

Efficient security for memory: A hardware perspective

Joydeep Rakshit*
PARL, Intel Labs

Contact: joydeep.rakshit@intel.com

*Collaborators: Shivam Swami (Micron) and Kartik Mohanram (University of Pittsburgh)



1st Workshop on MicroArchitectural Security
IIT Madras **October 11, 2019**

Agenda

- ❑ Memory security
 - ❑ The 3 basic pillars
 - ❑ Trusted Computing Base
- ❑ Data confidentiality
- ❑ Data integrity
- ❑ Access pattern obfuscation
 - ❑ Oblivious RAM
- ❑ Our work

Agenda

- ❑ Memory security
 - ❑ The 3 basic pillars
 - ❑ Trusted Computing Base
- ❑ Data confidentiality
- ❑ Data integrity
- ❑ Access pattern obfuscation
 - ❑ Oblivious RAM
- ❑ Our work

Memory Security

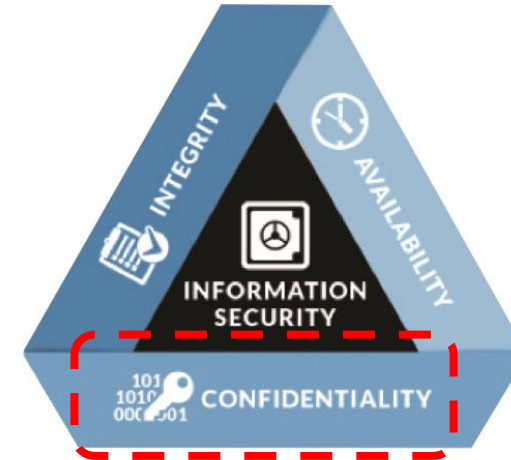
- ❑ Cornerstones of secure platform: **CIA** [1]
 - ❑ Confidentiality
 - ❑ Integrity
 - ❑ Availability



Credit: http://www.cybersafesolutions.com/wp-content/uploads/2016/08/CSS_ThreatPolicies_CIAgraphic.jpg

Memory Security

- ❑ Cornerstones of secure platform
 - ❑ **Confidentiality**
 - ❑ Plaintext data should not be visible to adversary
 - ❑ Integrity
 - ❑ Availability

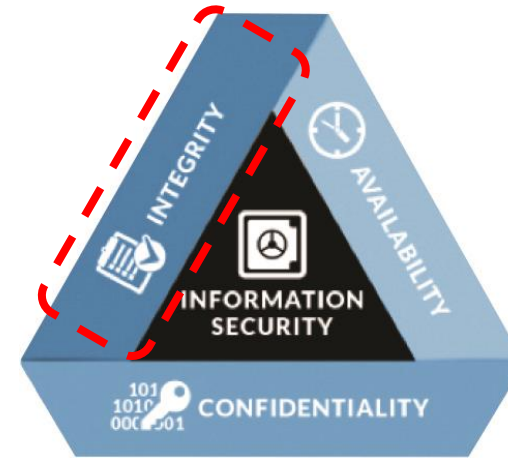


Credit: http://www.cybersafesolutions.com/wp-content/uploads/2016/08/CSS_ThreatPolicies_CIAgraphic.jpg

- [1] J. Cong *et al.*, “Improving privacy and lifetime of PCM-based main memory,” DSN, 2010
- [2] S. Chhabra and Y. Solihin, “i-NVMM: A secure non-volatile main memory system with incremental encryption,” ISCA, 2011
- [3] V. Young *et al.*, “DEUCE: Write-efficient encryption for non-volatile memories,” ASPLOS, 2015
- [4] A. Awad *et al.*, “Silent Shredder: Zero-cost shredding for secure non-volatile main memory controllers”, ASPLOS 2016
- [5] S. Swami *et al.*, “SECRET: Smartly EnCRypted energy EfficienT non-volatile memories”, DAC, 2016

Memory Security

- ❑ Cornerstones of secure platform
 - ❑ Confidentiality
 - ❑ **Integrity**
 - ❑ Adversary cannot perform *undetected* data tampering



Credit: http://www.cybersafesolutions.com/wp-content/uploads/2016/08/CSS_ThreatPolicies_CIAgraphic.jpg

Memory Security

- ❑ Cornerstones of secure platform
 - ❑ Confidentiality
 - ❑ Integrity
 - ❑ **Availability**
 - ❑ System (in this case, memory) should be able to serve requests without stalling.



Credit: http://www.cybersafesolutions.com/wp-content/uploads/2016/08/CSS_ThreatPolicies_CIAgraphic.jpg

[1] M. Qureshi *et al.*, “Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling”, MICRO, 2009

[2] N.H. Seong *et al.*, “Security Refresh: Prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping”, ISCA, 2010

[3] F. Huang *et al.*, “Security RBSG: Protecting phase change memory with security-level adjustable dynamic mapping”, PDPS, 2016.

Memory Security

- Cornerstones of secure platform

 - Confidentiality

 - Integrity

 - Availability

- Threat model

 - Trusted Computing Base (TCB)

Memory Security

- ❑ Cornerstones of secure platform

- ❑ Confidentiality
- ❑ Integrity
- ❑ Availability

- ❑ Threat model

- ❑ Trusted Computing Base (TCB) [1-4]

- ❑ Processor chip: Processor core, registers, caches, etc...
- ❑ Critical parts of OS



Secure



[1] R. B. Lee, "Security basics for computer architects," *Synthesis Lectures on Computer Architecture*, 2013

[2] G. E. Suh *et al.*, "Efficient memory integrity verification and encryption for secure processors," MICRO, 2003

[3] B. Rogers *et al.*, "Using address independent seed encryption and Bonsai Merkle Trees to make secure processors OS-and performance-friendly", MICRO, 2007

[4] A. D. Hilton *et al.*, "PoisonIvy: Safe speculation for secure memory," in MICRO, 2016

Memory Security

❑ Cornerstones of secure platform

- ❑ Confidentiality
- ❑ Integrity
- ❑ Availability

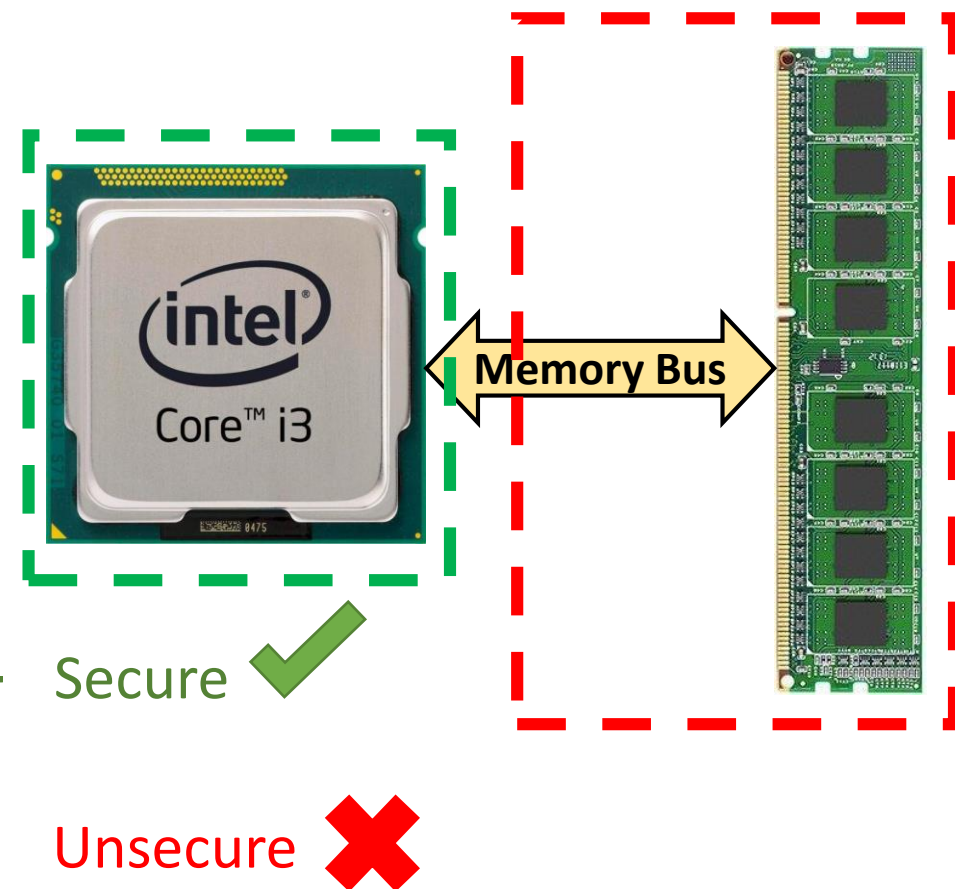
❑ Threat model

❑ Trusted Computing Base (TCB) [1-4]

- ❑ Processor chip: Processor core, registers, caches, etc...
- ❑ Critical parts of OS

❑ Untrusted components

- ❑ Off-chip resources: Memory, buses, etc.



[1] R. B. Lee, "Security basics for computer architects," *Synthesis Lectures on Computer Architecture*, 2013

[2] G. E. Suh *et al.*, "Efficient memory integrity verification and encryption for secure processors," *MICRO*, 2003

[3] B. Rogers *et al.*, "Using address independent seed encryption and Bonsai Merkle Trees to make secure processors OS-and performance-friendly", *MICRO*, 2007

[4] A. D. Hilton *et al.*, "PoisonIvy: Safe speculation for secure memory," in *MICRO*, 2016

Agenda

- Memory security
 - The 3 basic pillars
 - Trusted Computing Base
- Data confidentiality**
- Data integrity
- Access pattern obfuscation
 - Oblivious RAM
- Our work

Data Confidentiality

- ❑ Prevention of unauthorized leakage of plaintext data to adversary
- ❑ Attack vectors
 - ❑ Bus snooping [1,2]
 - ❑ Direct memory access (DMA)/Memory dump and scan [3]
 - ❑ Cold boot attacks [4]

- ❑ **Solution:** Encryption of data stored in memory

[1] A. Huang, "Hacking the Xbox: An Introduction to Reverse Engineering" No Starch Press, 2003

[2] A. B. Huang, "The Trusted PC: Skin-Deep Security", IEEE Computer, 2002

[3] A. Kumar, "Discovering Passwords in Memory", http://www.infosecwriters.com/text_resources/, 2004

[4] J. A. Halderman, "Lest We Remember: Cold Boot Attacks on Encryption Keys", USENIX Security Symposium, 2008

Data Confidentiality

- ❑ Direct data encryption
 - ❑ One-way function (cipher like AES) + Global key
 - ❑ Problem: Dictionary-based attacks
- ❑ **Objective: Spatial and temporal uniqueness in encryption**
 - ❑ *Spatial*: Ciphertexts of the same plaintext data at different addresses is unique
 - ❑ *Temporal*: Ciphertexts of the same plaintext data at same address is unique across different writes
- ❑ **Solution**: Counter-mode encryption (approved by NIST)

[1] A. Huang, "Hacking the Xbox: An Introduction to Reverse Engineering" No Starch Press, 2003

