D-Shield: Enabling Processor-side Encryption and Integrity Verification for Secure NVMe Drives

Md Hafizul Islam Chowdhuryy
University of Central Florida

Myoungsoo Jung
KAIST

Fan Yao
University of Central Florida

Amro Awad
NC State University

Presenter: Md Hafizul Islam Chowdhuryy
Computer Architecture and Systems Research Lab, University of Central Florida

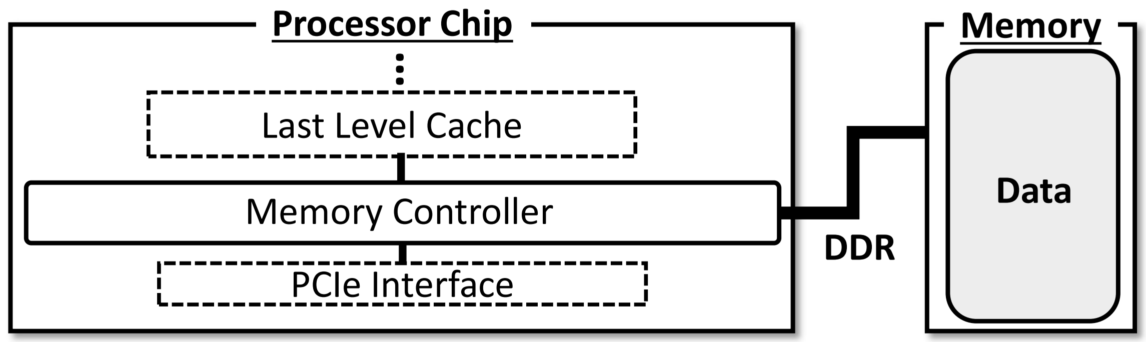
29th IEEE International Symposium on High-Performance Computer Architecture (**HPCA**-29)
February 25th – March 1st, 2023

Hardware Security Threats to System

- Data is the main target of exploitation
- Multiple attack vectors are possible for off-chip data

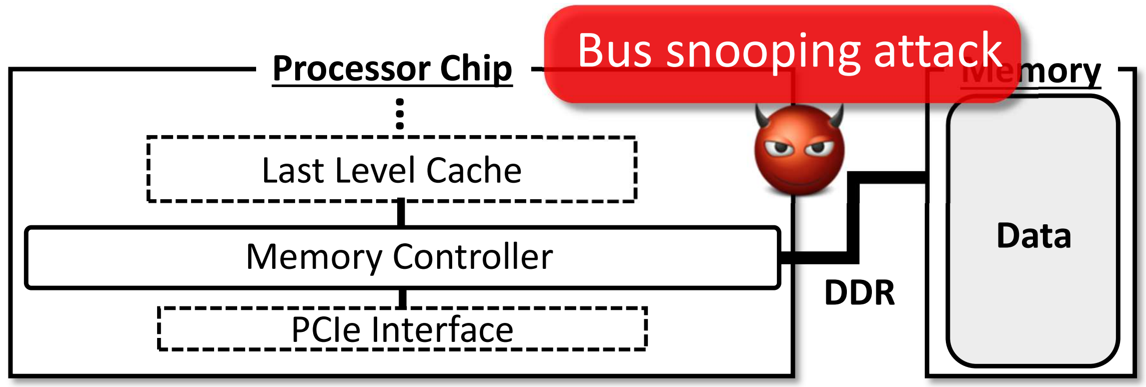
Hardware Security Threats to System

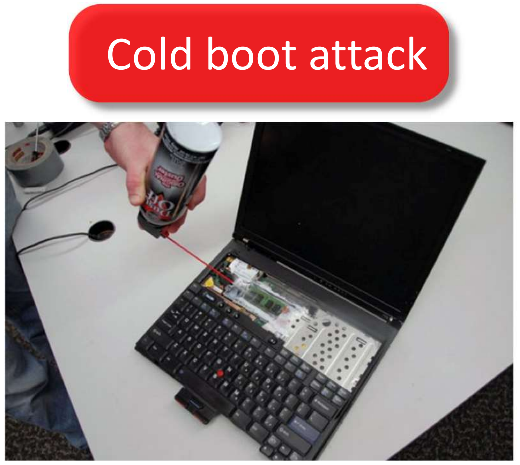
- Data is the main target of exploitation
- Multiple attack vectors are possible for off-chip data



Hardware Security Threats to System

- Data is the main target of exploitation
- Multiple attack vectors are possible for off-chip data

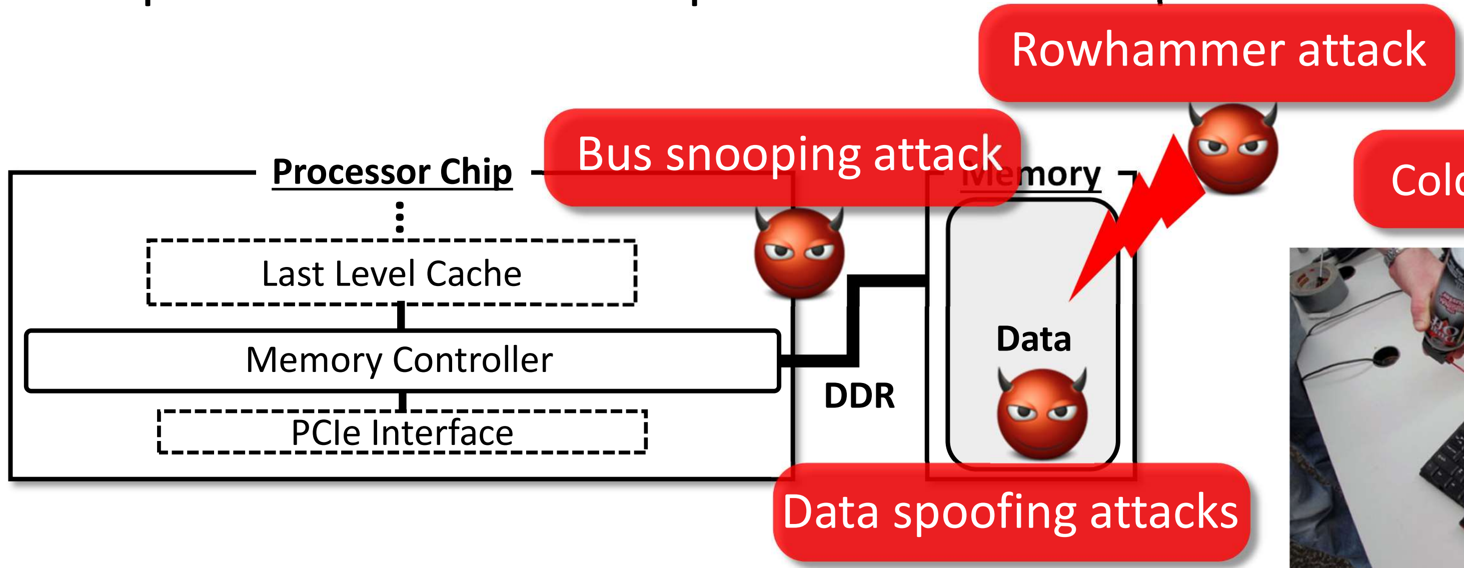




Source: iinfosecinstitute.com

Hardware Security Threats to System

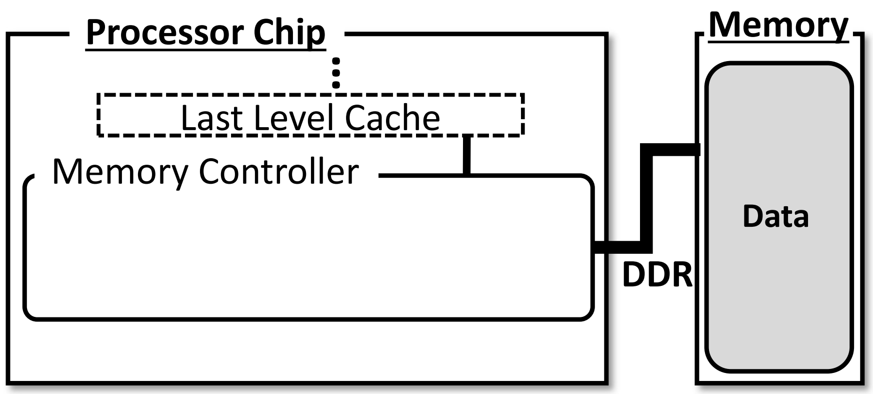
- Data is the main target of exploitation
- Multiple attack vectors are possible for off-chip data



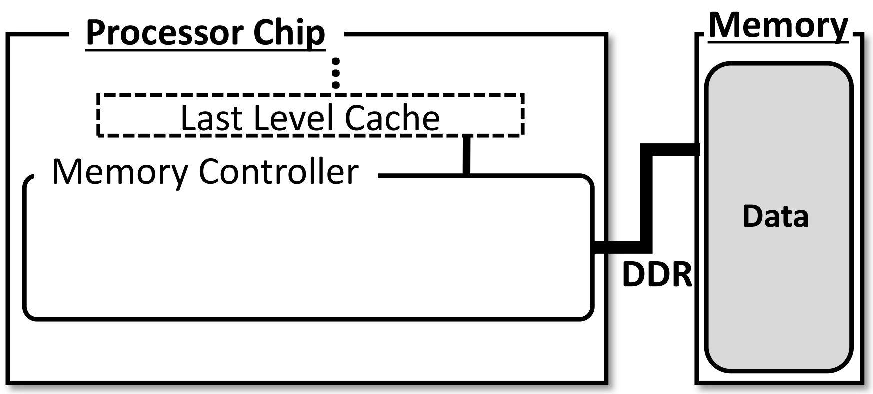
Cold boot attack

Source: iinfosecinstitute.com

Secure Memory Architecture

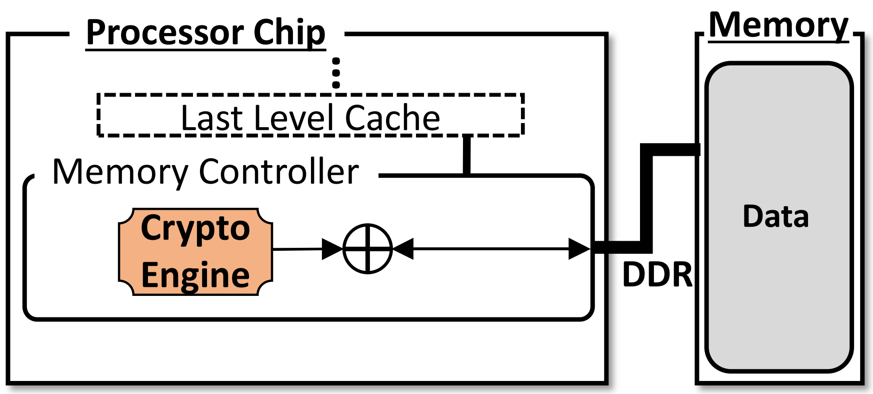


Secure Memory Architecture



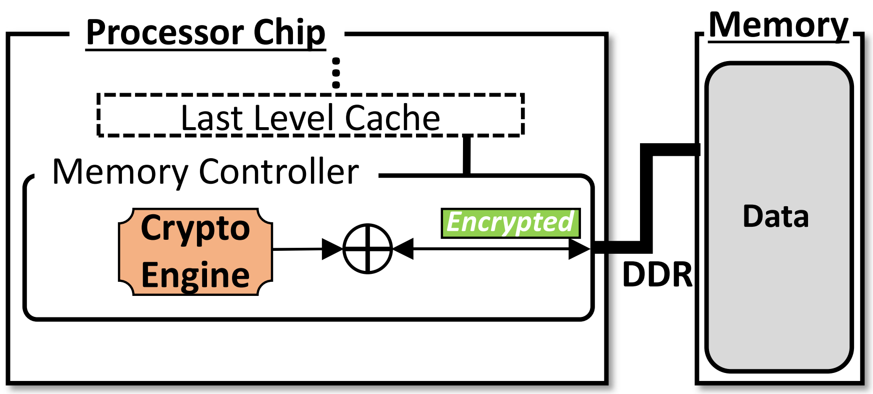
Limit trust boundary to the processor chip
Protect confidentiality and integrity of **off-chip data**

Secure Memory Architecture



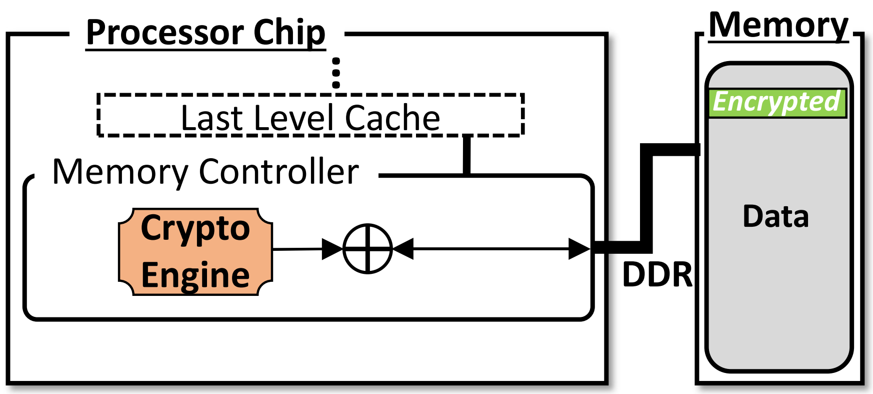
Limit trust boundary to the processor chip
Protect confidentiality and integrity of **off-chip data**

Secure Memory Architecture



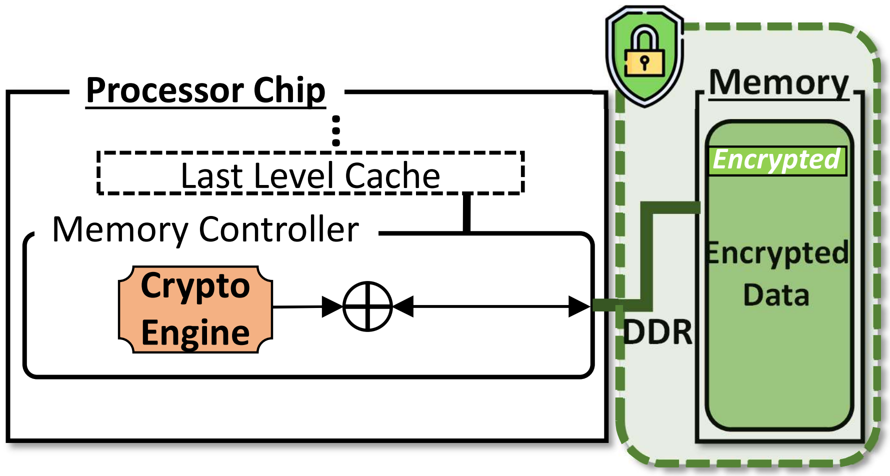
Limit trust boundary to the processor chip
Protect confidentiality and integrity of **off-chip data**

Secure Memory Architecture



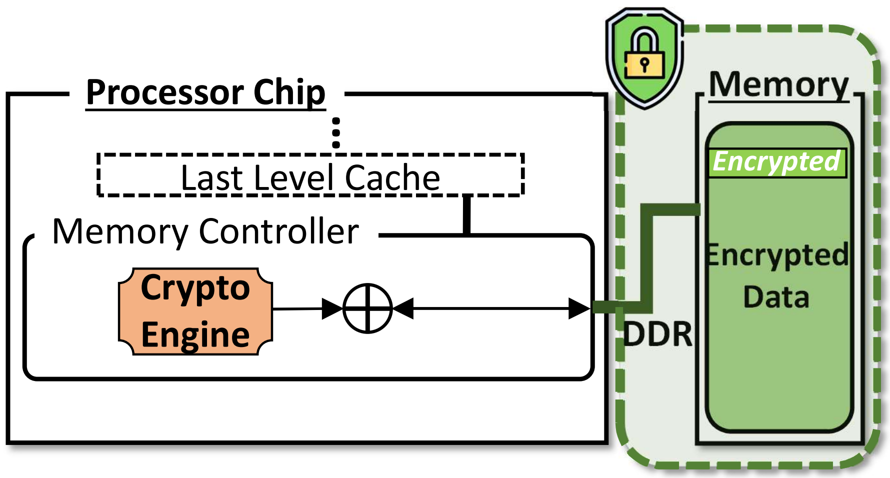
Limit trust boundary to the processor chip
Protect confidentiality and integrity of **off-chip data**

Secure Memory Architecture



Limit trust boundary to the processor chip
Protect confidentiality and integrity of **off-chip data**

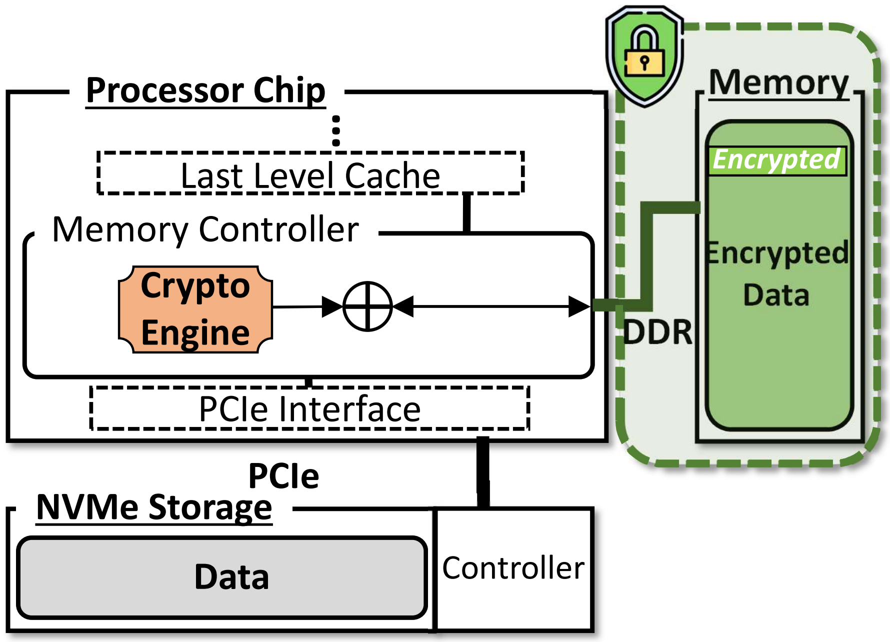
Secure Memory Architecture



Limit trust boundary to the processor chip
Protect confidentiality and integrity of **off-chip data**

Mainly focus on **memory security**

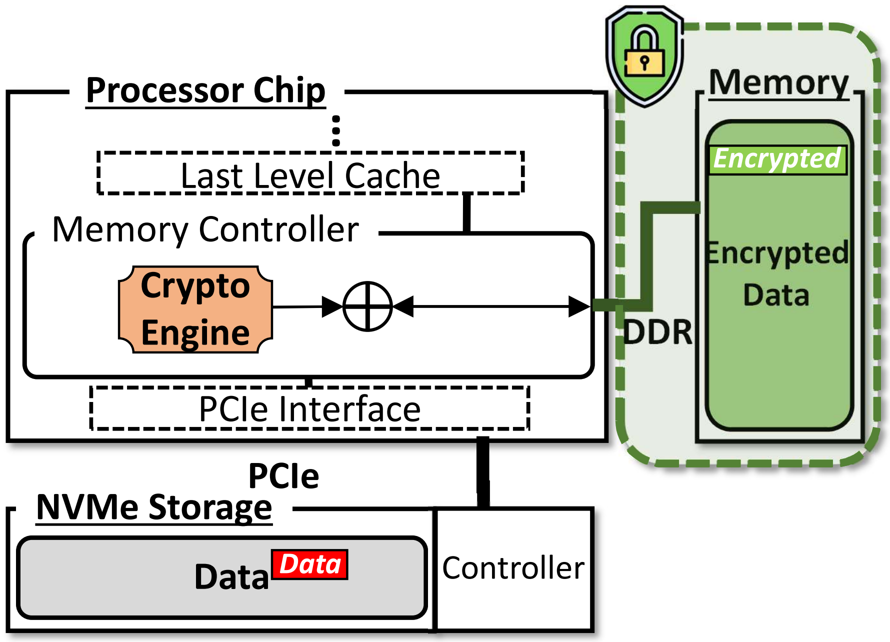
Secure Memory Architecture



Limit trust boundary to the processor chip
Protect confidentiality and integrity of **off-chip data**

Mainly focus on **memory security**

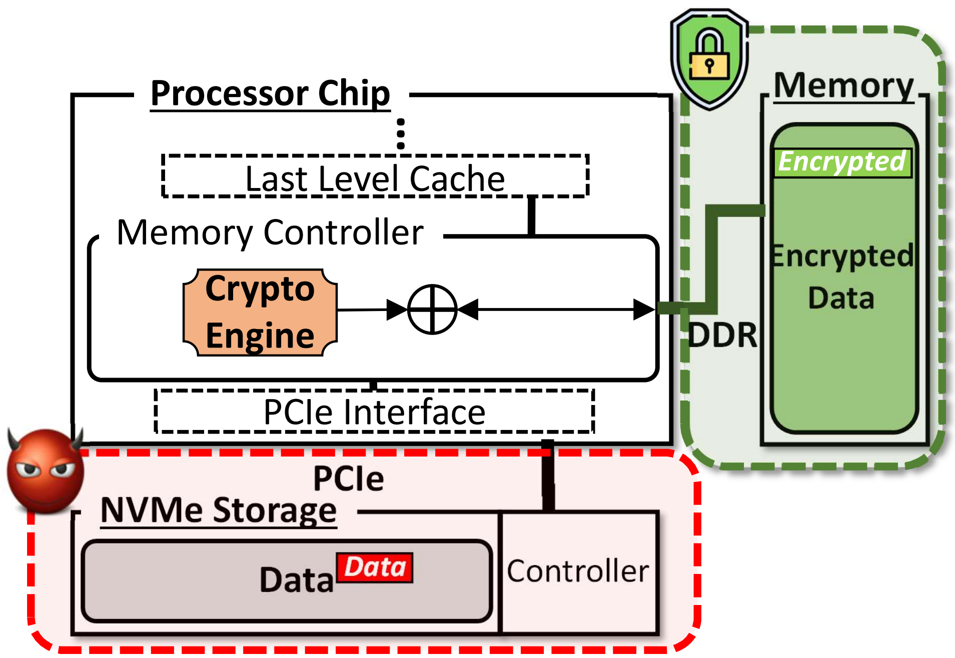
Secure Memory Architecture



Limit trust boundary to the processor chip
Protect confidentiality and integrity of **off-chip data**

Mainly focus on **memory security**

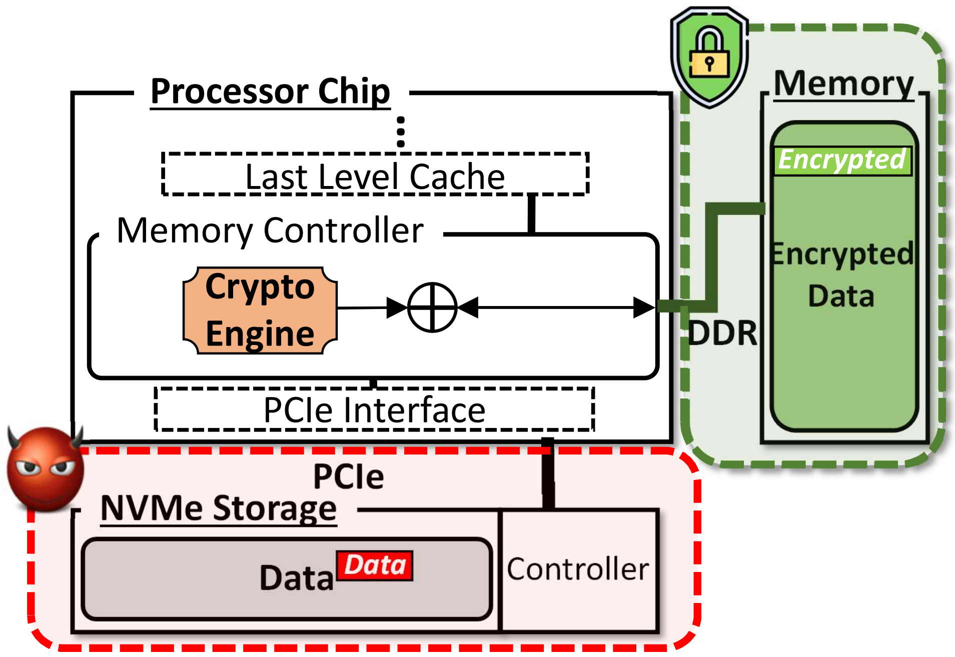
Secure Memory Architecture



Limit trust boundary to the processor chip
Protect confidentiality and integrity of **off-chip data**

Mainly focus on **memory security**

Secure Memory Architecture



Limit trust boundary to the processor chip
Protect confidentiality and integrity of **off-chip data**

Mainly focus on **memory security**

Storage security provided through **software** or **disk** itself
No processor-side support for fast storage security

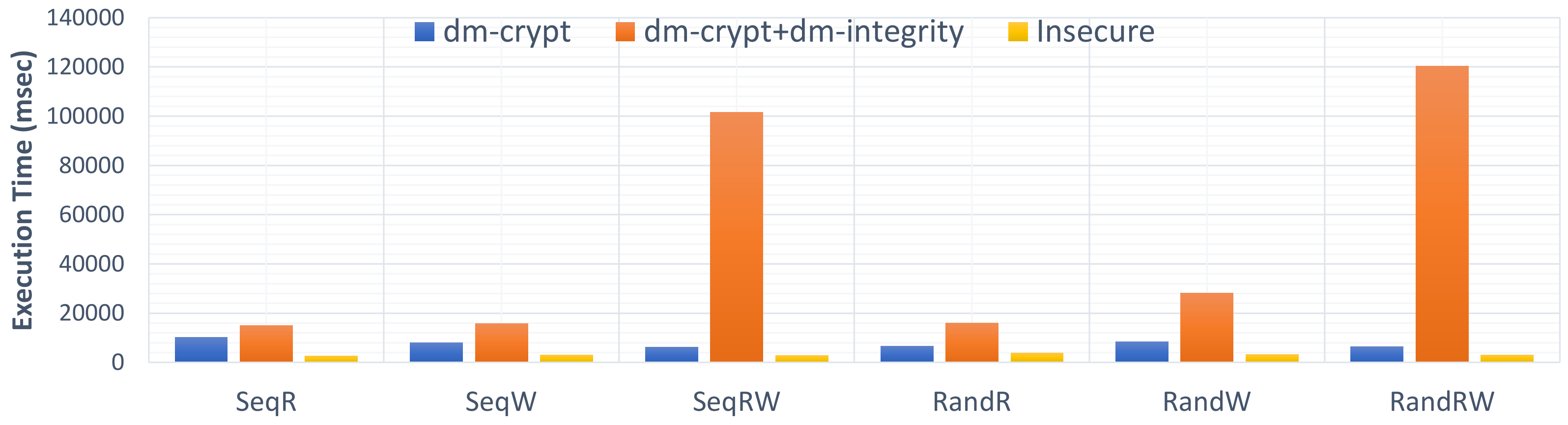
UCF

Need for Architectural Support for Storage Security

- Pitfalls of existing solutions:
    - **Self-encryption disks:** encrypts data in the storage itself
        - Do not protect physical attacks (i.e., bus snooping)

Need for Architectural Support for Storage Security

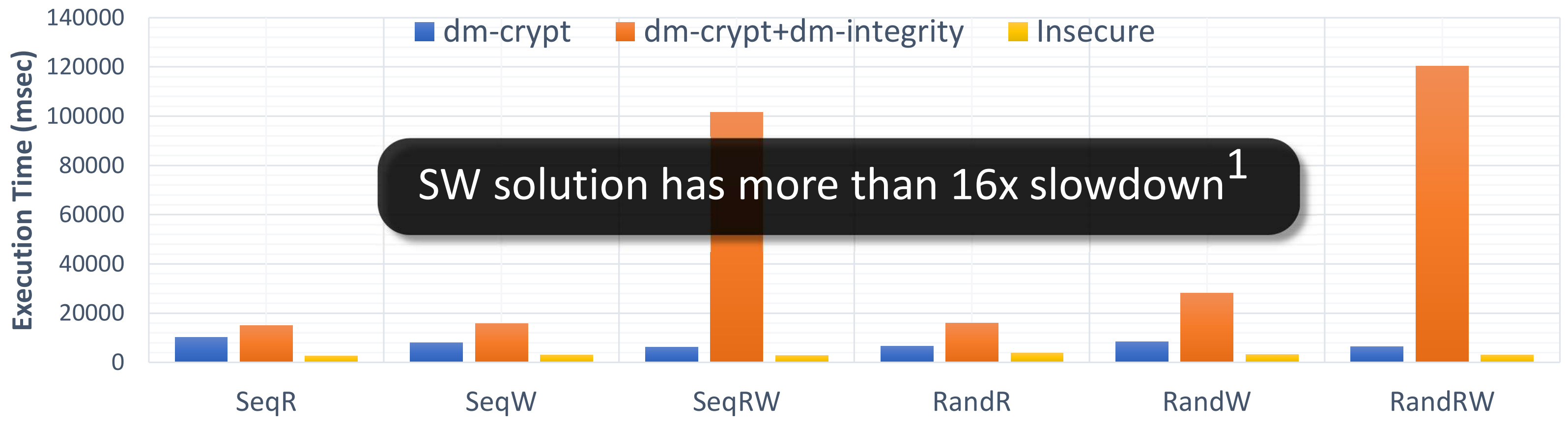
- Pitfalls of existing solutions:
  - **Self-encryption disks:** encrypts data in the storage itself
    - Do not protect physical attacks (i.e., bus snooping)
  - **Software-based** disk encryption and integrity checking?



1. Performed on Intel i7-9700K with Samsung 970 (NVMe)

Need for Architectural Support for Storage Security

- Pitfalls of existing solutions:
  - **Self-encryption disks:** encrypts data in the storage itself
    - Do not protect physical attacks (i.e., bus snooping)
  - **Software-based** disk encryption and integrity checking?

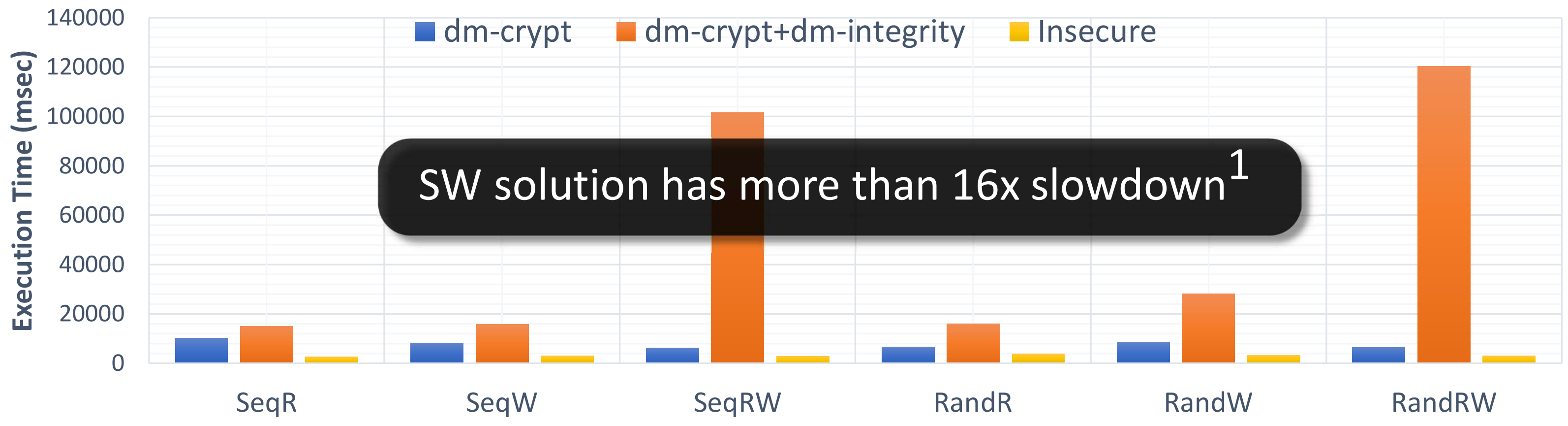


SW solution has more than 16x slowdown[1]

1. Performed on Intel i7-9700K with Samsung 970 (NVMe)

Need for Architectural Support for Storage Security

- Pitfalls of existing solutions:
  - **Self-encryption disks:** encrypts data in the storage itself
    - Do not protect physical attacks (i.e., bus snooping)
  - **Software-based** disk encryption and integrity checking?



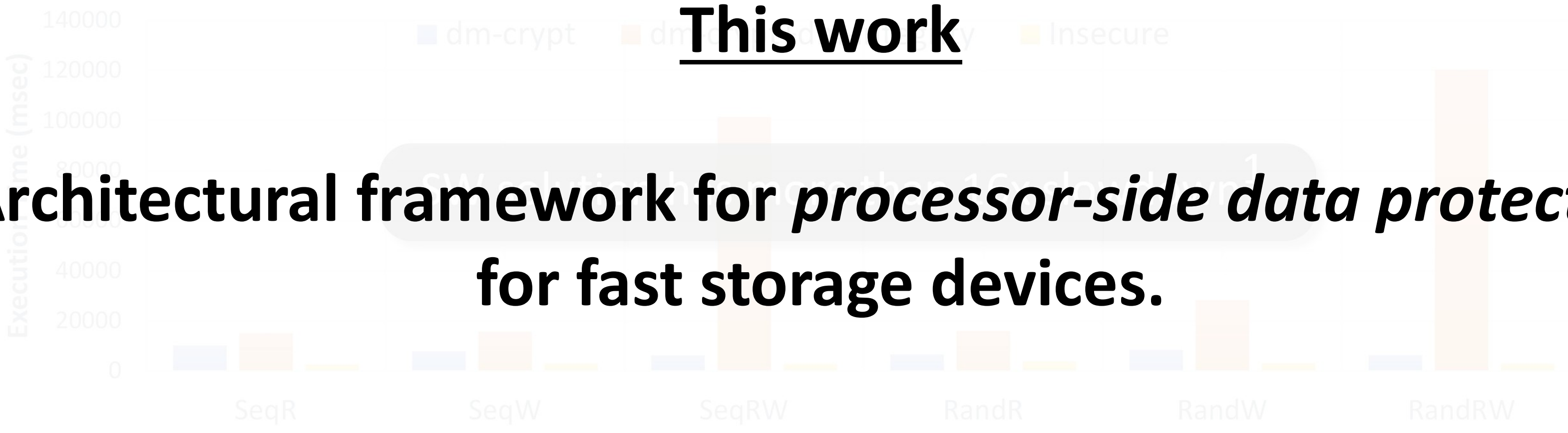
SW solution has more than 16x slowdown[1]

- Emerging ultra-fast storage devices will further increase the bottleneck
  - Microseconds range access latency (e.g., Intel Optane SSD)

1. Performed on Intel i7-9700K with Samsung 970 (NVMe)

Need for Architectural Support for Storage Security

- Pitfalls of existing solutions:
  - **Self-encryption disks:** encrypts data in the storage itself
    - Do not protect physical attacks (i.e., bus snooping)
  - **Software-based** disk encryption and integrity checking?



