# Unified Memory Protection with Multi-granular MAC and Integrity Tree for Heterogeneous Processors

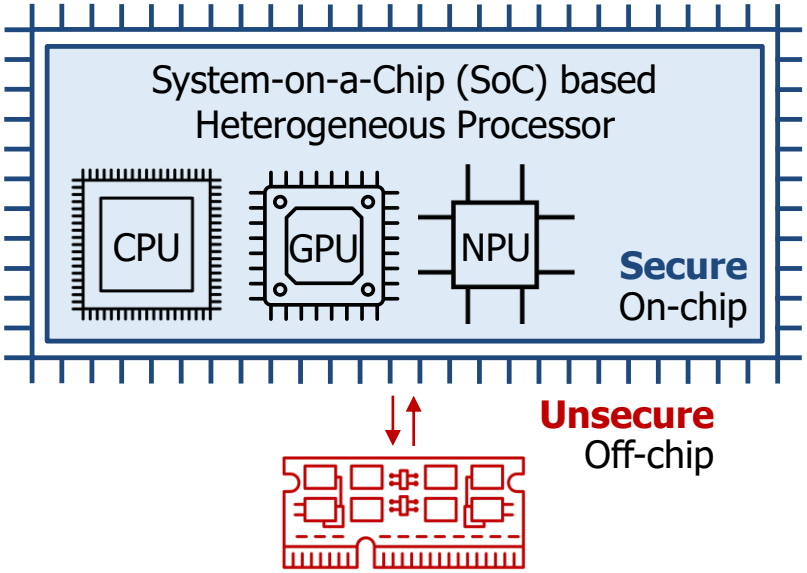
**Sunho Lee**[1], Seonjin Na[2], Jeongwon Choi[1]
Jinwon Pyo[1], Jaehyuk Huh[1]

KAIST[1], Georgia Tech[2]



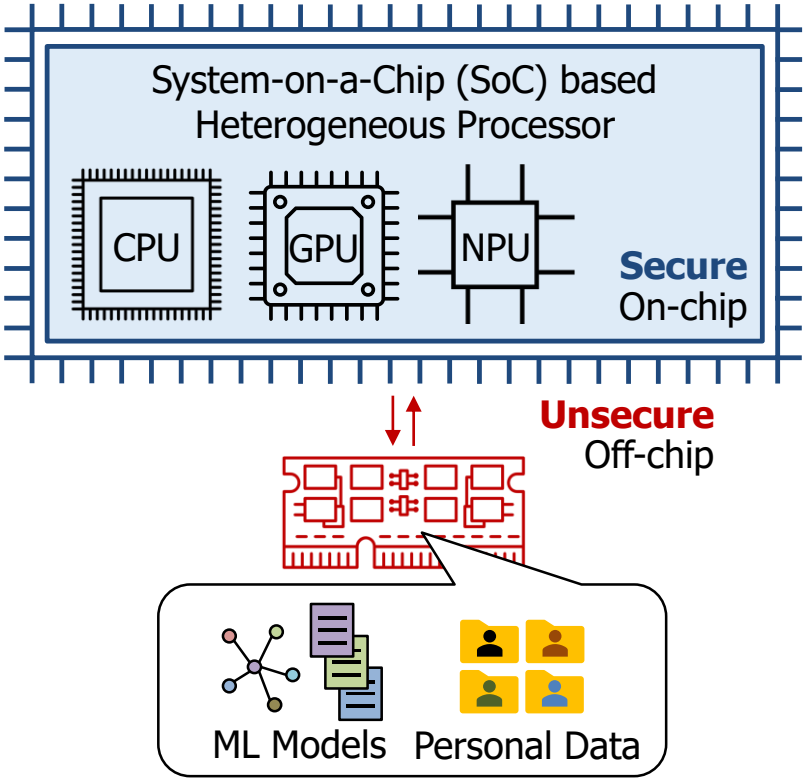
# Secure Heterogeneous Processor

- Heterogeneous processor: SoC with CPU, GPU, NPU



# Secure Heterogeneous Processor

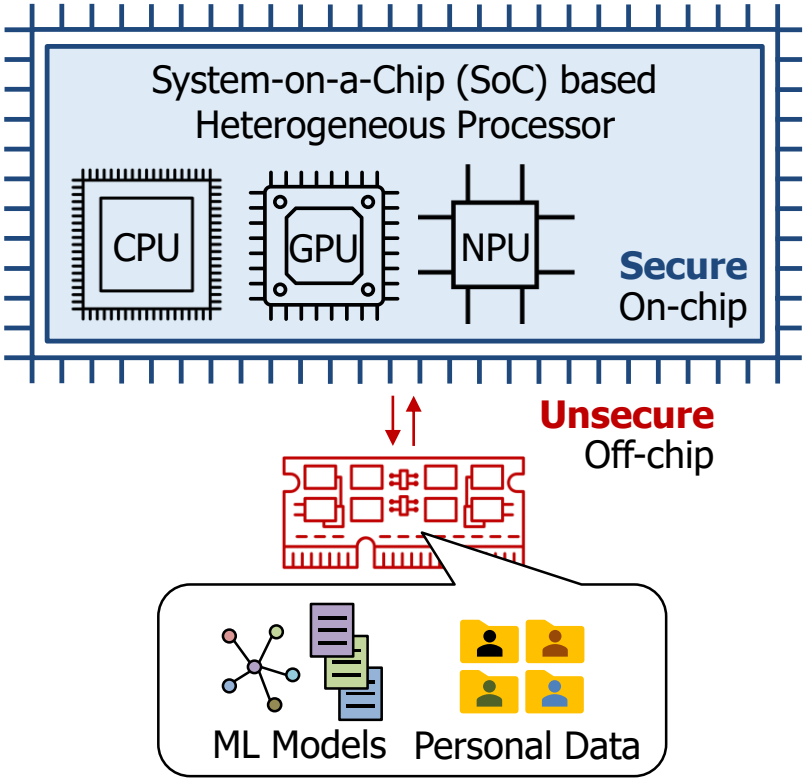
- Heterogeneous processor: SoC with CPU, GPU, NPU
- Data <u>confidentiality</u> & <u>integrity</u> are essential



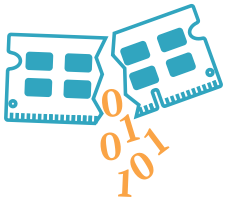

[1] Lest We Remember: Cold-Boot Attacks on Encryption Keys (USENIX Security 2008)
[2] RAMBleed: Reading Bits in Memory Without Accessing Them (S&P 2020)
[3] Direct Memory Attack the Kernel (DEFCON 2016; PCILeech)
[4] Handbook of Applied Cryptography (Menezes, Alfred J. et. al.)

# Secure Heterogeneous Processor

- Heterogeneous processor: SoC with CPU, GPU, NPU
- Data <u>confidentiality</u> & <u>integrity</u> are essential

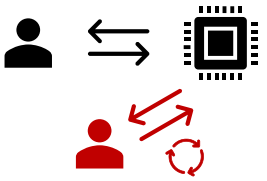



Cold Boot Attack [1]




Rowhammer Attack [2]

DMA Attack [3]

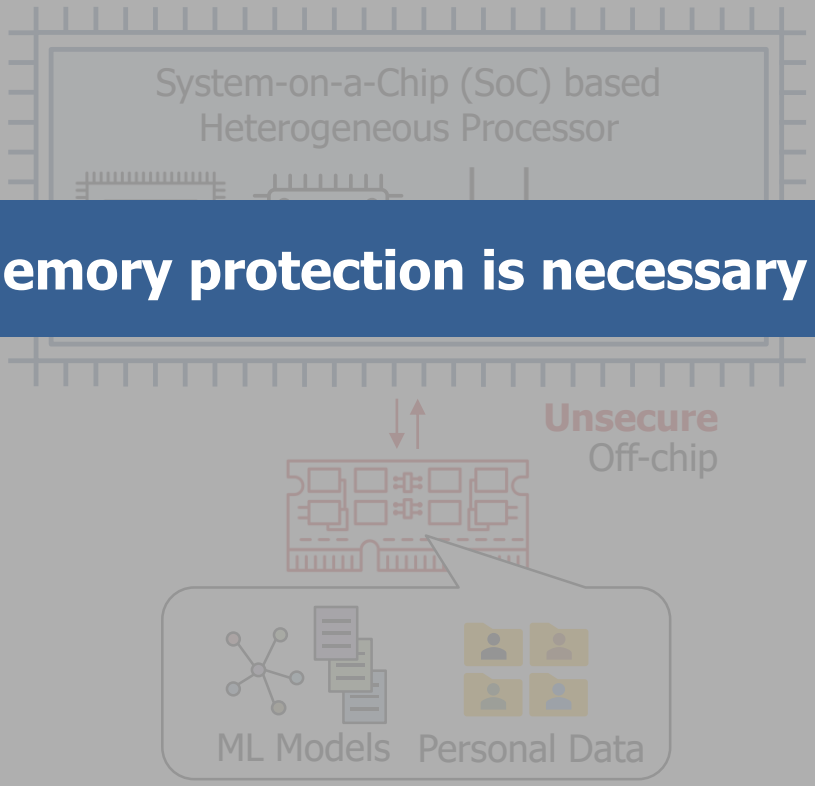


Replay Attack [4]

[1] Lest We Remember: Cold-Boot Attacks on Encryption Keys (USENIX Security 2008)
[2] RAMBleed: Reading Bits in Memory Without Accessing Them (S&P 2020)
[3] Direct Memory Attack the Kernel (DEFCON 2016; PCILeech)
[4] Handbook of Applied Cryptography (Menezes, Alfred J. et. al.)

# Secure Heterogeneous Processor

- Heterogeneous processor: SoC with CPU, GPU, NPU
- Data confidentiality & integrity are essential




**Memory protection is necessary for heterogeneous processors**


Cold Boot Attack [1]


Rowhammer Attack [2]

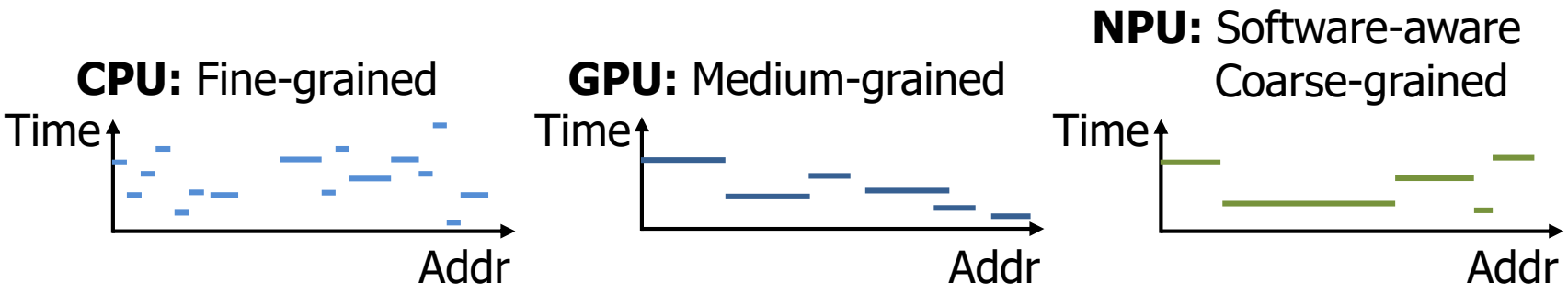

DMA Attack [3]


Replay Attack [4]

[1] Lest We Remember: Cold-Boot Attacks on Encryption Keys (USENIX Security 2008)
[2] RAMBleed: Reading Bits in Memory Without Accessing Them (S&P 2020)
[3] Direct Memory Attack the Kernel (DEFCON 2016; PCILeech)
[4] Handbook of Applied Cryptography (Menezes, Alfred J. et. al.)

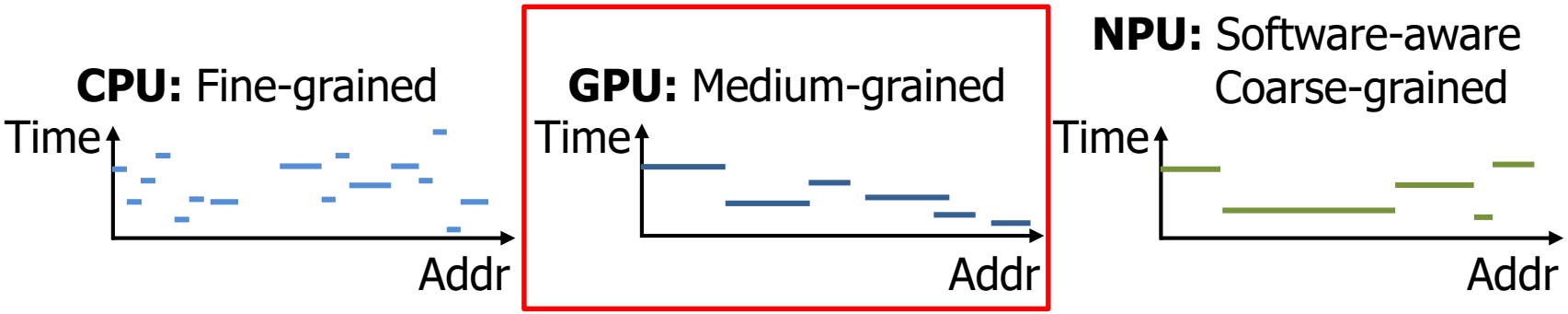
# Research Goal

- Existing memory protections are tailored to **specific, individual access patterns**



# Research Goal

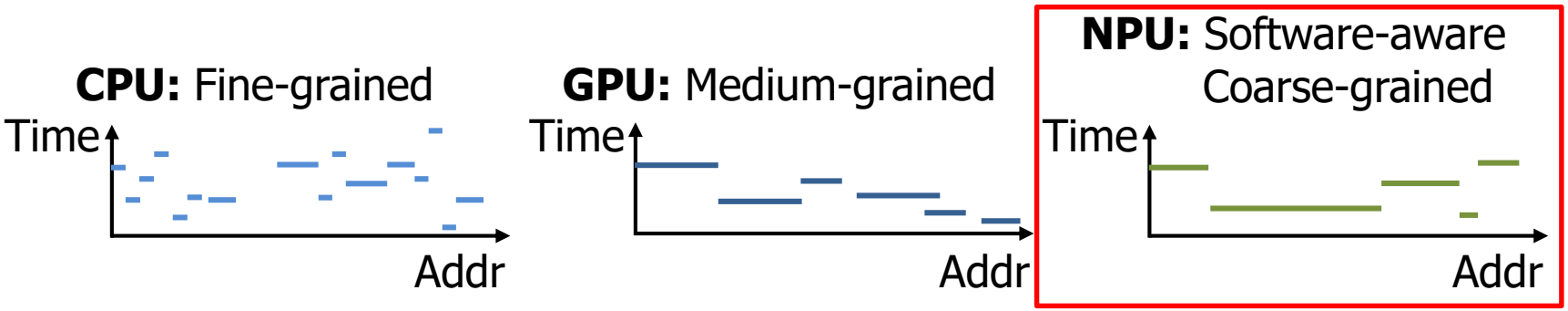
- Existing memory protections are tailored to **specific, individual access patterns**
  - Common Counters [1] → GPU medium-grained pattern




[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)

# Research Goal

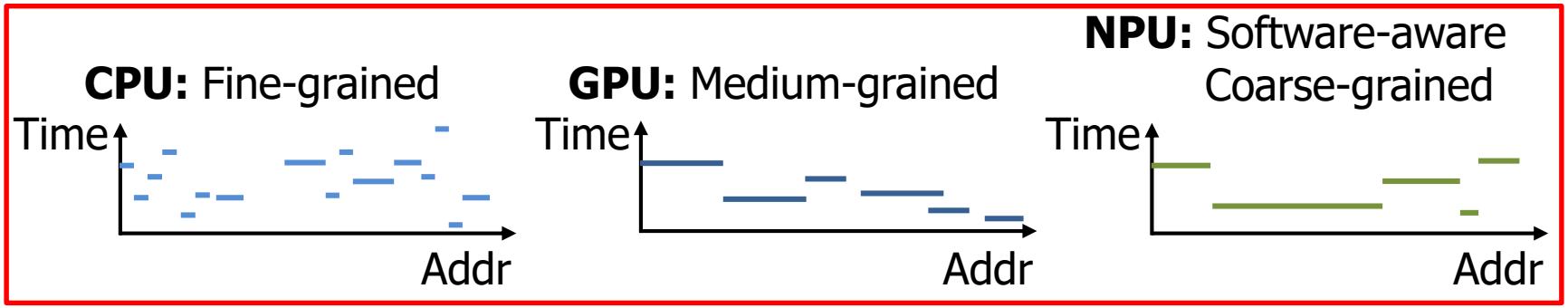
- Existing memory protections are tailored to **specific, individual access patterns**
  - Common Counters [1] → GPU medium-grained pattern
  - Studies of S/W-based counters→ NPU S/W-detected pattern




[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)

# Research Goal

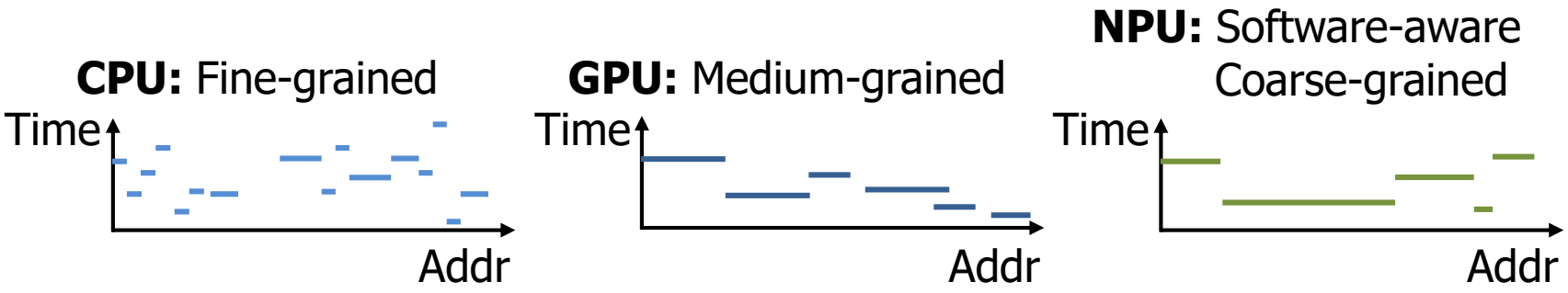
- Existing memory protections are tailored to **specific, individual access patterns**
  - Common Counters [1] → GPU medium-grained pattern
  - Studies of S/W-based counters→ NPU S/W-detected pattern
- Heterogeneous processor → **diverse access pattern**




[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)

# Research Goal

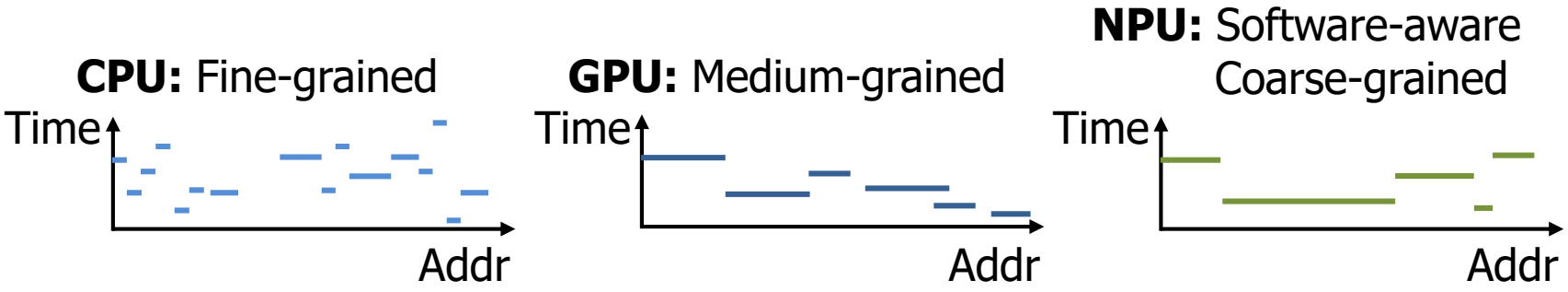
- Existing memory protections are tailored to **specific, individual access patterns**
  - Common Counters [1] → GPU medium-grained pattern
  - Studies of S/W-based counters→ NPU S/W-detected pattern
- Heterogeneous processor → **diverse access pattern**
  - A **unified** memory protection for all access patterns




[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)

# Research Goal

- Existing memory protections are tailored to **specific, individual access patterns**
  - Common Counters [1] → GPU medium-grained pattern
  - Studies of S/W-based counters→ NPU S/W-detected pattern
- Heterogeneous processor → **diverse access pattern**
  - A **unified** memory protection for all access patterns
  - Limitation of prior studies: Bypassing **integrity tree** optimization



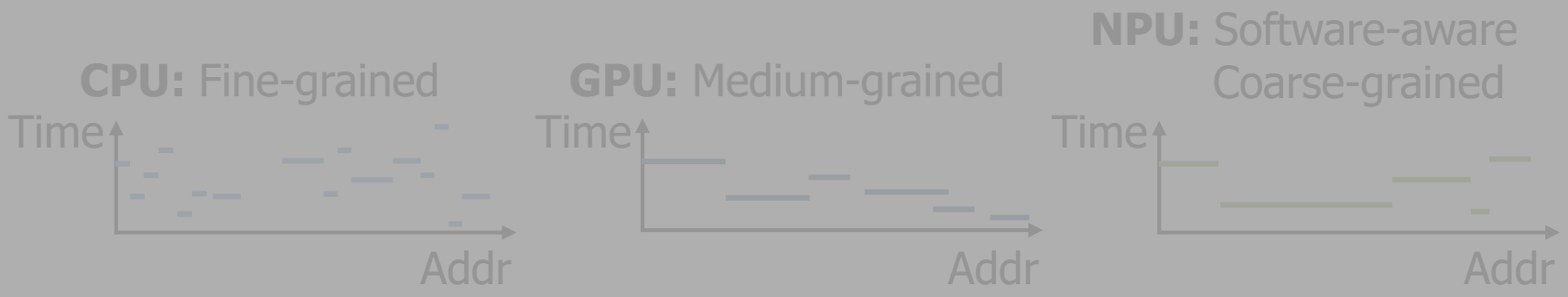

[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)

# Research Goal

- Existing memory protections are tailored to **specific, individual access patterns**
  - Common Counters [1] → GPU medium-grained pattern
  - Studies of S/W-based counters→ NPU S/W-detected pattern

**This study constructs a unified memory protection scheme with integrity tree optimization for heterogeneous processors**

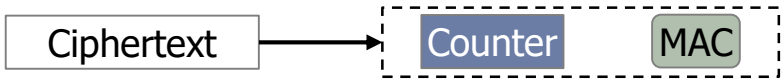
  - A **unified** memory protection for all access patterns
  - Limitation of prior studies: Bypassing **integrity tree** optimization



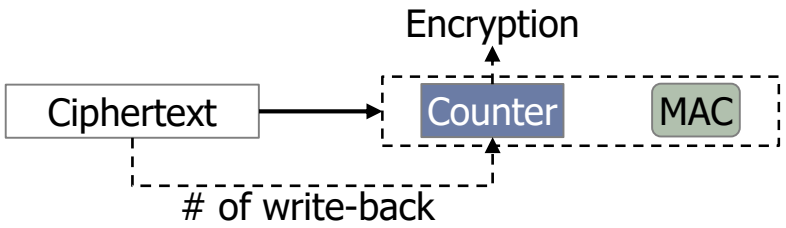

[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)

# Counter-mode Memory Protection

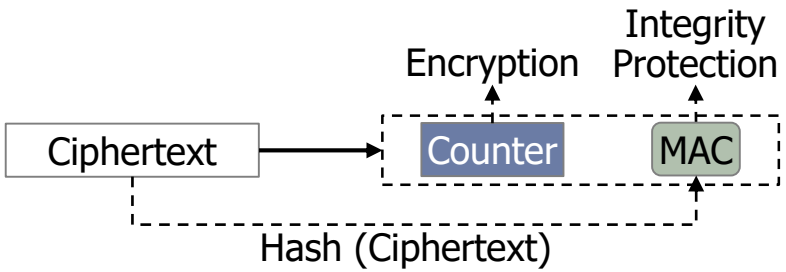
# Counter-mode Memory Protection



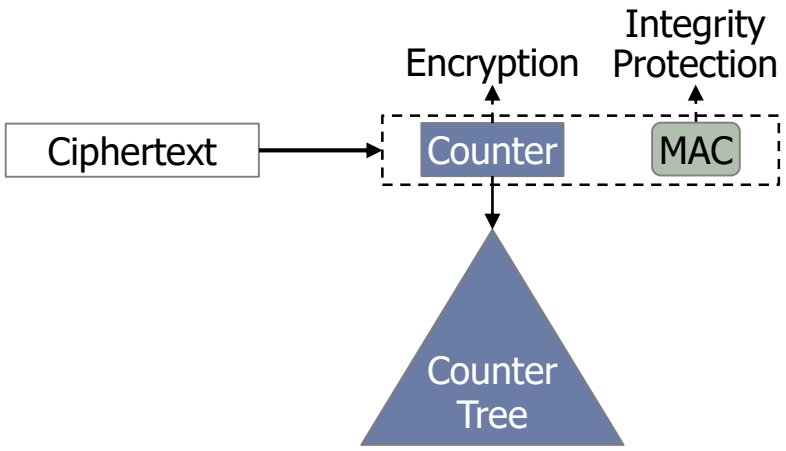
# Counter-mode Memory Protection



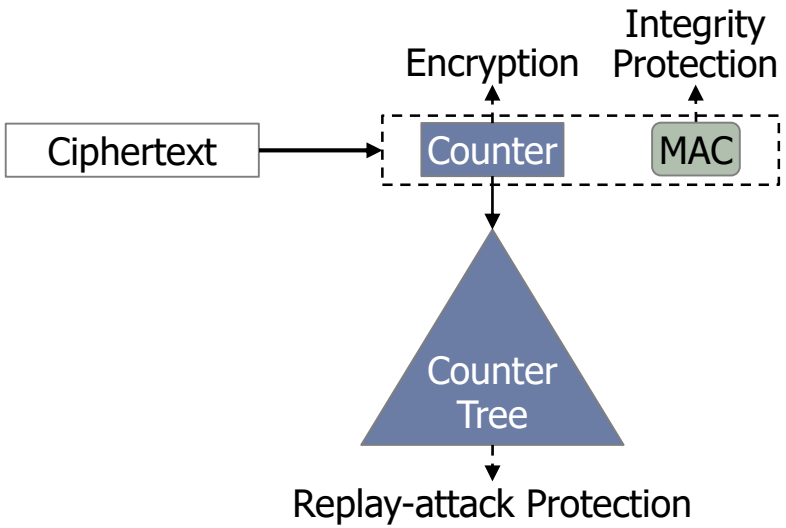
# Counter-mode Memory Protection



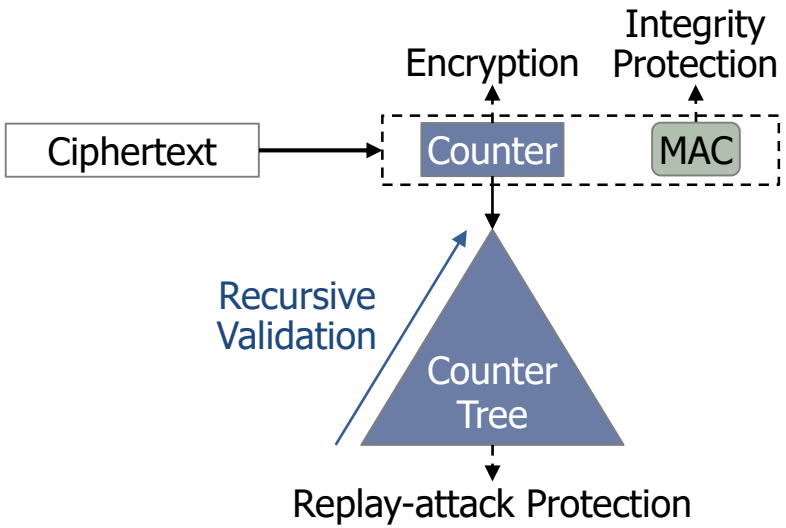
# Counter-mode Memory Protection



# Counter-mode Memory Protection

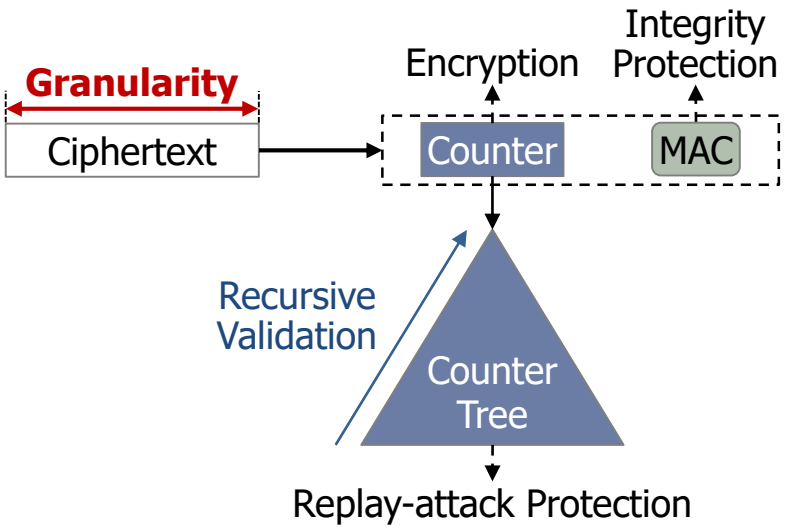


# Counter-mode Memory Protection



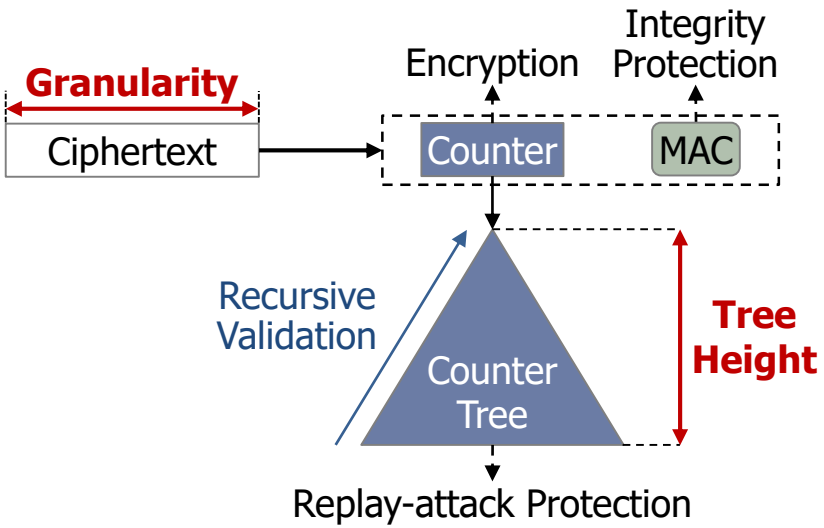
# Counter-mode Memory Protection

- Critical factors of memory protection
  - Amount of counters and MACs: Granularity



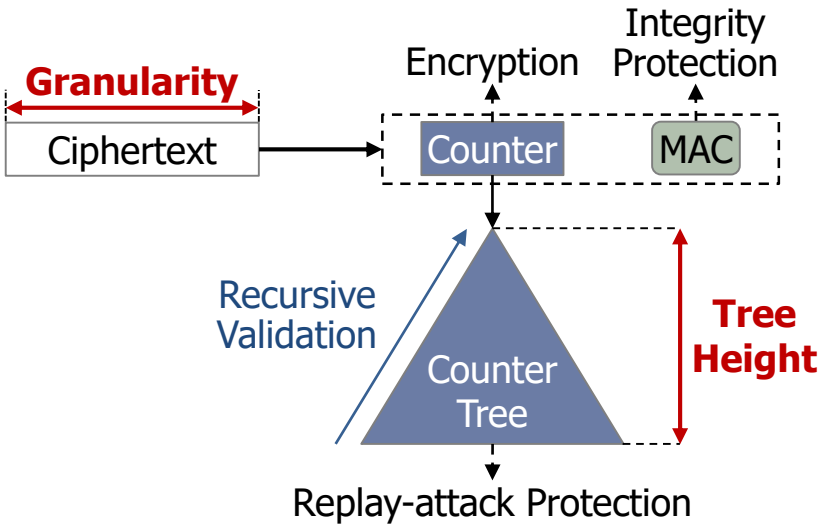
# Counter-mode Memory Protection

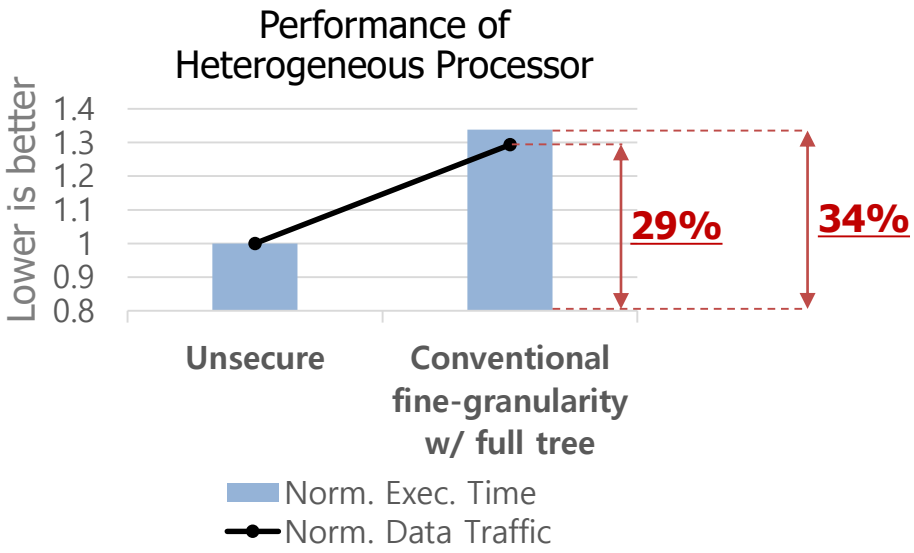
- Critical factors of memory protection
  - Amount of counters and MACs: Granularity
  - Overhead of recursive validation: Height of integrity tree



# Counter-mode Memory Protection

- Critical factors of memory protection
  - Amount of counters and MACs: Granularity
  - Overhead of recursive validation: Height of integrity tree
- 34% delay with 29% data traffic increment

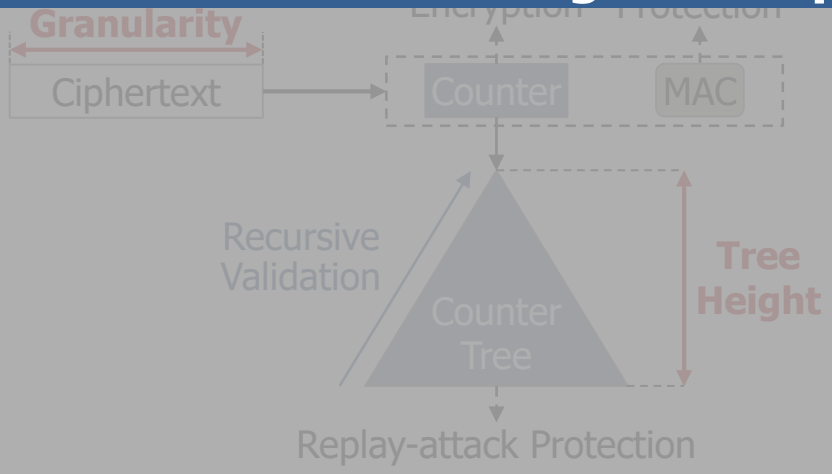


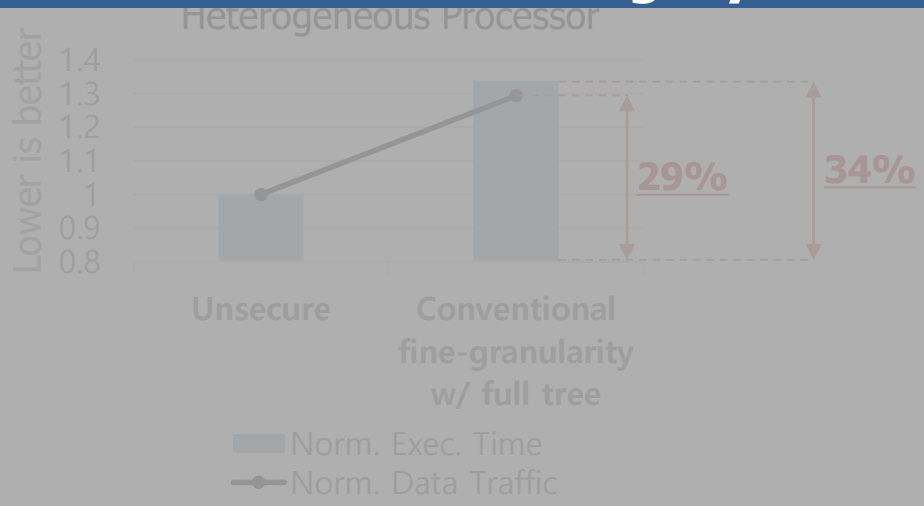



# Counter-mode Memory Protection

- Critical factors of memory protection
  - Amount of counters and MACs: Granularity
  - Overhead of recursive validation: Height of integrity tree
- 34% delay with 29% data traffic increment

**Significant overhead caused by the conventional 64B-granular protection with a full integrity tree**

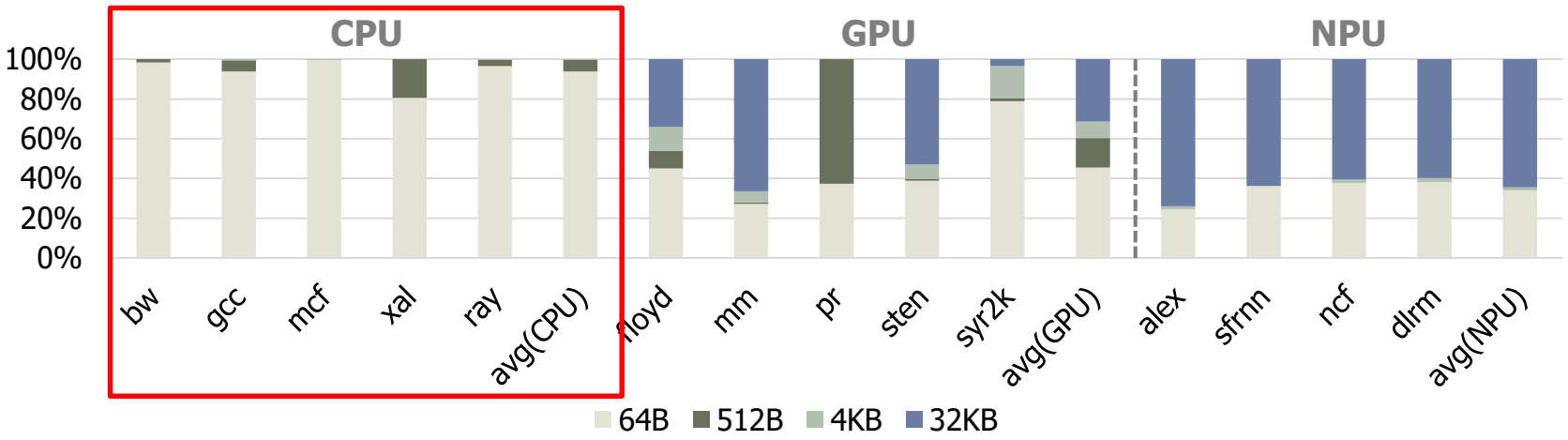


# Diverse Access Granularity

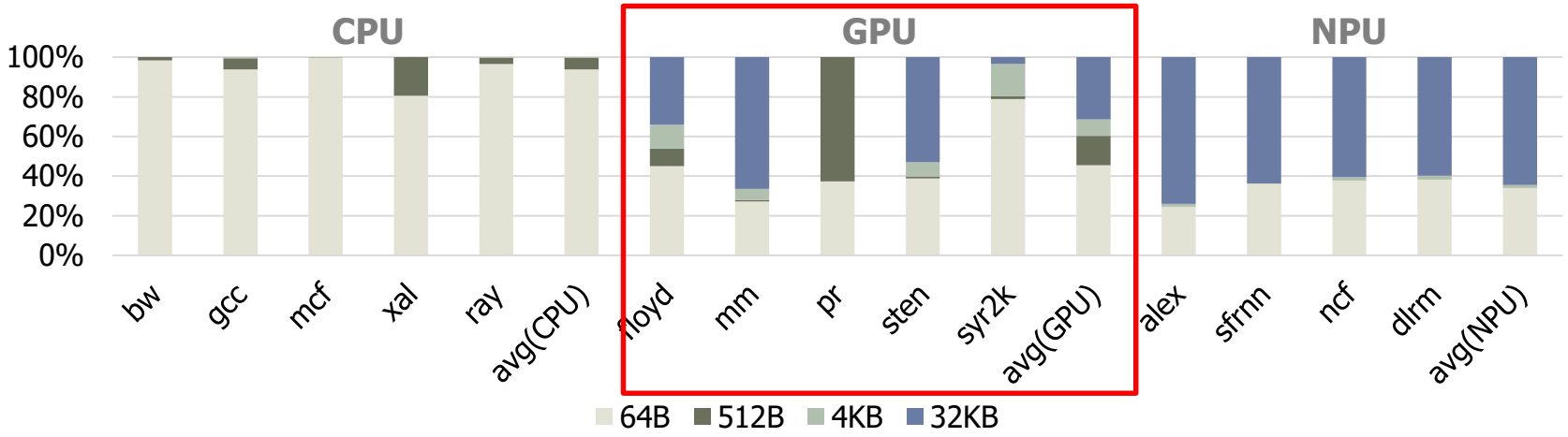
# Diverse Access Granularity

- Major access chunks (consecutive access blocks)
  - Fine-grained (64B): CPU



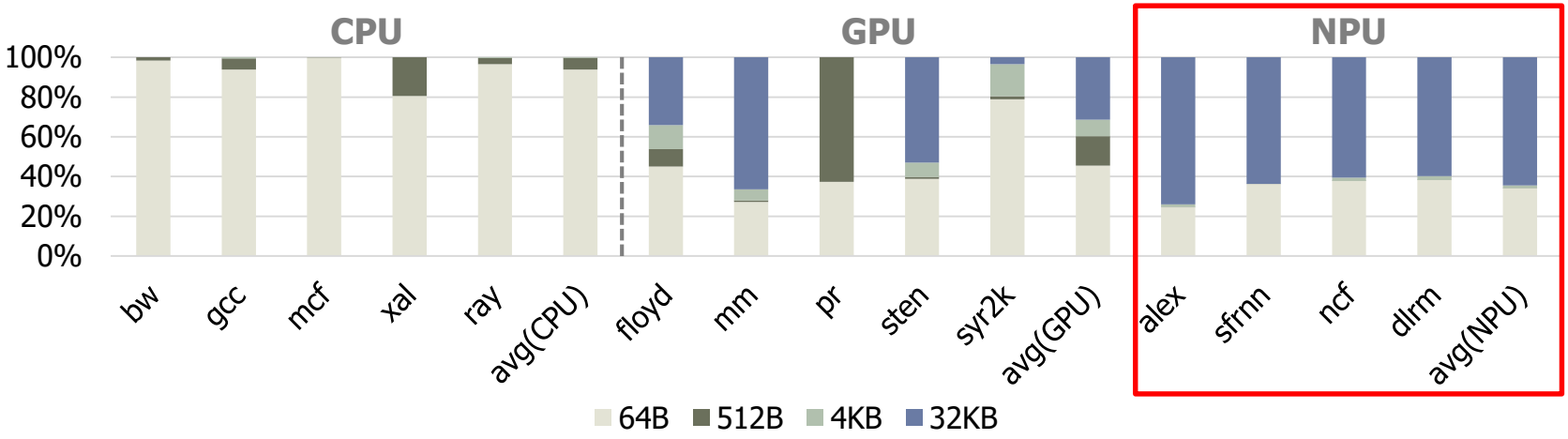
# Diverse Access Granularity

- Major access chunks (consecutive access blocks)
  - Fine-grained (64B): CPU
  - Medium-grained (512B, 4KB): GPU



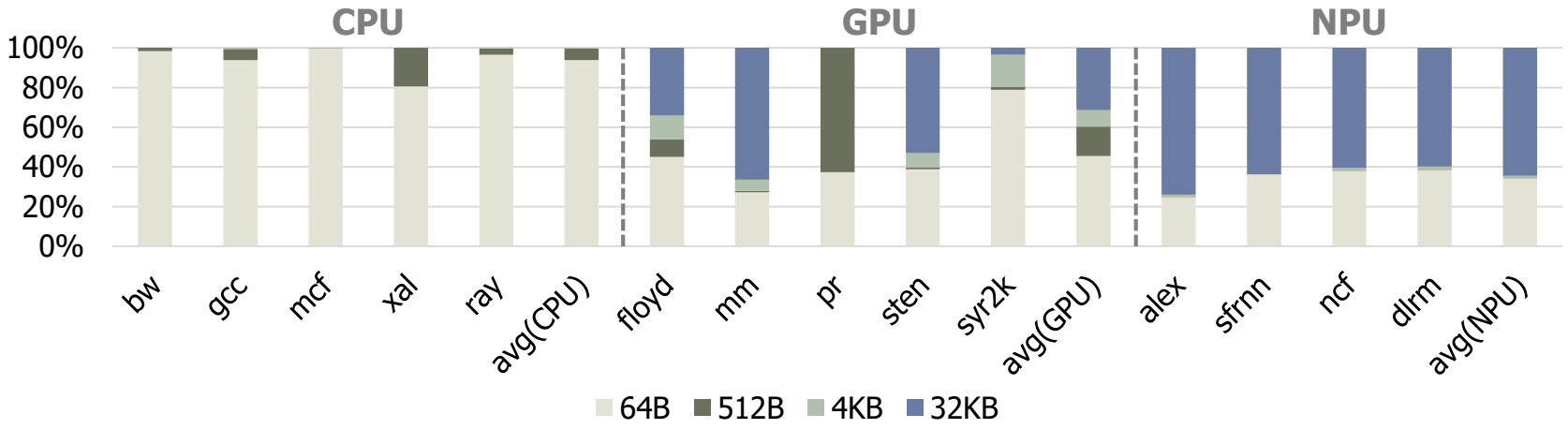
# Diverse Access Granularity

- Major access chunks (consecutive access blocks)
  - Fine-grained (64B): CPU
  - Medium-grained (512B, 4KB): GPU
  - Coarse-grained (32KB): NPU



# Diverse Access Granularity

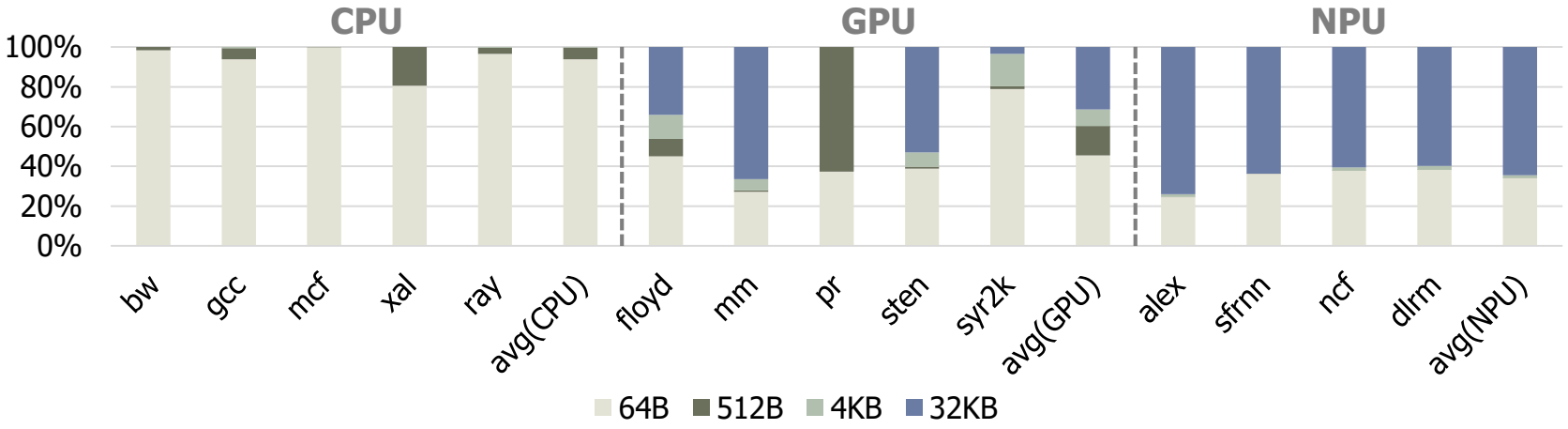
- Major access chunks (consecutive access blocks)
  - Fine-grained (64B): CPU
  - Medium-grained (512B, 4KB): GPU
  - Coarse-grained (32KB): NPU




**Matching security granularity to access granularity**

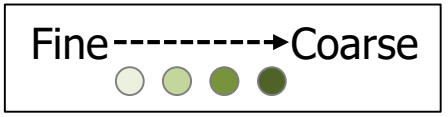
# Diverse Access Granularity

- Major access chunks (consecutive access blocks)
  - Fine-grained (64B): CPU
  - Medium-grained (512B, 4KB): GPU
  - Coarse-grained (32KB): NPU

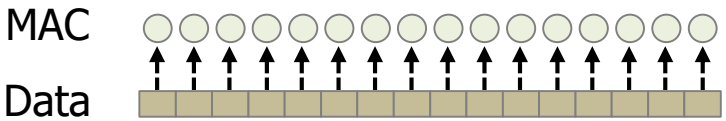



**Matching security granularity to access granularity**
**→ Requirement: Multi-granularity for MACs and counters**

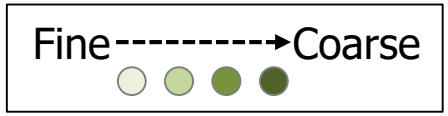
# Multi-granular MAC



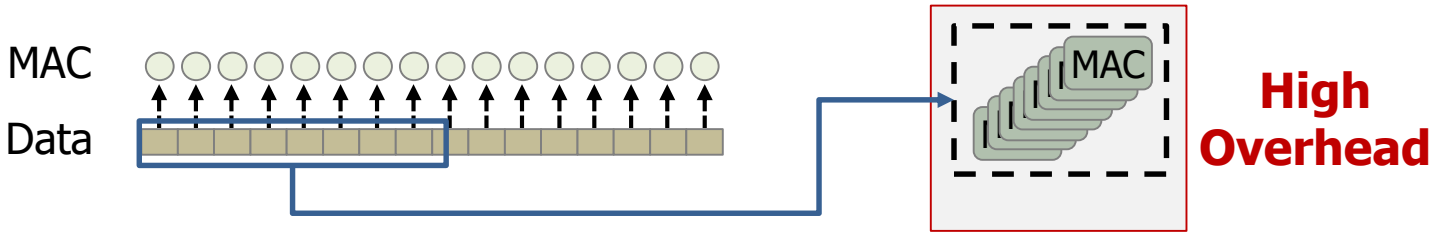

**Conventional Fine-granular MAC**



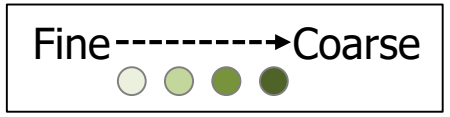
# Multi-granular MAC



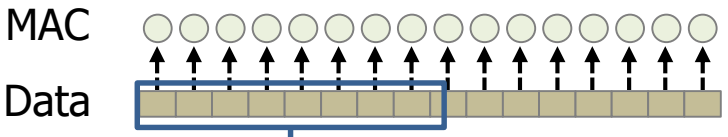

**Conventional Fine-granular MAC**

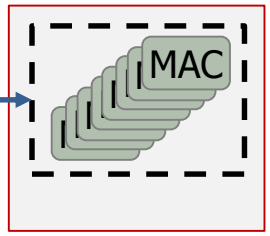


# Multi-granular MAC



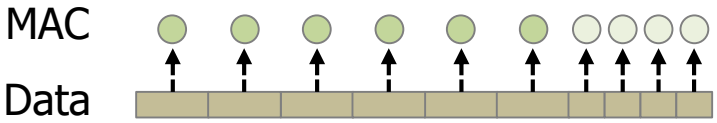

**Conventional Fine-granular MAC**



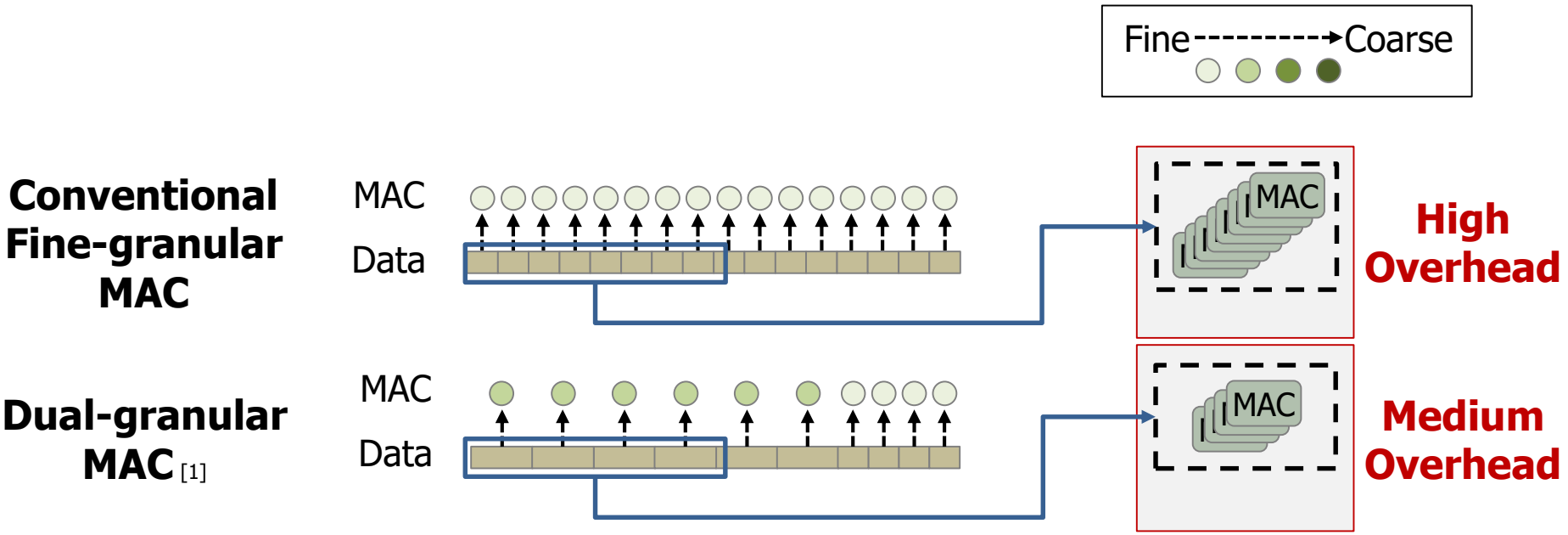

**High Overhead**

**Dual-granular MAC** [1]




[1] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)

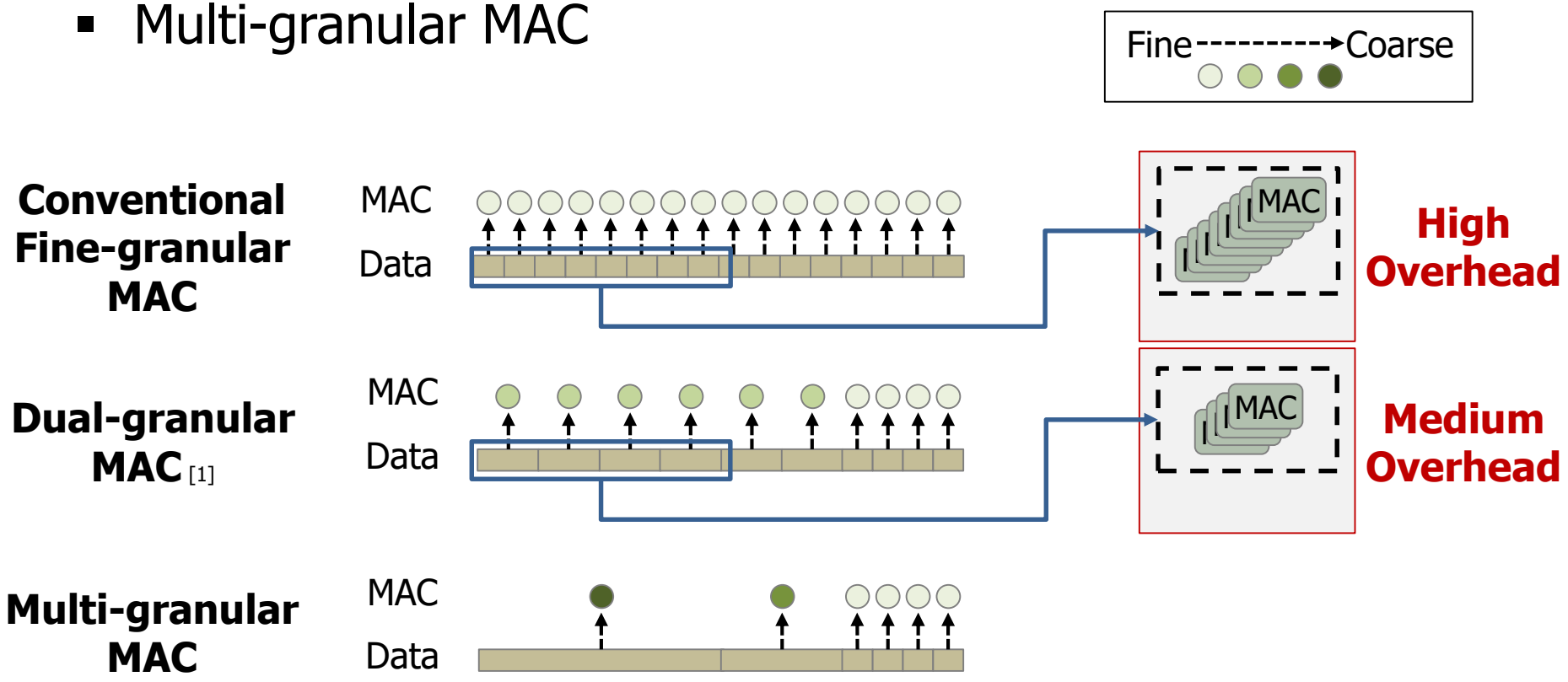
# Multi-granular MAC




[1] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)

# Multi-granular MAC

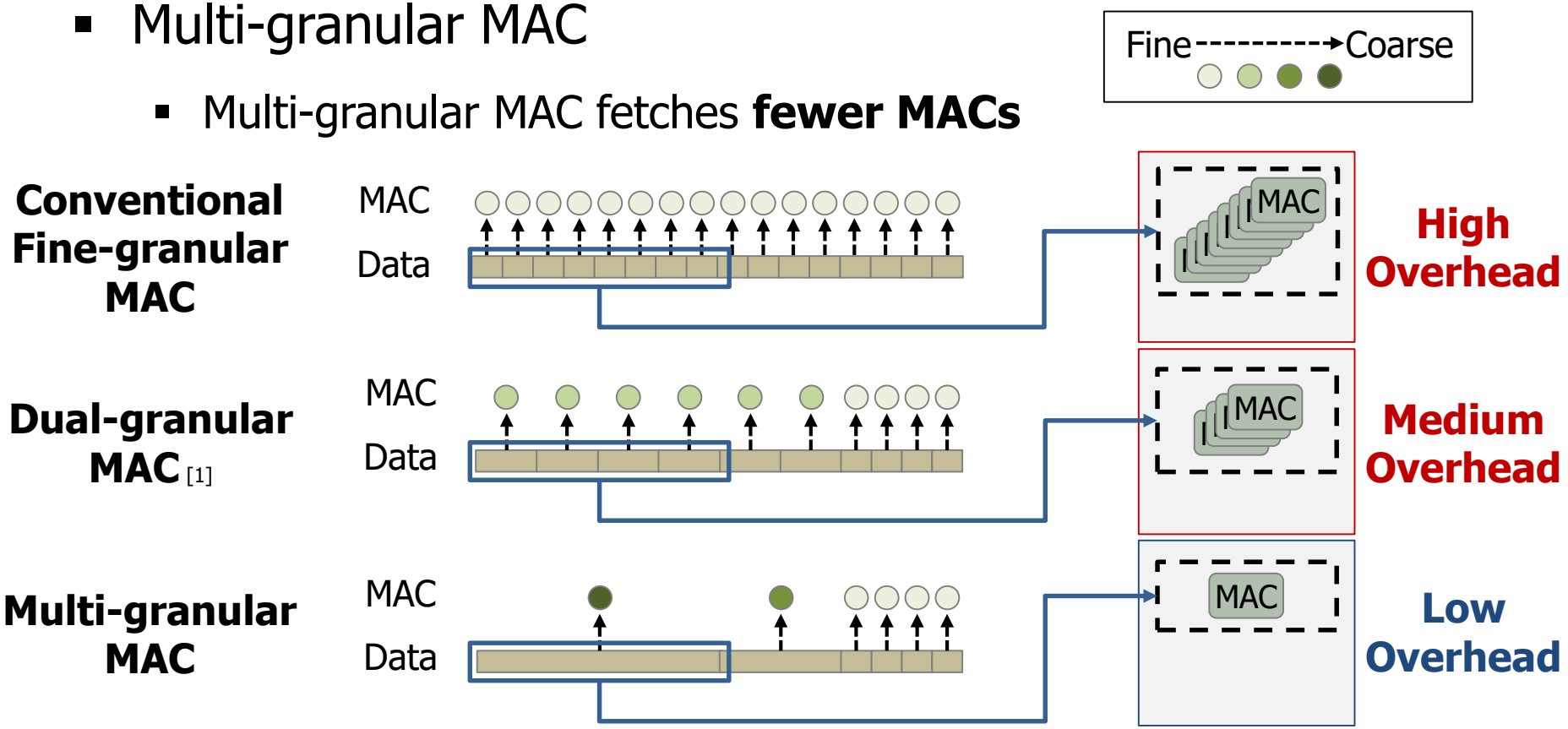
- Multi-granular MAC




[1] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)

# Multi-granular MAC

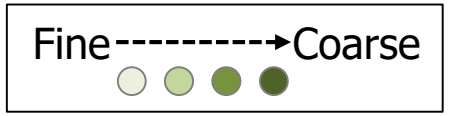
- Multi-granular MAC
  - Multi-granular MAC fetches **fewer MACs**

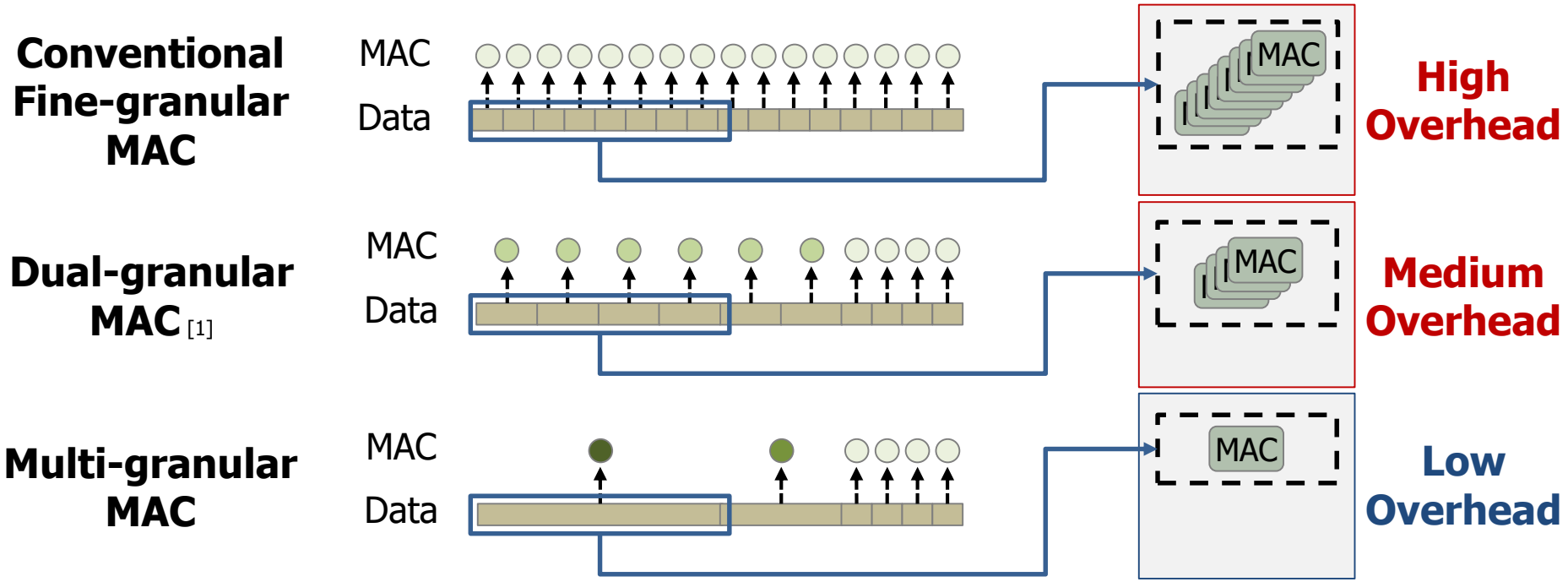



[1] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)

# Multi-granular MAC

- Multi-granular MAC
  - Multi-granular MAC fetches **fewer MACs**



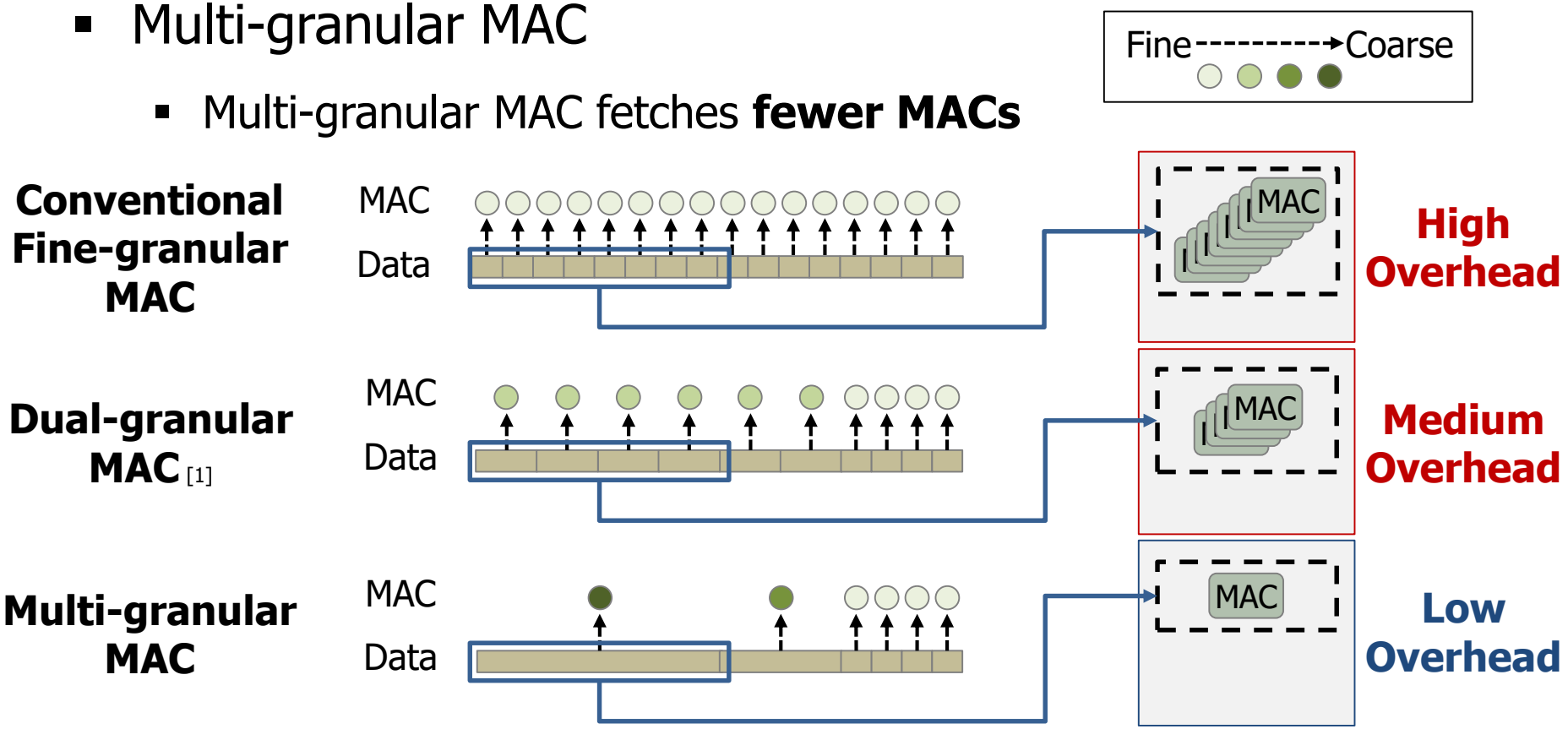




**Multi-granular MAC → Only managing the security granularity**

[1] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)

# Multi-granular MAC

- Multi-granular MAC
  - Multi-granular MAC fetches **fewer MACs**




**Multi-granular MAC → Only managing the security granularity**
**What about multi-granular counters?**

[1] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)

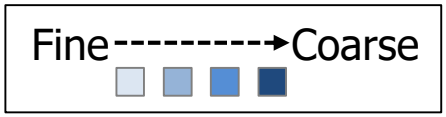
# Prior Multi-granular Counter

# Prior Multi-granular Counter

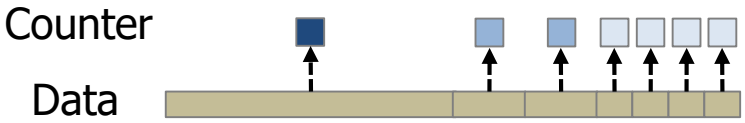
- Due to **integrity tree**, prior studies bypass integrity tree under specific conditions.