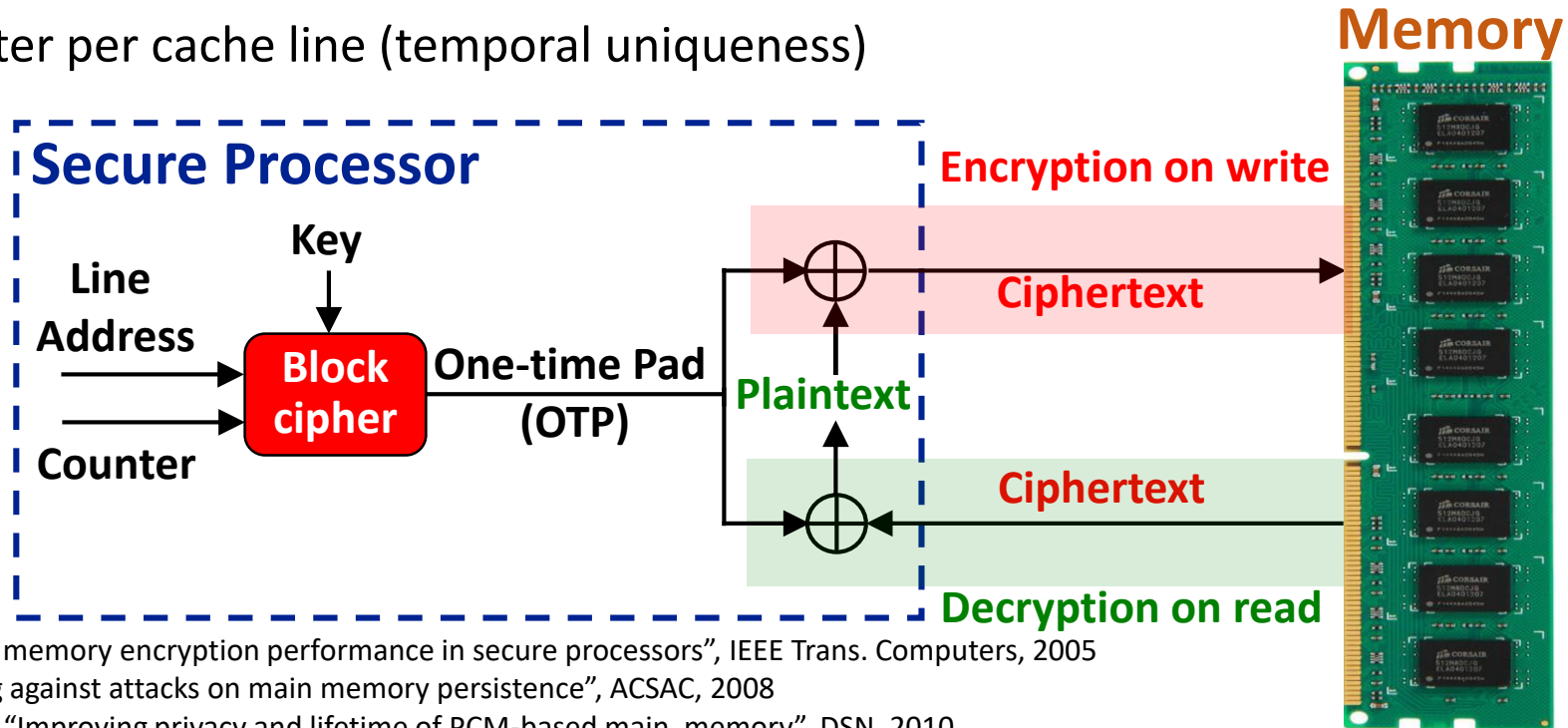
[2] A. B. Huang, "The Trusted PC: Skin-Deep Security", IEEE Computer, 2002

[3] A. Kumar, "Discovering Passwords in Memory", http://www.infosecwriters.com/text_resources/, 2004

[4] J. A. Halderman, "Lest We Remember: Cold Boot Attacks on Encryption Keys", USENIX Security Symposium, 2008

Data Confidentiality

- ❑ Counter mode encryption
 - ❑ One time pad (OTP): Random and unique
 - ❑ Secret key (randomness)
 - ❑ Cache line address (spatial uniqueness)
 - ❑ Counter per cache line (temporal uniqueness)



- [1] J. Yang *et al.*, "Improving memory encryption performance in secure processors", IEEE Trans. Computers, 2005
- [2] W. Enck *et al.*, "Defending against attacks on main memory persistence", ACSAC, 2008
- [3] J. Kong and H. Zhou, "Improving privacy and lifetime of PCM-based main memory", DSN, 2010
- [4] V. Young *et al.*, "DEUCE: Write-efficient encryption for non-volatile memories", ASPLOS, 2015

Data Confidentiality

- ❑ Counter mode encryption
 - ❑ Low decryption latency
 - ❑ OTP: Requires counter, address, secret key
 - ❑ Address + secret key available on processor
 - ❑ Counter stored along with data in memory
 - ❑ Needs to be on the processor
- ❑ Counter cache: Caches counters on processor

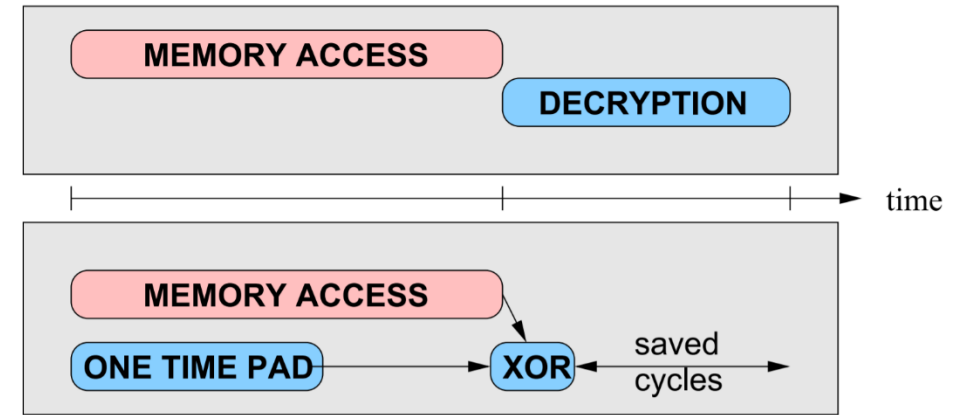
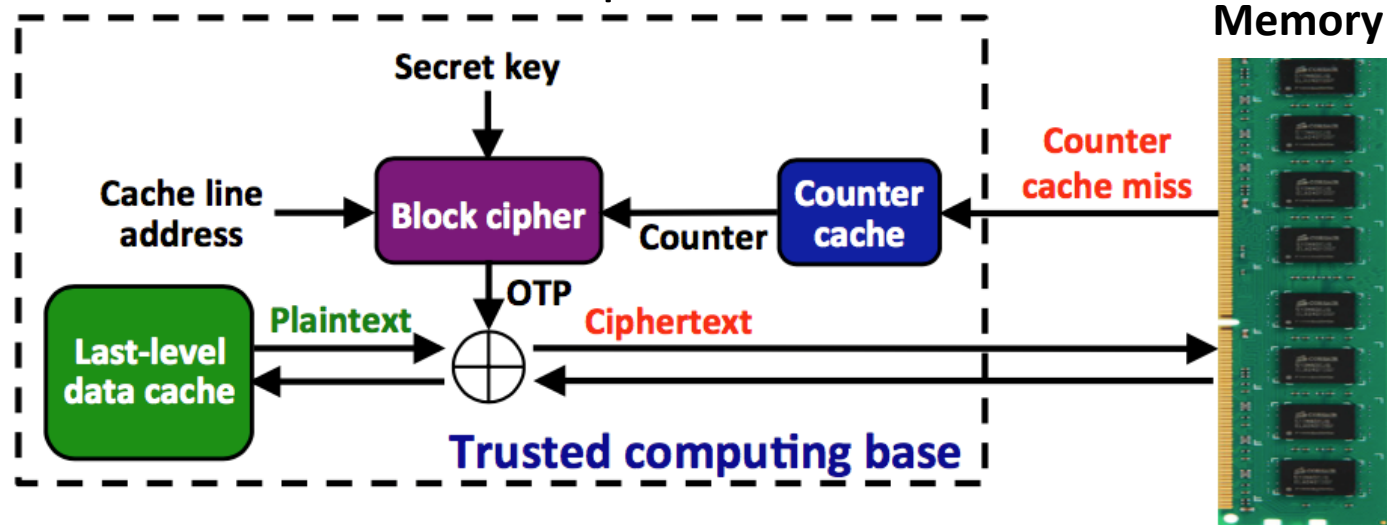


Figure credit: [1]



Agenda

- ❑ Memory security
 - ❑ The 3 basic pillars
 - ❑ Trusted Computing Base
- ❑ Data confidentiality
- ❑ Data integrity**
- ❑ Access pattern obfuscation
 - ❑ Oblivious RAM
- ❑ Our work

Data Integrity

- ❑ Prevention of *undetected*, unauthorized tampering of data by adversary

Data Integrity: Attacks

- ❑ Memory data integrity: Attacks
 - ❑ **Spoofing**



Data Integrity: Attacks

- ❑ Memory data integrity: Attacks
 - ❑ **Spoofing**



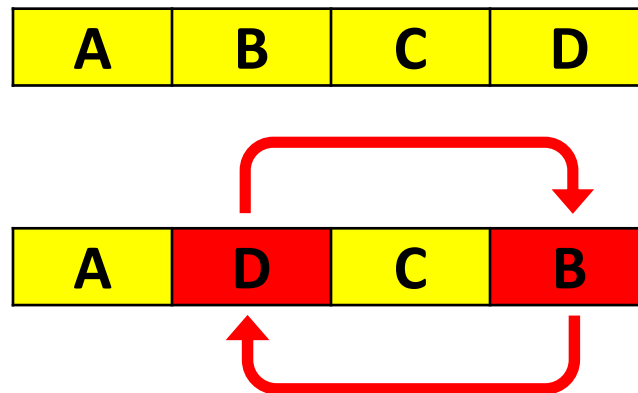
Attacker changes data at
a particular memory location

Data Integrity: Attacks

- ❑ Memory data integrity: Attacks

- ❑ Spoofing

- ❑ Splicing**



Attacker swaps data
between 2 memory locations

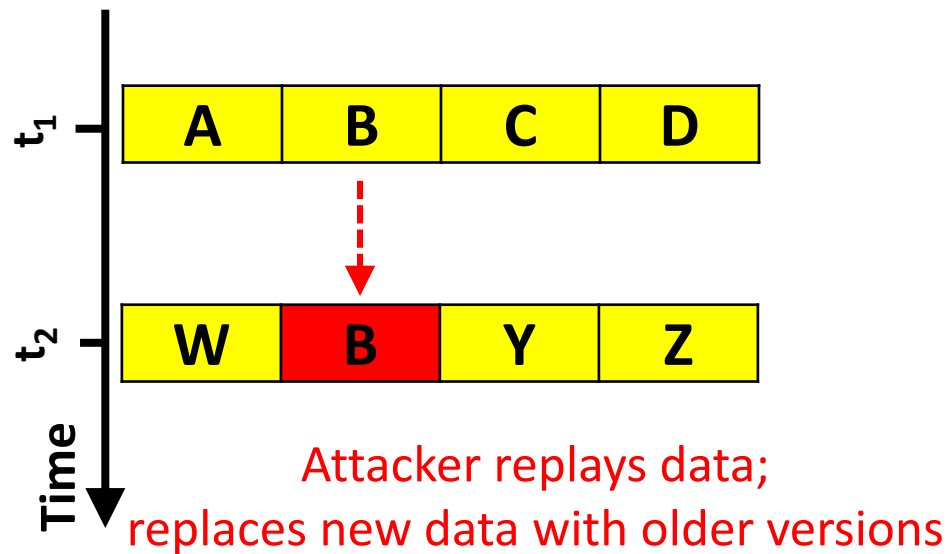
Data Integrity: Attacks

- ❑ Memory data integrity: Attacks

- ❑ Spoofing

- ❑ Splicing

- ❑ Replay**



Memory Authentication

- ❑ Memory data integrity: Authentication
 - ❑ HMAC: **H**ashed **M**essage **A**uthentication **C**ode
 - ❑ Keyed cryptographic hash signature
 - ❑ Stored along with data in memory
 - ❑ Prevents spoofing, prone to splicing and replay