SW solution has more than 16x slowdown[1]

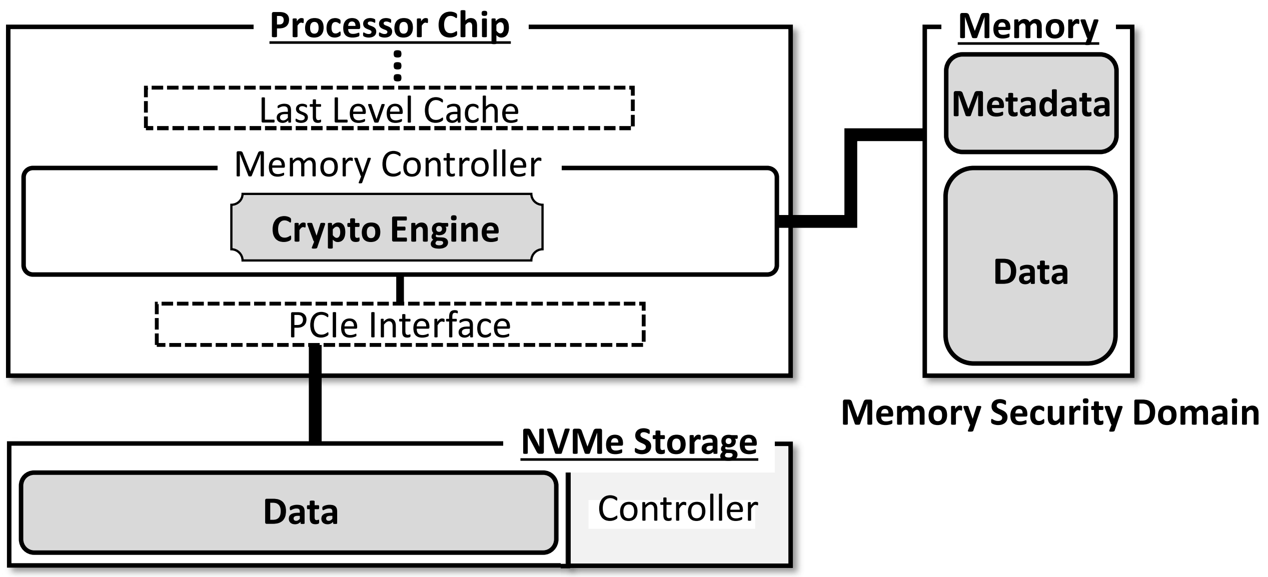
<u>This work</u>

Architectural framework for *processor-side data protection* for fast storage devices.

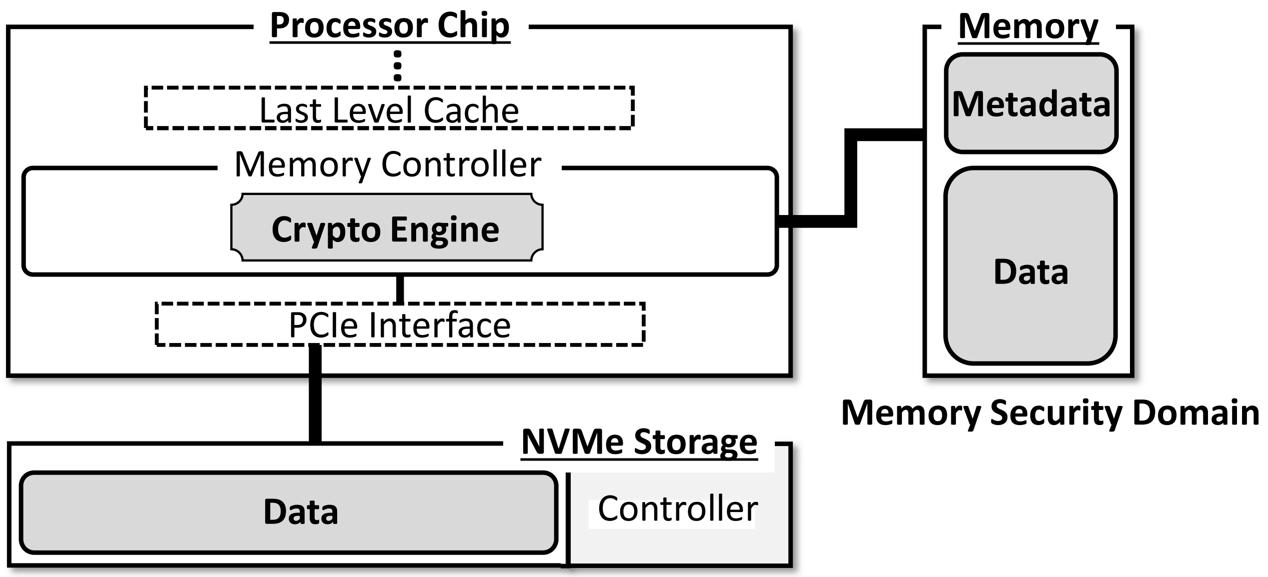
- Emerging ultra-fast storage devices will further increase the bottleneck
  - Microseconds range access latency (e.g., Intel Optane SSD)

1. Performed on Intel i7-9700K with Samsung 970 (NVMe)

Design Objectives

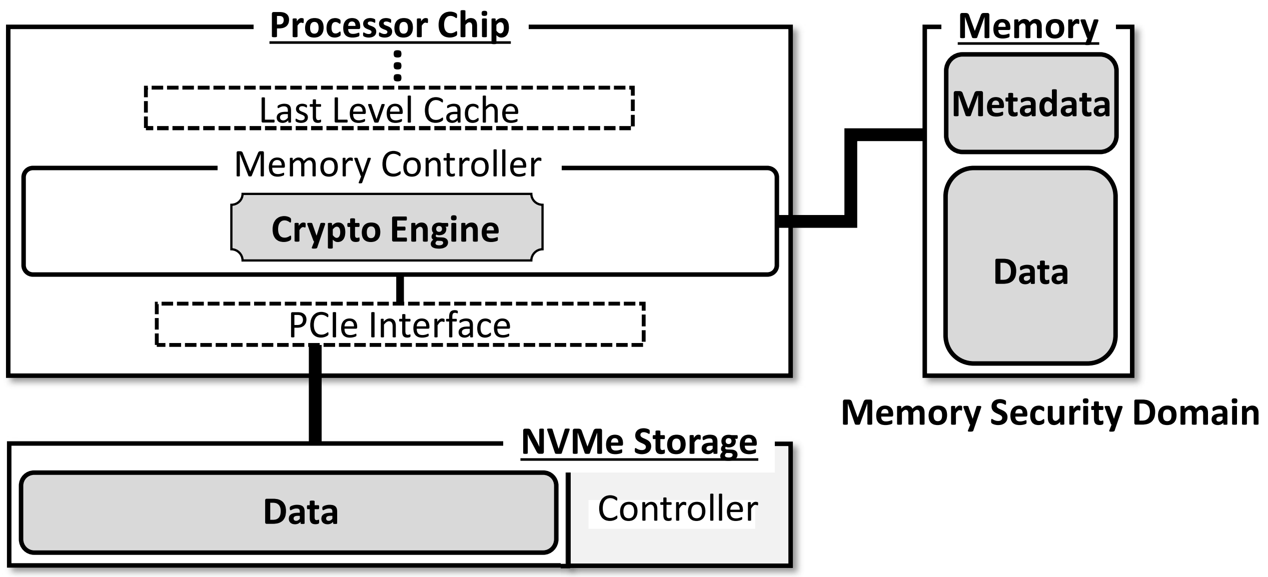


Design Objectives



Objective 1: processor-side support (CPU as root of trust)

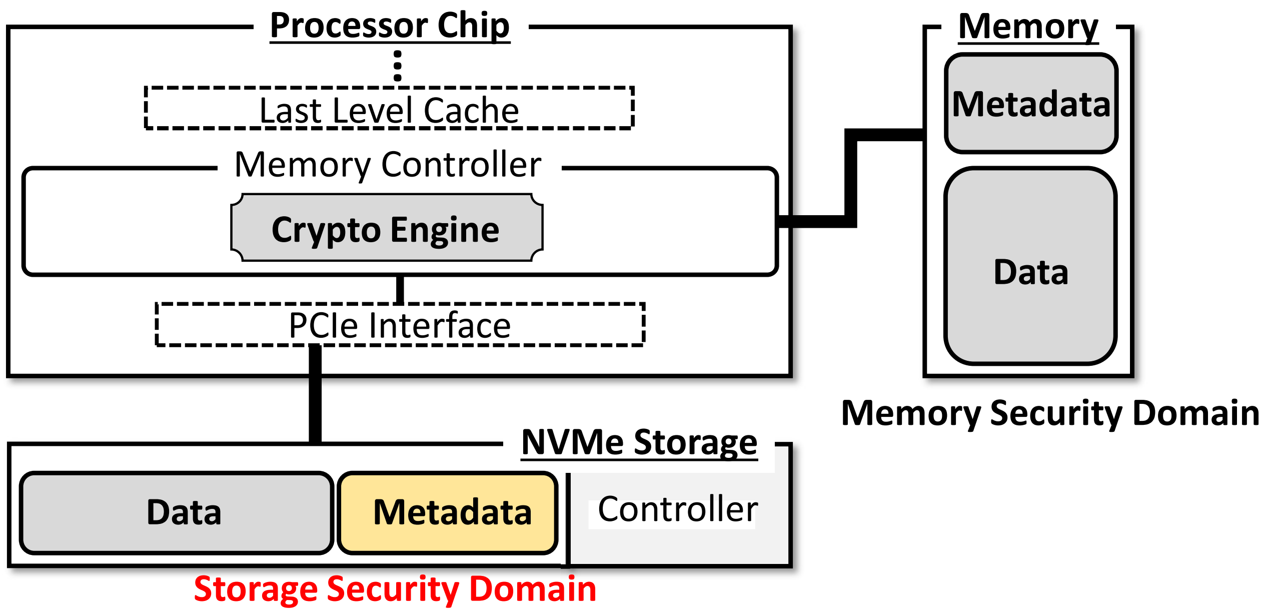
Design Objectives



Objective 1: processor-side support (CPU as root of trust)

Objective 2: transparent to storage devices and NVMe protocols

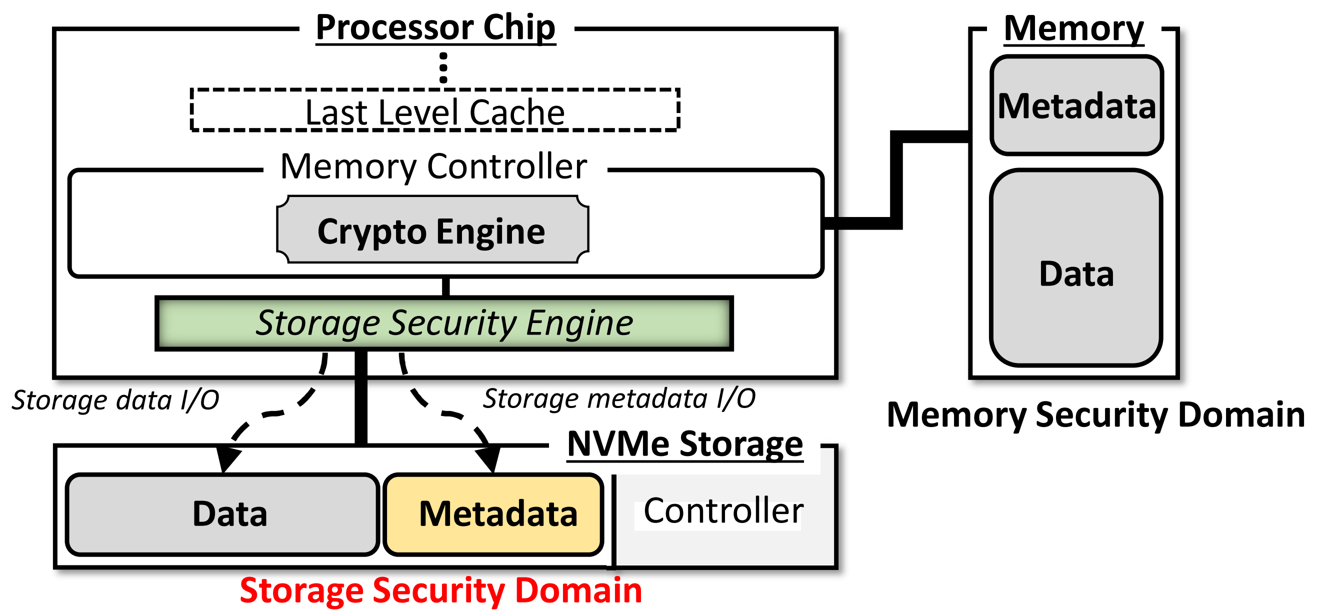
Design Objectives



Objective 1: processor-side support (CPU as root of trust)

Objective 2: transparent to storage devices and NVMe protocols

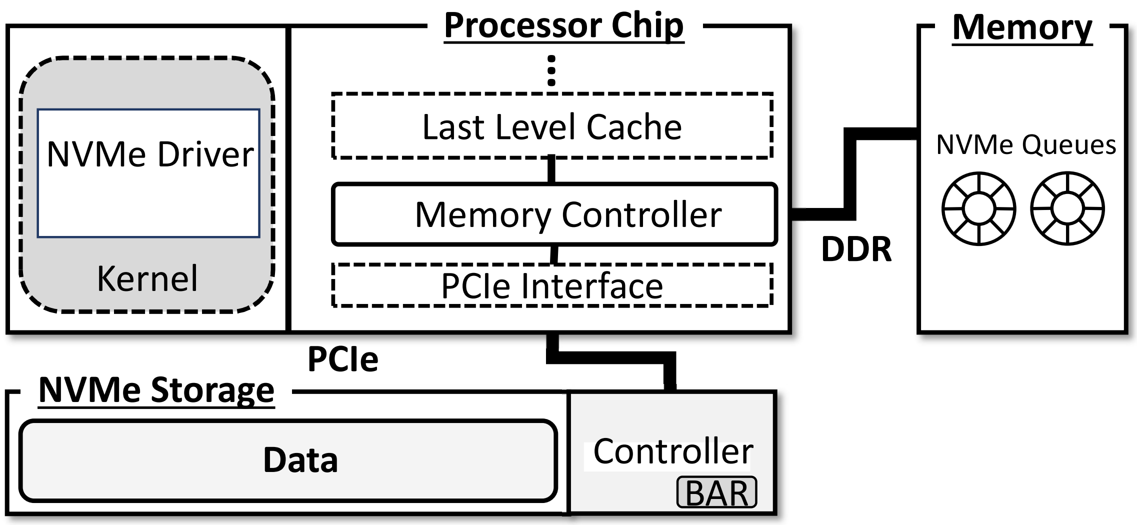
Design Objectives



Objective 1: processor-side support (CPU as root of trust)

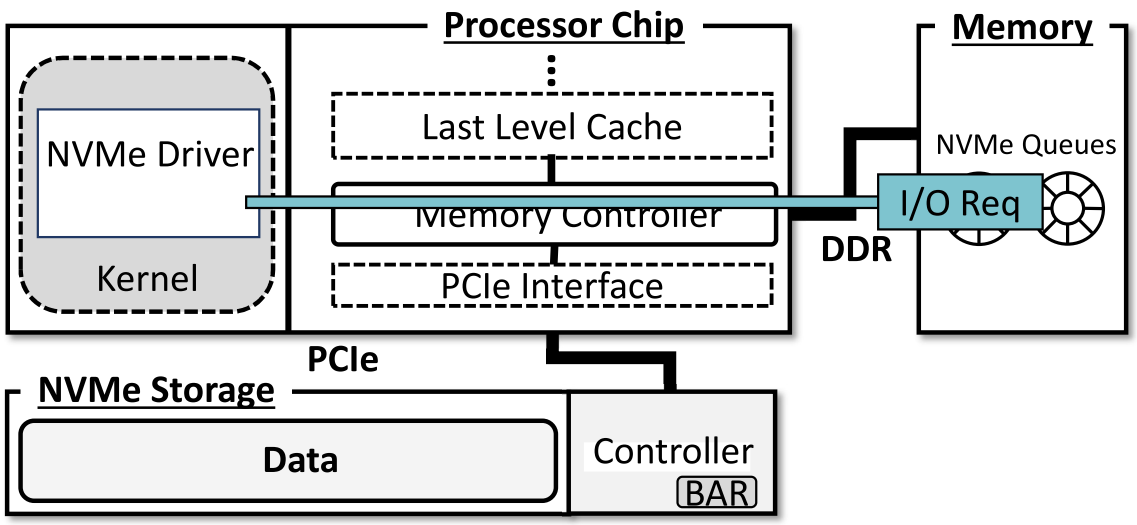
Objective 2: transparent to storage devices and NVMe protocols

Challenges



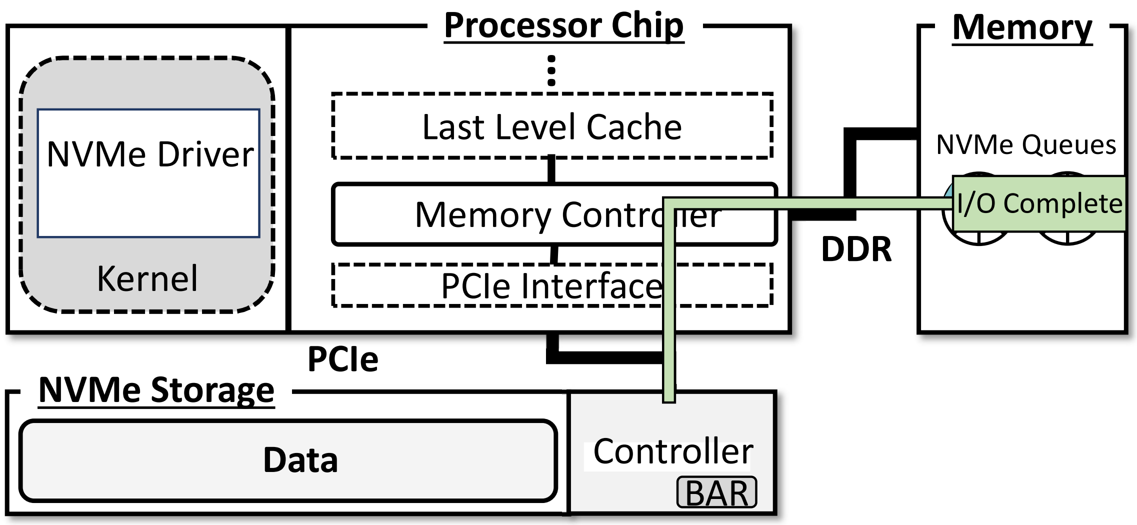
C1: Intricate SW/HW interactions → excessive software intervention can be expensive

Challenges



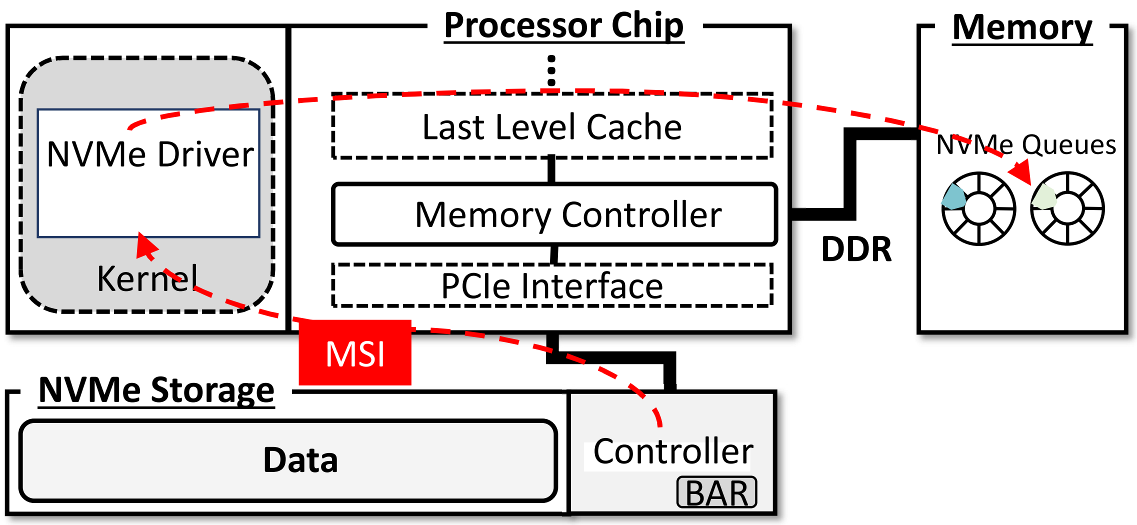
C1: Intricate SW/HW interactions → excessive software intervention can be expensive

Challenges



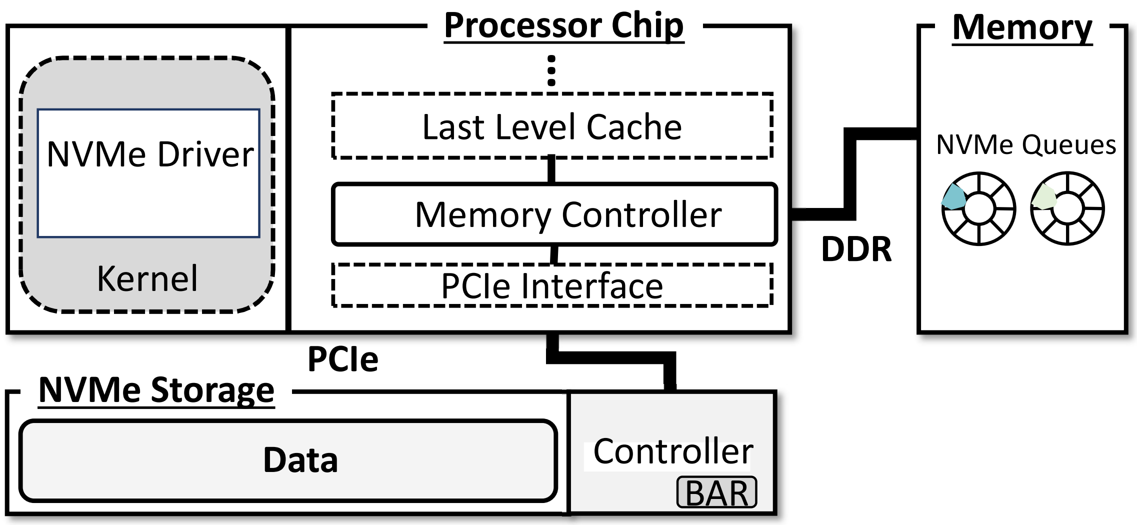
C1: Intricate SW/HW interactions → excessive software intervention can be expensive

Challenges



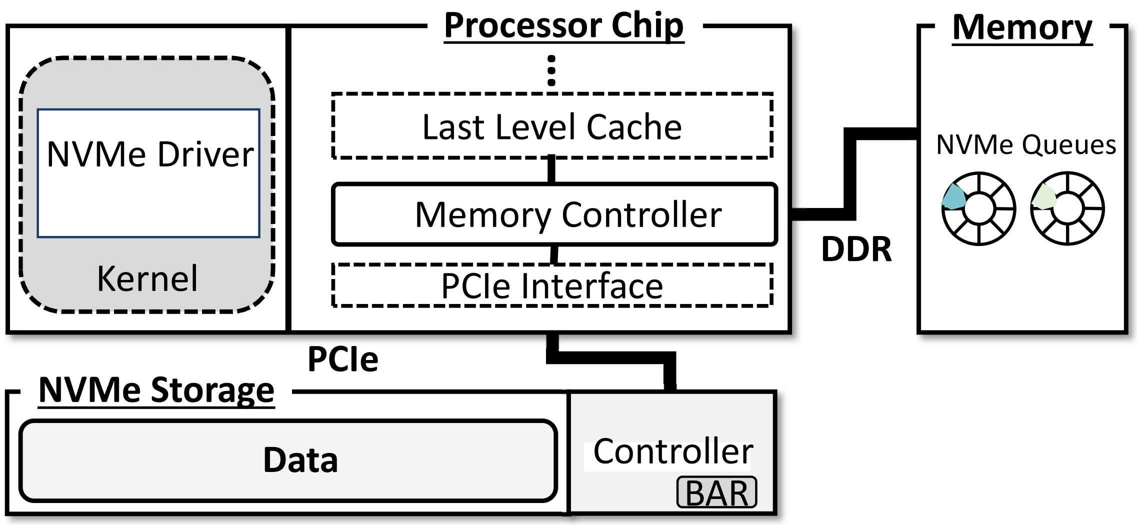
C1: Intricate SW/HW interactions → excessive software intervention can be expensive

Challenges



C1: Intricate SW/HW interactions → excessive software intervention can be expensive

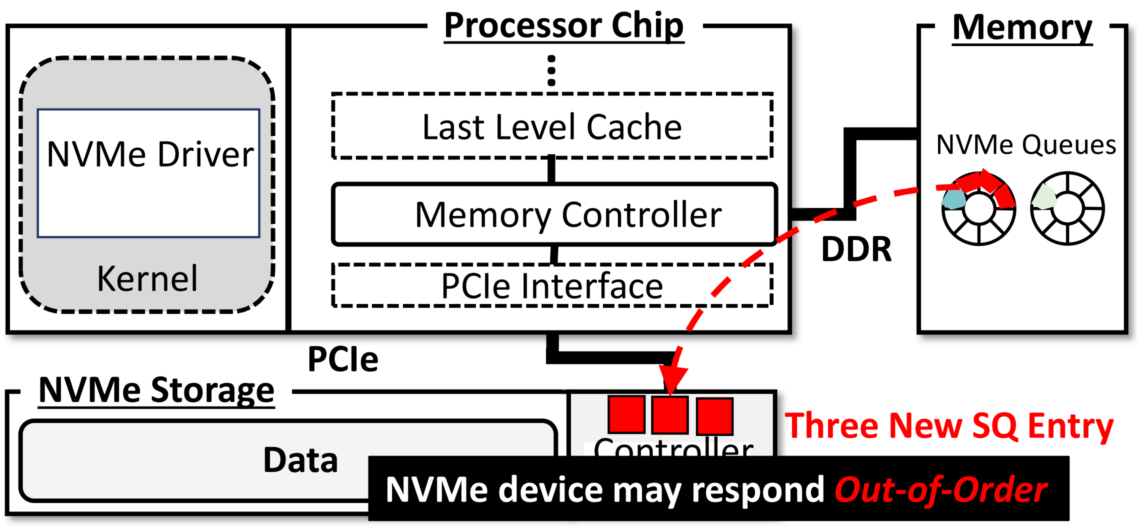
Challenges



C1: Intricate SW/HW interactions → excessive software intervention can be expensive

C2: Asynchronous control/data flow → requires hardware support to identify and map DMA requests

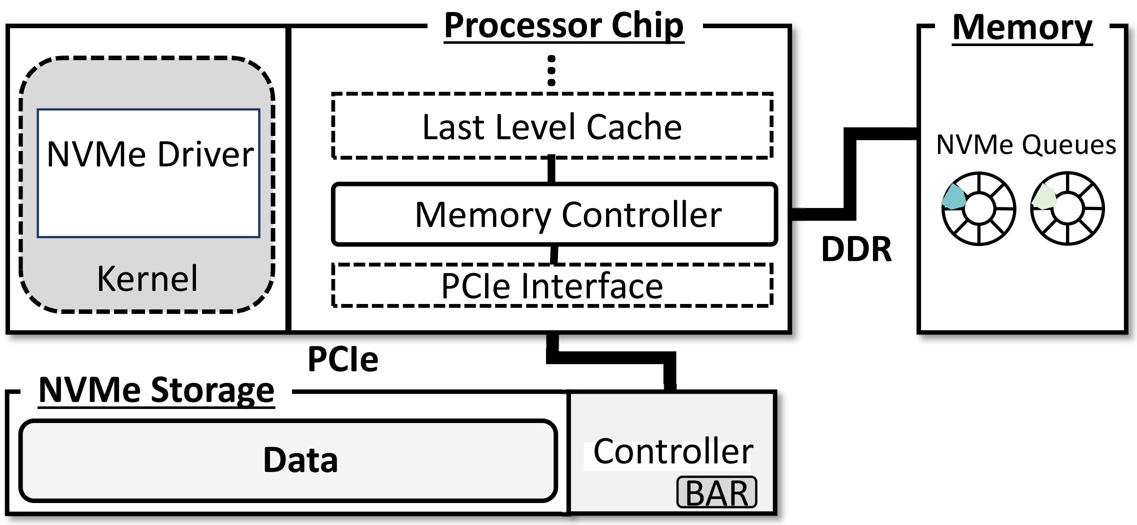
Challenges



C1: Intricate SW/HW interactions → excessive software intervention can be expensive

C2: Asynchronous control/data flow → requires hardware support to identify and map DMA requests

Challenges

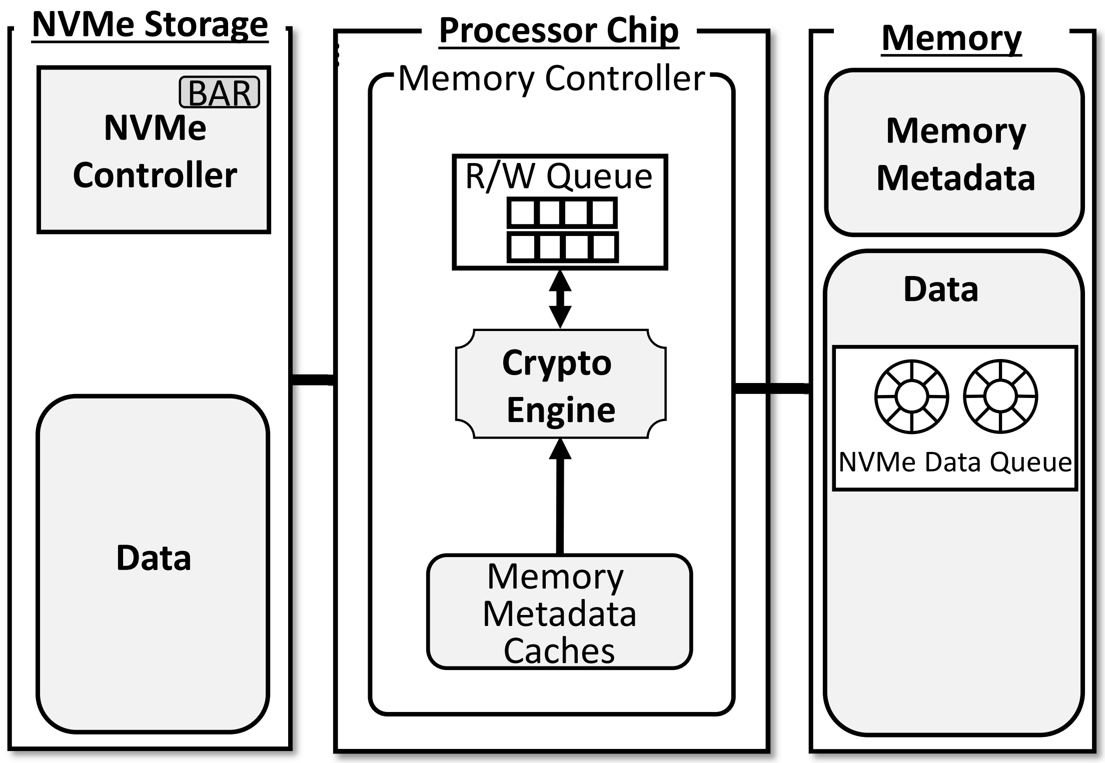


C1: Intricate SW/HW interactions → excessive software intervention can be expensive

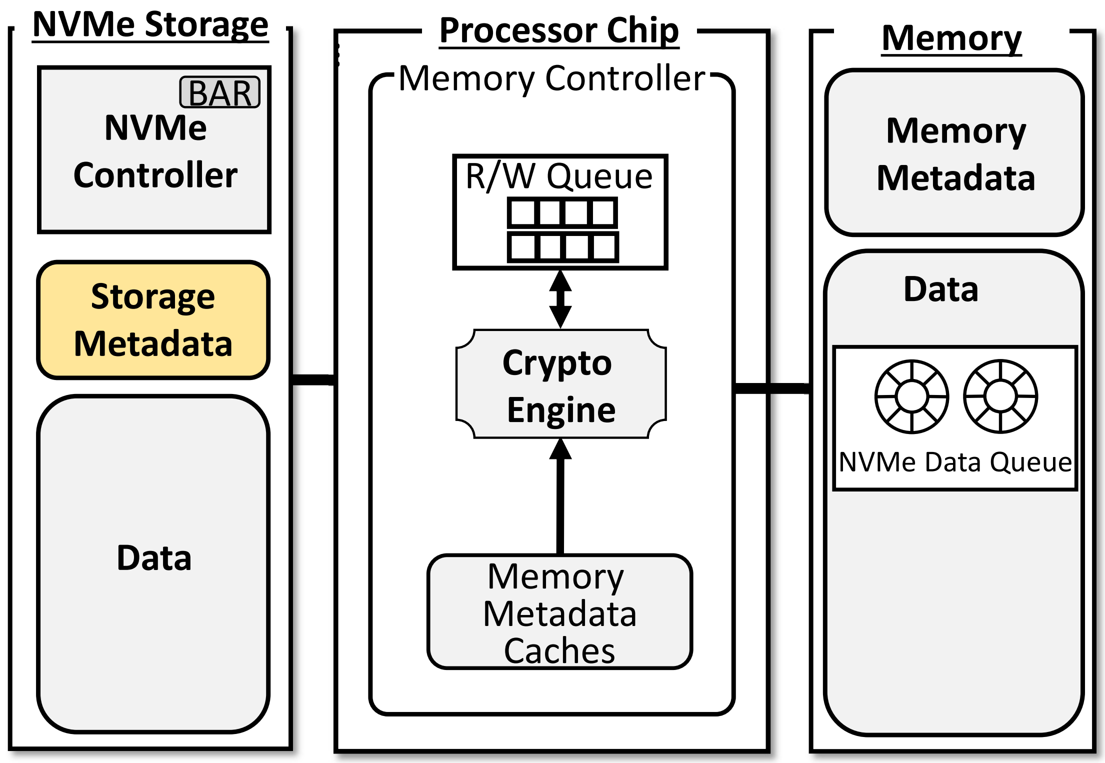
C2: Asynchronous control/data flow → requires hardware support to identify and map DMA requests

C3: Metadata I/O overheads → efficient metadata management tailored for storage I/O characteristics