# Prior Multi-granular Counter

- Due to **integrity tree**, prior studies bypass integrity tree under specific conditions.



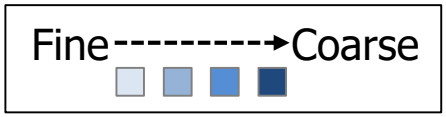

**Common Counters** [1]



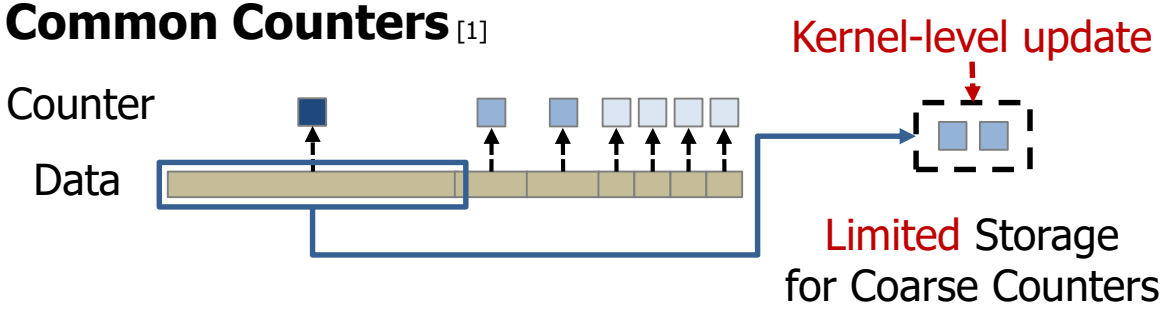

[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)

# Prior Multi-granular Counter

- Due to **integrity tree**, prior studies bypass integrity tree under specific conditions.



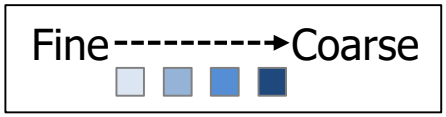

**Common Counters** [1]



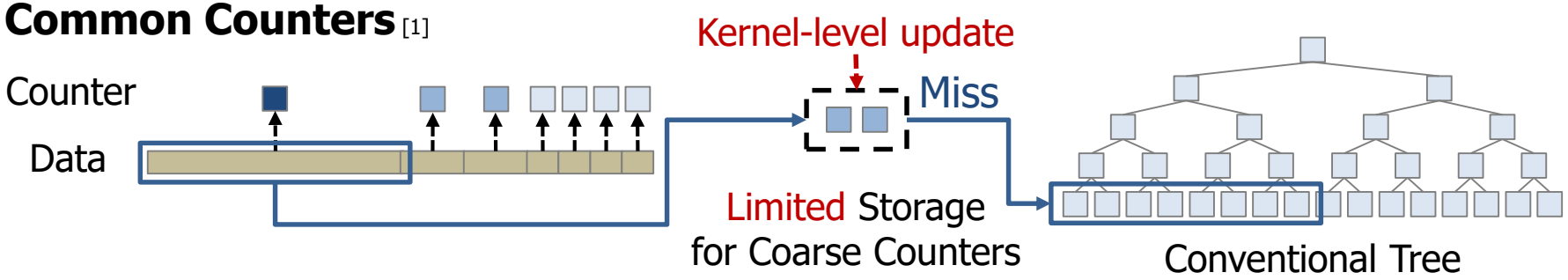

[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)

# Prior Multi-granular Counter

- Due to **integrity tree**, prior studies bypass integrity tree under specific conditions.



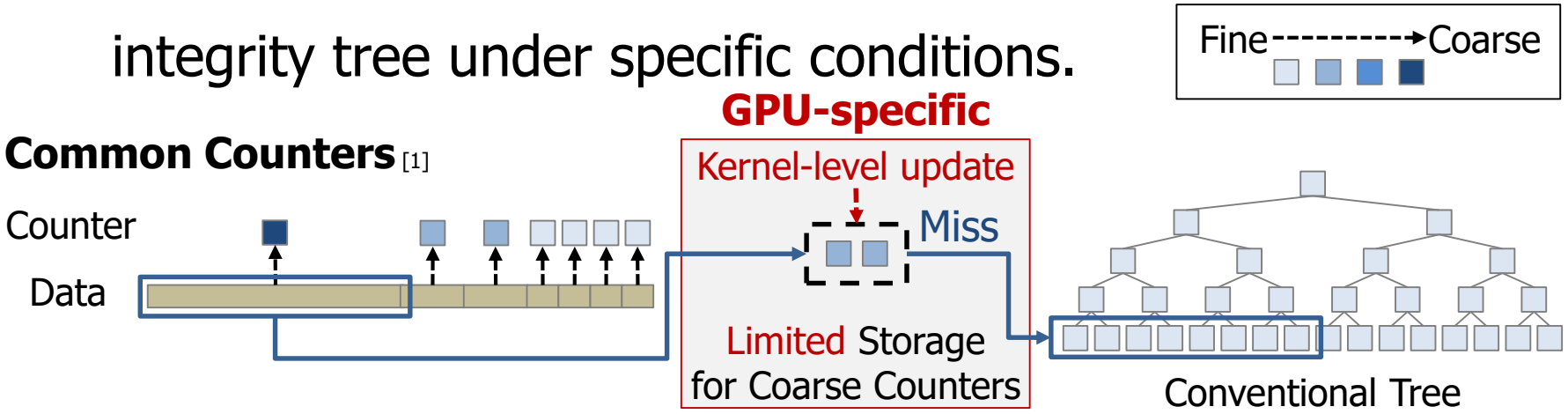

**Common Counters** [1]




[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)

# Prior Multi-granular Counter

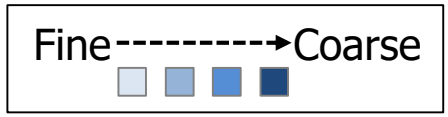
- Due to **integrity tree**, prior studies bypass integrity tree under specific conditions.




[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)

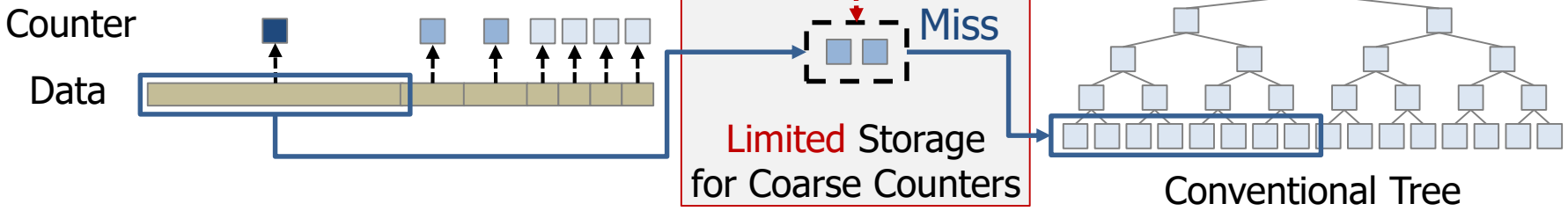
# Prior Multi-granular Counter

- Due to **integrity tree**, prior studies bypass integrity tree under specific conditions.

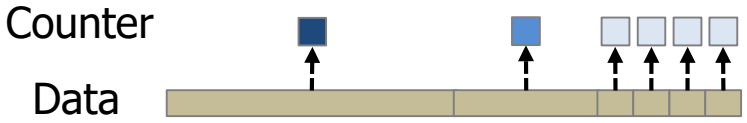



**GPU-specific**

**Common Counters** [1]



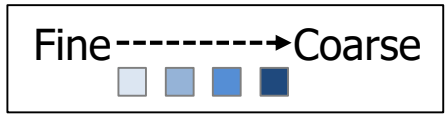

**S/W-managed Studies** [2-5]




[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] TNPU: Supporting Trusted Execution with Tree-less Integrity Protection for Neural Processing Unit (HPCA 2022)
[3] MGX: Near-zero Overhead Memory Protection for Data-intensive Accelerators (ISCA 2022)
[4] GuardNN: Secure Accelerator Architecture for Privacy-preserving Deep Learning (DAC 2022)
[5] SoftVN: Efficient Memory Protection via Software-provided Version Numbers (ISCA 2022)

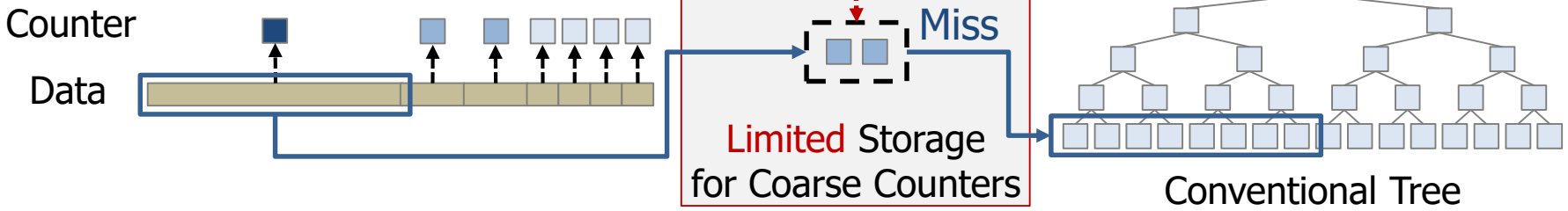
# Prior Multi-granular Counter

- Due to **integrity tree**, prior studies bypass integrity tree under specific conditions.

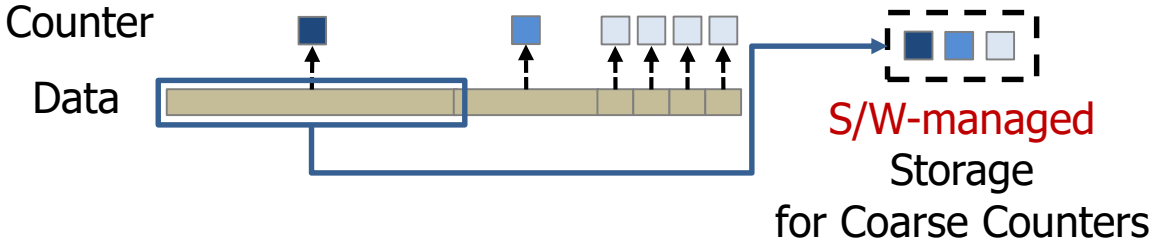



**GPU-specific**

**Common Counters** [1]



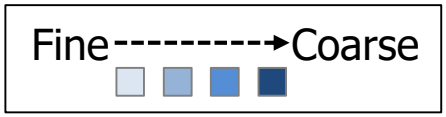

**S/W-managed Studies** [2-5]




[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] TNPU: Supporting Trusted Execution with Tree-less Integrity Protection for Neural Processing Unit (HPCA 2022)
[3] MGX: Near-zero Overhead Memory Protection for Data-intensive Accelerators (ISCA 2022)
[4] GuardNN: Secure Accelerator Architecture for Privacy-preserving Deep Learning (DAC 2022)
[5] SoftVN: Efficient Memory Protection via Software-provided Version Numbers (ISCA 2022)

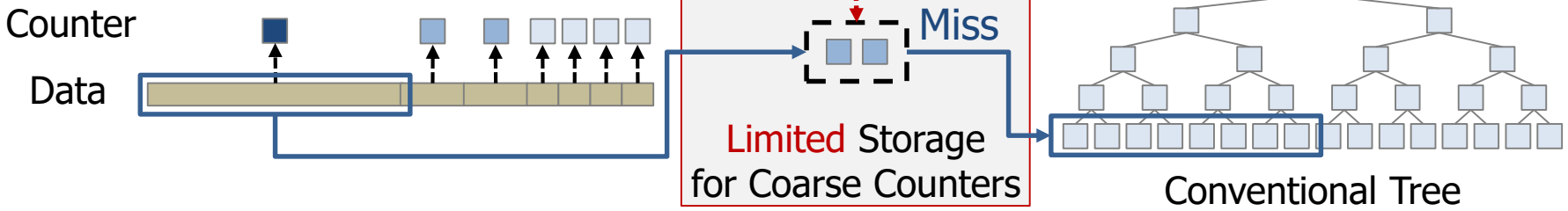
# Prior Multi-granular Counter

- Due to **integrity tree**, prior studies bypass integrity tree under specific conditions.

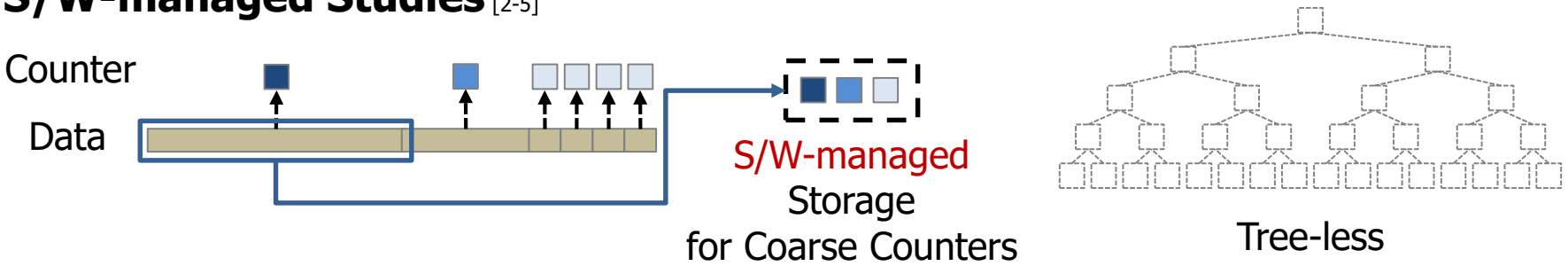



**GPU-specific**

**Common Counters** [1]



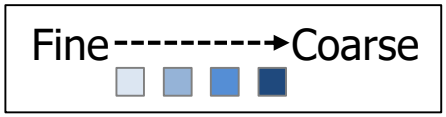

**S/W-managed Studies** [2-5]




[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] TNPU: Supporting Trusted Execution with Tree-less Integrity Protection for Neural Processing Unit (HPCA 2022)
[3] MGX: Near-zero Overhead Memory Protection for Data-intensive Accelerators (ISCA 2022)
[4] GuardNN: Secure Accelerator Architecture for Privacy-preserving Deep Learning (DAC 2022)
[5] SoftVN: Efficient Memory Protection via Software-provided Version Numbers (ISCA 2022)

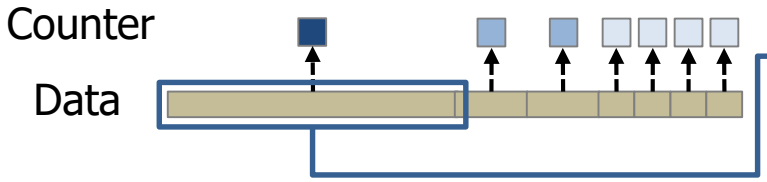
# Prior Multi-granular Counter

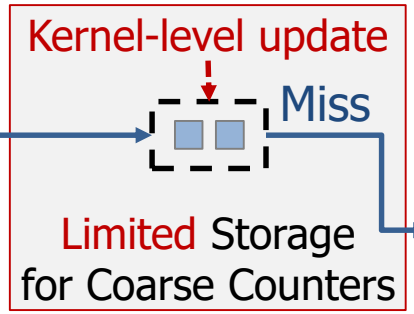
- Due to **integrity tree**, prior studies bypass integrity tree under specific conditions.

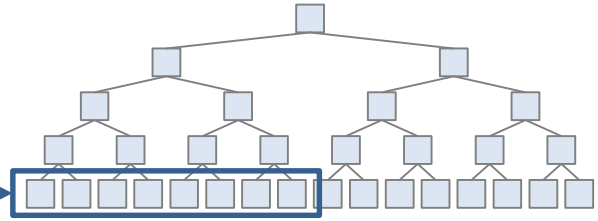



**GPU-specific**

**Common Counters** [1]



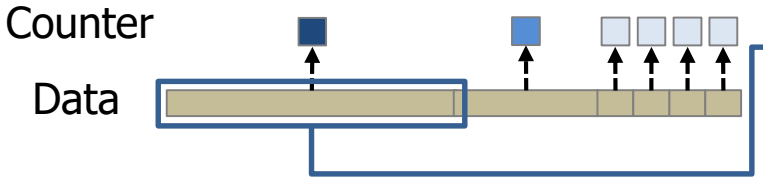



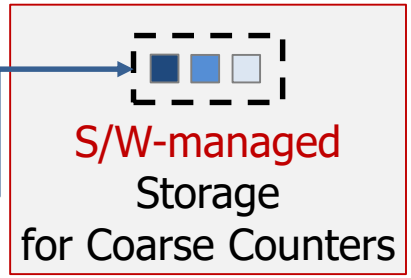



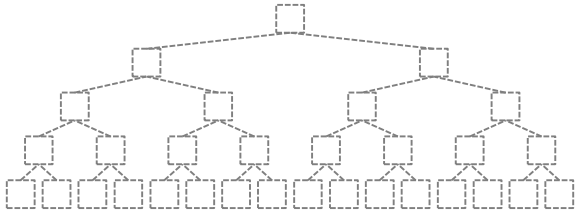
Conventional Tree

**S/W-managed Studies** [2-5]

**ML-specific**



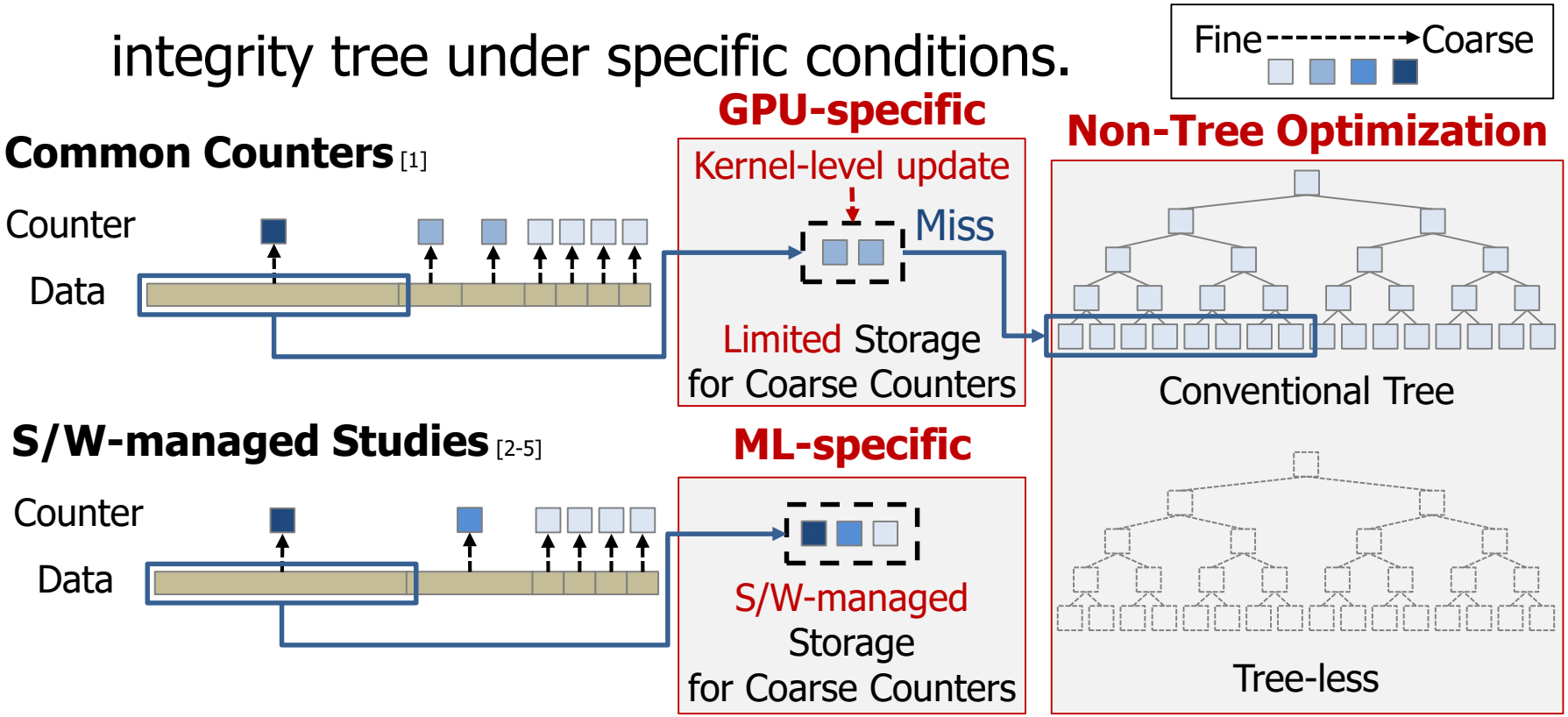



Tree-less

[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] TNPU: Supporting Trusted Execution with Tree-less Integrity Protection for Neural Processing Unit (HPCA 2022)
[3] MGX: Near-zero Overhead Memory Protection for Data-intensive Accelerators (ISCA 2022)
[4] GuardNN: Secure Accelerator Architecture for Privacy-preserving Deep Learning (DAC 2022)
[5] SoftVN: Efficient Memory Protection via Software-provided Version Numbers (ISCA 2022)

# Prior Multi-granular Counter

- Due to **integrity tree**, prior studies bypass integrity tree under specific conditions.

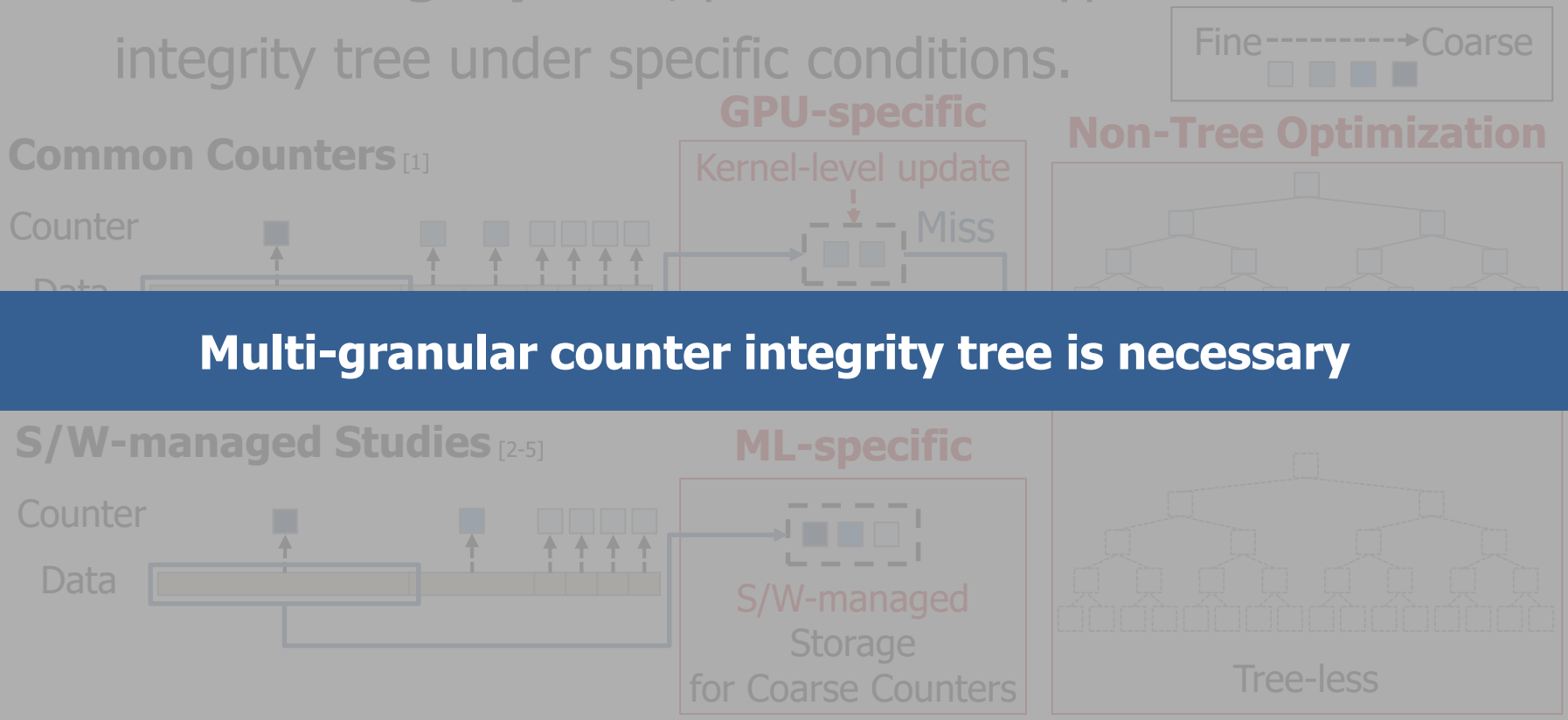




[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] TNPU: Supporting Trusted Execution with Tree-less Integrity Protection for Neural Processing Unit (HPCA 2022)
[3] MGX: Near-zero Overhead Memory Protection for Data-intensive Accelerators (ISCA 2022)
[4] GuardNN: Secure Accelerator Architecture for Privacy-preserving Deep Learning (DAC 2022)
[5] SoftVN: Efficient Memory Protection via Software-provided Version Numbers (ISCA 2022)

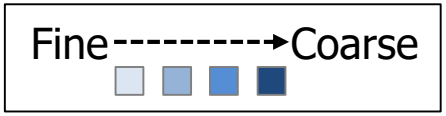
# Prior Multi-granular Counter

- Due to **integrity tree**, prior studies bypass integrity tree under specific conditions.

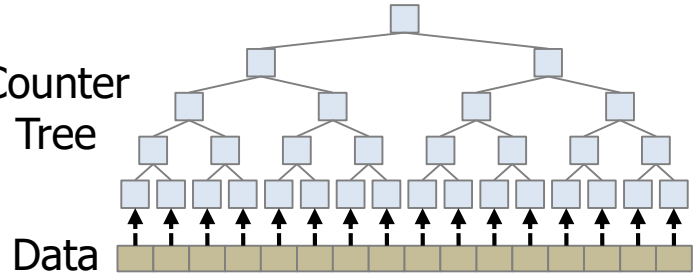



[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] TNPU: Supporting Trusted Execution with Tree-less Integrity Protection for Neural Processing Unit (HPCA 2022)
[3] MGX: Near-zero Overhead Memory Protection for Data-intensive Accelerators (ISCA 2022)
[4] GuardNN: Secure Accelerator Architecture for Privacy-preserving Deep Learning (DAC 2022)
[5] SoftVN: Efficient Memory Protection via Software-provided Version Numbers (ISCA 2022)

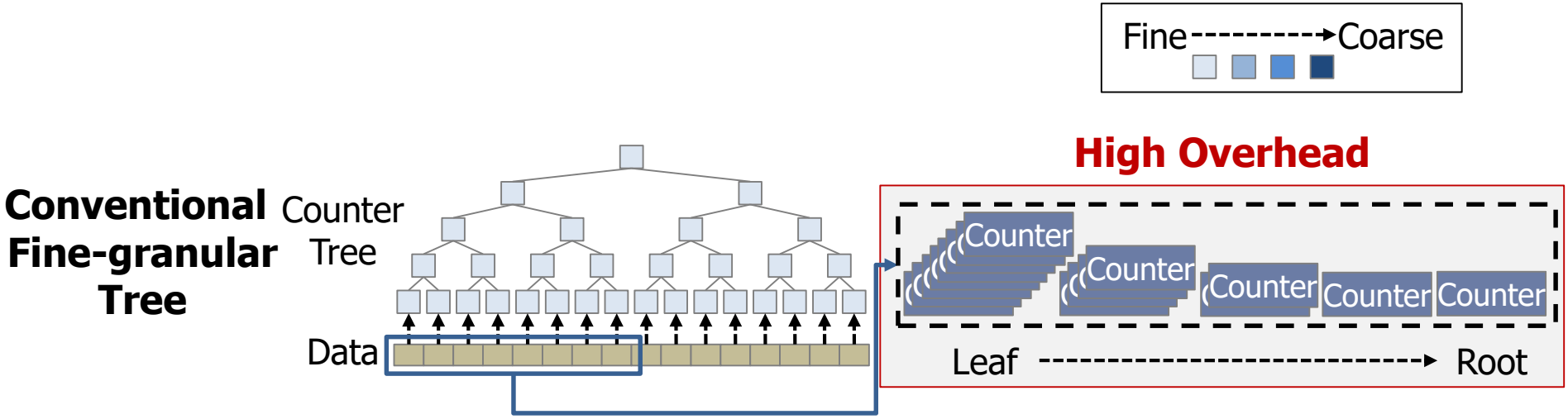
# Multi-granular Counter Integrity Tree



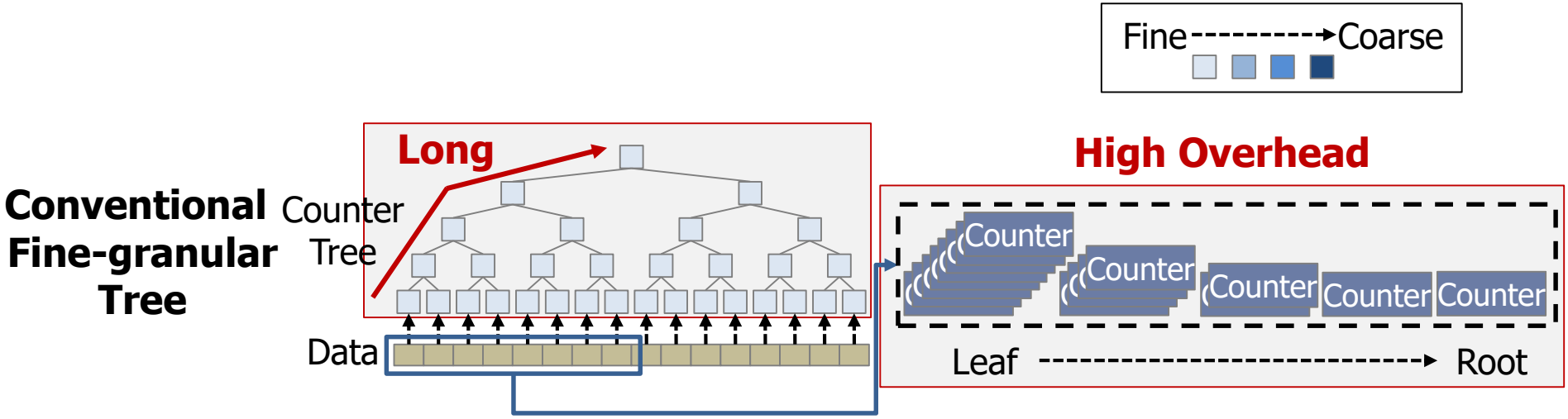

**Conventional Fine-granular Tree**



# Multi-granular Counter Integrity Tree

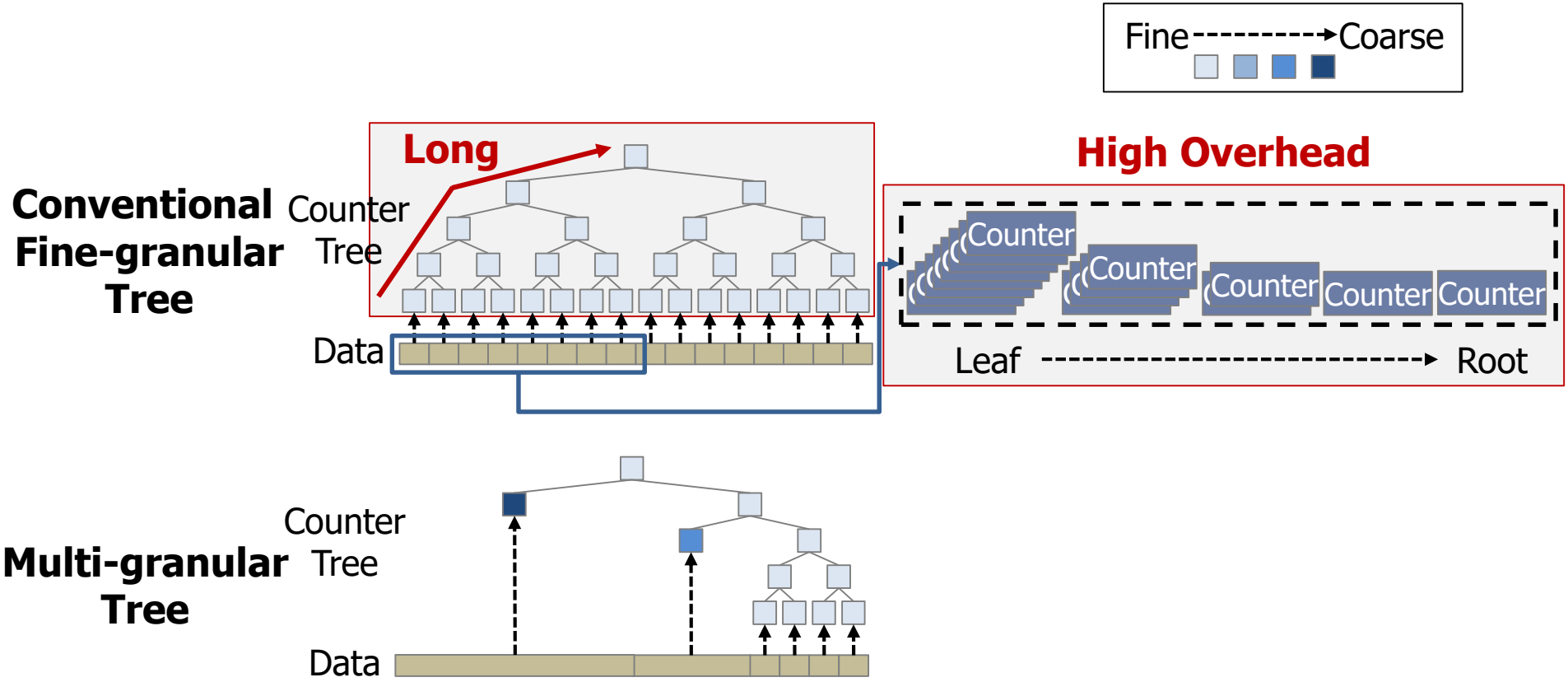


# Multi-granular Counter Integrity Tree



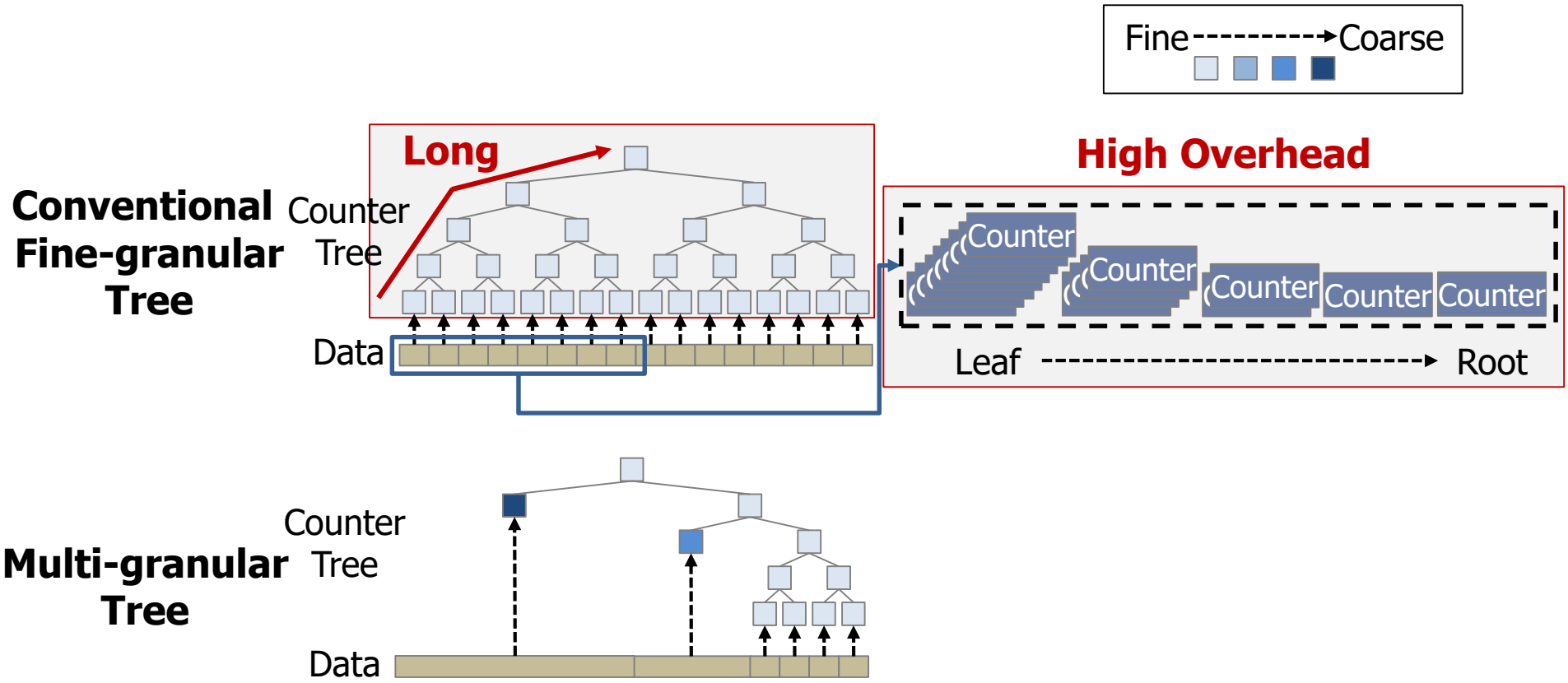
# Multi-granular Counter Integrity Tree

- Multi-granular tree



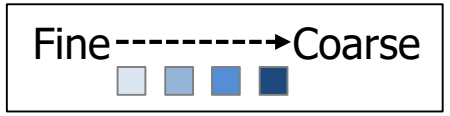
# Multi-granular Counter Integrity Tree

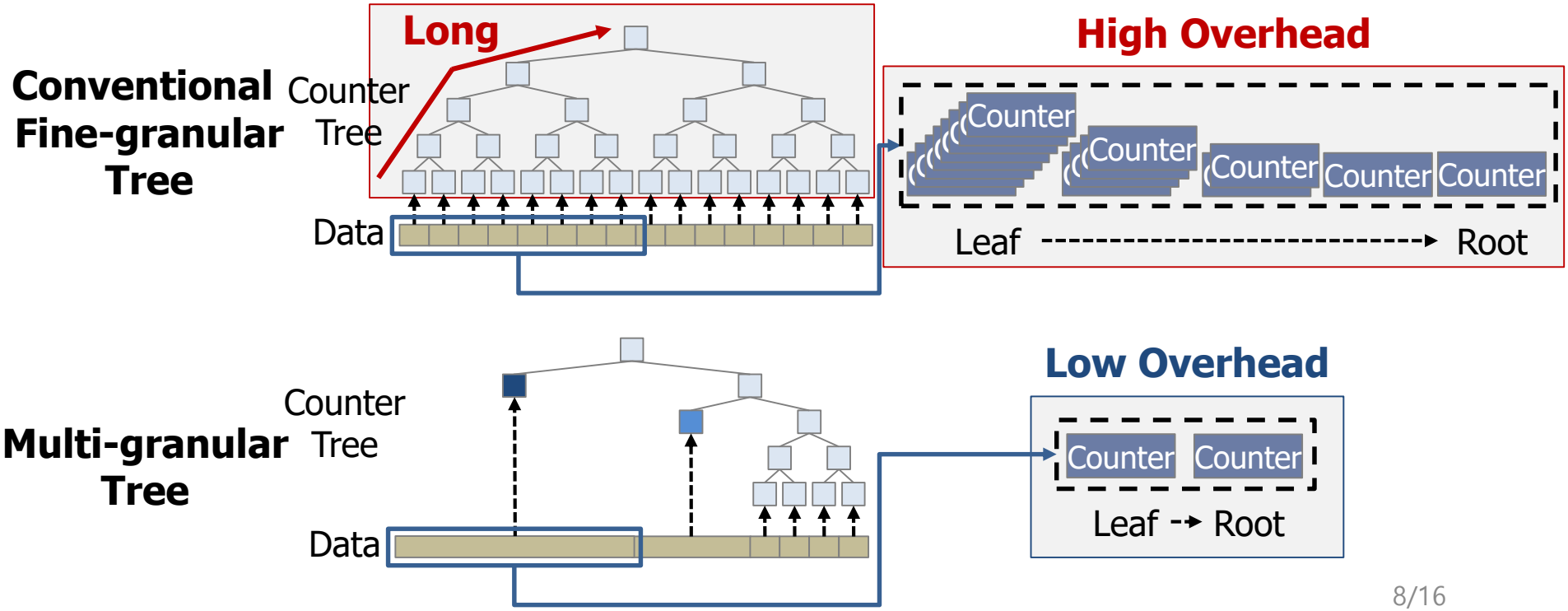
- Multi-granular tree
  - Counters w/ varying granularities are mapped to different levels



# Multi-granular Counter Integrity Tree

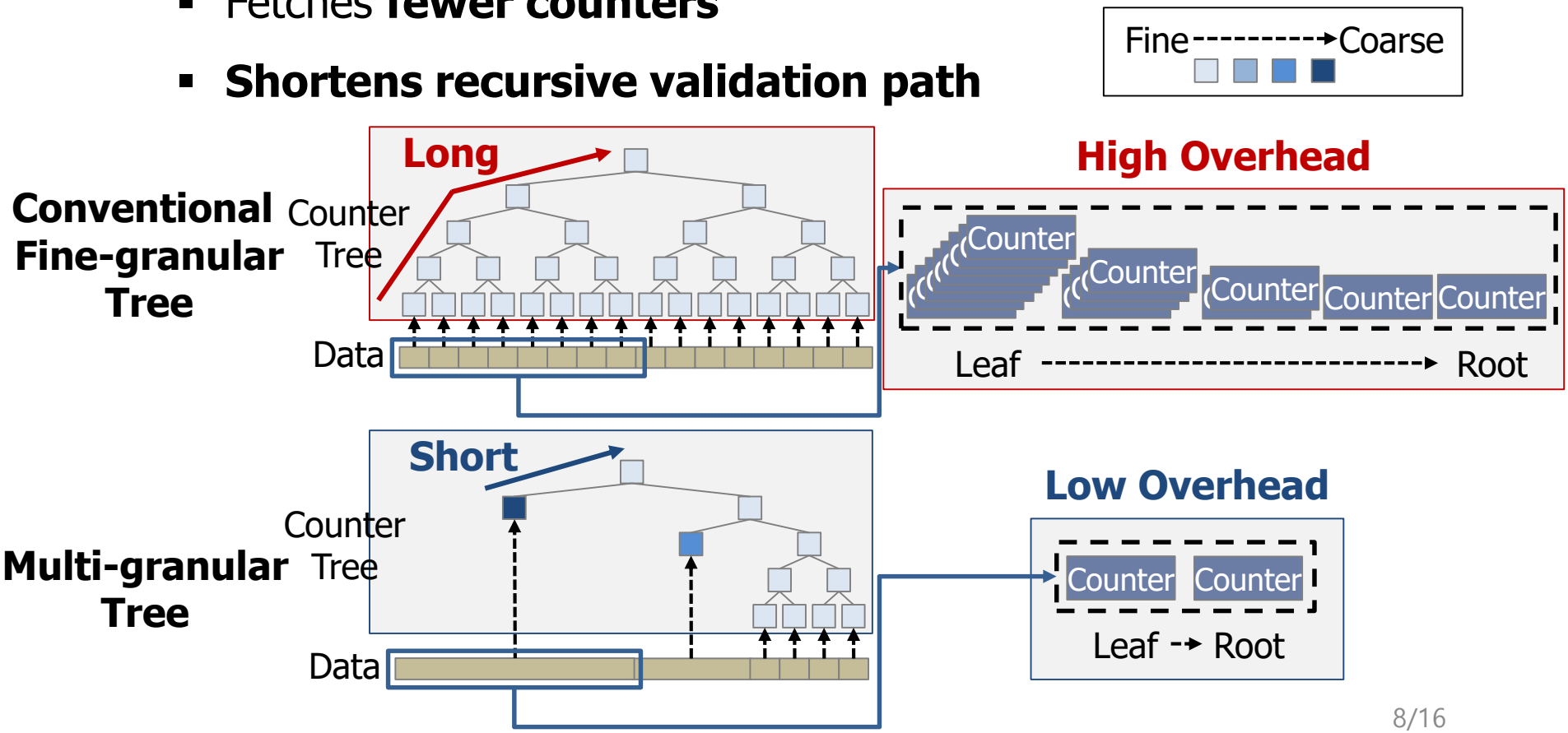
- Multi-granular tree
  - Counters w/ varying granularities are mapped to different levels
  - Fetches **fewer counters**



# Multi-granular Counter Integrity Tree

- Multi-granular tree
  - Counters w/ varying granularities are mapped to different levels
  - Fetches **fewer counters**
  - **Shortens recursive validation path**



# Multi-granular MAC&Tree

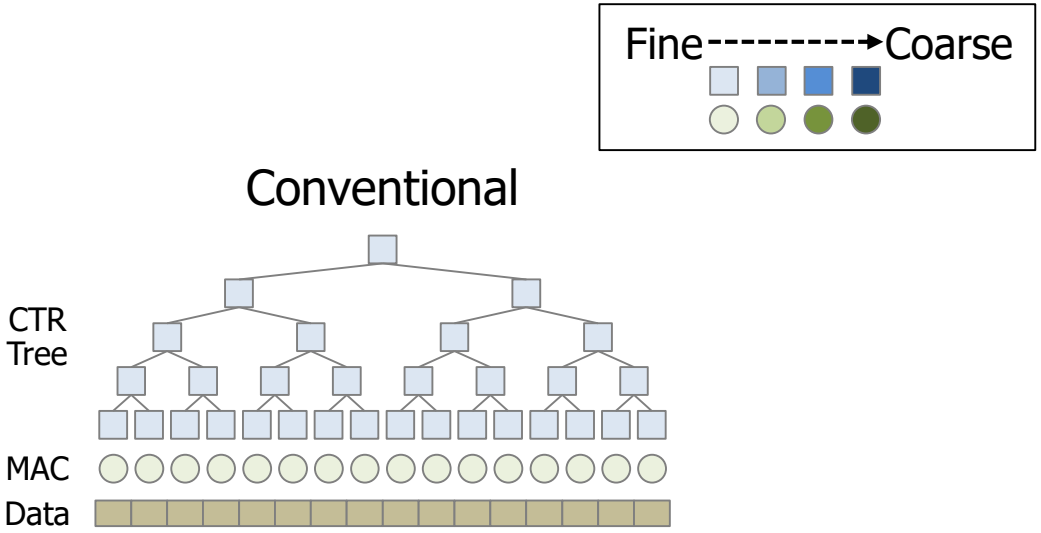
- Multi-granular MAC&Tree

# Multi-granular MAC&Tree

- Multi-granular MAC&Tree
  - Dynamically supports multi-granular MACs and a counter tree

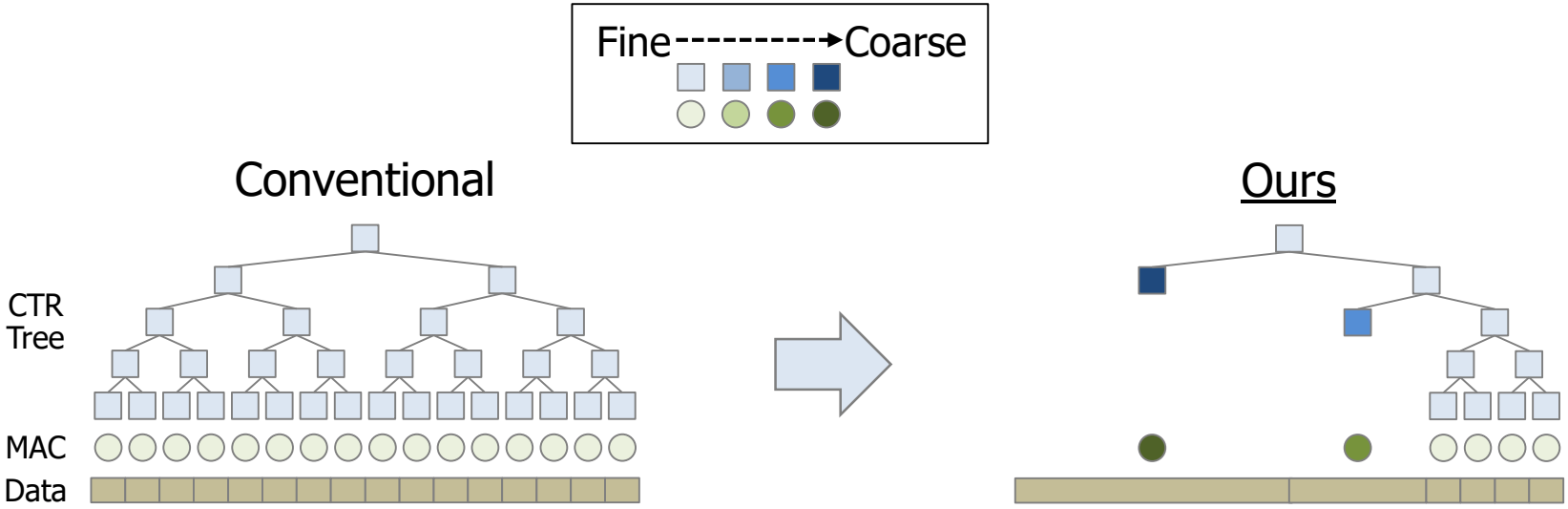
# Multi-granular MAC&Tree

- Multi-granular MAC&Tree
  - Dynamically supports multi-granular MACs and a counter tree



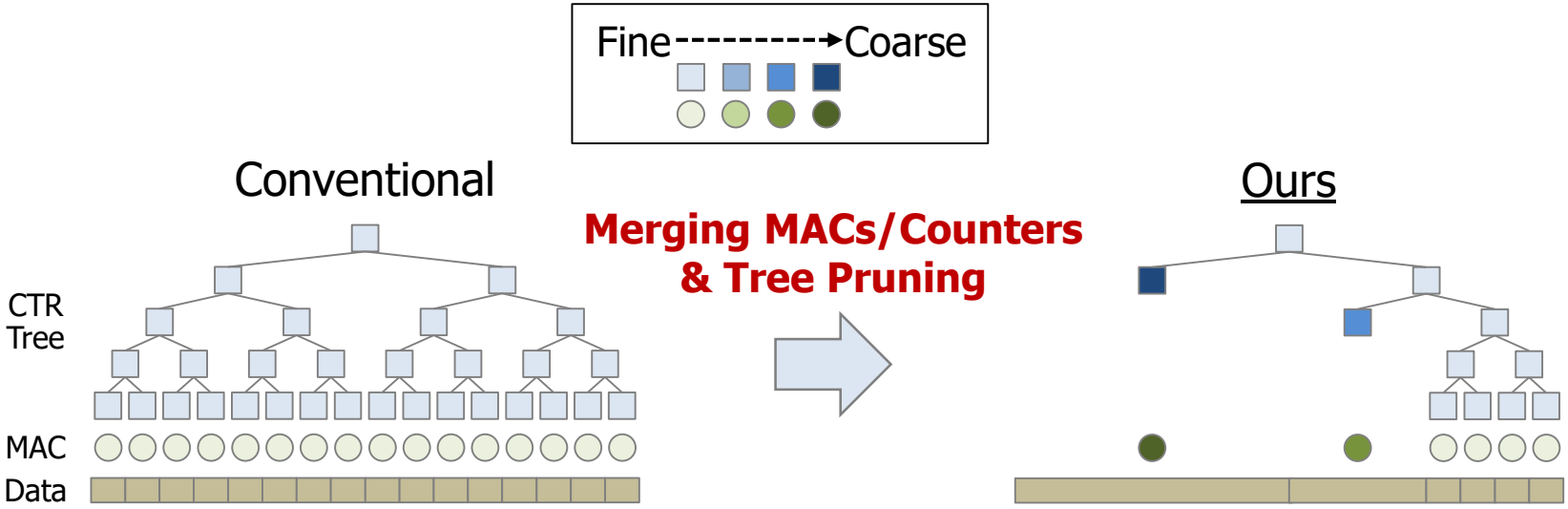
# Multi-granular MAC&Tree

- Multi-granular MAC&Tree
  - Dynamically supports multi-granular MACs and a counter tree



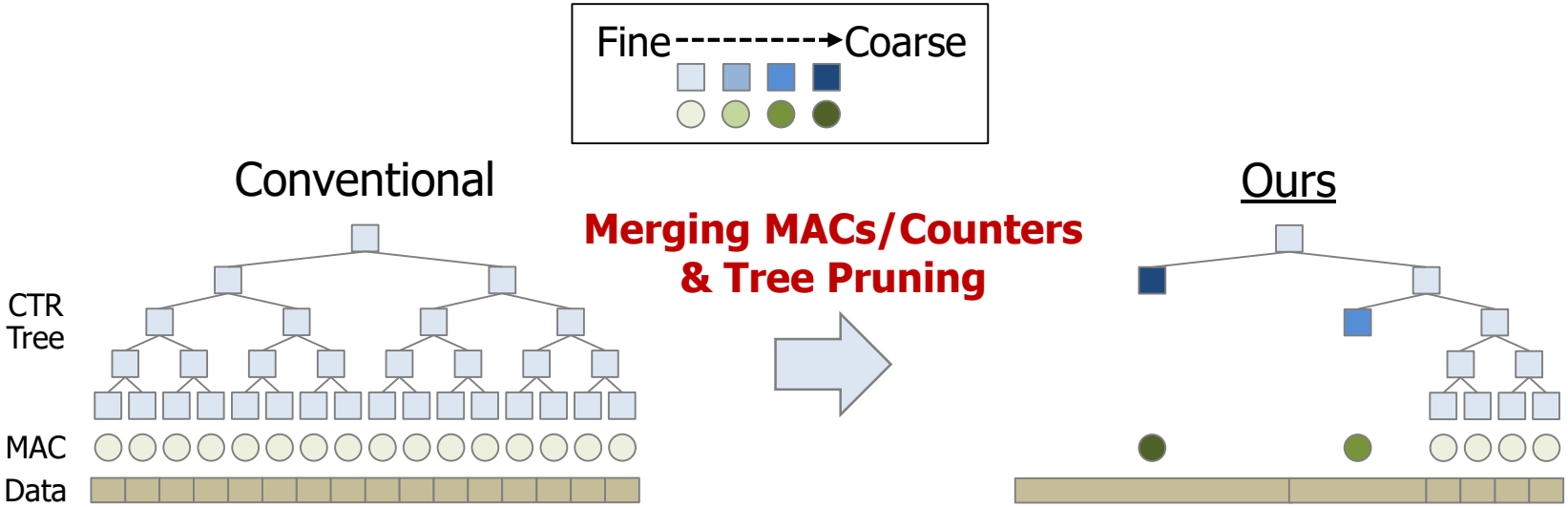
# Multi-granular MAC&Tree

- Multi-granular MAC&Tree
  - Dynamically supports multi-granular MACs and a counter tree
  - Key Idea: Merging MACs/counters & pruning a counter tree