$$DH = \text{HMAC}_k(D)$$

└─┬─> Secret key

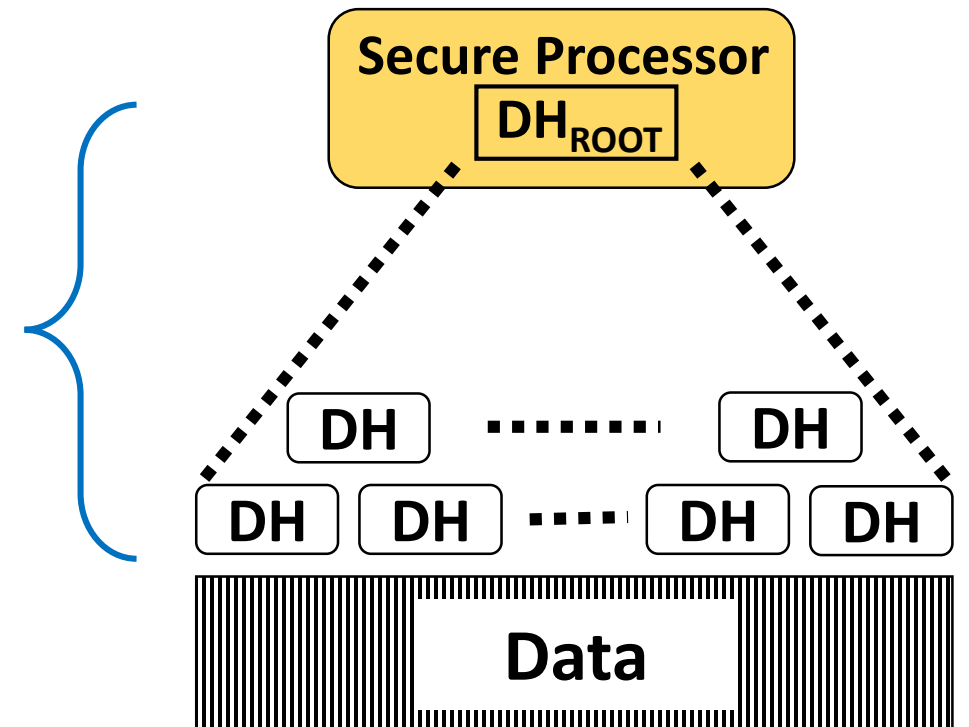


Memory Authentication

- ❑ Memory data integrity: Authentication
 - ❑ Merkle Tree

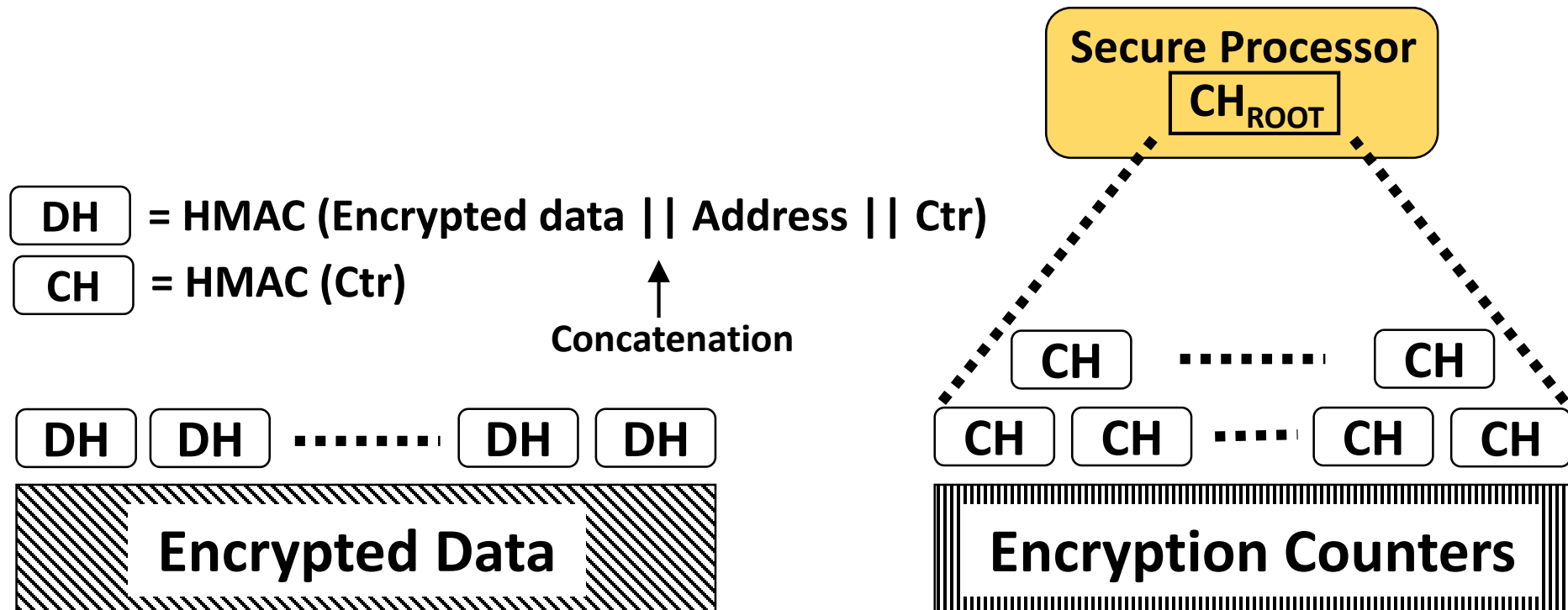
Merkle Tree (MT): Data structure constructed by *recursive hashing*, culminates in a root stored on secure processor.

- Secure root **cannot be spoofed, spliced, or replayed**, hence data is tamper-proof.



Memory Authentication

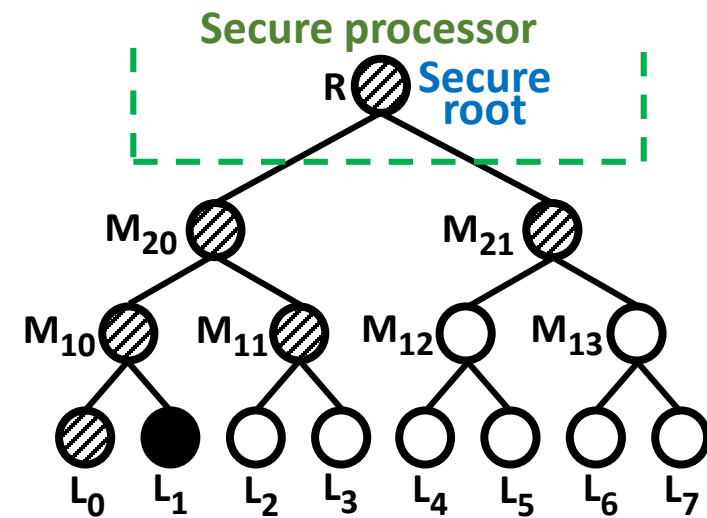
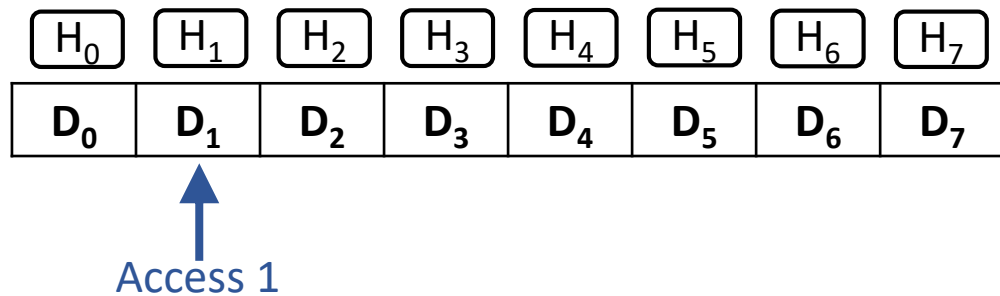
- ❑ Memory data integrity
 - ❑ Bonsai Merkle Tree (BMT) [1]
 - ❑ Authentication uses encryption counters
 - ❑ Protecting counters ensures trusted decryption



Memory Authentication

□ BMT: Operating details

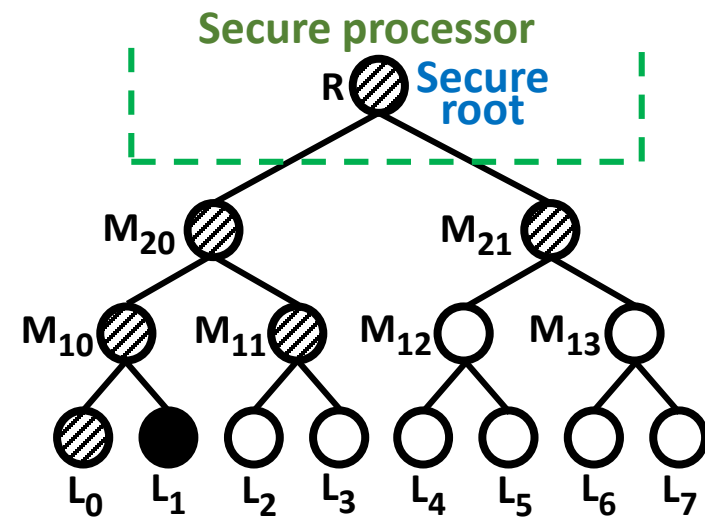
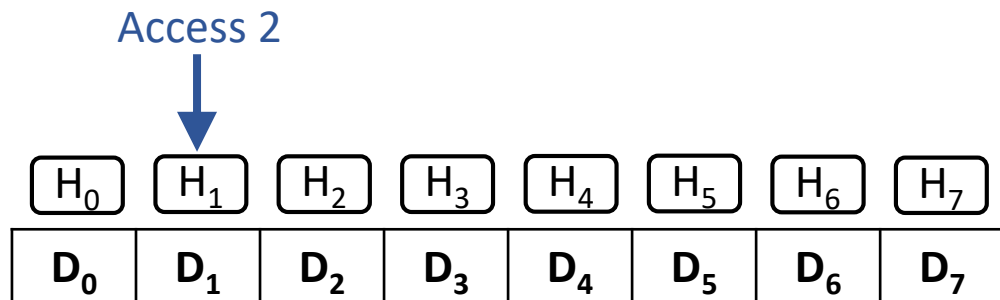
- Counter read/updated
- ◐ Nodes traversed for authentication



Memory Authentication

❑ BMT: Operating details

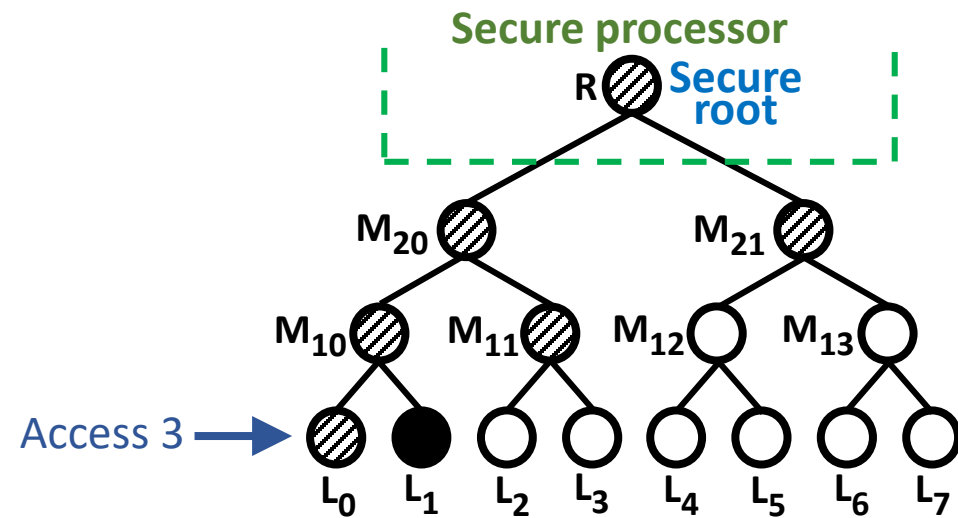
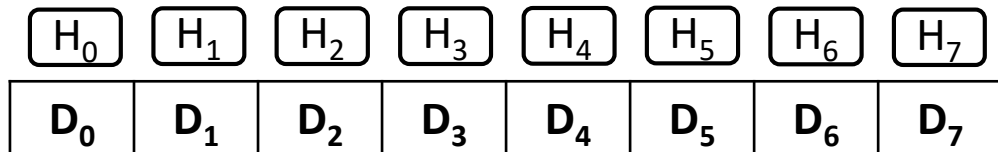
- Counter read/updated
- ◐ Nodes traversed for authentication



Memory Authentication

□ BMT: Operating details

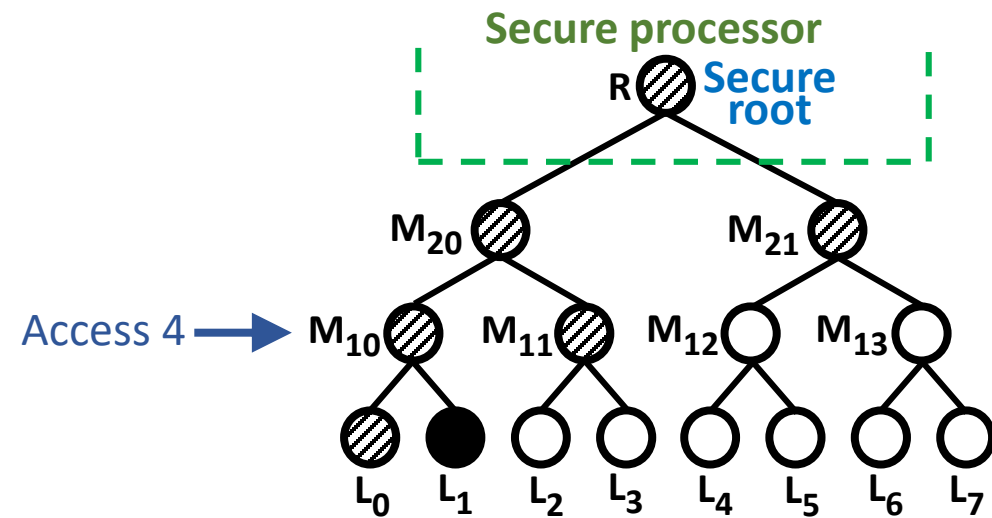
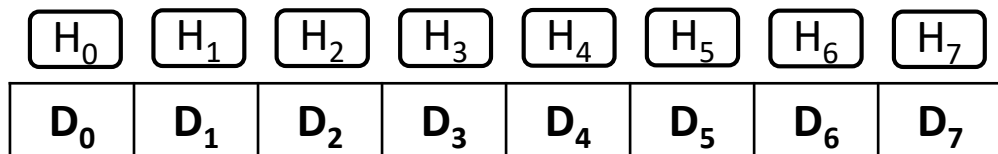
- Counter read/updated
- ◐ Nodes traversed for authentication