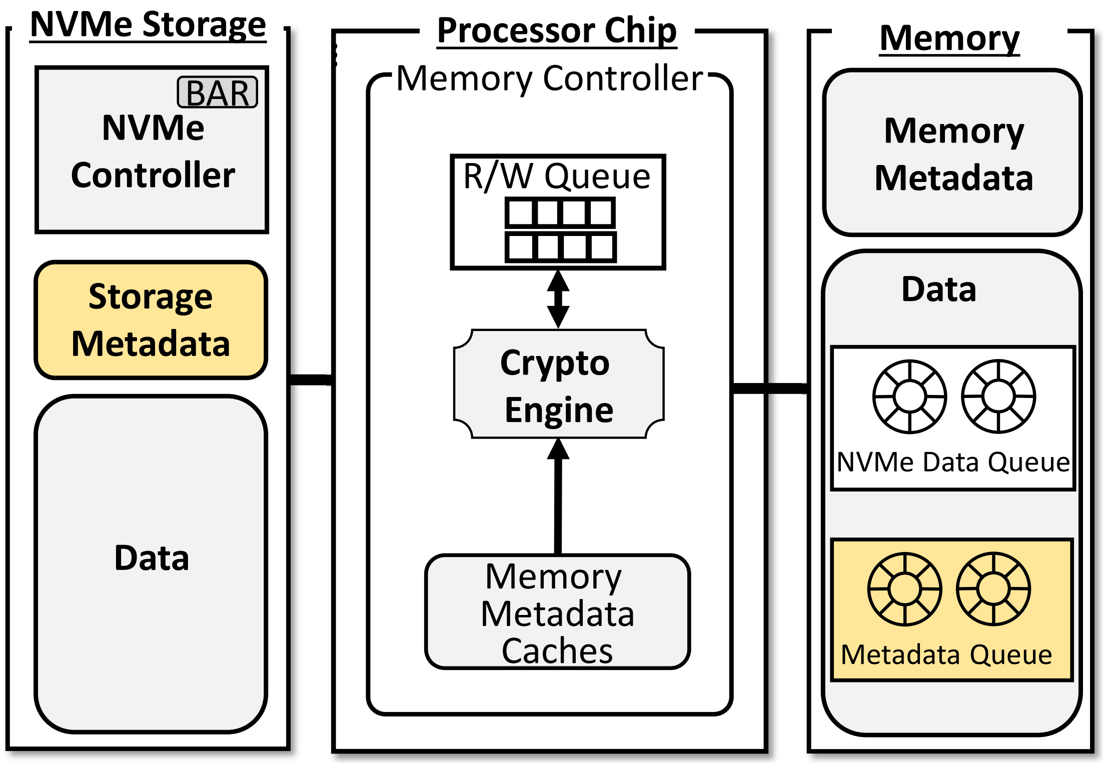
Basic D-Shield Design



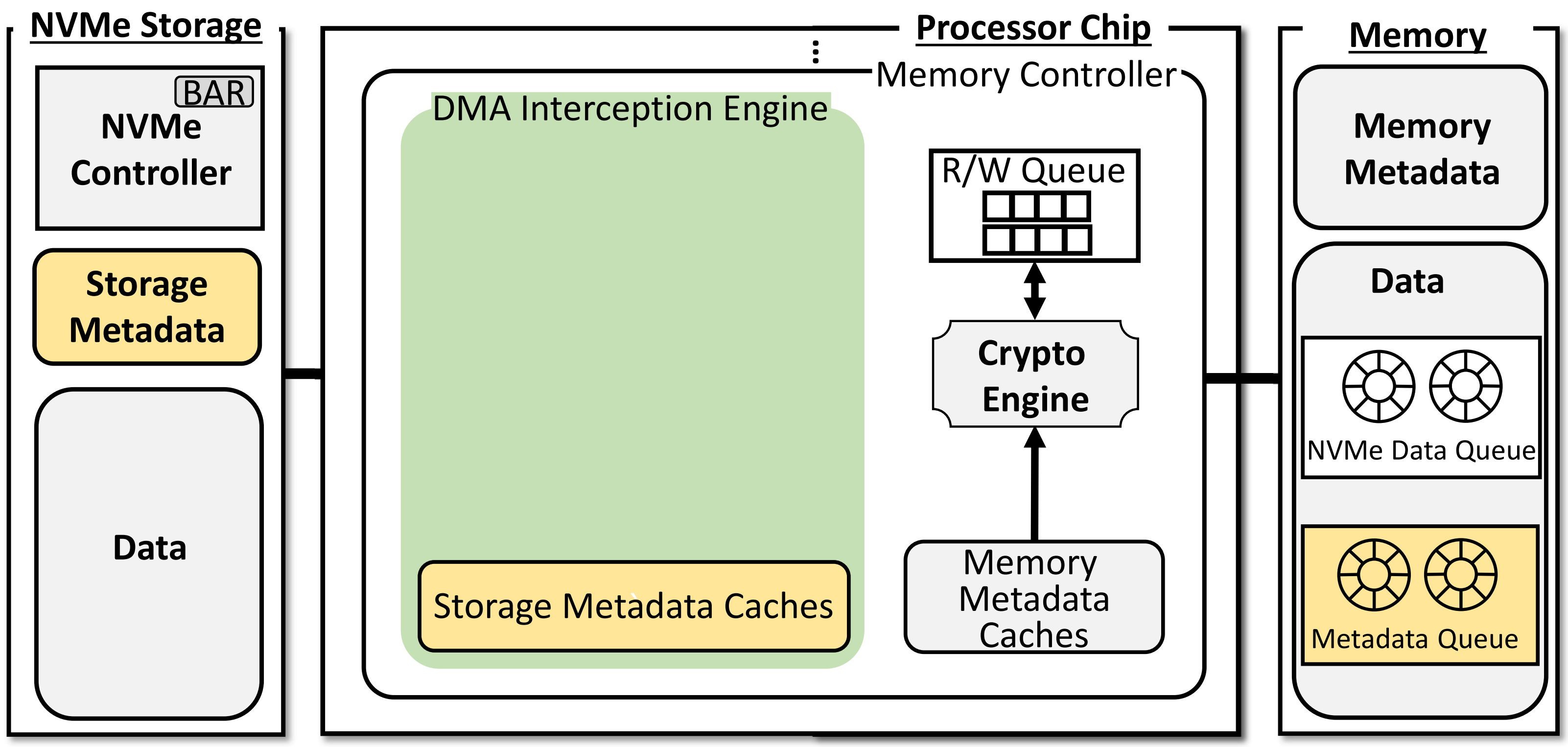
Basic D-Shield Design



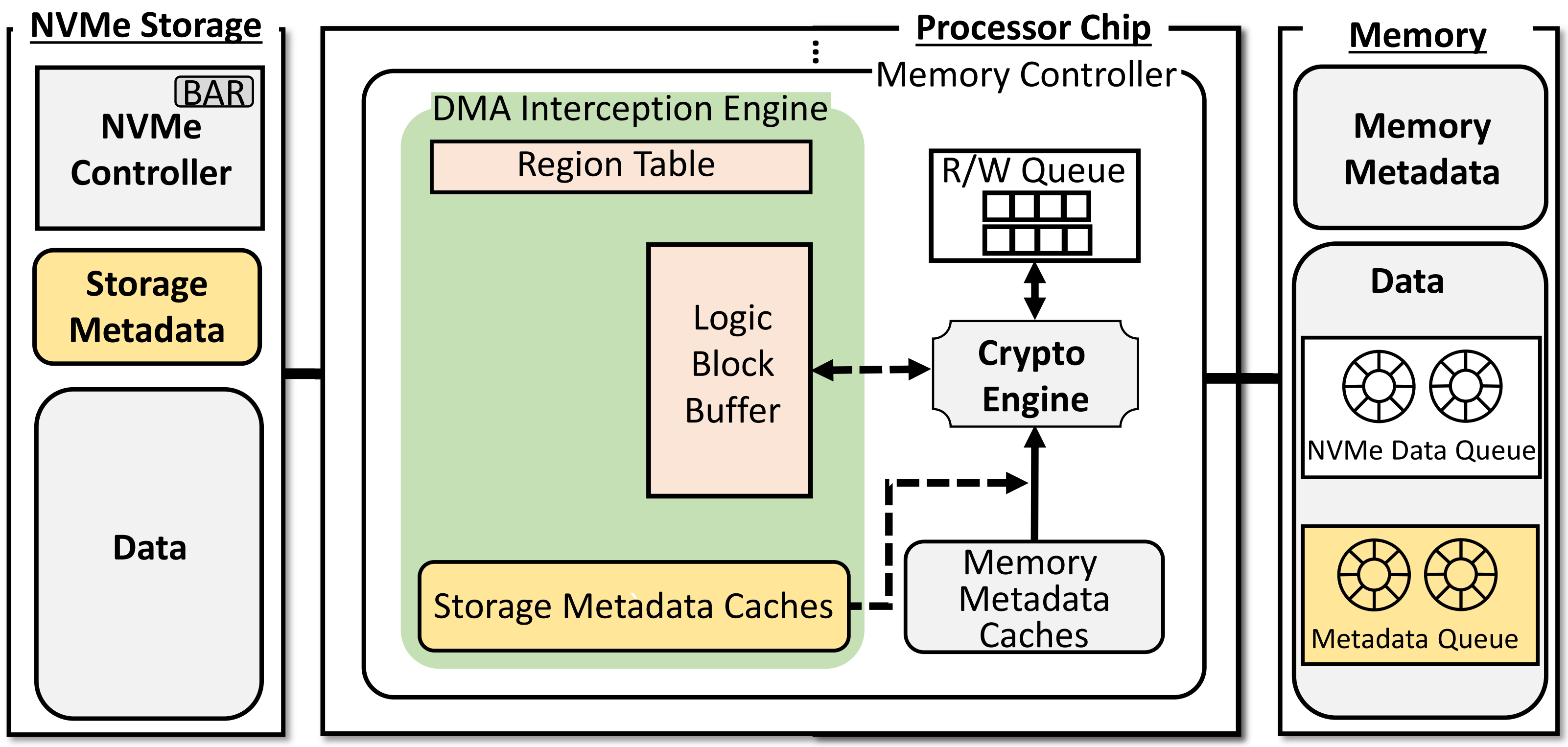
Basic D-Shield Design



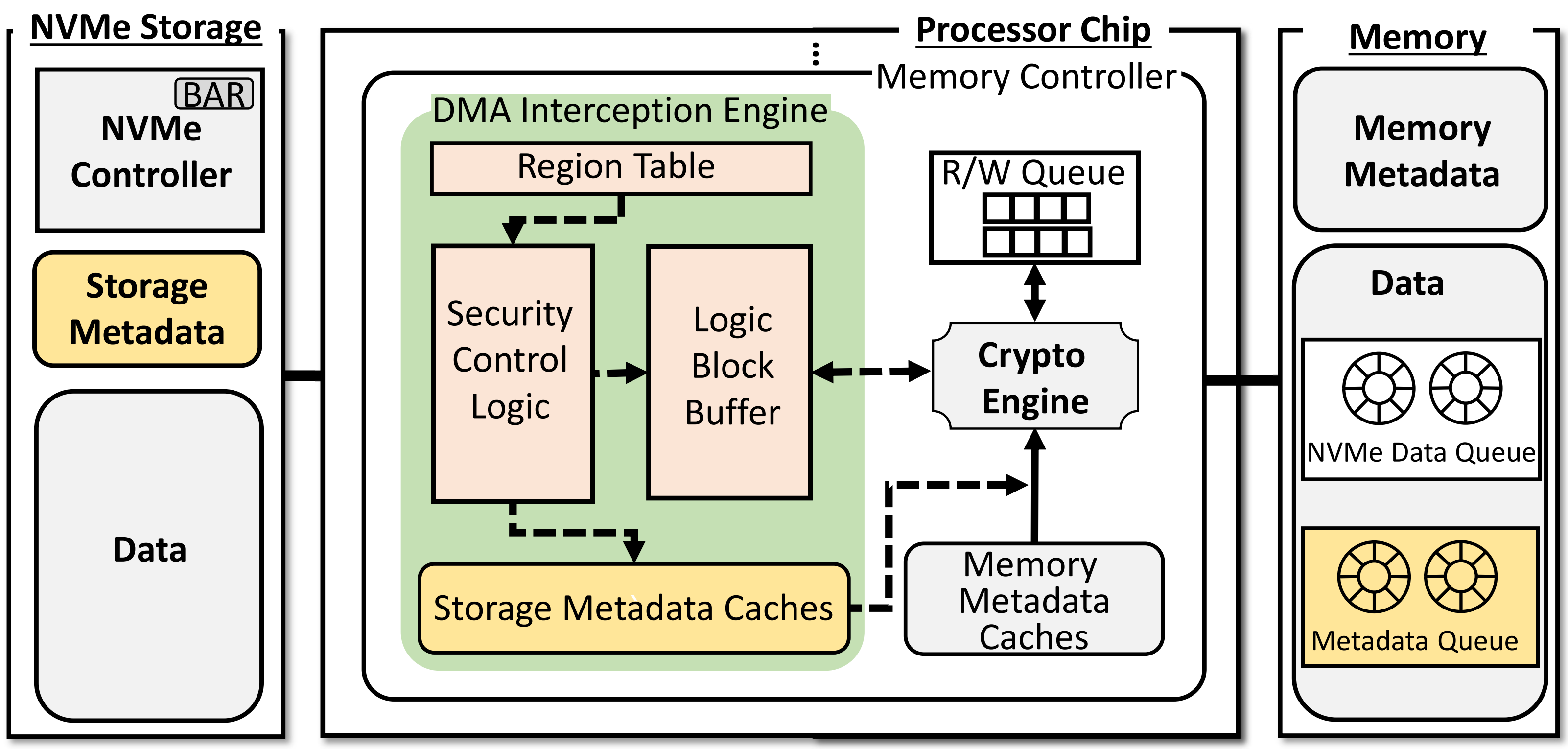
Basic D-Shield Design



Basic D-Shield Design

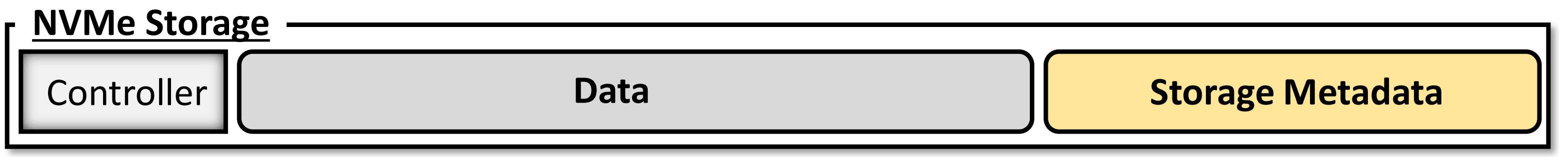


Basic D-Shield Design



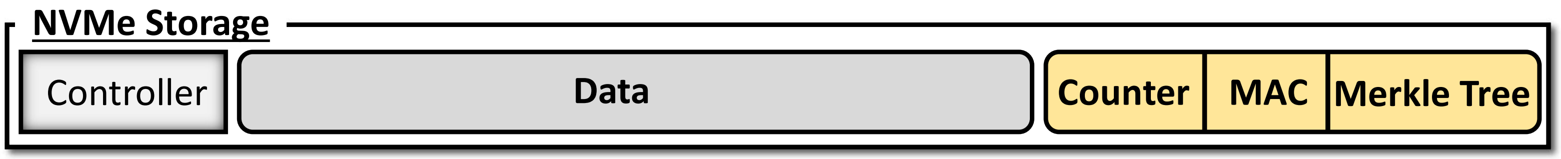
NVMe-optimized Security Metadata

- Storage metadata are stored separately in NVMe disks



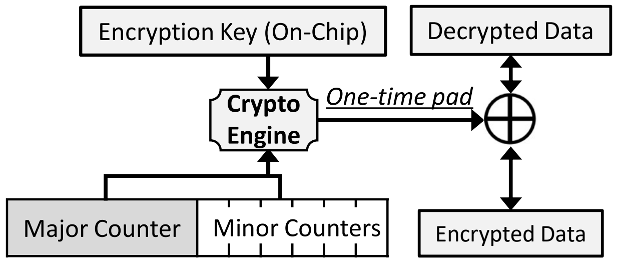
NVMe-optimized Security Metadata

- Storage metadata are stored separately in NVMe disks
- Three types of security metadata → similar to *Secure Memory*

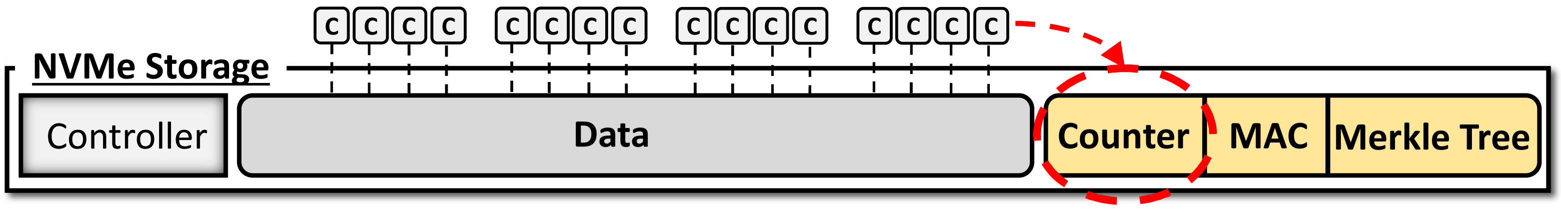


NVMe-optimized Security Metadata

- Storage metadata are stored separately in NVMe disks
- Three types of security metadata → similar to *Secure Memory*

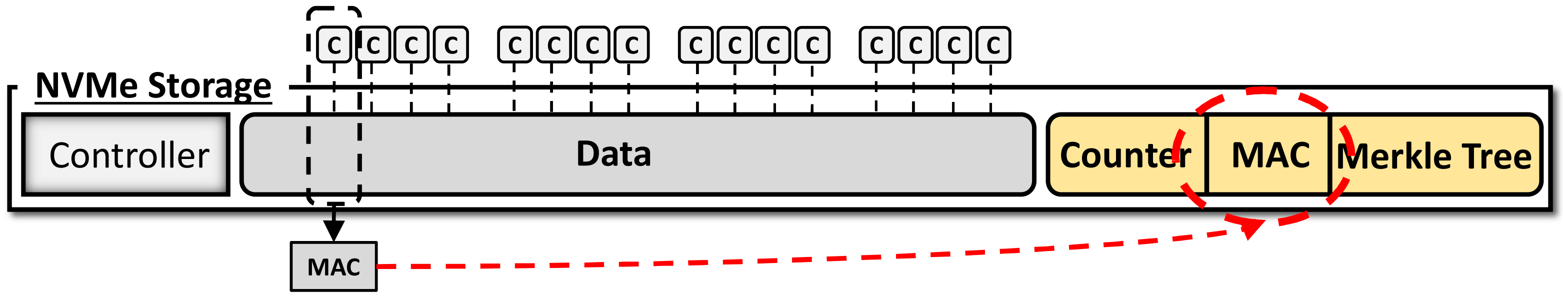


Counter-mode Encryption



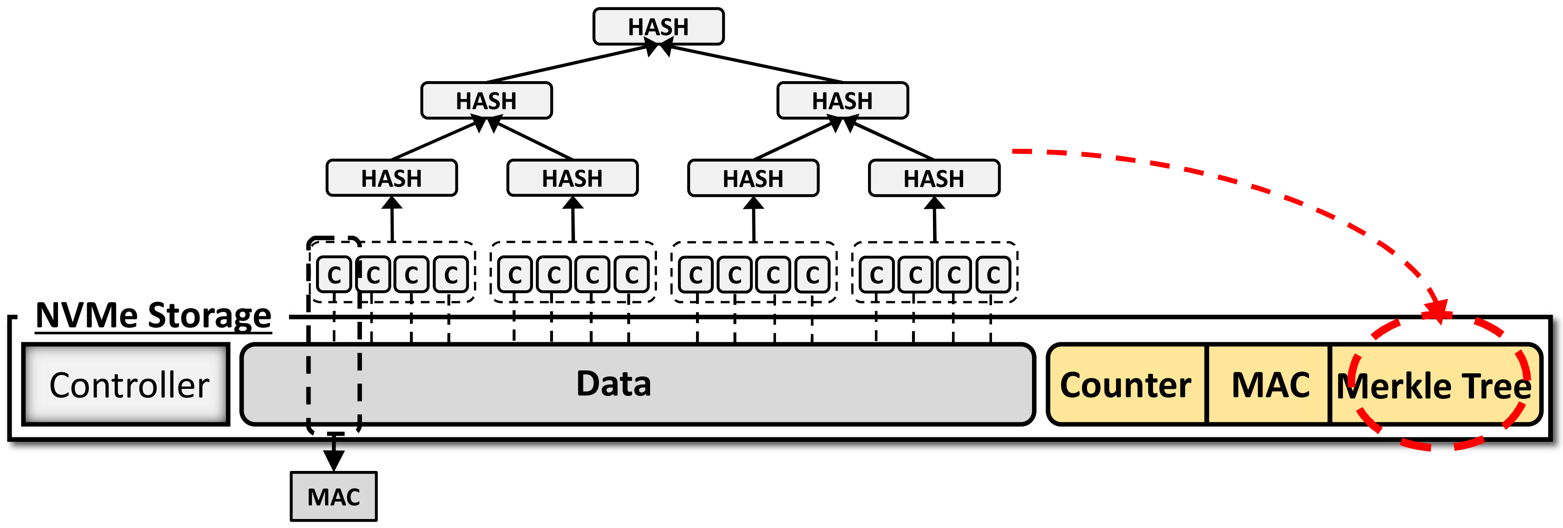
NVMe-optimized Security Metadata

- Storage metadata are stored separately in NVMe disks
- Three types of security metadata → similar to *Secure Memory*



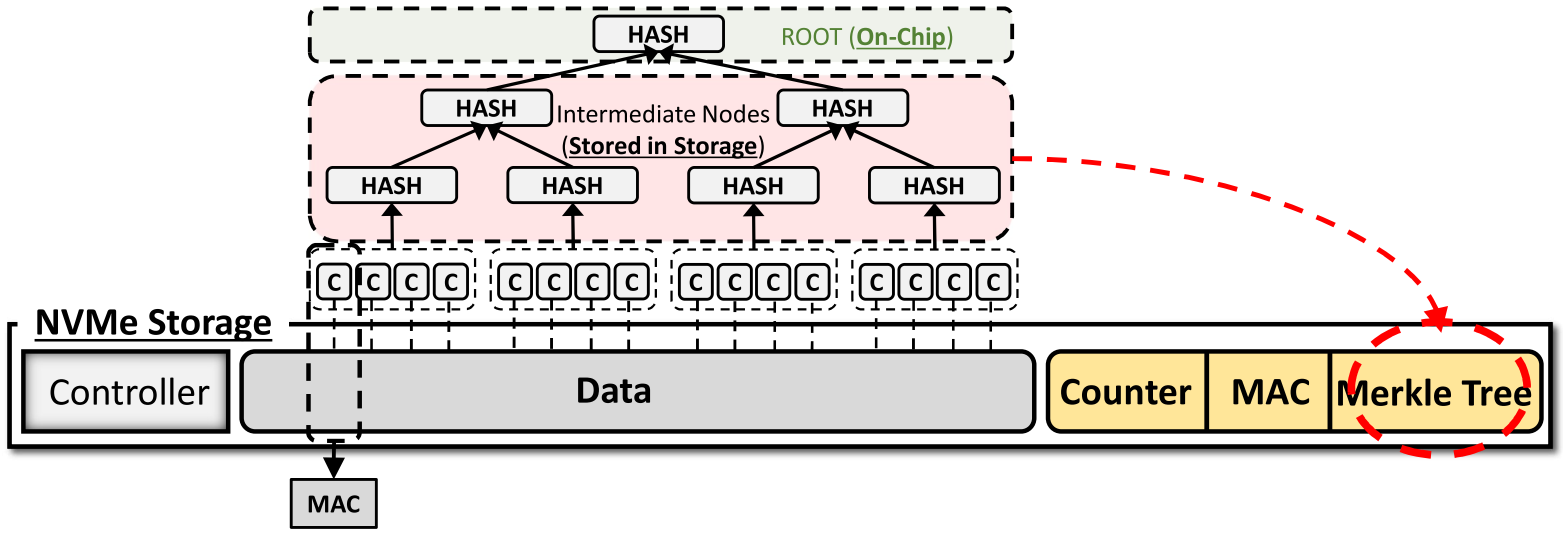
NVMe-optimized Security Metadata

- Storage metadata are stored separately in NVMe disks
- Three types of security metadata → similar to *Secure Memory*



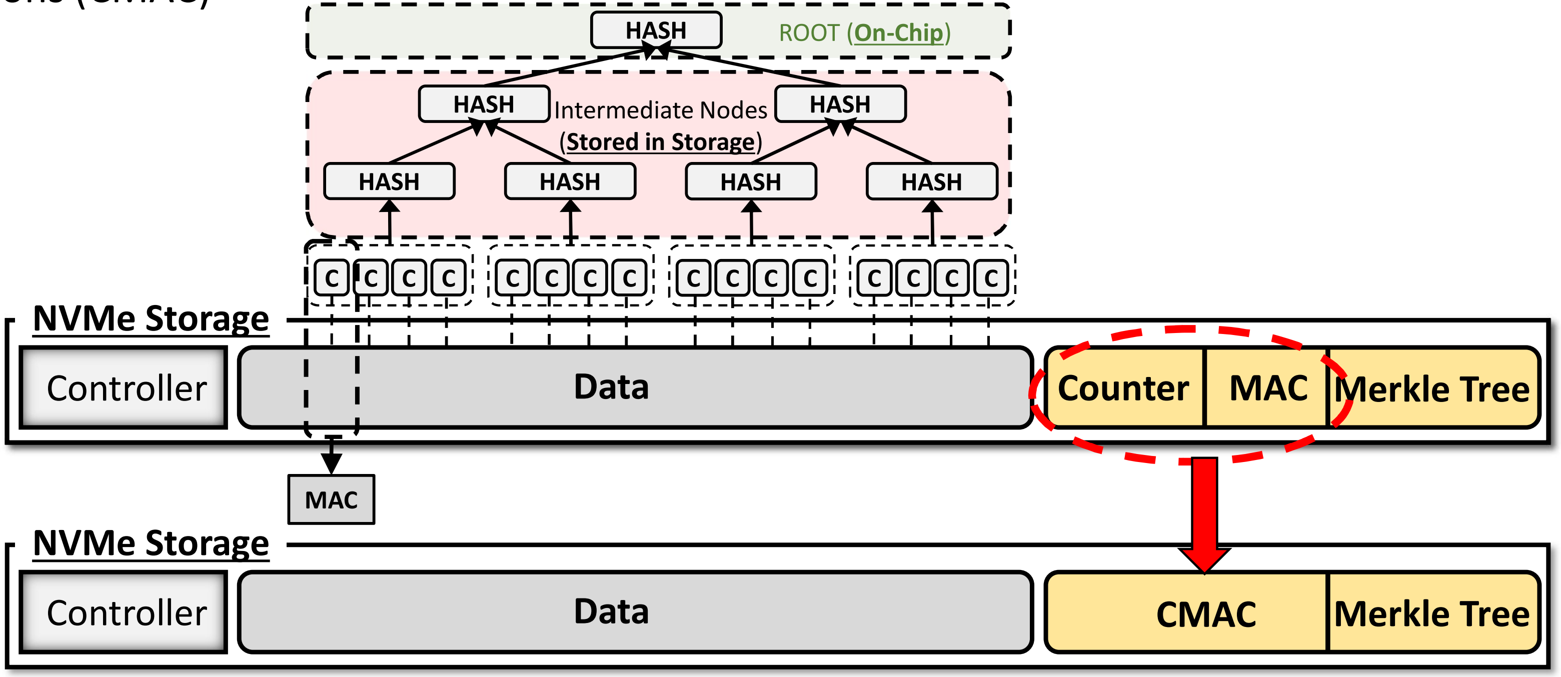
NVMe-optimized Security Metadata

- Storage metadata are stored separately in NVMe disks
- Three types of security metadata → similar to *Secure Memory*



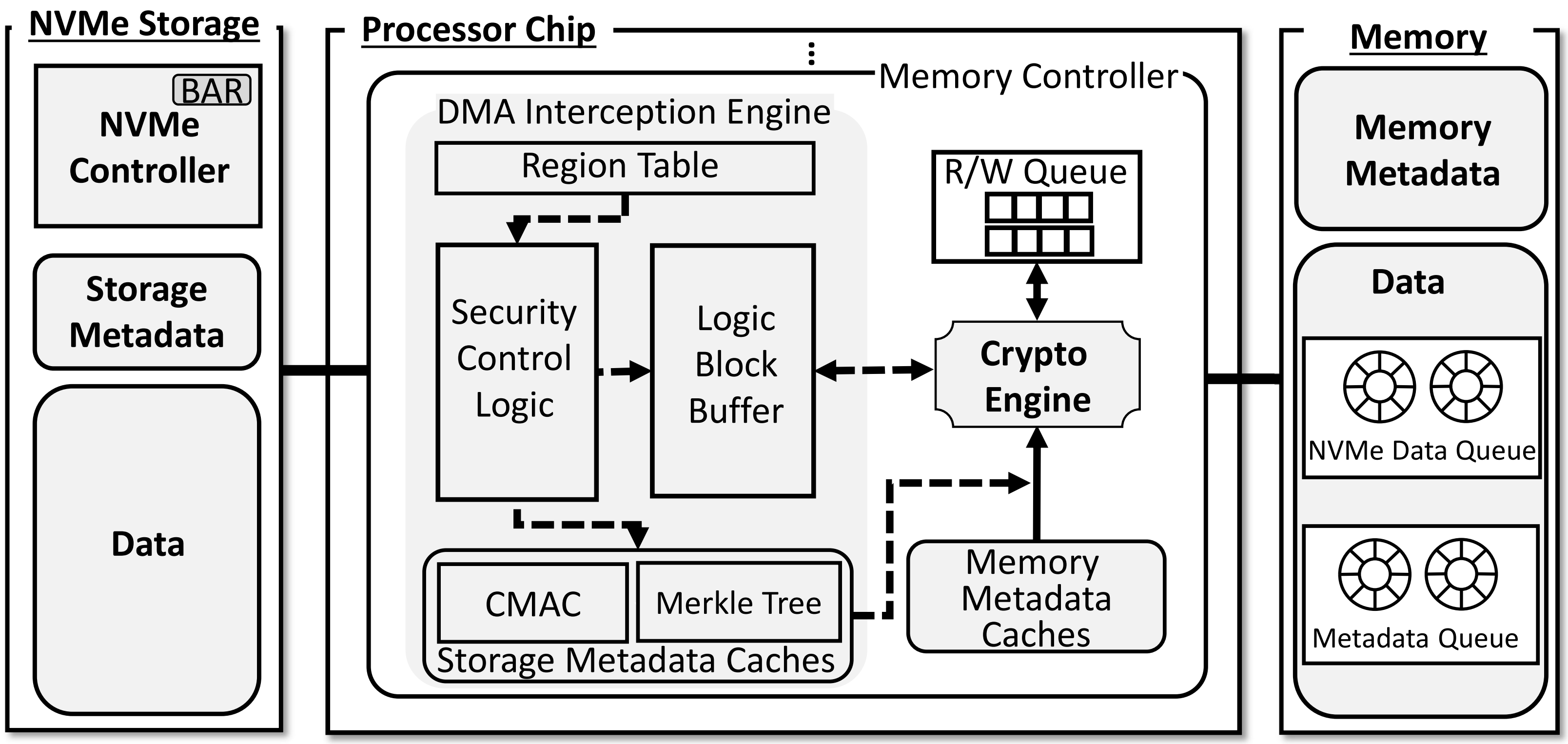
NVMe-optimized Security Metadata

- Storage metadata are stored separately in NVMe disks
- Three types of security metadata → similar to *Secure Memory*
- Counter and MAC are required for each data access → store them together to reduce metadata I/O operations (CMAC)

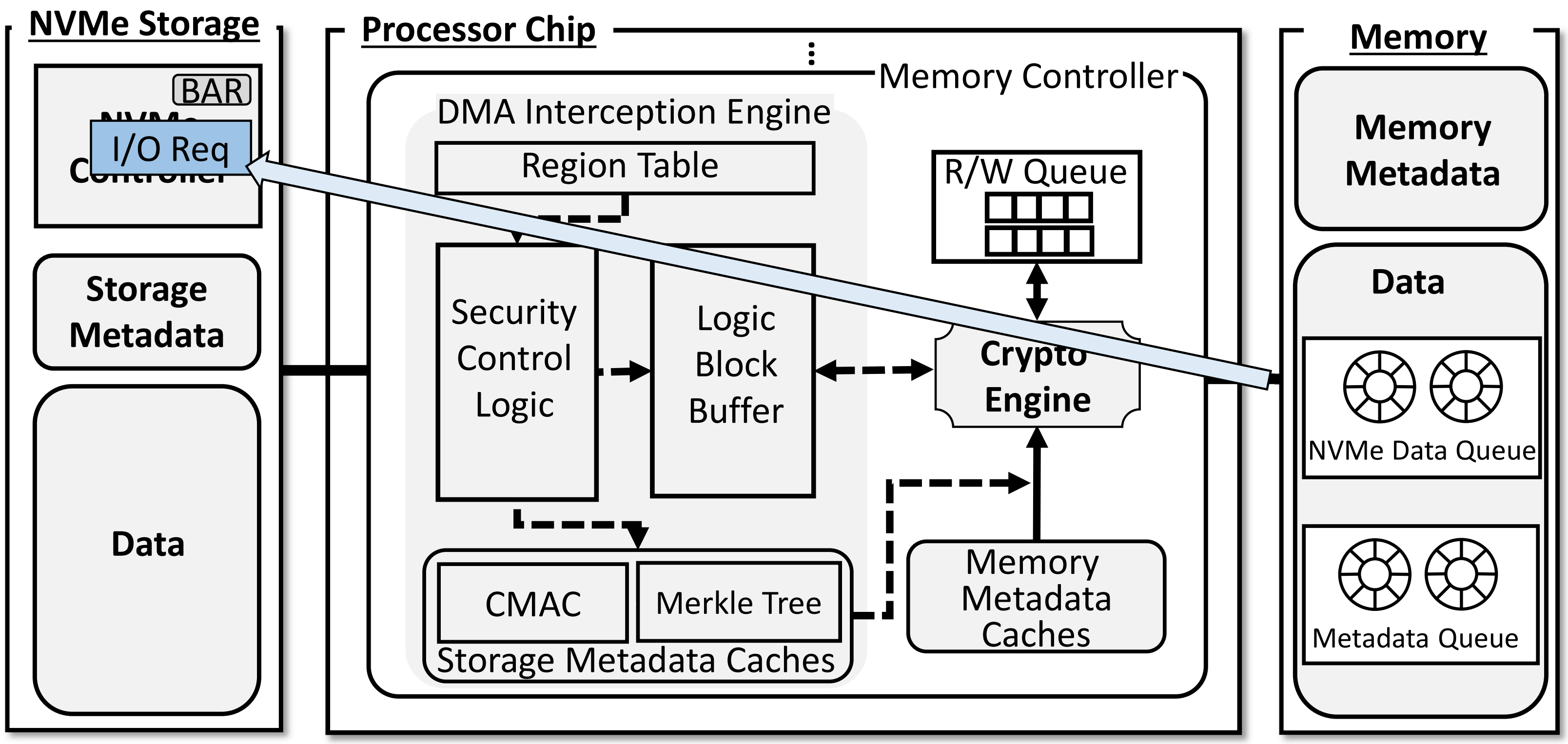


UCF

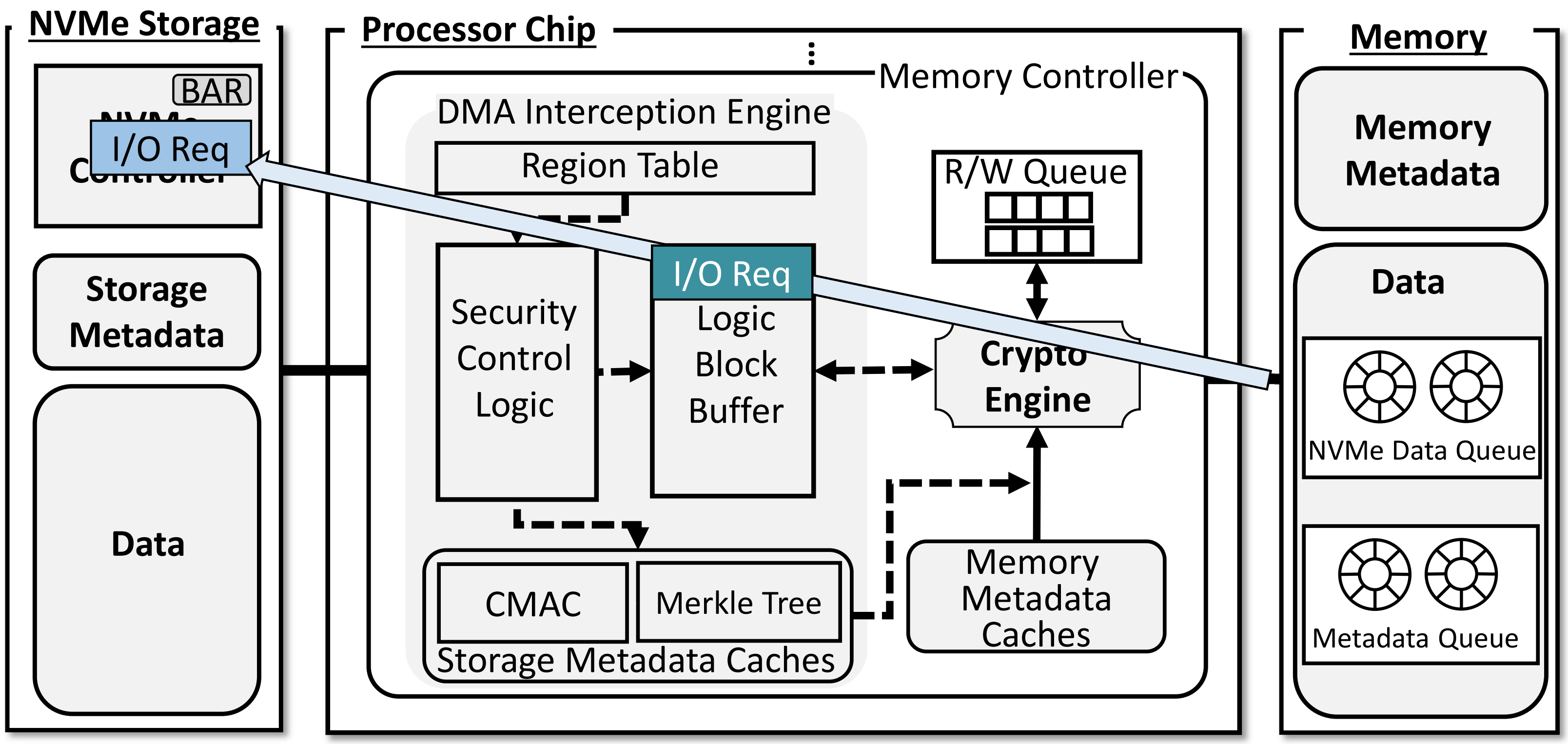
D-Shield Operation (Read)