# Multi-granular MAC&Tree

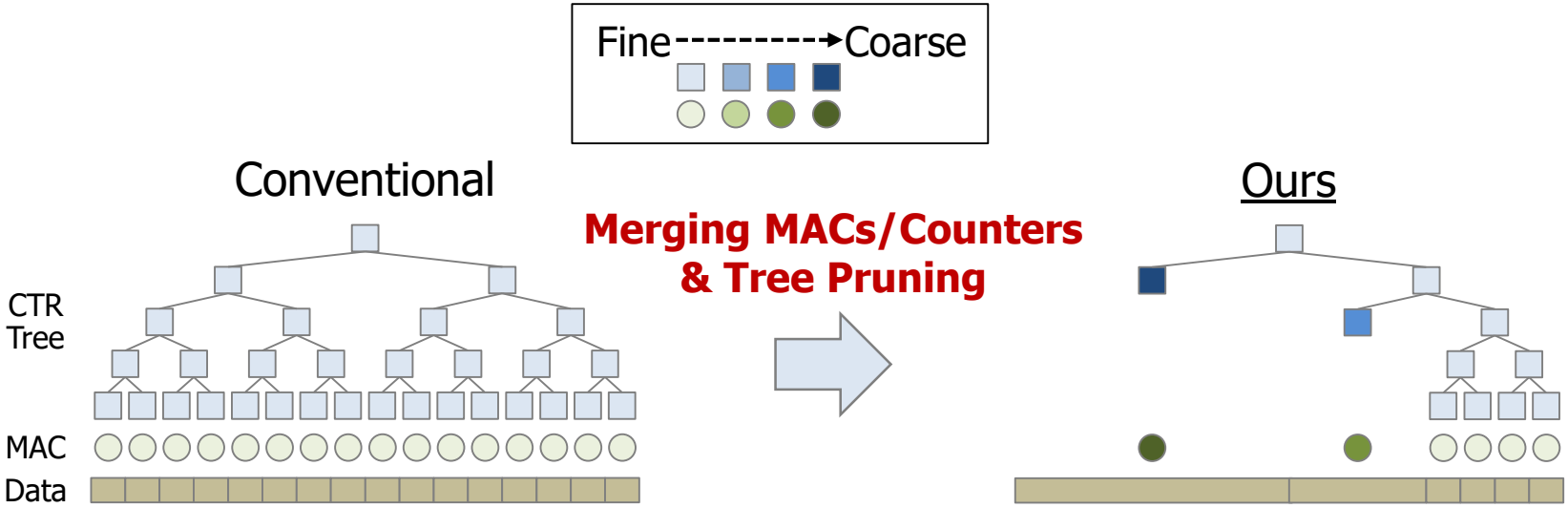
- Multi-granular MAC&Tree
  - Dynamically supports multi-granular MACs and a counter tree
  - Key Idea: Merging MACs/counters & pruning a counter tree




**1. How to dynamically detect granularity**

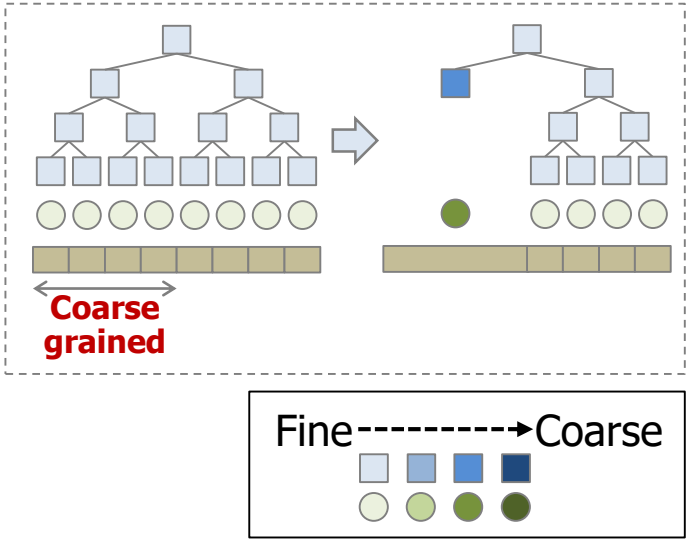
# Multi-granular MAC&Tree

- Multi-granular MAC&Tree
  - Dynamically supports multi-granular MACs and a counter tree
  - Key Idea: Merging MACs/counters & pruning a counter tree



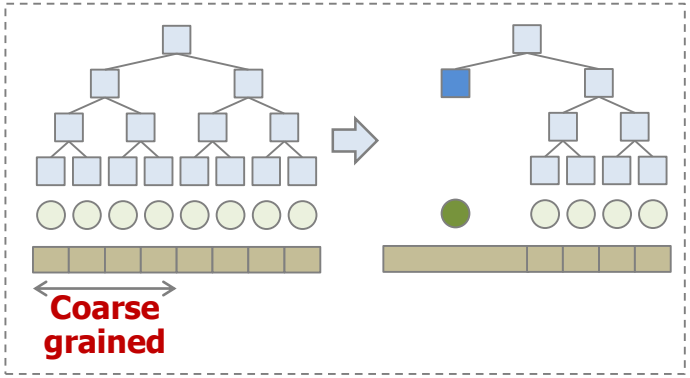

**1. How to dynamically detect granularity**
**2. How to switch granularity**

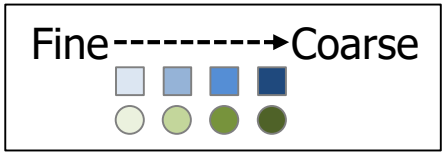
# Granularity Detection (Fine → Coarse)

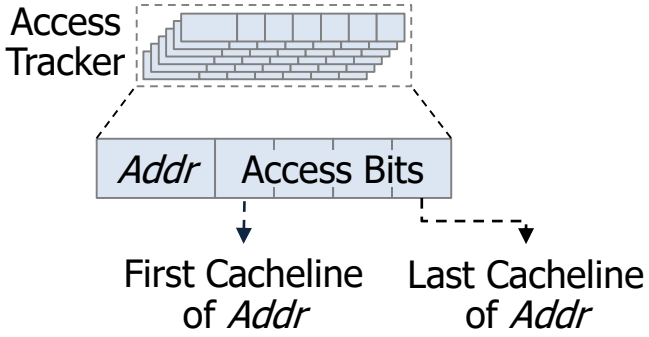


# Granularity Detection (Fine → Coarse)

- Access tracker
  - Records accessed addresses

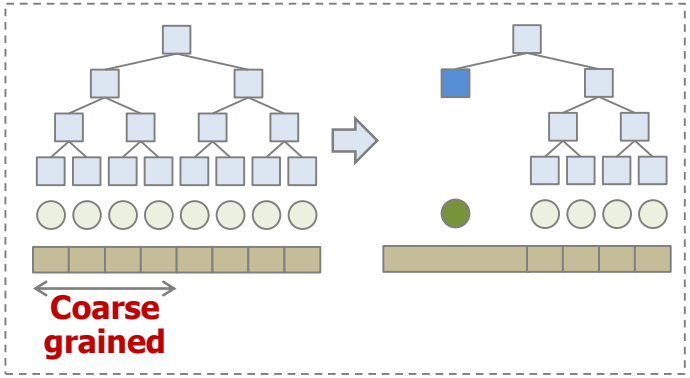


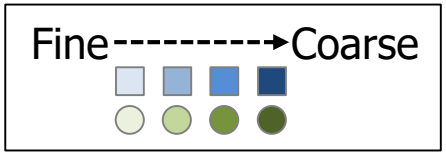



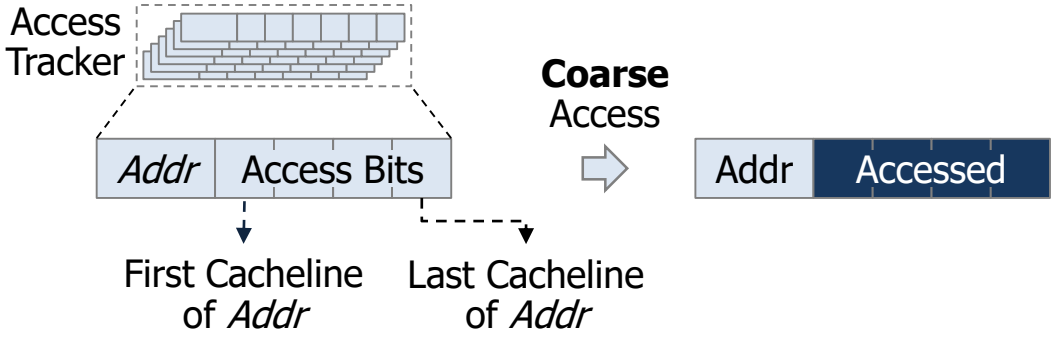



# Granularity Detection (Fine → Coarse)

- Access tracker
  - Records accessed addresses
  - Consecutive access bits are set

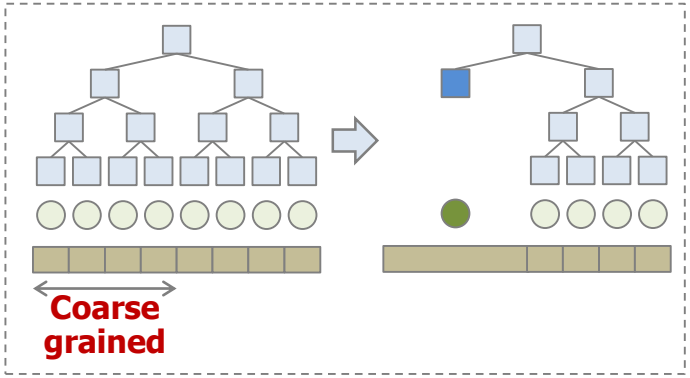


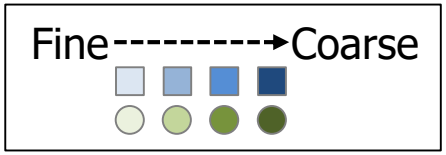



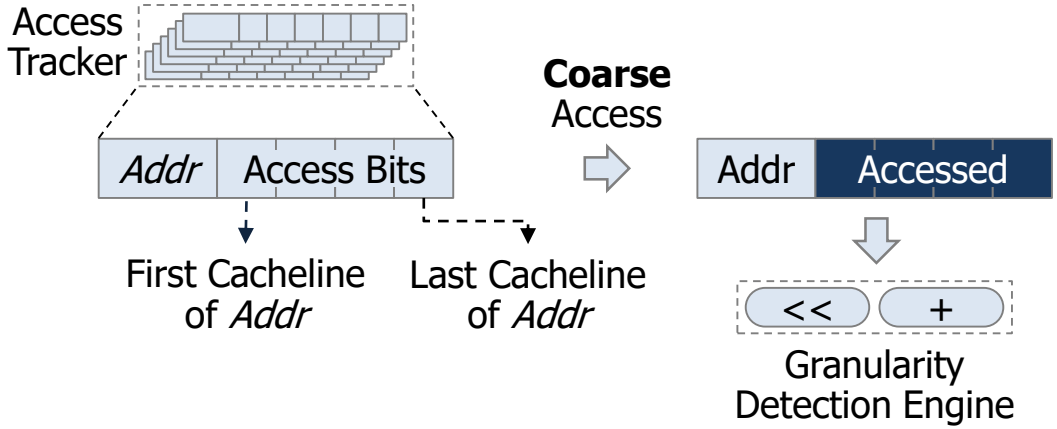



# Granularity Detection (Fine → Coarse)

- Access tracker
  - Records accessed addresses
  - Consecutive access bits are set
- Granularity detection engine
  - Computes a new granularity

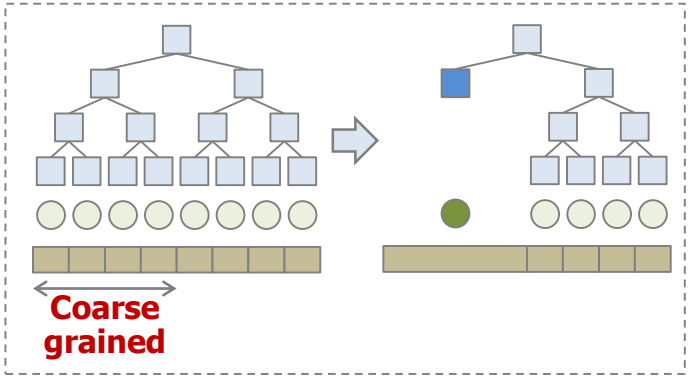


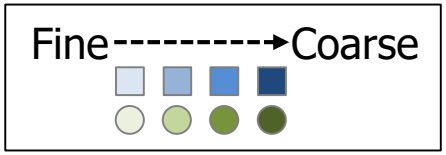



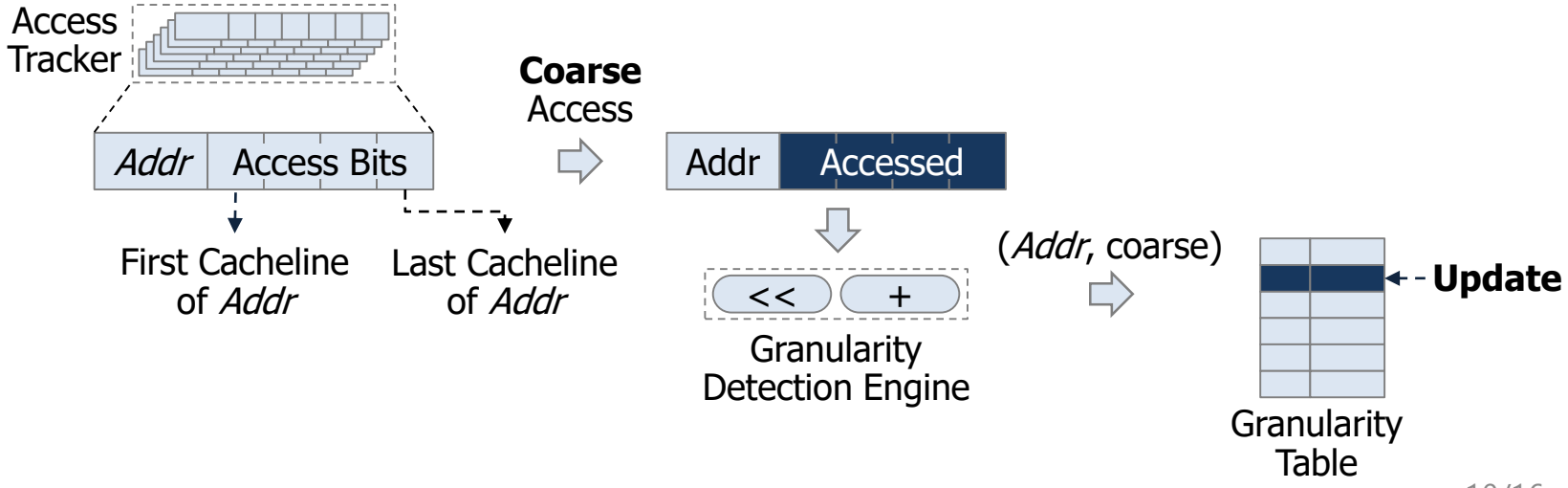



# Granularity Detection (Fine → Coarse)

- Access tracker
  - Records accessed addresses
  - Consecutive access bits are set
- Granularity detection engine
  - Computes a new granularity
  - Updates granularity table

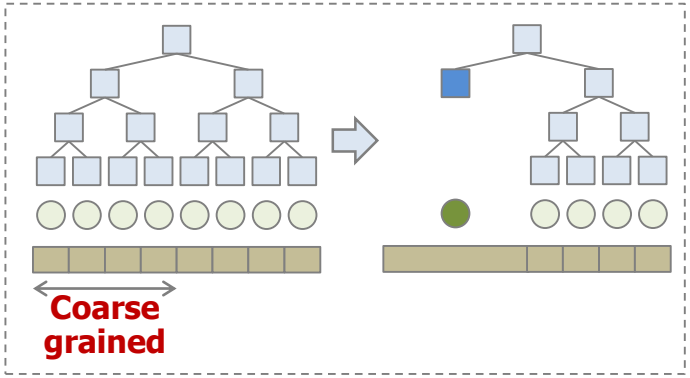


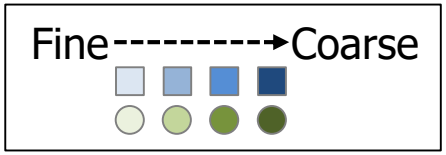



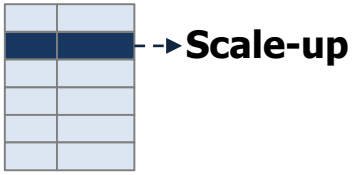



# Granularity Switching (Fine → Coarse)

- Granularity switching engine



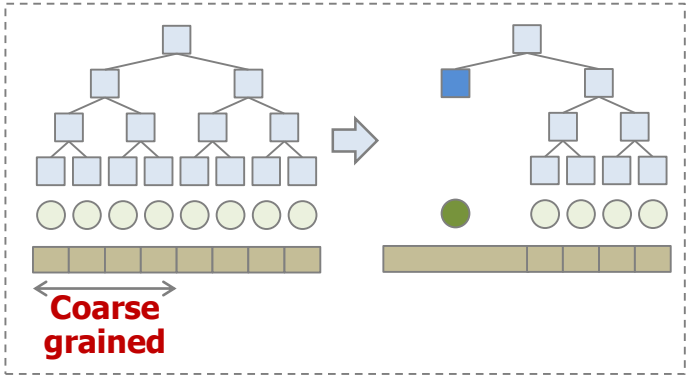



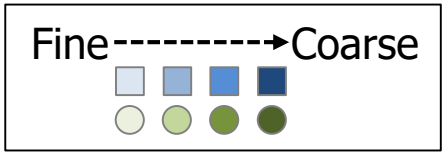

Granularity
Table

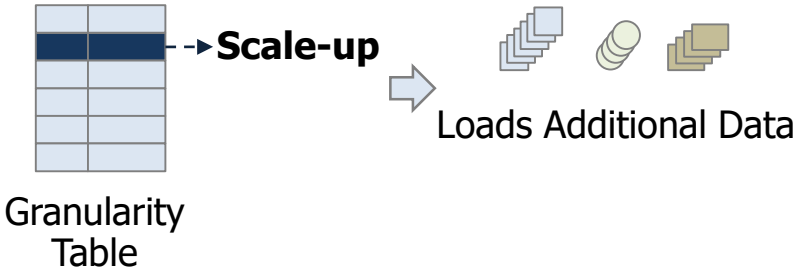
# Granularity Switching (Fine → Coarse)

- Granularity switching engine
  - Loads additional data

    → Old counters, MACs, data blocks



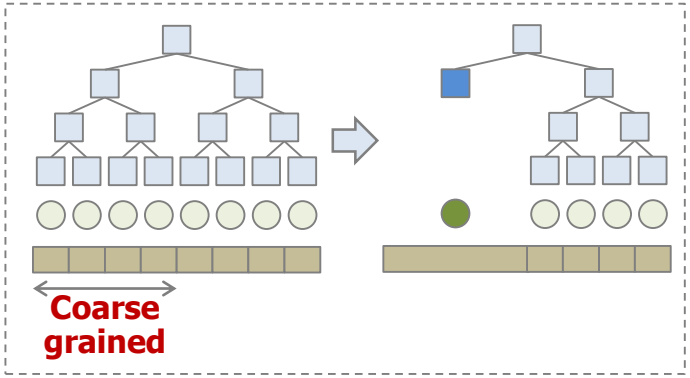



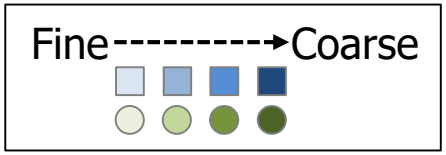



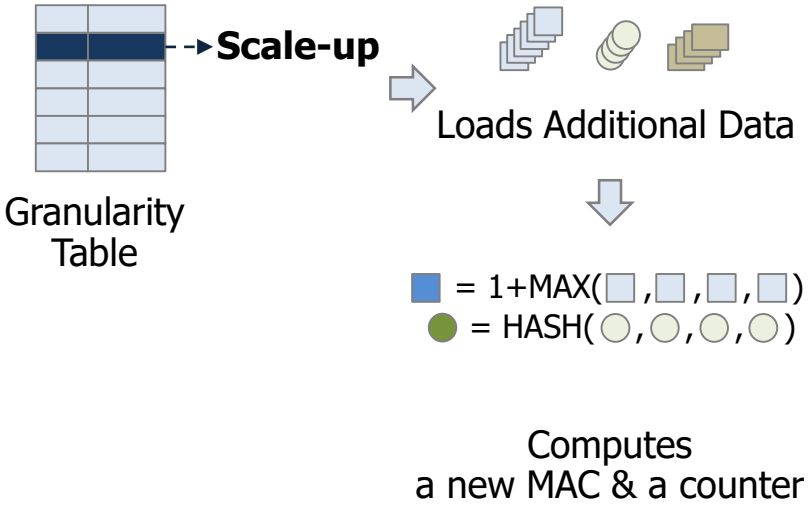
# Granularity Switching (Fine → Coarse)

- Granularity switching engine
  - Loads additional data

    → Old counters, MACs, data blocks
  - Computes counters, MACs

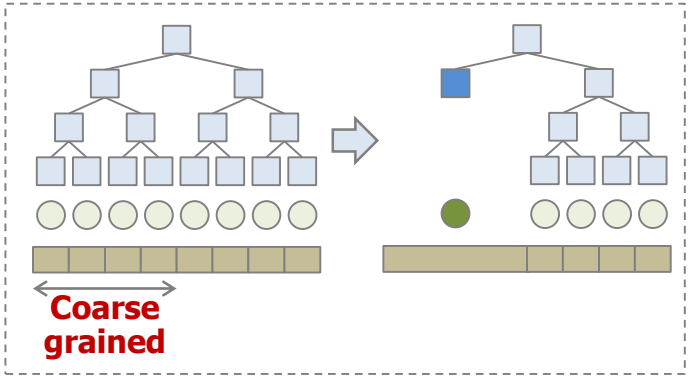


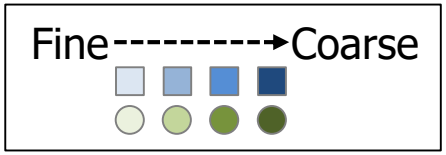



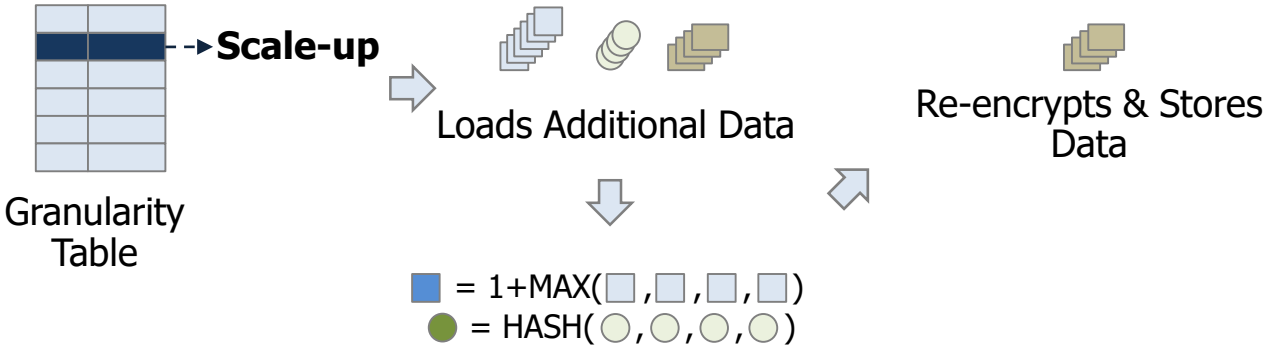



# Granularity Switching (Fine → Coarse)

- Granularity switching engine
  - Loads additional data
    → Old counters, MACs, data blocks
  - Computes counters, MACs
  - Re-encrypts old data

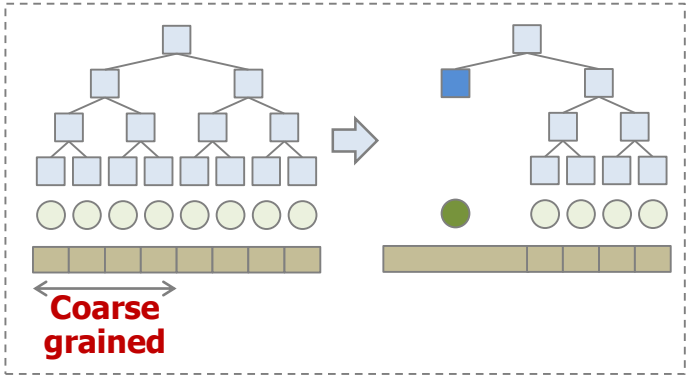


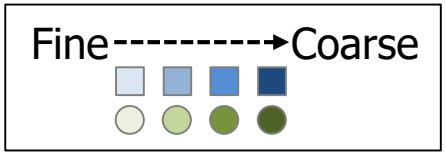



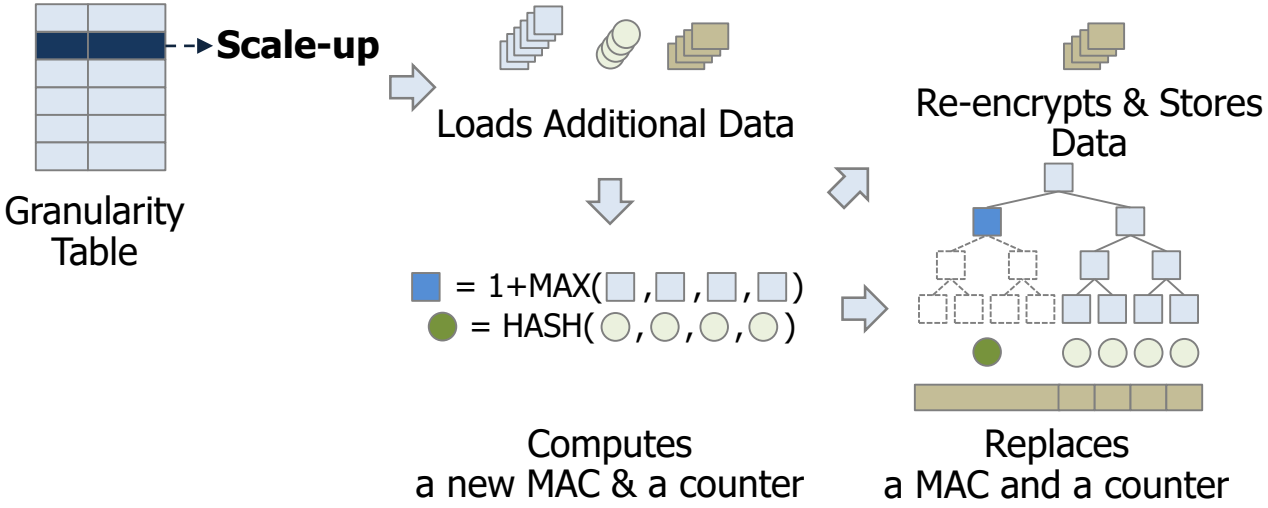



# Granularity Switching (Fine → Coarse)

- Granularity switching engine
  - Loads additional data
    → Old counters, MACs, data blocks
  - Computes counters, MACs
  - Re-encrypts old data
  - Updates & prunes integrity tree

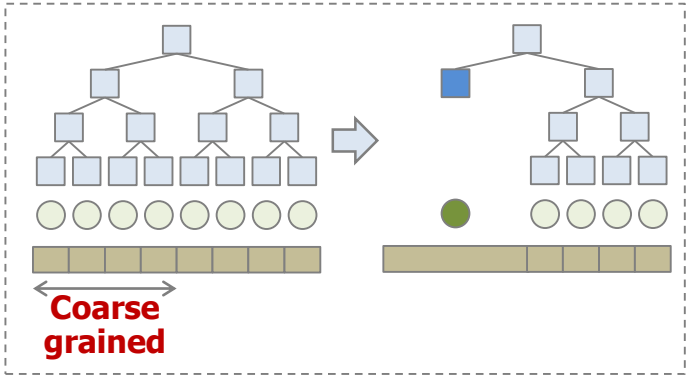


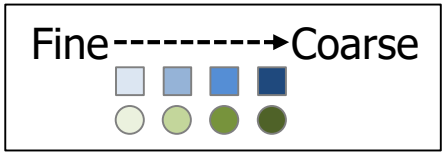



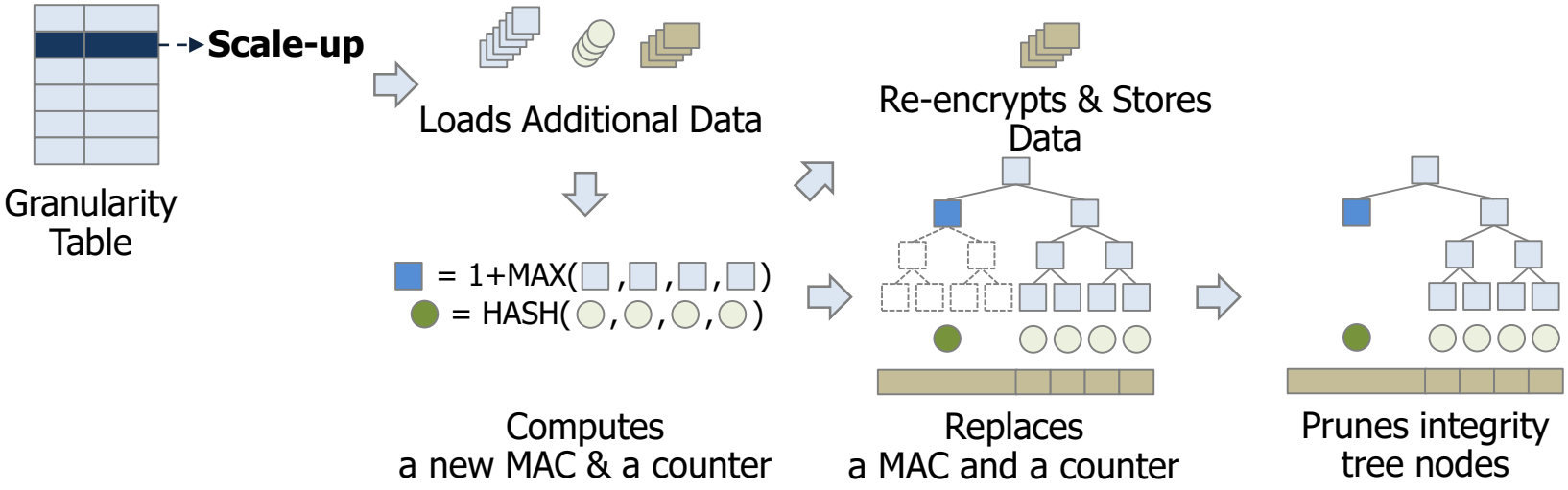



# Granularity Switching (Fine → Coarse)

- Granularity switching engine
  - Loads additional data
    → Old counters, MACs, data blocks
  - Computes counters, MACs
  - Re-encrypts old data
  - Updates & prunes integrity tree

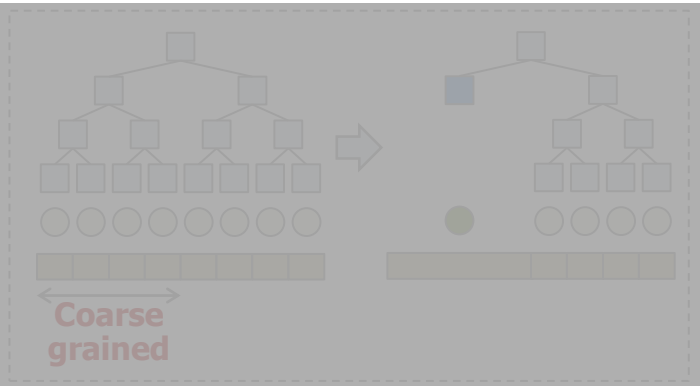


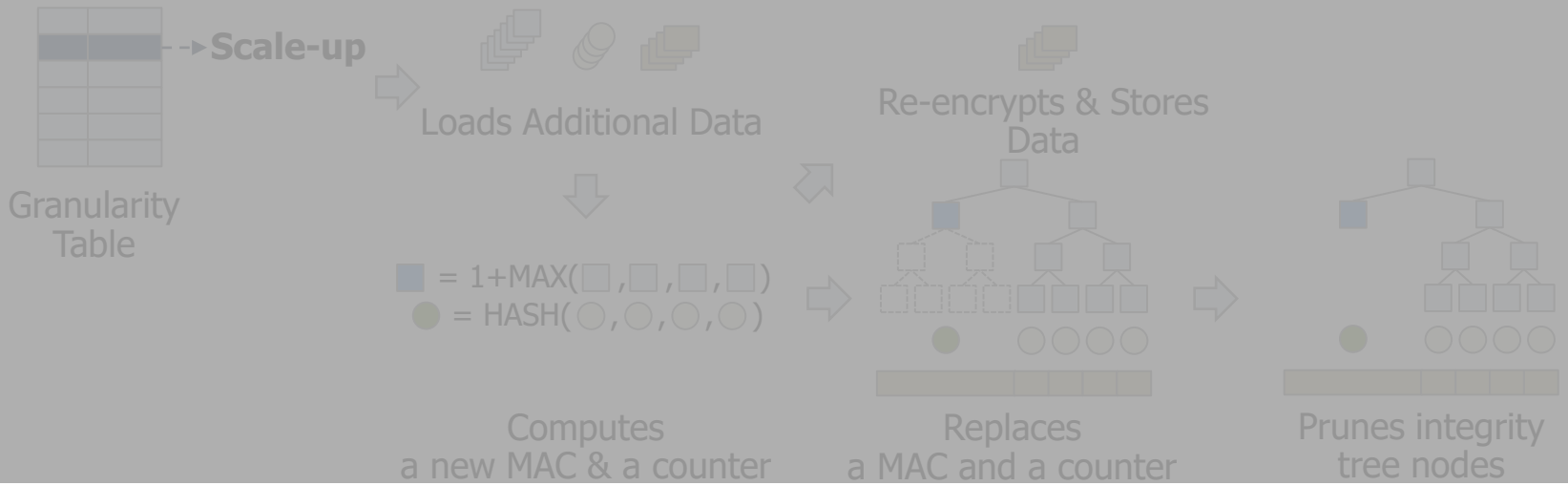



# Granularity Switching (Fine → Coarse)

- Granularity switching engine
  - Loads additional data
    → Old counters, MACs, data blocks
  - Computes counters, MACs




**Granularity switching requires significant overhead → Lazy switching**



# Evaluation Environment

# Evaluation Environment

- ChampSim (CPU) + MGPUSim (GPU) + mNPUsim (NPU)

# Evaluation Environment

- ChampSim (CPU) + MGPUSim (GPU) + mNPUsim (NPU)
- Configuration: Similar to NVIDIA Orin
  - ARM Cortex CPU + Ampere GPU + 2 x NVDLA with LPDDR4

| | CPU (Jetson AGX Orin ARM Cortex) | GPU (Jetson AGX Orin Ampere) | NPU (NVDLA) |
|---|---|---|---|
| Compute Engine | 8-core | 14 SMs | 45 x 45 Systolic Array |
| On-chip Storage | Cache (L1: 64KB, L2: 2MB) | Cache (L1: 192 KB, L2: 4MB) | Scratchpad Memory (2.2MB in total) |
| Frequency | 2.2GHz | 1GHz | 1GHz |
| Memory System | 2.4GHz, 17GB/s, LPDDR4 Memory System | | |

# Evaluation Environment

- Workloads & Scenarios
  - 14 workloads, 250 scenarios (all combinations)

# Evaluation Environment

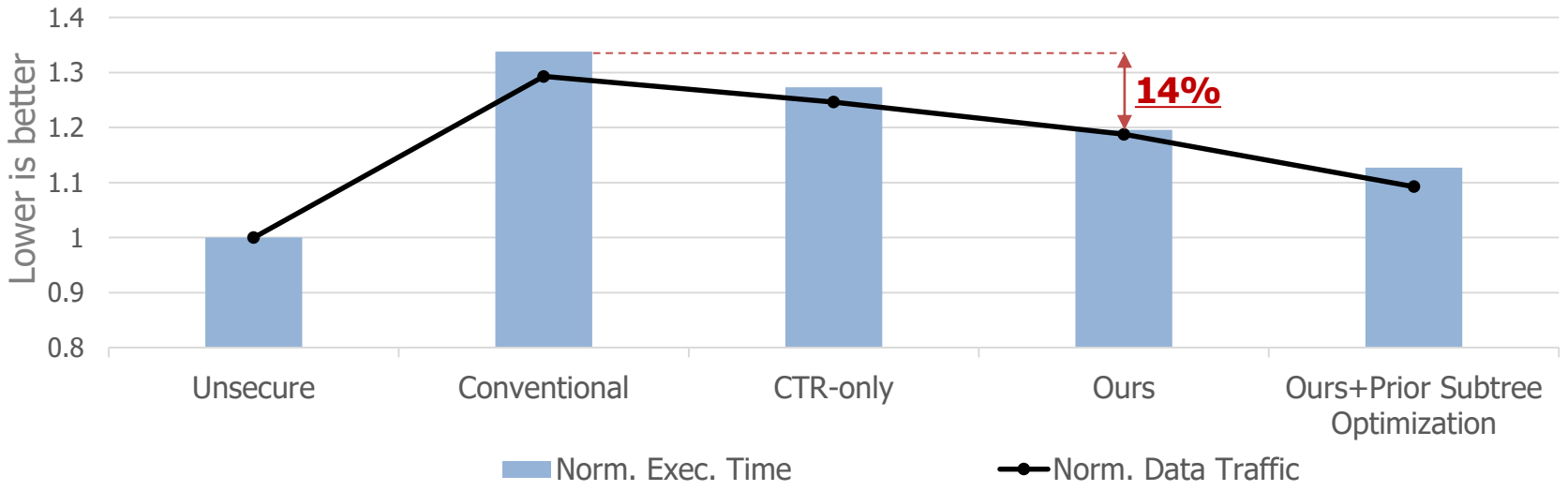
- Workloads & Scenarios
  - 14 workloads, 250 scenarios (all combinations)
  - Access pattern: Fine – ff – f – c – cc – Coarse | Diverse (d)
  - Traffic per cycles: Small (s) – Medium (m) – Large (l)

| | Workloads (access pattern-traffic per cycles) |
|---|---|
| CPU | bw (ff-s), gcc (ff-s), mcf (ff-m), xal (f-m), ray (ff-s) |
| GPU | syr2k (ff-m), pr (f-m), sten (c-l), mm (cc-m), floyd (d-s), |
| NPU | ncf (c-s), dlrm (c-s), sfrnn (c-l), alex (cc-m) |

# Evaluation Result (Avg of Processing Units)

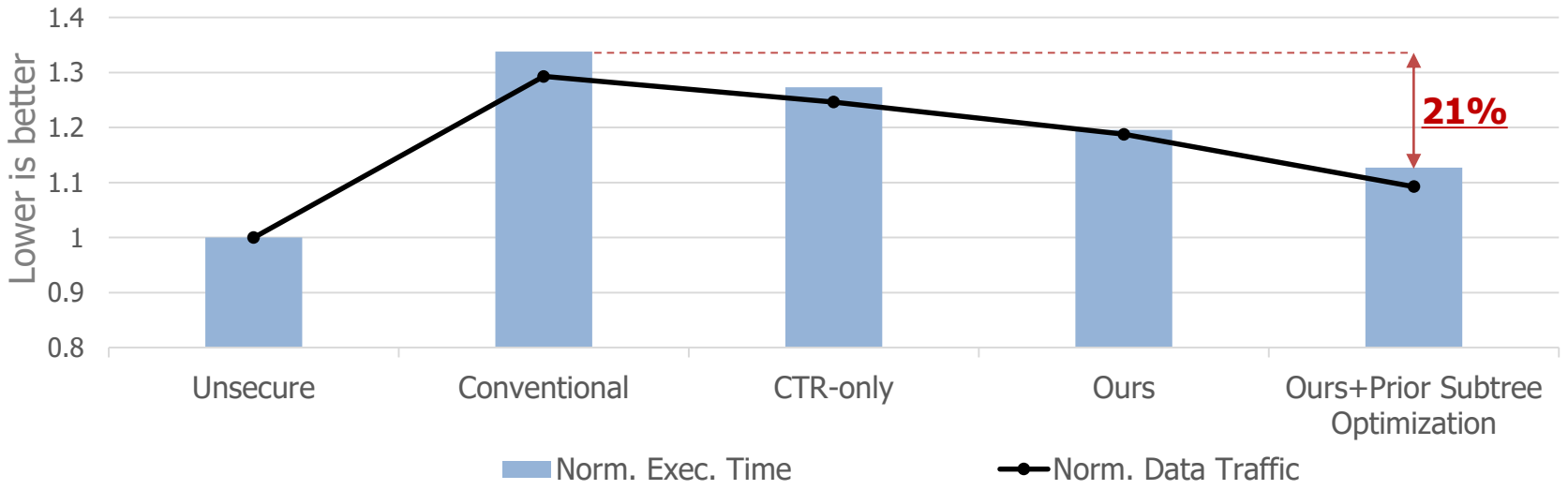
# Evaluation Result (Avg of Processing Units)

- **14%** improvement with **11%** data reduction



# Evaluation Result (Avg of Processing Units)

- **14%** improvement with **11%** data reduction
- Combining prior subtree optimization [1-4]
  - Performance improvement: **14%** → **21%**

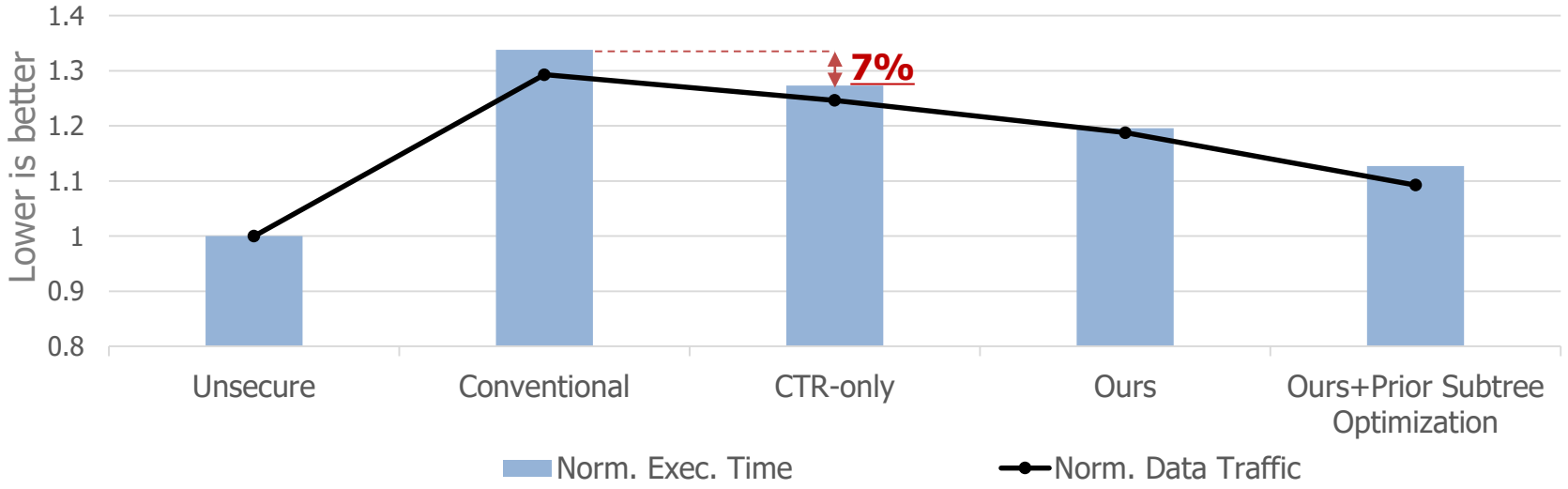




[1] Bonsai Merkle Forests: Efficiently Achieving Crash Consistency in Secure Persistent Memory (MICRO 2021)
[2] Scalable Memory Protection in the PENGLAI Enclave (OSDI 2021)
[3] Efficient Distributed Secure Memory with Migratable Merkle Tree (HPCA 2023)
[4] Data Enclave: A Data-Centric Trusted Execution Environment (HPCA 2024)

# Evaluation Result (Avg of Processing Units)

- **14%** improvement with **11%** data reduction
- Combining prior subtree optimization [1-4]
  - Performance improvement: **14%** → **21%**
  - CTR-only (**7%**), +MAC (**7%**), +Prior tree optimization (**7%**)

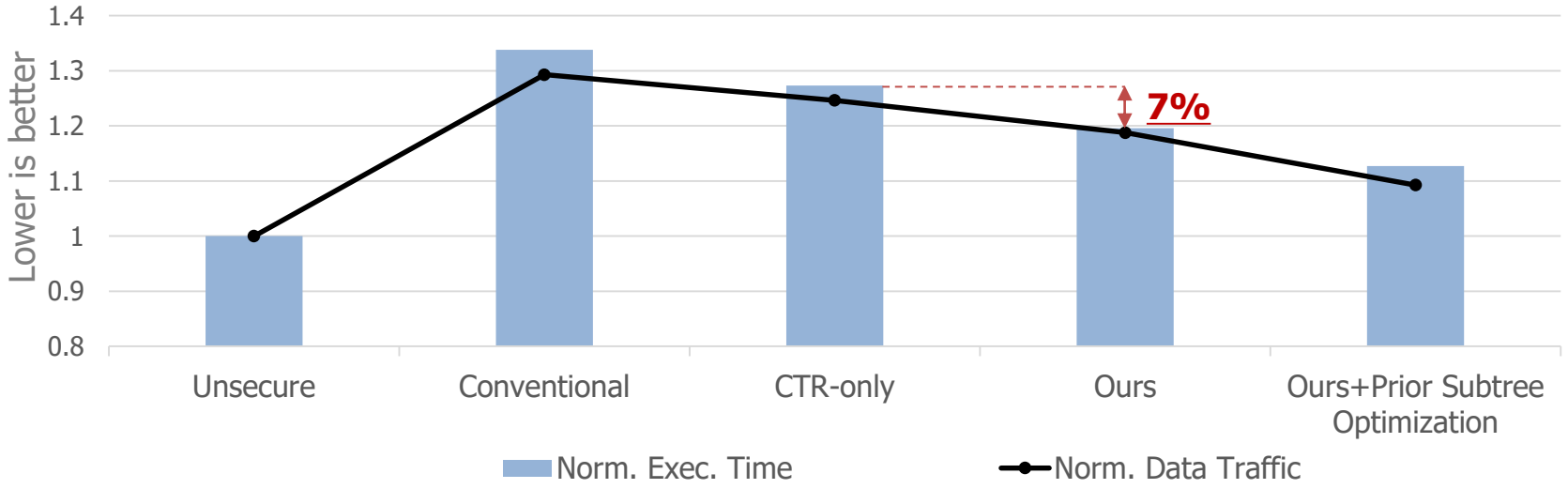




[1] Bonsai Merkle Forests: Efficiently Achieving Crash Consistency in Secure Persistent Memory (MICRO 2021)
[2] Scalable Memory Protection in the PENGLAI Enclave (OSDI 2021)
[3] Efficient Distributed Secure Memory with Migratable Merkle Tree (HPCA 2023)
[4] Data Enclave: A Data-Centric Trusted Execution Environment (HPCA 2024)

# Evaluation Result (Avg of Processing Units)

- **14%** improvement with **11%** data reduction
- Combining prior subtree optimization [1-4]
  - Performance improvement: **14%** → **21%**
  - CTR-only (**7%**), +MAC (**7%**), +Prior tree optimization (**7%**)

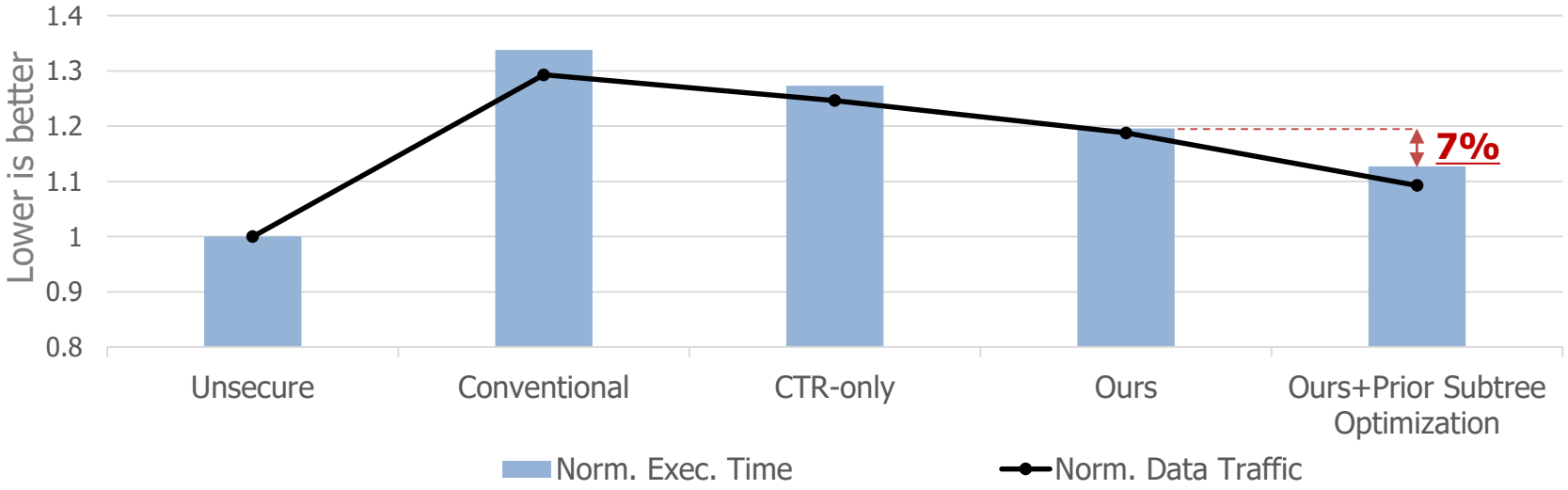




[1] Bonsai Merkle Forests: Efficiently Achieving Crash Consistency in Secure Persistent Memory (MICRO 2021)
[2] Scalable Memory Protection in the PENGLAI Enclave (OSDI 2021)
[3] Efficient Distributed Secure Memory with Migratable Merkle Tree (HPCA 2023)
[4] Data Enclave: A Data-Centric Trusted Execution Environment (HPCA 2024)

# Evaluation Result (Avg of Processing Units)

- **14%** improvement with **11%** data reduction
- Combining prior subtree optimization [1-4]
  - Performance improvement: **14%** → **21%**
  - CTR-only (**7%**), +MAC (**7%**), +Prior tree optimization (**7%**)



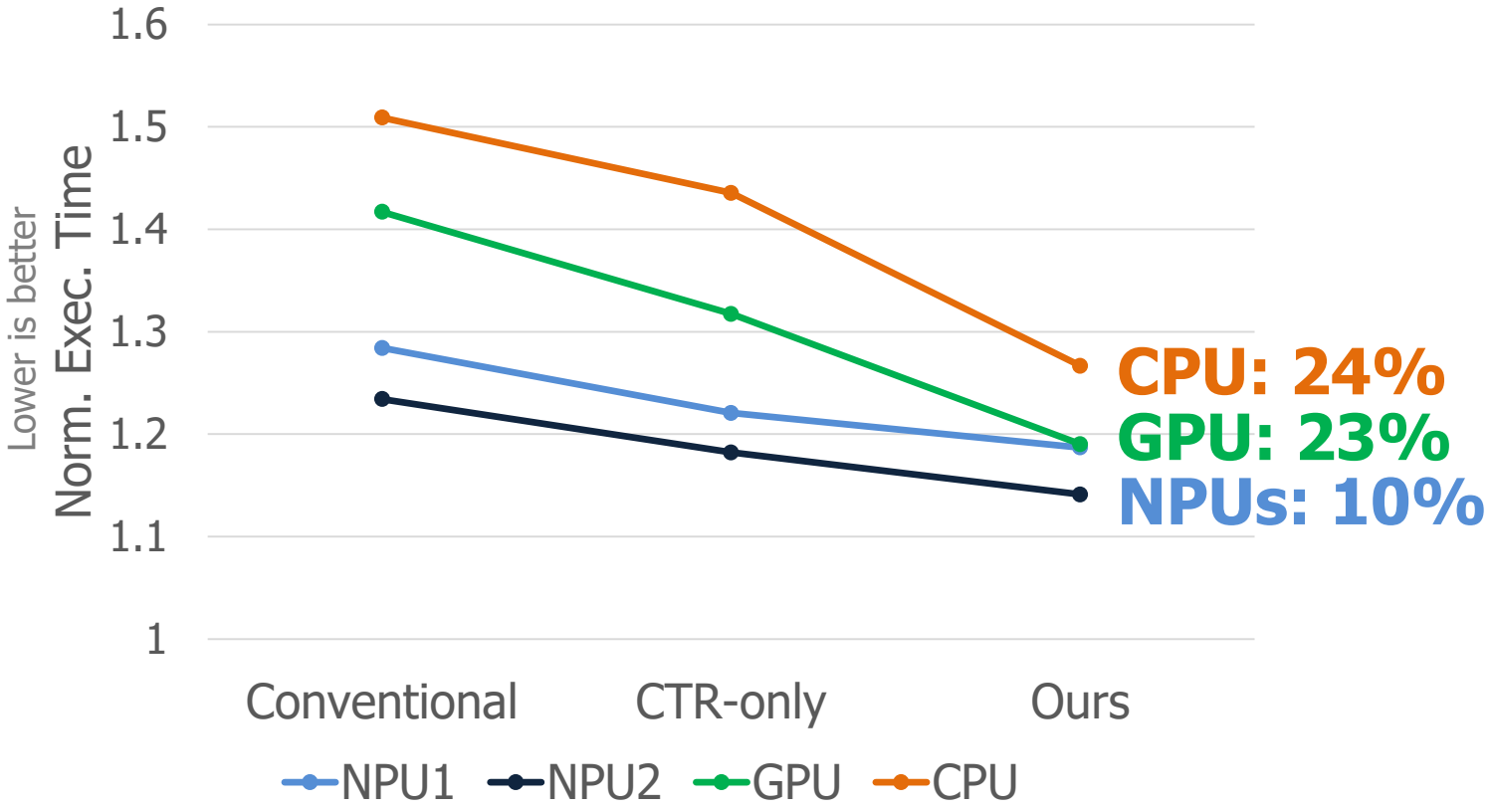


[1] Bonsai Merkle Forests: Efficiently Achieving Crash Consistency in Secure Persistent Memory (MICRO 2021)
[2] Scalable Memory Protection in the PENGLAI Enclave (OSDI 2021)
[3] Efficient Distributed Secure Memory with Migratable Merkle Tree (HPCA 2023)
[4] Data Enclave: A Data-Centric Trusted Execution Environment (HPCA 2024)

# Evaluation Result (Per Processing Unit)

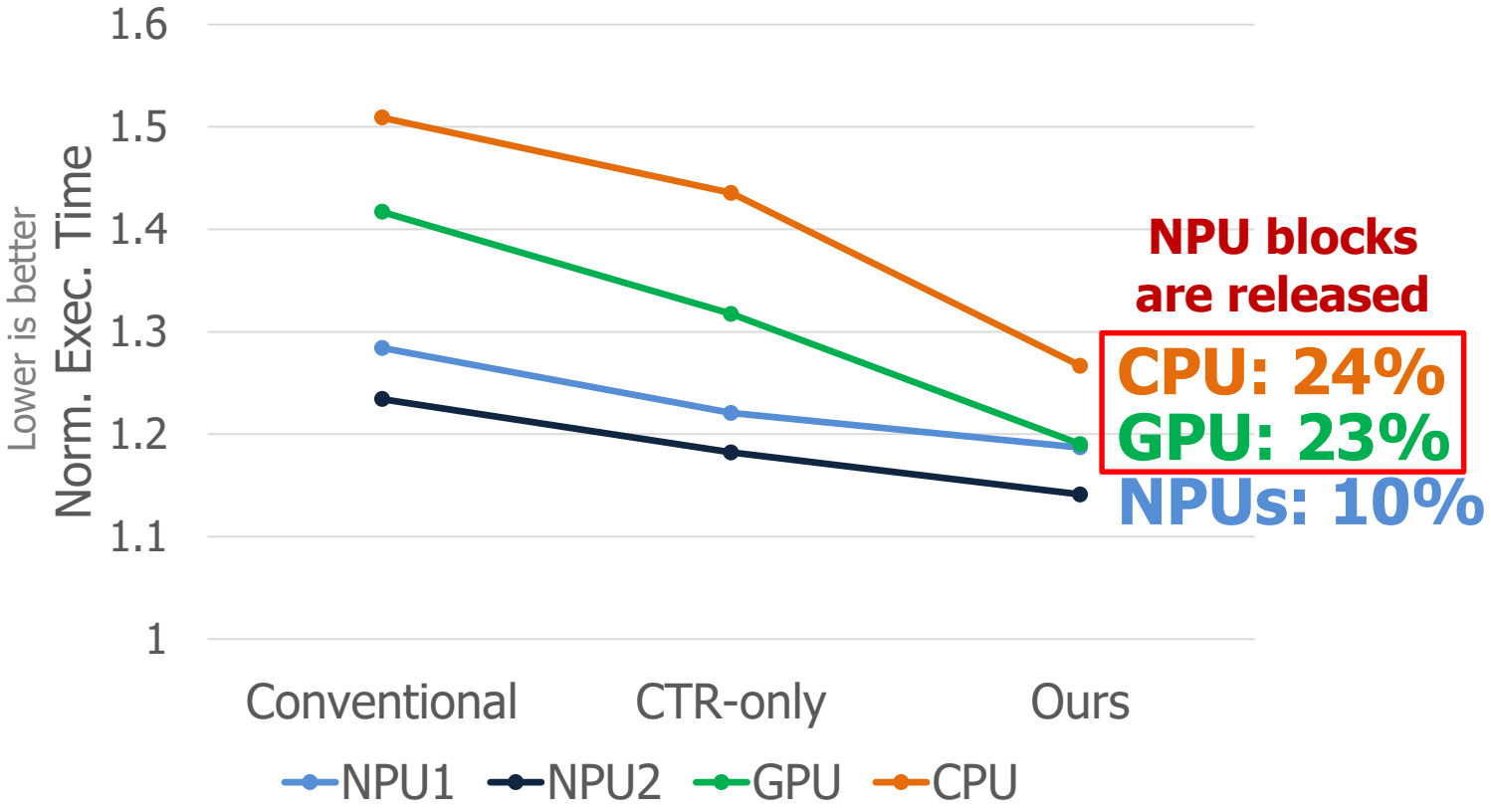
# Evaluation Result (Per Processing Unit)

- Performance improvement of each processing unit
  - CPU (**24%**), GPU (**23%**), NPU (**10%**)



# Evaluation Result (Per Processing Unit)

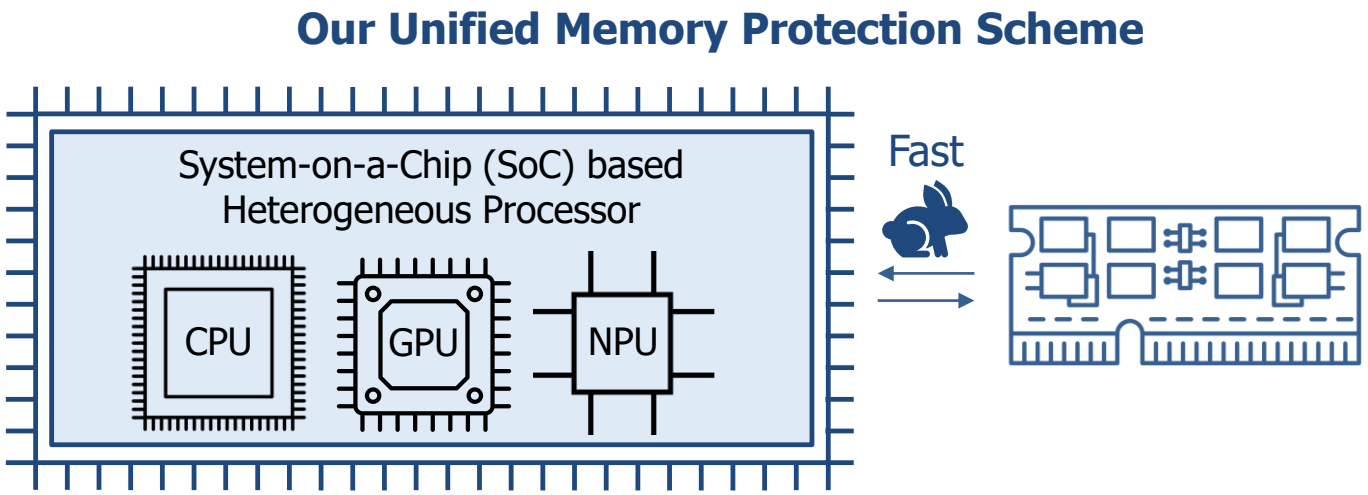
- Performance improvement of each processing unit
  - CPU (**24%**), GPU (**23%**), NPU (**10%**)



# Conclusion

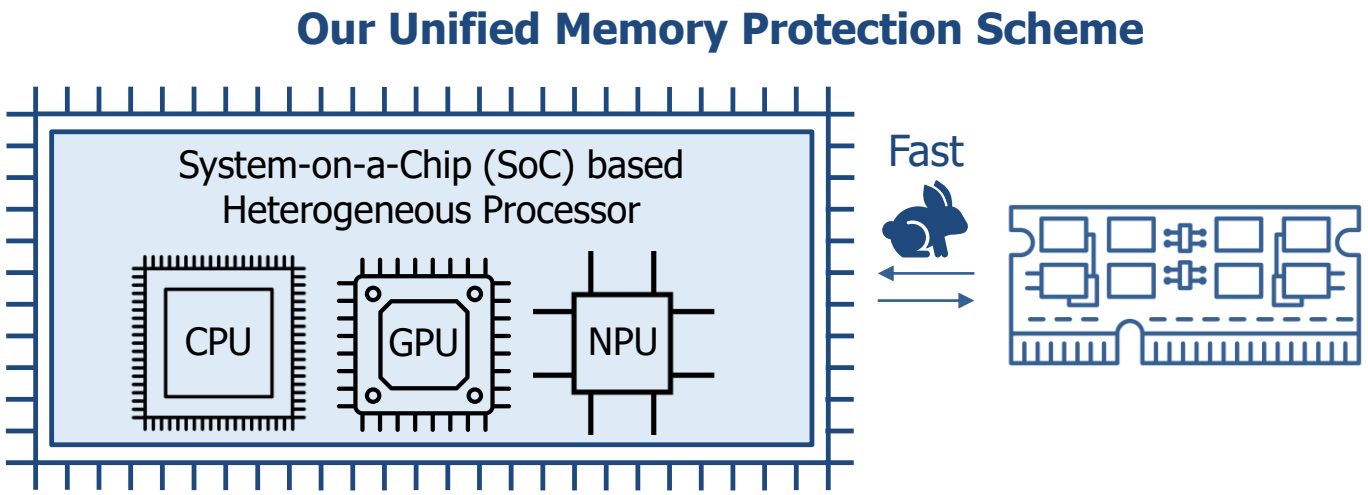
# Conclusion

- Unified memory protection for heterogeneous processor



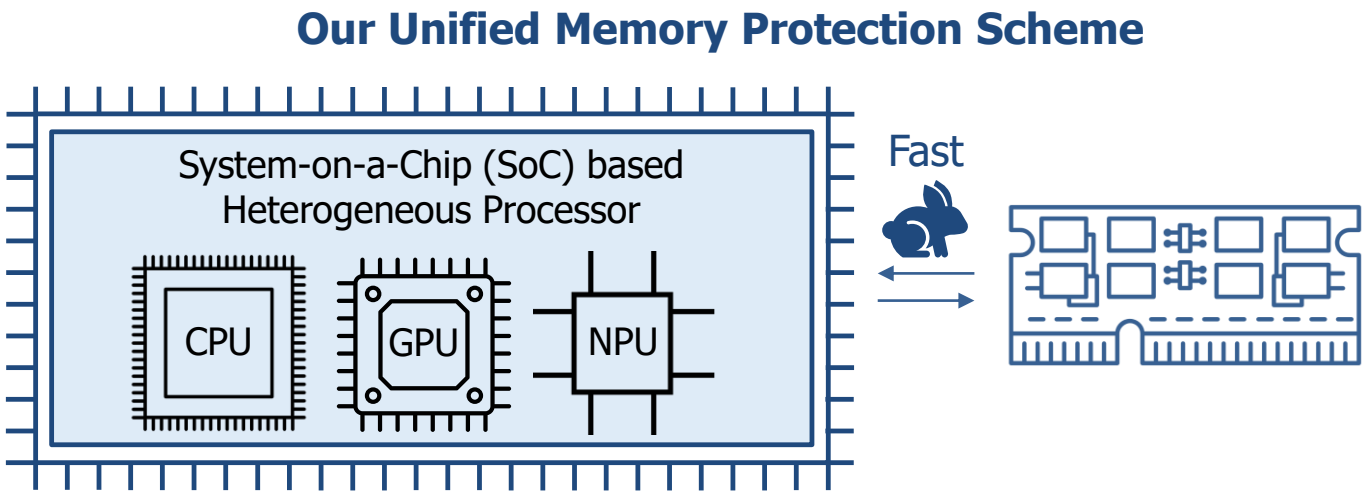
# Conclusion

- Unified memory protection for heterogeneous processor
  - **Multi-granular MAC & Integrity Tree**



# Conclusion

- Unified memory protection for heterogeneous processor
  - **Multi-granular MAC & Integrity Tree**
  - Challenge: Diverse access pattern

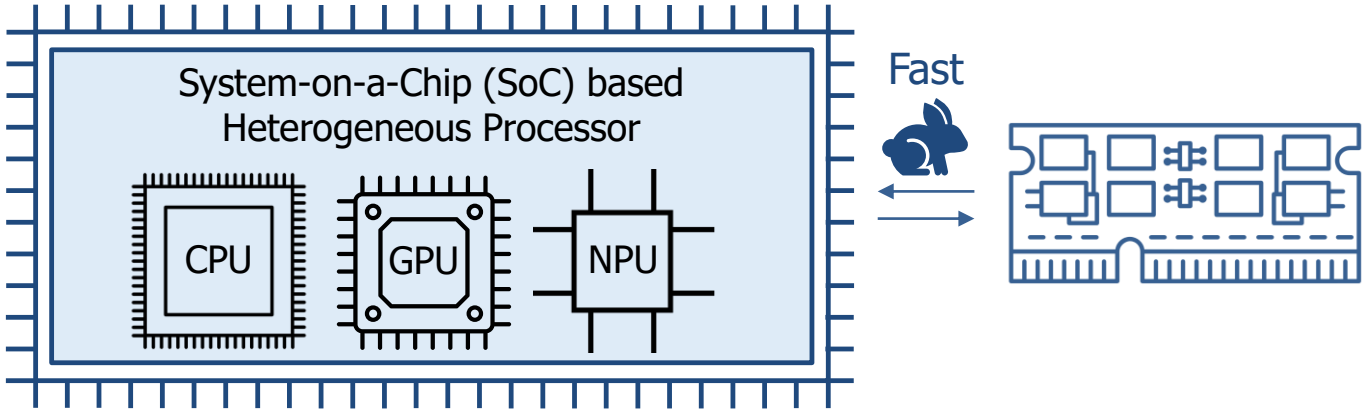


# Conclusion

- Unified memory protection for heterogeneous processor
  - **Multi-granular MAC & Integrity Tree**
  - Challenge: Diverse access pattern

- Improvement: **14%** (w/o subtree opt.), **21%** (w/ subtree opt.)

