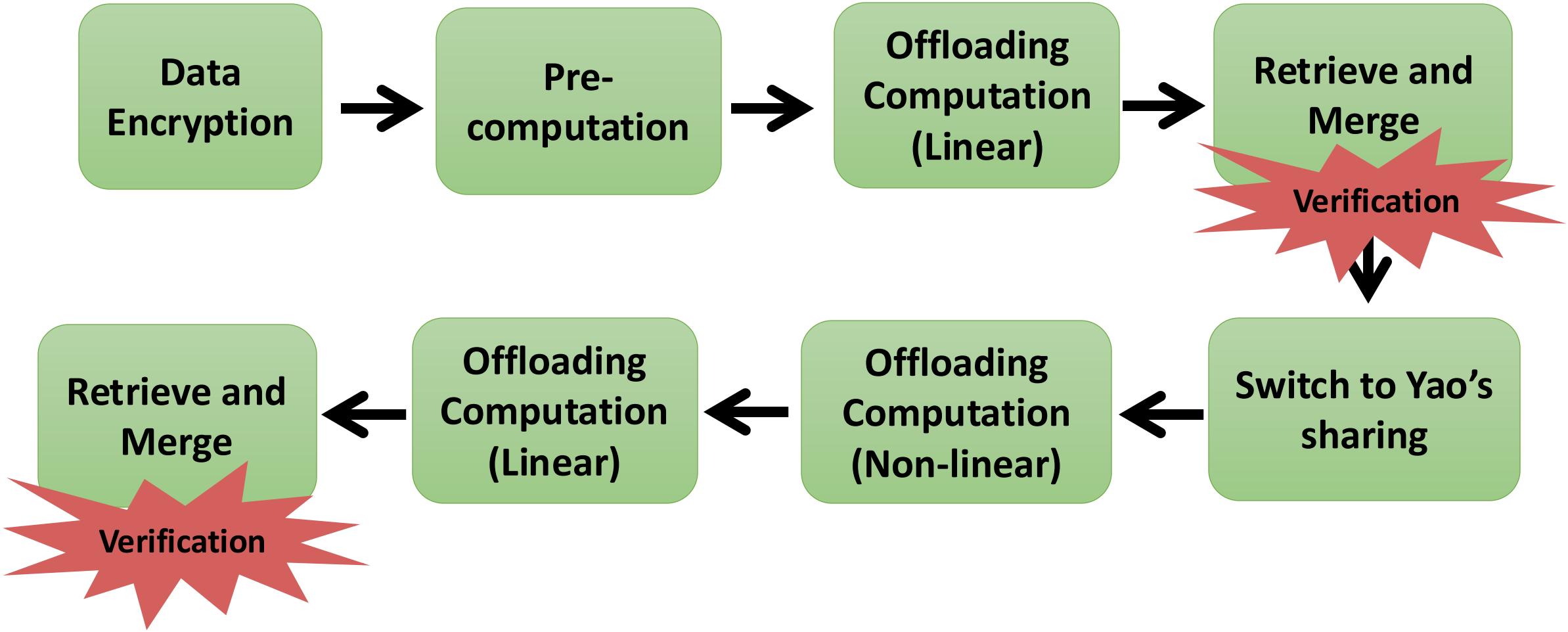
High-Level Security Scheme: Linear Operations



High-Level Security Scheme: Non-linear Operations

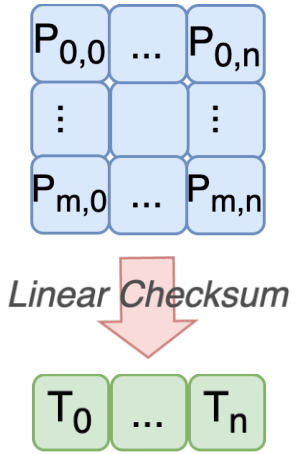


High-Level Security Scheme: Non-linear Operations



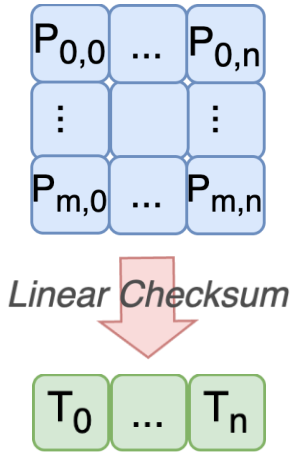
Verification Scheme for secure PIM

- Message Authentication Codes (MACs)
- Linear checksum to generate tags



Verification Scheme for secure PIM

- Message Authentication Codes (MACs)
- Linear checksum to generate tags
- MAC-then-encrypt