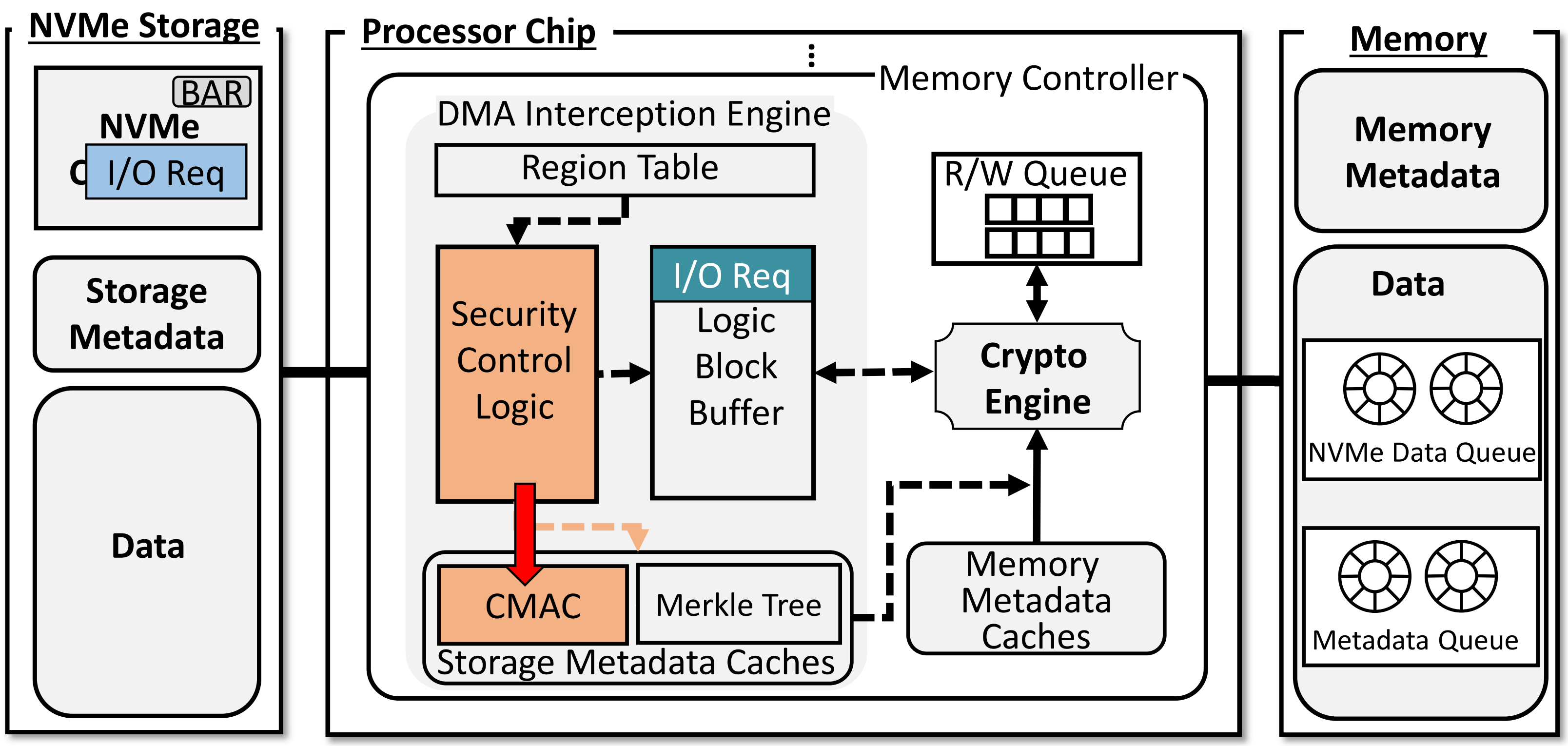
D-Shield Operation (Read)



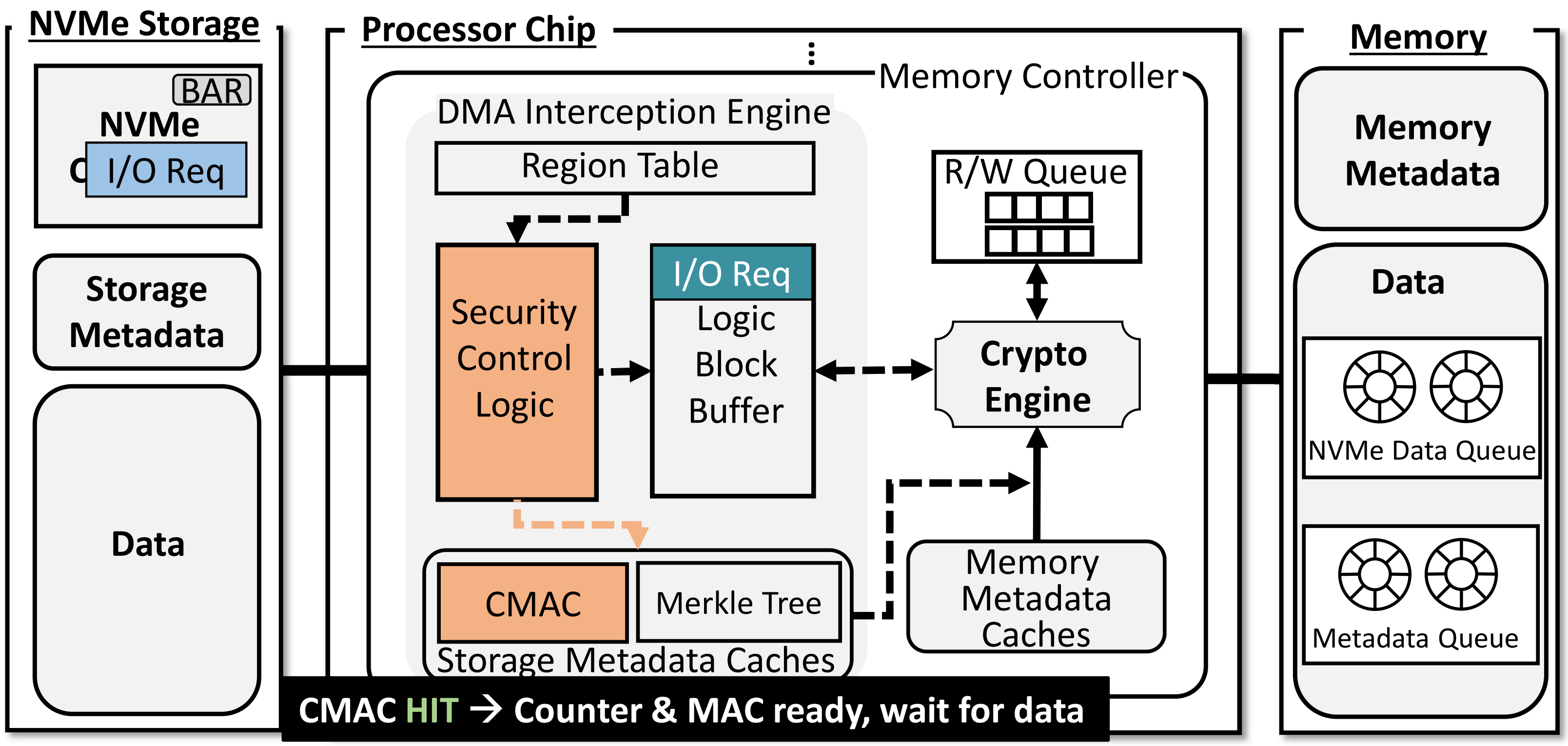
D-Shield Operation (Read)



D-Shield Operation (Read)

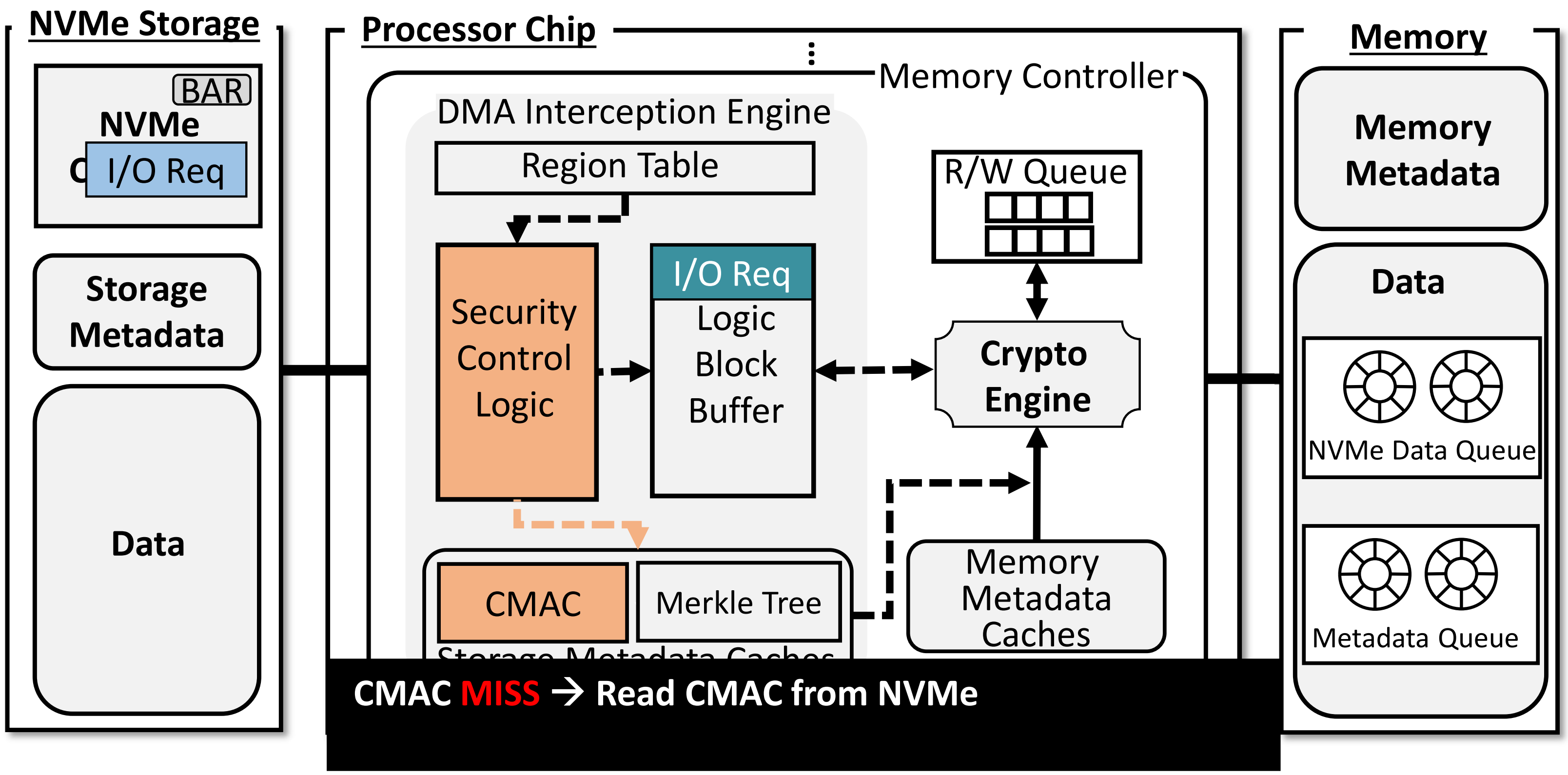


D-Shield Operation (Read)



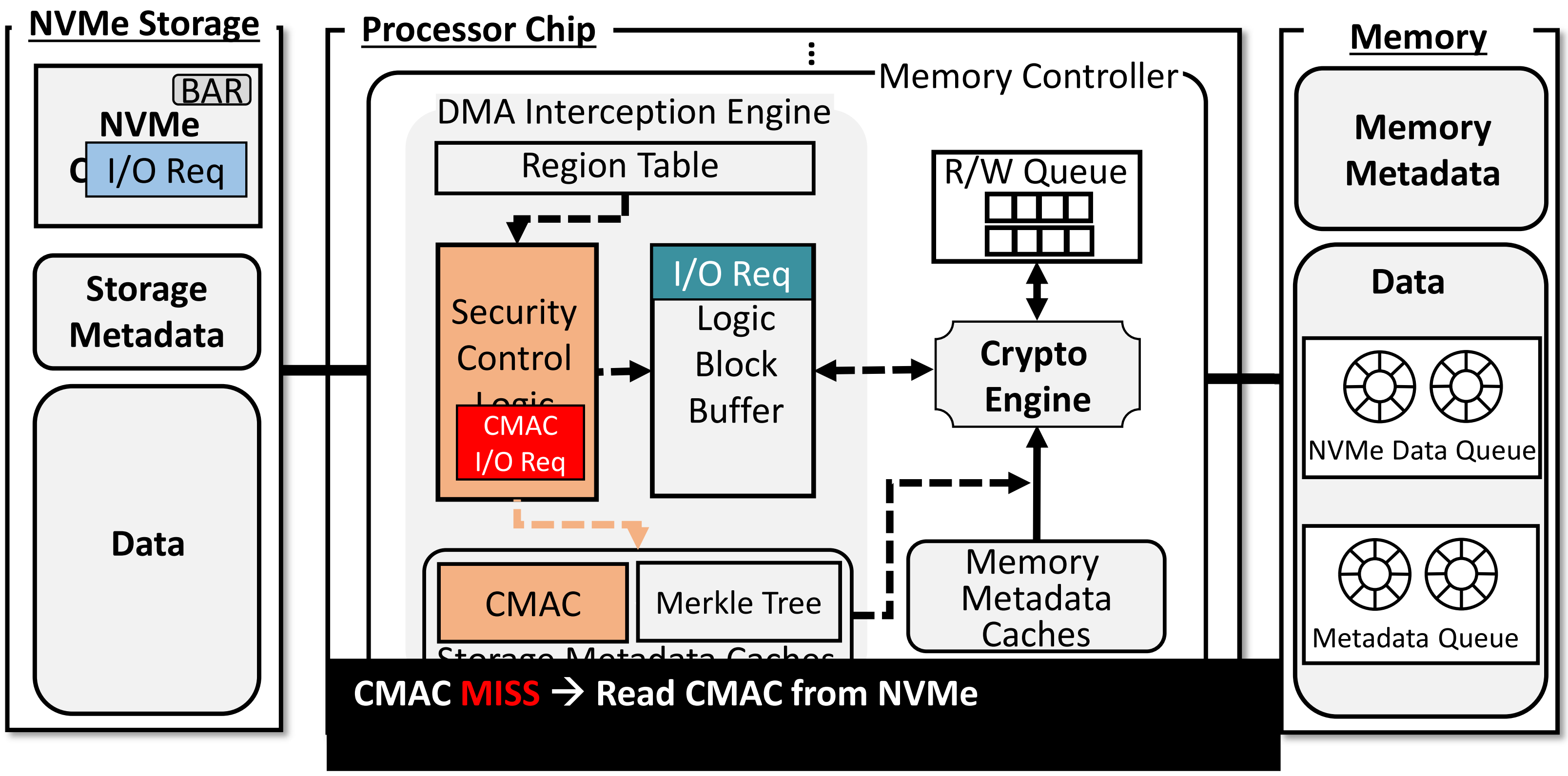
CMAC HIT → Counter & MAC ready, wait for data

D-Shield Operation (Read)



CMAC MISS → Read CMAC from NVMe

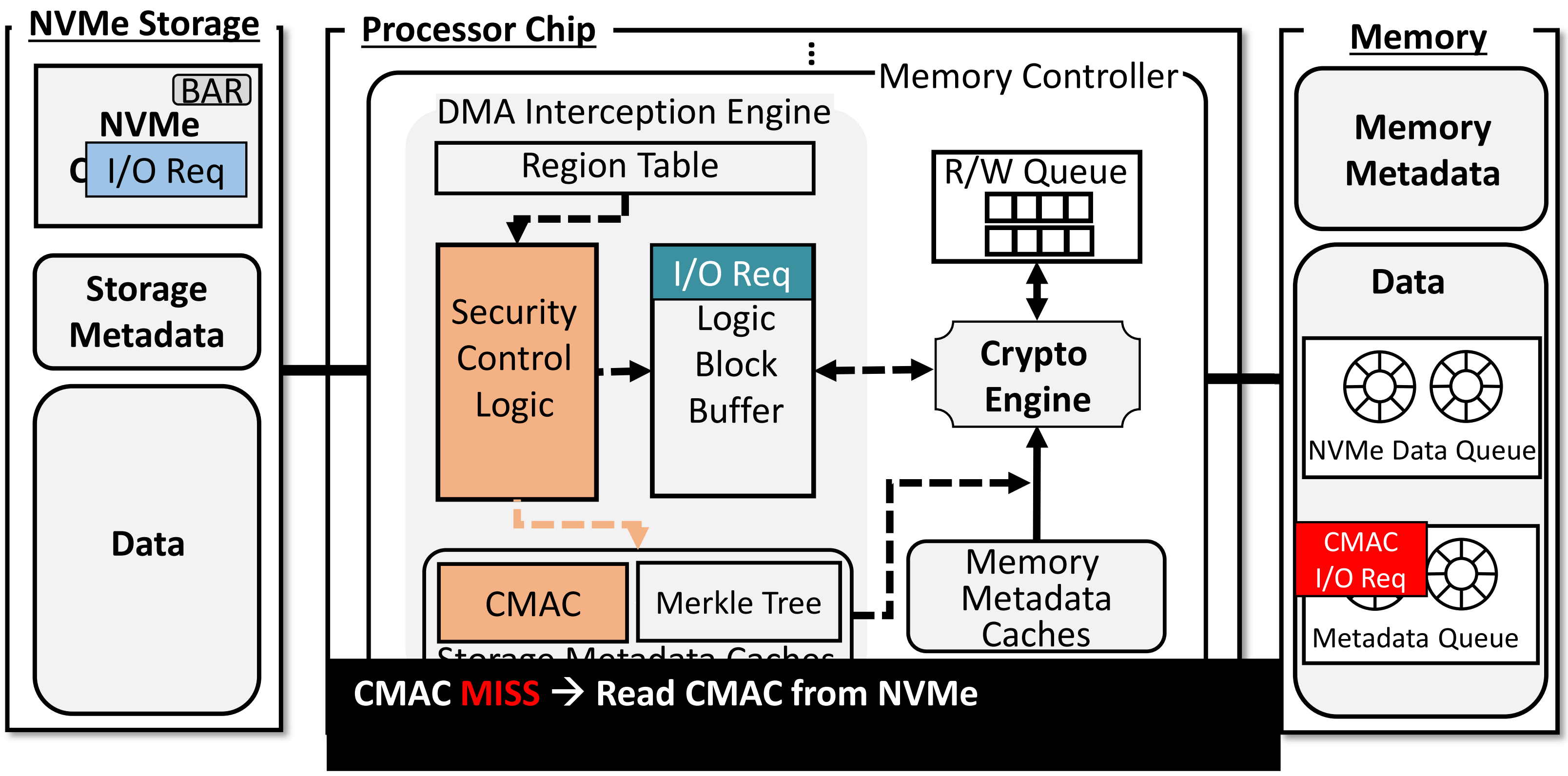
D-Shield Operation (Read)



CMAC MISS → Read CMAC from NVMe

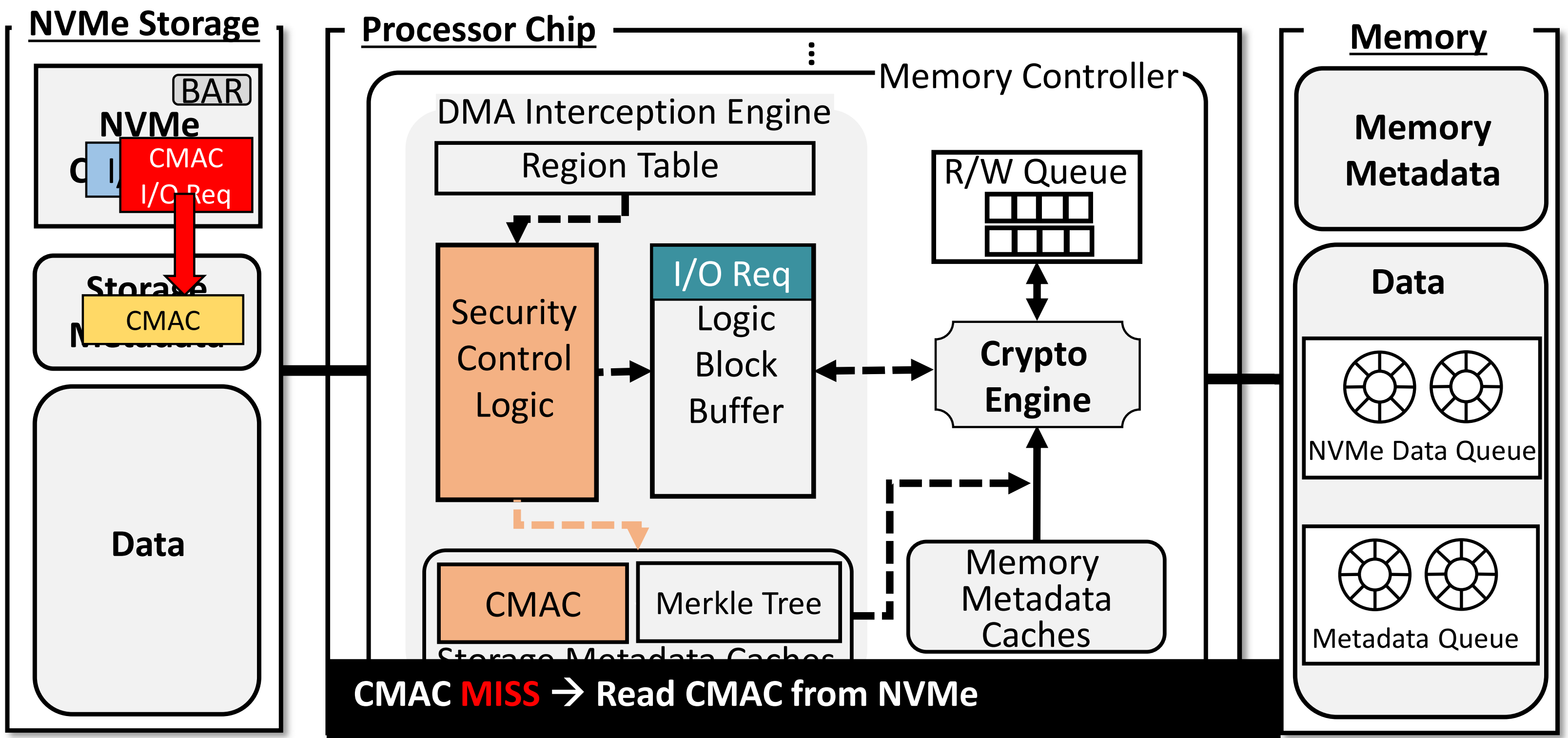
UCF

D-Shield Operation (Read)



CMAC MISS → Read CMAC from NVMe

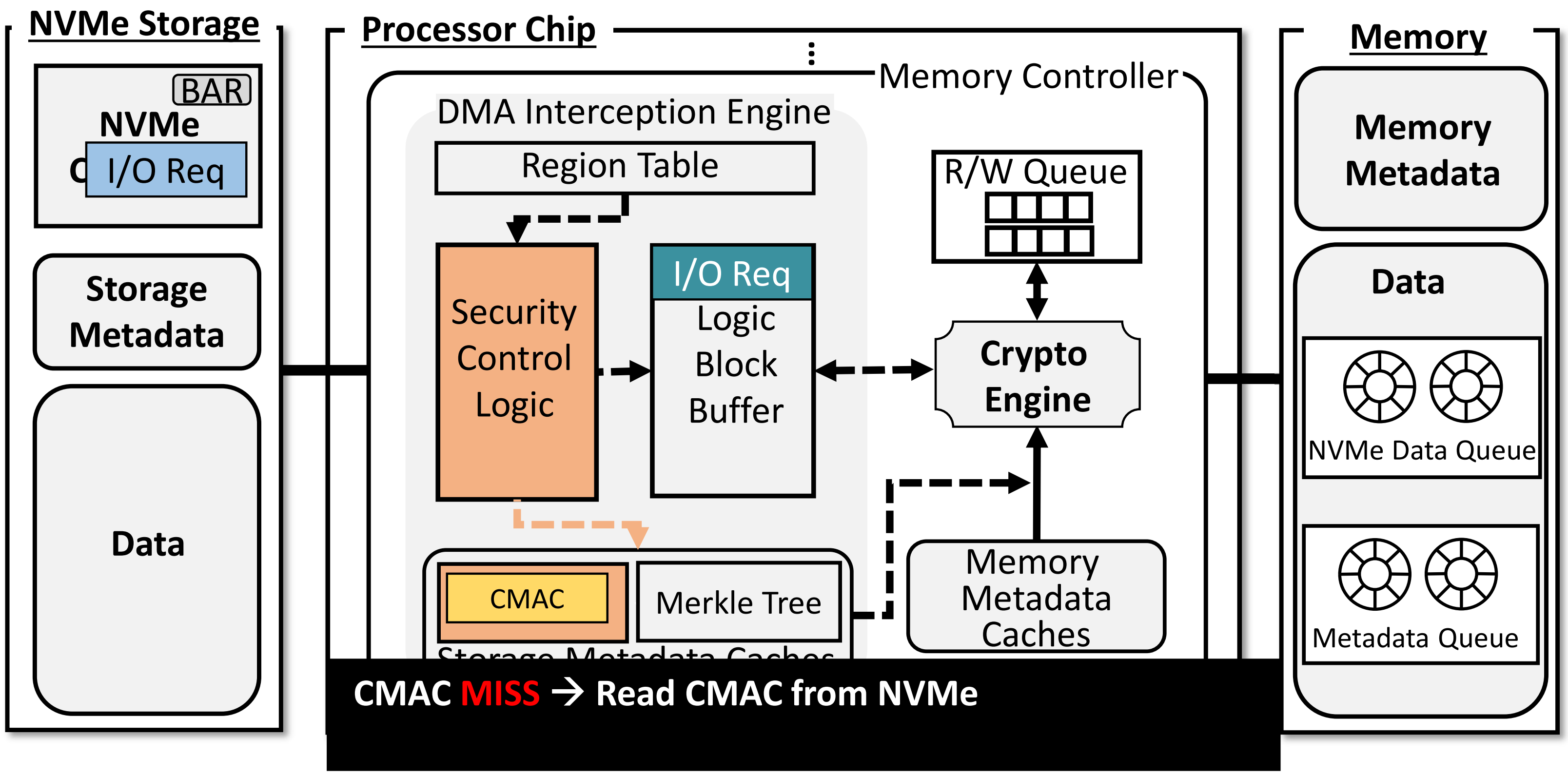
D-Shield Operation (Read)



CMAC MISS → Read CMAC from NVMe

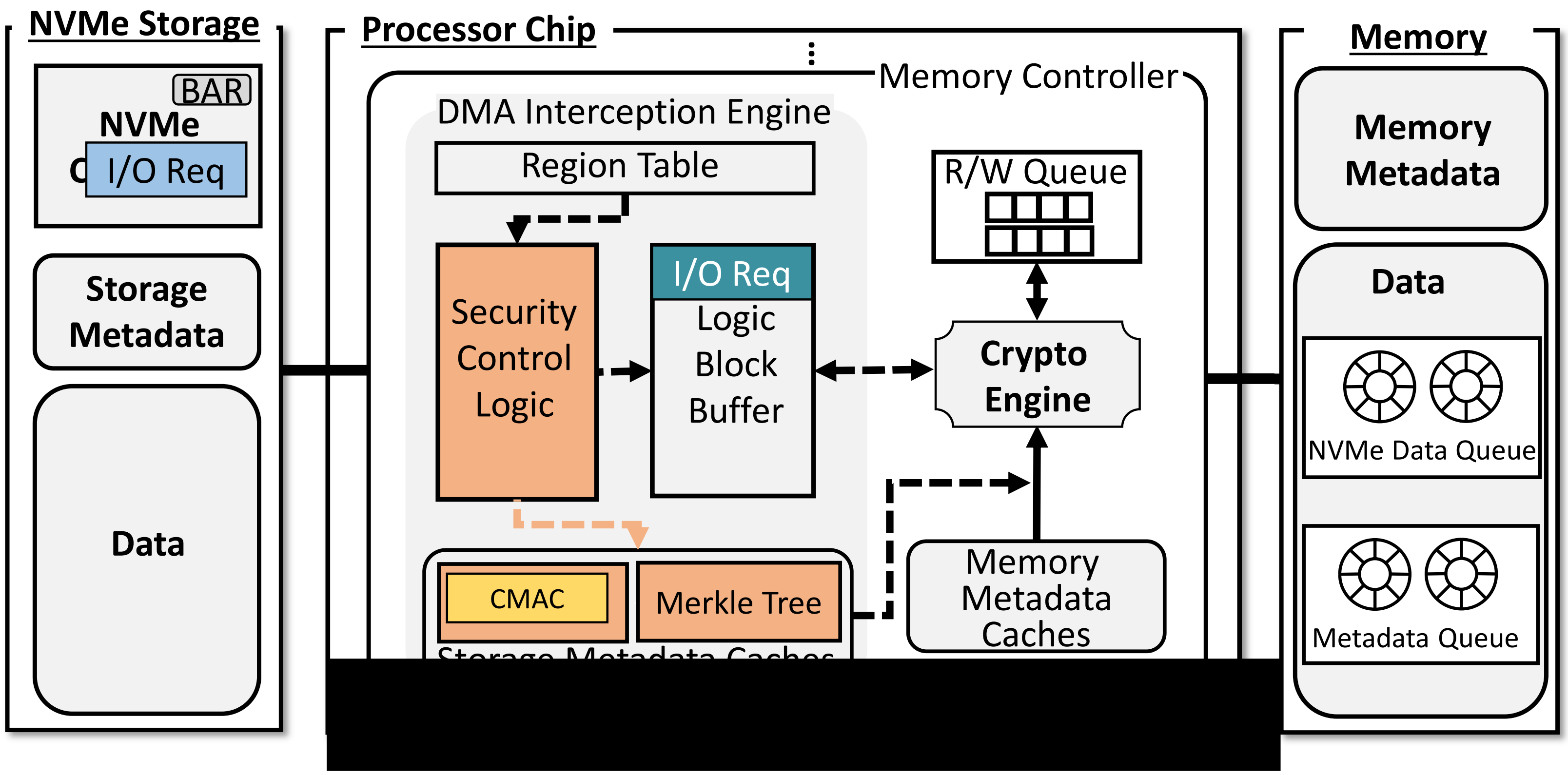
UCF

D-Shield Operation (Read)