**Our Unified Memory Protection Scheme**



# Thank you

# Backup Slide

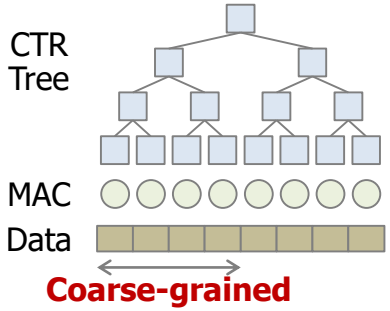
# Detailed Granularity Switching

# Detailed Granularity Switching

* Scale-up (Fine → Coarse)

# Detailed Granularity Switching

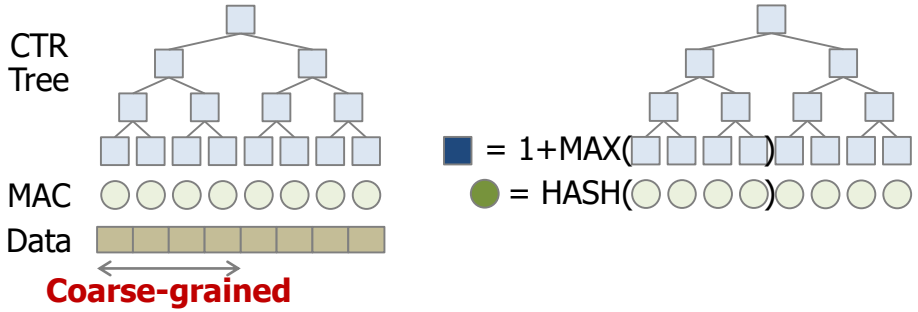
* Scale-up (Fine → Coarse)




1) Detect scale-up

# Detailed Granularity Switching

* Scale-up (Fine → Coarse)

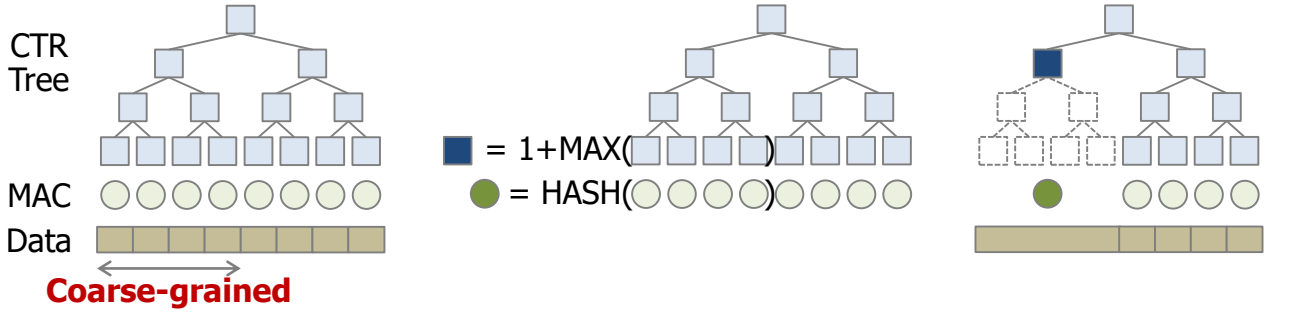



1) Detect scale-up

2) Compute new MAC/CTR

# Detailed Granularity Switching

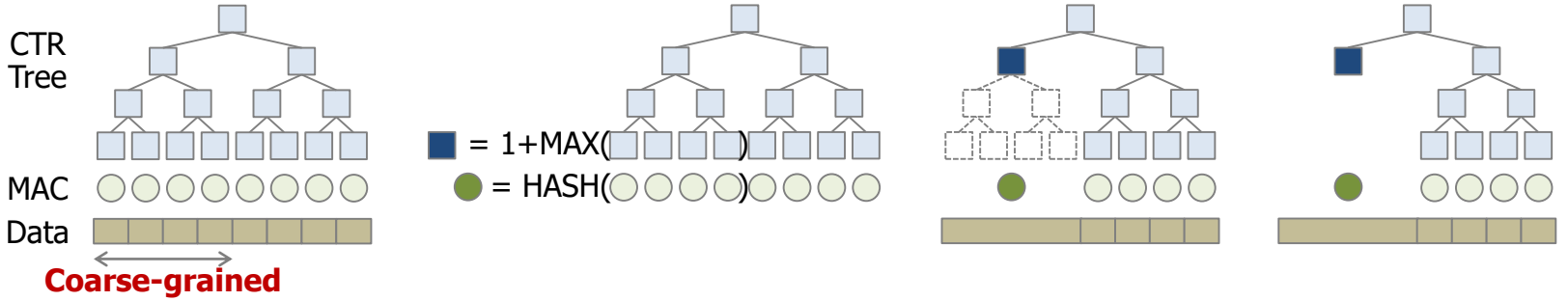
* Scale-up (Fine → Coarse)




1) Detect scale-up  2) Compute new MAC/CTR  3) Replace MAC/CTR

# Detailed Granularity Switching

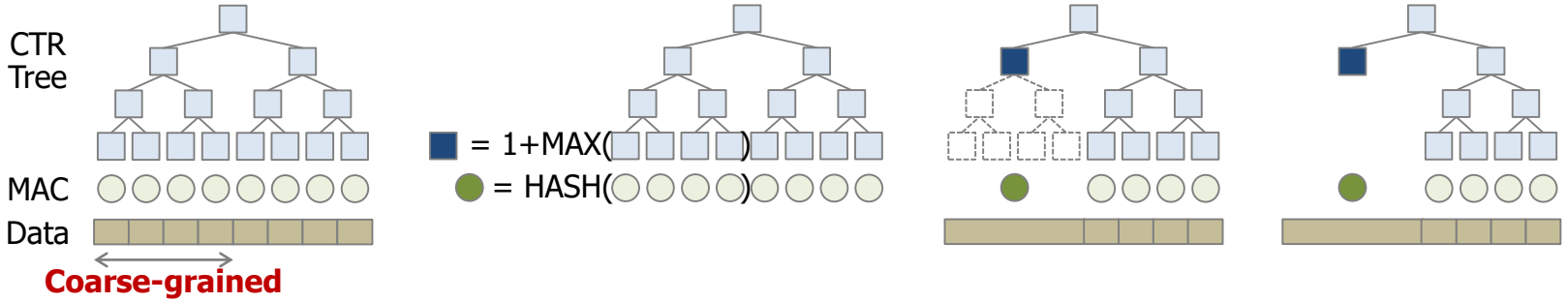
* Scale-up (Fine → Coarse)




1) Detect scale-up 2) Compute new MAC/CTR 3) Replace MAC/CTR 4) Prune CTR tree

# Detailed Granularity Switching

* Scale-up (Fine → Coarse)

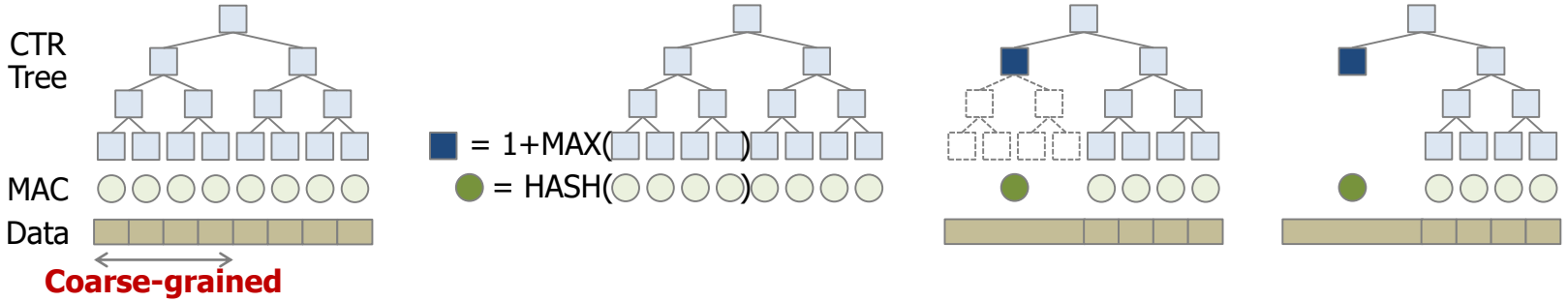



1) Detect scale-up 2) Compute new MAC/CTR 3) Replace MAC/CTR 4) Prune CTR tree

* Scale-down (Coarse → Fine)

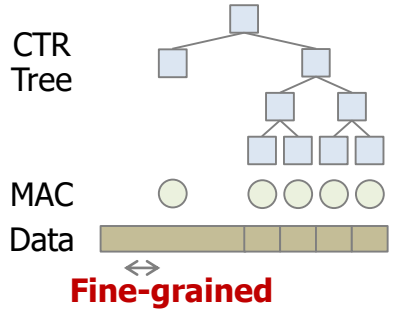
# Detailed Granularity Switching

* Scale-up (Fine → Coarse)




1) Detect scale-up 2) Compute new MAC/CTR 3) Replace MAC/CTR 4) Prune CTR tree

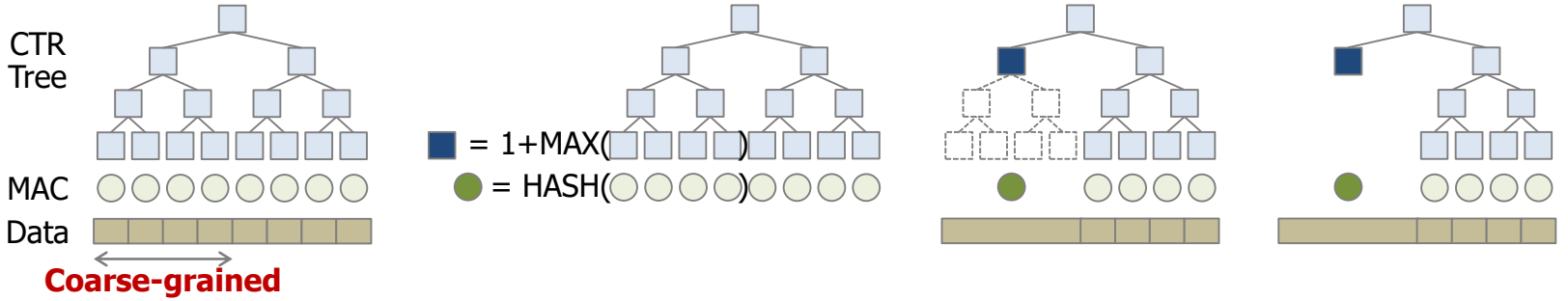
* Scale-down (Coarse → Fine)




1) Detect scale-down

# Detailed Granularity Switching

* Scale-up (Fine → Coarse)



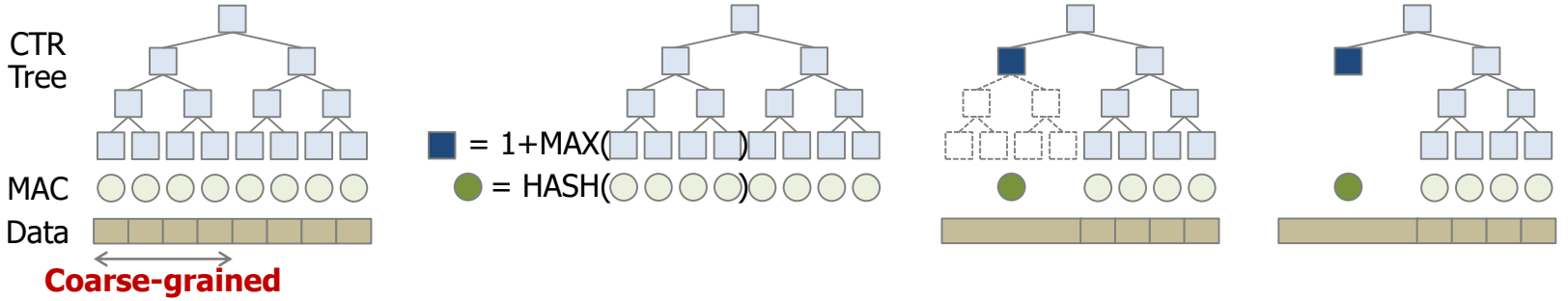

1) Detect scale-up 2) Compute new MAC/CTR 3) Replace MAC/CTR 4) Prune CTR tree

* Scale-down (Coarse → Fine)


1) Detect scale-down 2) Compute new MAC/CTR

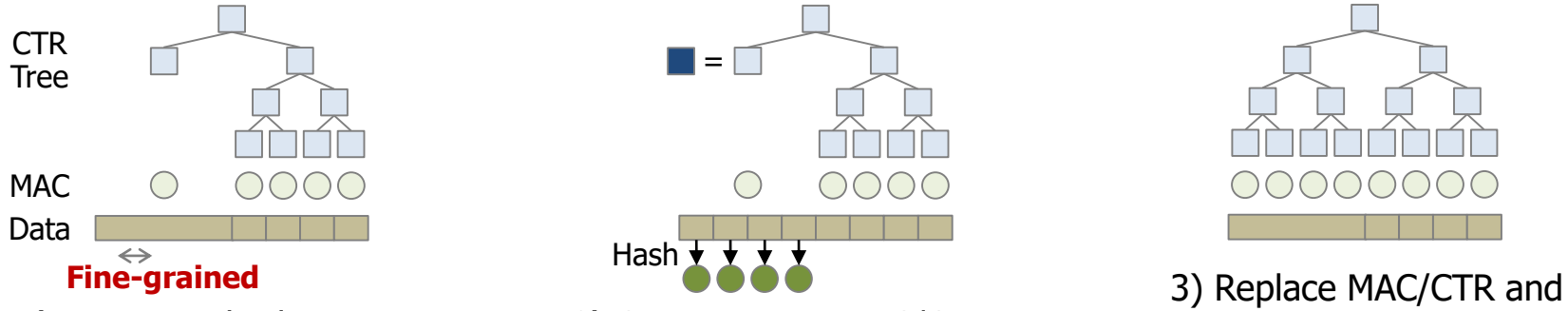
# Detailed Granularity Switching

* Scale-up (Fine → Coarse)




1) Detect scale-up 2) Compute new MAC/CTR 3) Replace MAC/CTR 4) Prune CTR tree

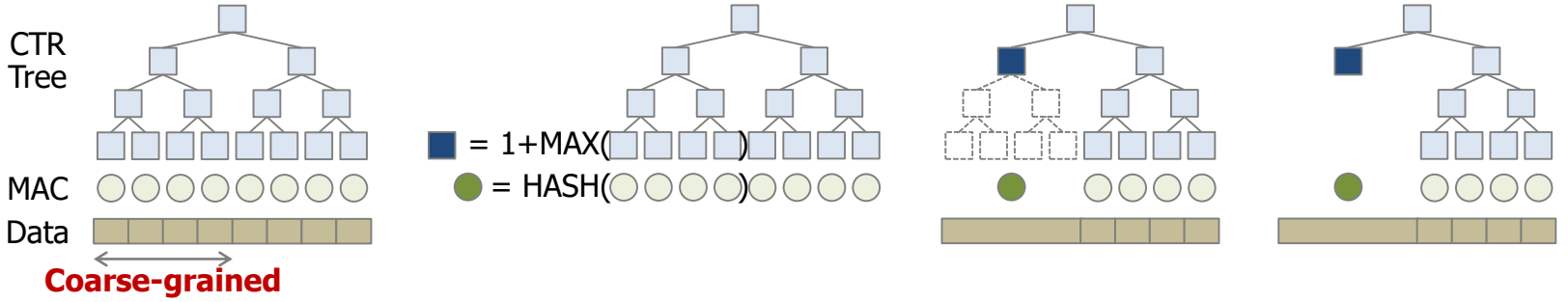
* Scale-down (Coarse → Fine)




1) Detect scale-down 2) Compute new MAC/CTR 3) Replace MAC/CTR and restore CTR tree

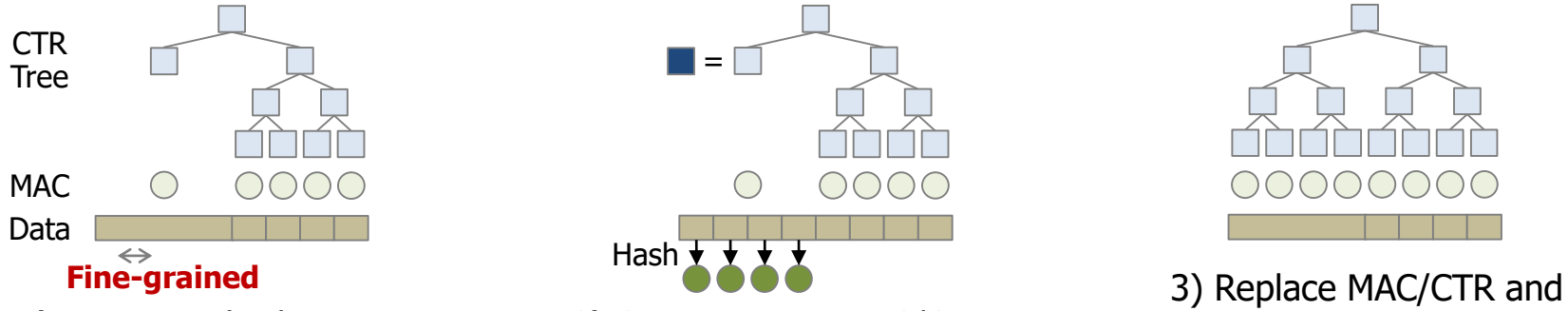
# Detailed Granularity Switching

* Scale-up (Fine → Coarse)




1) Detect scale-up 2) Compute new MAC/CTR 3) Replace MAC/CTR 4) Prune CTR tree

* Scale-down (Coarse → Fine)




1) Detect scale-down 2) Compute new MAC/CTR 3) Replace MAC/CTR and restore CTR tree

**Granularity switching requires significant overhead! → Lazy switching**

# Lazy Switching Overhead by MAC

# Lazy Switching Overhead by MAC

# Lazy Switching Overhead by MAC

$$Coarse\ MAC = HASH\ (Fine\ MACs)$$

* Scale-up

# Lazy Switching Overhead by MAC

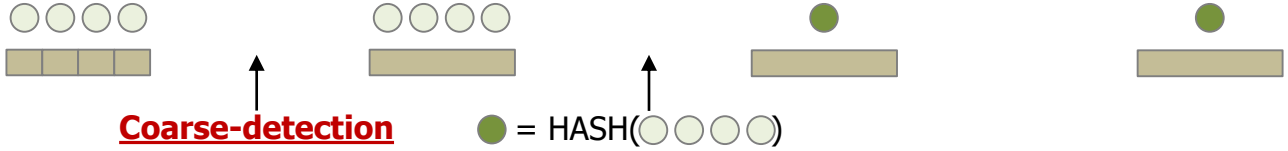
$$Coarse\ MAC = HASH\ (Fine\ MACs)$$

* Scale-up

# Lazy Switching Overhead by MAC

$$Coarse\ MAC = HASH\ (Fine\ MACs)$$

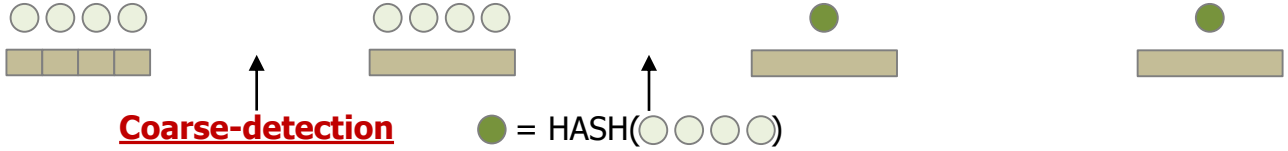
* Scale-up



# Lazy Switching Overhead by MAC

$$Coarse\ MAC = HASH\ (Fine\ MACs)$$

* Scale-up

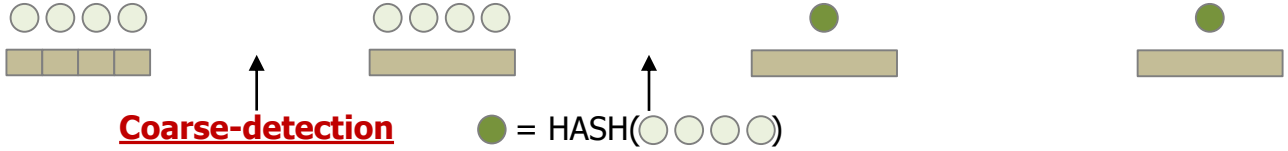



* Scale-down

# Lazy Switching Overhead by MAC

$$Coarse\ MAC = HASH\ (Fine\ MACs)$$

* Scale-up

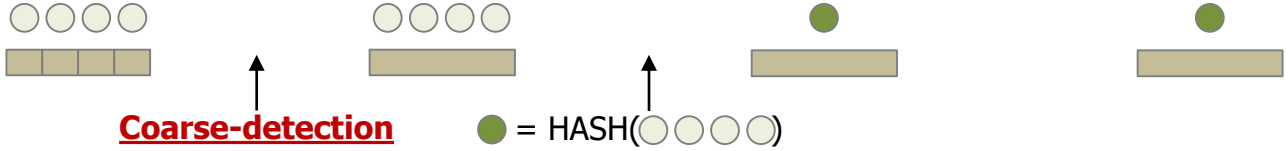



* Scale-down

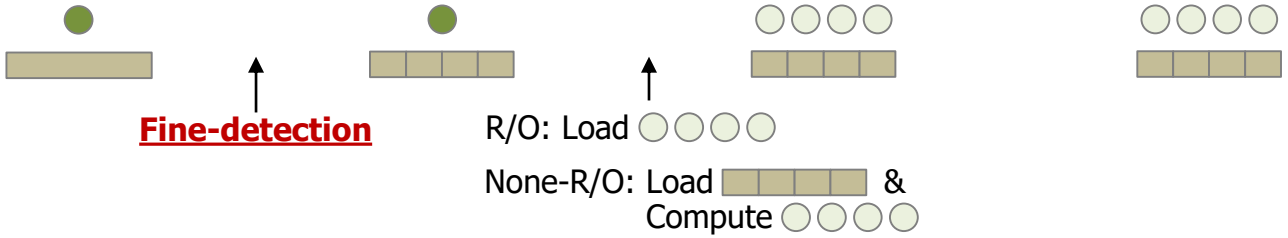
# Lazy Switching Overhead by MAC

$$Coarse\ MAC = HASH\ (Fine\ MACs)$$

* Scale-up



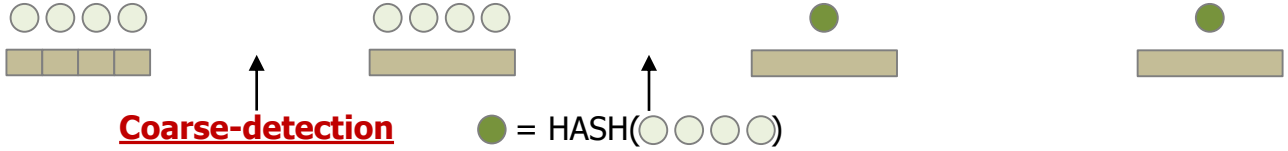

* Scale-down



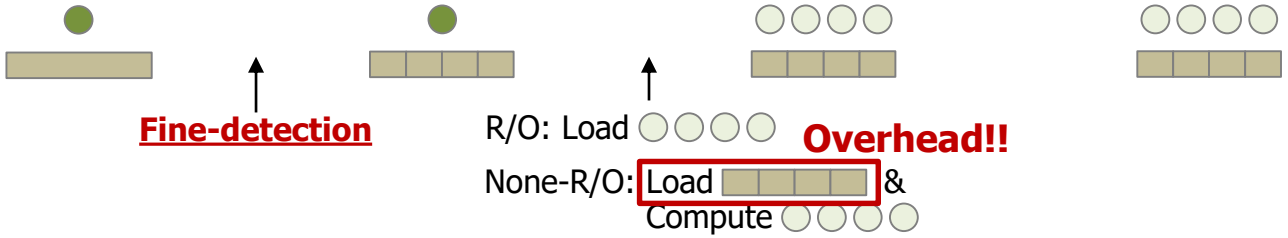
# Lazy Switching Overhead by MAC

$$Coarse\ MAC = HASH\ (Fine\ MACs)$$

* Scale-up




* Scale-down

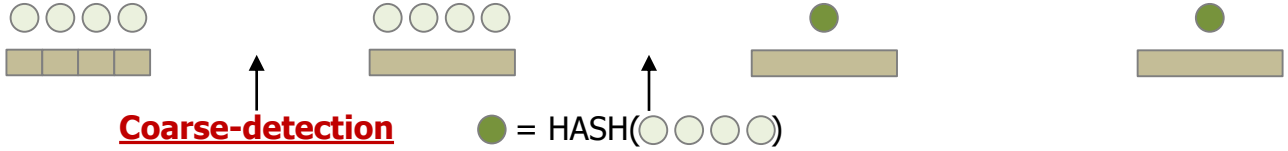


# Lazy Switching Overhead by MAC

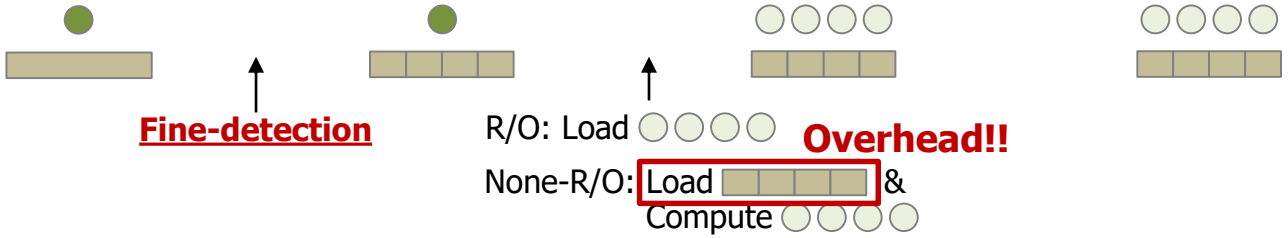
- **97.2%** of reqs → Hidden by **lazy switching & R/O**
  - Only **2.8%** of reqs makes moderate overhead (ld data chunks)

$$Coarse\ MAC = HASH\ (Fine\ MACs)$$

* Scale-up




* Scale-down

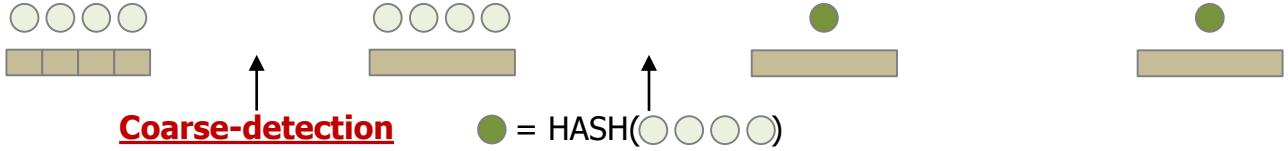


# Lazy Switching Overhead by MAC

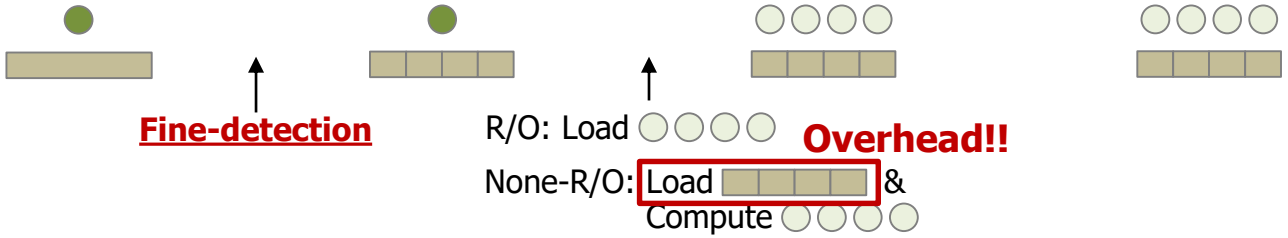
- **97.2%** of reqs → Hidden by **lazy switching & R/O**
  - Only **2.8%** of reqs makes moderate overhead (ld data chunks)

$$Coarse\ MAC = HASH\ (Fine\ MACs)$$

* Scale-up




* Scale-down



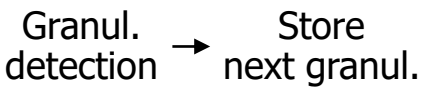

**Lazy switching considerably reduces switching overhead!!**

# Lazy Switching Overhead by CTR Tree

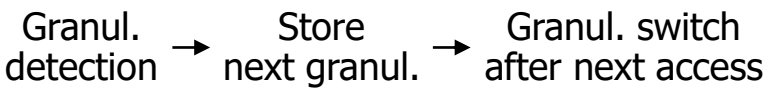
# Lazy Switching Overhead by CTR Tree

Granul.
detection

# Lazy Switching Overhead by CTR Tree

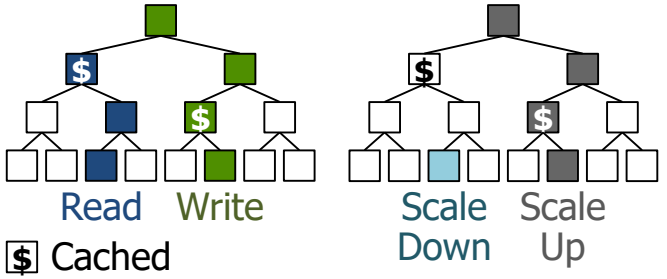


# Lazy Switching Overhead by CTR Tree



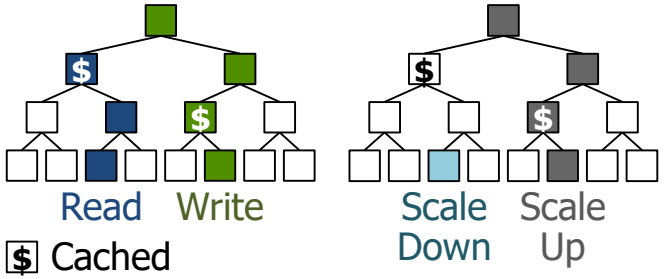
# Lazy Switching Overhead by CTR Tree

Granul. detection → Store next granul. → Granul. switch after next access

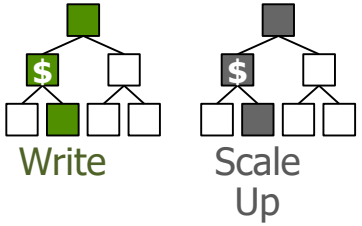


# Lazy Switching Overhead by CTR Tree

Granul. detection → Store next granul. → Granul. switch after next access

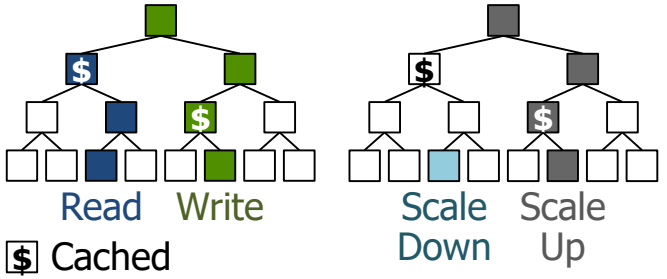



* Scale-up with WAR/WAW

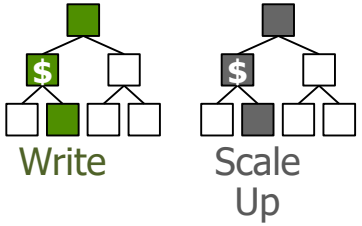


# Lazy Switching Overhead by CTR Tree

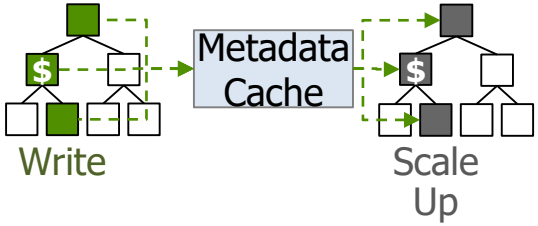
Granul. detection → Store next granul. → Granul. switch after next access




* Scale-up with WAR/WAW

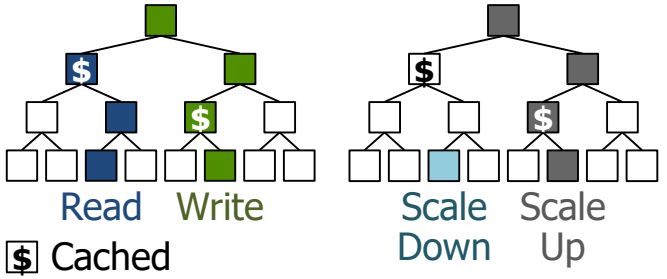



* Scale-up with RAW

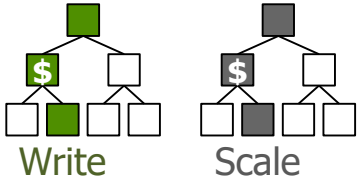


# Lazy Switching Overhead by CTR Tree

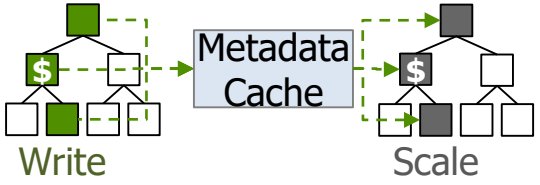
Granul. detection → Store next granul. → Granul. switch after next access



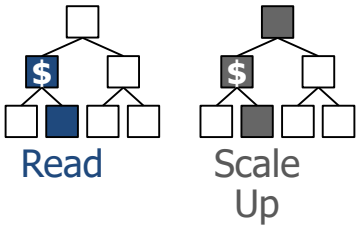

* Scale-up with WAR/WAW




* Scale-up with RAW

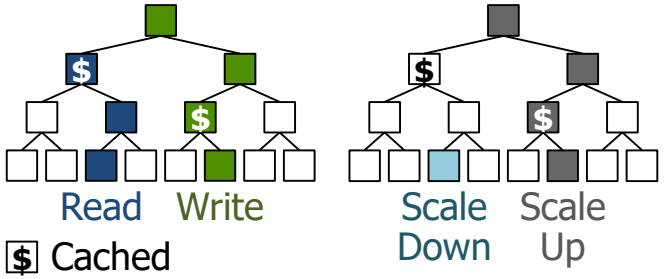



* Scale-up with RAR

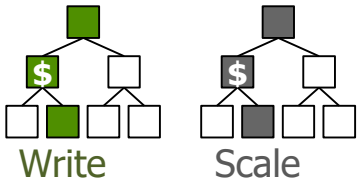


# Lazy Switching Overhead by CTR Tree

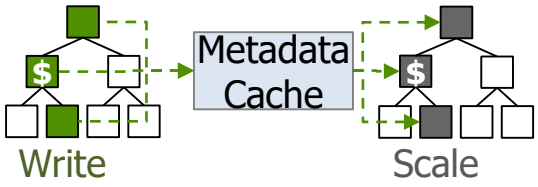
Granul. detection → Store next granul. → Granul. switch after next access



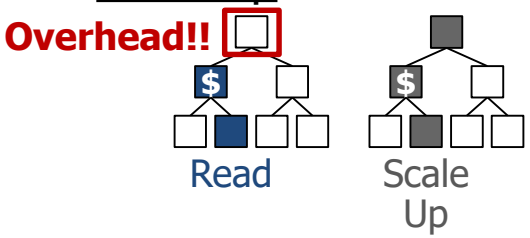

* Scale-up with WAR/WAW




* Scale-up with RAW

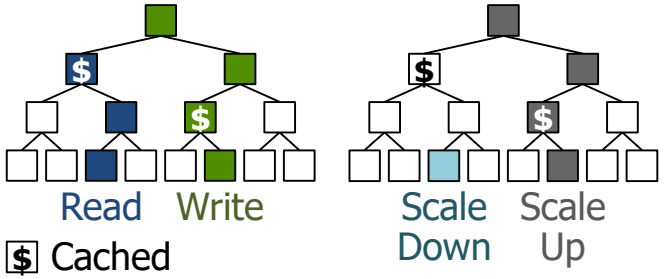



* Scale-up with RAR

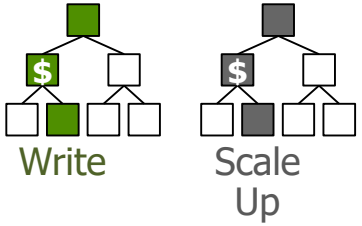


# Lazy Switching Overhead by CTR Tree

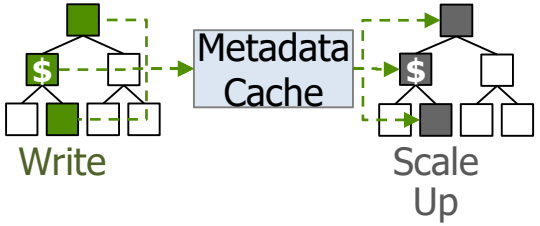
Granul. detection → Store next granul. → Granul. switch after next access



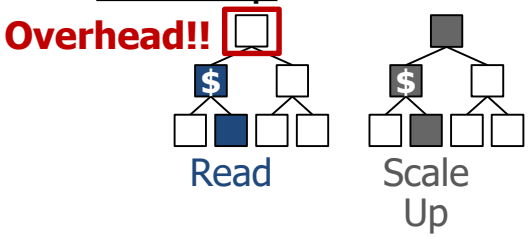

* Scale-up with WAR/WAW



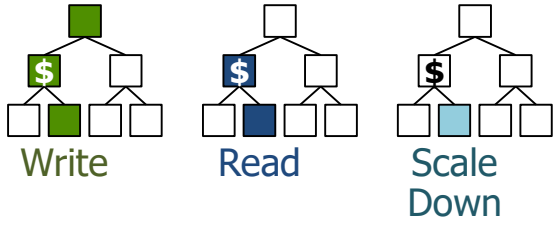

* Scale-up with RAW




* Scale-up with RAR




* Scale-down

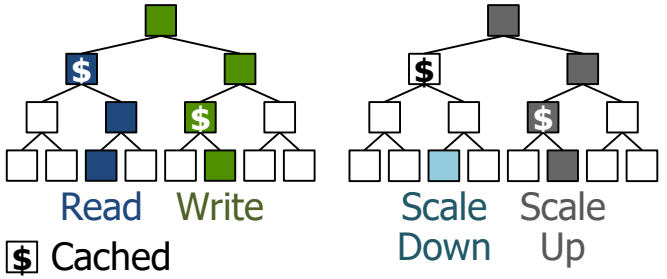



Backup

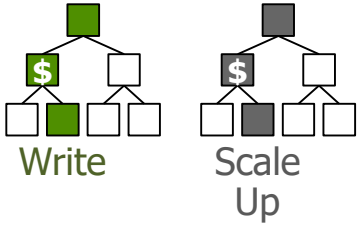
# Lazy Switching Overhead by CTR Tree

- **91.2%** of reqs → Hidden by **lazy switching**
  - Only **8.2%** of reqs makes low overhead (read req → write req)

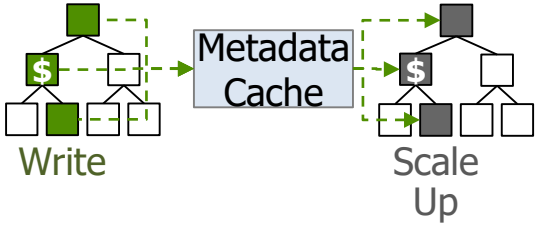
Granul. detection → Store next granul. → Granul. switch after next access



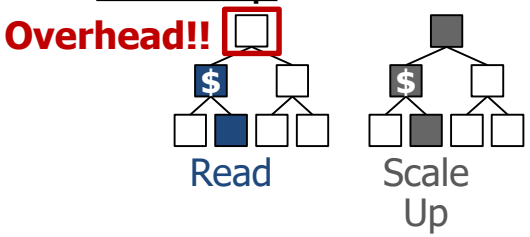

* Scale-up with WAR/WAW



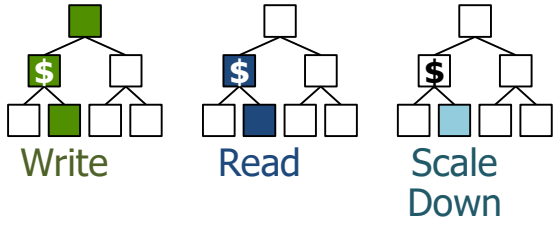

* Scale-up with RAW




* Scale-up with RAR




* Scale-down




Backup

# Pros of Matching Granularity

# Pros of Matching Granularity

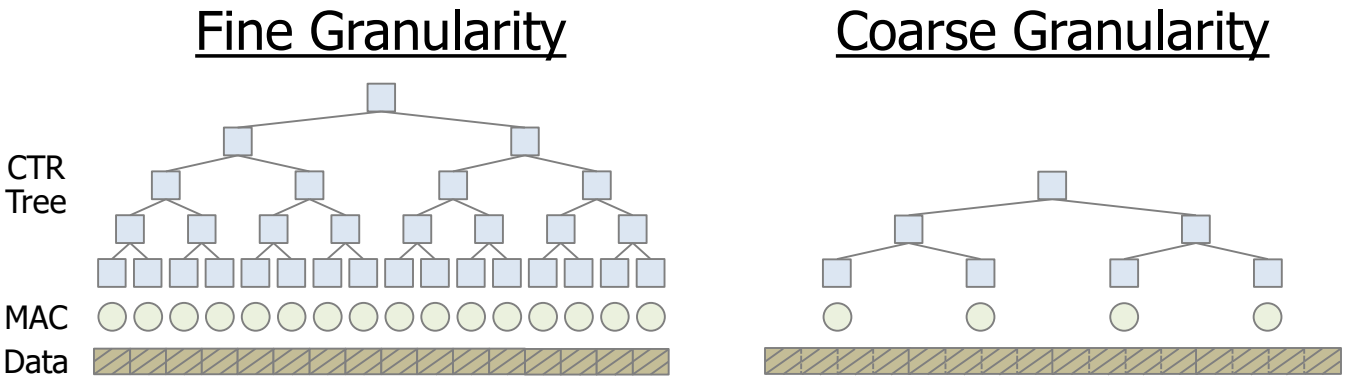
- Proper granularity → Reduce security metadata

# Pros of Matching Granularity

- Proper granularity → Reduce security metadata
- Wrong granularity → Data load penalty

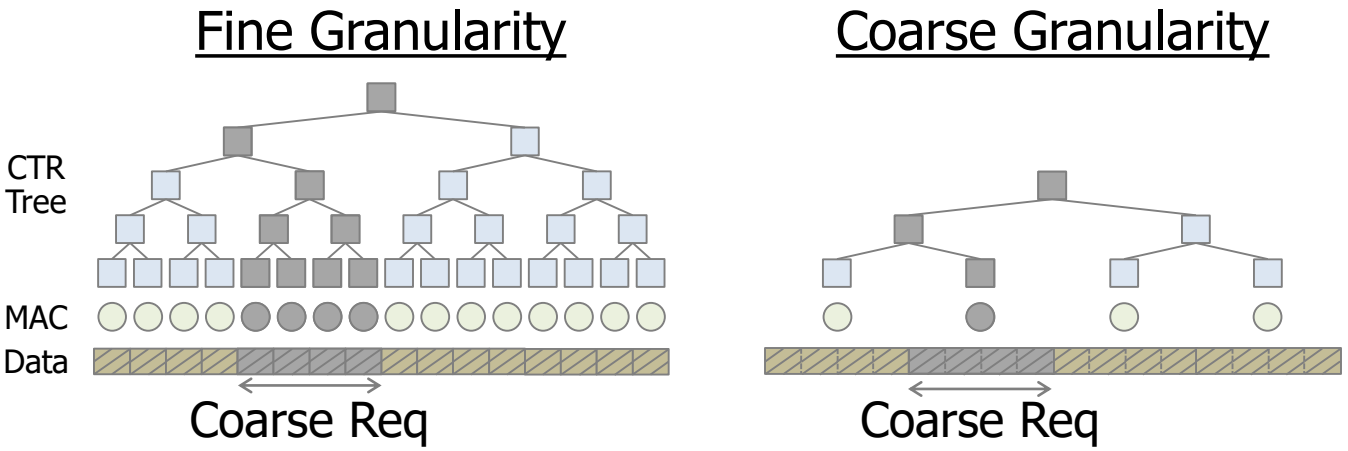
# Pros of Matching Granularity

- Proper granularity → Reduce security metadata
- Wrong granularity → Data load penalty



# Pros of Matching Granularity

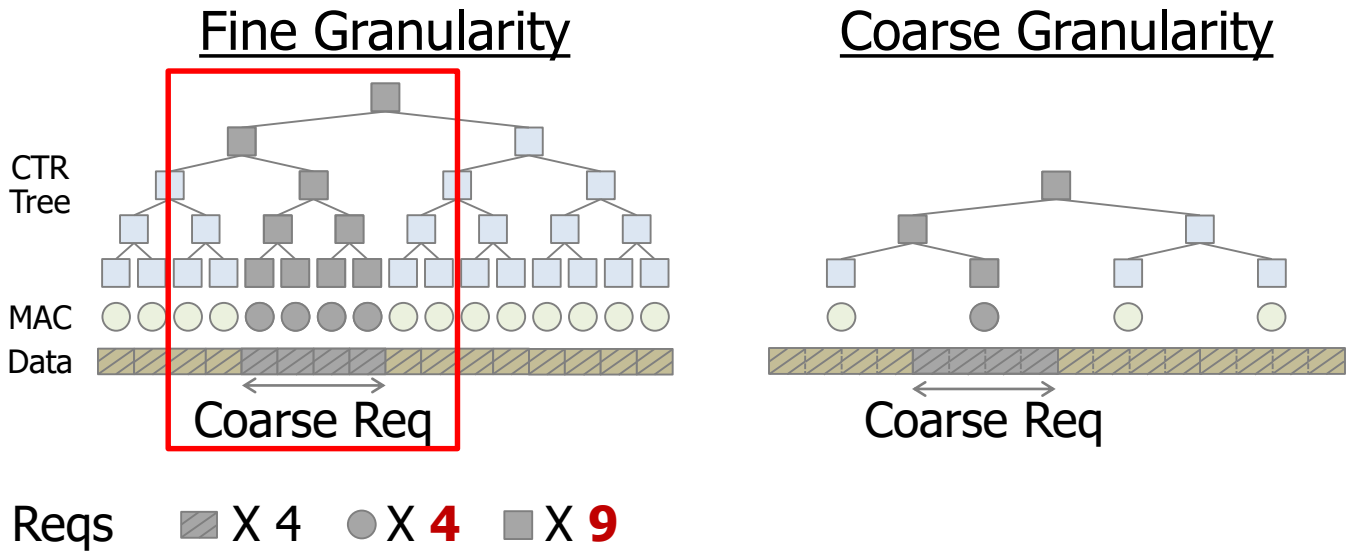
- Proper granularity → Reduce security metadata
- Wrong granularity → Data load penalty




Coarse Reqs

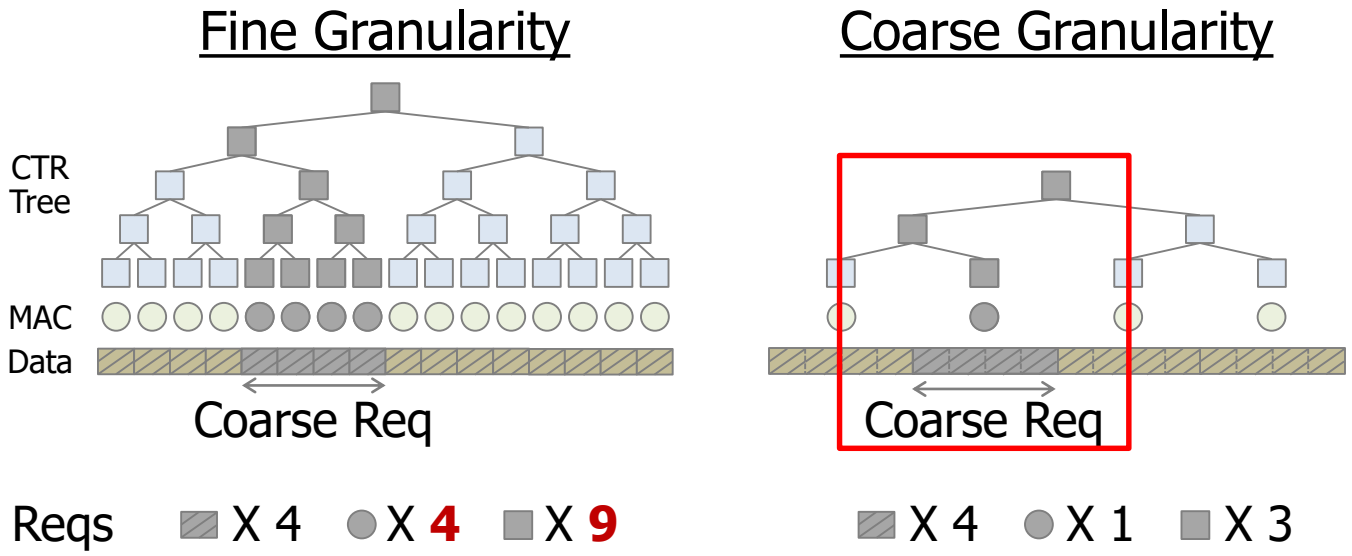
# Pros of Matching Granularity

- Proper granularity → Reduce security metadata
- Wrong granularity → Data load penalty



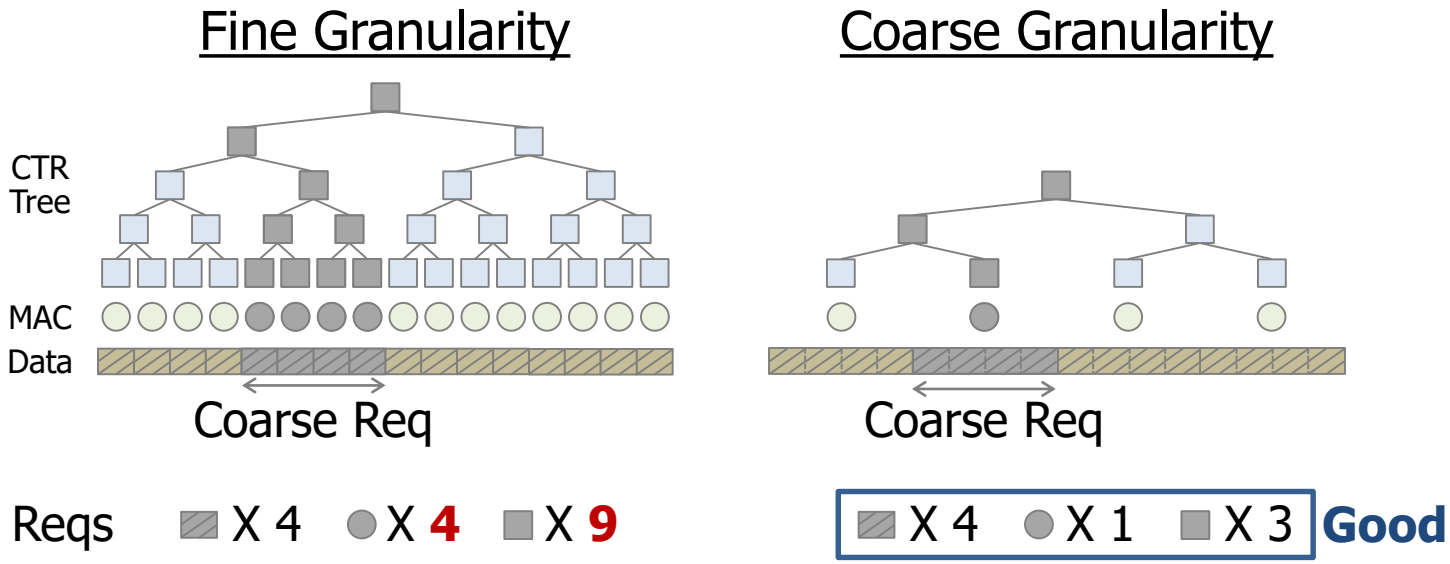
# Pros of Matching Granularity

- Proper granularity → Reduce security metadata
- Wrong granularity → Data load penalty



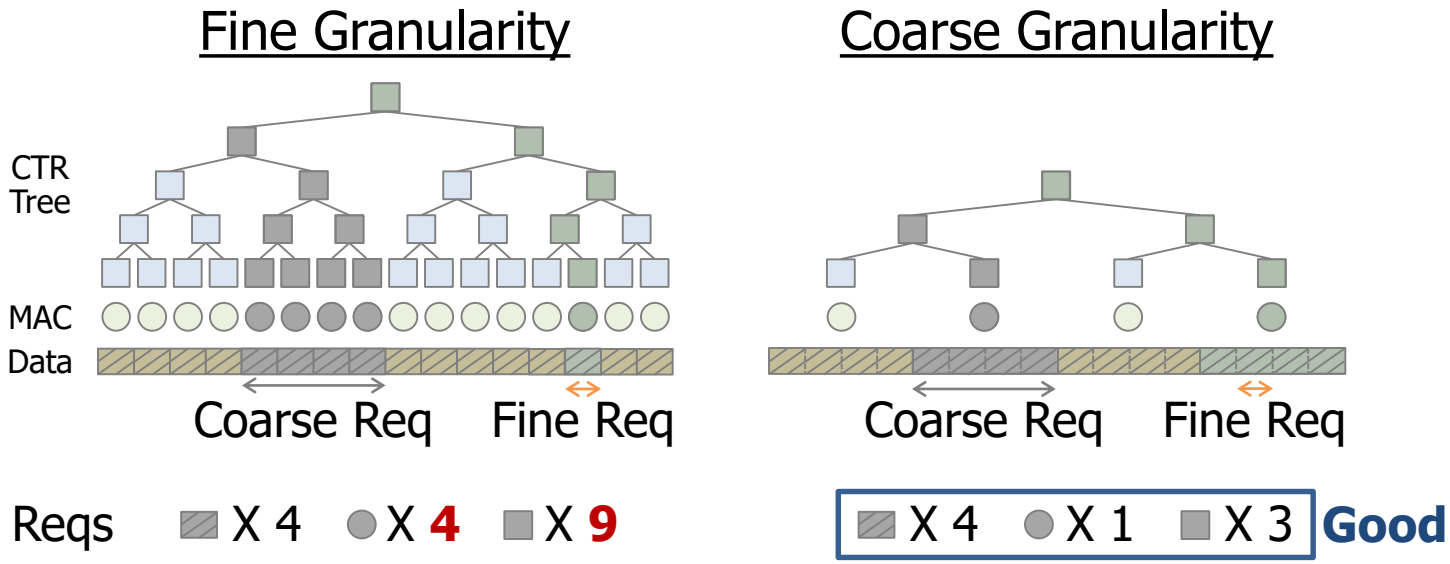
# Pros of Matching Granularity

- Proper granularity → Reduce security metadata
- Wrong granularity → Data load penalty