$C = Encr(T_{addr}, V)$

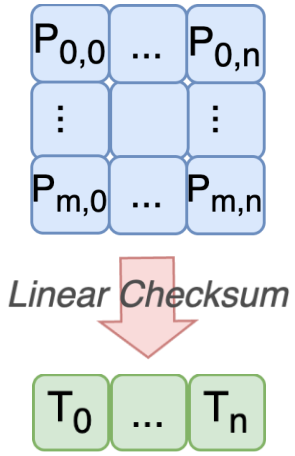
Generate one-time pad with TEE

Ciphertext

$CT = T - C$

Verification Scheme for secure PIM

- Message Authentication Codes (MACs)
- Linear checksum to generate tags
- MAC-then-encrypt
- Treat them as data to verify the computation.



Ciphertext

$C = Encr(T_{addr}, V)$ → $CT = T - C$

Generate one-time pad with TEE

TEE $Tag_1 = C * a$

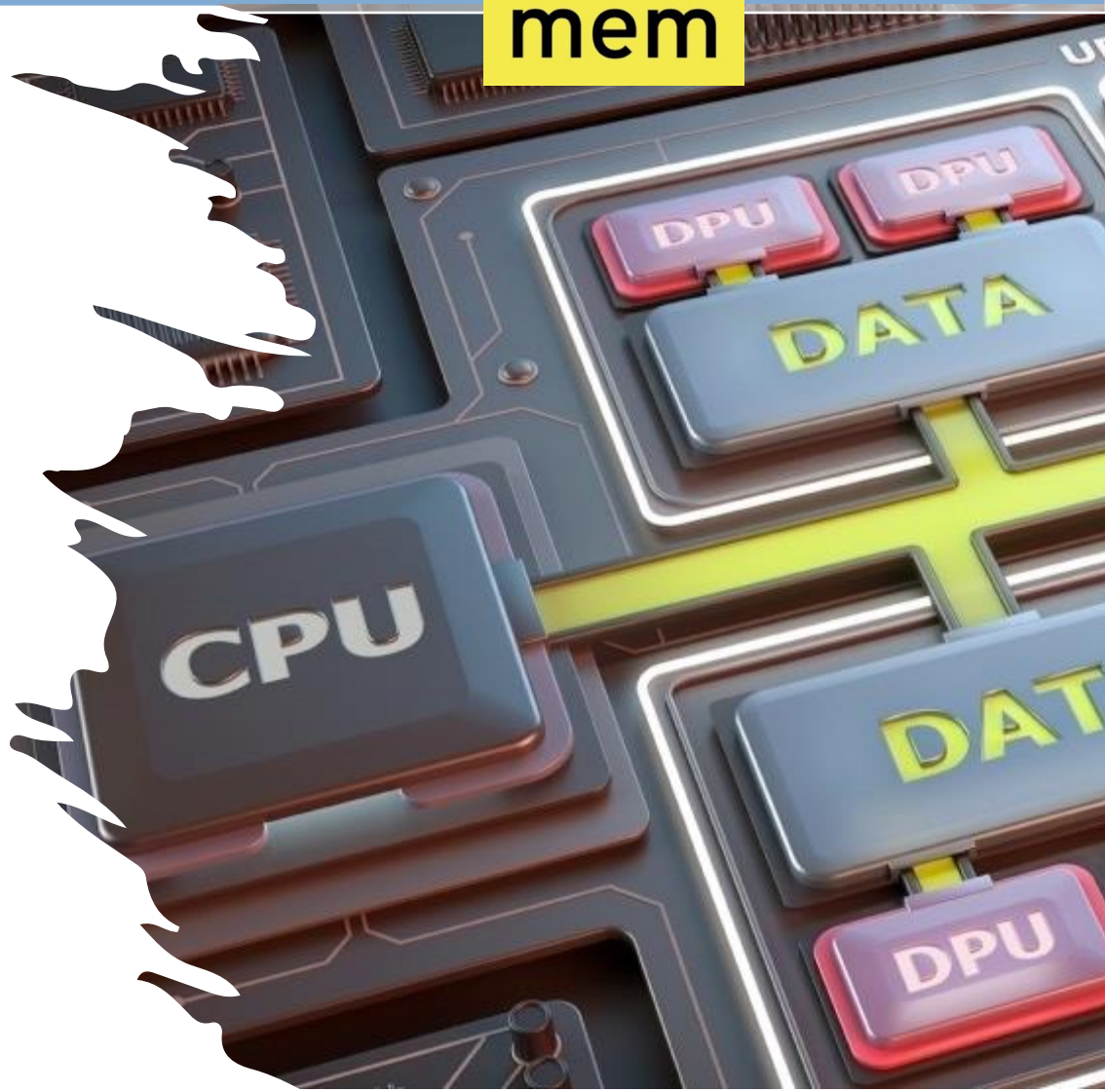
PIM $Tag_2 = CT * a$

$Tag = Tag_1 + Tag_2$

Hardware Configuration

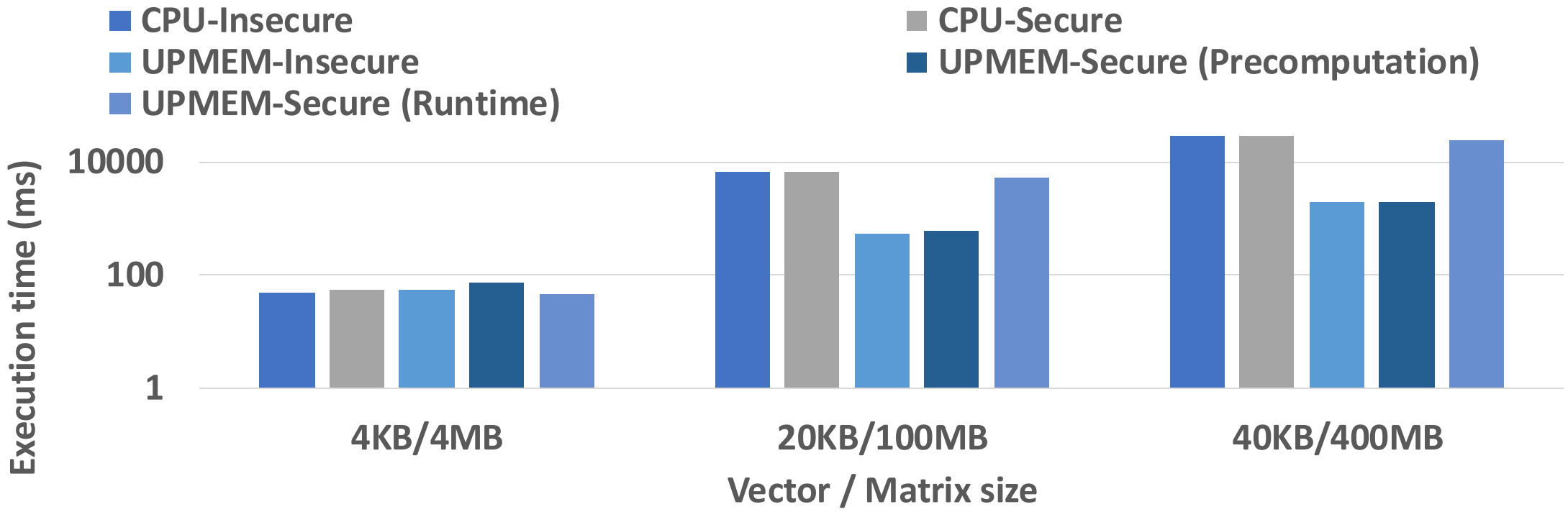
- 20 UPMEM PIM hardware
  - DDR4-2400 integrated with DPUs
  - 2,560 DPUs
  - 160 GB of memory
- Host CPU:
  - 2-socket Intel Xeon Silver4110