Memory Authentication

□ BMT: Operating details

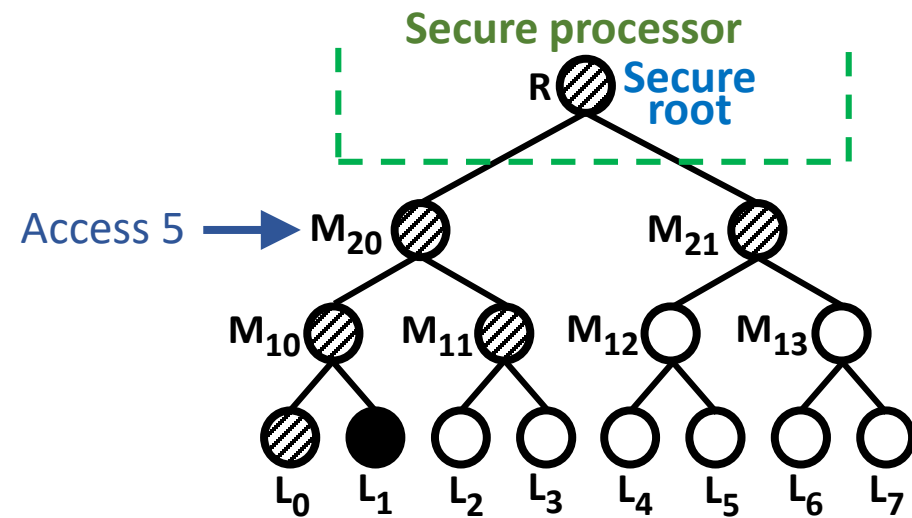
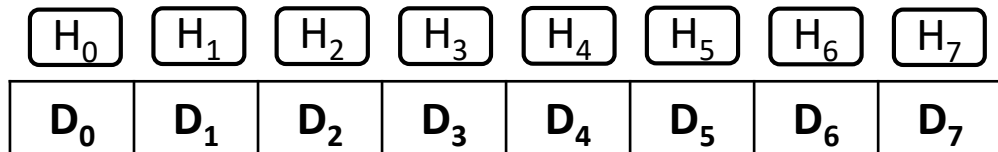
- Counter read/updated
- ◐ Nodes traversed for authentication



Memory Authentication

□ BMT: Operating details

- Counter read/updated
- ◐ Nodes traversed for authentication



Agenda

- ❑ Memory security
 - ❑ The 3 basic pillars
 - ❑ Trusted Computing Base
- ❑ Data confidentiality
- ❑ Data integrity
- ❑ Access pattern obfuscation
 - ❑ Oblivious RAM
- ❑ Our work

Revisiting Confidentiality

- ❑ Confidentiality
 - ❑ Ensured by data encryption
 - ❑ Security assumption: **Secret key is secure**
 - ❑ **Does not hold in presence of side-channels**
- ❑ Traditional memory request for encrypted memory
 - ❑ Data → encrypted
 - ❑ Address and commands → unencrypted

Revisiting Confidentiality

- ❑ Illustration: Security vulnerabilities of exposing access patterns
 - ❑ A section of exponentiation function in public-key cryptosystems [1]
 - ❑ $y = x^k$; where x = input, k = secret key
 - ❑ Following the access pattern, the attacker can decipher the secret key
 - ❑ No knowledge of addresses of 'r' and 's', it gets 2 possible options, the key and it's complement

```
for i=0 to N-1{  
  if (secret_key[i] == 0)  
    r = <computation1>  
  else  
    s = <computation2> }
```

Address of 'r': 0x00

Address of 's': 0x20



0x00, 0x20, 0x20, 0x00, ...



Revisiting Confidentiality

- ❑ Plaintext address presents security vulnerabilities
 - ❑ Access patterns can reveal
 - ❑ **Encryption keys**, eliminating the security guarantees of encryption [1,2]
 - ❑ **Control flow graph** for the program, leading to IP leakage [3]
- ❑ Require access pattern obfuscation for confidentiality

[1] M. S. Islam *et al.*, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation," NDSS, 2012

[2] T. John *et al.*, "Connecting the dots: Privacy leakage via write-access patterns to the main memory," HOST, 2017

[3] X. Zhuang *et al.*, "HIDE: An infrastructure for efficiently protecting information leakage on the address bus," ASPLOS, 2004

Access Pattern Obfuscation

- ❑ *Objective*: Memory access pattern obfuscation [1]
 - ❑ No information leakage about
 - ❑ Address being accessed
 - ❑ Plaintext data at the accessed address
 - ❑ Type of memory access: Read or write
 - ❑ Linkability: Whether same address is being accessed

Oblivious RAM

- ❑ Oblivious RAM algorithms
 - ❑ Widely used to achieve complete access pattern obfuscation
- ❑ 2 major categories
 - ❑ Hierarchical ORAM [1-3]
 - ❑ Memory arranged as a tiered hierarchy
 - ❑ Suitable for large client storage
 - ❑ Tree based ORAM [4-7]
 - ❑ Memory arranged as a binary/ N -ary tree
 - ❑ Suitable for small client storage
 - ❑ ***Path ORAM*: simplest and most efficient**

[1] R.Ostrovsky *et al.*, "Private information storage", STOC, 1997

[2] E. Stefanov *et al.*, "Towards practical Oblivious RAM", NDSS, 2012

[3] J. Dutrich *et al.*, "Burst ORAM: Minimizing ORAM response times for bursty access patterns", USENIX Security, 2014

[4] E. Shi, *et al.*, "Oblivious RAM with $O((\log n)^3)$ worst-case cost", ASIACRYPT, 2011

[5] E. Stefanov *et al.*, "Path ORAM: An Extremely Simple Oblivious RAM Protocol", CCS, 2013

[6] R. Wang *et al.*, "Cooperative Path-ORAM for effective memory bandwidth sharing in server settings," HPCA, 2017.

[7] R. Wang *et al.*, "D-ORAM: Path-ORAM delegation for low execution interference on cloud servers with untrusted memory," HPCA 2018.

Path ORAM

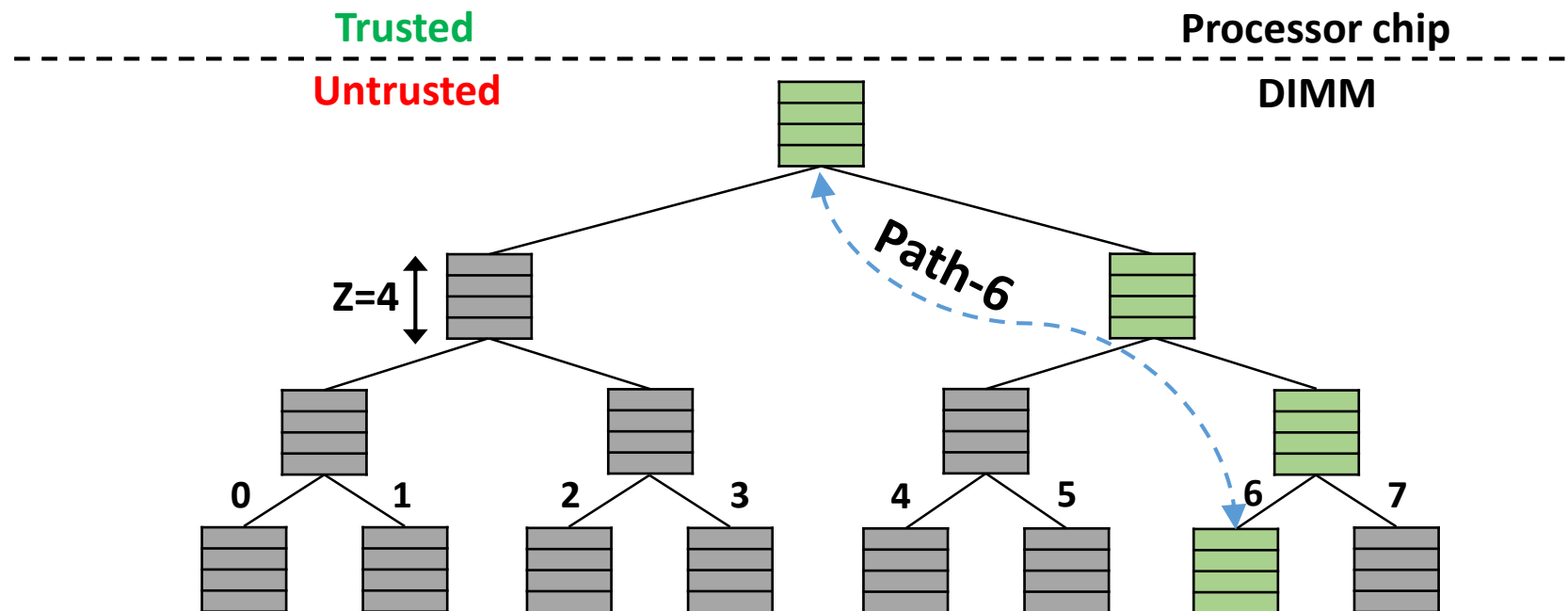
- ❑ Path ORAM comprised of
 - ❑ Untrusted external memory
 - ❑ Trusted ORAM controller

- ❑ Address nomenclature



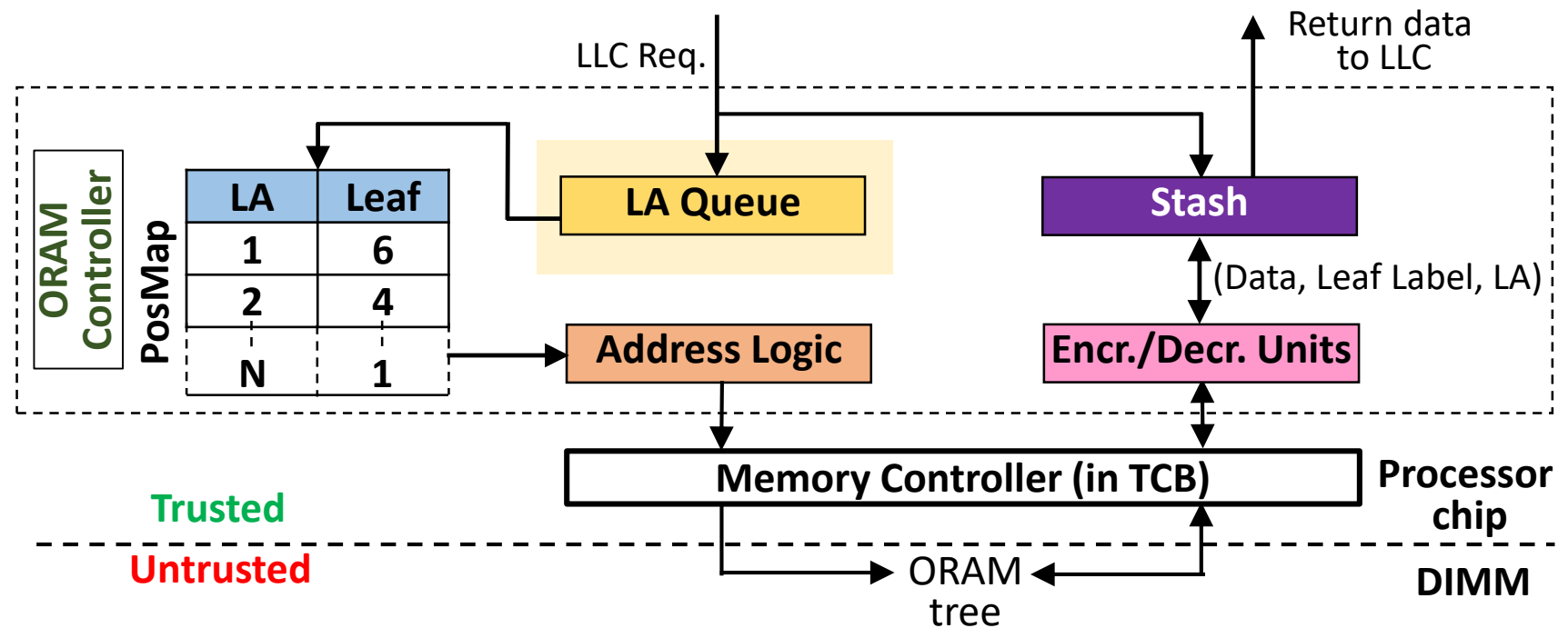
Path ORAM

- ❑ External memory organized as a binary tree
 - ❑ Each logical address (LA) data block is randomly mapped to a leaf (LeafID)
- ❑ The unique path from the root at any leaf x is called path- x
- ❑ Each node is termed a *bucket*
 - ❑ Each bucket can hold Z encrypted data/dummy blocks (cache lines)