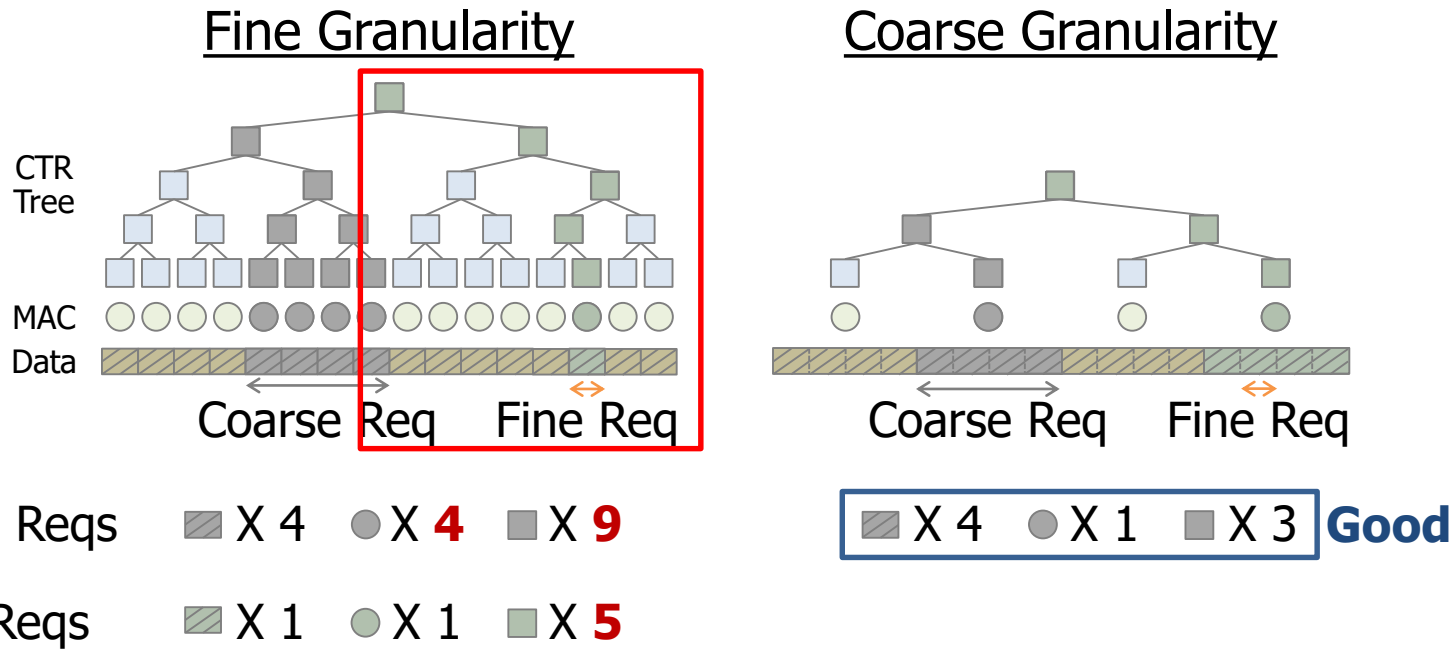
# Pros of Matching Granularity

- Proper granularity → Reduce security metadata
- Wrong granularity → Data load penalty



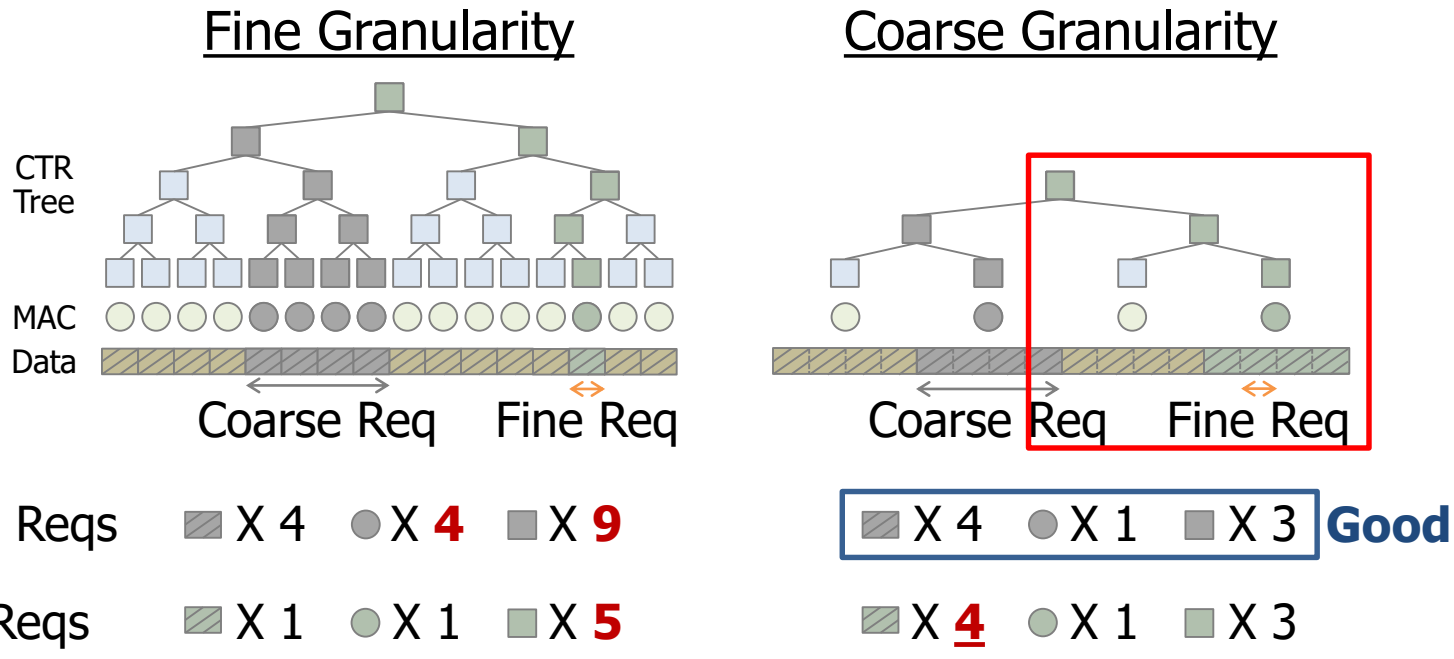
# Pros of Matching Granularity

- Proper granularity → Reduce security metadata
- Wrong granularity → Data load penalty



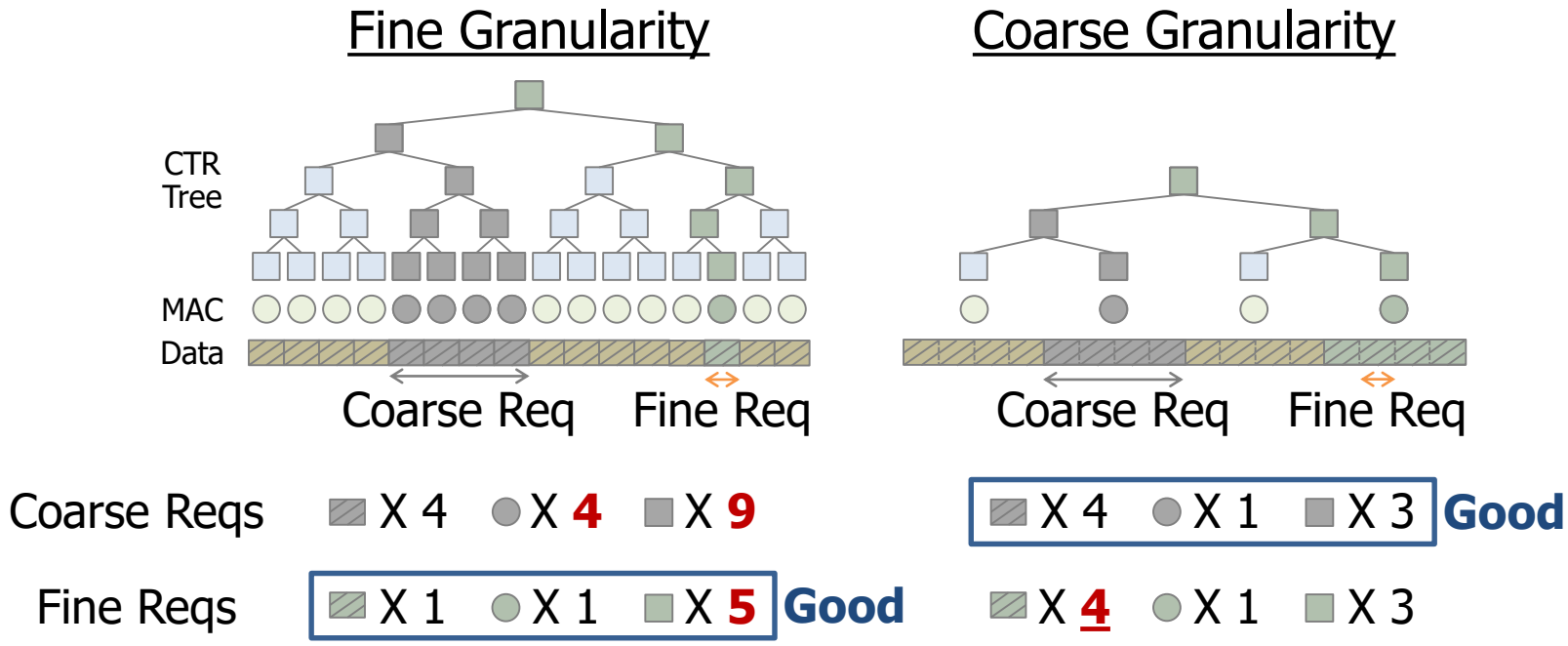
# Pros of Matching Granularity

- Proper granularity → Reduce security metadata
- Wrong granularity → Data load penalty



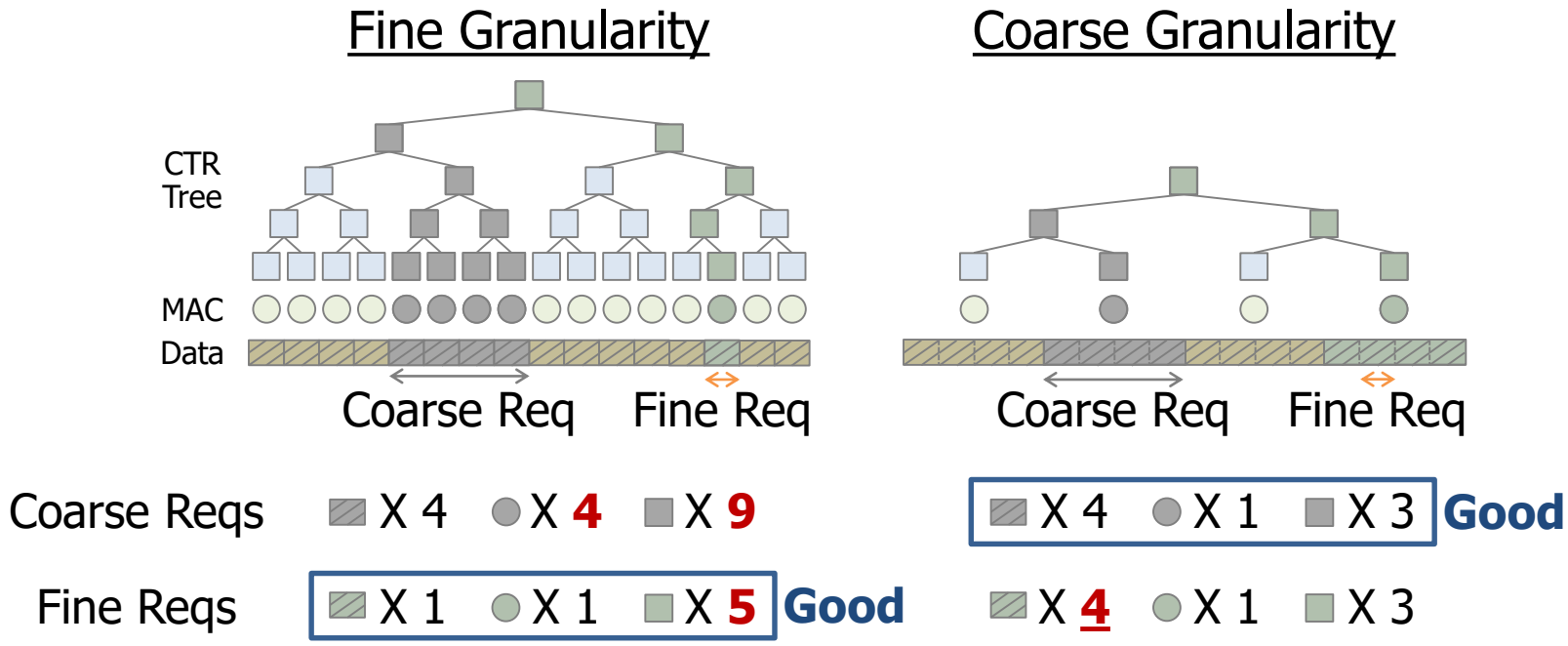
# Pros of Matching Granularity

- Proper granularity → Reduce security metadata
- Wrong granularity → Data load penalty



# Pros of Matching Granularity

- Proper granularity → Reduce security metadata
- Wrong granularity → Data load penalty




**Granularity-managed MAC&tree makes efficient memory protection**

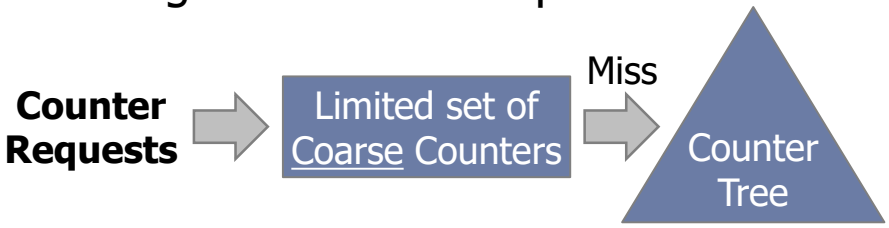
# Prior Domain-specific Memory Protections

- No prior study using **integrity tree pruning** or **multi-granular MAC&counter**

# Prior Domain-specific Memory Protections

- No prior study using **integrity tree pruning** or **multi-granular MAC&counter**

1. Dual-granular & GPU-optimized Counter [1]



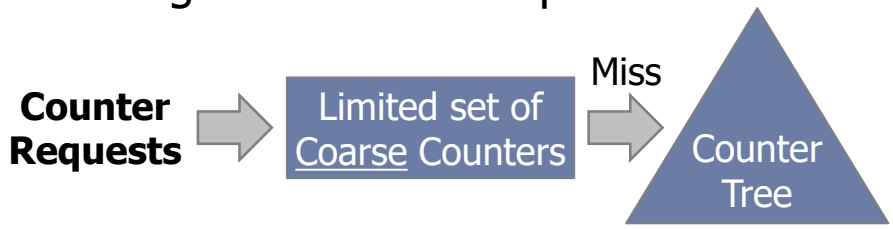

Dual CTRs, limited CTRs, CTR-only, device-specific

[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)

# Prior Domain-specific Memory Protections

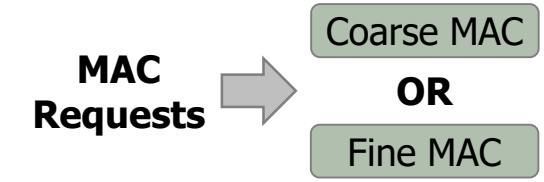
- No prior study using **integrity tree pruning** or **multi-granular MAC&counter**

1. Dual-granular & GPU-optimized Counter [1]




Dual CTRs, limited CTRs, CTR-only, device-specific

2. Dual-granular MAC [2]



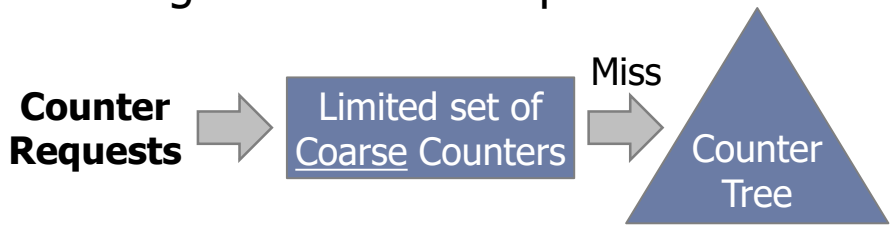

Dual MACs, MAC-only

[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)

# Prior Domain-specific Memory Protections

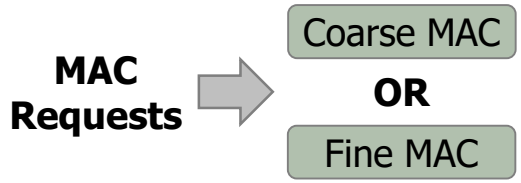
- No prior study using **integrity tree pruning** or **multi-granular MAC&counter**

1. Dual-granular & GPU-optimized Counter [1]



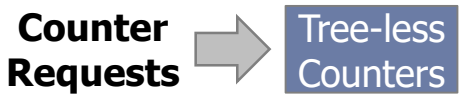

Dual CTRs, limited CTRs, CTR-only, device-specific

2. Dual-granular MAC [2]




Dual MACs, MAC-only

3. S/W Counter [3-6]



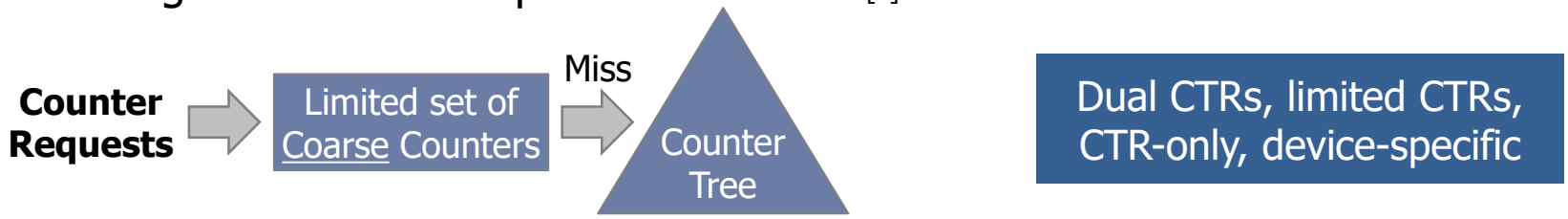

Domain-specific


[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)
[3] TNPU: Supporting Trusted Execution with Tree-less Integrity Protection for Neural Processing Unit (HPCA 2022)
[4] MGX: Near-zero Overhead Memory Protection for Data-intensive Accelerators (ISCA 2022)
[5] GuardNN: Secure Accelerator Architecture for Privacy-preserving Deep Learning (DAC 2022)
[6] SoftVN: Efficient Memory Protection via Software-provided Version Numbers (ISCA 2022)

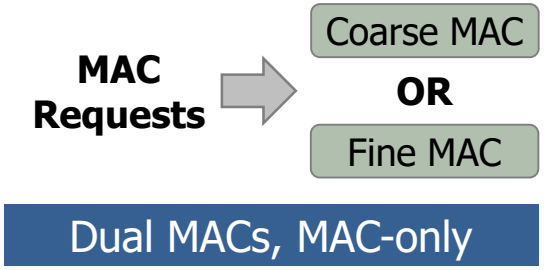
# Prior Domain-specific Memory Protections

- No prior study using **integrity tree pruning** or **multi-granular MAC&counter**

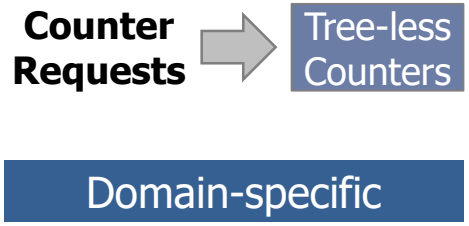
1. Dual-granular & GPU-optimized Counter [1]



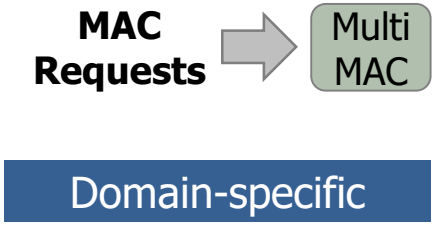

2. Dual-granular MAC [2]




3. S/W Counter [3-6]




4. S/W MAC [4]

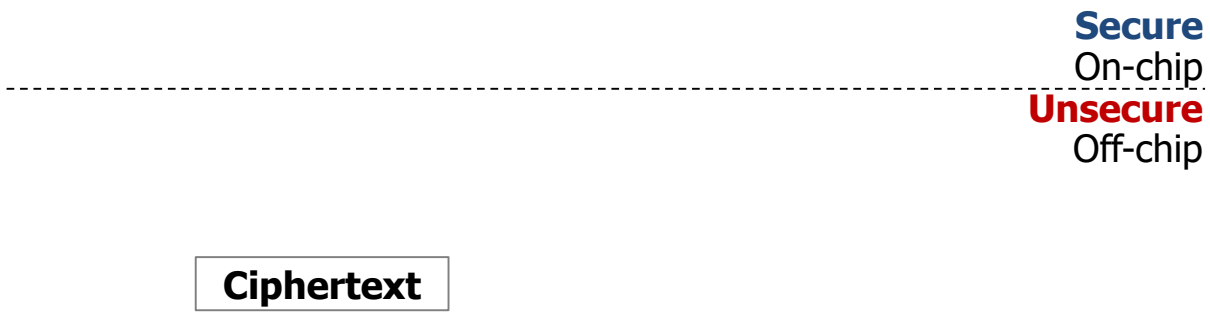




[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)
[3] TNPU: Supporting Trusted Execution with Tree-less Integrity Protection for Neural Processing Unit (HPCA 2022)
[4] MGX: Near-zero Overhead Memory Protection for Data-intensive Accelerators (ISCA 2022)
[5] GuardNN: Secure Accelerator Architecture for Privacy-preserving Deep Learning (DAC 2022)
[6] SoftVN: Efficient Memory Protection via Software-provided Version Numbers (ISCA 2022)

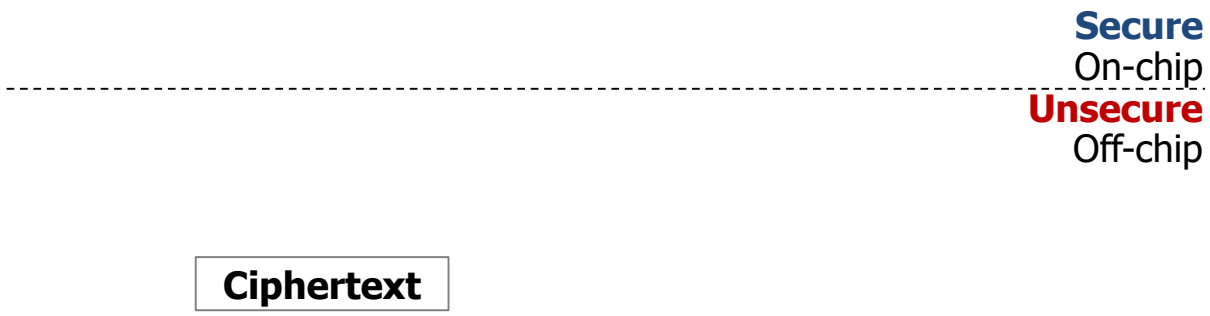
# Detailed of Counter-mode Protection

# Detailed of Counter-mode Protection



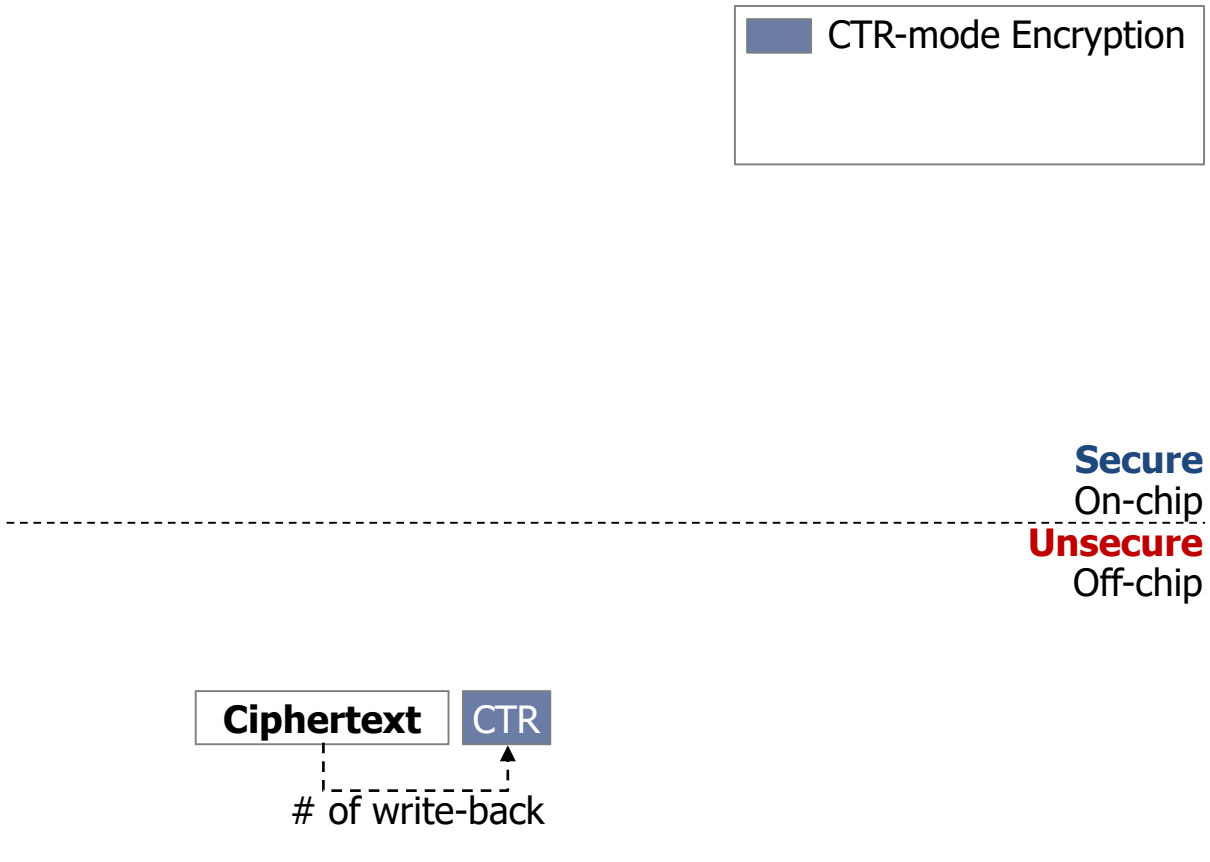
# Detailed of Counter-mode Protection

- CTR-mode encryption: confidentiality



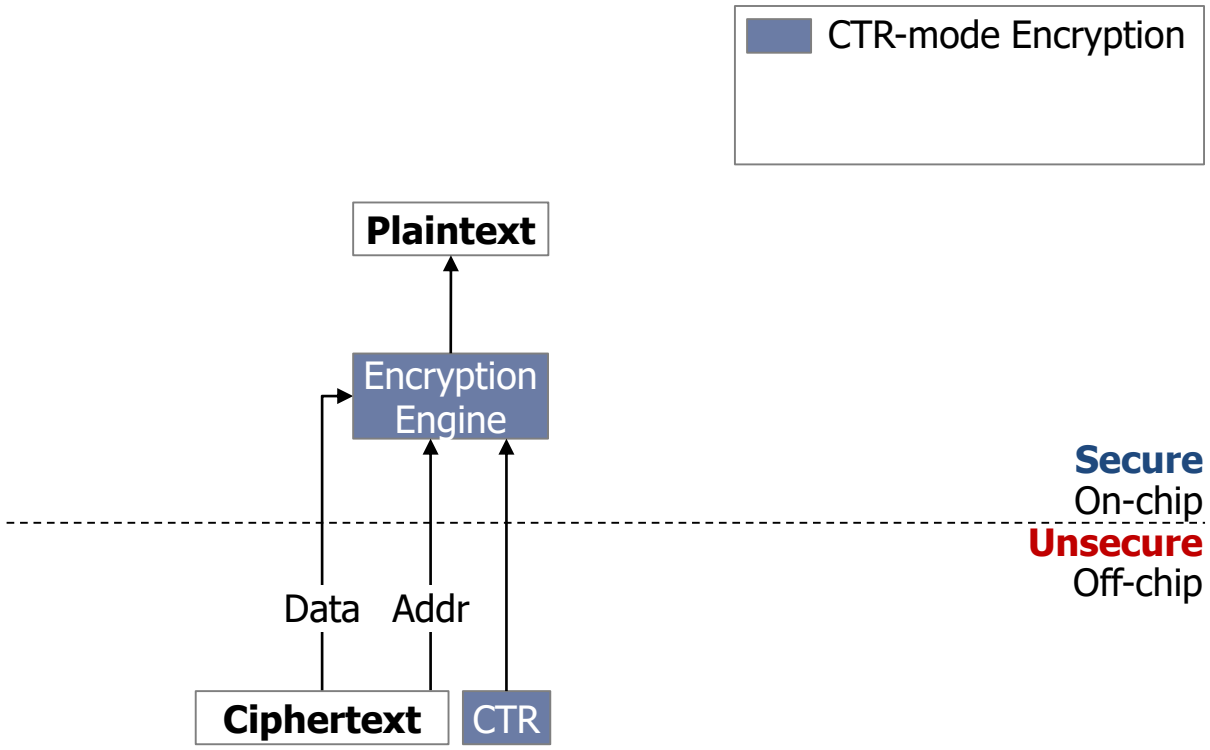
# Detailed of Counter-mode Protection

- CTR-mode encryption: confidentiality



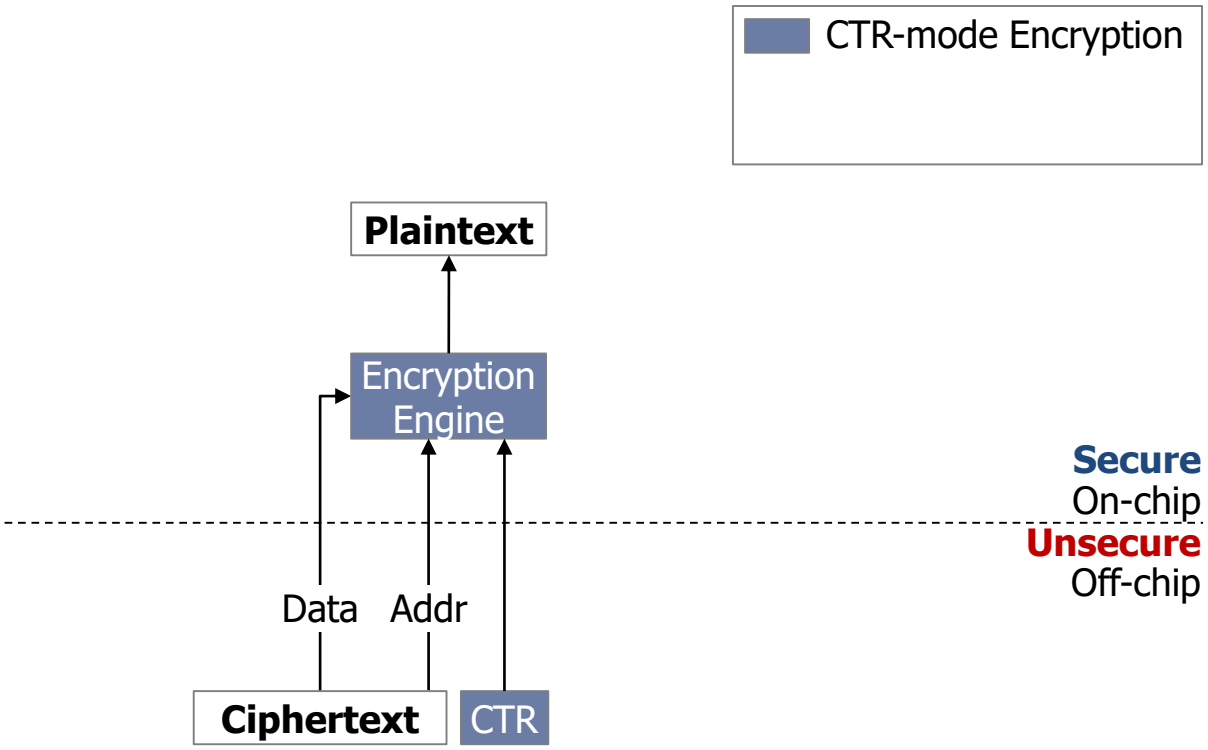
# Detailed of Counter-mode Protection

- CTR-mode encryption: confidentiality



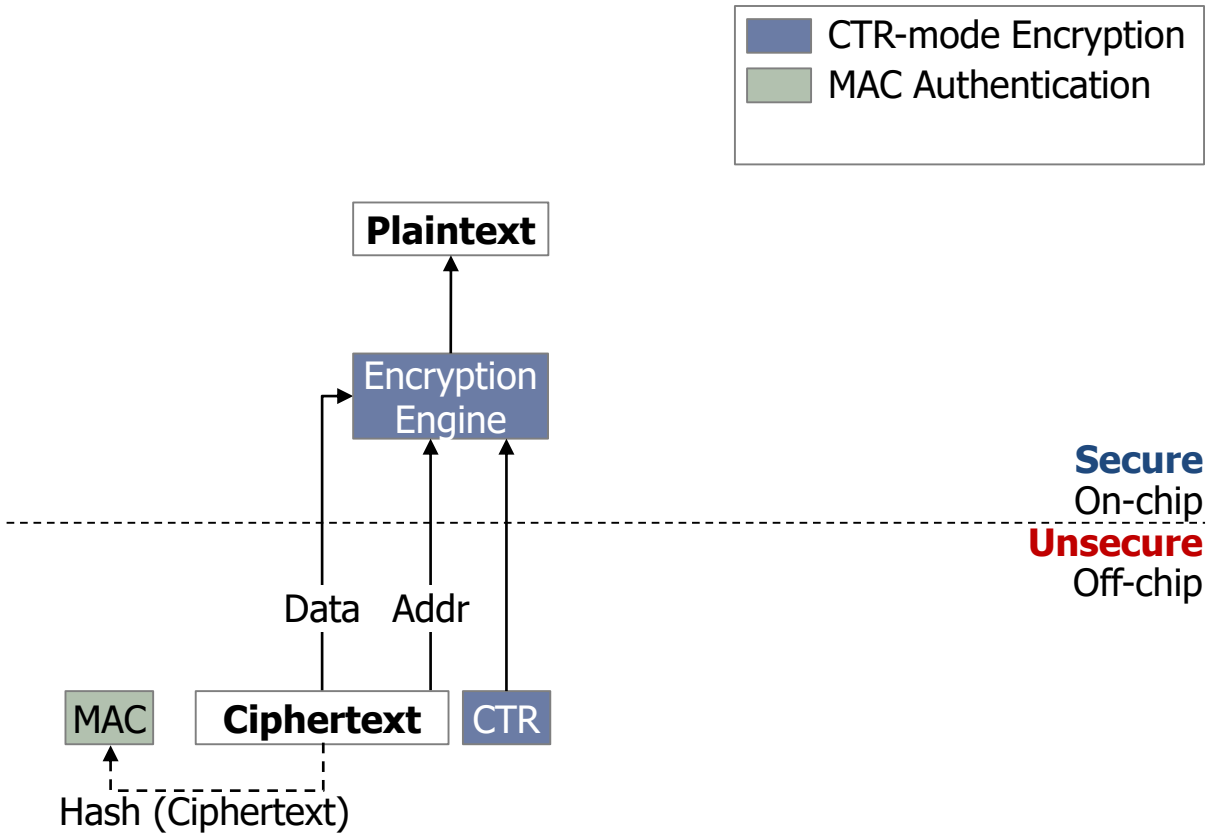
# Detailed of Counter-mode Protection

- CTR-mode encryption: confidentiality
- MAC authentication: value-integrity



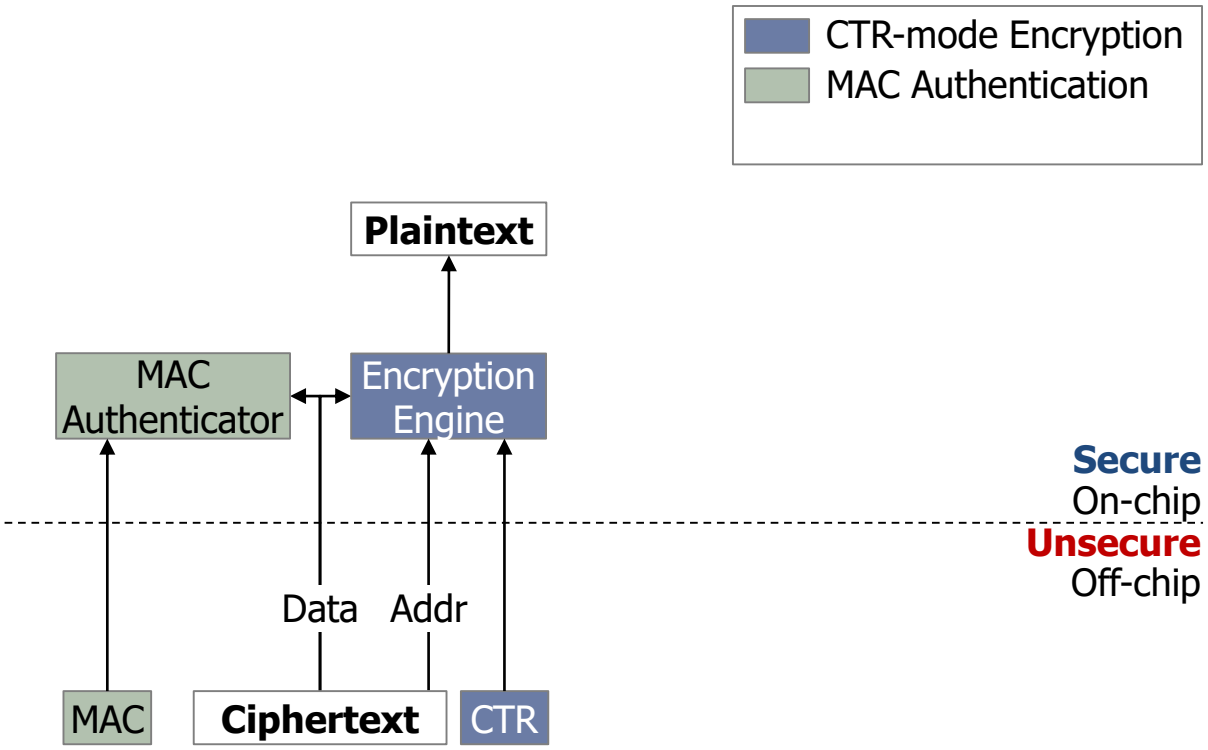
# Detailed of Counter-mode Protection

- CTR-mode encryption: confidentiality
- MAC authentication: value-integrity



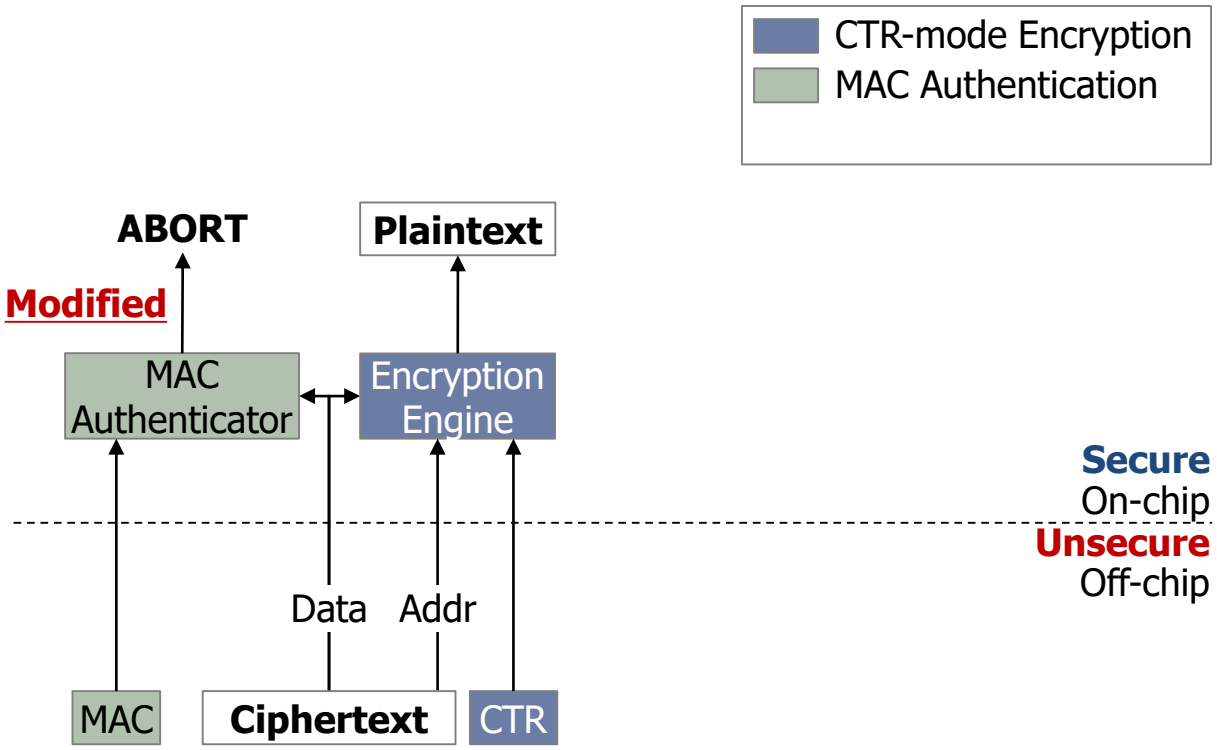
# Detailed of Counter-mode Protection

- CTR-mode encryption: confidentiality
- MAC authentication: value-integrity



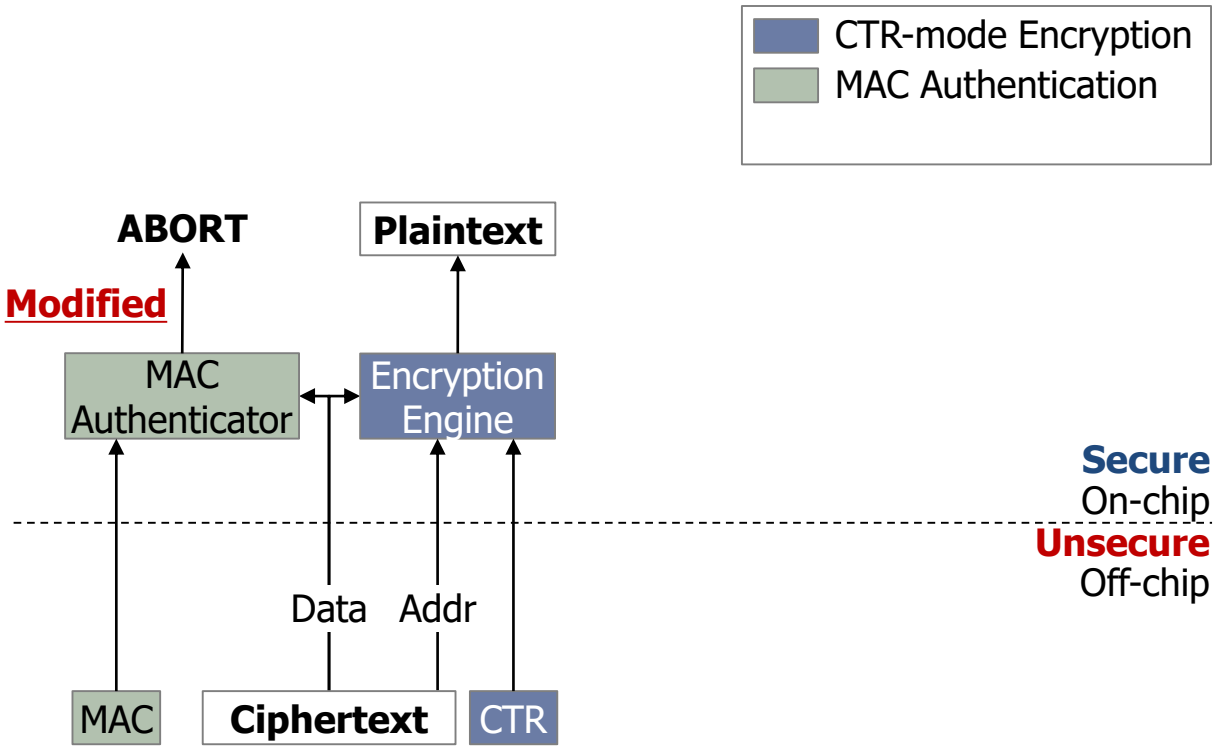
# Detailed of Counter-mode Protection

- CTR-mode encryption: confidentiality
- MAC authentication: value-integrity



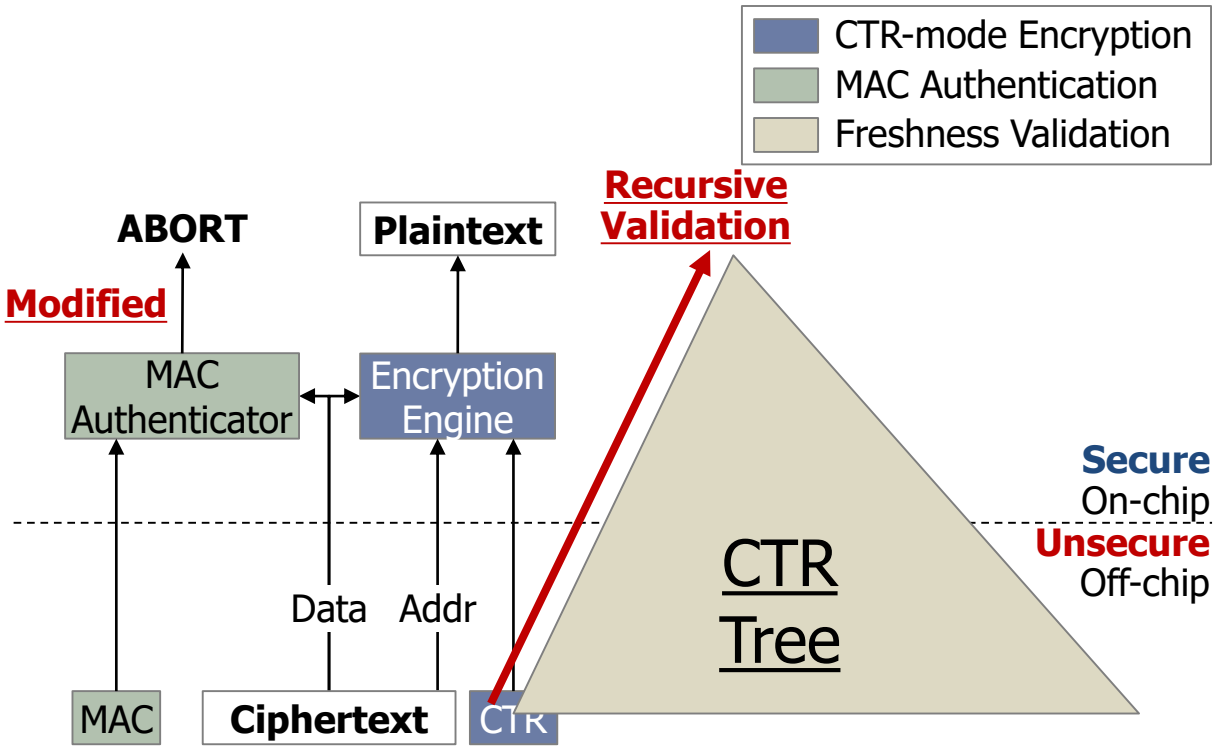
# Detailed of Counter-mode Protection

- CTR-mode encryption: confidentiality
- MAC authentication: value-integrity
- Freshness validation: replay-attack



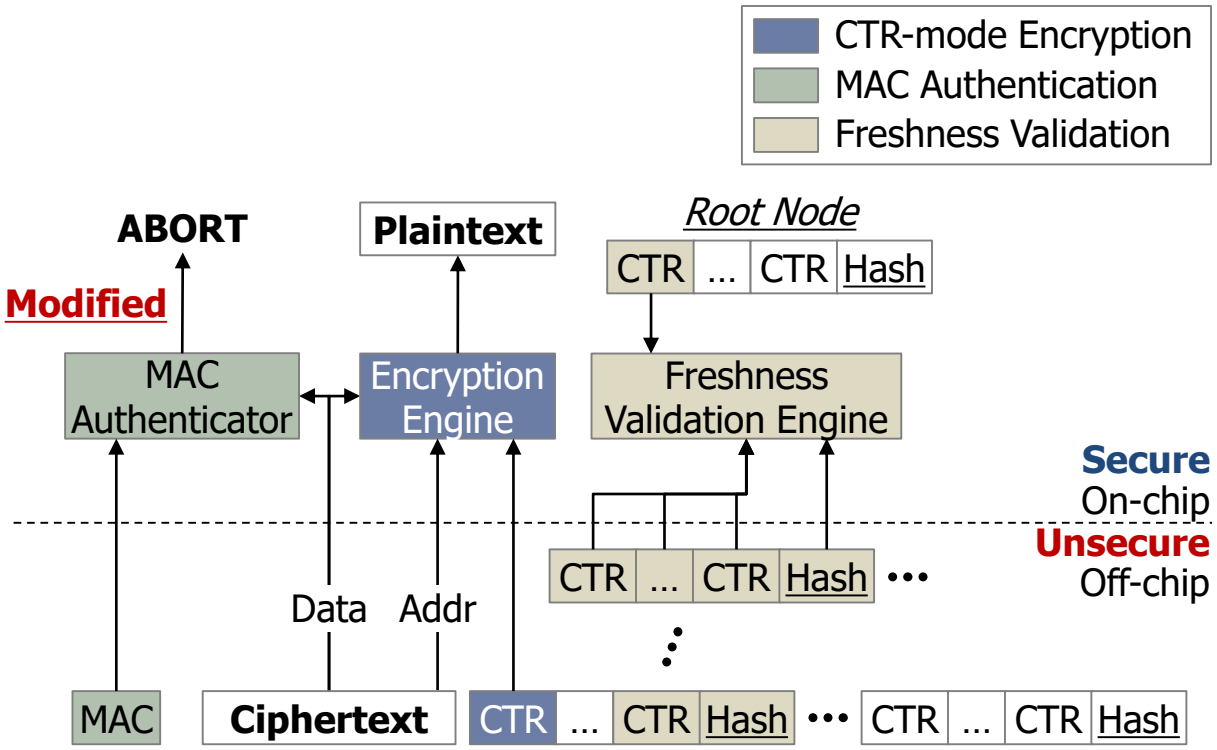
# Detailed of Counter-mode Protection

- CTR-mode encryption: confidentiality
- MAC authentication: value-integrity
- Freshness validation: replay-attack



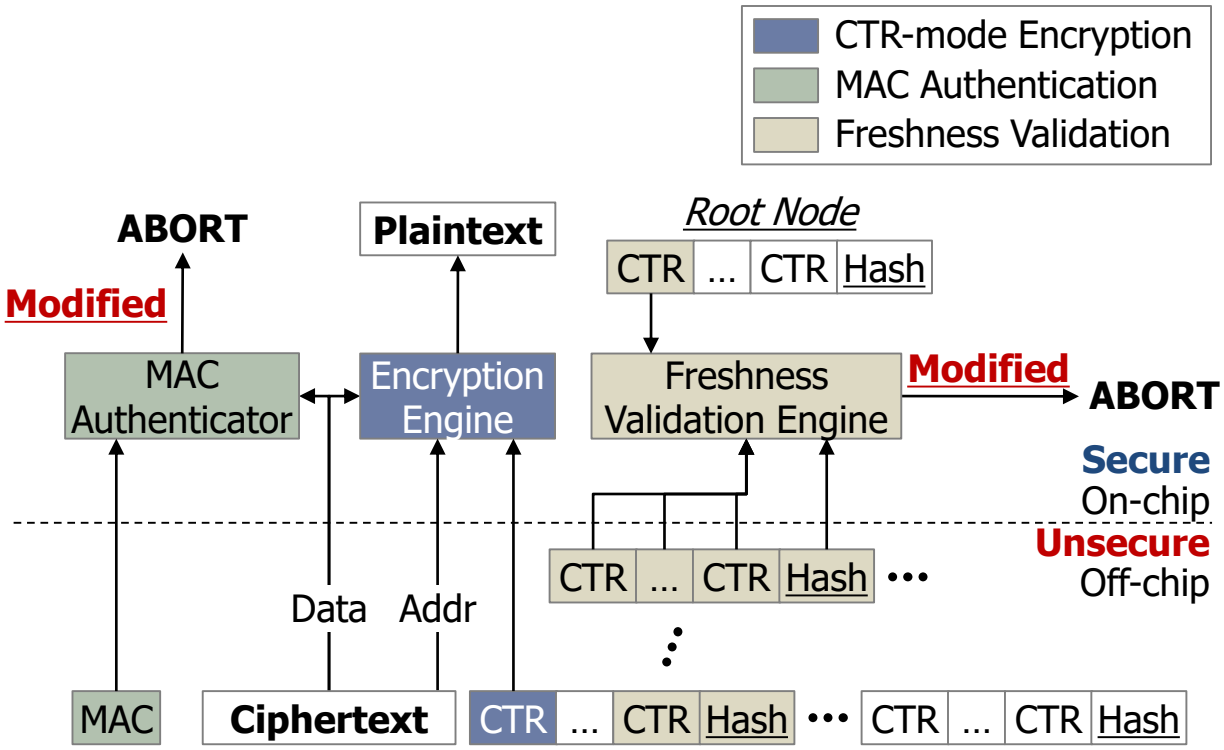
# Detailed of Counter-mode Protection

- CTR-mode encryption: confidentiality
- MAC authentication: value-integrity
- Freshness validation: replay-attack



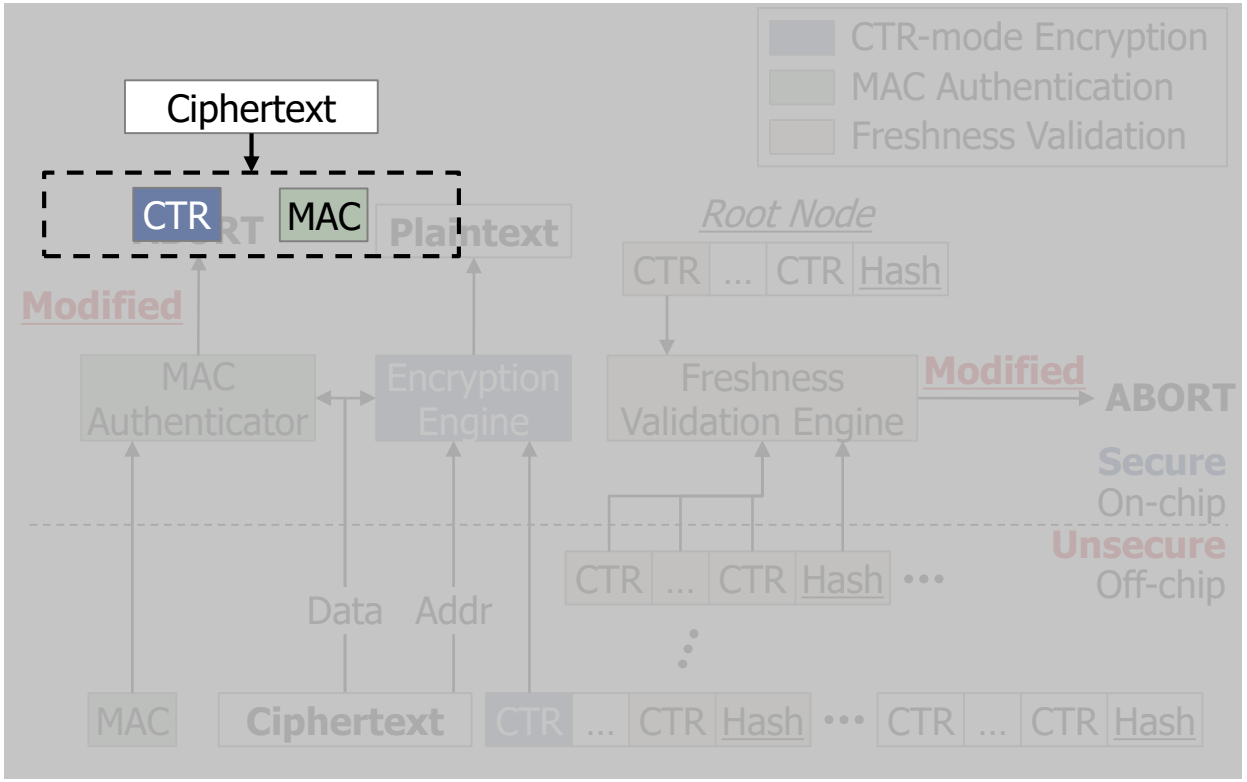
# Detailed of Counter-mode Protection

- CTR-mode encryption: confidentiality
- MAC authentication: value-integrity
- Freshness validation: replay-attack



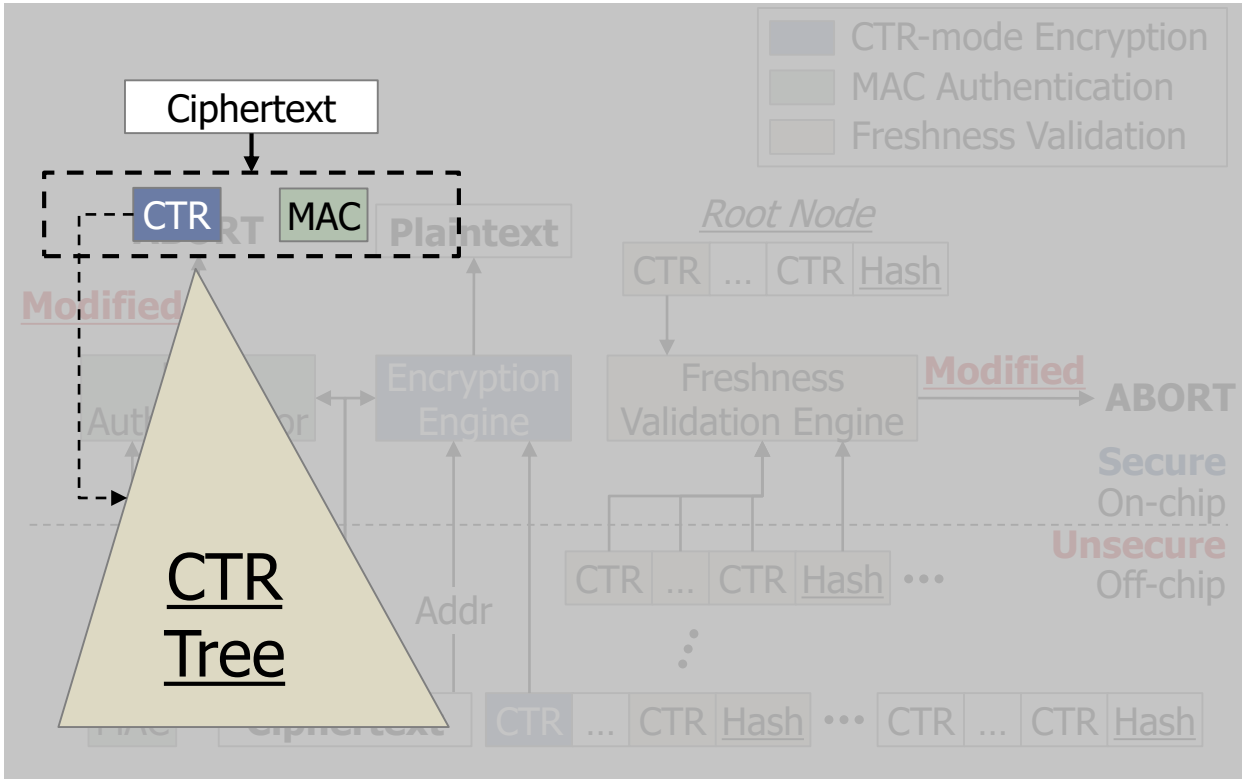
# Detailed of Counter-mode Protection

- CTR-mode encryption: confidentiality
- MAC authentication: value-integrity
- Freshness validation: replay-attack



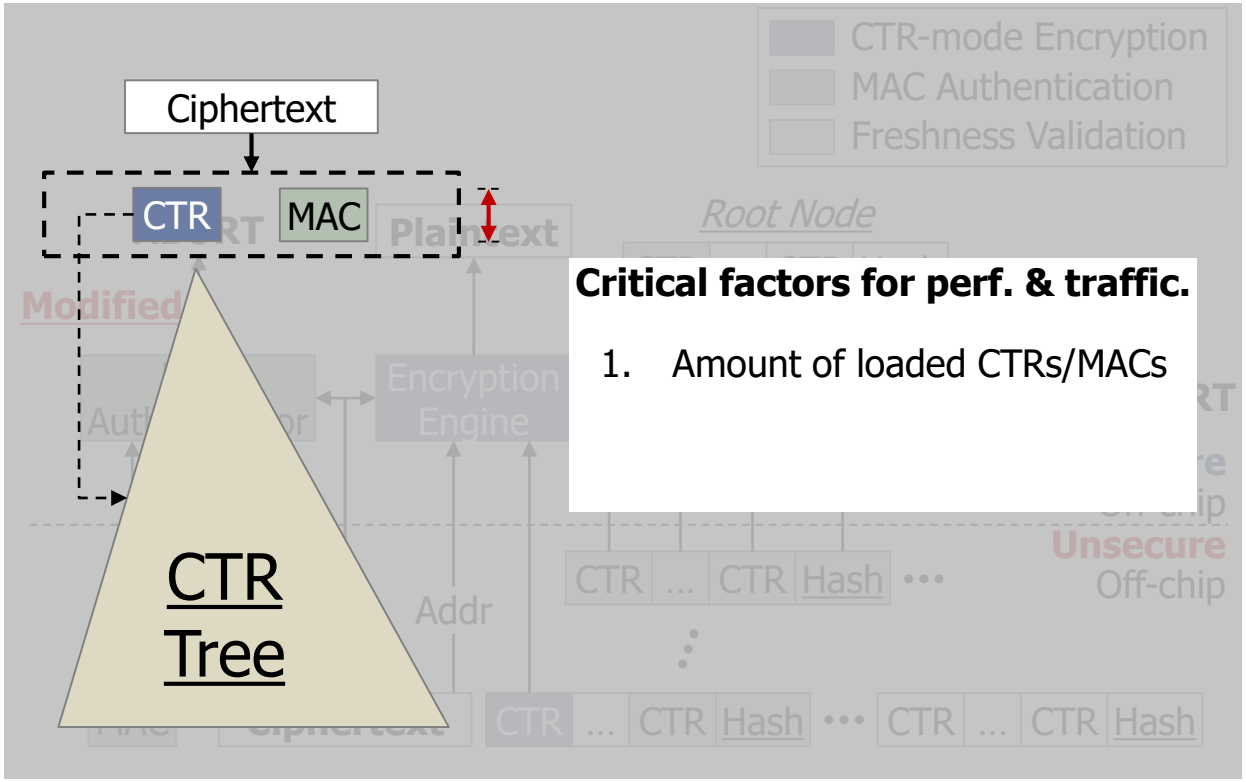
# Detailed of Counter-mode Protection

- CTR-mode encryption: confidentiality
- MAC authentication: value-integrity
- Freshness validation: replay-attack