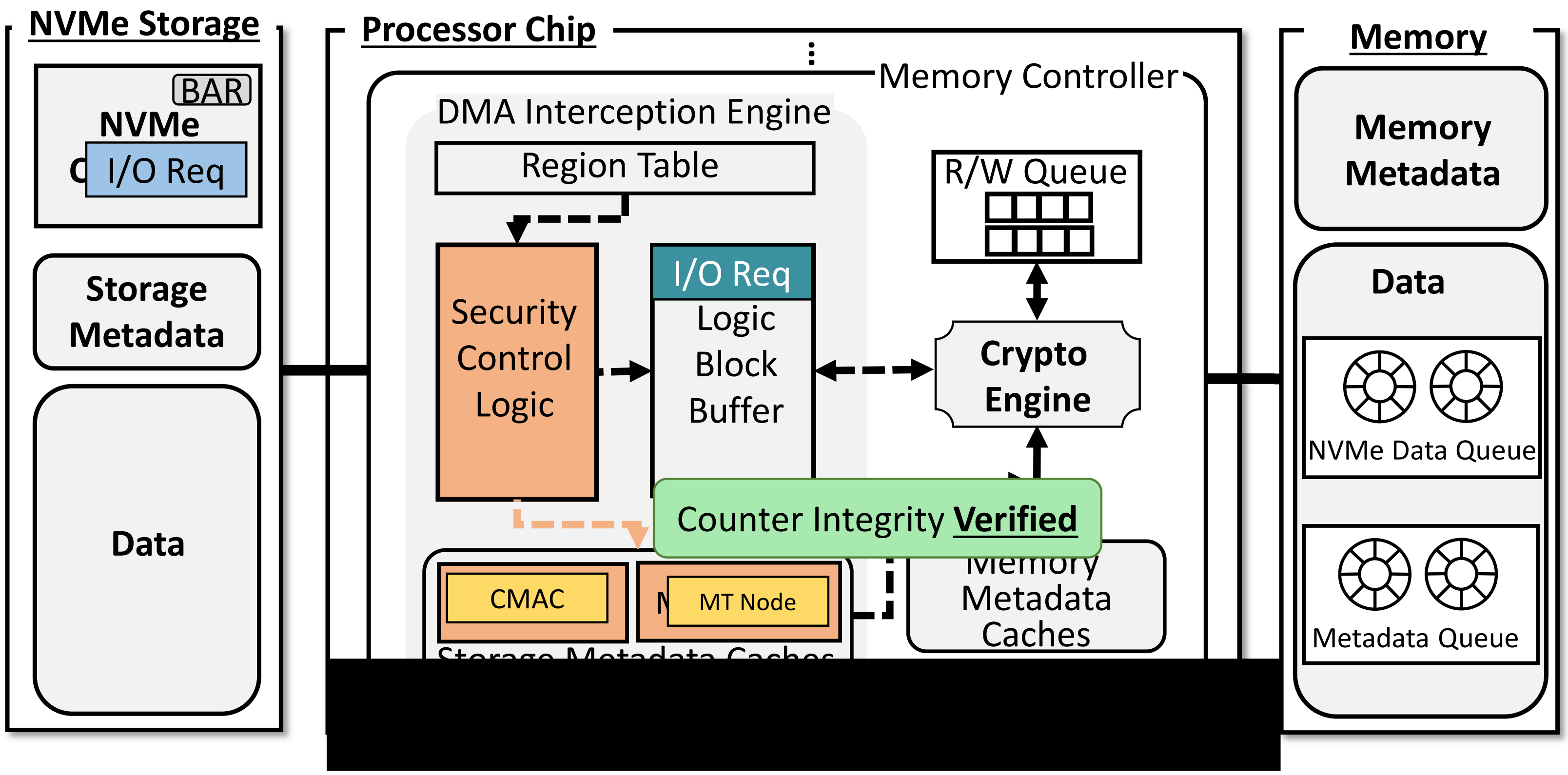


CMAC MISS → Read CMAC from NVMe

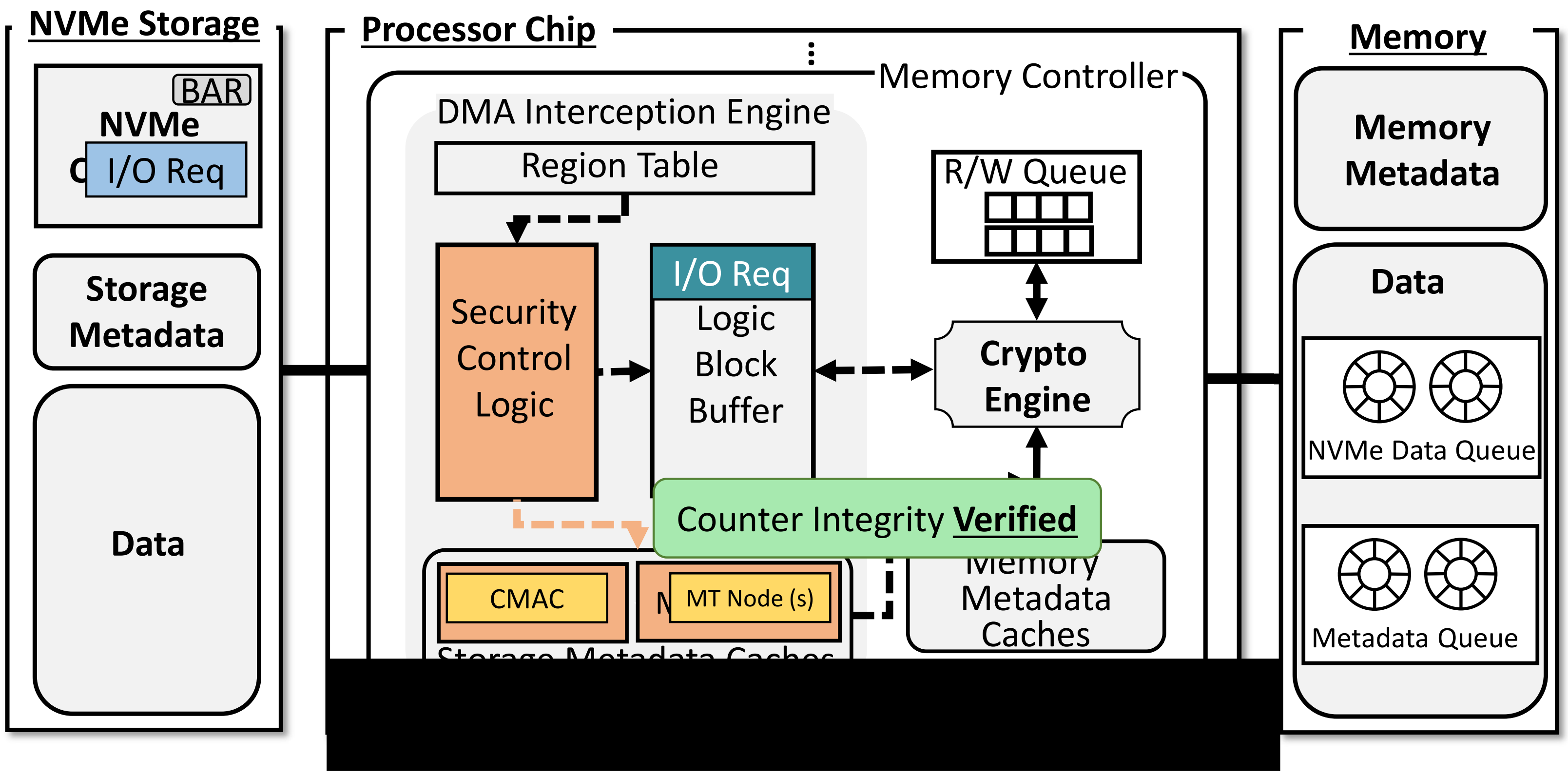
D-Shield Operation (Read)



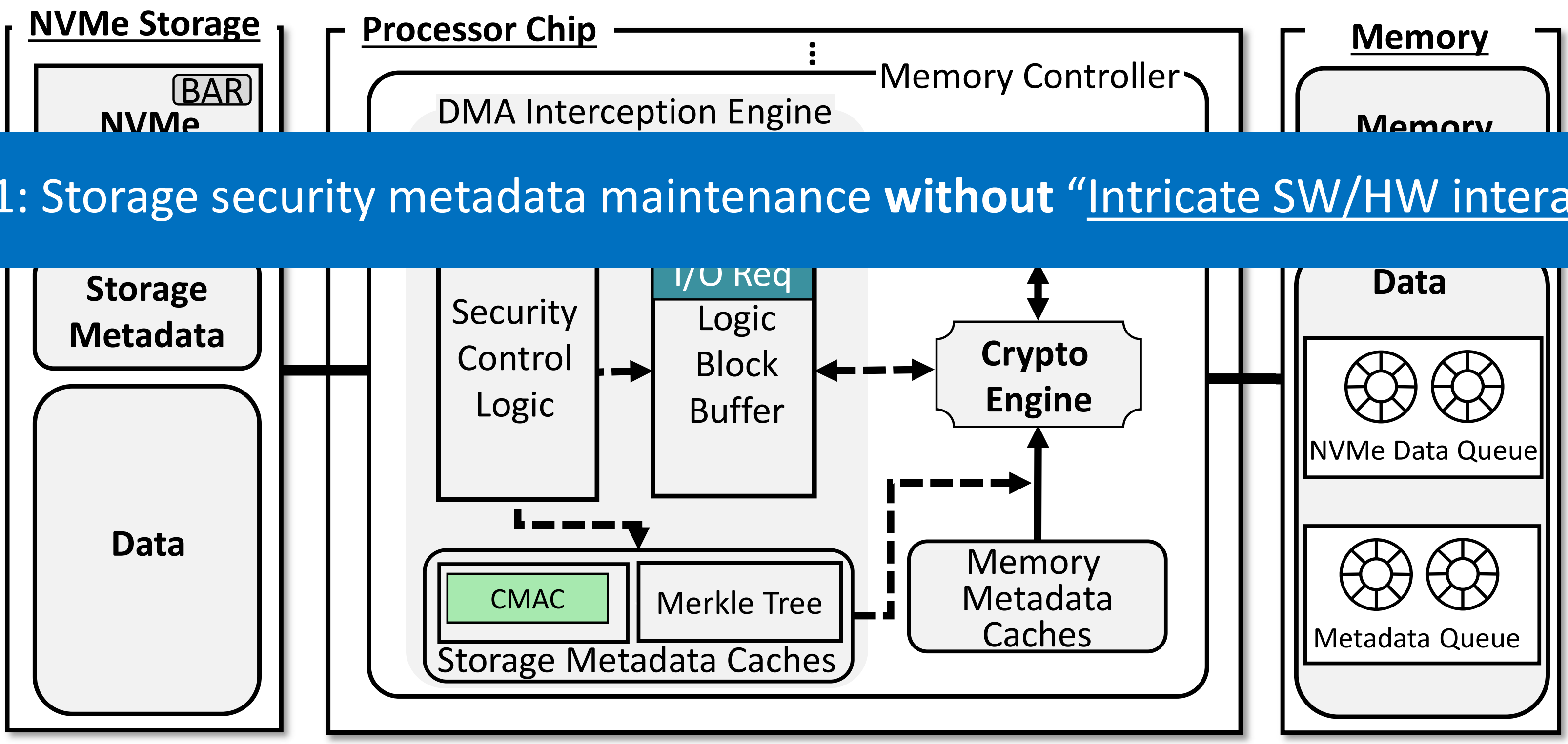
D-Shield Operation (Read)



D-Shield Operation (Read)

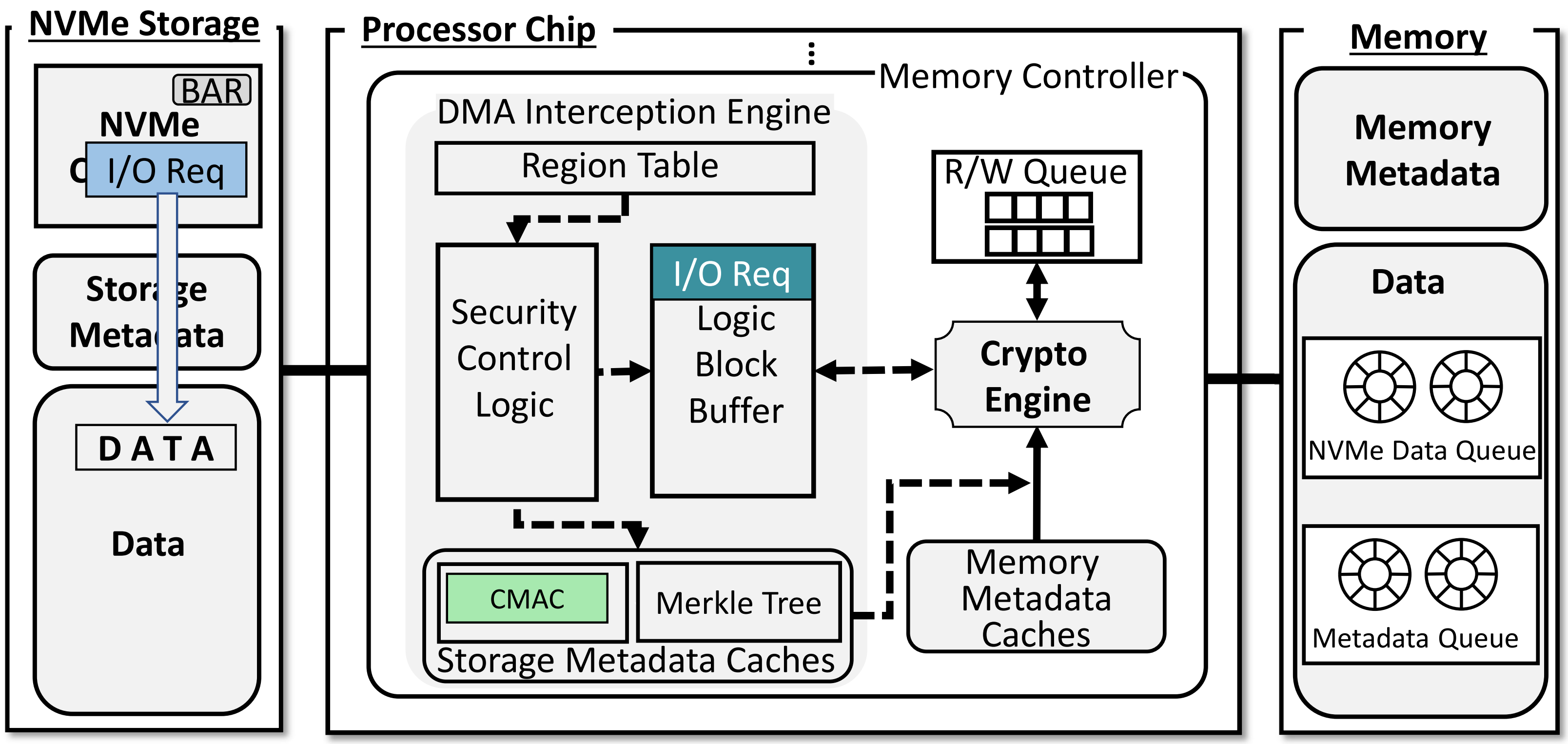


D-Shield Operation (Read)



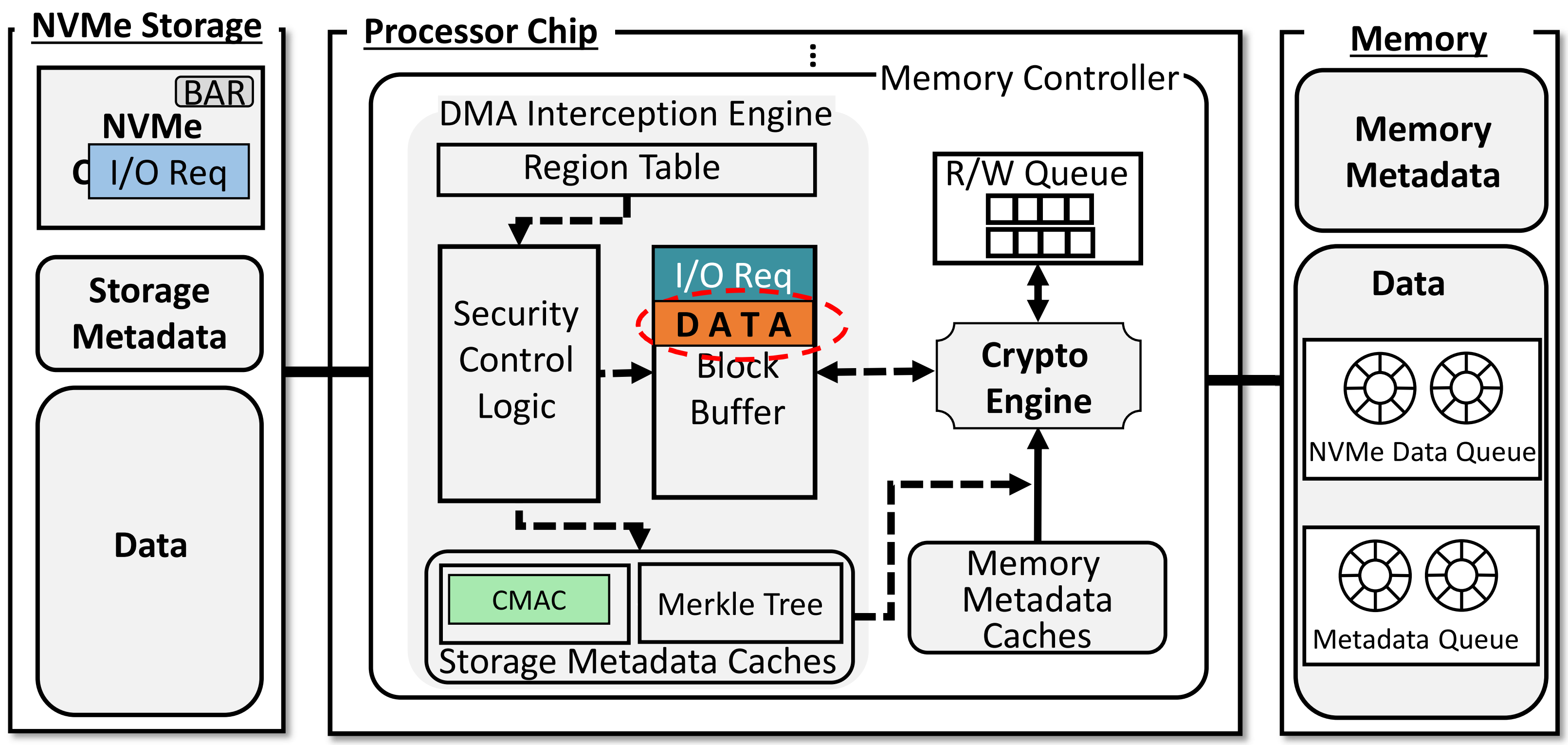
C1: Storage security metadata maintenance **without** "Intricate SW/HW interactions"

D-Shield Operation (Read)

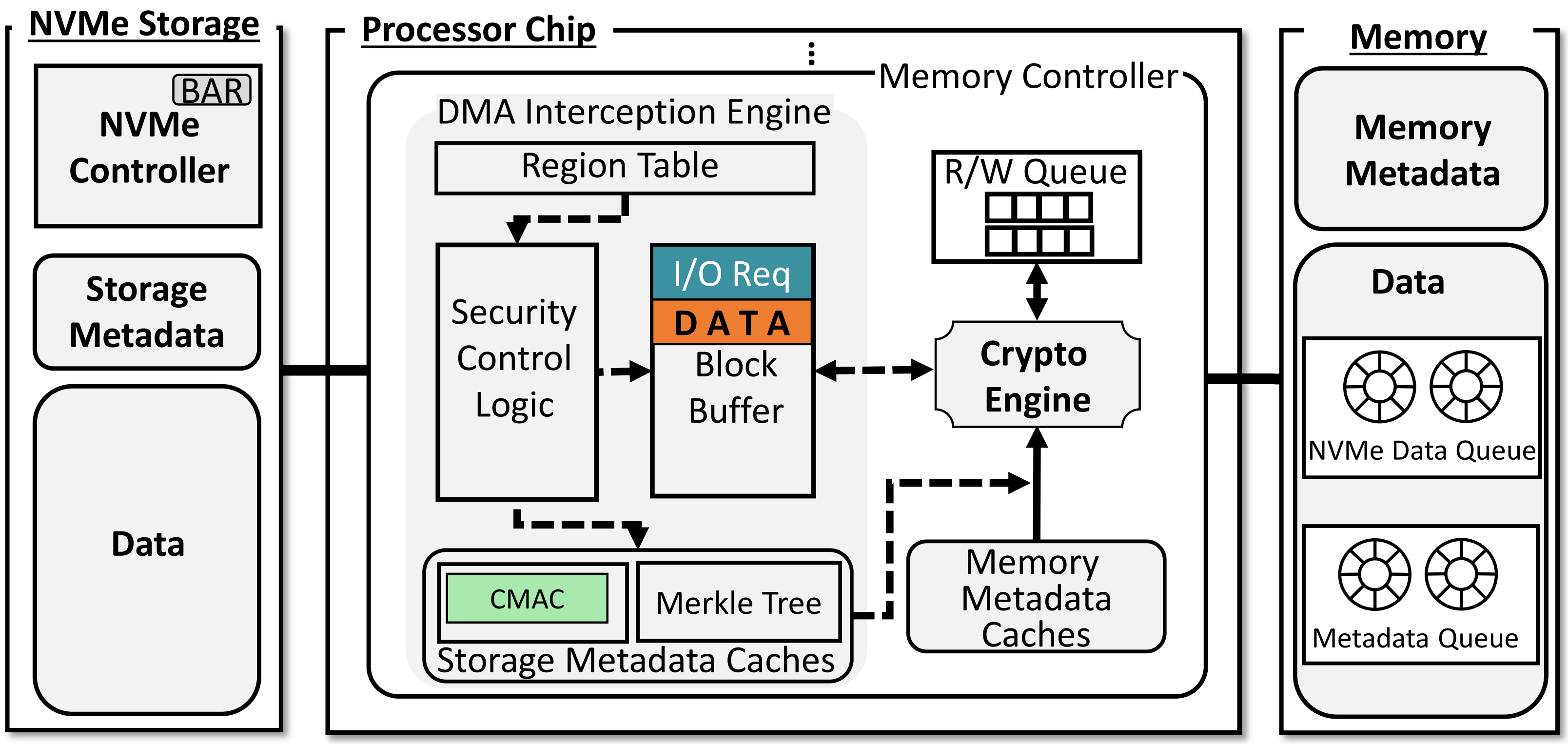


UCF

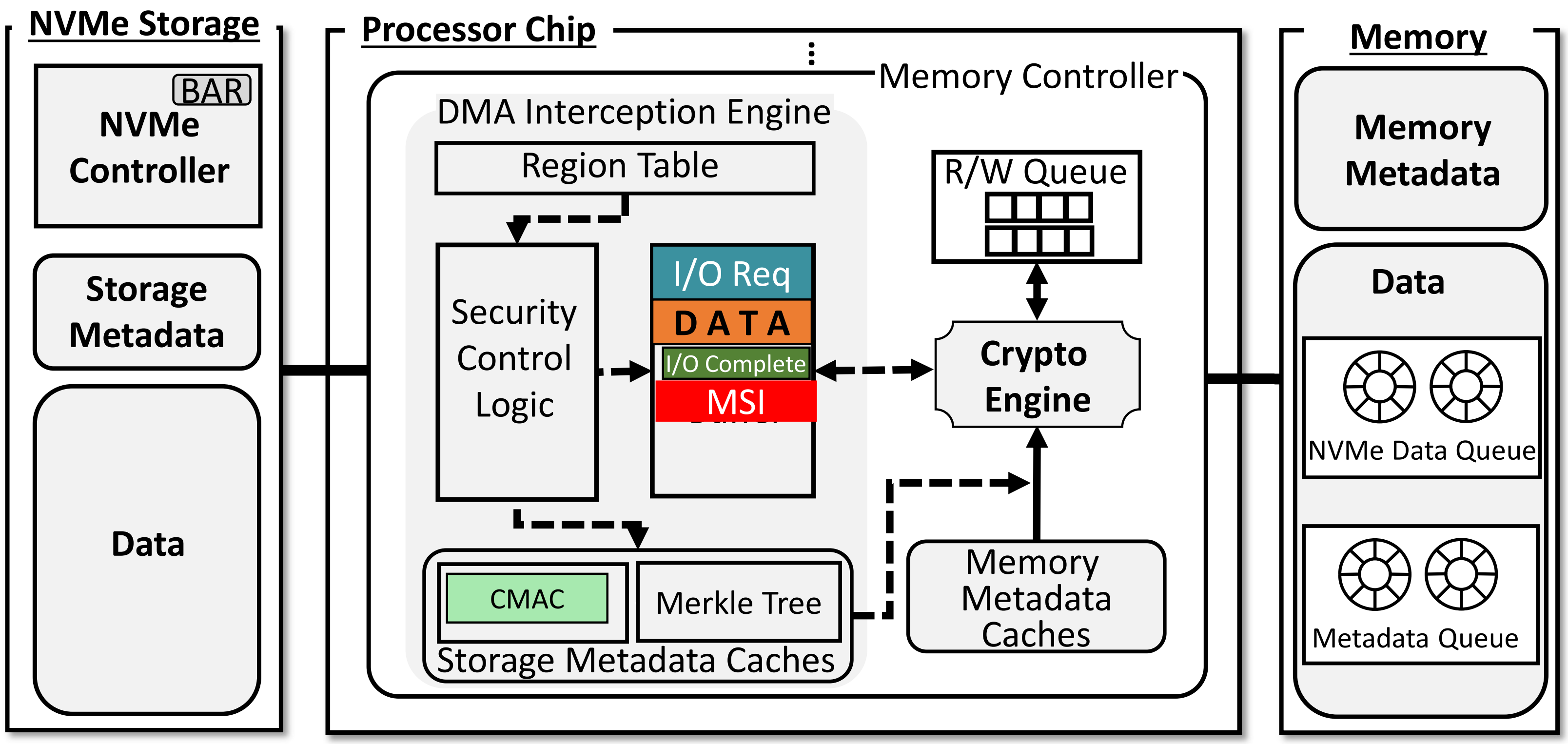
D-Shield Operation (Read)



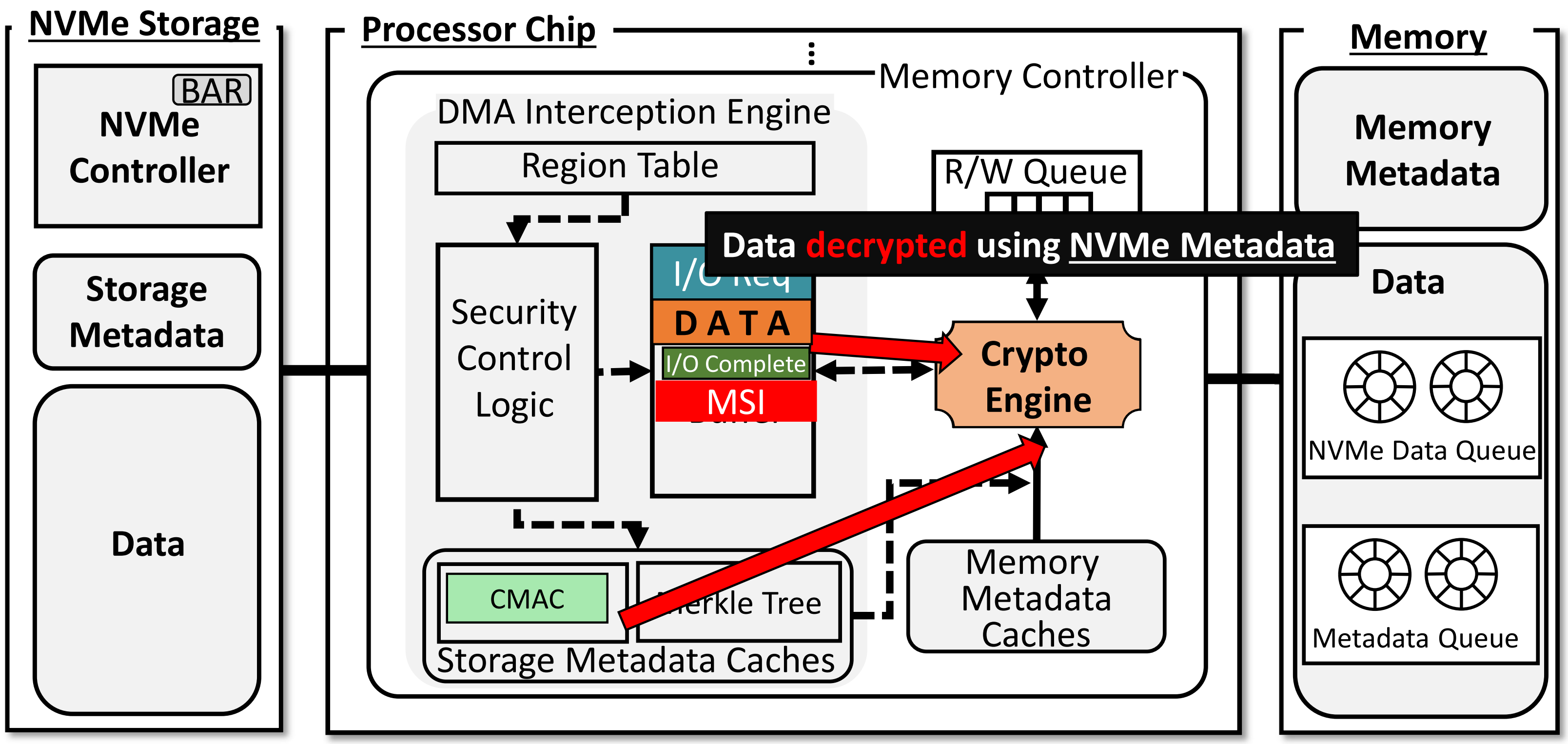
D-Shield Operation (Read)



D-Shield Operation (Read)

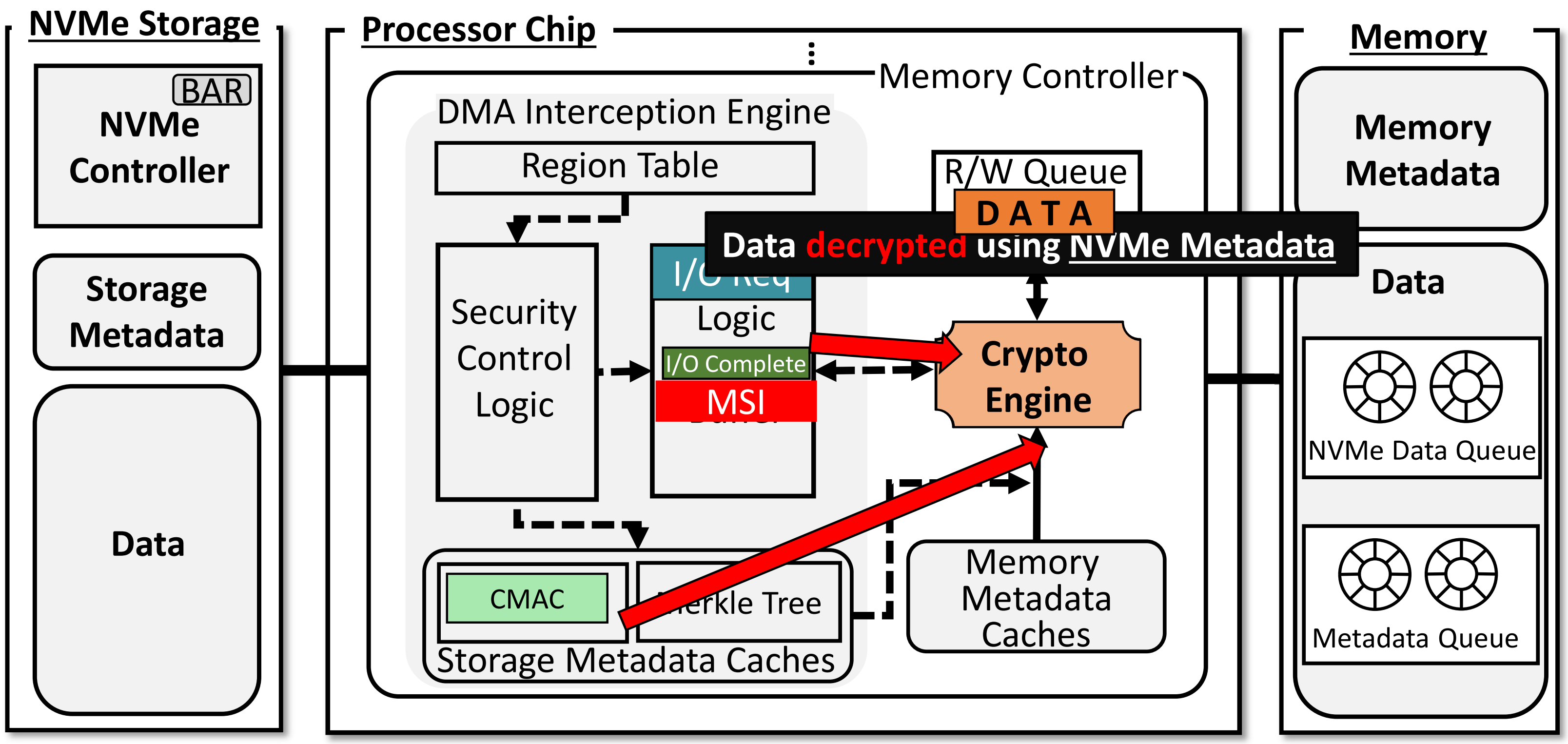


D-Shield Operation (Read)



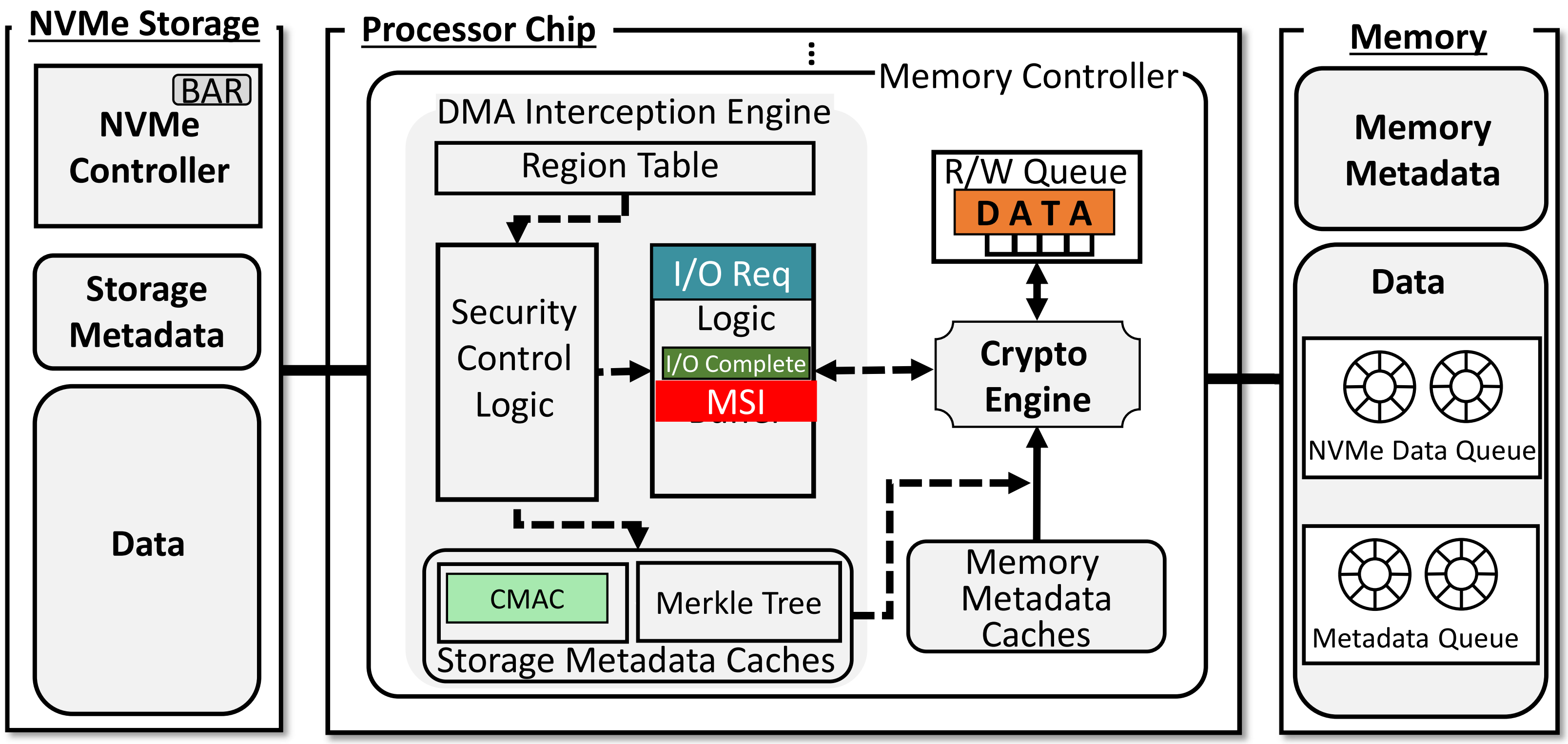
UCF

D-Shield Operation (Read)