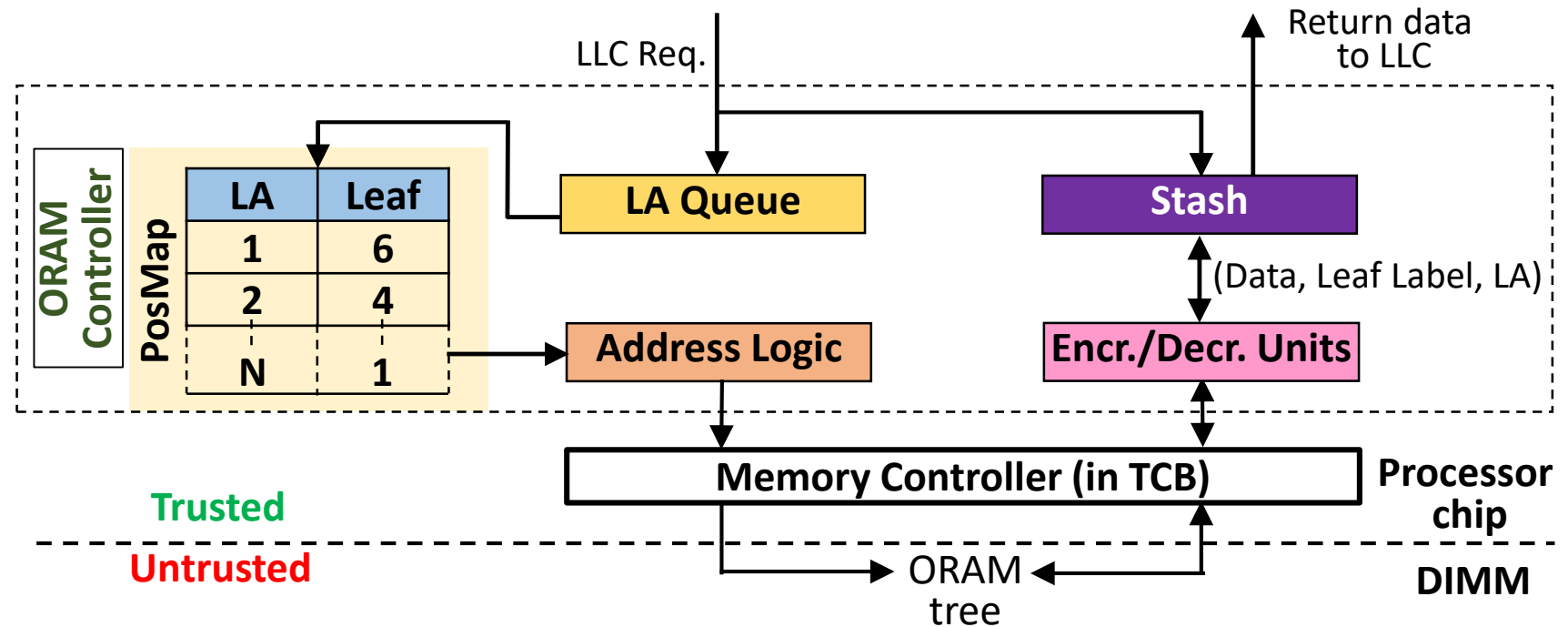
Path ORAM

- ❑ ORAM controller: Frontend
 - ❑ **LA queue**: Buffers memory requests from LLC misses
 - ❑ **PosMap**: Stores random mapping to LA to leaves (LeafID)
 - ❑ **Address Logic**: Generates ORAM addresses for nodes on a given path



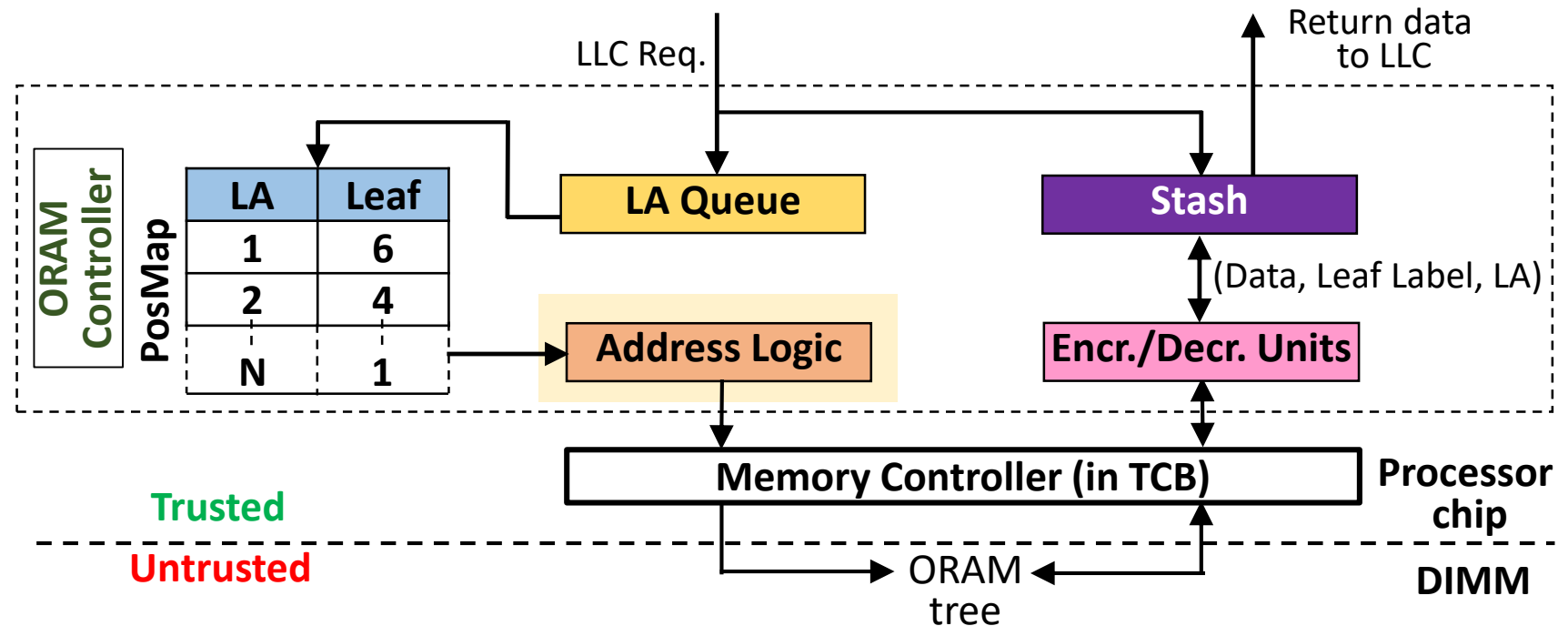
Path ORAM

- ❑ ORAM controller: Frontend
 - ❑ **LA queue**: Buffers memory requests from LLC misses
 - ❑ **PosMap**: Stores random mapping to LA to leaves (LeafID)
 - ❑ **Address Logic**: Generates ORAM addresses for nodes on a given path



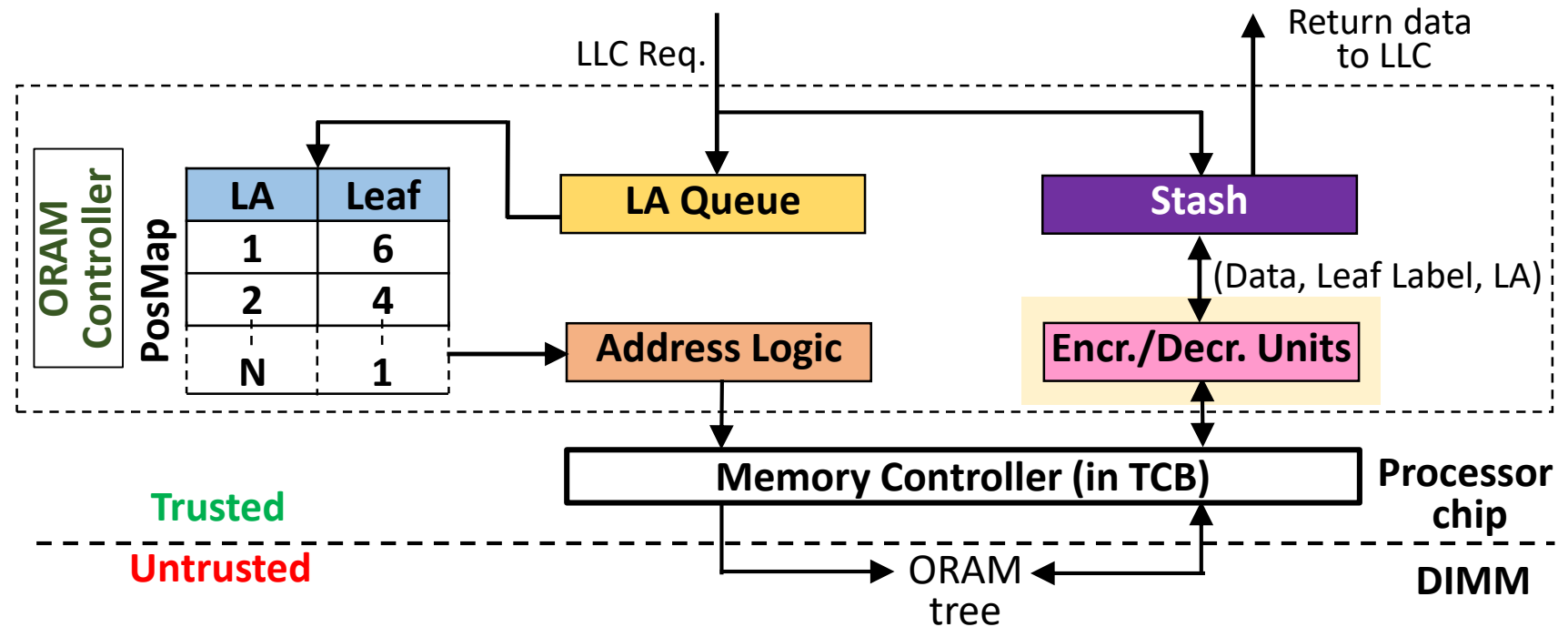
Path ORAM

- ❑ ORAM controller: Frontend
 - ❑ **LA queue**: Buffers memory requests from LLC misses
 - ❑ **PosMap**: Stores random mapping to LA to leaves (LeafID)
 - ❑ **Address Logic**: Generates ORAM addresses for nodes on a given path