# Detailed of Counter-mode Protection

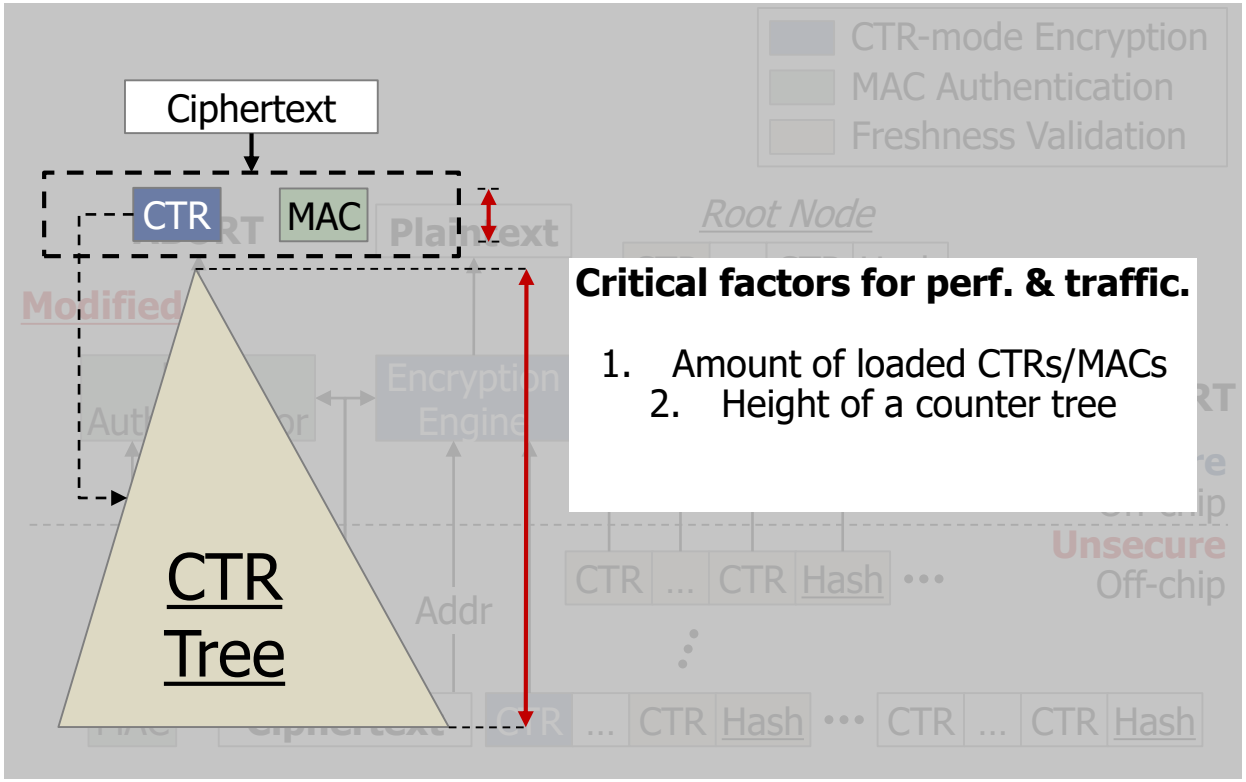
- CTR-mode encryption: confidentiality
- MAC authentication: value-integrity
- Freshness validation: replay-attack




Backup

# Detailed of Counter-mode Protection

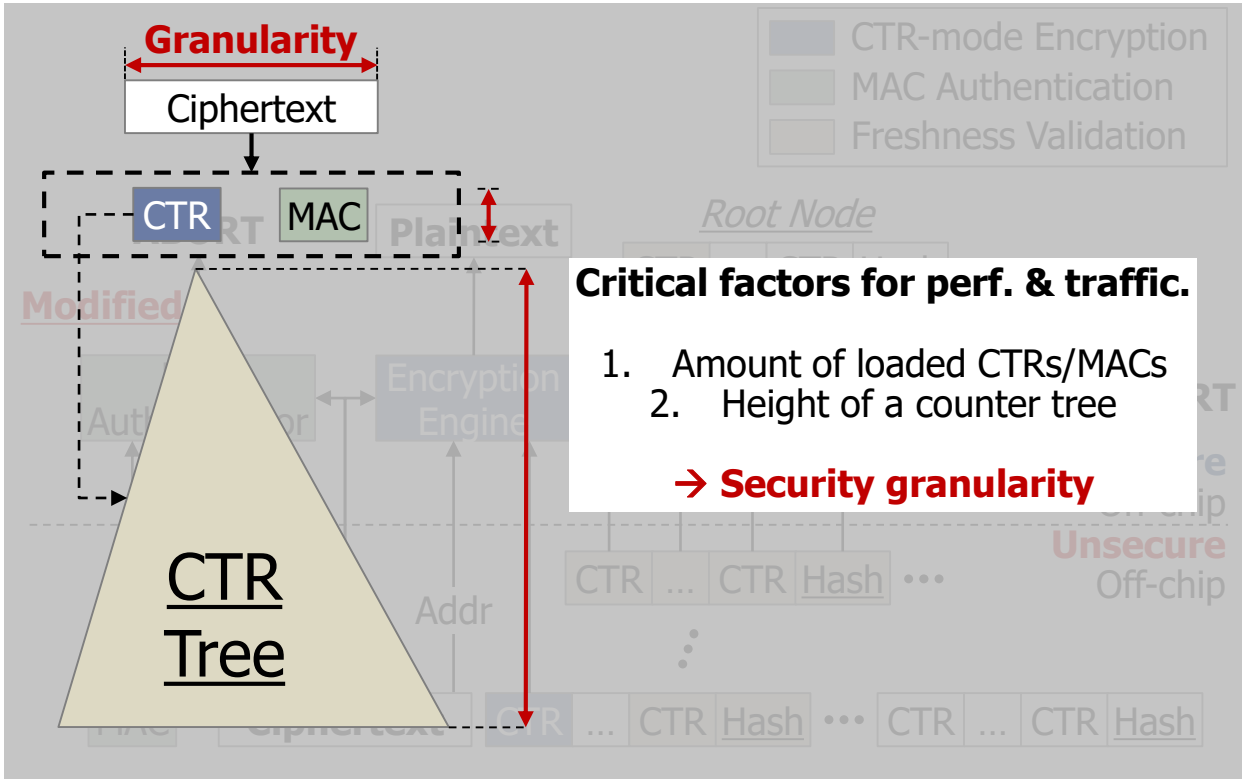
- CTR-mode encryption: confidentiality
- MAC authentication: value-integrity
- Freshness validation: replay-attack




Backup

# Detailed of Counter-mode Protection

- CTR-mode encryption: confidentiality
- MAC authentication: value-integrity
- Freshness validation: replay-attack

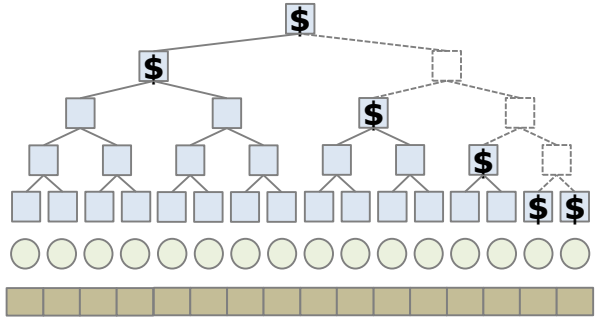


# Combined with Prior CTR Tree Optimization

# Combined with Prior CTR Tree Optimization

- Prior hotness-based integrity tree optimization scheme (Subtree optimization)[1-4]

Prior Subtree Optimization

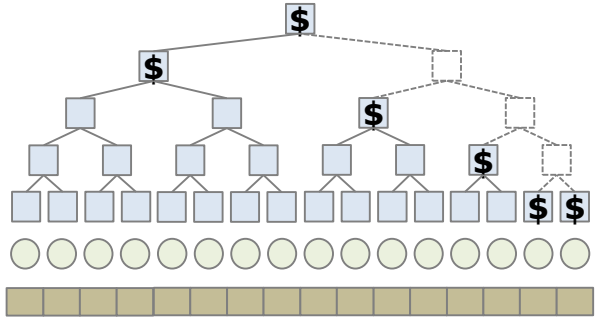



[1] Bonsai Merkle Forests: Efficiently Achieving Crash Consistency in Secure Persistent Memory (MICRO 2021)
[2] Scalable Memory Protection in the PENGLAI Enclave (OSDI 2021)
[3] Efficient Distributed Secure Memory with Migratable Merkle Tree (HPCA 2023)
[4] Data Enclave: A Data-Centric Trusted Execution Environment (HPCA 2024)

# Combined with Prior CTR Tree Optimization

- Prior hotness-based integrity tree optimization scheme (Subtree optimization)[1-4]
  - Caching highly used roots of subtrees

Prior Subtree Optimization

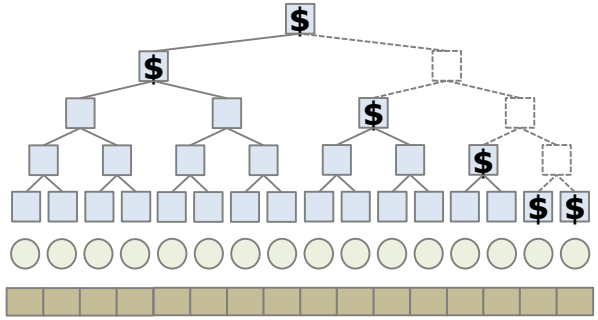



[1] Bonsai Merkle Forests: Efficiently Achieving Crash Consistency in Secure Persistent Memory (MICRO 2021)
[2] Scalable Memory Protection in the PENGLAI Enclave (OSDI 2021)
[3] Efficient Distributed Secure Memory with Migratable Merkle Tree (HPCA 2023)
[4] Data Enclave: A Data-Centric Trusted Execution Environment (HPCA 2024)

# Combined with Prior CTR Tree Optimization

- Prior hotness-based integrity tree optimization scheme (Subtree optimization)[1-4]
  - Caching highly used roots of subtrees
  - Pruned unused nodes

Prior Subtree Optimization

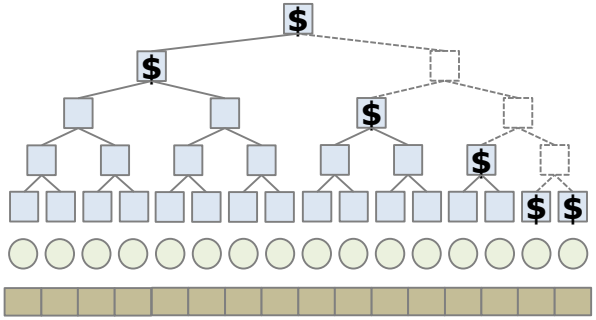



[1] Bonsai Merkle Forests: Efficiently Achieving Crash Consistency in Secure Persistent Memory (MICRO 2021)
[2] Scalable Memory Protection in the PENGLAI Enclave (OSDI 2021)
[3] Efficient Distributed Secure Memory with Migratable Merkle Tree (HPCA 2023)
[4] Data Enclave: A Data-Centric Trusted Execution Environment (HPCA 2024)

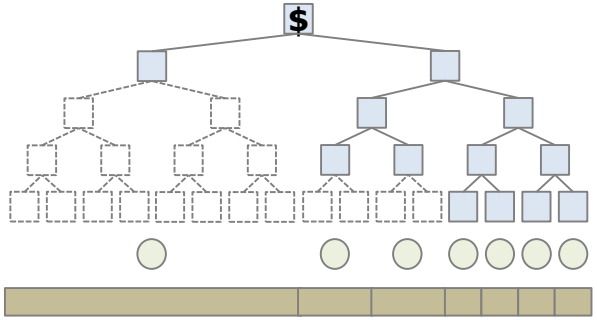
# Combined with Prior CTR Tree Optimization

- Prior hotness-based integrity tree optimization scheme (Subtree optimization)[1-4]
  - Caching highly used roots of subtrees
  - Pruned unused nodes

Prior Subtree Optimization




Ours

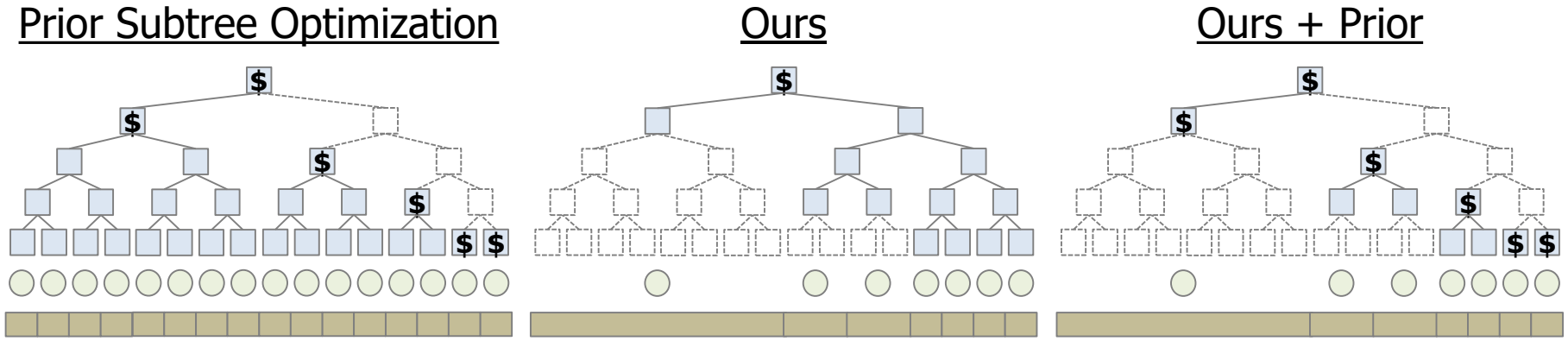




[1] Bonsai Merkle Forests: Efficiently Achieving Crash Consistency in Secure Persistent Memory (MICRO 2021)
[2] Scalable Memory Protection in the PENGLAI Enclave (OSDI 2021)
[3] Efficient Distributed Secure Memory with Migratable Merkle Tree (HPCA 2023)
[4] Data Enclave: A Data-Centric Trusted Execution Environment (HPCA 2024)

# Combined with Prior CTR Tree Optimization

- Prior hotness-based integrity tree optimization scheme (Subtree optimization)[1-4]
  - Caching highly used roots of subtrees
  - Pruned unused nodes

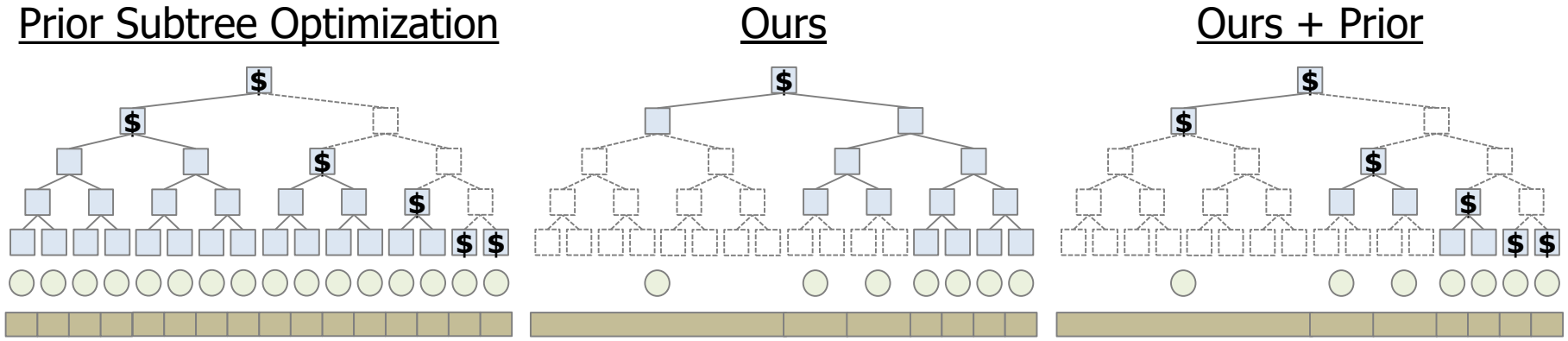




[1] Bonsai Merkle Forests: Efficiently Achieving Crash Consistency in Secure Persistent Memory (MICRO 2021)
[2] Scalable Memory Protection in the PENGLAI Enclave (OSDI 2021)
[3] Efficient Distributed Secure Memory with Migratable Merkle Tree (HPCA 2023)
[4] Data Enclave: A Data-Centric Trusted Execution Environment (HPCA 2024)

# Combined with Prior CTR Tree Optimization

- Prior hotness-based integrity tree optimization scheme (Subtree optimization)[1-4]
  - Caching highly used roots of subtrees
  - Pruned unused nodes




**Multi-granular MAC&Tree further improves prior solutions!!**

[4] Data Enclave: A Data-Centric Trusted Execution Environment (HPCA 2024)

# Design of Multi-granular MAC&Tree

# Design of Multi-granular MAC&Tree

1. Granularity Switching

# Design of Multi-granular MAC&Tree

1. Granularity Switching
2. Granularity Detection

# Design of Multi-granular MAC&Tree

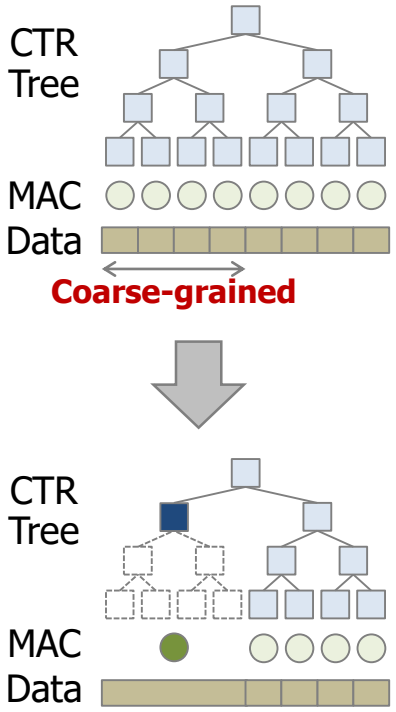
1. Granularity Switching

2. Granularity Detection

3. Multi-granularity Memory Protection

# Design of Multi-granular MAC&Tree

1. Granularity Switching



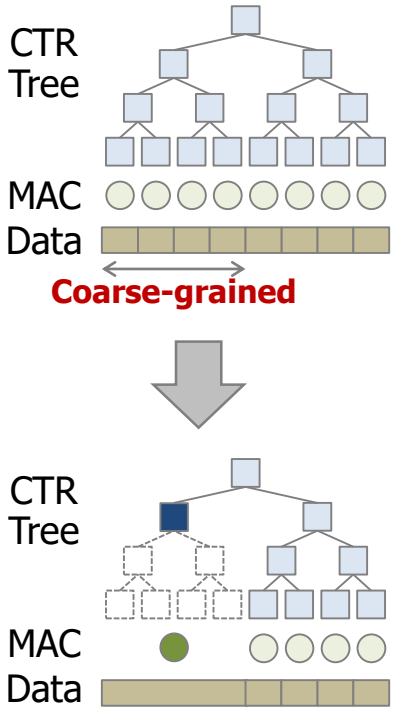

MAC/CTR Merging
& CTR Tree Pruning

2. Granularity Detection

3. Multi-granularity
Memory Protection

# Design of Multi-granular MAC&Tree

1. Granularity Switching




MAC/CTR Merging
& CTR Tree Pruning

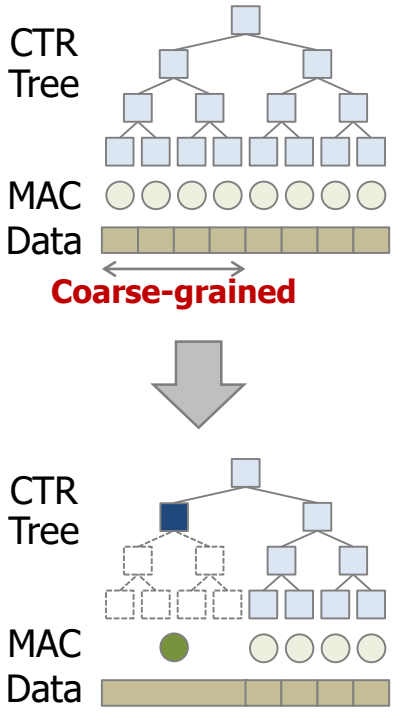
2. Granularity Detection

Dynamic Access Tracking

3. Multi-granularity
Memory Protection

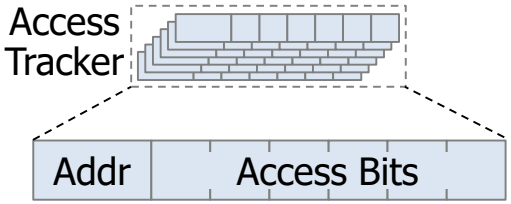
# Design of Multi-granular MAC&Tree

1. Granularity Switching




MAC/CTR Merging
& CTR Tree Pruning

2. Granularity Detection

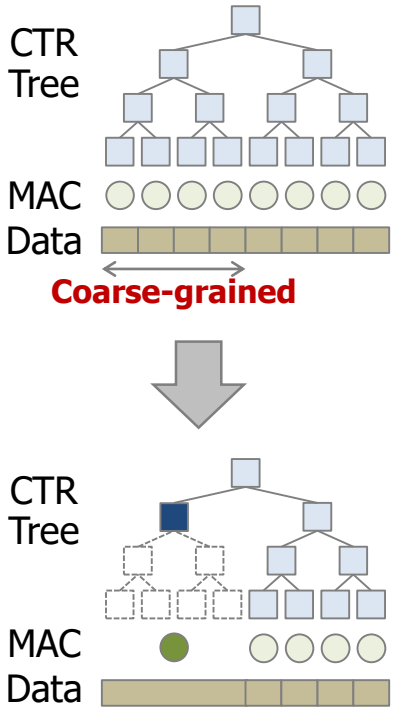



Dynamic Access Tracking

3. Multi-granularity
Memory Protection

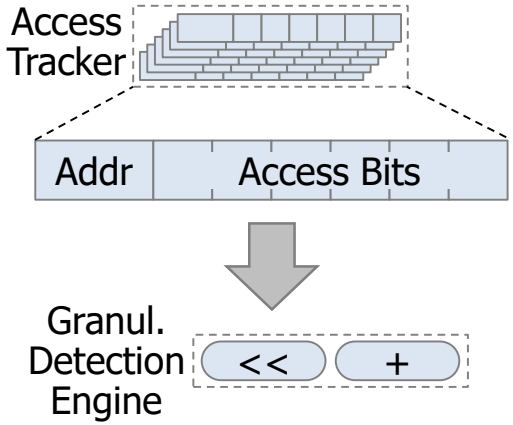
# Design of Multi-granular MAC&Tree

1. Granularity Switching




MAC/CTR Merging
& CTR Tree Pruning

2. Granularity Detection

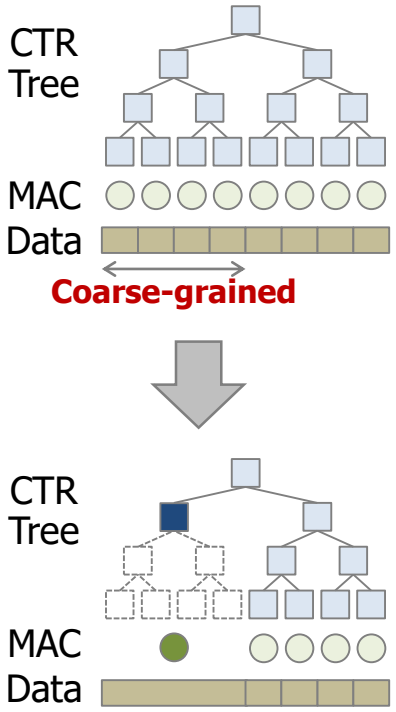



Dynamic Access Tracking

3. Multi-granularity
Memory Protection

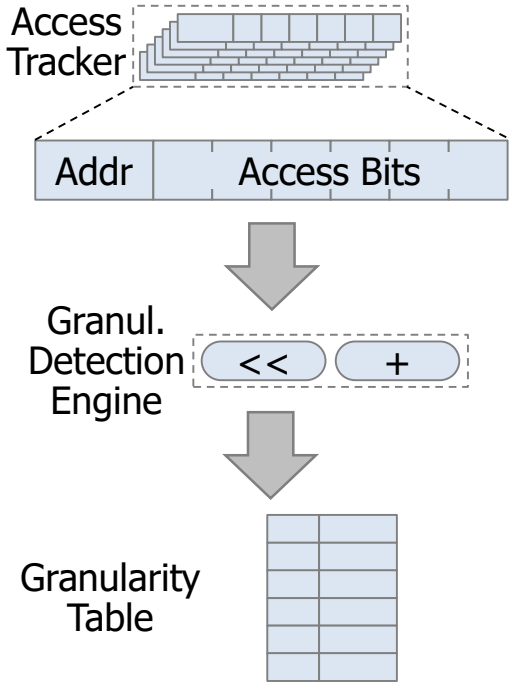
# Design of Multi-granular MAC&Tree

1. Granularity Switching




MAC/CTR Merging
& CTR Tree Pruning

2. Granularity Detection

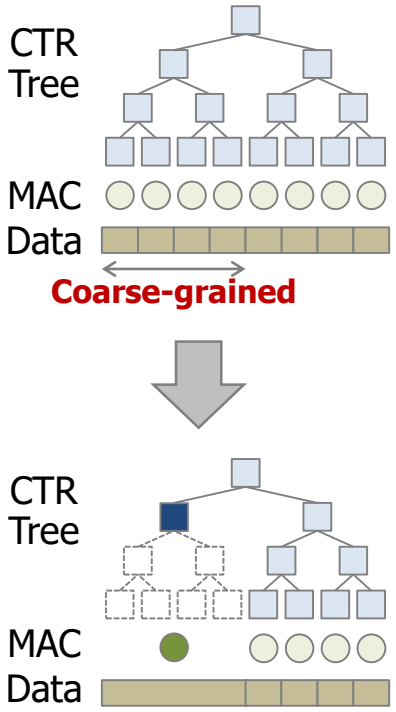



Dynamic Access Tracking

3. Multi-granularity
Memory Protection

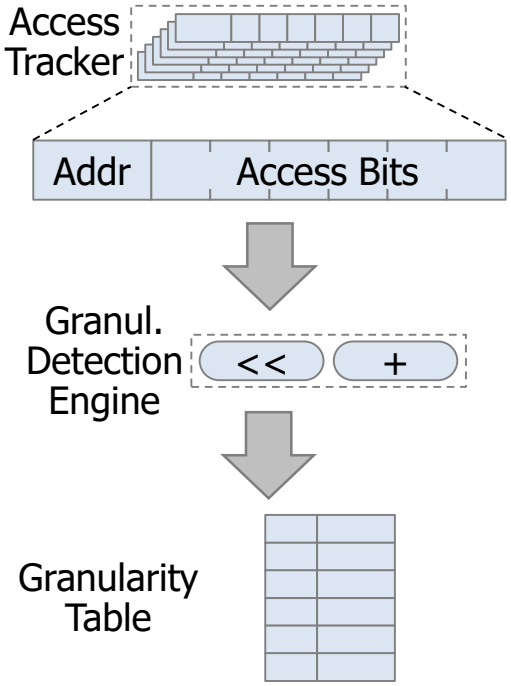
# Design of Multi-granular MAC&Tree

1. Granularity Switching




MAC/CTR Merging
& CTR Tree Pruning

2. Granularity Detection



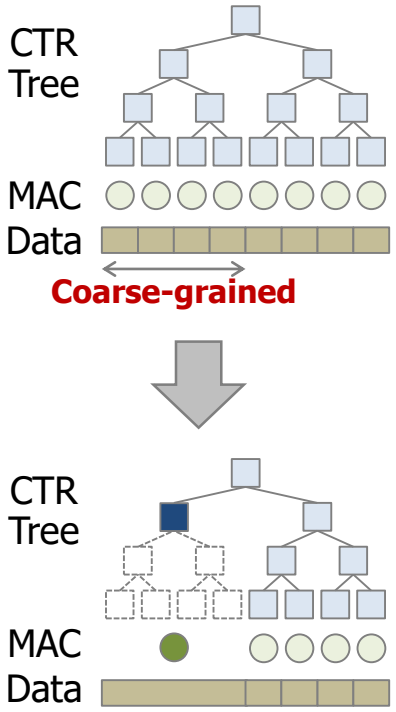

Dynamic Access Tracking

3. Multi-granularity
Memory Protection

Granularity-aware Protection

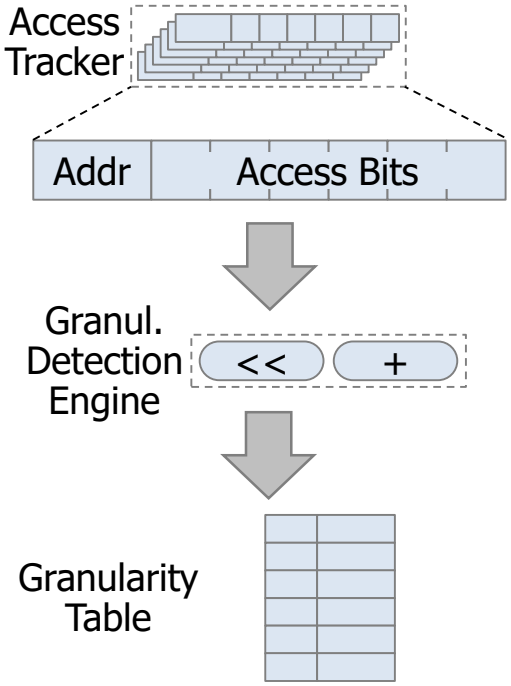
# Design of Multi-granular MAC&Tree

1. Granularity Switching



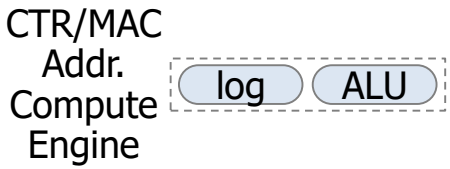

MAC/CTR Merging
& CTR Tree Pruning

2. Granularity Detection




Dynamic Access Tracking

3. Multi-granularity Memory Protection

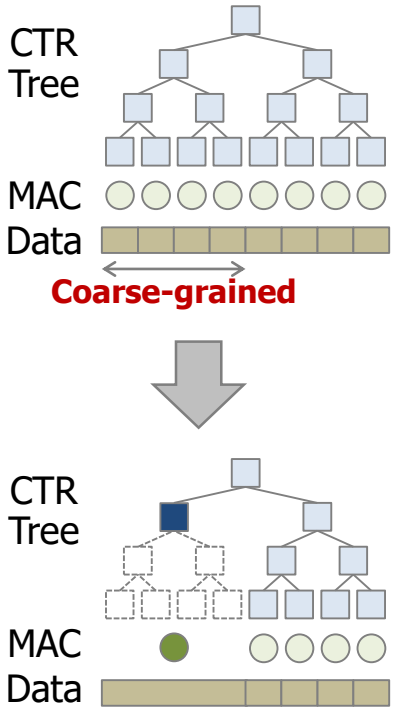



Granularity-aware Protection

Backup

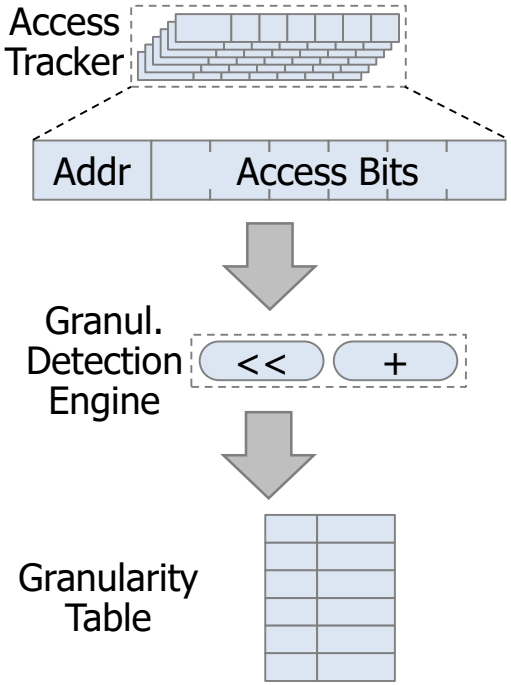
# Design of Multi-granular MAC&Tree

## 1. Granularity Switching



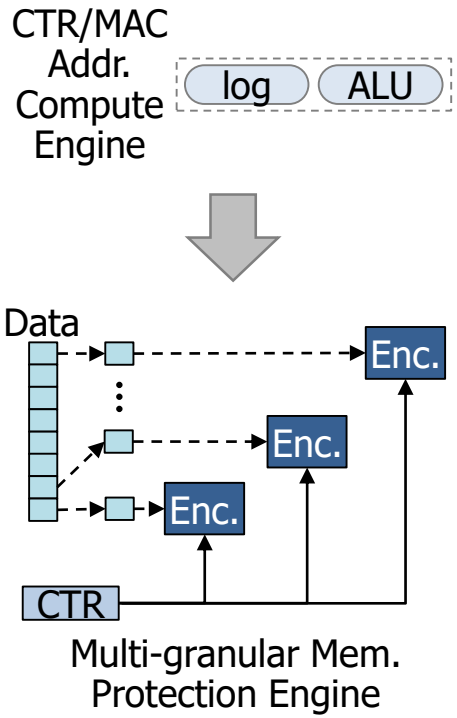

MAC/CTR Merging
& CTR Tree Pruning

## 2. Granularity Detection




Dynamic Access Tracking

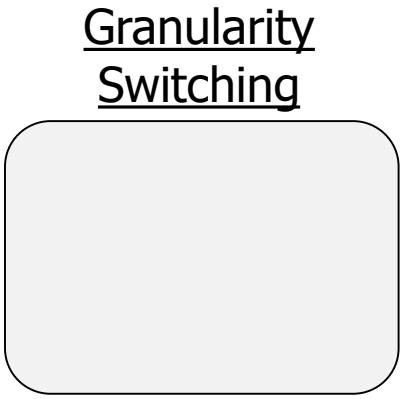
## 3. Multi-granularity Memory Protection