- Applications
  - Multilayer Perceptron (MLP)
  - Deep Learning Recommendation Models (DLRM)
  - Logistic Regression
  - Linear Regression



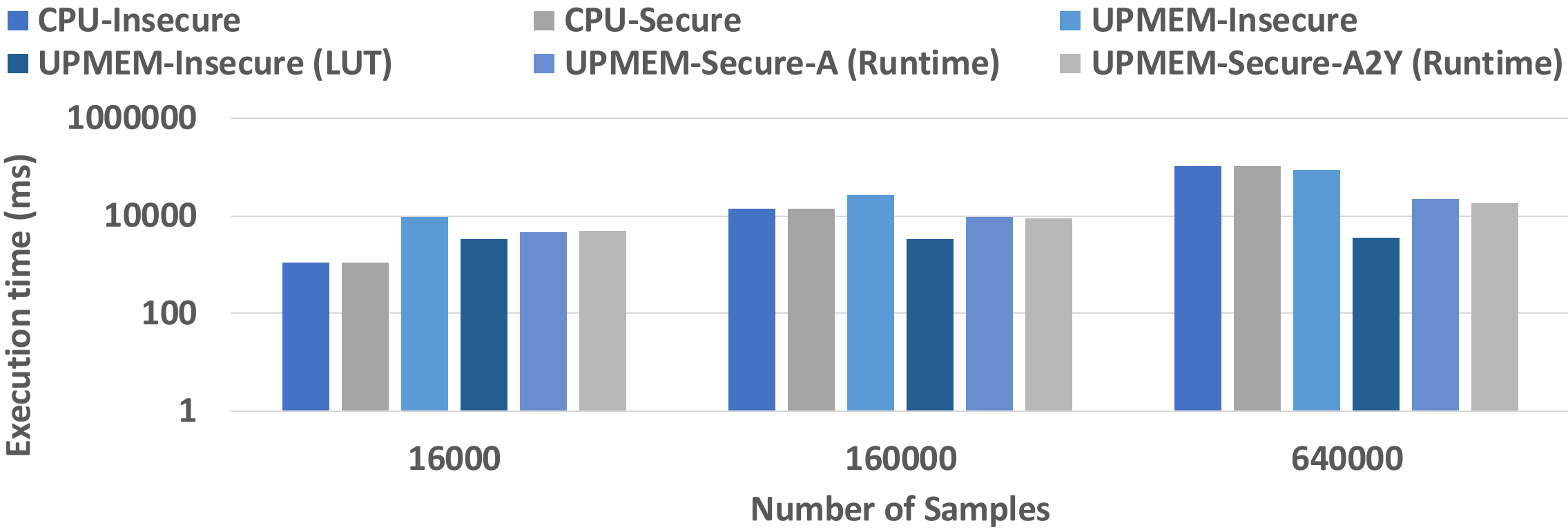
Performance Evaluation -- MLP

- UPMEM-Secure achieves up to 15X speedup over CPU-Secure
- UPMEM-Secure demonstrated minimal performance overhead (4%) over UPMEM-Insecure
- Verification overhead is negligible



Performance Evaluation -- Logistic Regression Training

- UPMEM-Secure achieves 5× speedup compared to CPU-Secure
- UPMEM-Secure experiences a 6× slowdown compared to UPMEM-Insecure (LUT-Based)
- UPMEM-Secure achieves a 4× speedup compared to UPMEM-Insecure (non-LUT-Based)



Key Takeaways

- ✓ **MPC + Precomputation** provides a workload balance between TEE and PIM to **accelerate** computation **up to 15x** while **preserving data privacy and integrity**
- ✓ **By switching** between different MPC schemes, we support secure computation of **both linear and nonlinear operations**
- ✓ **Using linear checksum**, we enable **low-overhead verification**

Our scheme enables low-overhead secure computing on real-world PIMs

Thank you!