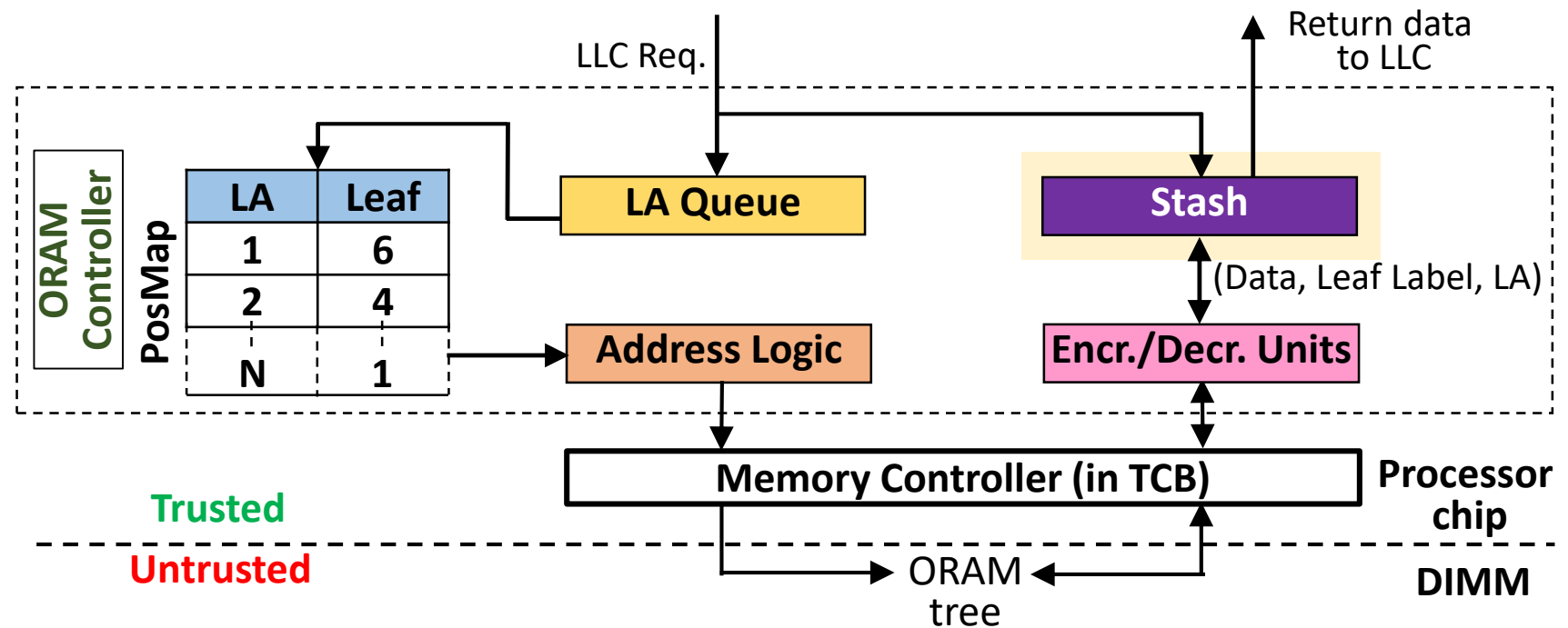
Path ORAM

- ❑ ORAM controller: Backend
 - ❑ **Encryption/decryption units:** Encrypts/decrypts data block
 - ❑ **Stash:** Stores decrypted data blocks fetched from ORAM



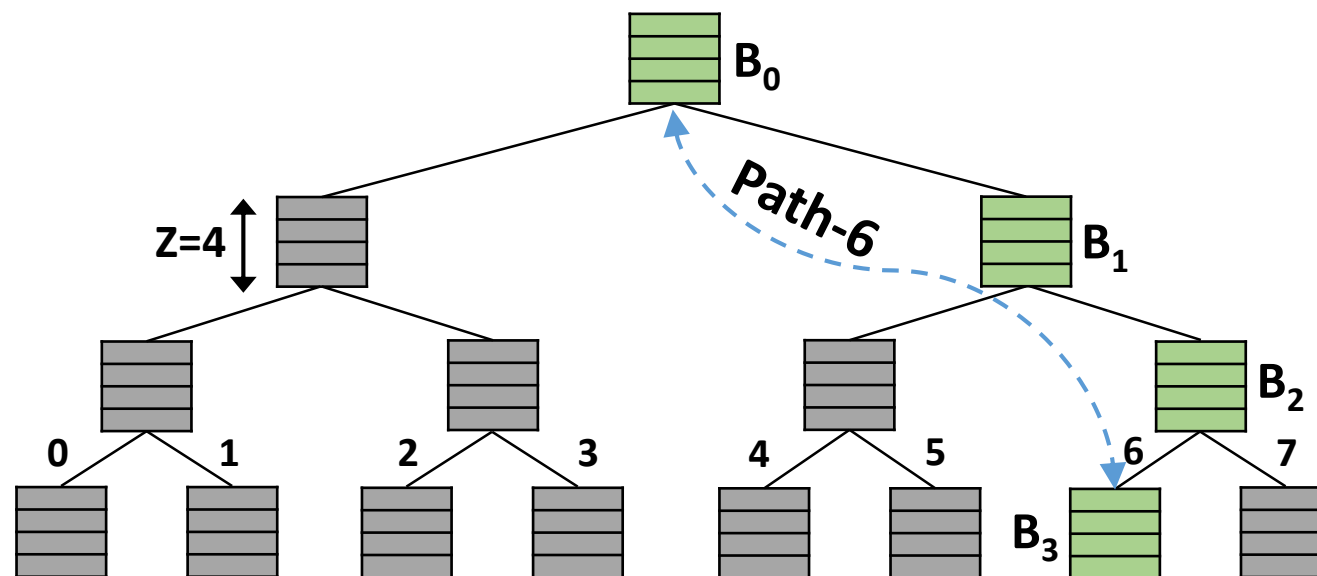
Path ORAM

- ❑ ORAM controller: Backend
 - ❑ **Encryption/decryption units:** Encrypts/decrypts data block
 - ❑ **Stash:** Stores decrypted data blocks fetched from ORAM



Path ORAM

- ❑ The *main invariant* by construction
 - ❑ A data block, mapped to a leaf x is either in **any** bucket on the path- x or in the stash
- ❑ On an LLC miss
 - ❑ The ORAM controller is queried with the *accessORAM* (*addr, op, data*) interface
 - ❑ The ORAM controller performs an ORAM access
 - ❑ Read phase
 - ❑ Write phase

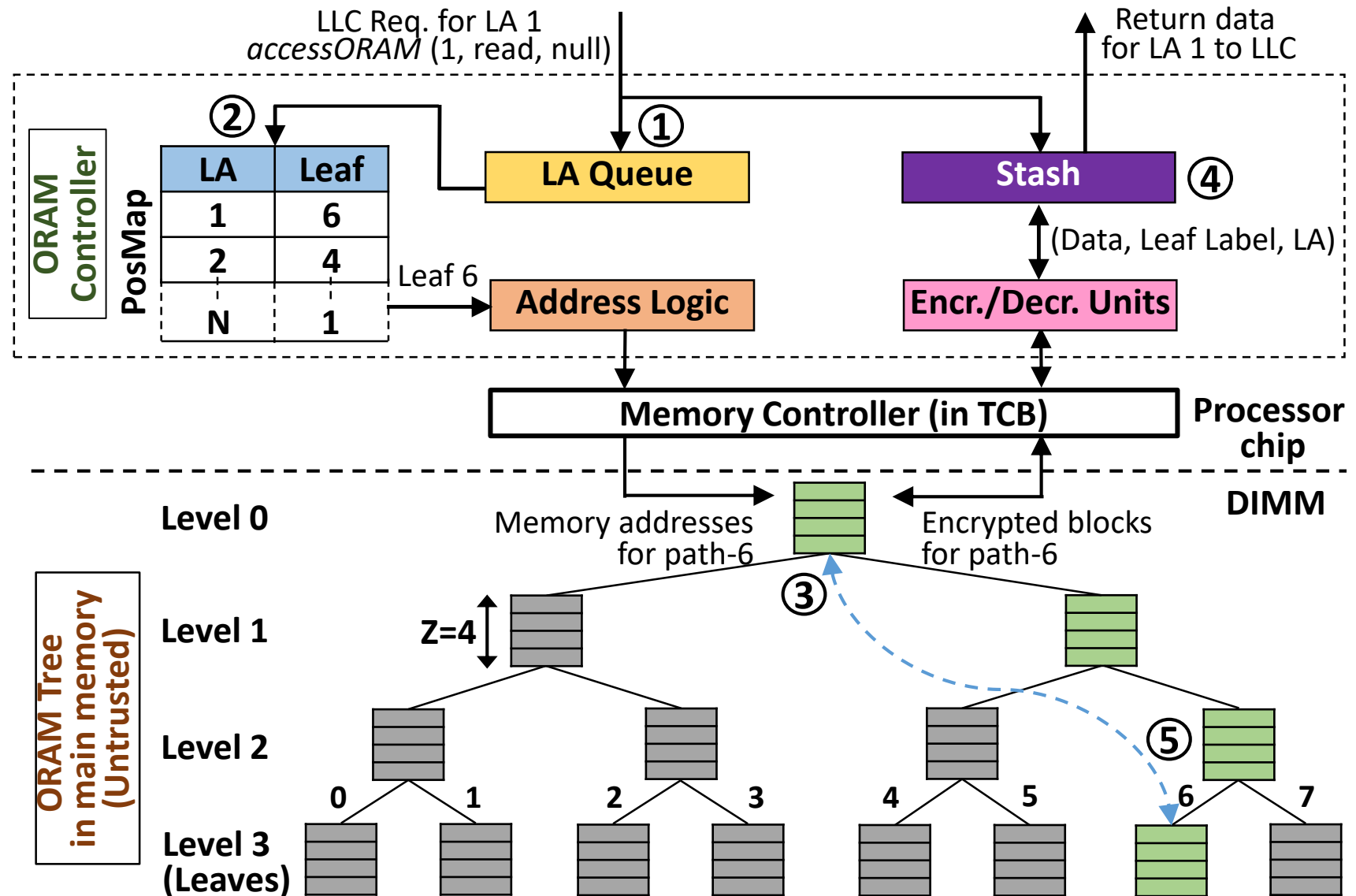


Path ORAM

Path ORAM access

Illustration

Read LA 1



Path ORAM

- ❑ Security
 - ❑ Address: Randomized mapping secret from adversary
 - ❑ Plaintext data: Encrypted
 - ❑ Type of memory access: Both read and write phase for each ORAM access
 - ❑ Linkability: Remapping after every access, path access
- ❑ New ORAM architectures
 - ❑ Must preserve baseline Path ORAM guarantees
 - ❑ Must not leak *additional useful information*

Agenda

- ❑ Memory security
 - ❑ The 3 basic pillars
 - ❑ Trusted Computing Base
- ❑ Data confidentiality
- ❑ Data integrity
- ❑ Access pattern obfuscation
 - ❑ Oblivious RAM
- ❑ **Our work**

Emerging Non-Volatile Memories

- ❑ Main memory requirements and DRAM drawbacks
 - ❑ Capacity: DRAM density hard to scale [1]
 - ❑ Energy: High DRAM refresh power due to leakage [2-8]
- ❑ PCM and RRAM: Emerging NVMs [2-8]
 - ❑ Better scalability
 - ❑ High data density (MLC – 2 bits/cell, TLC – 3 bits/cell)
 - ❑ Data persistence – no refresh power

[1] International Technology Roadmap for Semiconductors, 2011

[2] M.K.Qureshi *et al.*, “Scalable high performance main memory system using phase-change memory technology”, ISCA, 2009

[3] B. C. Lee *et al.*, “Phase change technology and the future of main memory,” IEEE Micro, 2010

[4] A. Ferreira *et al.*, “Increasing PCM main memory lifetime,” DATE, 2010

[5] S. Sheu *et al.*, “Fast-write resistive RAM (RRAM) for embedded applications,” IEEE Design and Test of Computers, 2011

[6] S. Bock *et al.*, “Analyzing the impact of useless write-backs on the endurance and energy consumption of PCM main memory,” ISPASS, 2011

[7] L. Jiang *et al.*, “Improving write operations in MLC phase change memory,” HPCA, 2012

[8] C. Xu *et al.*, “Understanding the trade-offs in multi-level cell ReRAM memory design,” DAC, 2013

Emerging Non-Volatile Memories

- ❑ Main memory requirements and DRAM drawbacks
 - ❑ Capacity: DRAM density hard to scale [1]
 - ❑ Energy: High DRAM refresh power due to leakage [2-8]
- ❑ PCM and RRAM: Emerging NVMs [2-8]
 - ❑ Better scalability
 - ❑ High data density (MLC – 2 bits/cell, TLC – 3 bits/cell)
 - ❑ Data persistence – no refresh power
 - ❑ Low endurance
 - ❑ High write energy/latency

[1] International Technology Roadmap for Semiconductors, 2011

[2] M.K.Qureshi *et al.*, “Scalable high performance main memory system using phase-change memory technology”, ISCA, 2009

[3] B. C. Lee *et al.*, “Phase change technology and the future of main memory,” IEEE Micro, 2010

[4] A. Ferreira *et al.*, “Increasing PCM main memory lifetime,” DATE, 2010

[5] S. Sheu *et al.*, “Fast-write resistive RAM (RRAM) for embedded applications,” IEEE Design and Test of Computers, 2011

[6] S. Bock *et al.*, “Analyzing the impact of useless write-backs on the endurance and energy consumption of PCM main memory,” ISPASS, 2011

[7] L. Jiang *et al.*, “Improving write operations in MLC phase change memory,” HPCA, 2012