Granularity-aware Protection

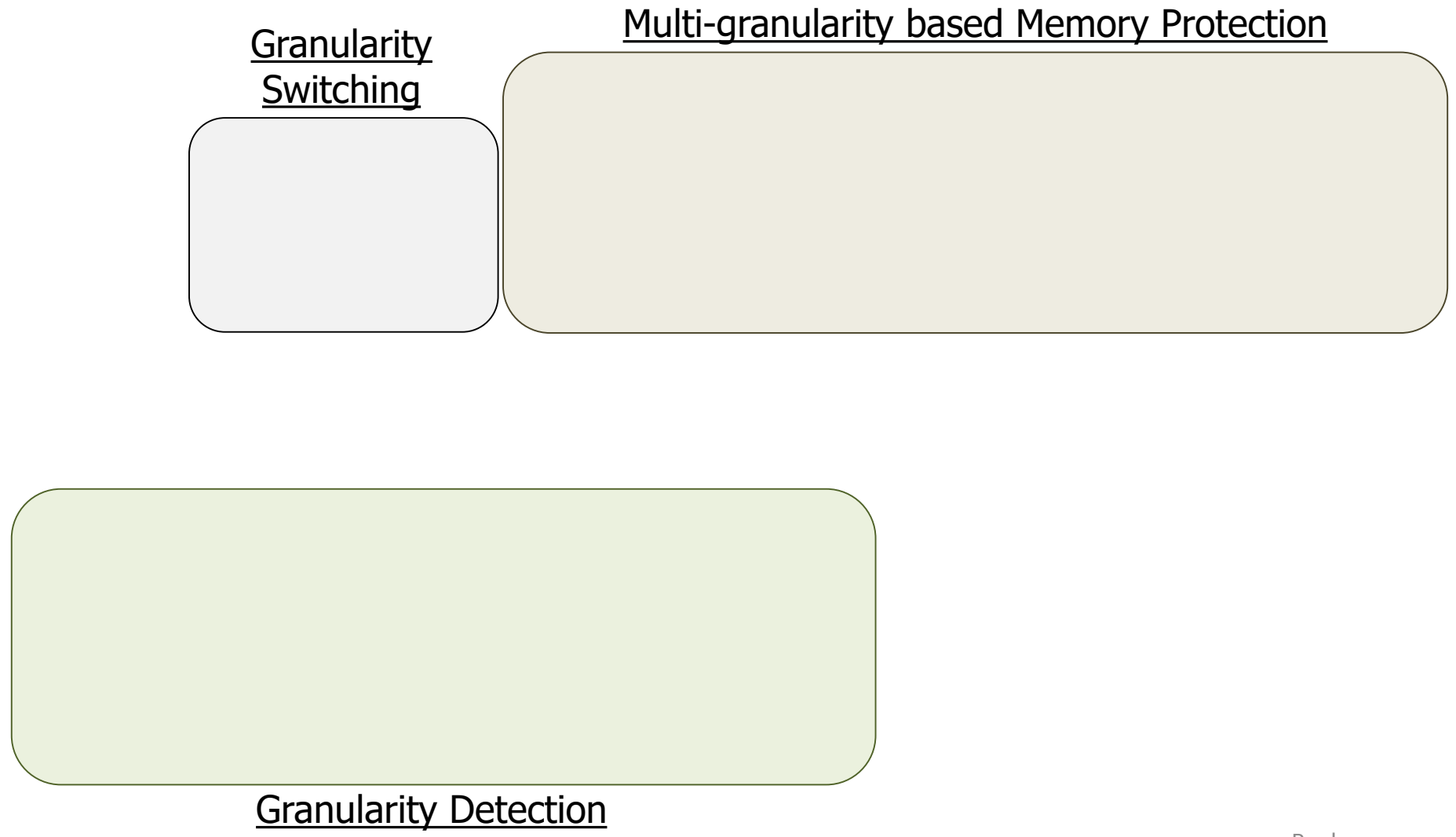
# Design of Multi-granular MAC&Tree

# Design of Multi-granular MAC&Tree

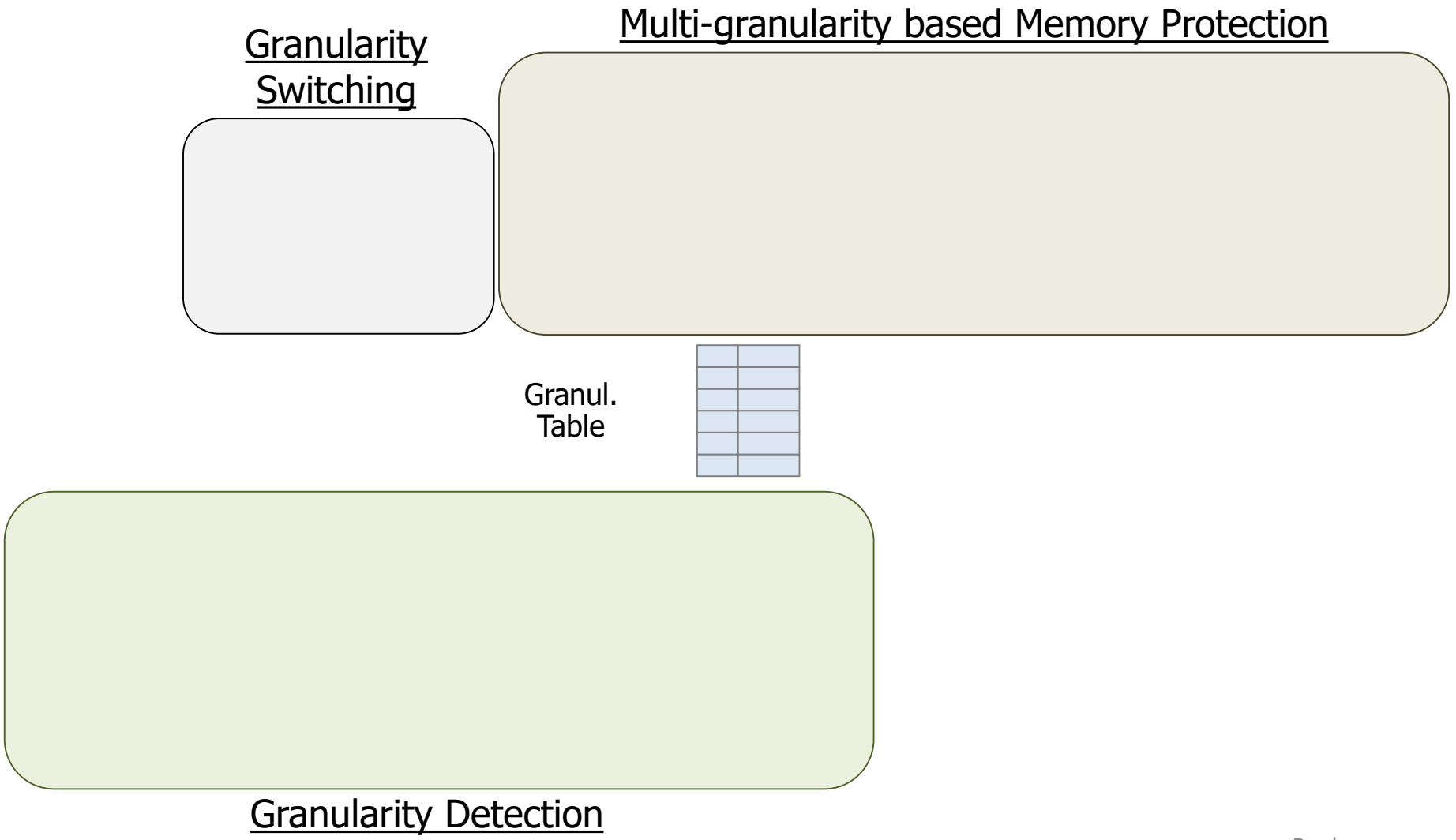


# Design of Multi-granular MAC&Tree

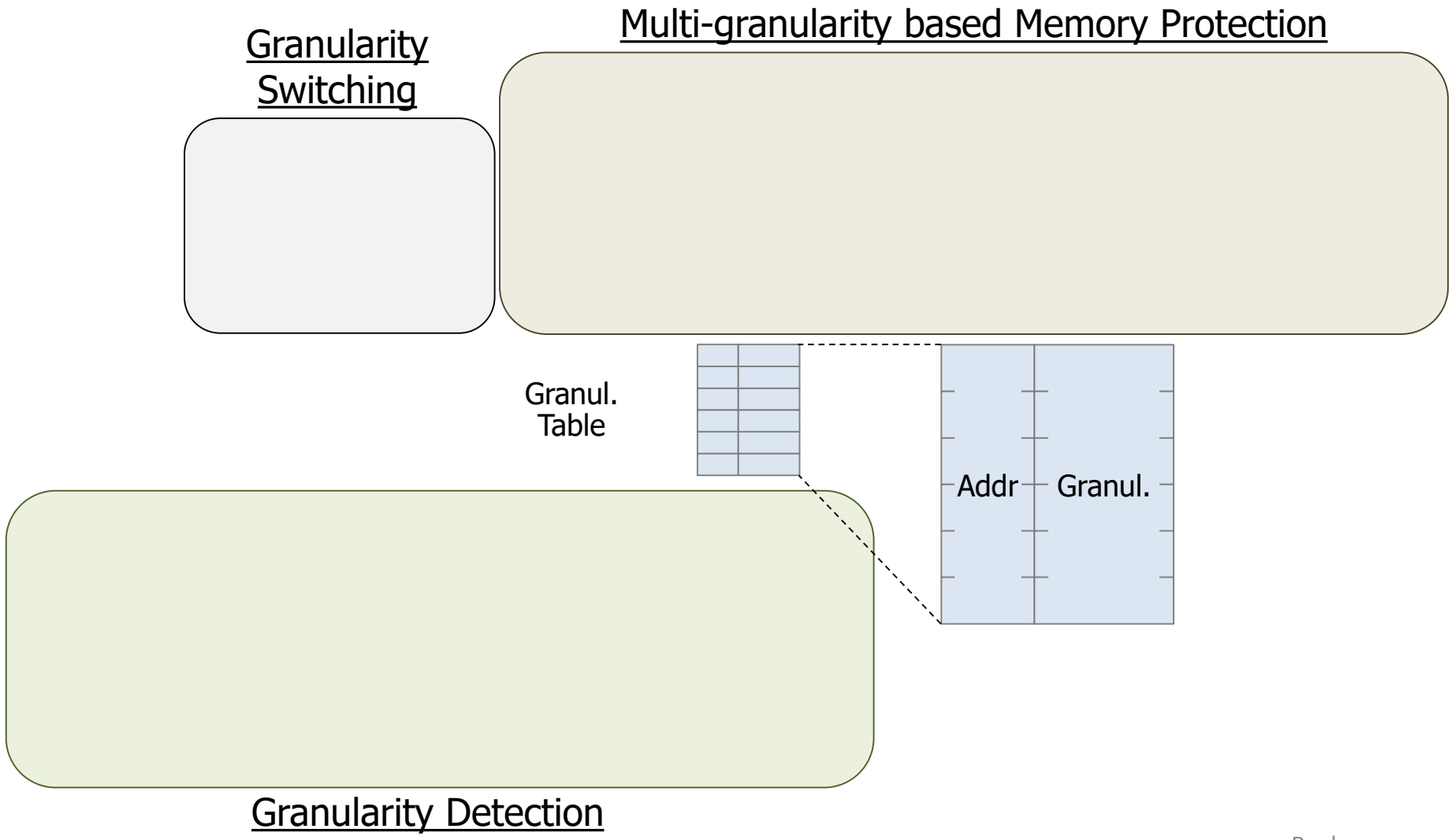
# Design of Multi-granular MAC&Tree



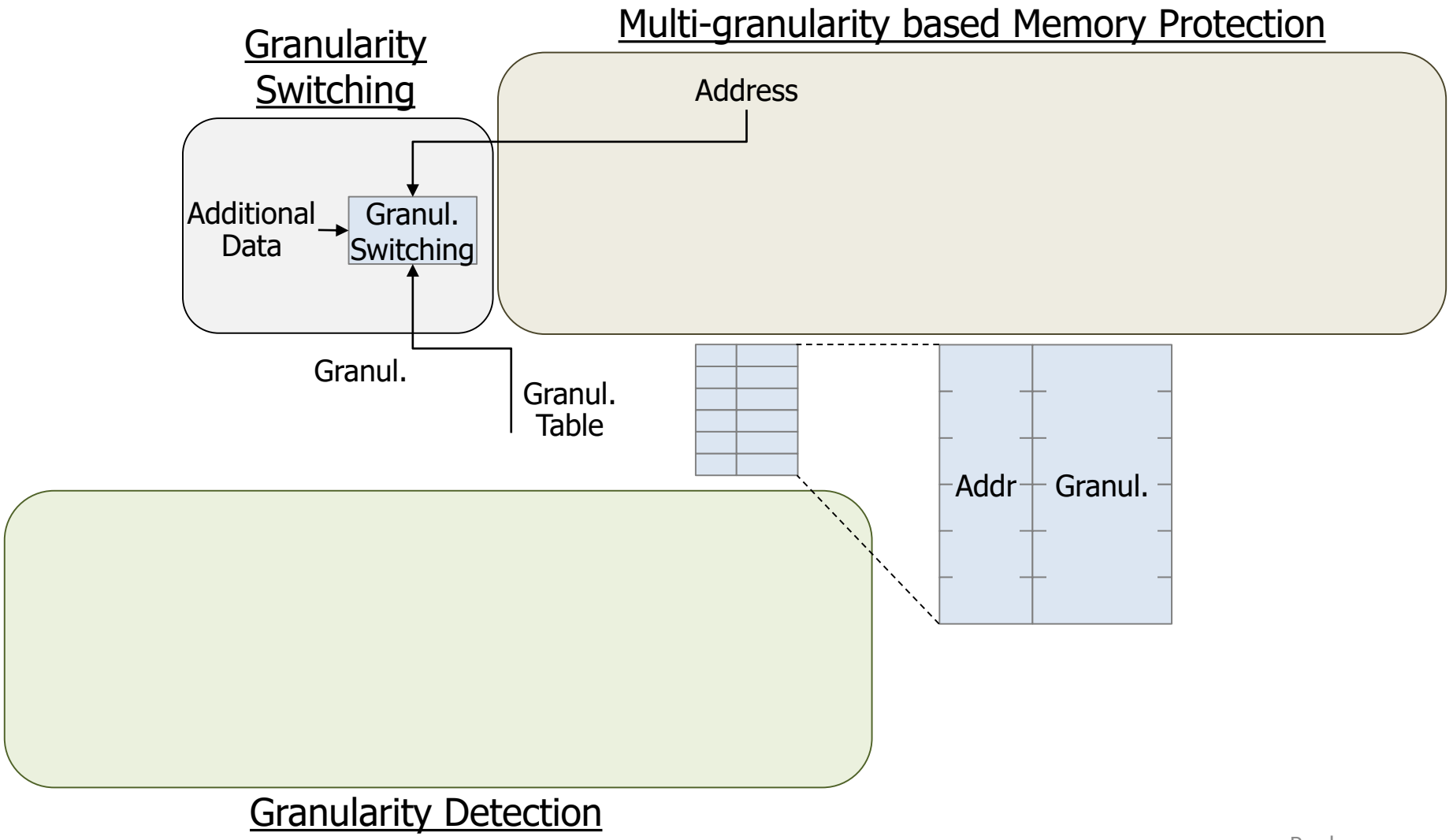
# Design of Multi-granular MAC&Tree



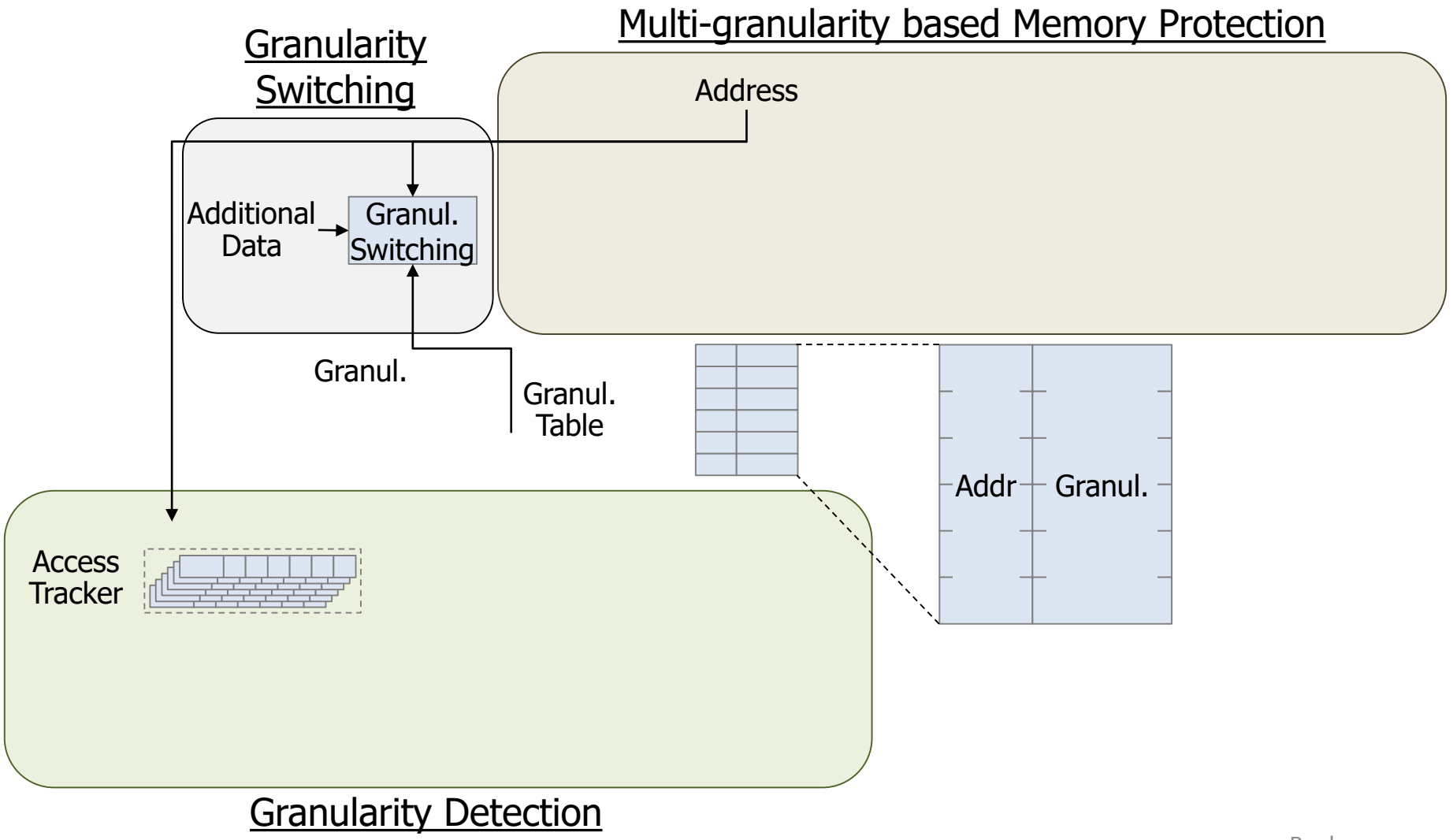
# Design of Multi-granular MAC&Tree



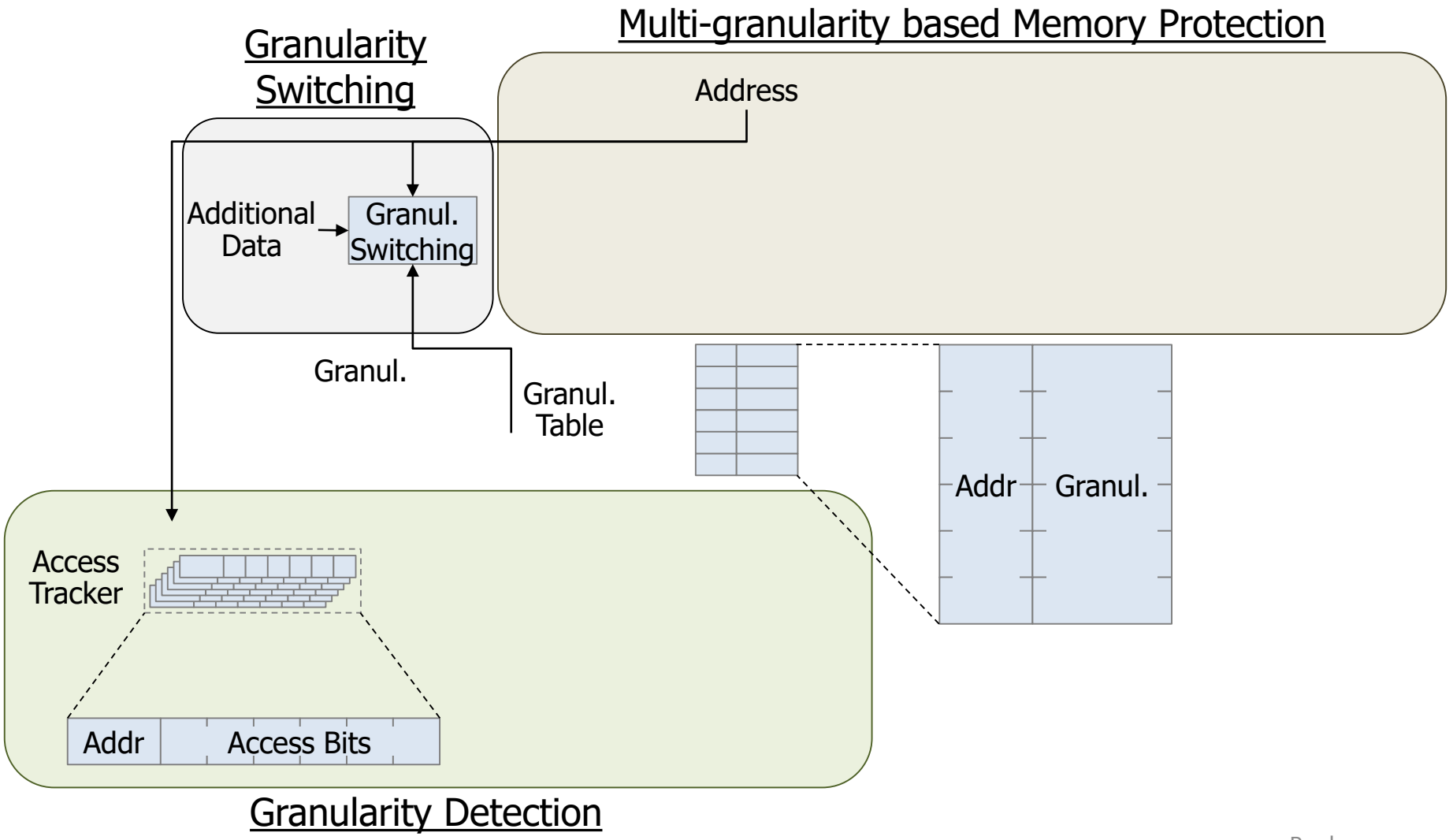
# Design of Multi-granular MAC&Tree



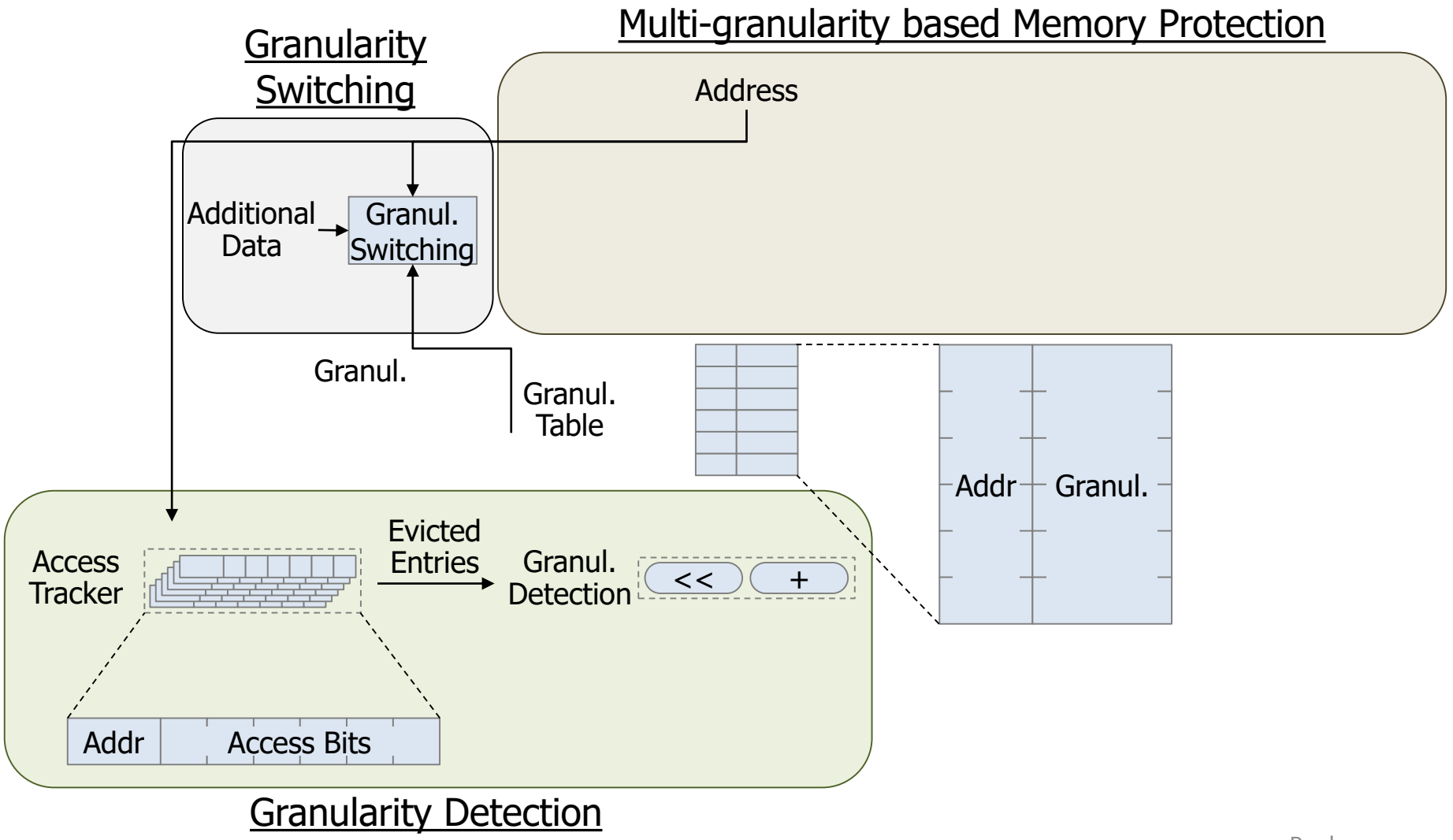
# Design of Multi-granular MAC&Tree



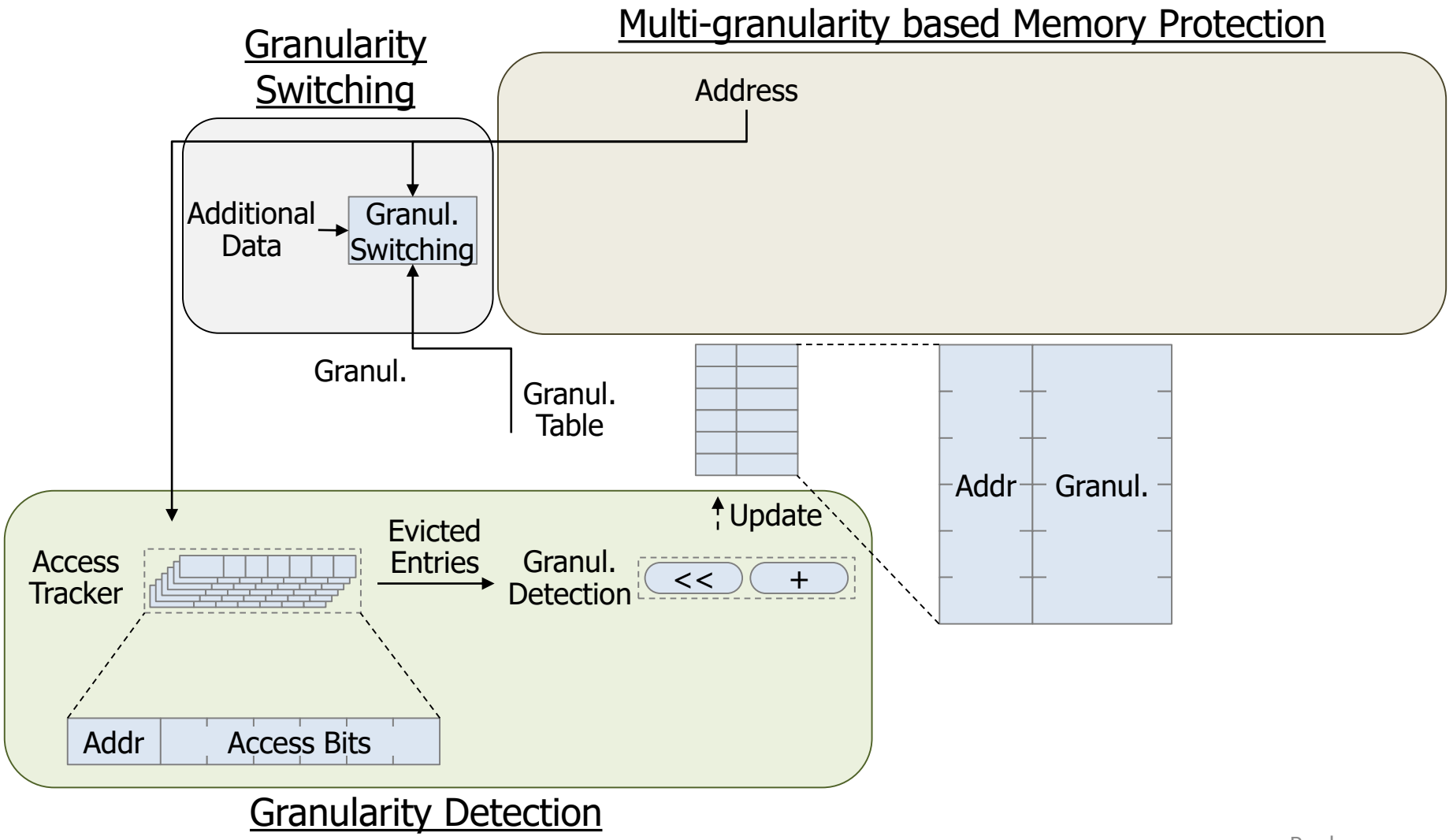
# Design of Multi-granular MAC&Tree



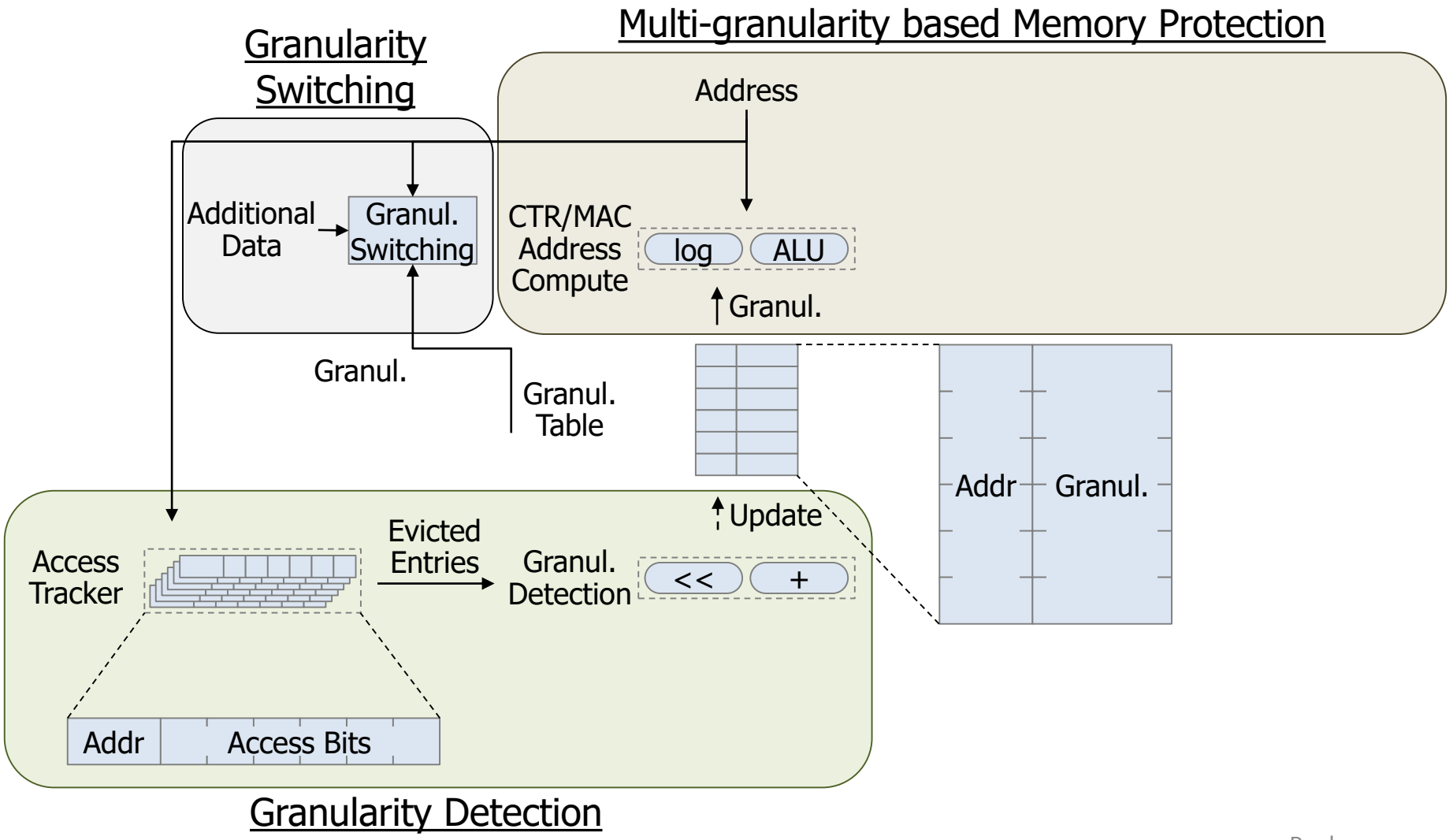
# Design of Multi-granular MAC&Tree



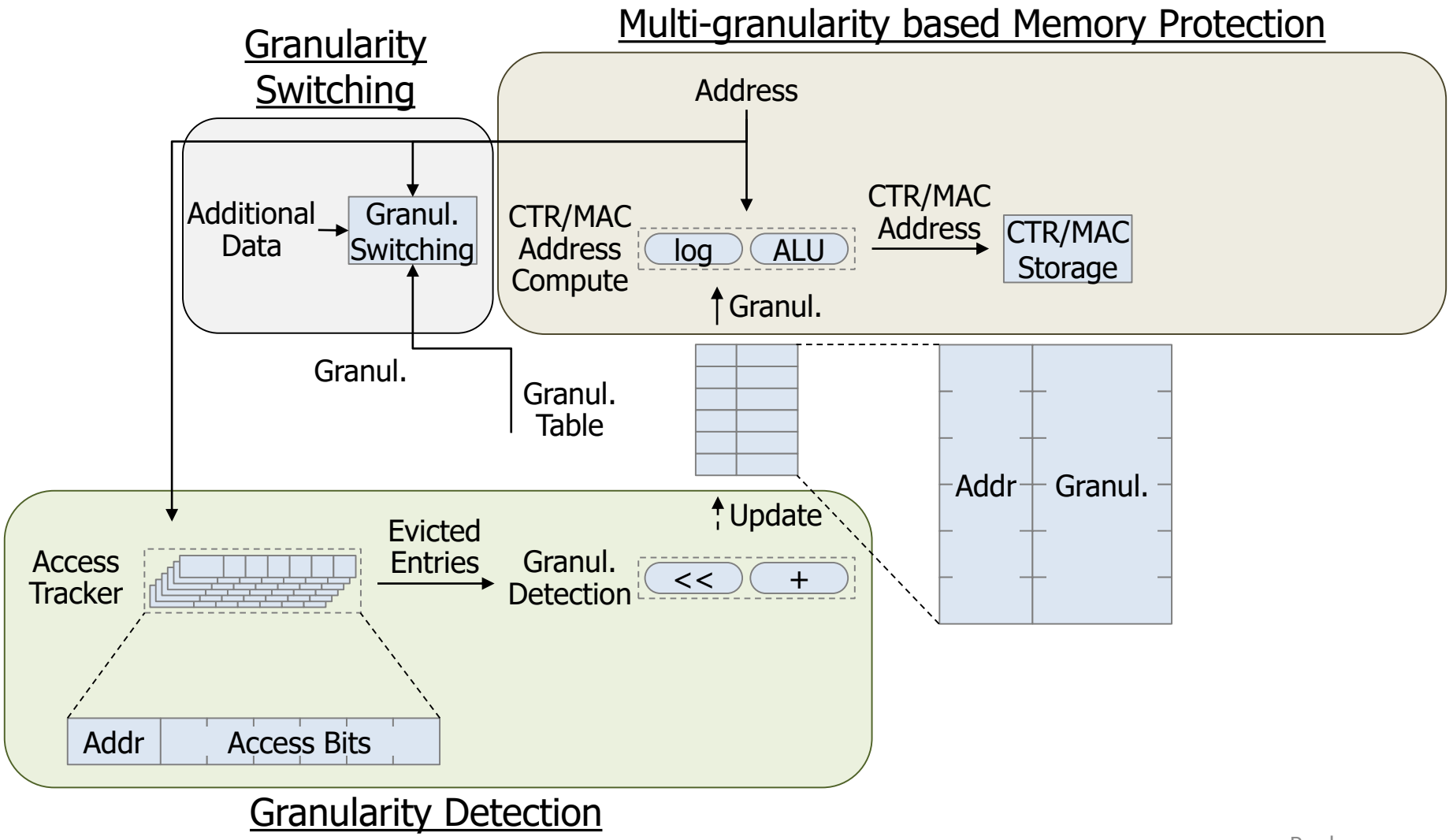
# Design of Multi-granular MAC&Tree



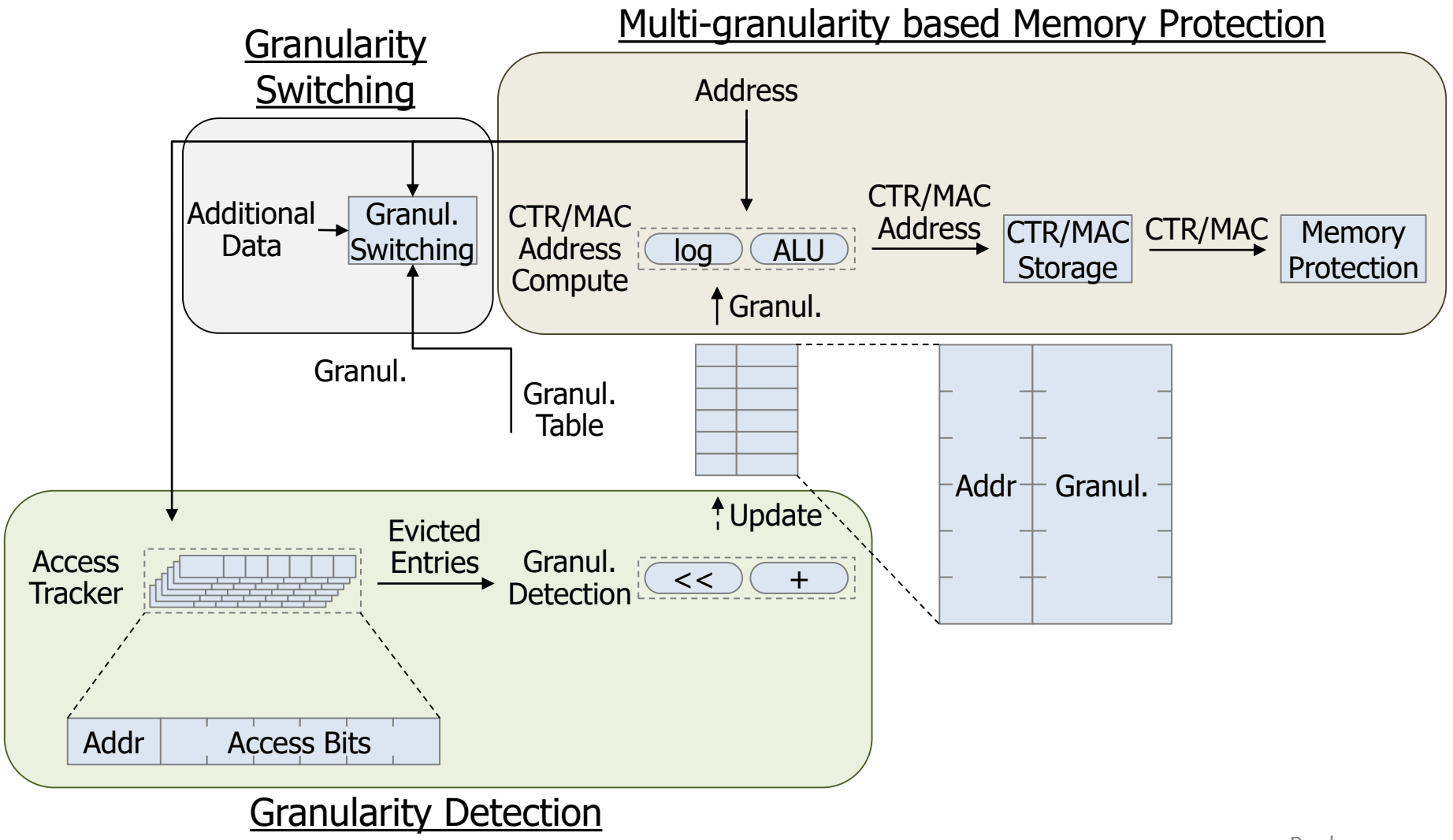
# Design of Multi-granular MAC&Tree



# Design of Multi-granular MAC&Tree



# Design of Multi-granular MAC&Tree



# Recent Memory Protection Studies

# Recent Memory Protection Studies

| Study | Target | Multi CTR | Int. Tree Opt. | Multi MAC | Dynamic Update | Target App. |
|---|---|---|---|---|---|---|
| *SoftVN* | CPU | O | X | X | X | ML-specific |
| *Common Counters* | GPU | Dual | X | X | X | General |
| *Adaptive* | GPU | X | X | Dual | O | General |
| *TNPU* | NPU | O | X | X | X | ML-specific |
| *Tunable Tree* | NPU | O | Sub Optimal | X | X | General |
| *MGX* | NPU | O | X | O | X | ML-specific |
| *GuardNN* | NPU | O | X | X | X | ML-specific |
| *TensorTEE* | CPU+NPU | O | X | O | O | ML-specific |
| **Ours** | **CPU+GPU +NPU** | **O** | **Optimal** | **O** | **O** | **General** |

# Prior Integrity Tree Optimization

# Prior Integrity Tree Optimization

| Study | Target | Multi CTR | Int. Tree Opt. | Multi MAC | Dynamic Update | Target App. |
|---|---|---|---|---|---|---|
| *Bonsai Merkle Forests* | CPU | X | Sub Optimal | X | X | General |
| *PENGLAI* | GPU | X | Sub Optimal | X | X | General |
| *Migratable Merkle Tree* | GPU | X | Sub Optimal | X | X | General |
| *Data Enclave* | NPU | X | Sub Optimal | X | X | General |
| **Ours** | CPU+GPU +NPU | O | Optimal | O | O | General |

# Encryption & Integrity Validation using Coarse-grained MAC & Counter

# Encryption & Integrity Validation using Coarse-grained MAC & Counter

- Coarse-MAC & counter encryption, integrity validation

# Encryption & Integrity Validation using Coarse-grained MAC & Counter

- Coarse-MAC & counter encryption, integrity validation
  - $CTR_{course(1 \cdots k)} = \mathrm{MAX}(CTR_{fine\,1}, CTR_{fine\,2}, \cdots, CTR_{fine\,k})$

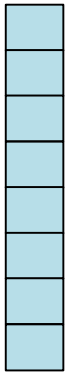
# Encryption & Integrity Validation using Coarse-grained MAC & Counter

- Coarse-MAC & counter encryption, integrity validation
  - $CTR_{course(1 \cdots k)} = \mathrm{MAX}(CTR_{fine\,1}, CTR_{fine\,2}, \cdots, CTR_{fine\,k})$
  - $MAC_{course(1 \cdots k)} =$
    $\mathrm{HASH}(\cdots \mathrm{HASH}(\mathrm{HASH}(MAC_{fine\,1}), MAC_{fine\,2}), \cdots, MAC_{fine\,k})$

# Encryption & Integrity Validation using Coarse-grained MAC & Counter

- Coarse-MAC & counter encryption, integrity validation
  - $CTR_{course(1 \cdots k)} = \mathrm{MAX}(CTR_{fine\ 1}, CTR_{fine\ 2}, \cdots, CTR_{fine\ k})$
  - $MAC_{course(1 \cdots k)} =$
    $\mathrm{HASH}(\cdots \mathrm{HASH}(\mathrm{HASH}(MAC_{fine\ 1}), MAC_{fine\ 2}), \cdots, MAC_{fine\ k})$

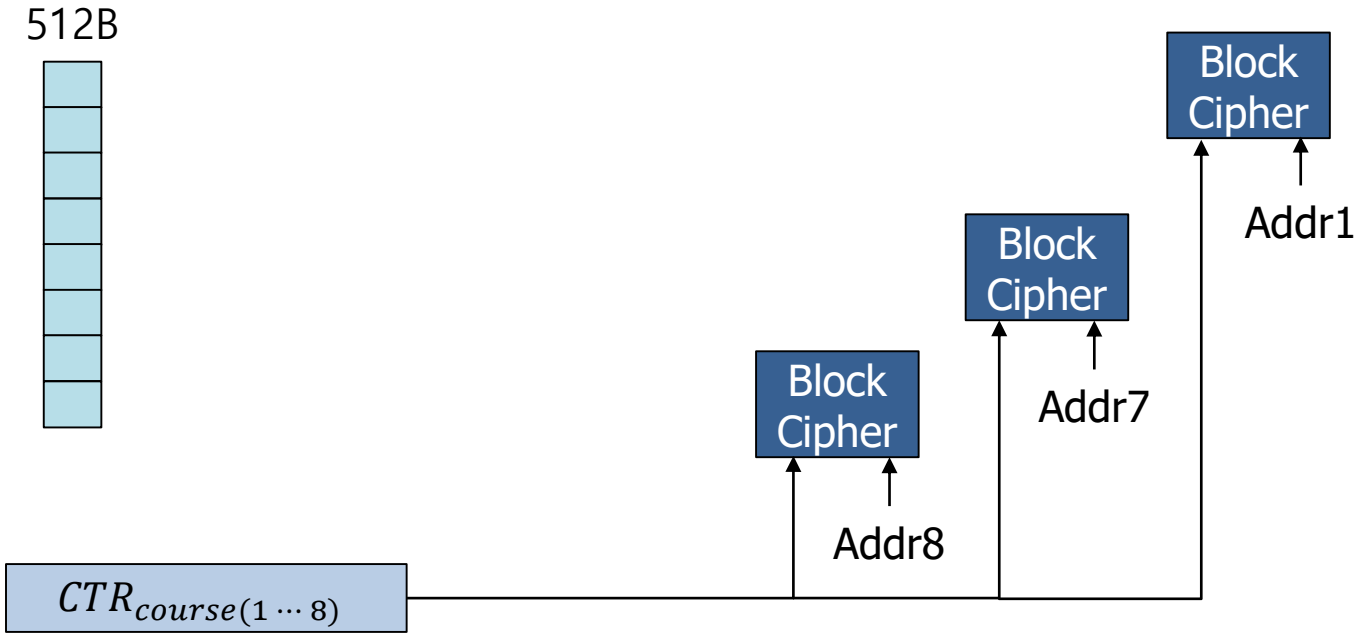
512B



$CTR_{course(1 \cdots 8)}$

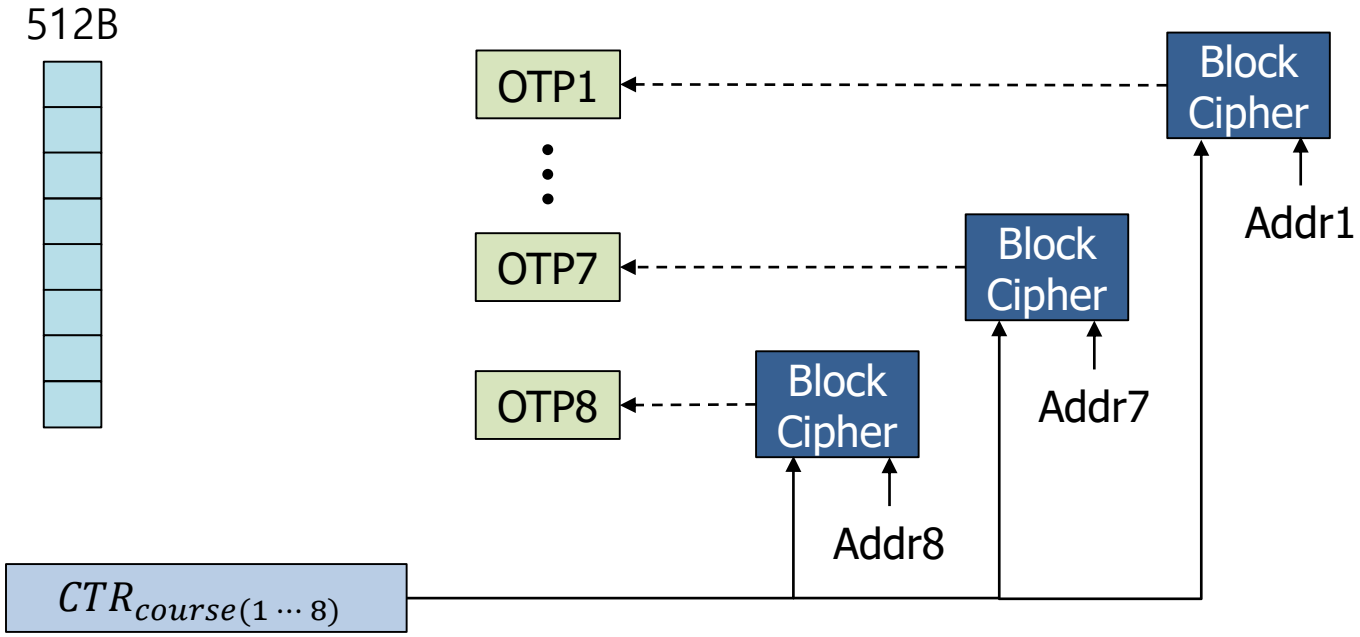
# Encryption & Integrity Validation using Coarse-grained MAC & Counter

- Coarse-MAC & counter encryption, integrity validation
  - $CTR_{course(1 \cdots k)} = \mathrm{MAX}(CTR_{fine\ 1}, CTR_{fine\ 2}, \cdots, CTR_{fine\ k})$
  - $MAC_{course(1 \cdots k)} =$ $\mathrm{HASH}(\cdots \mathrm{HASH}(\mathrm{HASH}(MAC_{fine\ 1}), MAC_{fine\ 2}), \cdots, MAC_{fine\ k})$



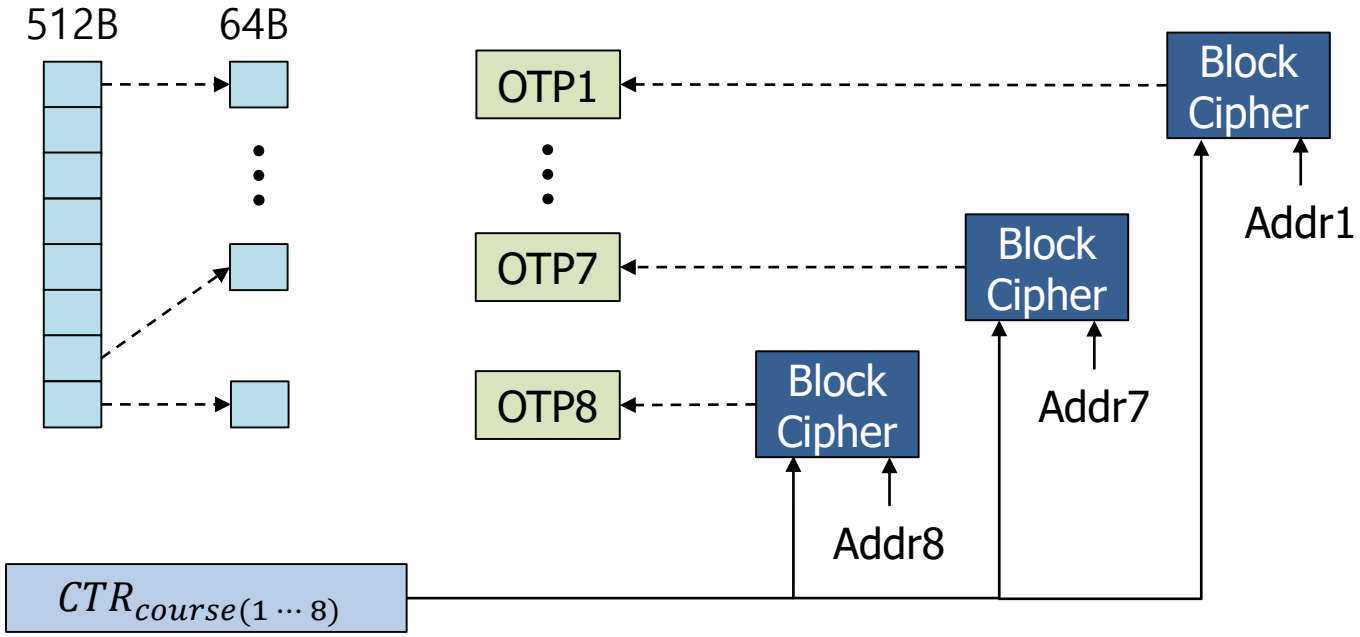
# Encryption & Integrity Validation using Coarse-grained MAC & Counter

- Coarse-MAC & counter encryption, integrity validation
  - $CTR_{course(1 \cdots k)} = \mathrm{MAX}(CTR_{fine\ 1}, CTR_{fine\ 2}, \cdots, CTR_{fine\ k})$
  - $MAC_{course(1 \cdots k)} =$ $\mathrm{HASH}(\cdots \mathrm{HASH}(\mathrm{HASH}(MAC_{fine\ 1}), MAC_{fine\ 2}), \cdots, MAC_{fine\ k})$



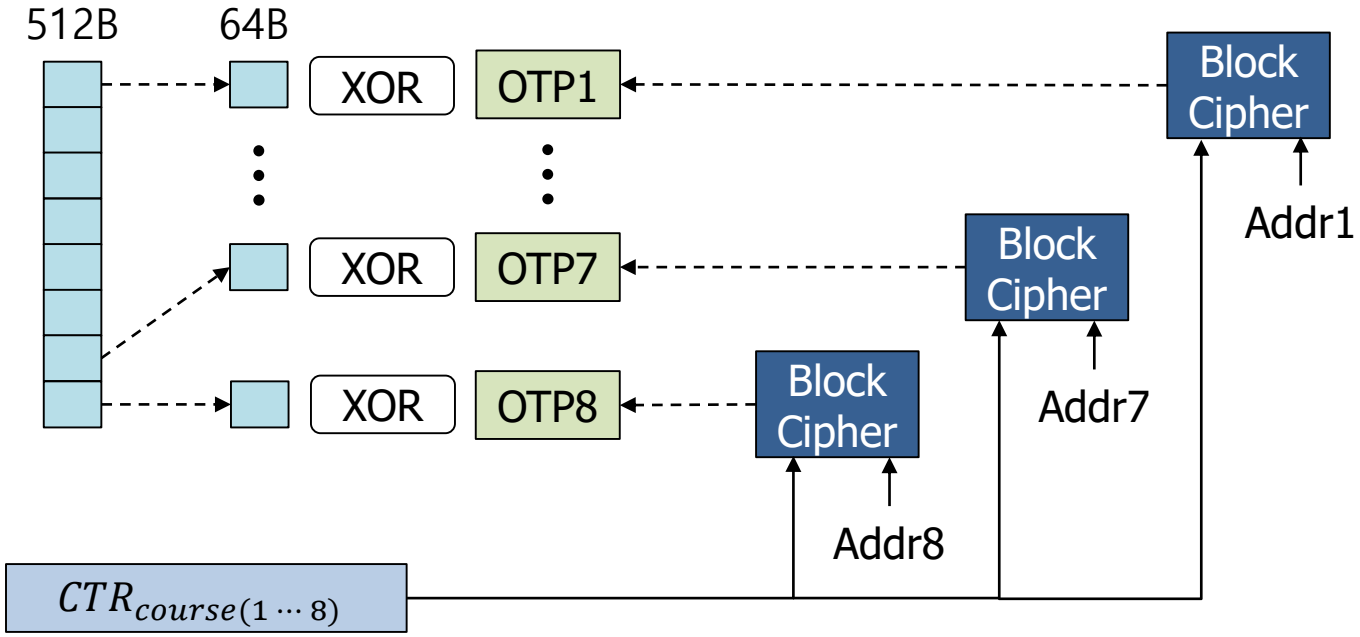
# Encryption & Integrity Validation using Coarse-grained MAC & Counter

- Coarse-MAC & counter encryption, integrity validation
  - $CTR_{course(1 \cdots k)} = \mathrm{MAX}(CTR_{fine\ 1}, CTR_{fine\ 2}, \cdots, CTR_{fine\ k})$
  - $MAC_{course(1 \cdots k)} = \mathrm{HASH}(\cdots \mathrm{HASH}(\mathrm{HASH}(MAC_{fine\ 1}), MAC_{fine\ 2}), \cdots, MAC_{fine\ k})$



# Encryption & Integrity Validation using Coarse-grained MAC & Counter

- Coarse-MAC & counter encryption, integrity validation
  - $CTR_{course(1 \cdots k)} = \mathrm{MAX}(CTR_{fine\ 1}, CTR_{fine\ 2}, \cdots, CTR_{fine\ k})$
  - $MAC_{course(1 \cdots k)} =$ $\mathrm{HASH}(\cdots \mathrm{HASH}(\mathrm{HASH}(MAC_{fine\ 1}), MAC_{fine\ 2}), \cdots, MAC_{fine\ k})$



# Address of MAC & Counter

# Address of MAC & Counter

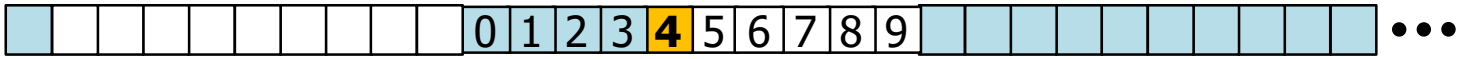
- Chunk-level index computation

# Address of MAC & Counter

- Chunk-level index computation
- Recursive parent call from leaf counters

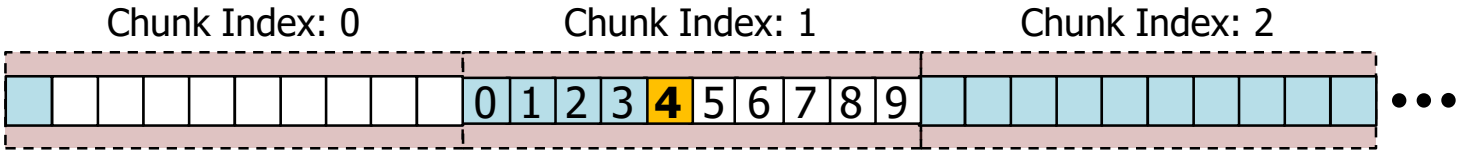
# Address of MAC & Counter

- Chunk-level index computation
- Recursive parent call from leaf counters



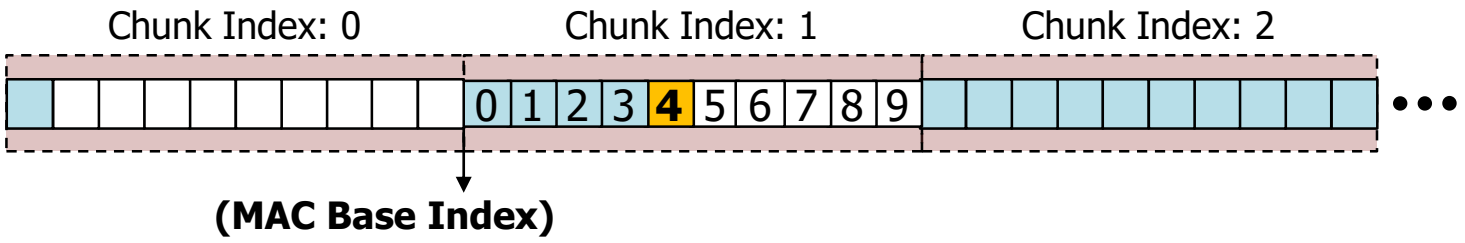
# Address of MAC & Counter

- Chunk-level index computation
- Recursive parent call from leaf counters



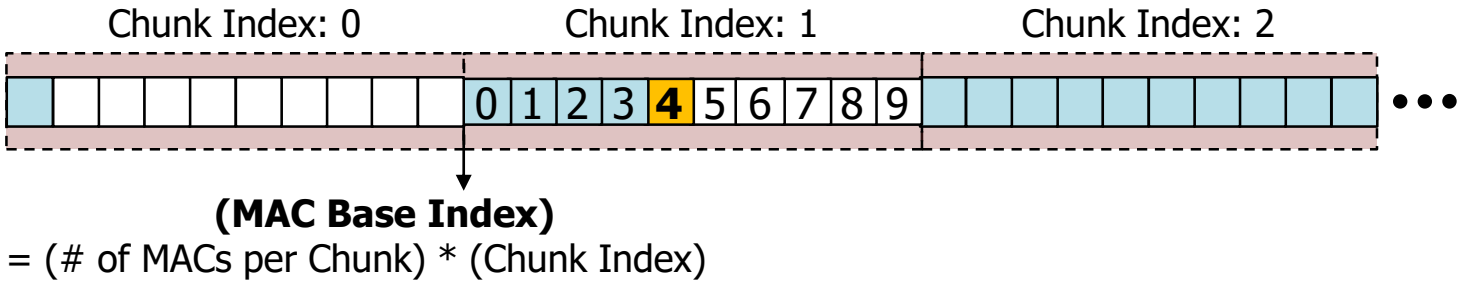
# Address of MAC & Counter

- Chunk-level index computation
- Recursive parent call from leaf counters



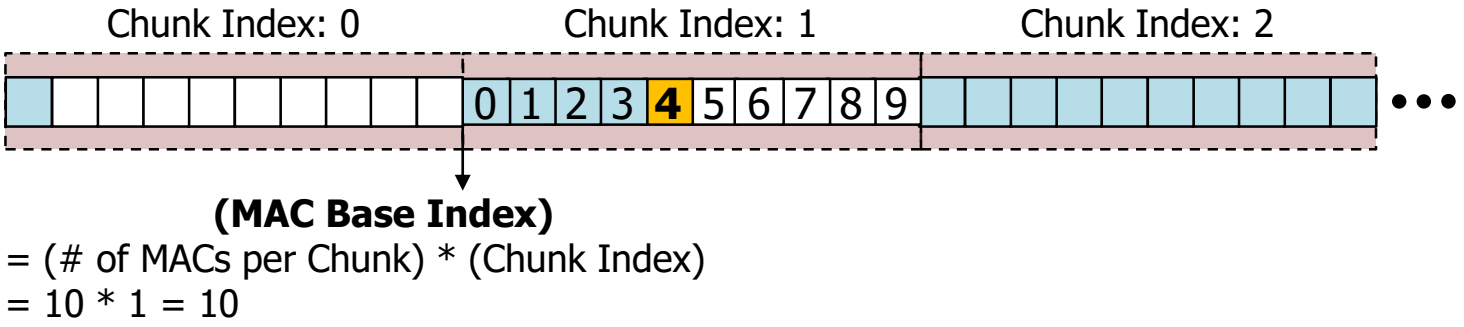
# Address of MAC & Counter

- Chunk-level index computation
- Recursive parent call from leaf counters



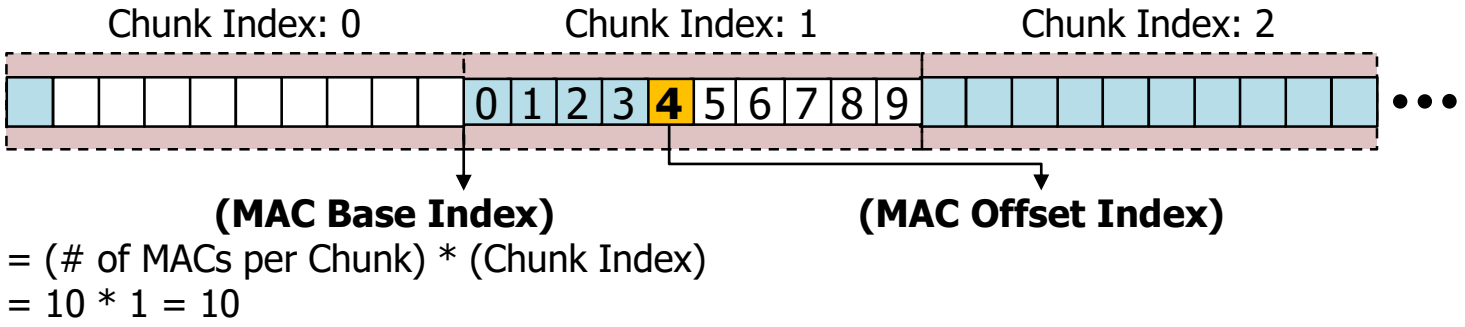
# Address of MAC & Counter

- Chunk-level index computation
- Recursive parent call from leaf counters



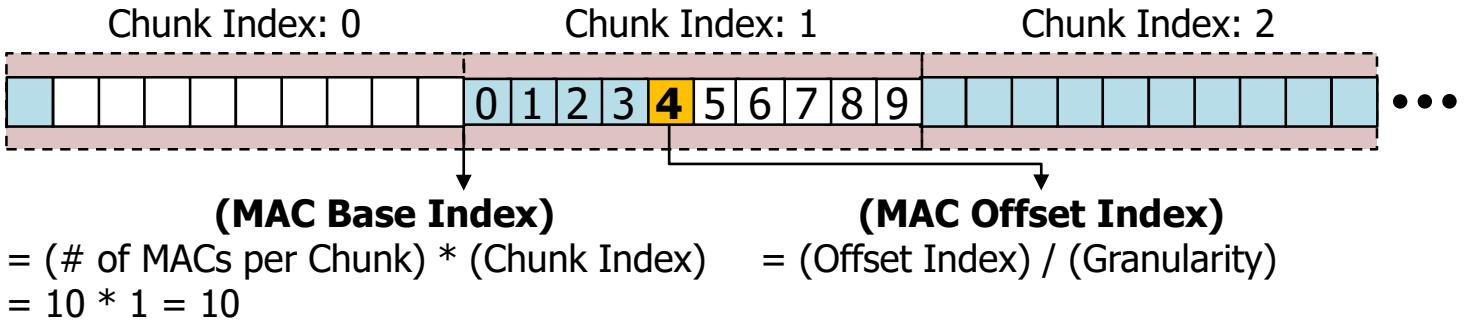
# Address of MAC & Counter

- Chunk-level index computation
- Recursive parent call from leaf counters



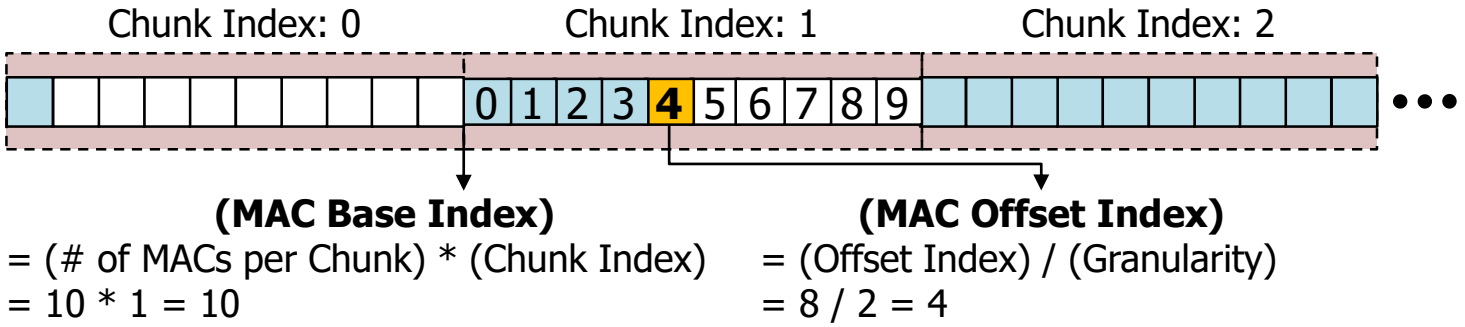
# Address of MAC & Counter

- Chunk-level index computation
- Recursive parent call from leaf counters



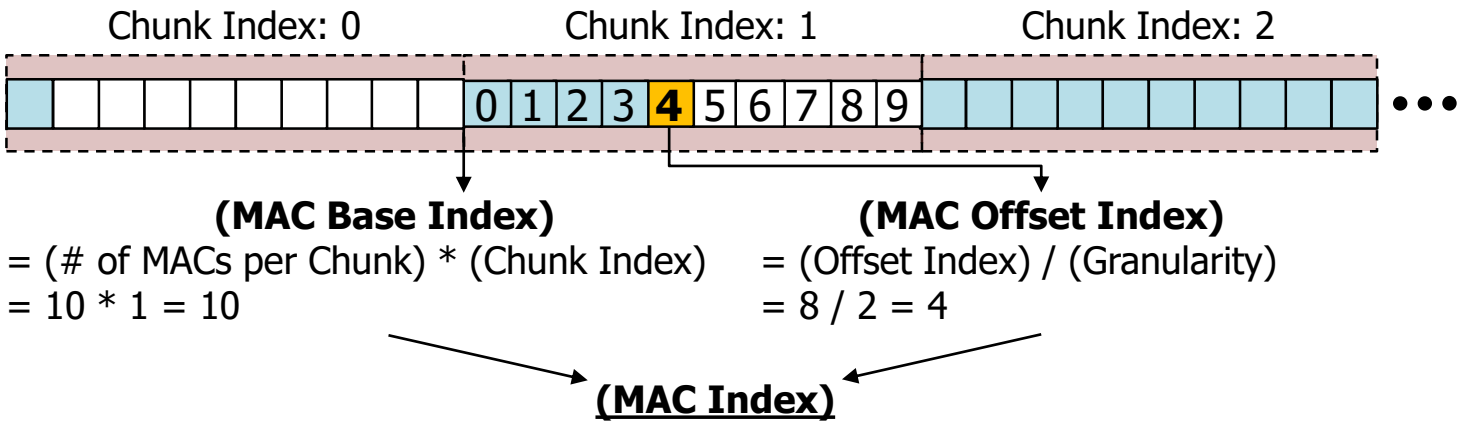
# Address of MAC & Counter

- Chunk-level index computation
- Recursive parent call from leaf counters



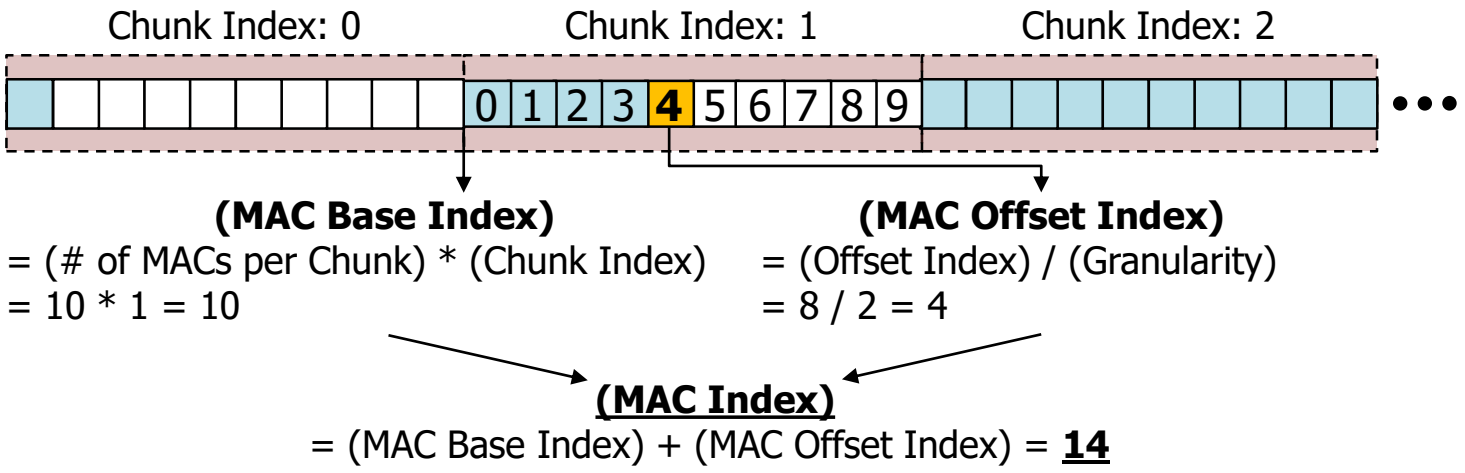
# Address of MAC & Counter

- Chunk-level index computation
- Recursive parent call from leaf counters



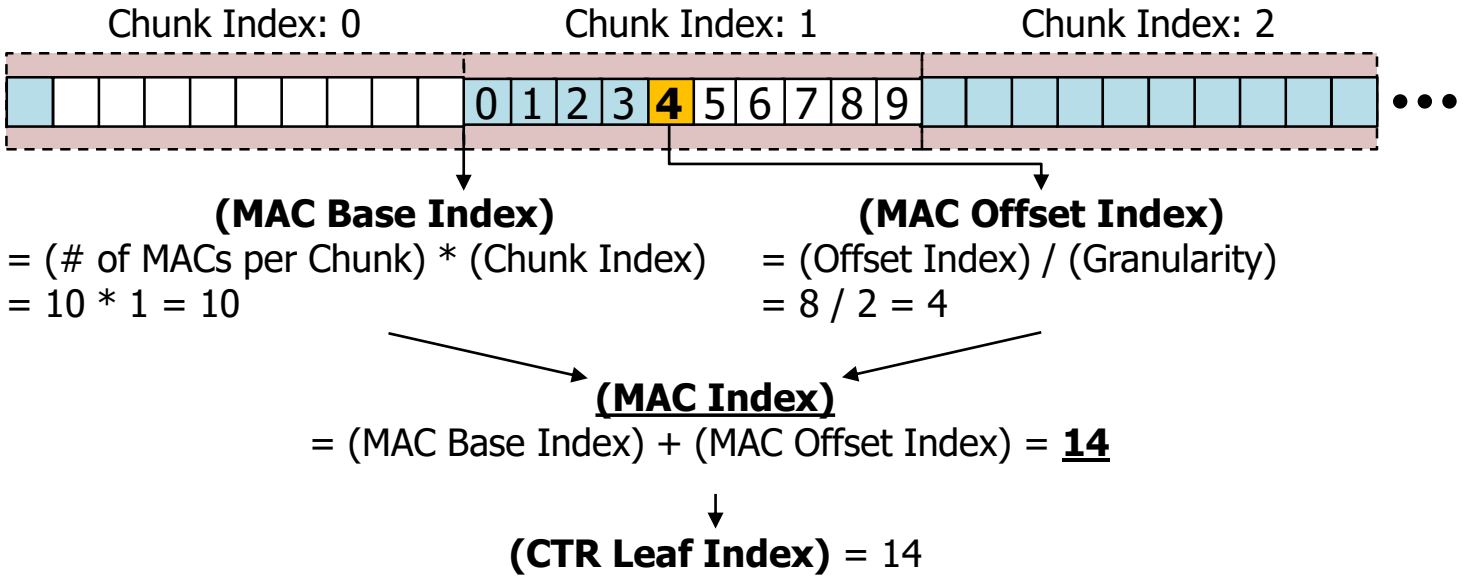
# Address of MAC & Counter

- Chunk-level index computation
- Recursive parent call from leaf counters



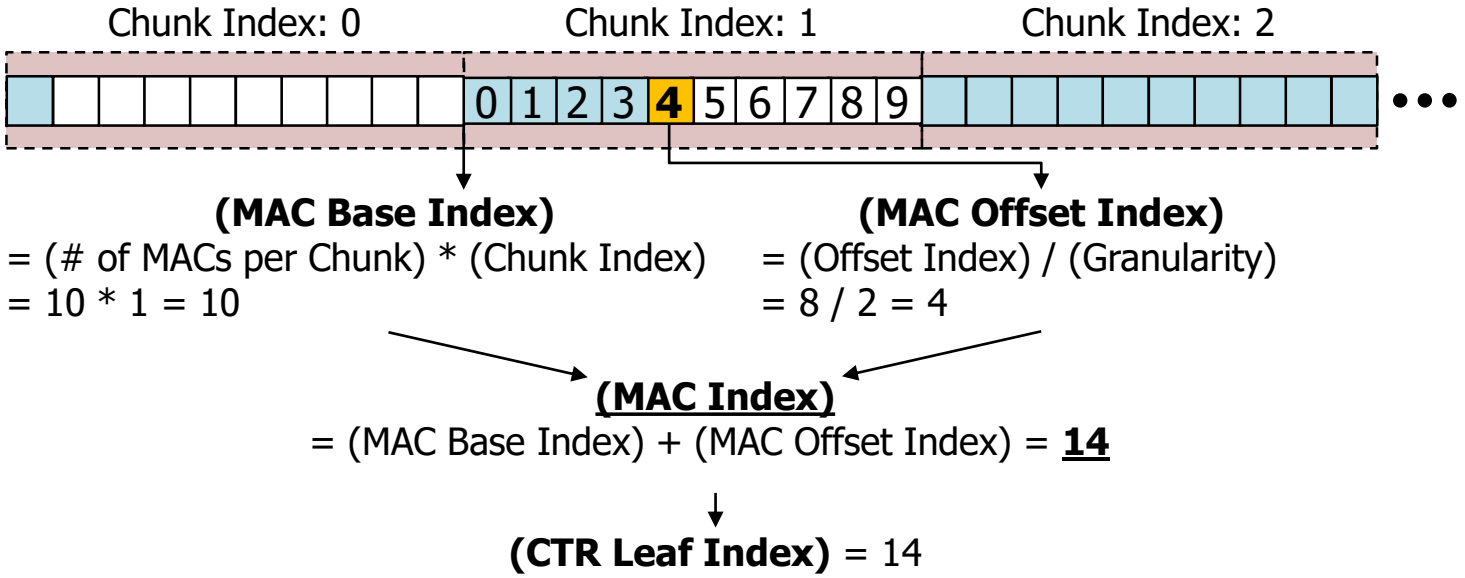
# Address of MAC & Counter

- Chunk-level index computation
- Recursive parent call from leaf counters



# Address of MAC & Counter

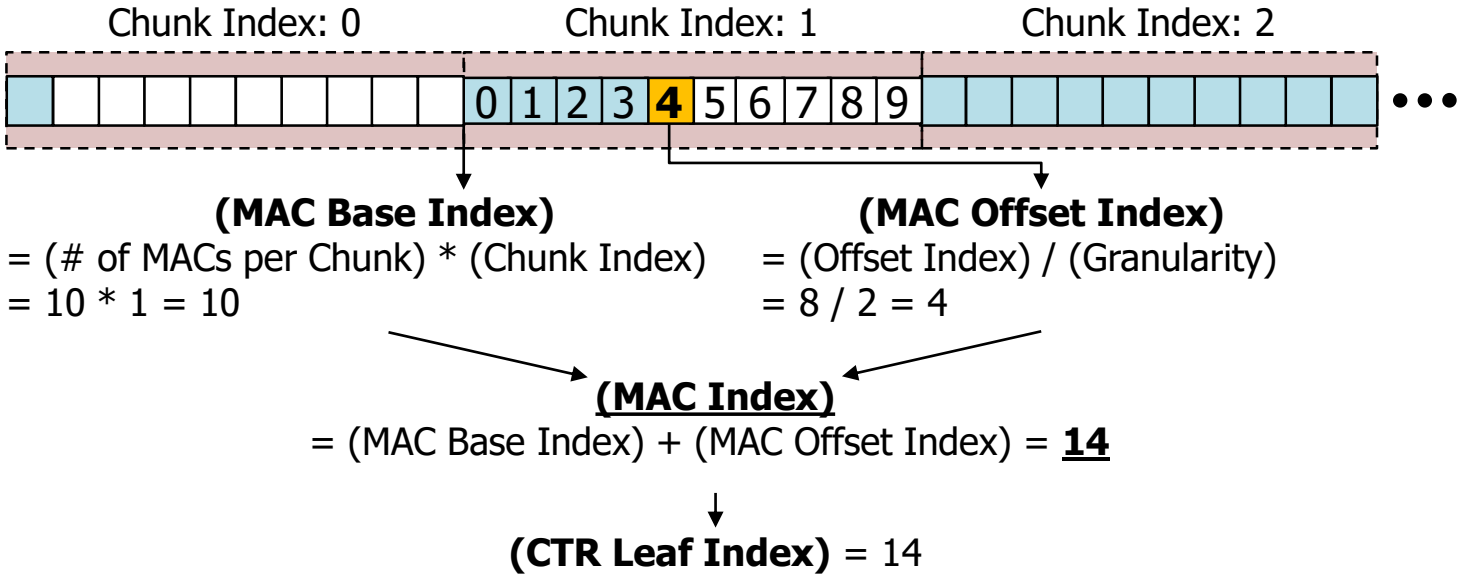
- Chunk-level index computation
- Recursive parent call from leaf counters




**(# of Parents)**

# Address of MAC & Counter

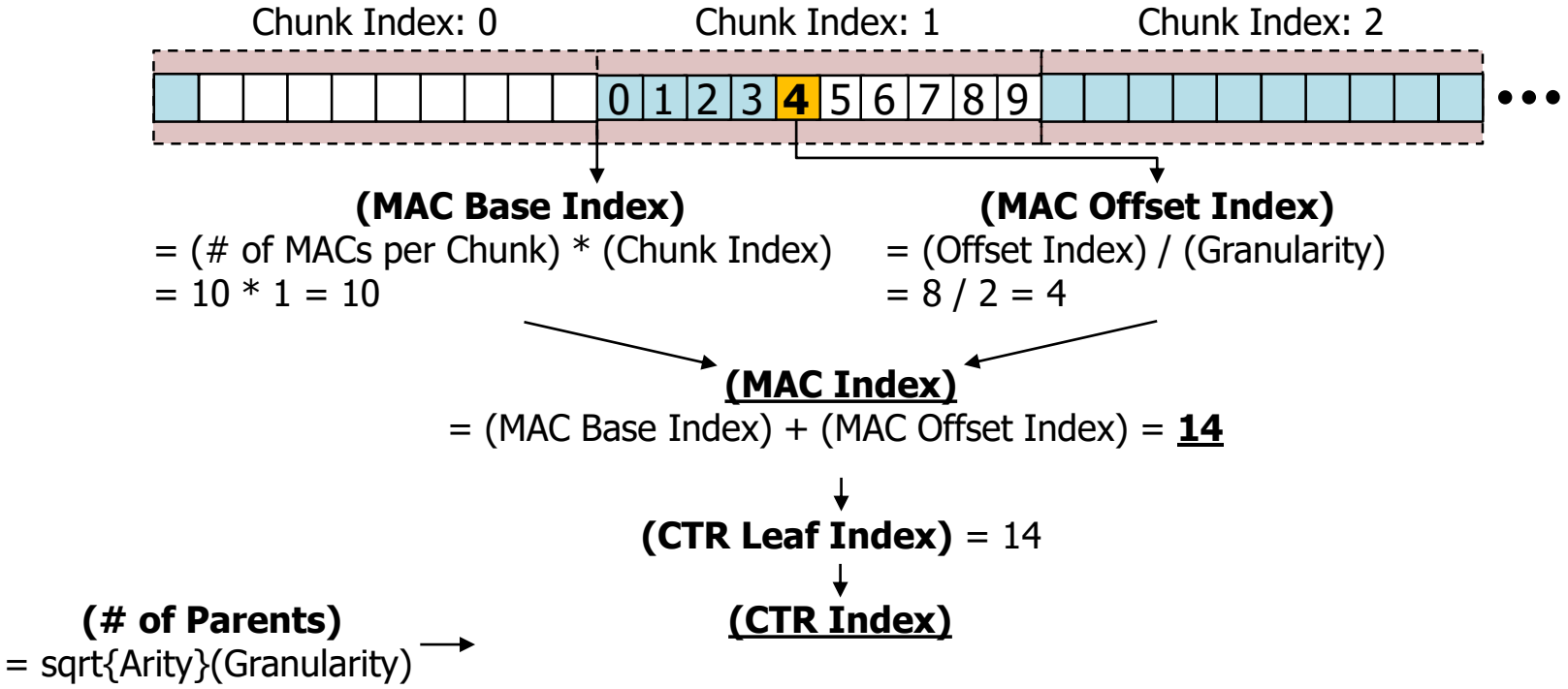
- Chunk-level index computation
- Recursive parent call from leaf counters




**(# of Parents)**
= sqrt{Arity}(Granularity)

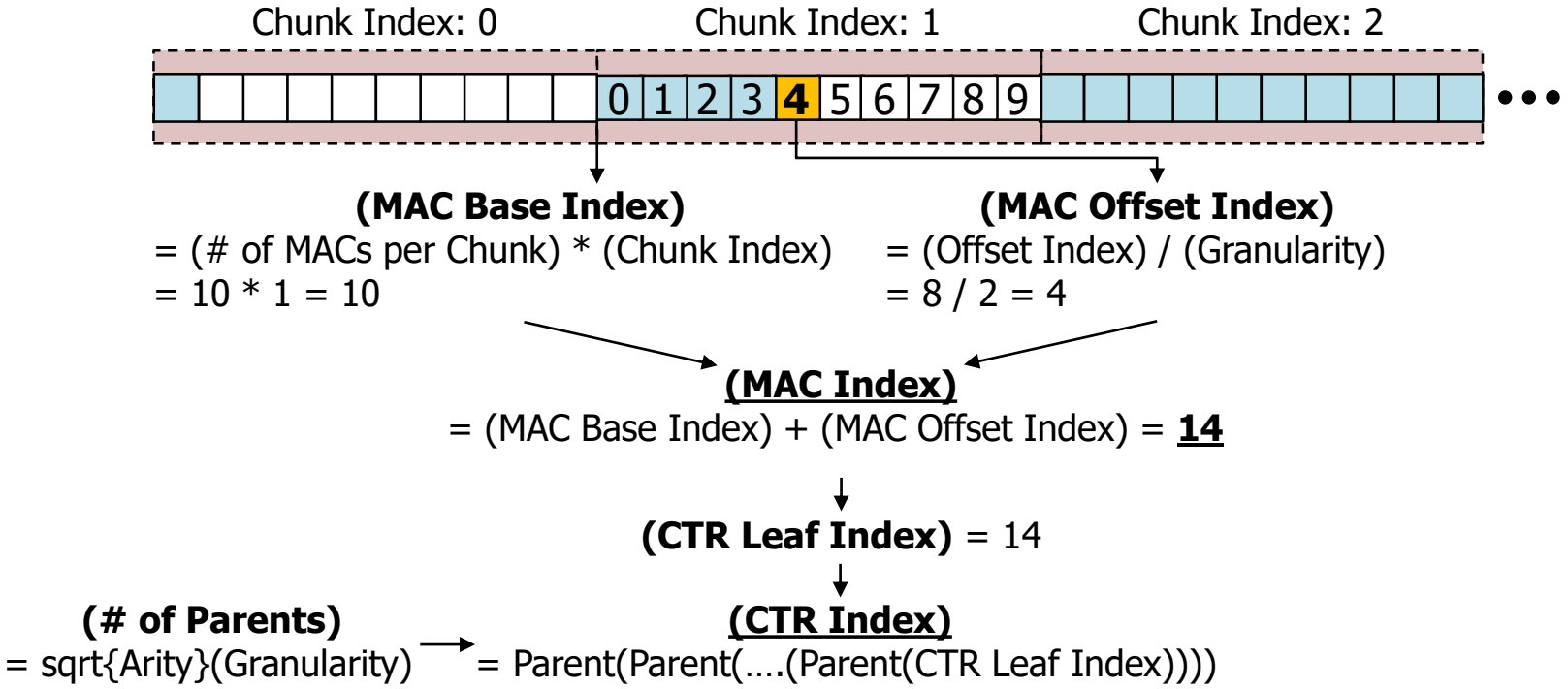
# Address of MAC & Counter

- Chunk-level index computation
- Recursive parent call from leaf counters



# Address of MAC & Counter

- Chunk-level index computation
- Recursive parent call from leaf counters



# Workload Analysis & Selected Scenarios

# Workload Analysis & Selected Scenarios

- Workloads & Scenarios

# Workload Analysis & Selected Scenarios

- Workloads & Scenarios
  - 14 workloads, 250 scenarios (all combinations)
  - Access pattern: Fine – ff – f – c – cc – Coarse | Diverse (d)
  - Traffic per cycles: Small (s) – Medium (m) – Large (l)