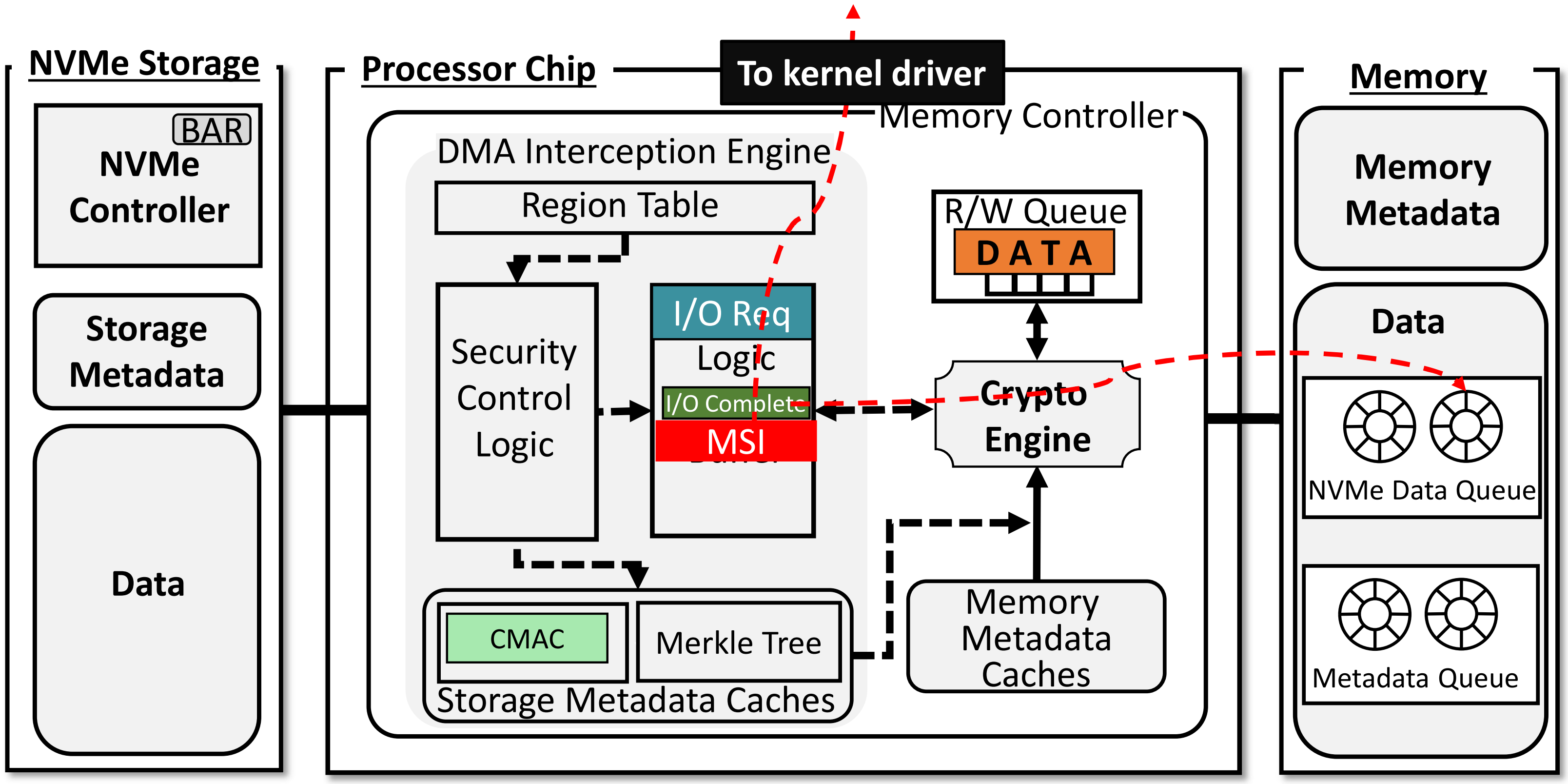


UCF

D-Shield Operation (Read)

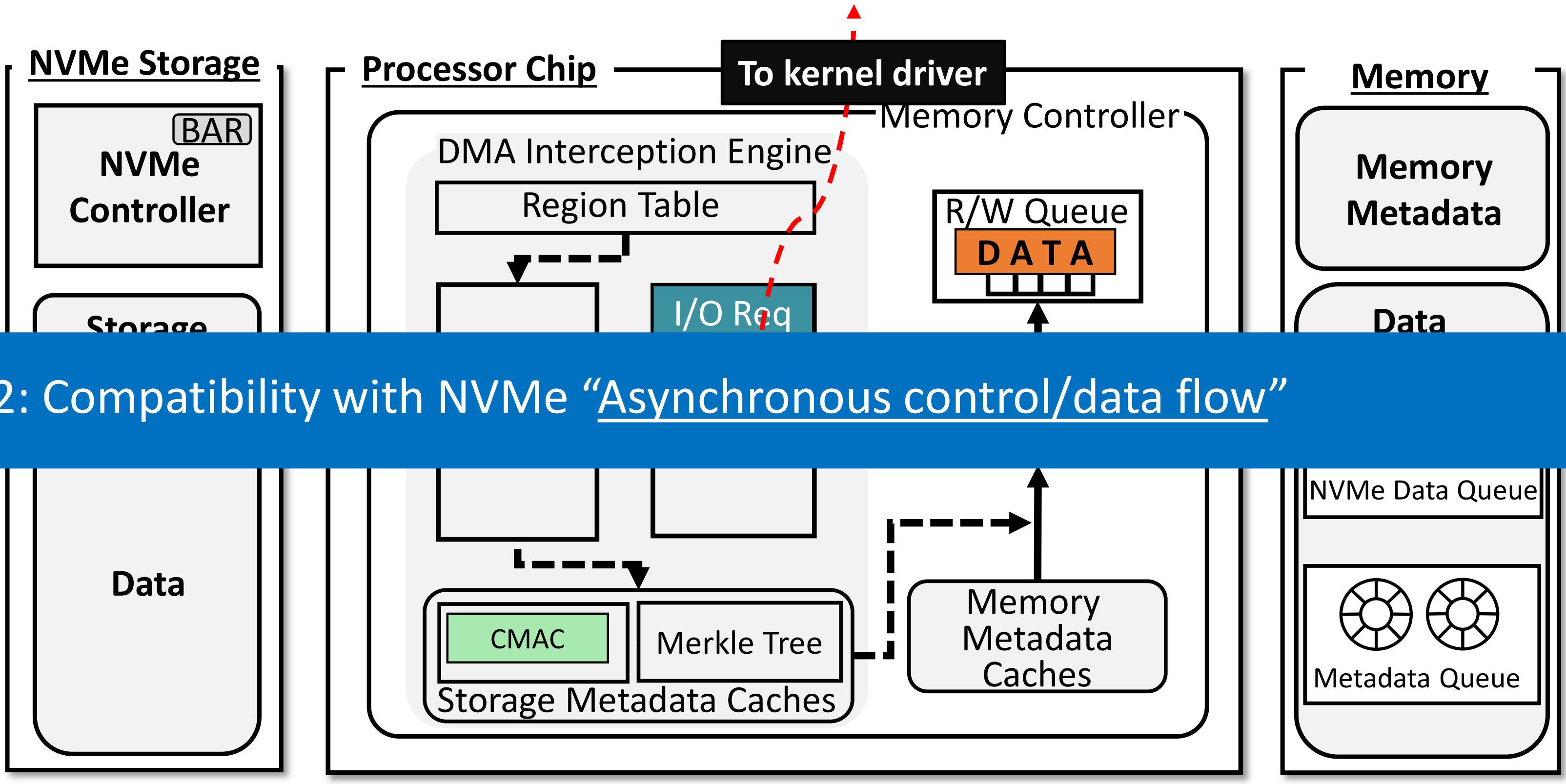


D-Shield Operation (Read)



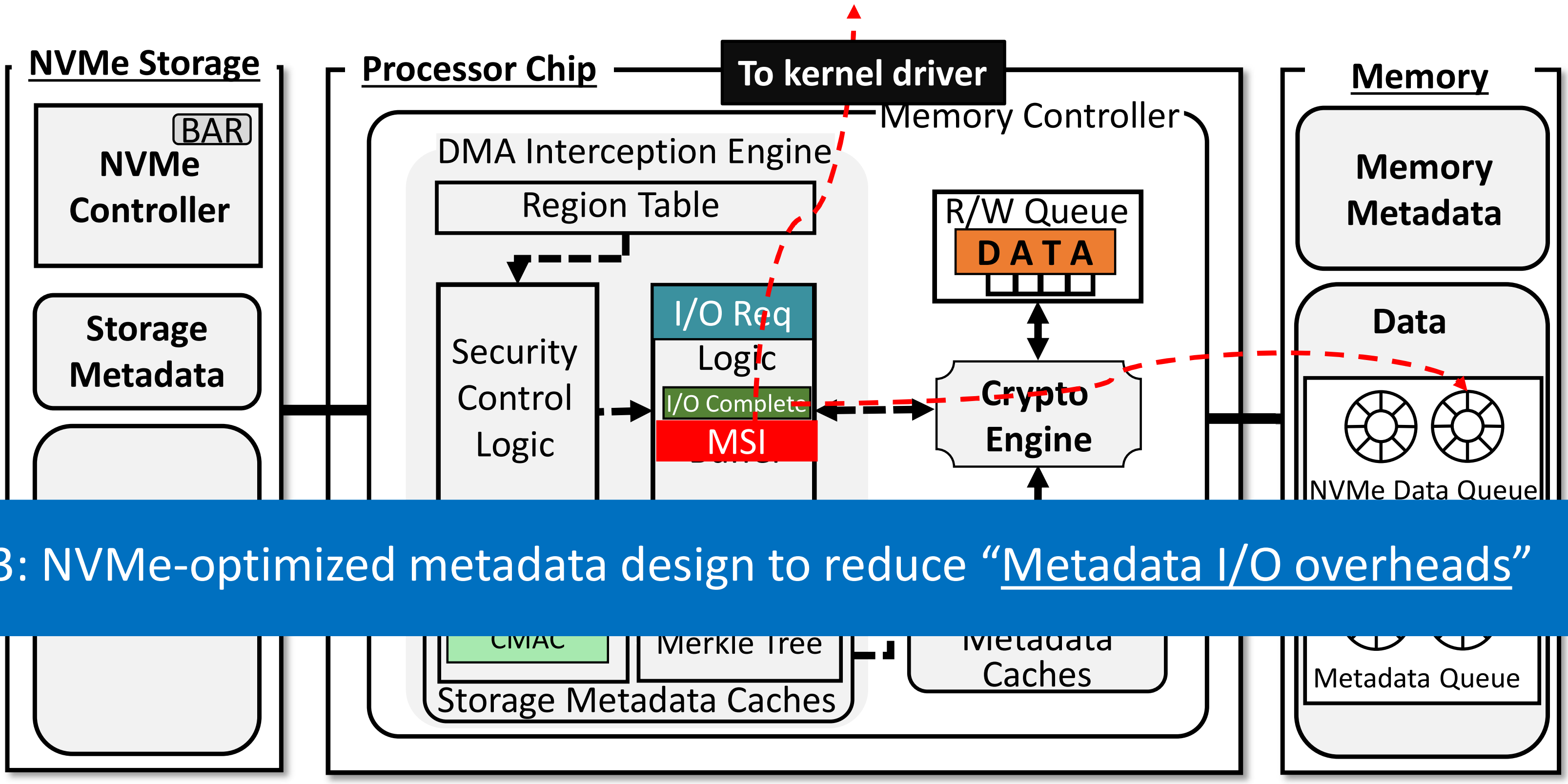
UCF

D-Shield Operation (Read)



C2: Compatibility with NVMe "Asynchronous control/data flow"

D-Shield Operation (Read)



C3: NVMe-optimized metadata design to reduce "Metadata I/O overheads"

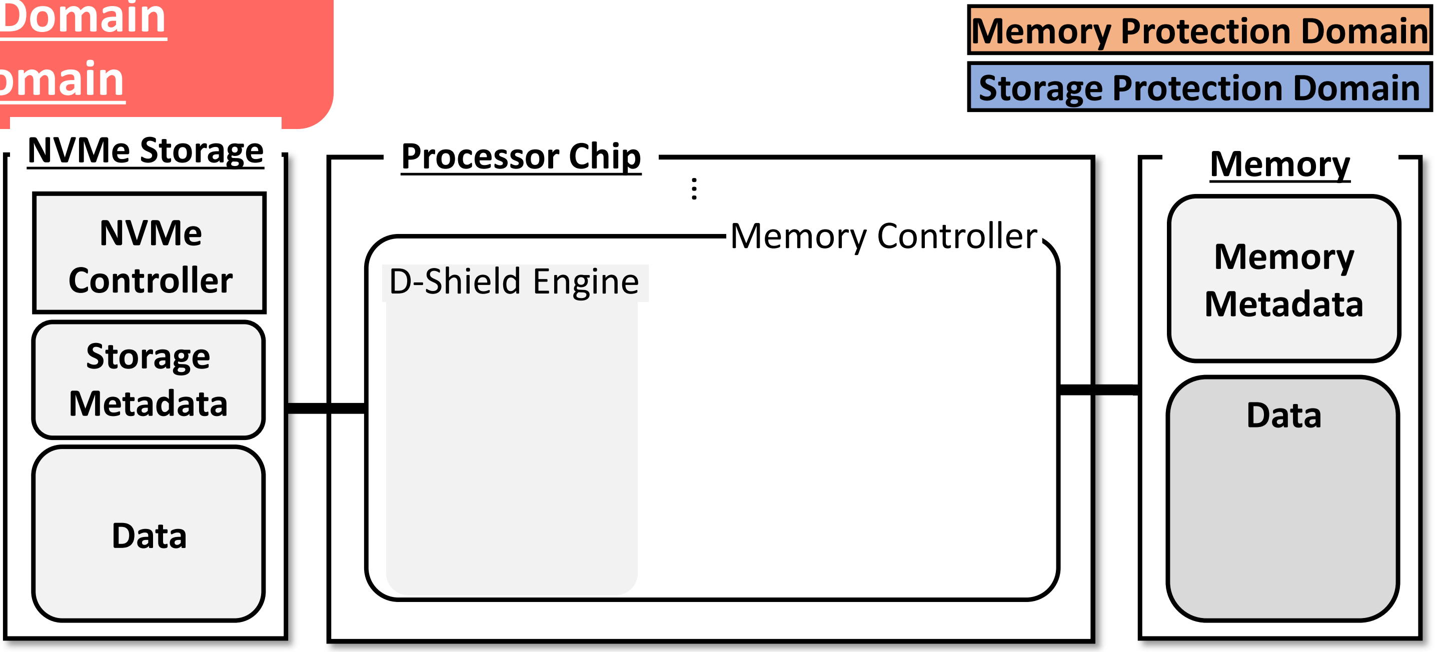
D-Shield-Hyb: Cross Domain Access Optimization

- Basic D-Shield provides proper off-chip data security with standalone protection for memory and storage

D-Shield-Hyb: Cross Domain Access Optimization

- Basic D-Shield provides proper off-chip data security with standalone protection for memory and storage

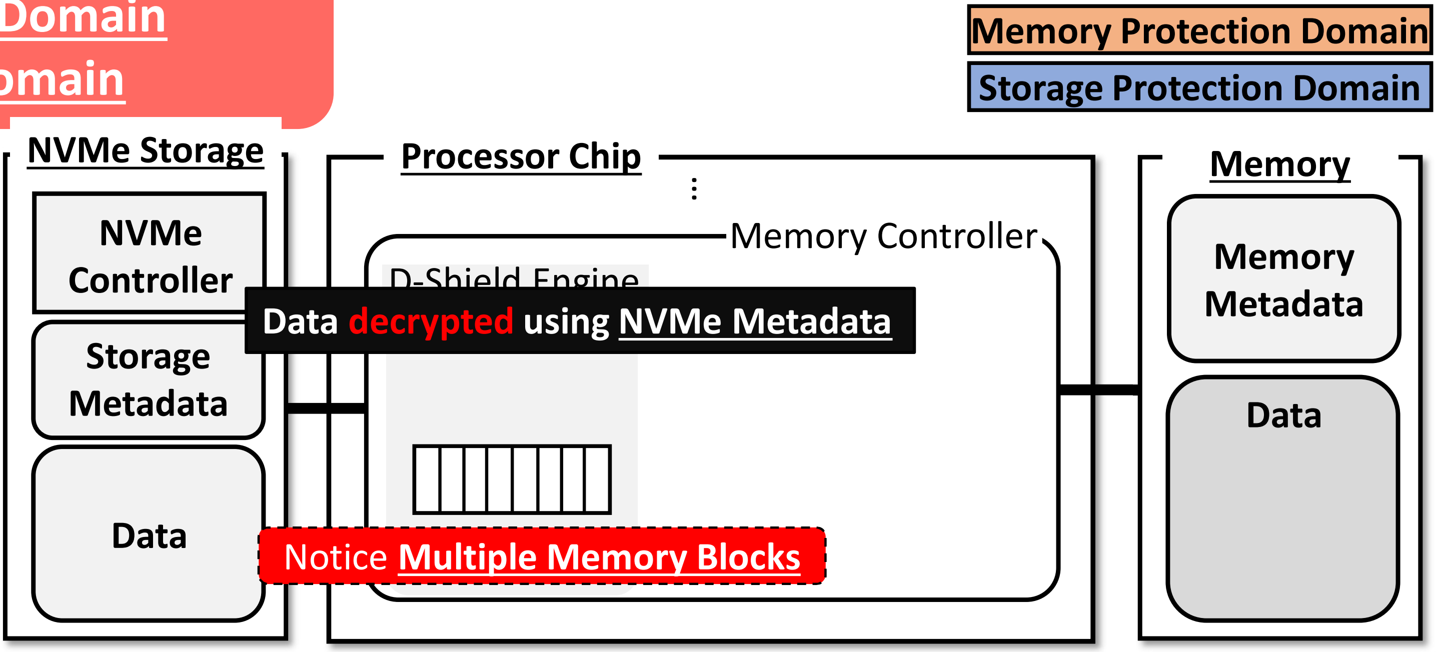
Cross-domain data transfer:
From Storage Domain
To Memory Domain



D-Shield-Hyb: Cross Domain Access Optimization

- Basic D-Shield provides proper off-chip data security with standalone protection for memory and storage

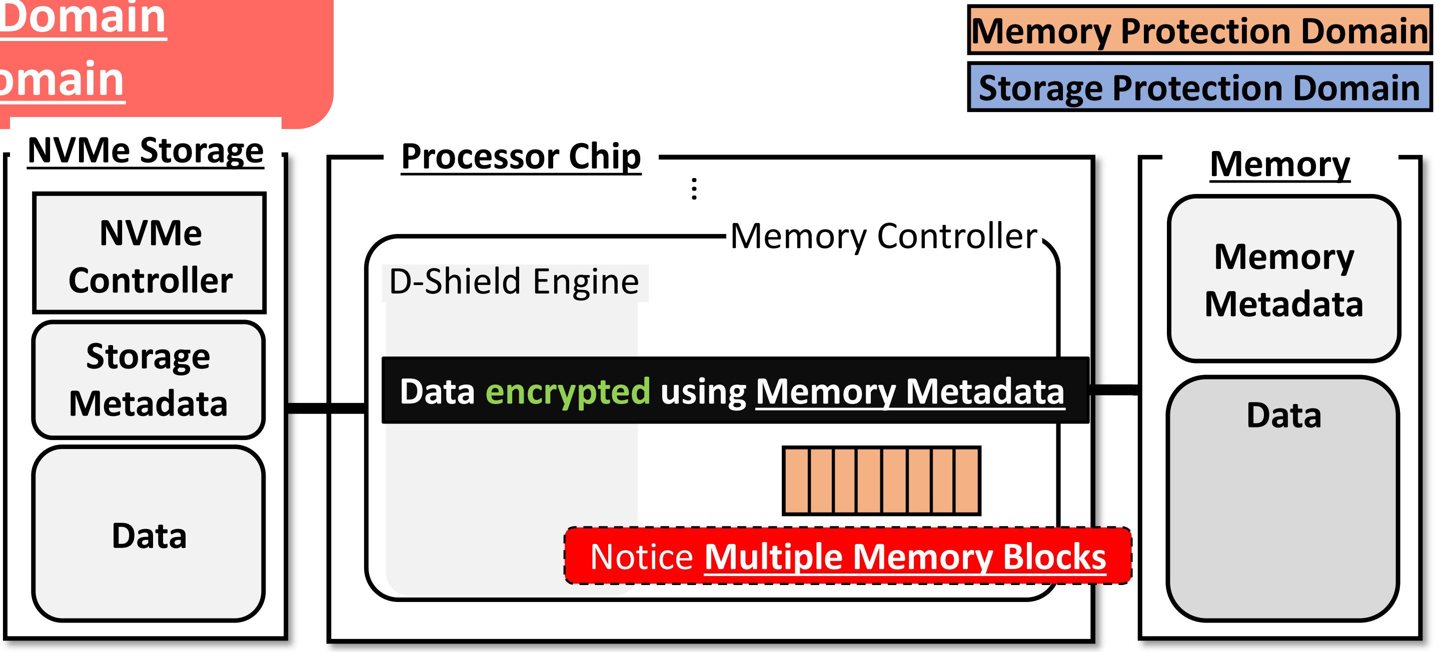
Cross-domain data transfer:
From Storage Domain
To Memory Domain



D-Shield-Hyb: Cross Domain Access Optimization

- Basic D-Shield provides proper off-chip data security with standalone protection for memory and storage

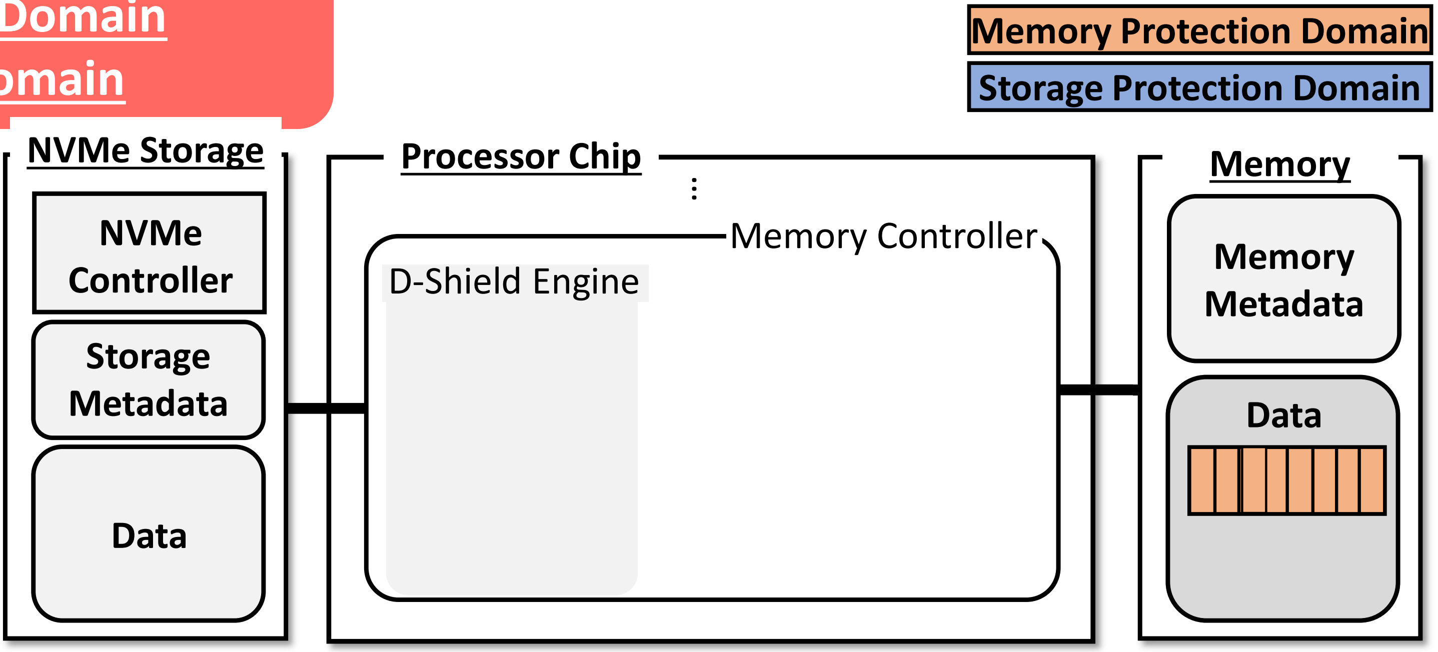
Cross-domain data transfer:
From Storage Domain
To Memory Domain



D-Shield-Hyb: Cross Domain Access Optimization

- Basic D-Shield provides proper off-chip data security with standalone protection for memory and storage

Cross-domain data transfer:
From Storage Domain
To Memory Domain

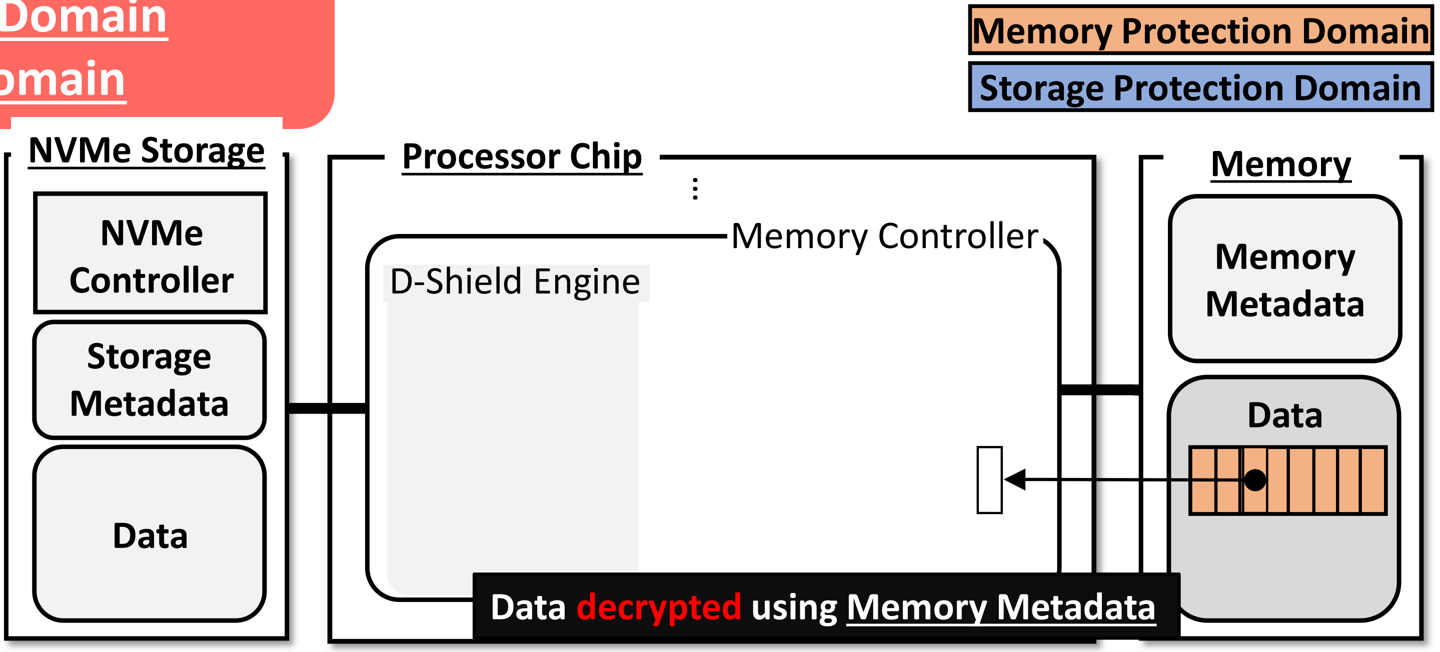


UCF

D-Shield-Hyb: Cross Domain Access Optimization

- Basic D-Shield provides proper off-chip data security with standalone protection for memory and storage

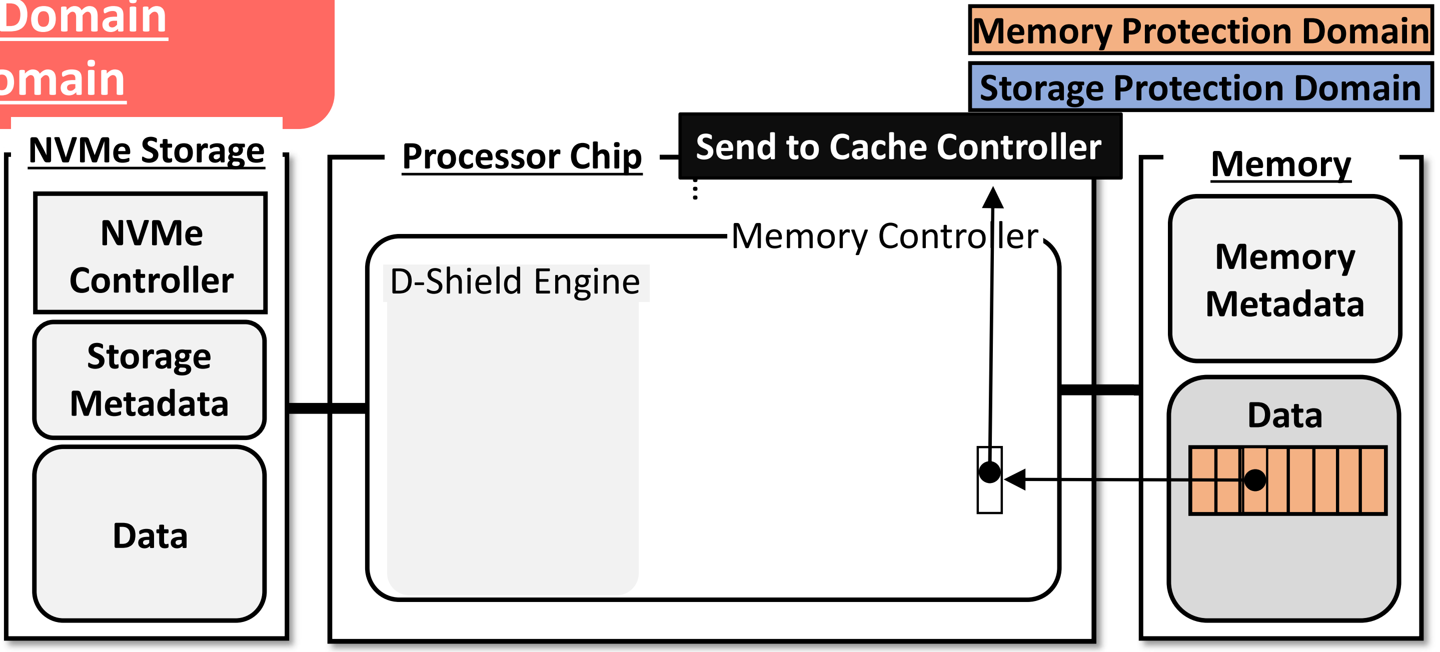
Cross-domain data transfer:
From Storage Domain
To Memory Domain



D-Shield-Hyb: Cross Domain Access Optimization

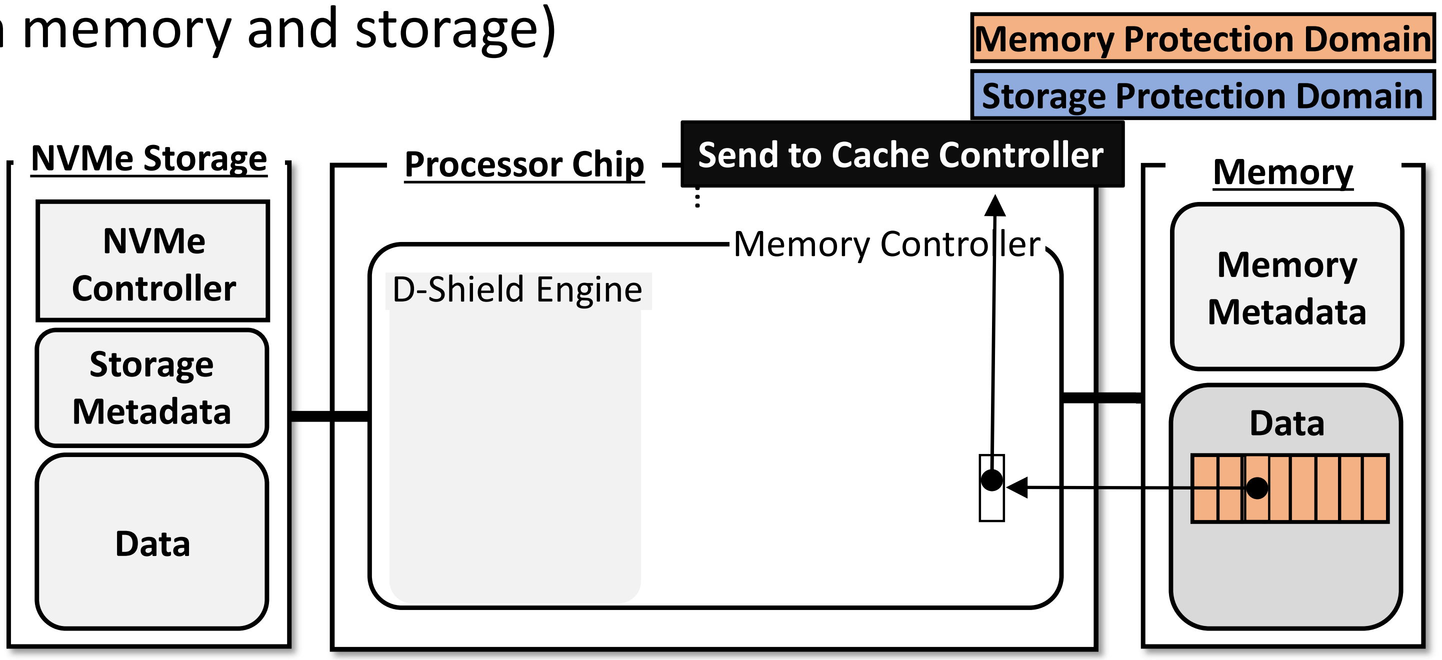
- Basic D-Shield provides proper off-chip data security with standalone protection for memory and storage

Cross-domain data transfer:
From Storage Domain
To Memory Domain



D-Shield-Hyb: Cross Domain Access Optimization

- Basic D-Shield provides proper off-chip data security with standalone protection for memory and storage
- Re-encryption of data is needed as it transfers between domains (i.e., between memory and storage)



UCF

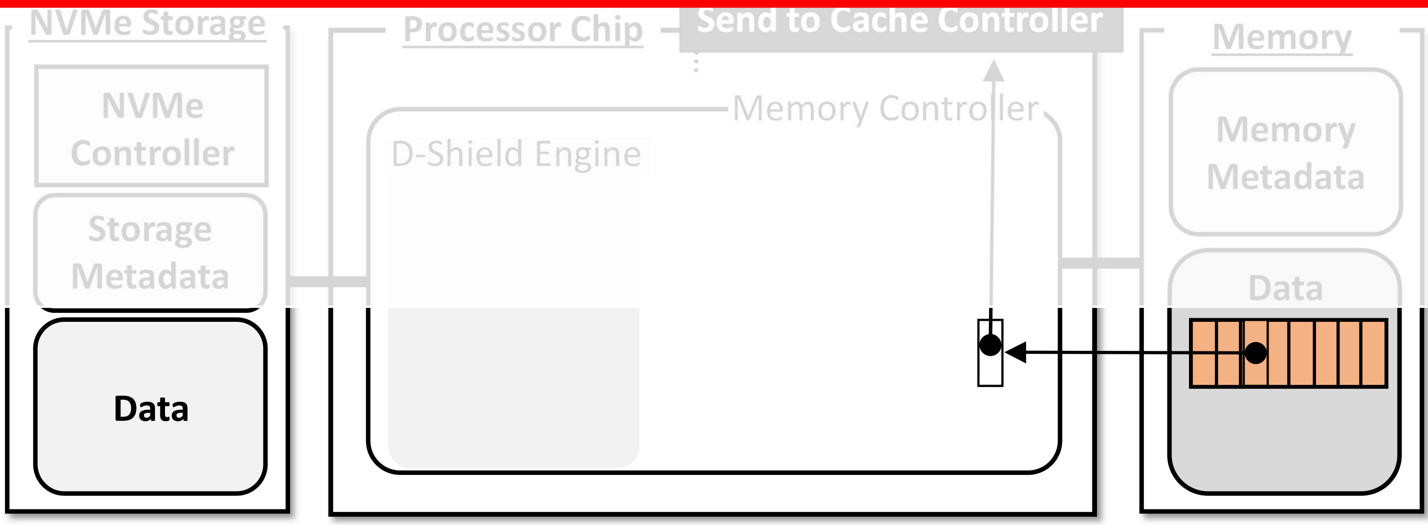
D-Shield-Hyb: Cross Domain Access Optimization

- Basic D-Shield provides proper off-chip data security with standalone protection for memory and storage

Moving data between protection domains (i.e., memory and storage) can be *expensive*