[8] C. Xu *et al.*, “Understanding the trade-offs in multi-level cell ReRAM memory design,” DAC, 2013

Emerging Non-Volatile Memories

❑ PCM and RRAM: Emerging NVMs

❑ Better scalability

❑ High data density (MLC – 2 bits/cell, TLC – 3 bits/cell)

❑ Data persistence – no refresh power

❑ Low endurance

❑ High write energy/latency



Architecture based solutions

1. Cell flip reduction [1-3]
2. Wear levelling and error-correction [4-6]
3. Data mapping [7-9]

[1] B. Young *et al.*, “A low power phase change random access memory using a data-comparison write scheme,” ISCS, 2007

[2] S. Cho *et al.*, “Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance,” MICRO, 2009

[3] P. Palangappa *et al.*, “Compex: Compression-expansion coding for energy, latency, and lifetime improvements in MLC/TLC NVM”, HPCA, 2016

[4] M. Qureshi *et al.*, “Enhancing lifetime and security of PCM-based main memory with Start-Gap wear leveling,” MICRO, 2009

[5] S. Schechter *et al.*, “Use ECP, not ECC, for hard failures in resistive memories”, ISCA, 2010

[6] R. Wang *et al.*, “SD-PCM: Constructing reliable super dense Phase Change Memory under write disturbance”, ASPLOS 2015

[7] L. Jiang *et al.*, “Improving write operations in MLC phase change memory”, HPCA, 2012

[8] X. Zhang *et al.*, “TriState-SET: Proactive SET for improved performance of MLC phase change memories”, ICCD, 2015

[9] J.Li *et al.*, “Write-once-memory-code phase change memory”, DATE, 2014

Emerging Non-Volatile Memories

- ❑ PCM and RRAM: Emerging NVMs
 - ❑ Better scalability
 - ❑ High data density (MLC – 2 bits/cell, TLC – 3 bits/cell)
 - ❑ Data persistence – no refresh power
 - ❑ Low endurance
 - ❑ High write energy/latency
 - ❑ **Security vulnerabilities [1-5]**

[1] J. Cong *et al.*, “Improving privacy and lifetime of PCM-based main memory,” DSN, 2010



[2] S. Chhabra and Y. Solihin, “i-NVMM: A secure non-volatile main memory system with incremental encryption,” ISCA, 2011

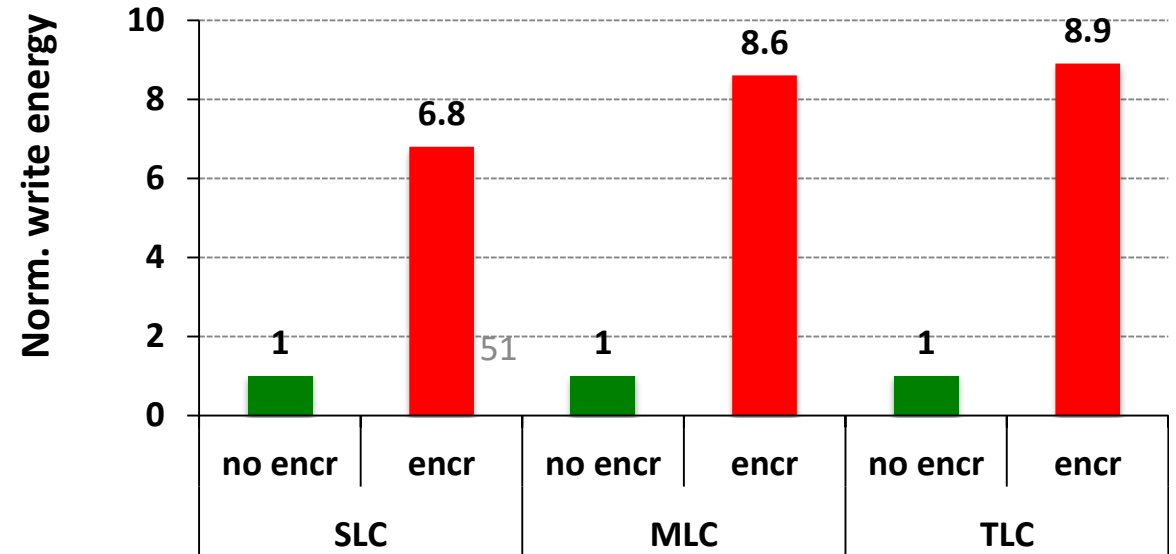
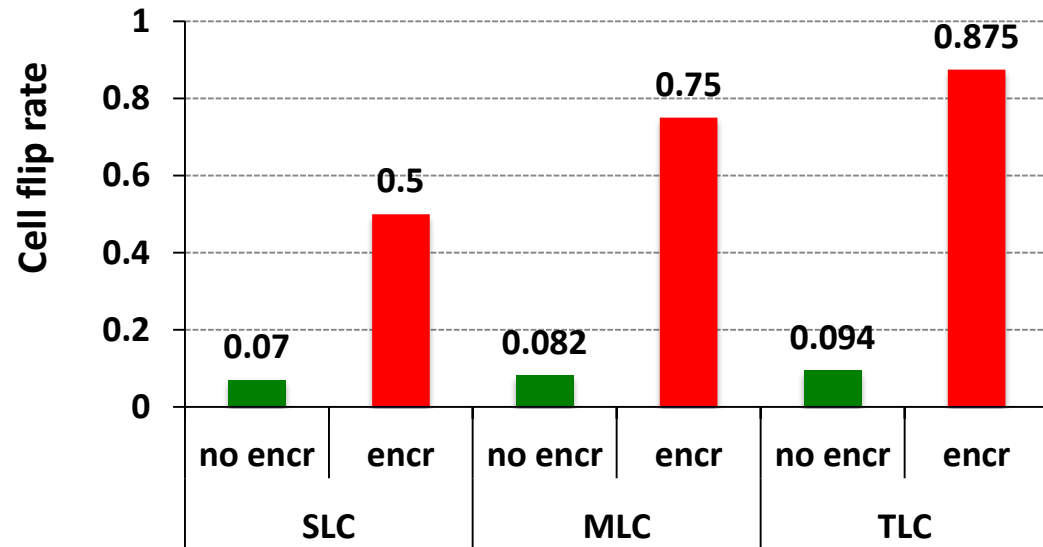
[3] V. Young *et al.*, “DEUCE: Write-efficient encryption for non-volatile memories,” ASPLOS, 2015

[4] A. Awad *et al.*, “Silent Shredder: Zero-cost shredding for secure non-volatile main memory controllers”, ASPLOS 2016

[5] S. Swami *et al.*, “SECRET: Smartly EnCRypted energy EfficienT non-volatile memories”, DAC, 2016

Challenges

- ❑ Memory encryption overheads
 - ❑ Write reduction schemes (e.g., DCW, FNW) are ineffective
 - ❑ Cell flip rate 
 - ❑ Write energy 



Challenges

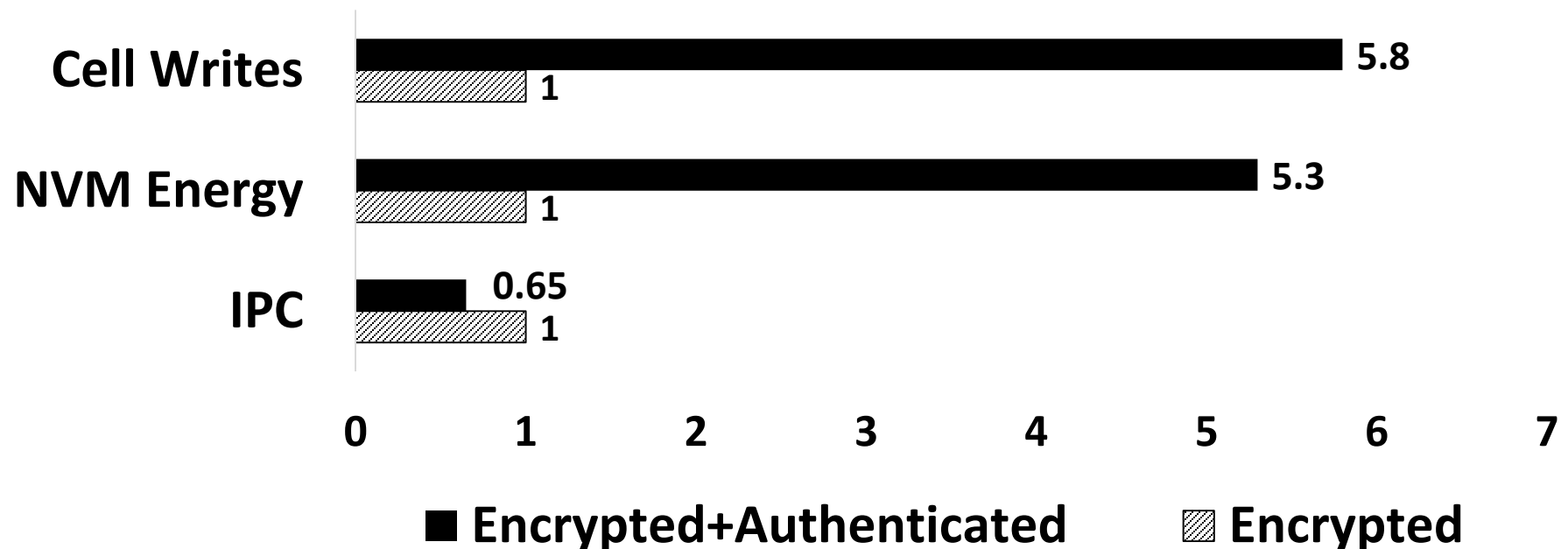
- ❑ Memory authentication overheads

- ❑ HMAC

- ❑ High entropy → High cell writes

- ❑ Merkle Tree

- ❑ HMAC node fetch/update → Additional memory accesses



Path ORAM

- ❑ Path ORAM overheads

- ❑ Access amplification

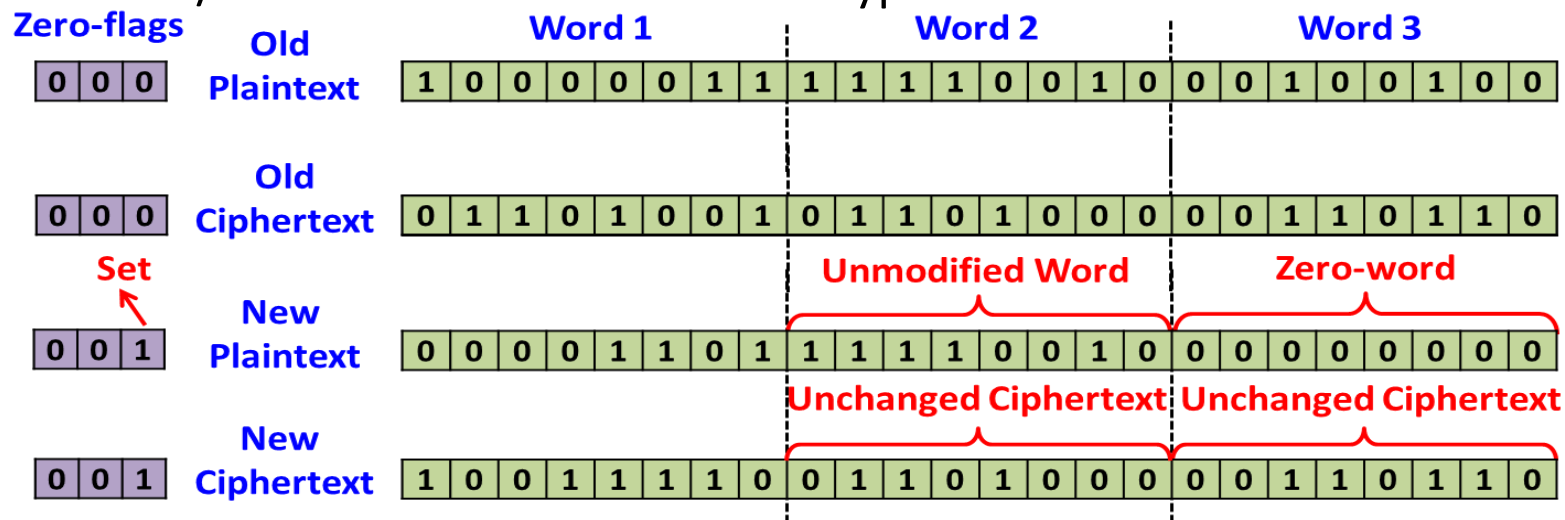
- ❑ High read latency: 10x (in DRAM), 40x (in SLC PCM)

- ❑ System performance degradation: 4x (DRAM), 10x (in SLC PCM)