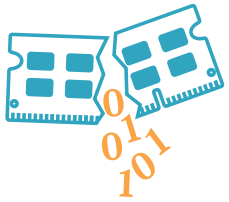
| | Workloads (access pattern-traffic per cycles) |
|---|---|
| CPU | bw (ff-s), gcc (ff-s), mcf (ff-m), xal (f-m), ray (ff-s) |
| GPU | syr2k (ff-m), pr (f-m), sten (c-l), mm (cc-m), floyd (d-s), |
| NPU | ncf (c-s), dlrm (c-s), sfrnn (c-l), alex (cc-m) |

| ID | (CPU, GPU, NPU1, NPU2) |
|---|---|
| ***ff*** | (bw, syr2k, ncf, dlrm), (mcf, syr2k, sfrnn, dlrm), (gcc, floyd, sfrnn, ncf) |
| ***f*** | (xal, pr, sfrnn, ncf), (xal, pr, ncf, ncf) |
| ***c*** | (gcc, sten, alex, dlrm), (bw, sten, ncf, ncf), (mcf, sten, sfrnn, sfrnn) |
| ***cc*** | (xal, mm, alex, dlrm), (ray, mm, alex, alex), (ray, Floyd, alex, alex) |

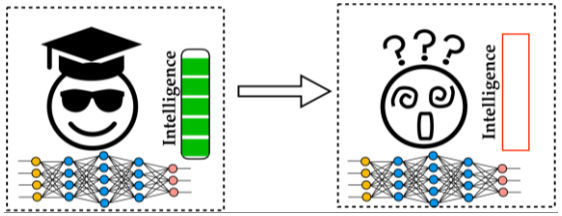
# Rowhammer Attacks

```
1 code1a:
2   mov (X), %eax
3   mov (Y), %ebx
4   clflush (X)
5   clflush (Y)
6   mfence
7   jmp code1a
```

Bit-flip[1]




RamBleed[2]




DeepHammer[3]


[1] Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (ISCA 2014)
[2] RAMBleed: Reading Bits in Memory Without Accessing Them (S&P 2020)
[3] DeepHammer: Depleting the Intelligence of Deep Neural Networks through Targeted Chain of Bit Flips (USENIX Security 2020)

# More Design Descriptions in Our Paper

- Lazy-switching analysis
- Cacheline fragmentation issue
- CTR/MAC addressing for multi-granularity
- Coarse-grained memory protection engine using parallel counter sharing and nested MAC hasing
- Misprediction handler
- Efficient granularity representation
- Hardware overhead
- Comparison to prior subtree optimization schemes

# More Results in Our Paper

- The ratio of stream chunks
- Performance analysis of selected scenarios
- End-to-end performance
- Drawbacks of the per-device (static) granularity
- Performance comparison with dual-granularity
- Switching overhead measurement
- Security cache hit ratio improvement
- Hardware overhead

# Temp Slide

# Research Objective

Constructs a **general** and efficient memory protection scheme for heterogeneous processors

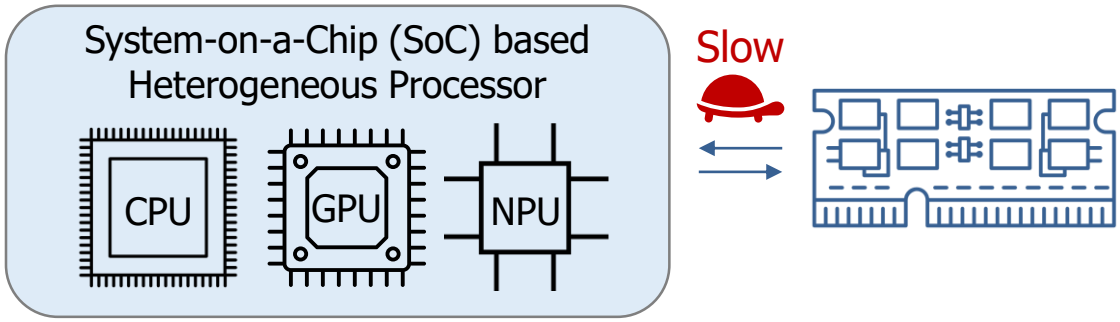
Temp

# Research Objective

Constructs a **general** and efficient memory protection scheme for heterogeneous processors

- Challenge 1: Heterogeneous processors have **diverse access pattern**

**Conventional Memory Protection: High Overhead**

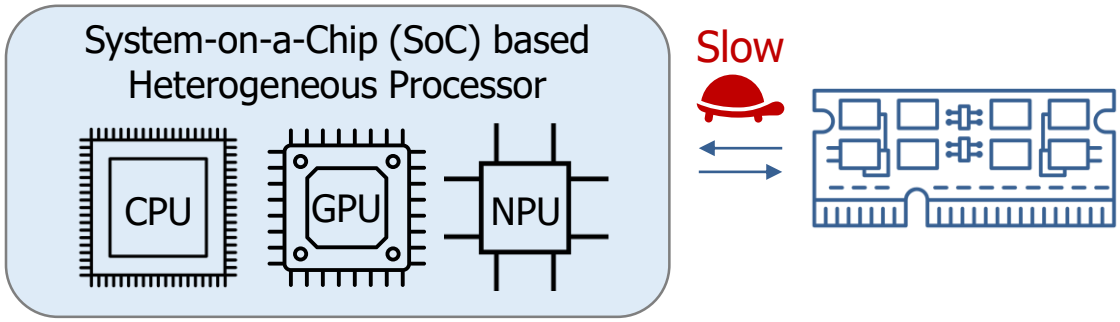


# Research Objective

Constructs a **general** and efficient memory protection scheme for heterogeneous processors

- Challenge 1: Heterogeneous processors have **diverse access pattern**
- Challenge 2: Each prior protection **only for a specific access pattern**

**Conventional Memory Protection: High Overhead**

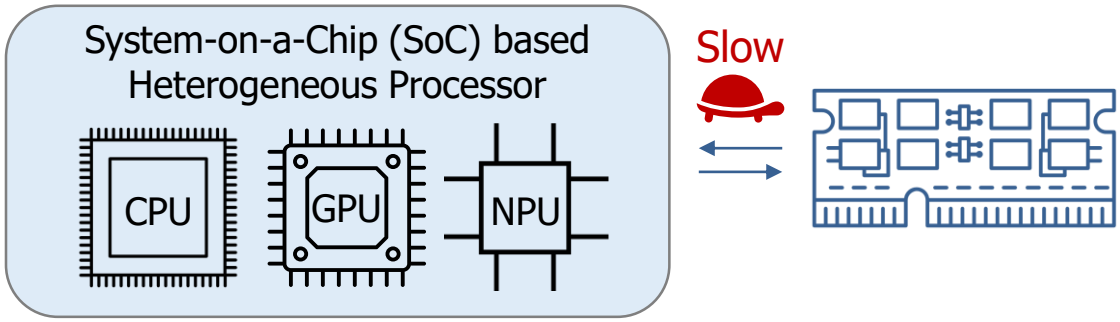


# Research Objective

Constructs a **general** and efficient memory protection scheme for heterogeneous processors

- Challenge 1: Heterogeneous processors have **diverse access pattern**
- Challenge 2: Each prior protection **only for a specific access pattern**
  - For example, GPU coarse-grained pattern, NPU software-detected pattern

**Conventional Memory Protection: High Overhead**



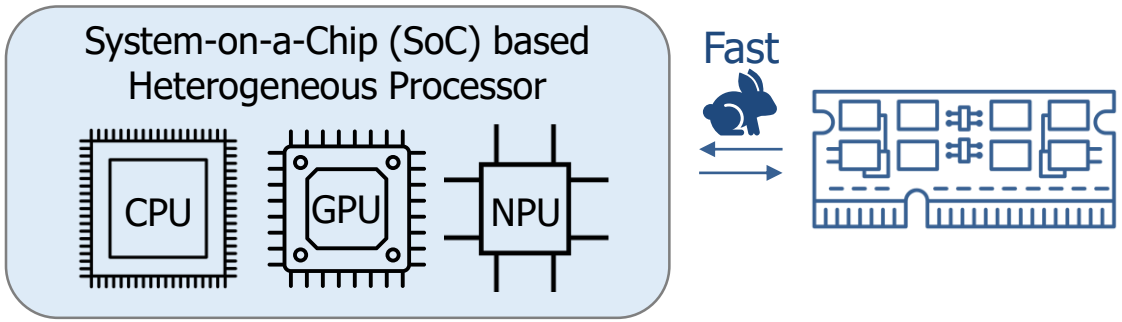
# Research Objective

Constructs a **general** and efficient memory protection scheme for heterogeneous processors

- Challenge 1: Heterogeneous processors have **diverse access pattern**
- Challenge 2: Each prior protection **only for a specific access pattern**
  - For example, GPU coarse-grained pattern, NPU software-detected pattern

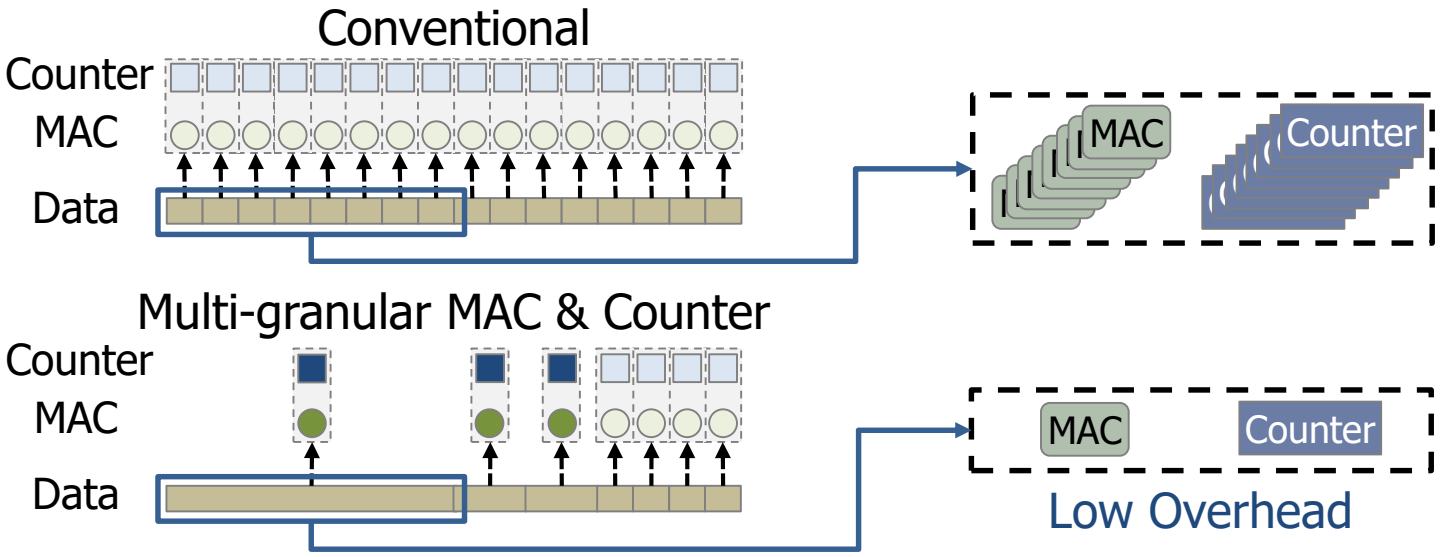
→ We **unified** prior studies with our novel multi-granular tree

**Our Unified Memory Protection Scheme**



# Multi-granular MAC and Counter

- Multi-granular MAC and counter
  - Multi-granular MAC and counter fetches **small # of MACs and counters** for coarse-grained access




**Multi-granularity can reduce memory protection overhead**
**However, how maintain a counter integrity tree?**

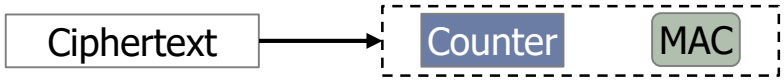
# Prior Memory Protection Schemes

# Prior Memory Protection Schemes

**Counter-mode Protection**

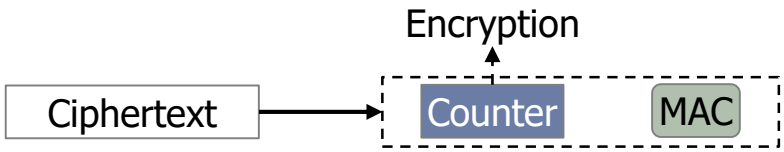
# Prior Memory Protection Schemes

**Counter-mode Protection**



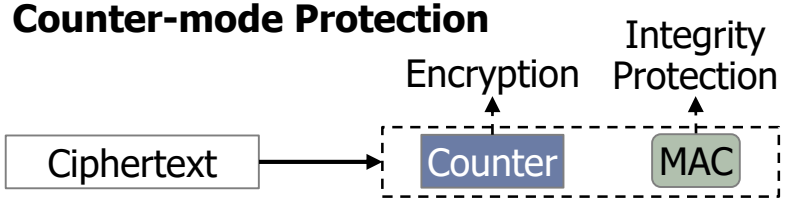
# Prior Memory Protection Schemes

**Counter-mode Protection**



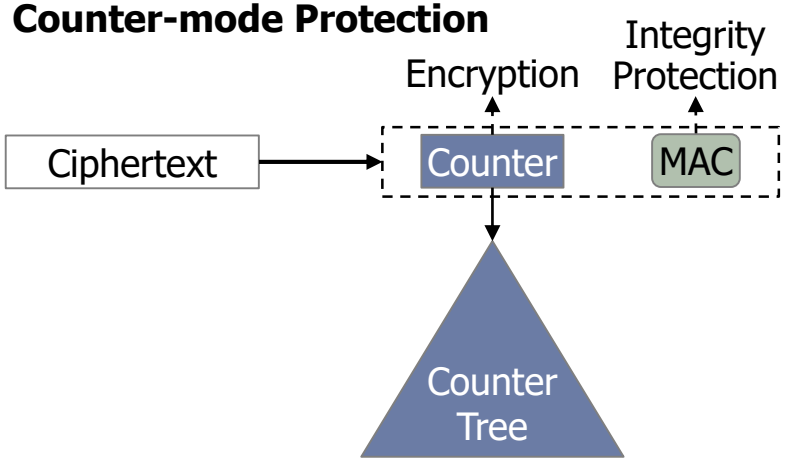
# Prior Memory Protection Schemes

**Counter-mode Protection**



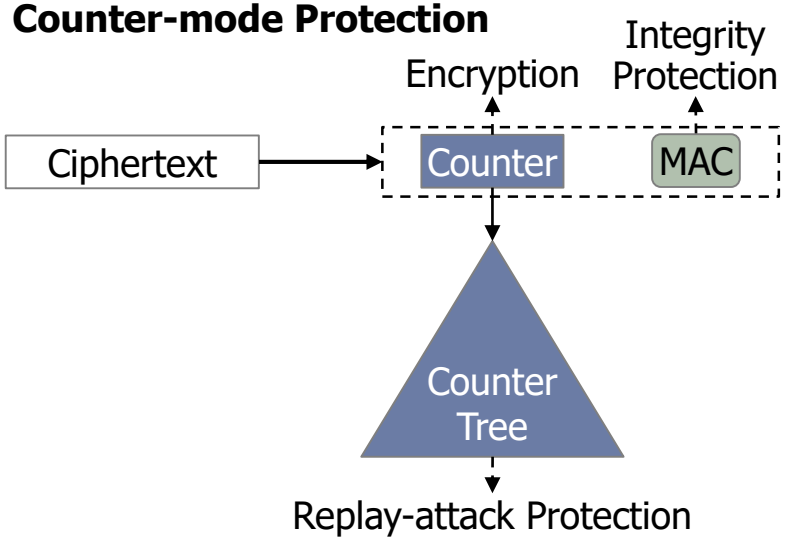
# Prior Memory Protection Schemes

**Counter-mode Protection**



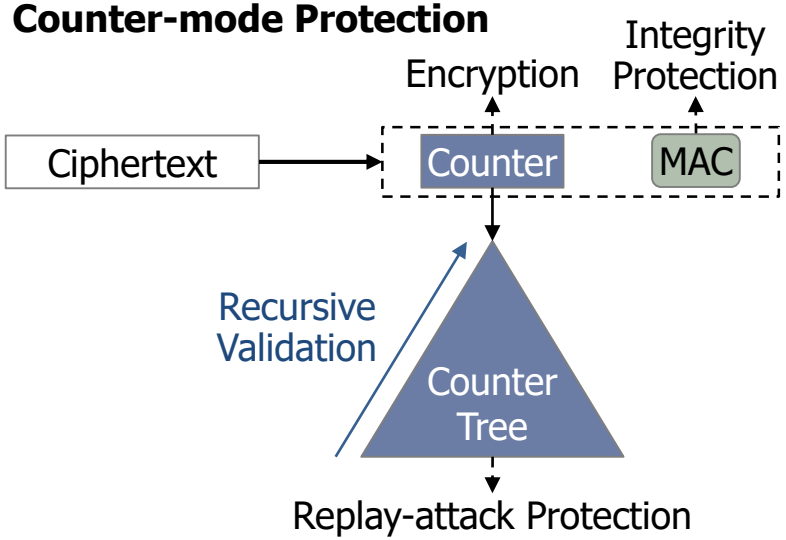
# Prior Memory Protection Schemes

**Counter-mode Protection**

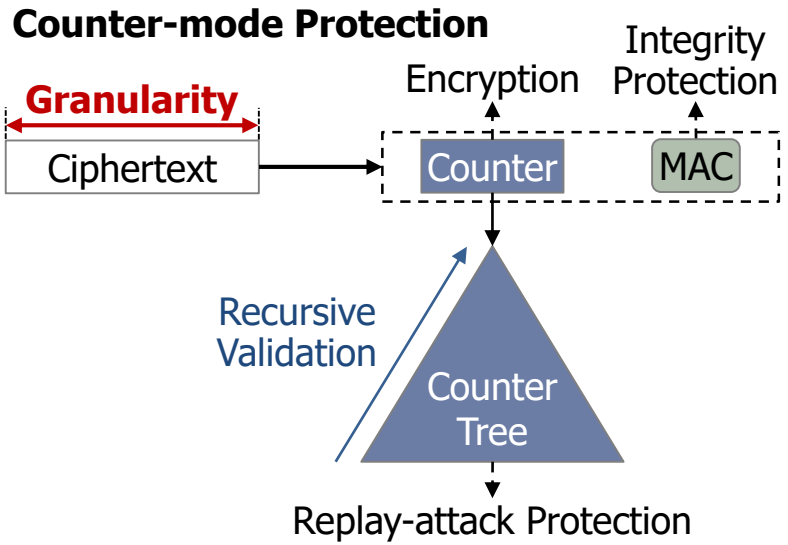


# Prior Memory Protection Schemes

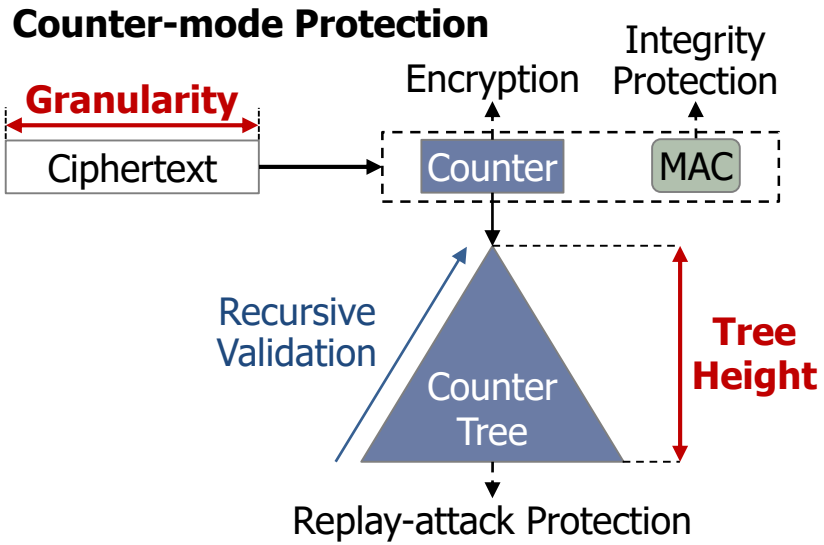
**Counter-mode Protection**



# Prior Memory Protection Schemes

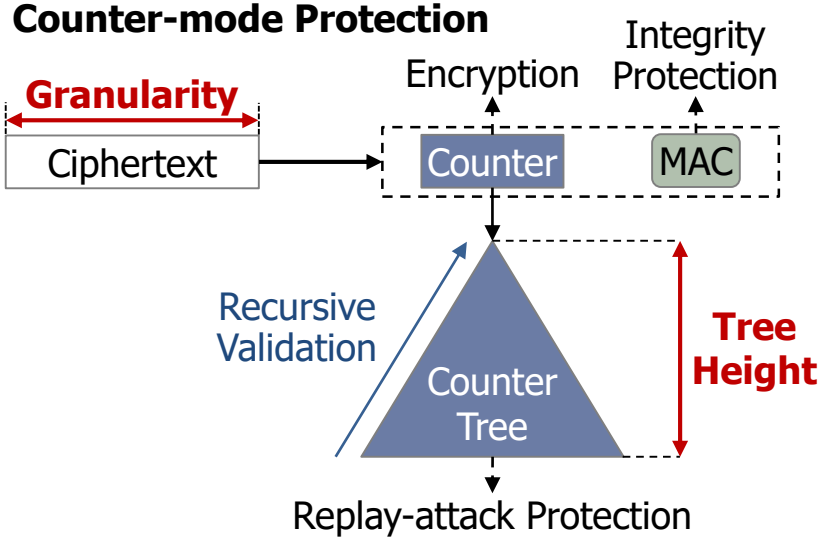


# Prior Memory Protection Schemes



# Prior Memory Protection Schemes

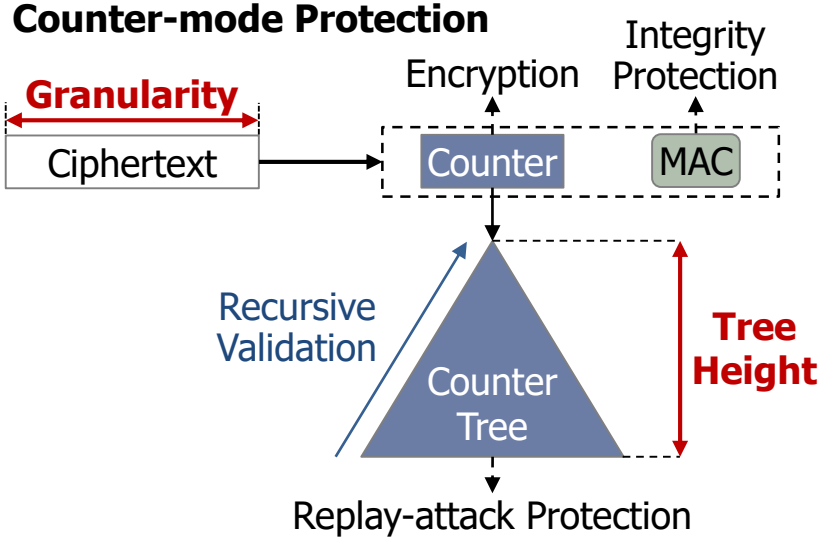
**Counter-mode Protection**



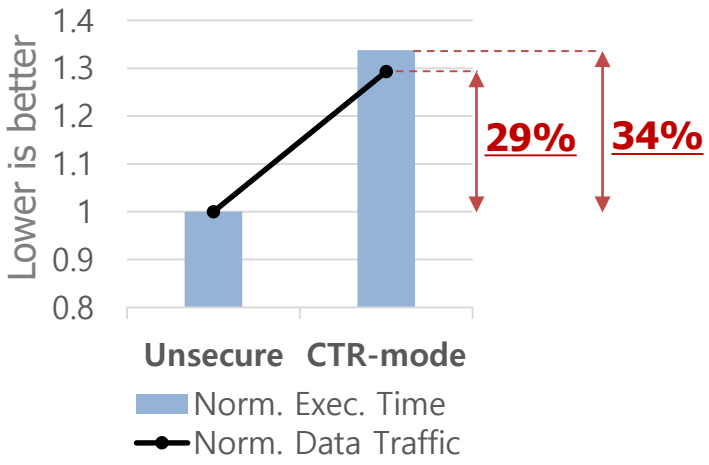

**Conventional 64B-granular Protection**

# Prior Memory Protection Schemes

**Counter-mode Protection**

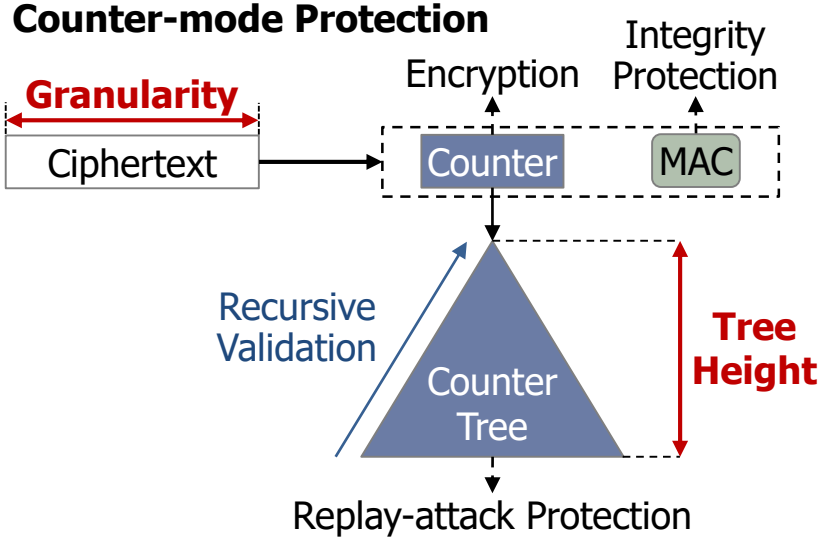



**Conventional 64B-granular Protection**

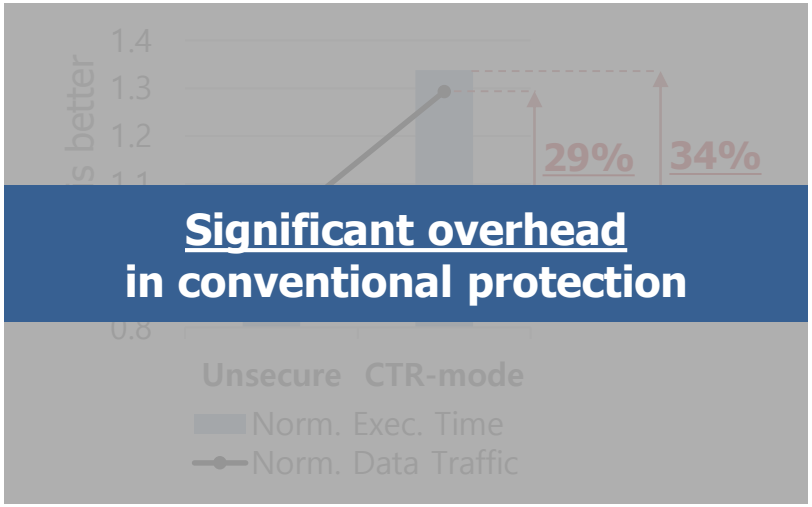


# Prior Memory Protection Schemes

**Counter-mode Protection**

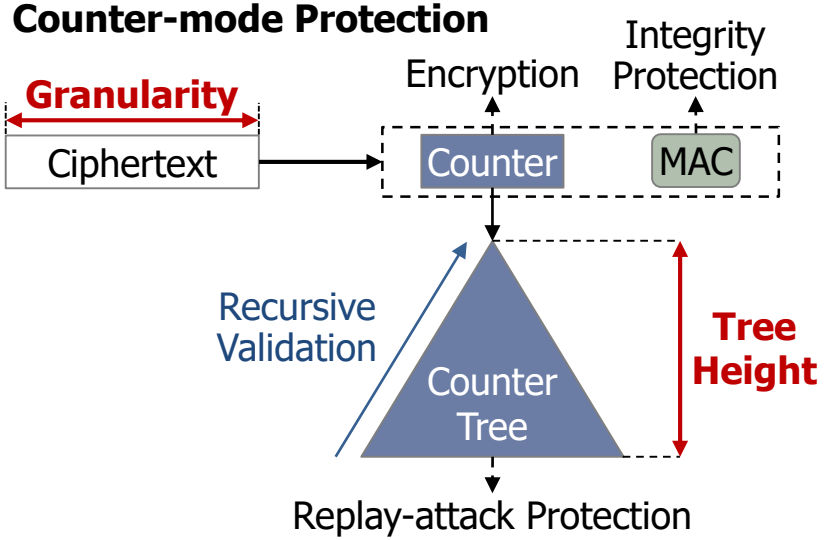



**Conventional 64B-granular Protection**

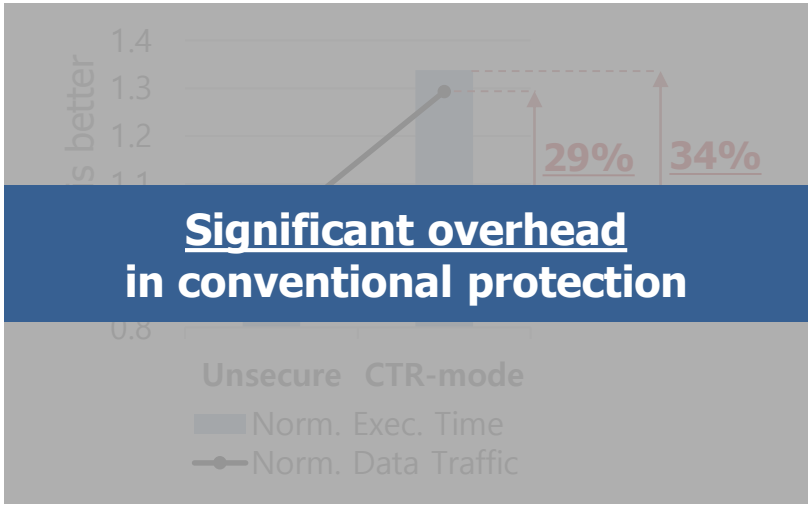


# Prior Memory Protection Schemes

**Counter-mode Protection**



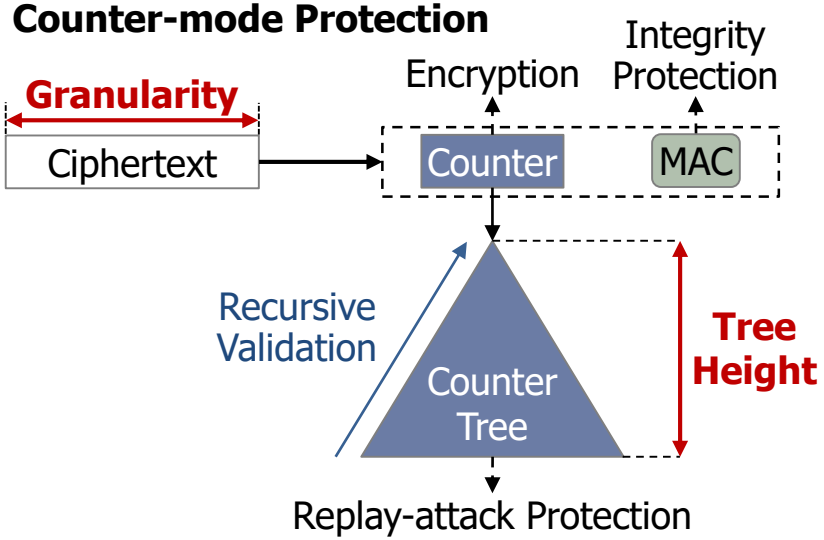

**Conventional 64B-granular Protection**



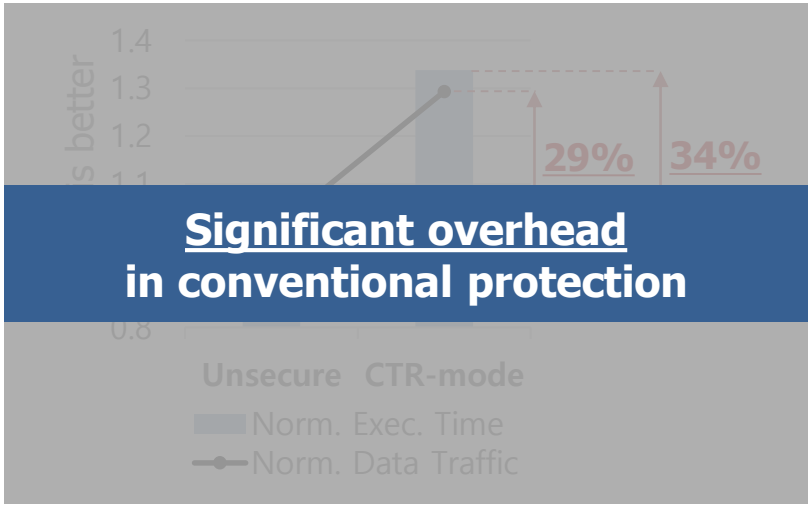

**Prior Domain-specific Memory Protection**

# Prior Memory Protection Schemes

**Counter-mode Protection**




**Conventional 64B-granular Protection**



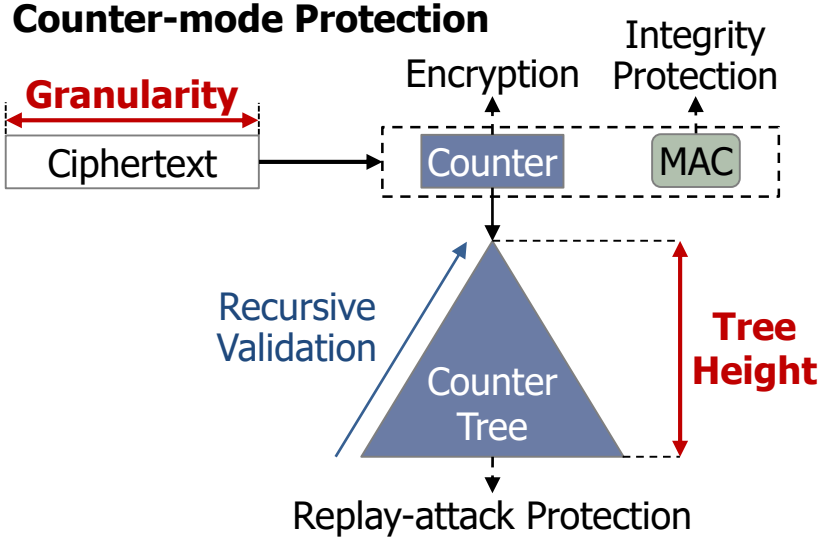

**Prior Domain-specific Memory Protection**

1. Common Counters [1]

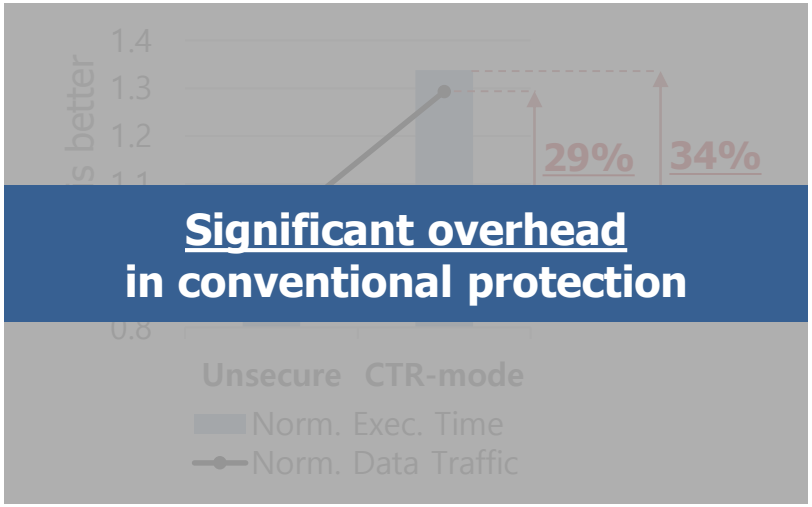
[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)

# Prior Memory Protection Schemes

**Counter-mode Protection**




**Conventional 64B-granular Protection**



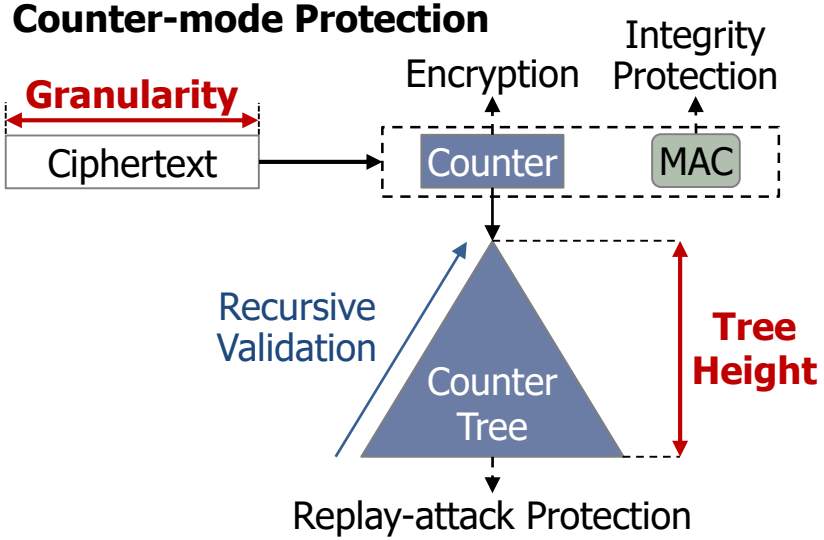

**Prior Domain-specific Memory Protection**

1. Common Counters [1]
2. Dual-MAC [2]

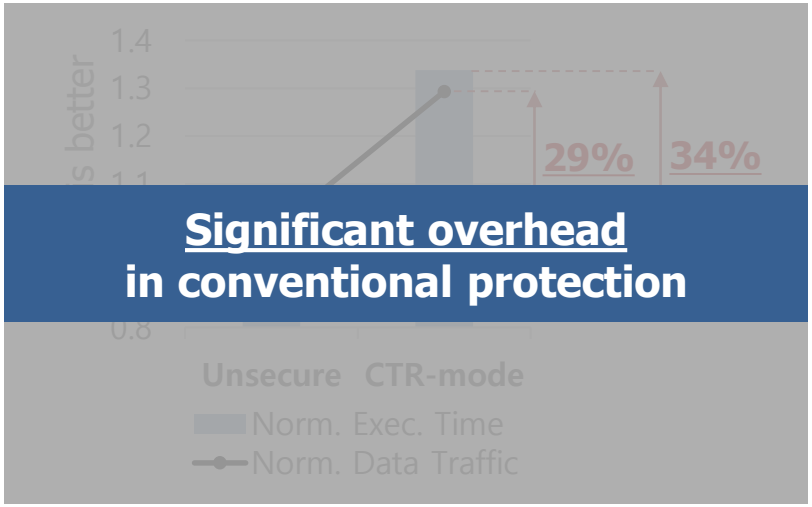
[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)

# Prior Memory Protection Schemes

**Counter-mode Protection**




**Conventional 64B-granular Protection**



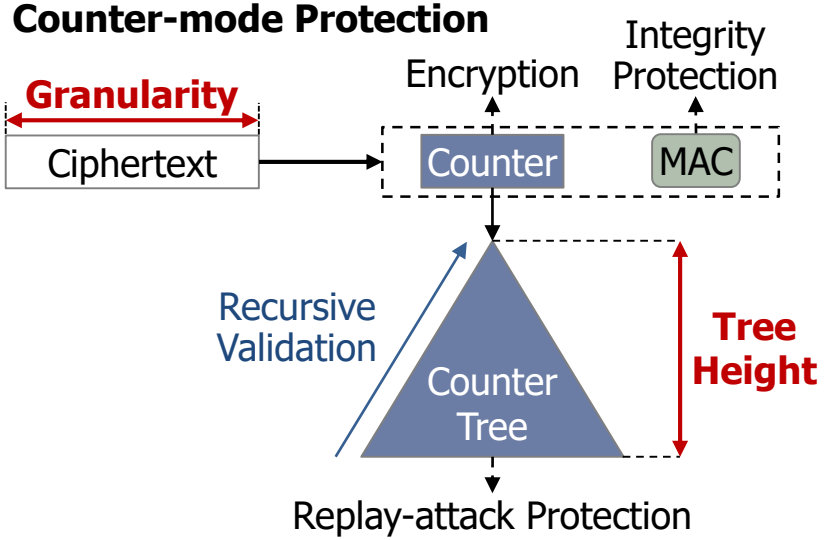

**Prior Domain-specific Memory Protection**

1. Common Counters [1]
2. Dual-MAC [2]
3. Software-managed Granularity [3-4]

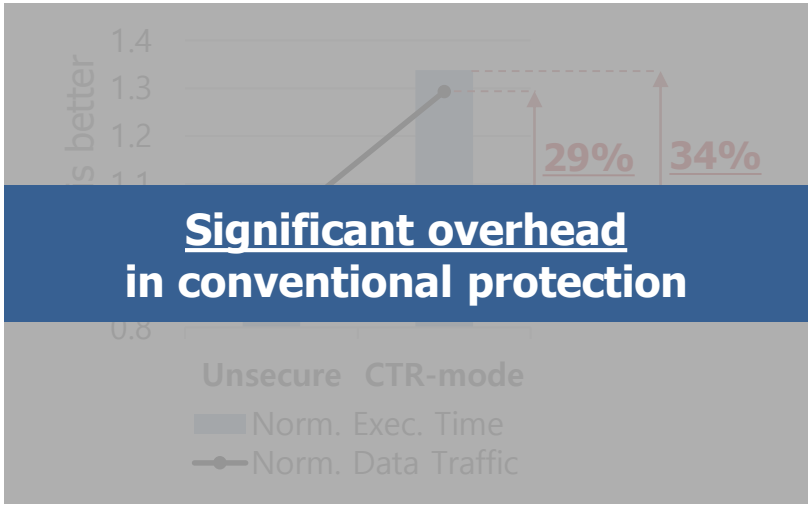
[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)
[3] MGX: Near-zero Overhead Memory Protection for Data-intensive Accelerators (ISCA 2022)
[4] GuardNN: Secure Accelerator Architecture for Privacy-preserving Deep Learning (DAC 2022)

# Prior Memory Protection Schemes

**Counter-mode Protection**

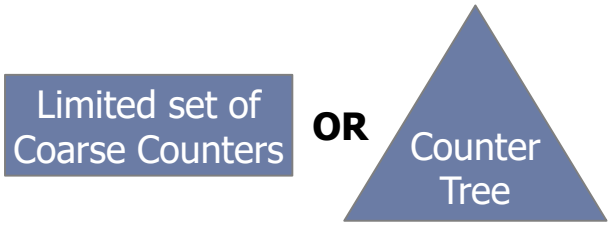



**Conventional 64B-granular Protection**




**Prior Domain-specific Memory Protection**

1. Common Counters [1]



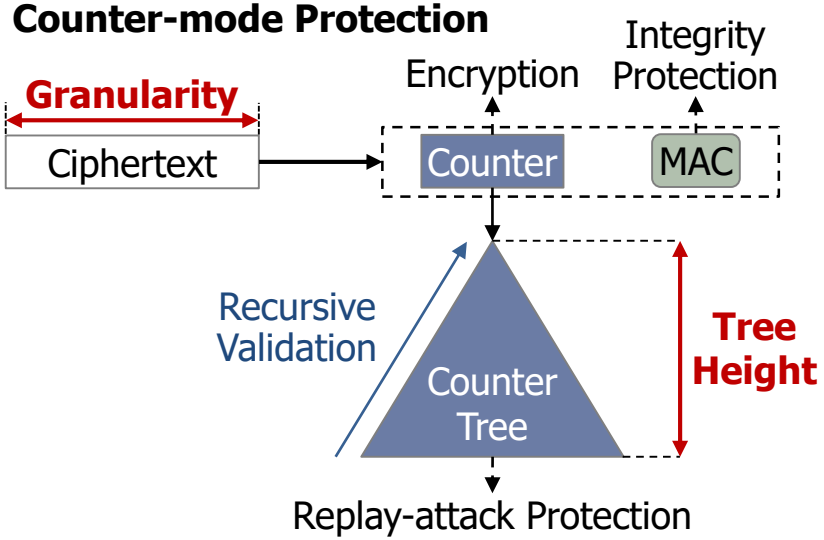

2. Dual-MAC [2]

3. Software-managed Granularity [3-4]

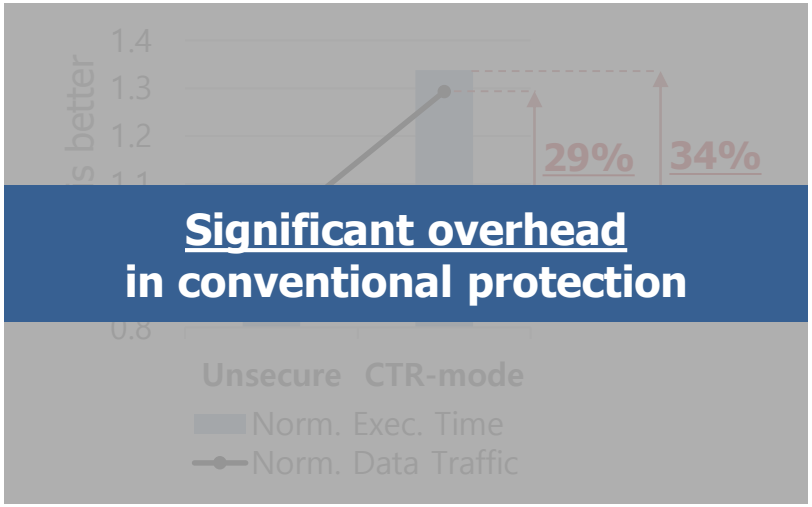

[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)
[3] MGX: Near-zero Overhead Memory Protection for Data-intensive Accelerators (ISCA 2022)
[4] GuardNN: Secure Accelerator Architecture for Privacy-preserving Deep Learning (DAC 2022)

# Prior Memory Protection Schemes

**Counter-mode Protection**

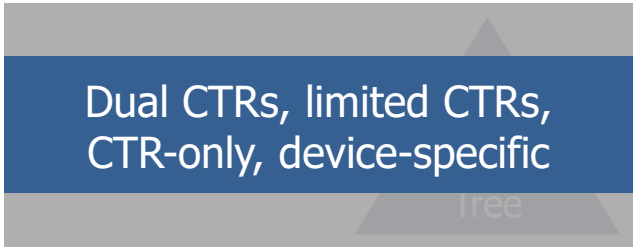



**Conventional 64B-granular Protection**




**Prior Domain-specific Memory Protection**

1. Common Counters [1]
2. Dual-MAC [2]
3. Software-managed Granularity [3-4]

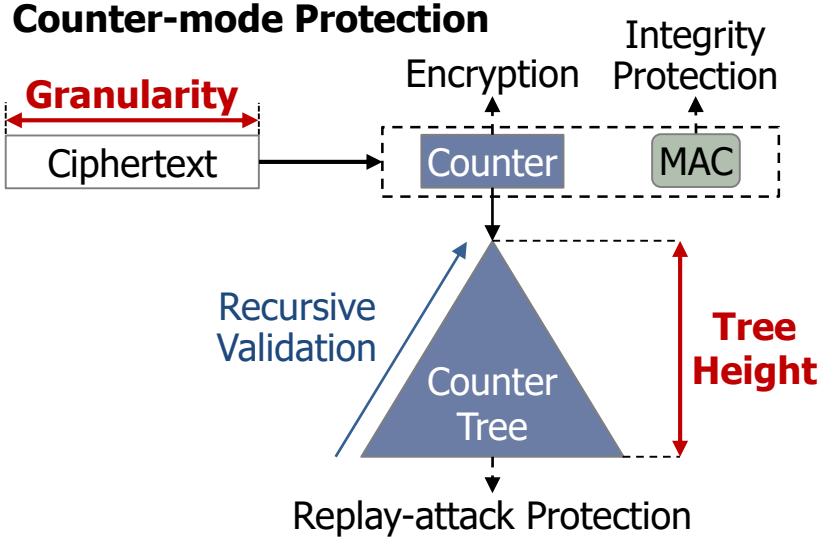



[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)
[3] MGX: Near-zero Overhead Memory Protection for Data-intensive Accelerators (ISCA 2022)
[4] GuardNN: Secure Accelerator Architecture for Privacy-preserving Deep Learning (DAC 2022)

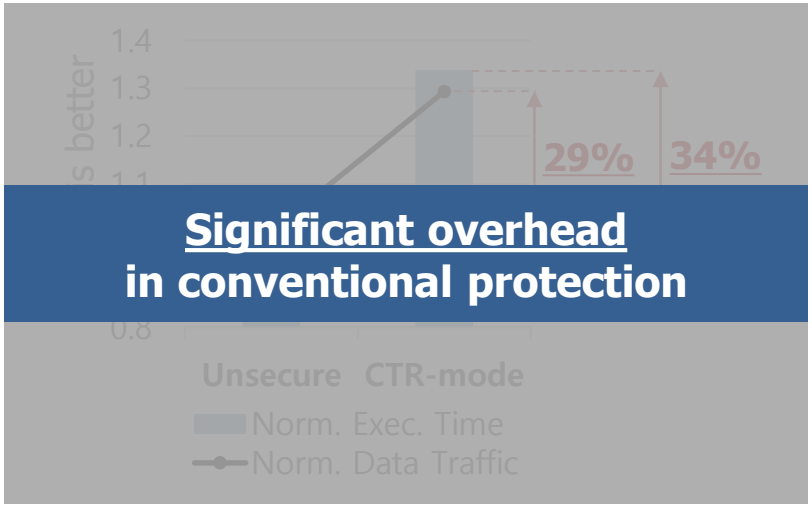
Temp

# Prior Memory Protection Schemes

**Counter-mode Protection**




**Conventional 64B-granular Protection**




**Prior Domain-specific Memory Protection**

1. Common Counters [1]

Dual CTRs, limited CTRs, CTR-only, device-specific

2. Dual-MAC [2]

Coarse MAC

**OR**

Fine MAC

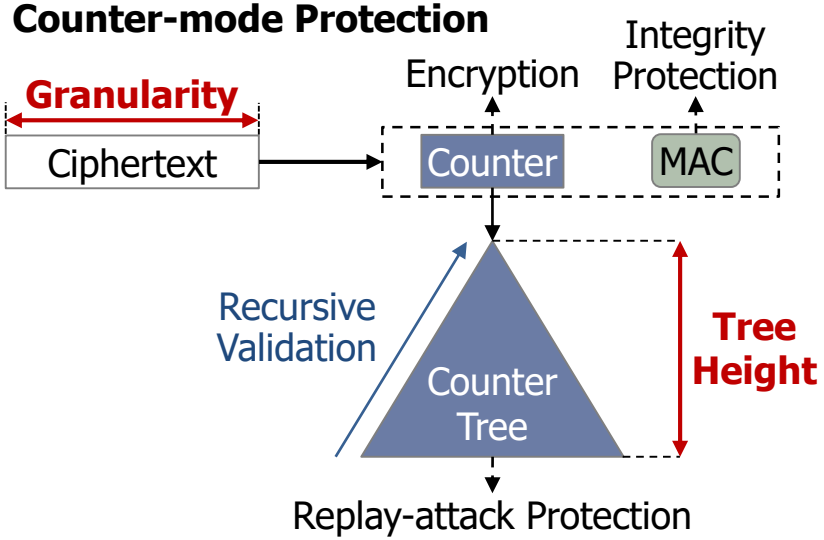
3. Software-managed Granularity [3-4]

[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)
[3] MGX: Near-zero Overhead Memory Protection for Data-intensive Accelerators (ISCA 2022)
[4] GuardNN: Secure Accelerator Architecture for Privacy-preserving Deep Learning (DAC 2022)

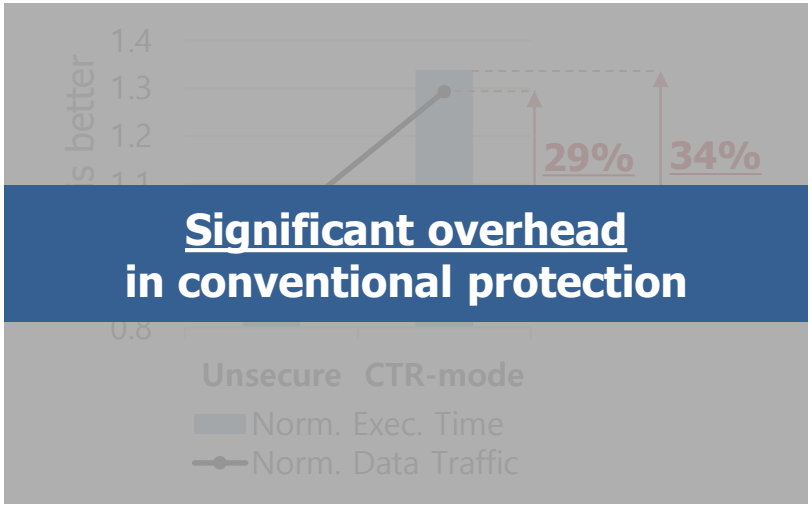
Temp

# Prior Memory Protection Schemes

**Counter-mode Protection**




**Conventional 64B-granular Protection**




**Prior Domain-specific Memory Protection**

1. Common Counters [1]

Dual CTRs, limited CTRs, CTR-only, device-specific

2. Dual-MAC [2]

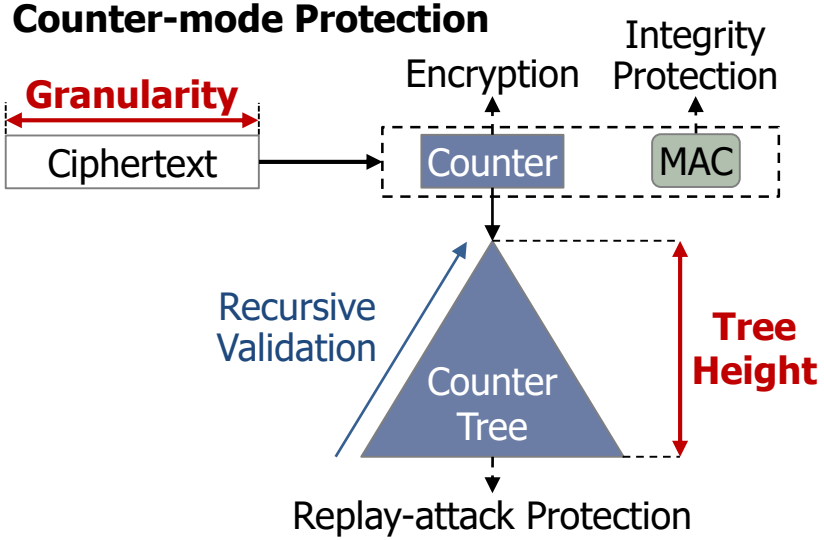
Dual MACs, MAC-only

3. Software-managed Granularity [3-4]

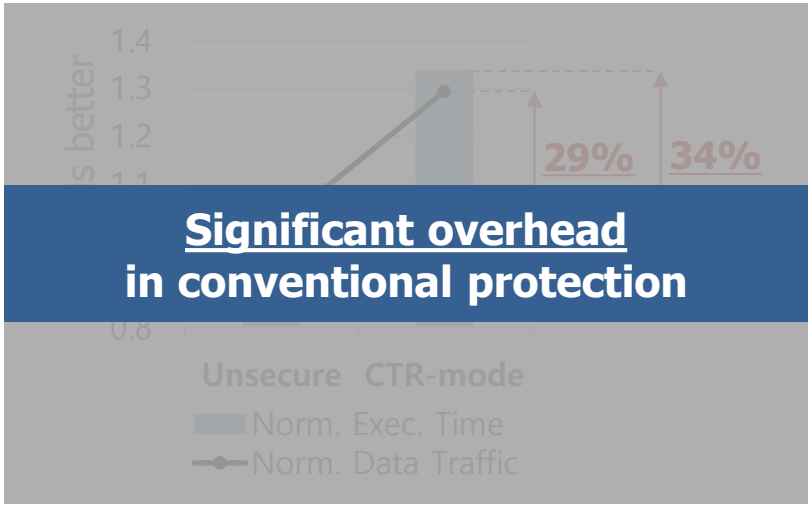
[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)
[3] MGX: Near-zero Overhead Memory Protection for Data-intensive Accelerators (ISCA 2022)
[4] GuardNN: Secure Accelerator Architecture for Privacy-preserving Deep Learning (DAC 2022)

# Prior Memory Protection Schemes

### Counter-mode Protection




### Conventional 64B-granular Protection




### Prior Domain-specific Memory Protection

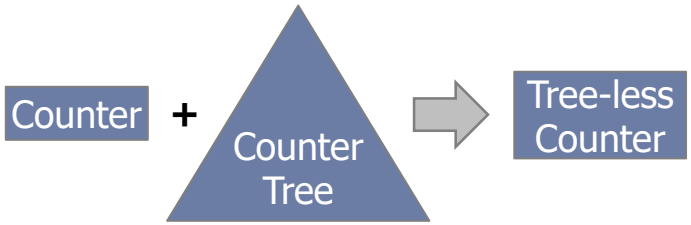
1. Common Counters [1]

Dual CTRs, limited CTRs, CTR-only, device-specific

2. Dual-MAC [2]

Dual MACs, MAC-only

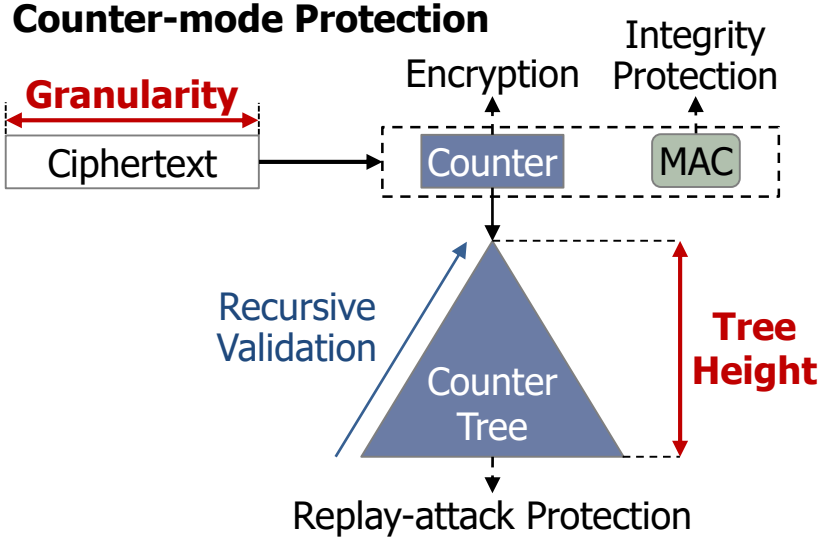
3. Software-managed Granularity [3-4]



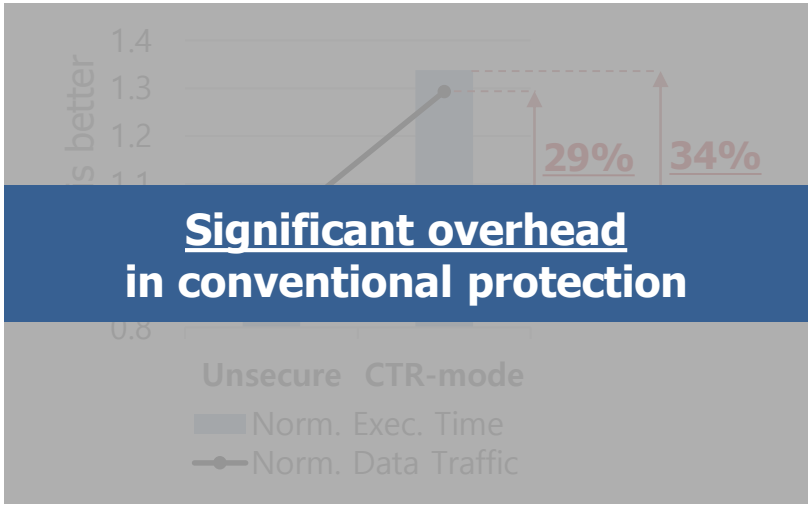

[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)
[3] MGX: Near-zero Overhead Memory Protection for Data-intensive Accelerators (ISCA 2022)
[4] GuardNN: Secure Accelerator Architecture for Privacy-preserving Deep Learning (DAC 2022)

# Prior Memory Protection Schemes

**Counter-mode Protection**




**Conventional 64B-granular Protection**




**Prior Domain-specific Memory Protection**

1. Common Counters [1]

Dual CTRs, limited CTRs, CTR-only, device-specific

2. Dual-MAC [2]

Dual MACs, MAC-only

3. Software-managed Granularity [3-4]

Application-specific, device-specific

[1] Common Counters: Compressed Encryption Counters for Secure GPU Memory (HPCA 2021)
[2] Adaptive Security Support for Heterogeneous Memory on GPUs (HPCA 2022)
[3] MGX: Near-zero Overhead Memory Protection for Data-intensive Accelerators (ISCA 2022)
[4] GuardNN: Secure Accelerator Architecture for Privacy-preserving Deep Learning (DAC 2022)

Temp