Additional utilization of the cryptographic engine

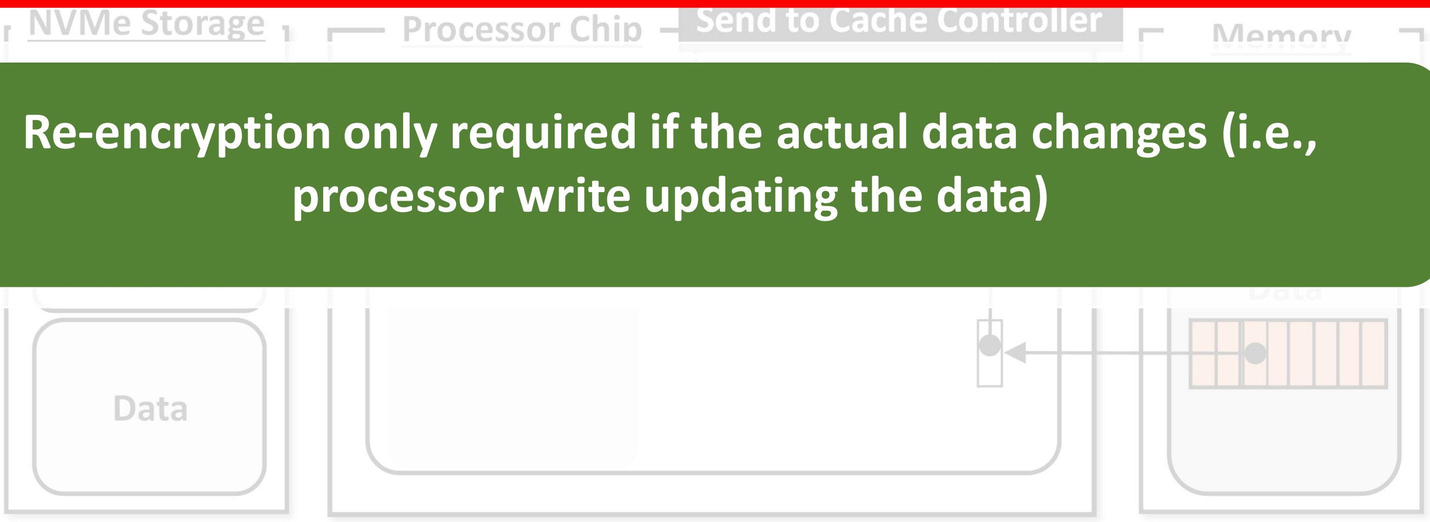
Prolonged NVMe data path



D-Shield-Hyb: Cross Domain Access Optimization

- Basic D-Shield provides proper off-chip data security with standalone protection for memory and storage

Moving data between protection domains (i.e., memory and storage) can be *expensive*
Additional utilization of the cryptographic engine
Prolonged NVMe data path



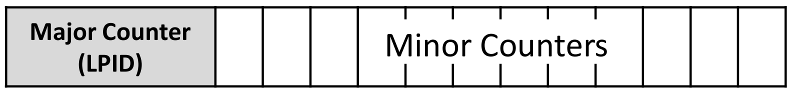
Re-encryption only required if the actual data changes (i.e., processor write updating the data)

D-Shield-Hyb: Cross Domain Access Optimization

- Bookkeep the ownership of logic blocks **in memory**

D-Shield-Hyb: Cross Domain Access Optimization

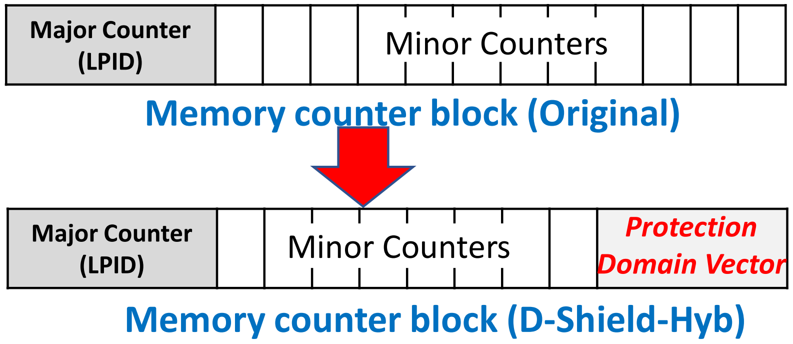
- Bookkeep the ownership of logic blocks **in memory**



Memory counter block (Original)

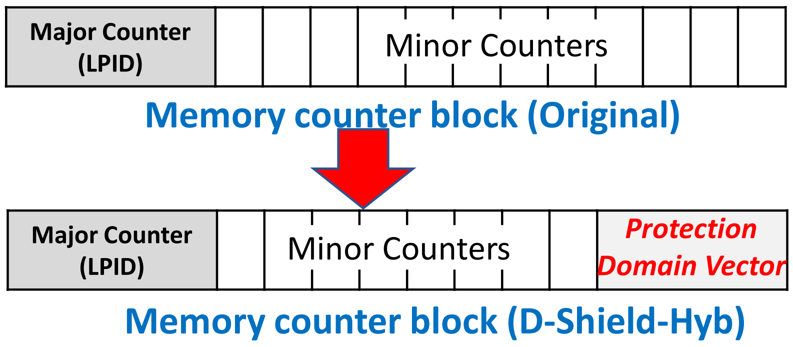
D-Shield-Hyb: Cross Domain Access Optimization

- Bookkeep the ownership of logic blocks **in memory**

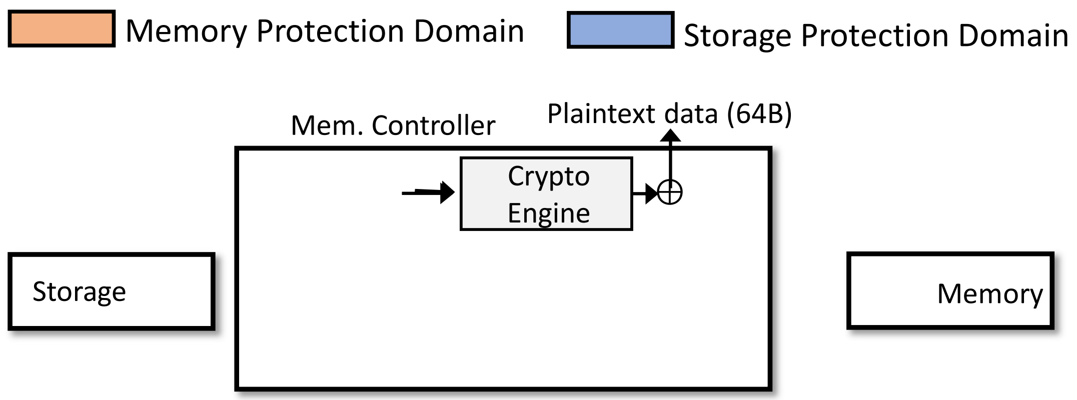


D-Shield-Hyb: Cross Domain Access Optimization

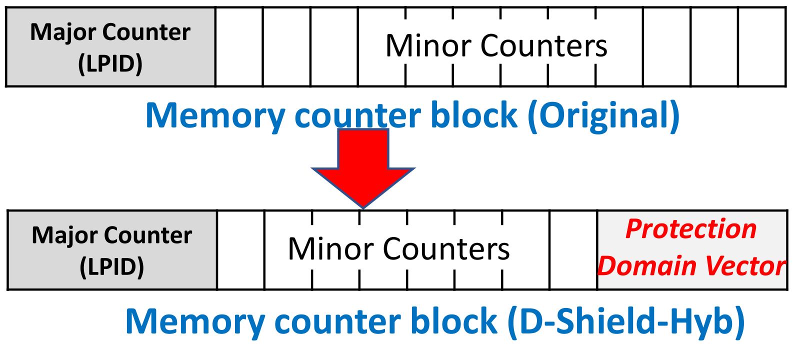
- Bookkeep the ownership of logic blocks **in memory**
- Track the security domain for transferred data block



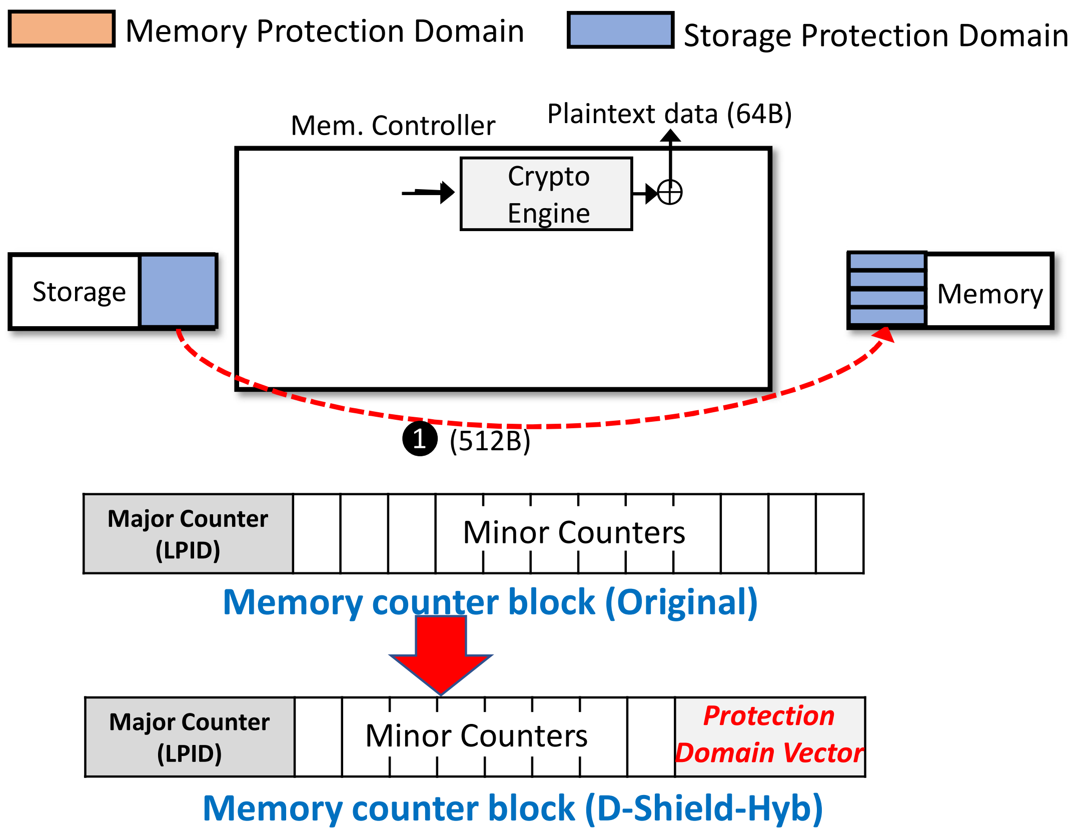
D-Shield-Hyb: Cross Domain Access Optimization



- Bookkeep the ownership of logic blocks **in memory**
- Track the security domain for transferred data block

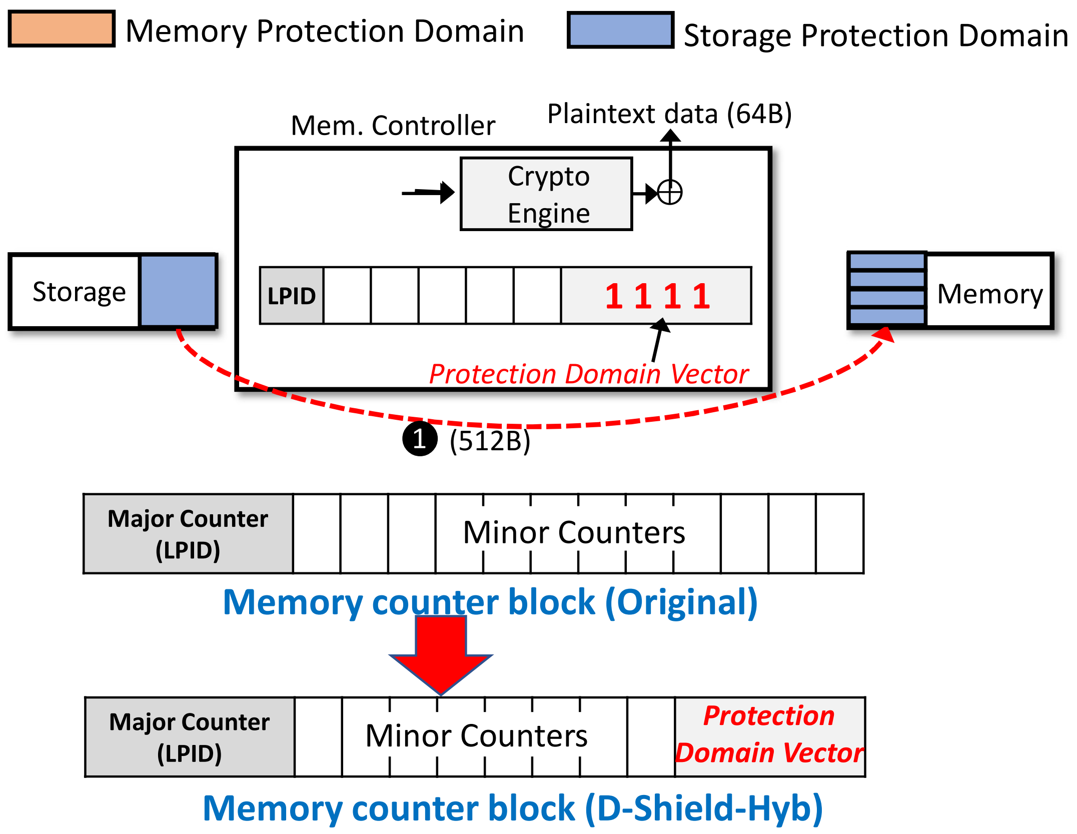


D-Shield-Hyb: Cross Domain Access Optimization



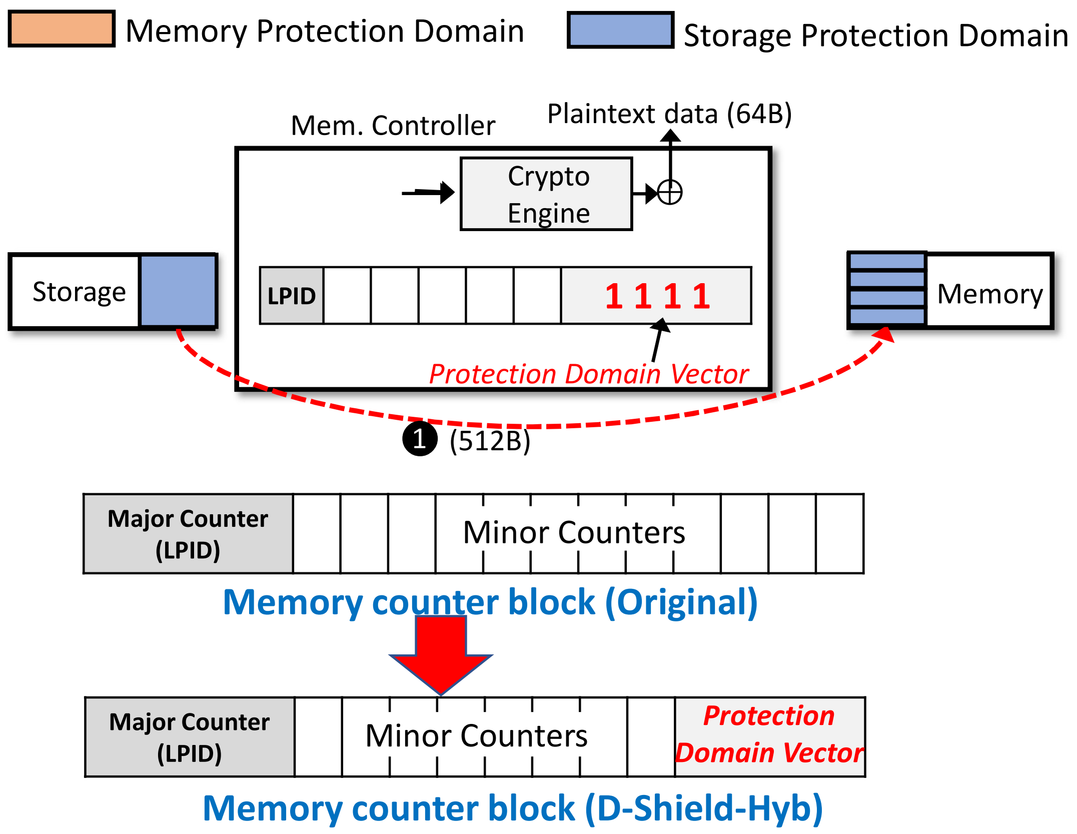
- Bookkeep the ownership of logic blocks **in memory**
- Track the security domain for transferred data block

D-Shield-Hyb: Cross Domain Access Optimization



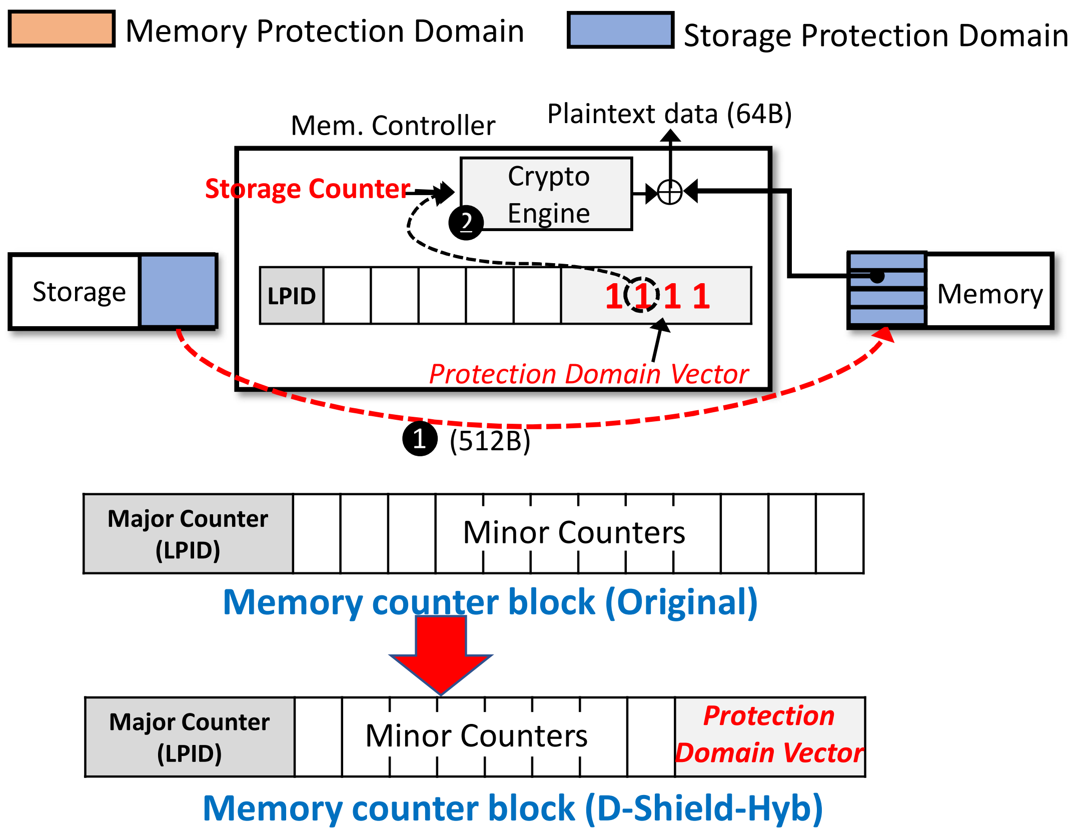
- Bookkeep the ownership of logic blocks **in memory**
- Track the security domain for transferred data block

D-Shield-Hyb: Cross Domain Access Optimization



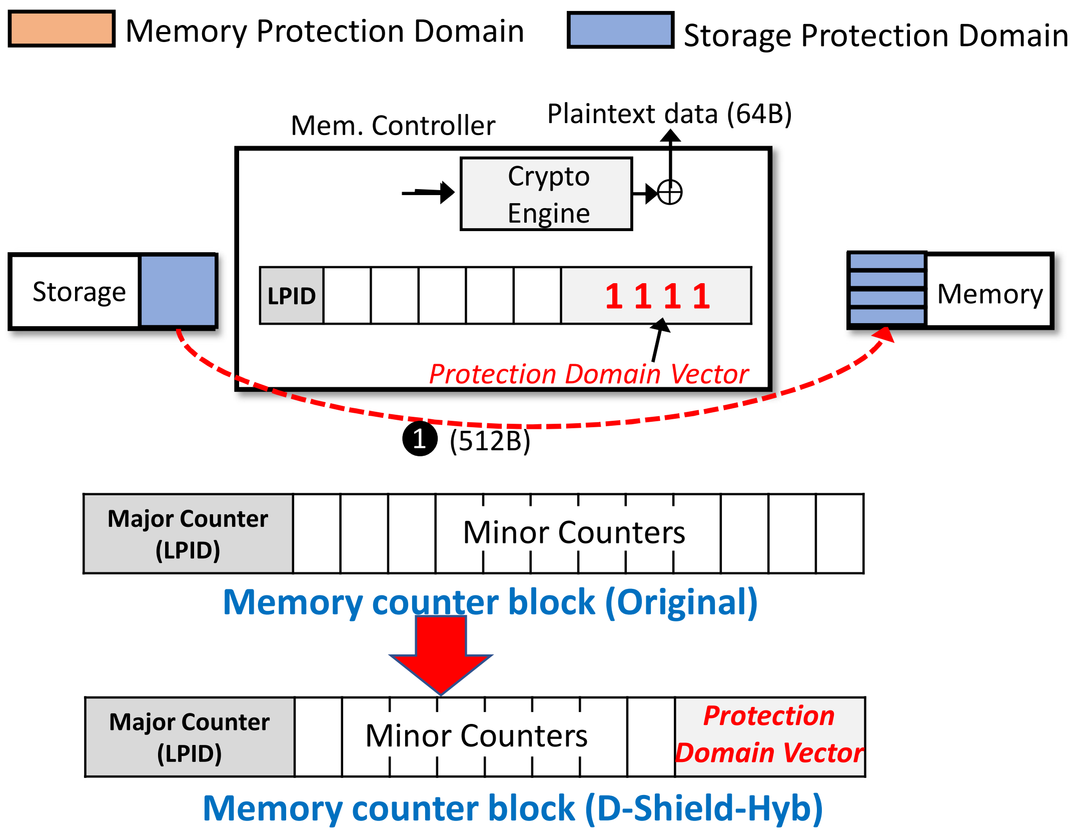
- Bookkeep the ownership of logic blocks **in memory**
- Track the security domain for transferred data block
- Performs only one iteration of decryption/encryption ***on-demand***

D-Shield-Hyb: Cross Domain Access Optimization



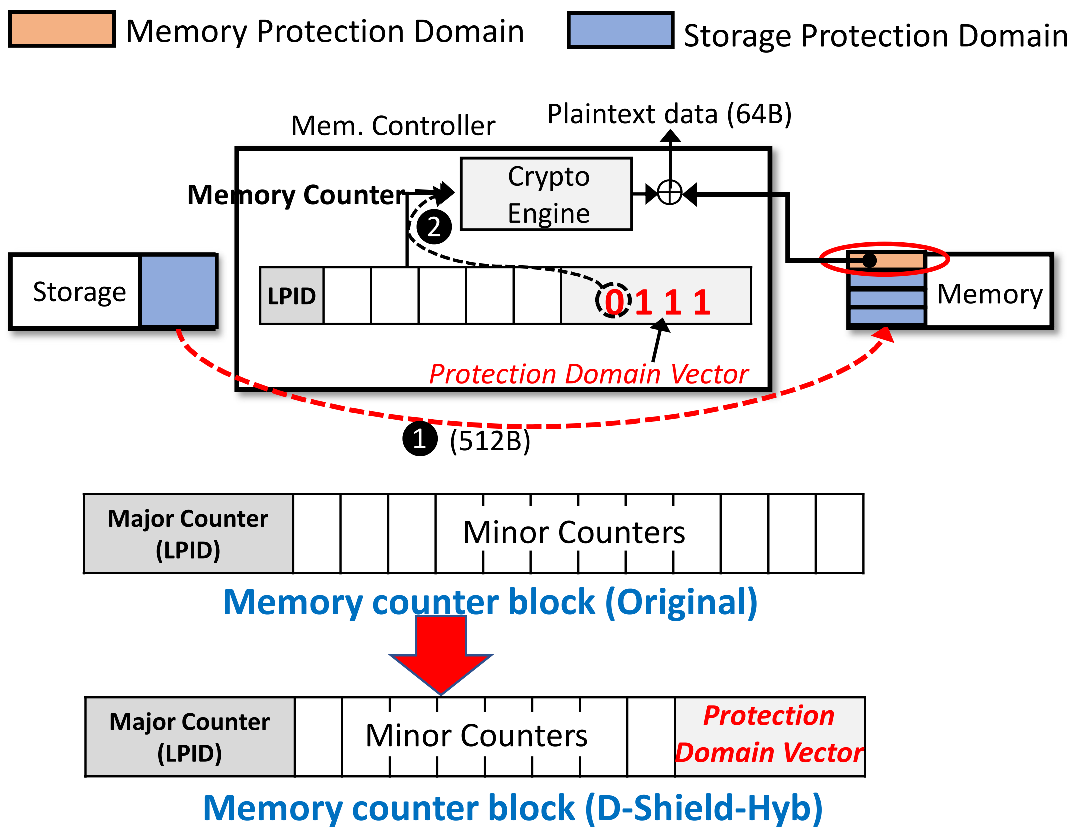
- Bookkeep the ownership of logic blocks **in memory**
- Track the security domain for transferred data block
- Performs only one iteration of decryption/encryption ***<u>on-demand</u>***

D-Shield-Hyb: Cross Domain Access Optimization



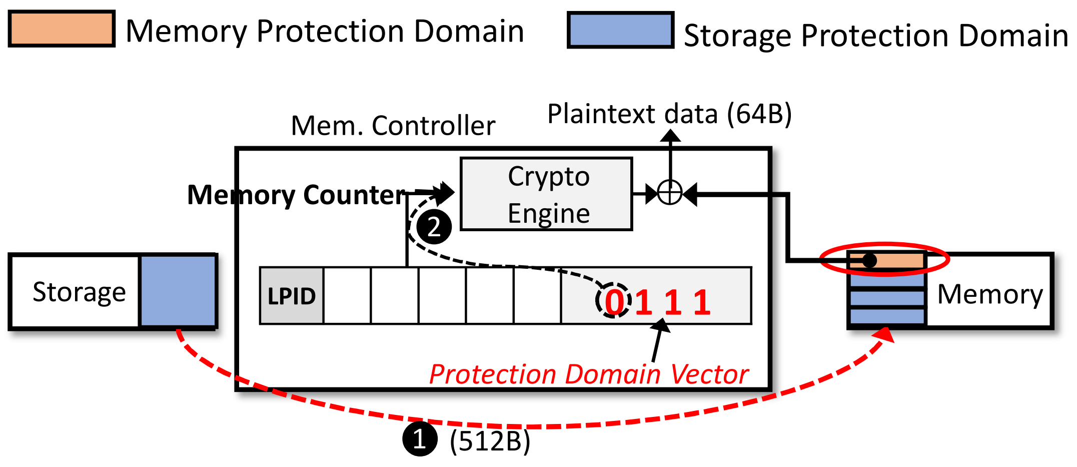
- Bookkeep the ownership of logic blocks **in memory**
- Track the security domain for transferred data block
- Performs only one iteration of decryption/encryption ***on-demand***

D-Shield-Hyb: Cross Domain Access Optimization



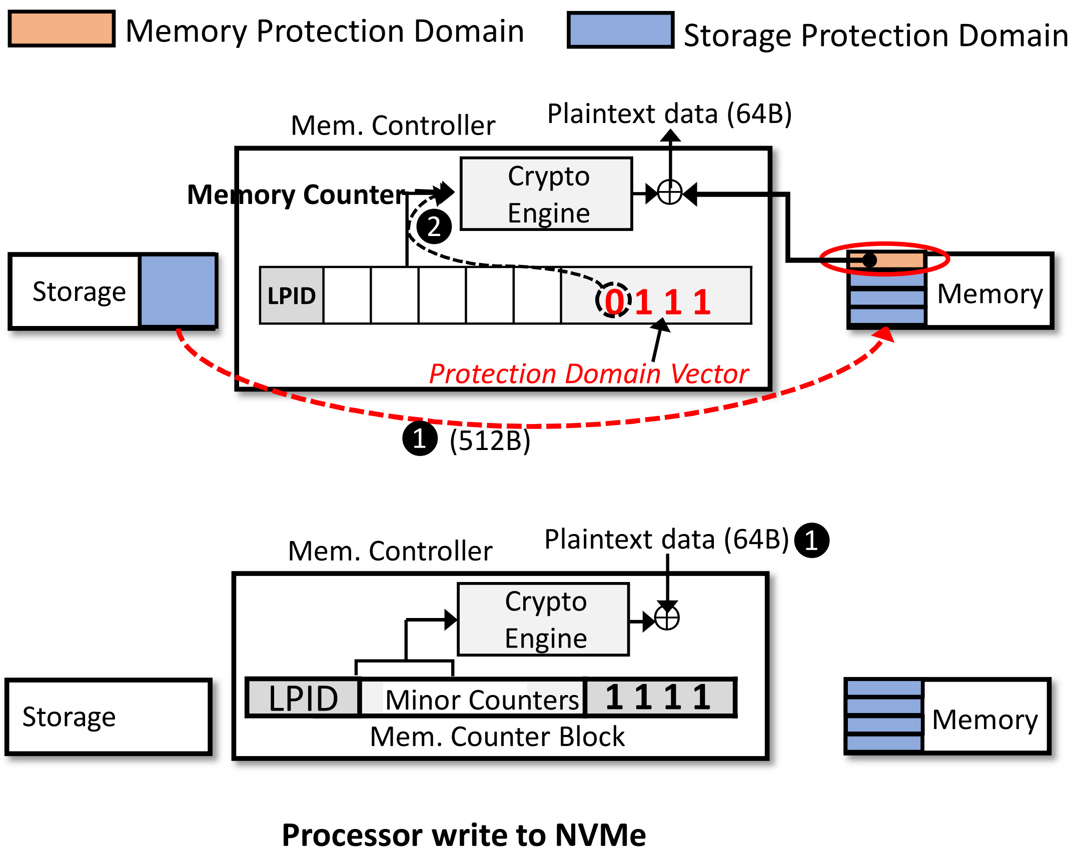
- Bookkeep the ownership of logic blocks **in memory**
- Track the security domain for transferred data block
- Performs only one iteration of decryption/encryption ***<u>on-demand</u>***

D-Shield-Hyb: Cross Domain Access Optimization



- Bookkeep the ownership of logic blocks **in memory**
- Track the security domain for transferred data block
- Performs only one iteration of decryption/encryption ***on-demand***

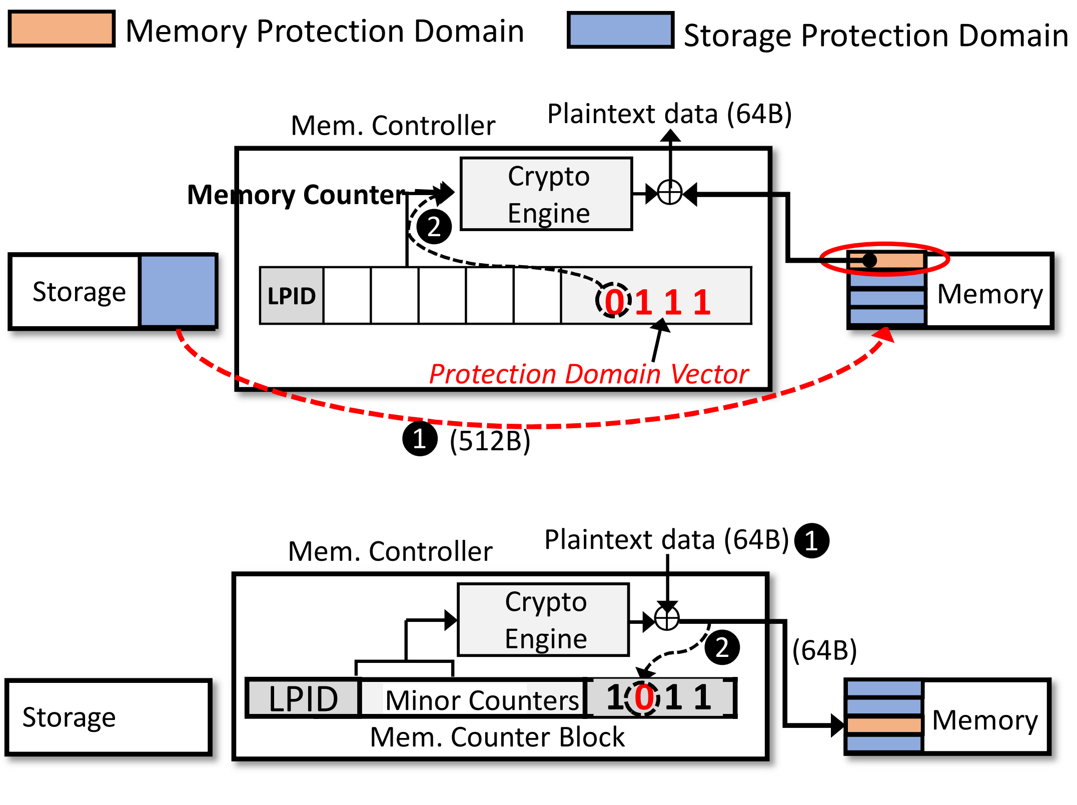
D-Shield-Hyb: Cross Domain Access Optimization



Processor write to NVMe

- Bookkeep the ownership of logic blocks **in memory**
- Track the security domain for transferred data block
- Performs only one iteration of decryption/encryption ***<u>on-demand</u>***

D-Shield-Hyb: Cross Domain Access Optimization

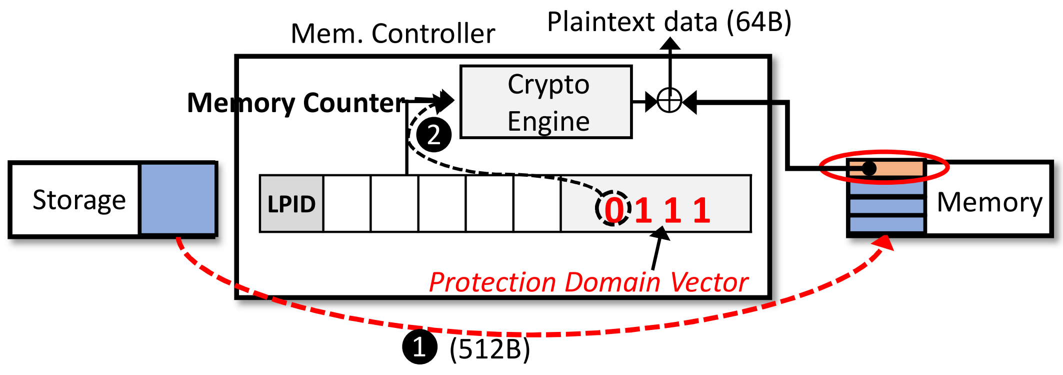


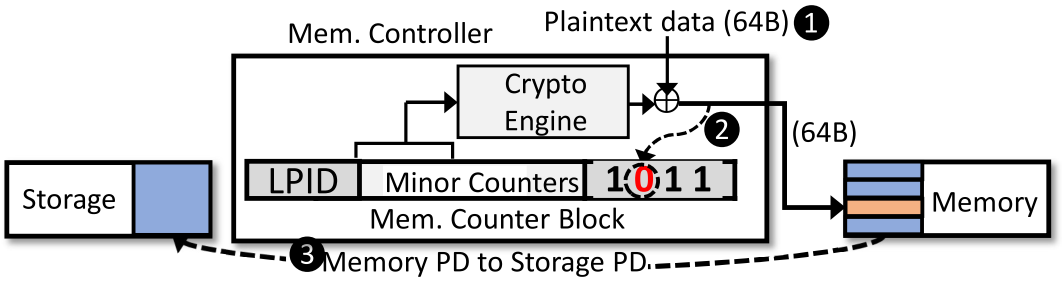
Processor write to NVMe

- Bookkeep the ownership of logic blocks **in memory**
- Track the security domain for transferred data block
- Performs only one iteration of decryption/encryption ***on-demand***