- ❑ High write energy: 9x (in DRAM), 45x (SLC PCM)

SECRET

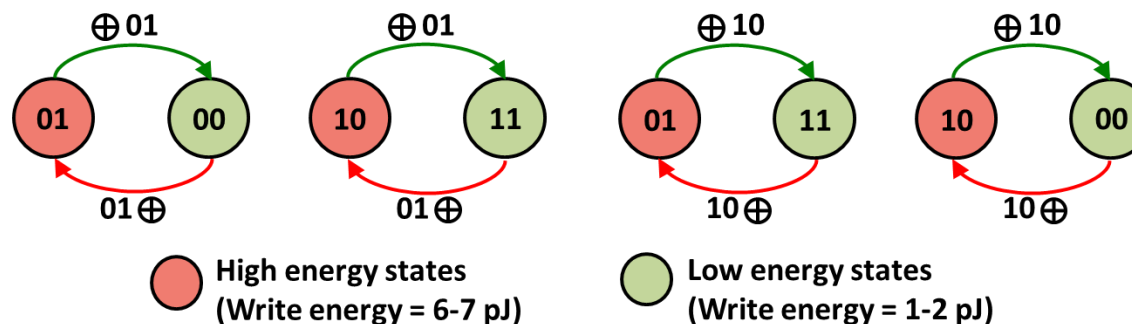
- ❑ **Objective:** Design secure MLC/TLC NVMs with
 - ❑ High endurance
 - ❑ Low write energy
- ❑ **Smartly EnCRypted Energy Efficient (SECRET) NVMs [1]**
 - ❑ **Smart encryption:** Reduces data re-encryption rate
 - ❑ Unmodified and/or zero-words are not re-encrypted



[1] S. Swami *et al.*, "SECRET: Smartly EnCRypted energy Efficient non-volatile memories", DAC, 2016

SECRET

- ❑ **Objective:** Design secure MLC/TLC NVMs with
 - ❑ High endurance
 - ❑ Low write energy
- ❑ **Smartly EnCRypted Energy EfficienT (SECRET) NVMs**
 - ❑ **Smart encryption:** Reduces data re-encryption rate
 - ❑ Unmodified and/or zero-words are not re-encrypted
 - ❑ **Energy masking:** Reduces write energy of ciphertext
 - ❑ XOR-based transformation from high to low energy state
- ❑ **Impact:** In comparison to state-of-the-art DEUCE [1]
 - ❑ 50% reduction in write energy
 - ❑ 20% improvement in lifetime

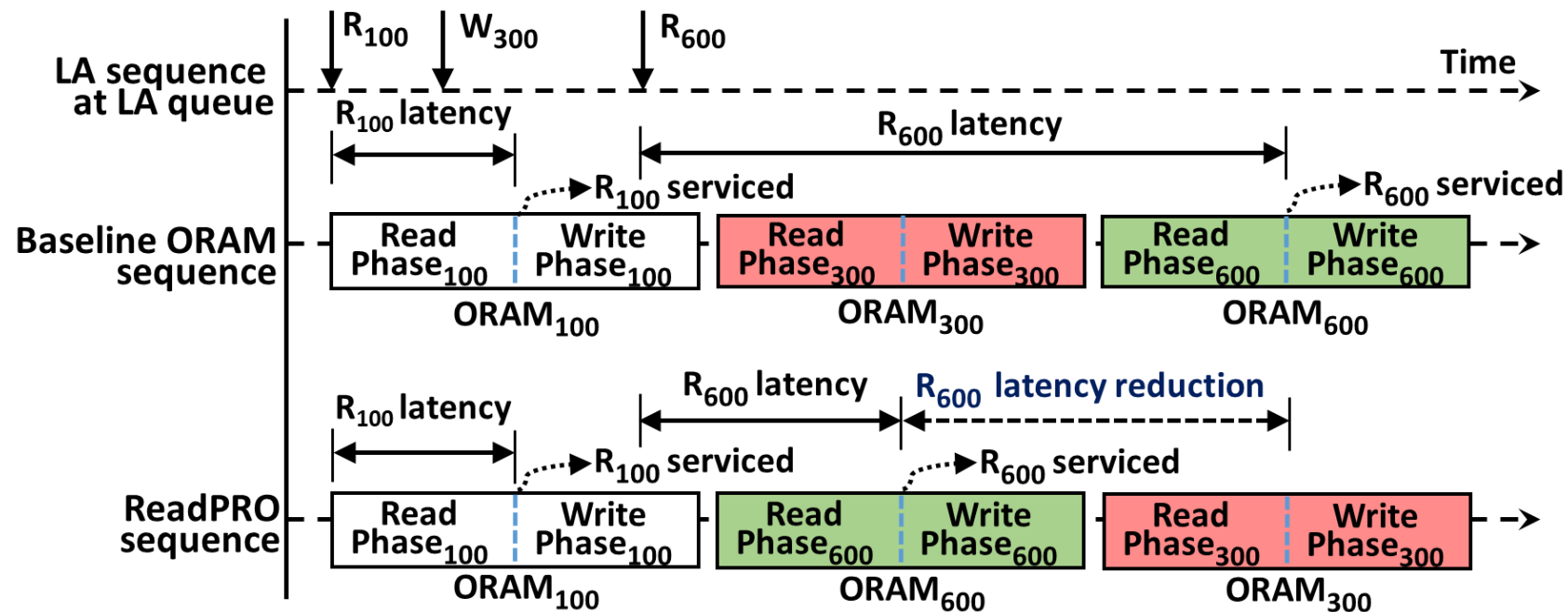


ReadPRO

- ❑ *Observation:* Write buffering in memory controller not effective in ORAM
 - ❑ Memory controller receives the requests after address translation
 - ❑ Allows read phase of LA write requests to pass through
 - ❑ Practical write buffer overflows within **1-2 write phases**
 - ❑ **Increasing write buffer capacity in memory controller: Ineffective**
 - ❑ 4x/8x/16x increase in write buffer size: 5%/8%/12% decrease in read latency

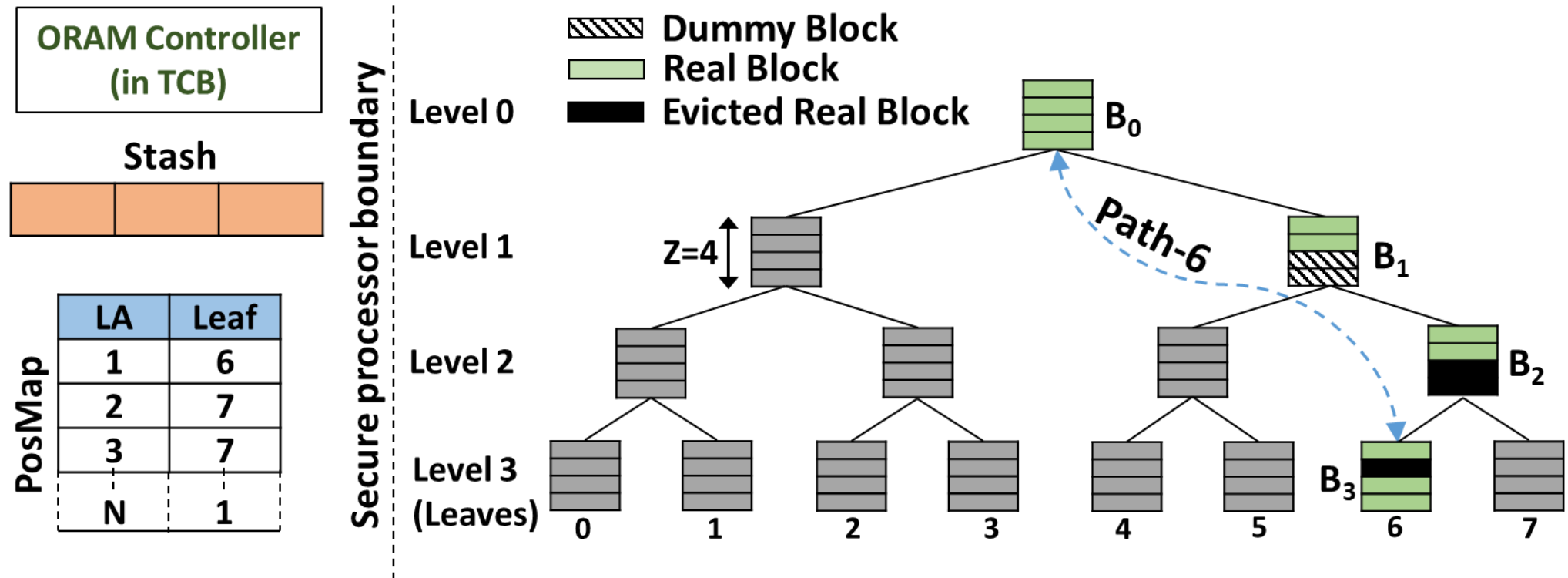
ReadPRO

- ❑ *ReadPRO*: Read promotion scheduling in ORAM [1]
 - ❑ Perform read prioritization higher up in the memory hierarchy at LLC-ORAM controller interface
 - ❑ Prioritization before ORAM address translation
 - ❑ Decrease blocking effects from non-critical ORAM accesses of LA writes



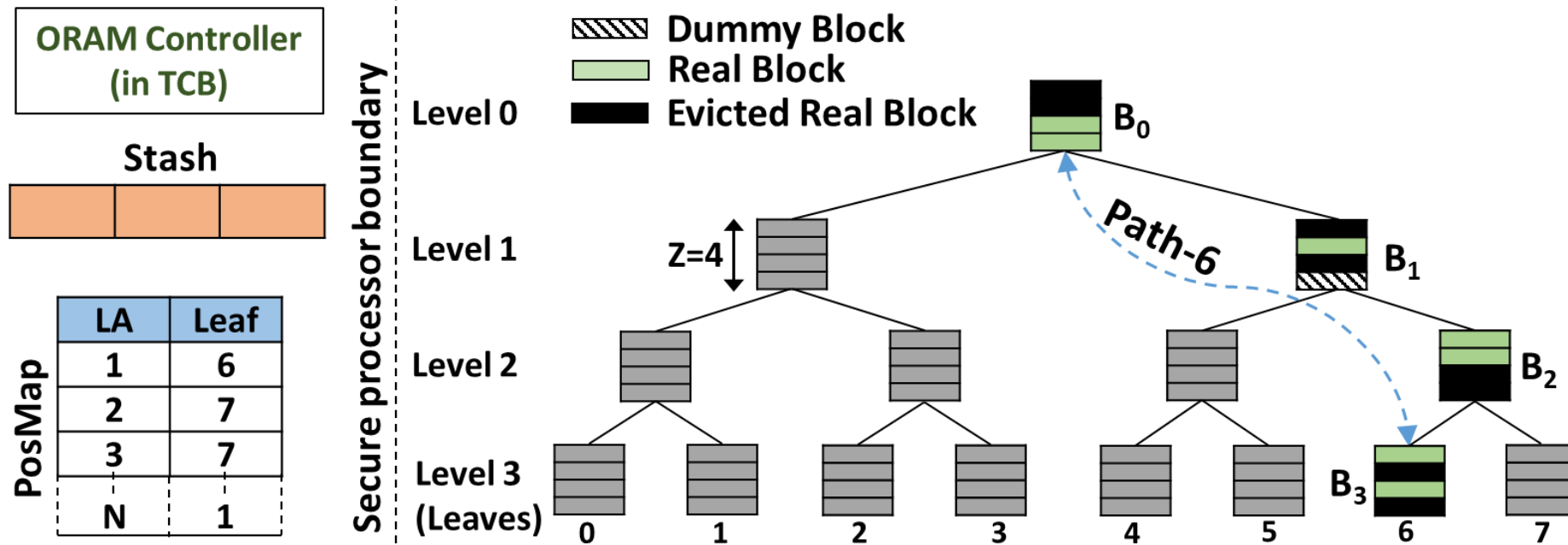
LEO

- ❑ All blocks on a path are re-encrypted during a write phase
- ❑ Only a few blocks are updated; they require mandatory re-encryptions
 - ❑ Exposes real blocks on a path; not secure



LEO

- LEO: Low Overhead Encryption ORAM for Non-Volatile Memories [1]
 - Evaluate highest on a path for a given write phase (MR_{MAX})
 - Enforce MR_{MAX} block re-encryptions in all buckets
 - For buckets with $MR_{COUNT} < MR_{MAX}$
 - Select $(MR_{MAX} - MR_{COUNT})$ blocks randomly to be re-encrypted



THANK YOU
Q&A

PARL is hiring