D-Shield-Hyb: Cross Domain Access Optimization

🟧 Memory Protection Domain 🟦 Storage Protection Domain





Processor write to NVMe

- Bookkeep the ownership of logic blocks **in memory**
- Track the security domain for transferred data block
- Performs only one iteration of decryption/encryption ***on-demand***

D-Shield-Pro: In-Memory Caching

- Storage metadata cache misses have high overheads

D-Shield-Pro: In-Memory Caching

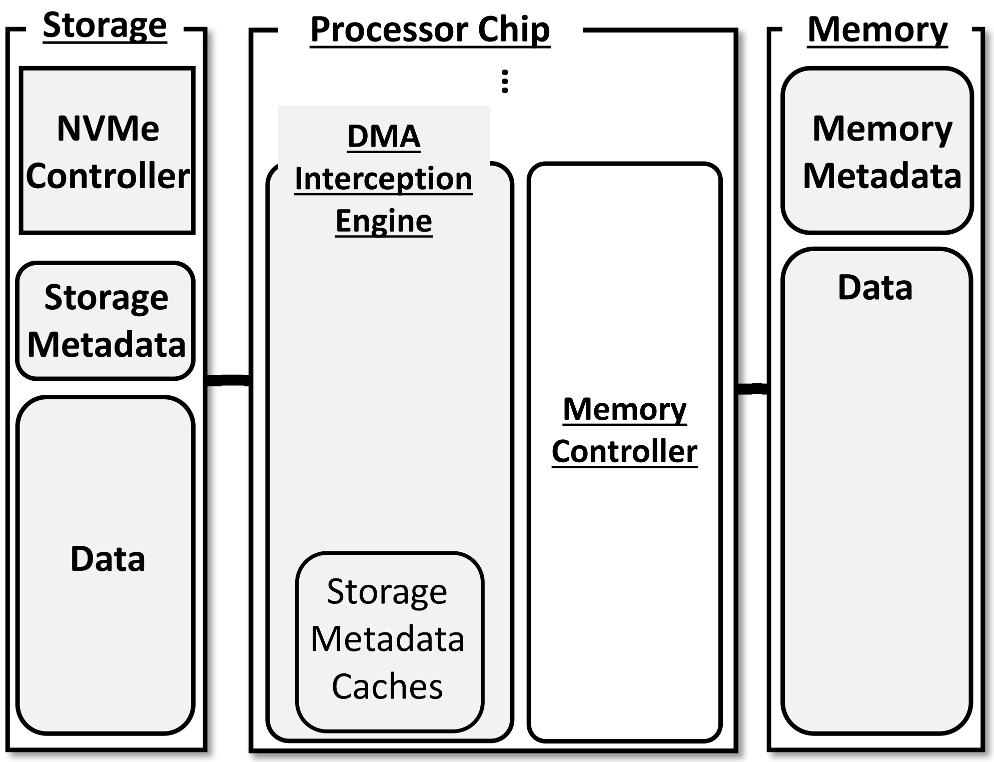
- Storage metadata cache misses have high overheads
- Miss in CMAC block is more expensive since it may require additional metadata access (i.e., for Merkle tree blocks)

D-Shield-Pro: In-Memory Caching

- Storage metadata cache misses have high overheads
- Miss in CMAC block is more expensive since it may require additional metadata access (i.e., for Merkle tree blocks)
- **Idea**: *In-memory CMAC block caching* to increase the CMAC block hit ratio

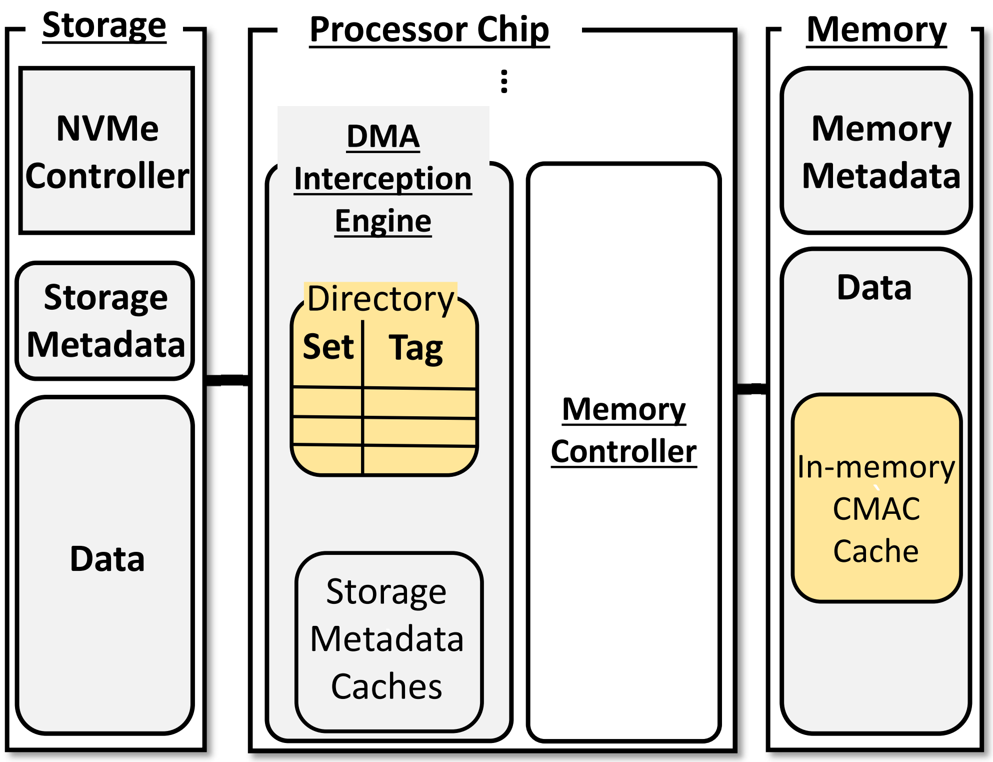
D-Shield-Pro: In-Memory Caching

- Storage metadata cache misses have high overheads
- Miss in CMAC block is more expensive since it may require additional metadata access (i.e., for Merkle tree blocks)
- **Idea**: *<u>In-memory CMAC block caching</u>* to increase the CMAC block hit ratio



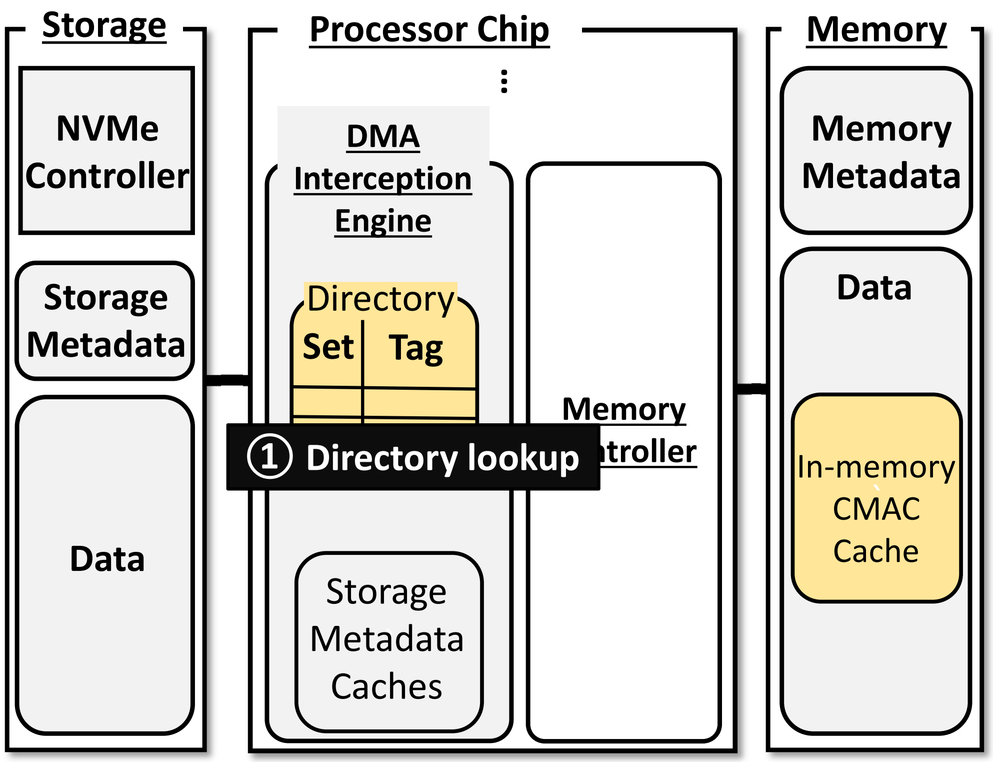
D-Shield-Pro: In-Memory Caching

- Storage metadata cache misses have high overheads
- Miss in CMAC block is more expensive since it may require additional metadata access (i.e., for Merkle tree blocks)
- **Idea**: *In-memory CMAC block caching* to increase the CMAC block hit ratio



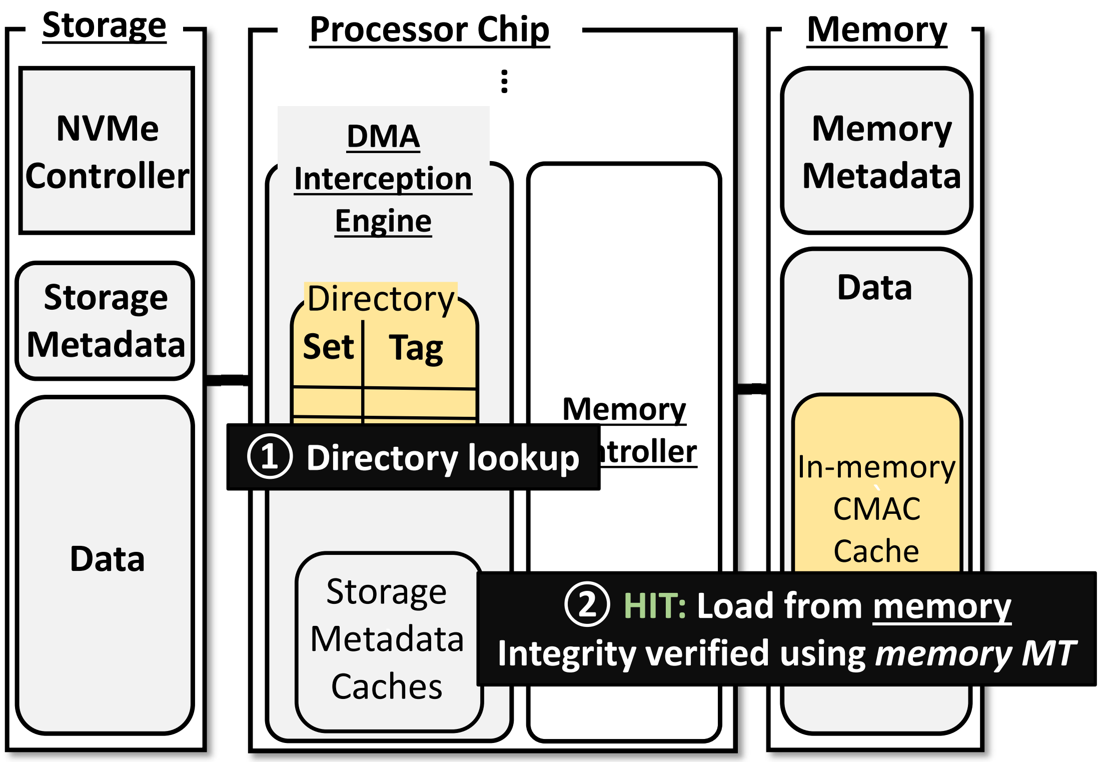
D-Shield-Pro: In-Memory Caching

- Storage metadata cache misses have high overheads
- Miss in CMAC block is more expensive since it may require additional metadata access (i.e., for Merkle tree blocks)
- **Idea**: *In-memory CMAC block caching* to increase the CMAC block hit ratio



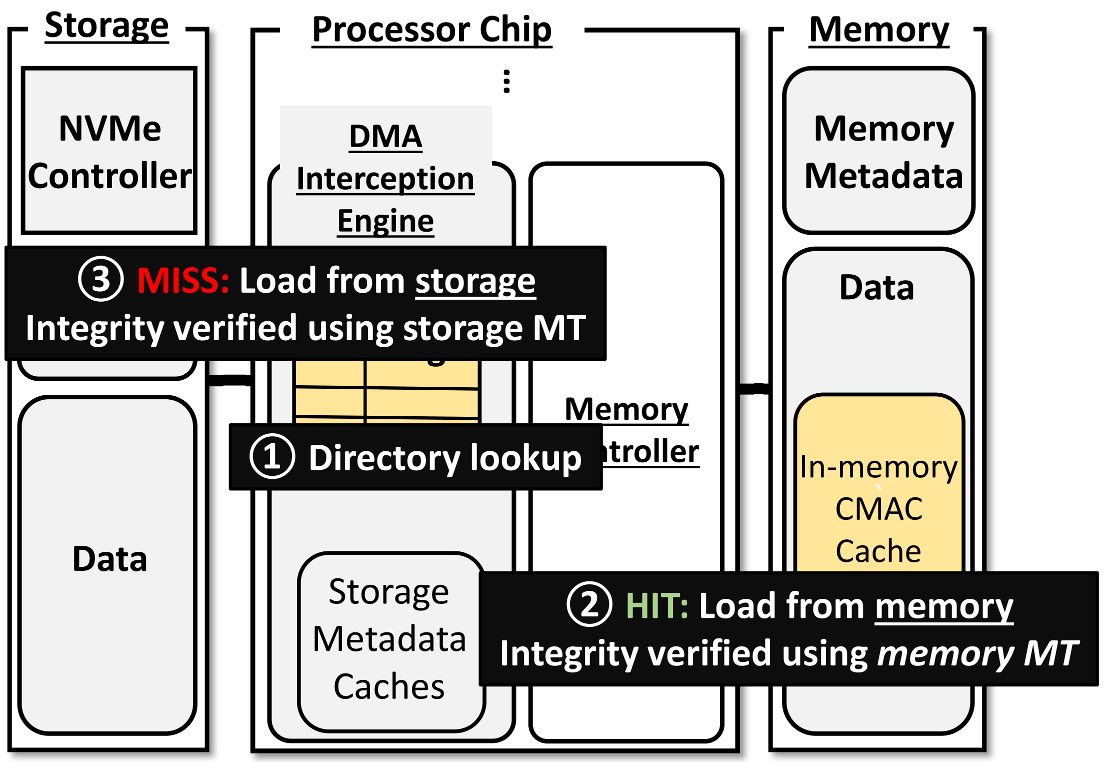
D-Shield-Pro: In-Memory Caching

- Storage metadata cache misses have high overheads
- Miss in CMAC block is more expensive since it may require additional metadata access (i.e., for Merkle tree blocks)
- **Idea**: *In-memory CMAC block caching* to increase the CMAC block hit ratio



D-Shield-Pro: In-Memory Caching

- Storage metadata cache misses have high overheads
- Miss in CMAC block is more expensive since it may require additional metadata access (i.e., for Merkle tree blocks)
- **Idea**: *In-memory CMAC block caching* to increase the CMAC block hit ratio



UCF

Experimental Setup

- **Simulator:** Gem5-based full-system simulation (SimpleSSD)
- **OS:** Ubuntu 18.04; **Kernel:** Linux 4.9

Hardware	***Configurations***
Processor	4-core, 3.0 GHz in-order, x86
L1 I/D-cache	Private, 64KB, 4-way
L2 cache	Shared, 16MB, 16-way
Main memory	DDR4 based 16GB
Cryptographic Engine	
Encryption/Hash operation (64B)	40 cycles
DMA Interception Engine	
Metadata cache	256KB 8-way each
Hash operation (512B)	320 cycles
NVMe Disk	
Capacity	512GB
Cell model	Z-NAND based MLC PCM
Avg. random access latency(µS)	READ: 10.5, WRITE: 9

Evaluation Methodology

❖ **Workloads:**

- **I/O intensive applications:** Flexible I/O
- **Server-class applications:** database, document storage system
- **Graph algorithms:** YCSB suite (twitter follow network)

Evaluation Methodology

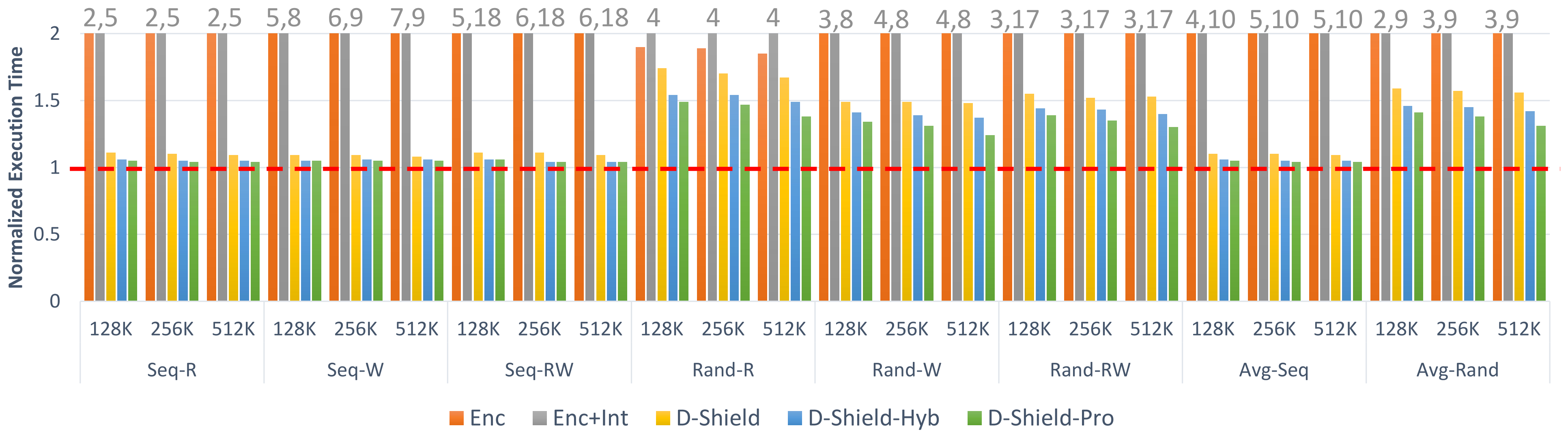
❖ **Workloads:**

- **I/O intensive applications:** Flexible I/O
- **Server-class applications:** database, document storage system
- **Graph algorithms:** YCSB suite (twitter follow network)

❖ **Baselines (All variants include *secure memory*):**

- **Insecure:** Default NVMe storage system without security mechanism
- **Enc:** dm-crypt-based encryption for NVMe storage system
- **Enc+Int:** dm-crypt with dm-integrity for NVMe disk encryption and integrity checking

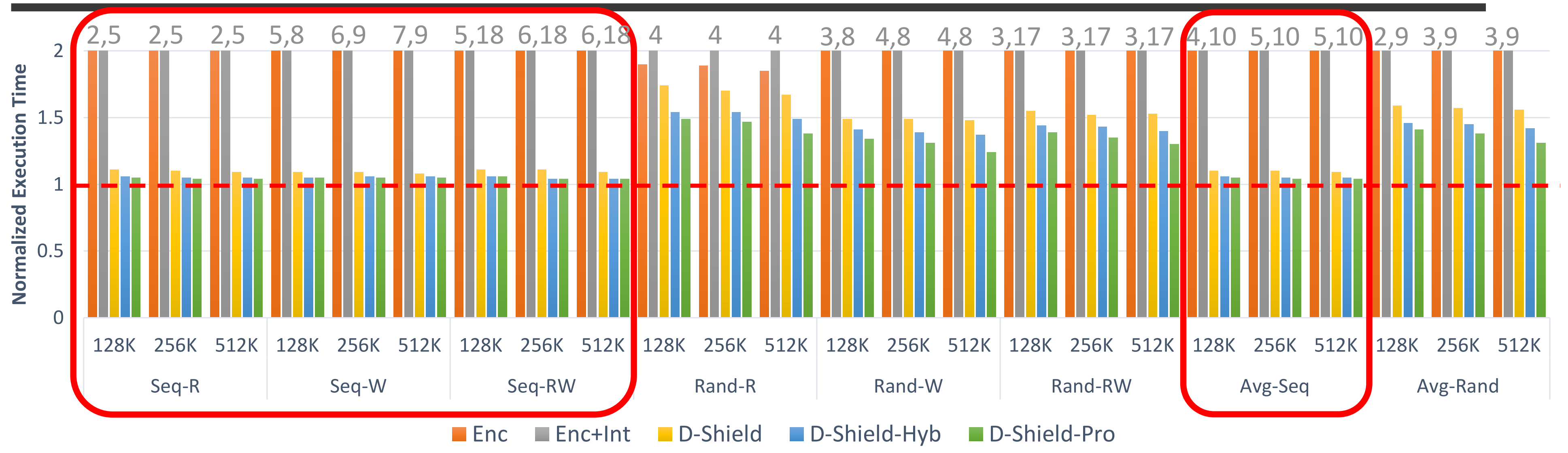
Evaluations: D-Shield Performance



# of transactions (from left to right): **128K, 256K, 512K**

Runtime (normalized to Insecure) of FIO benchmark

Evaluations: D-Shield Performance



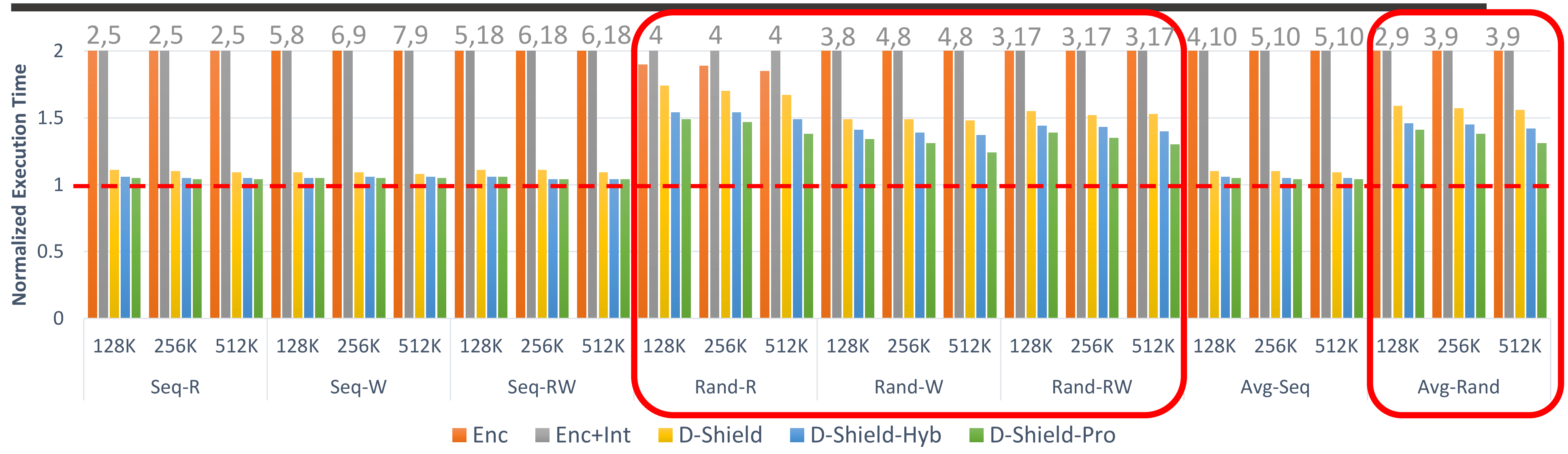
# of transactions (from left to right): **128K, 256K, 512K**

Runtime (normalized to Insecure) of FIO benchmark

Sequential workloads: 4.4% overhead compared to *Insecure* (Avg)

4.1x less compared to **Enc**, **10x less** compared to **Enc+Int**

Evaluations: D-Shield Performance



# of transactions (from left to right): **128K, 256K, 512K**

Runtime (normalized to Insecure) of FIO benchmark

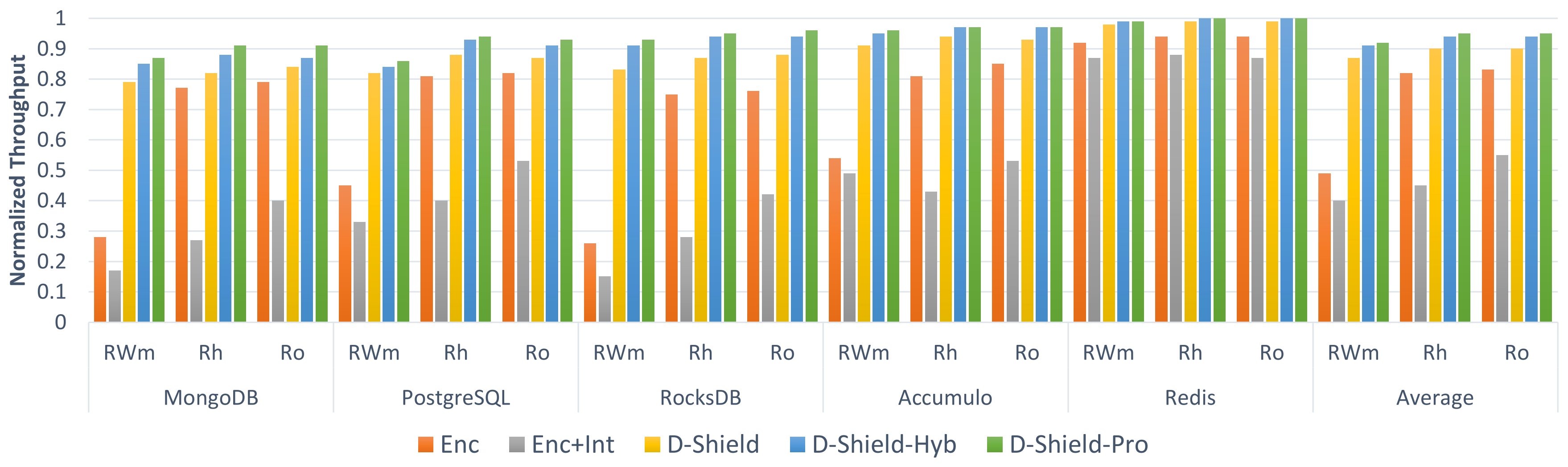
Sequential workloads: 4.4% overhead compared to *Insecure* (Avg)

4.1x less compared to **Enc**, **10x less** compared to **Enc+Int**

Random workloads: 39% overhead compared to *Insecure* (Avg)

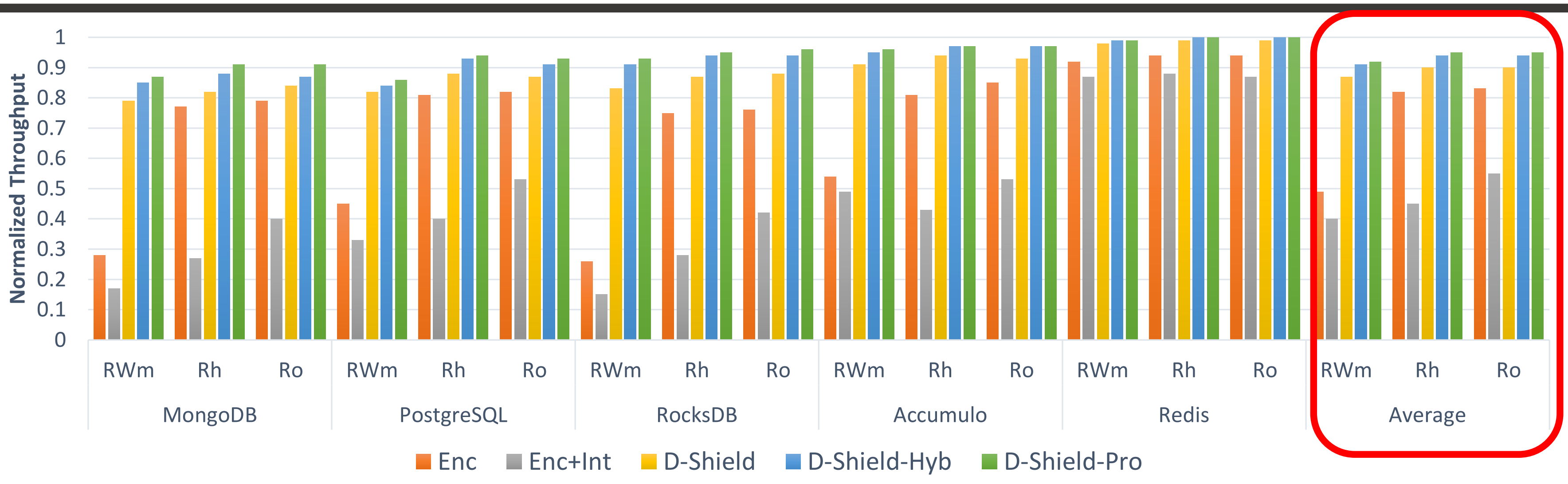
2.05x less compared to **Enc**, **7x less** compared to **Enc+Int**

Evaluations: D-Shield Performance



Throughput of D-Shield on real-world server applications

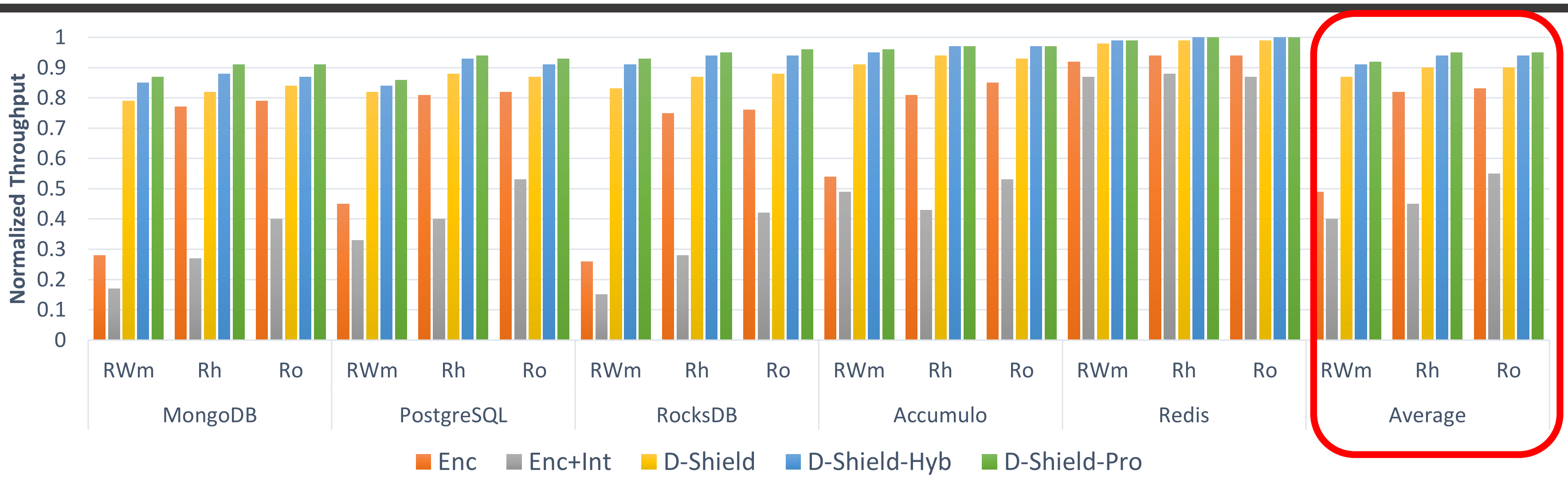
Evaluations: D-Shield Performance



Throughput of D-Shield on real-world server applications

D-Shield can maintain 94% (Avg) throughput compared to *Insecure*

Evaluations: D-Shield Performance



Throughput of D-Shield on real-world server applications

D-Shield can maintain 94% (Avg) throughput compared to *Insecure*

24% higher compared to <u>**Enc**</u> and **49%** higher compared to <u>**Enc+Int**</u>

Evaluations: Hardware Overhead

- NVMe storage overhead: **3.14%** for **Security Metadata**
- On-chip storage overhead: **2x256KB** for **Storage Metadata Caches**

  552 Bytes for **Region Table**
- In-memory storage: **128MB** for **In-memory Cache** (D-Shield-Pro only)

Evaluations: Hardware Overhead

• NVMe storage overhead:	**3.14%** for **Security Metadata**
• On-chip storage overhead:	**2x256KB** for **Storage Metadata Caches**
	552 Bytes for **Region Table**
• In-memory storage:	**128MB** for **In-memory Cache** (D-Shield-Pro only)

- D-Shield on-chip logic:
  - Implemented using Verilog
  - Synthesized with Synopsis DC with 45nm

Evaluations: Hardware Overhead

- NVMe storage overhead: **3.14%** for **Security Metadata**
- On-chip storage overhead: **2x256KB** for **Storage Metadata Caches**

  552 Bytes for **Region Table**
- In-memory storage: **128MB** for **In-memory Cache** (D-Shield-Pro only)
- D-Shield on-chip logic:
  - Implemented using Verilog
  - Synthesized with Synopsis DC with 45nm

Module	**Area (mm^2)**
Logic Block Buffer	0.23
Security Control Logic	0.08

UCF

Conclusion

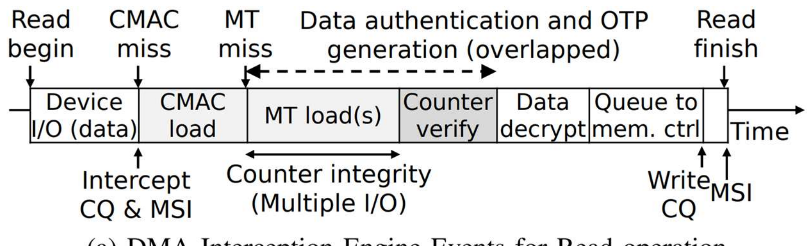
- **Existing storage protection offers limited security and impose high overhead**
- **D-Shield offers architectural framework for processor-side storage security**
- **D-Shield-Hybrid optimizes cross-domain data transfer substantially**
- **D-Shield-Pro reduces metadata overheads through in-memory caching**
- **Modest performance overheads in real-world workloads**
  - While providing state-of-the-art data security

More on Paper

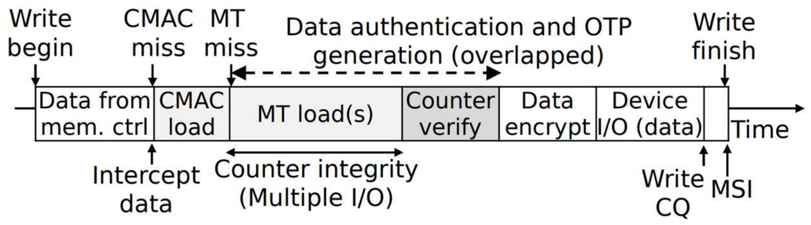
- **Architectural design space explorations**

More on Paper

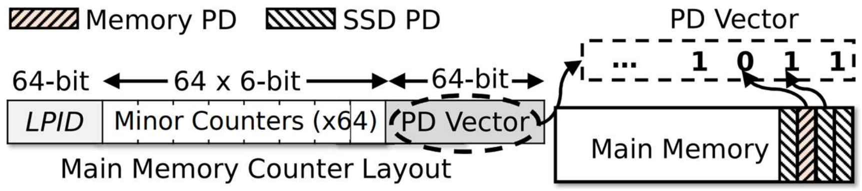
- **Architectural design space explorations**
- **Additional details on D-Shield designs:**
  - Complete R/W paths
  - Metadata arrangements and maintenance



(a) DMA Interception Engine Events for Read operation



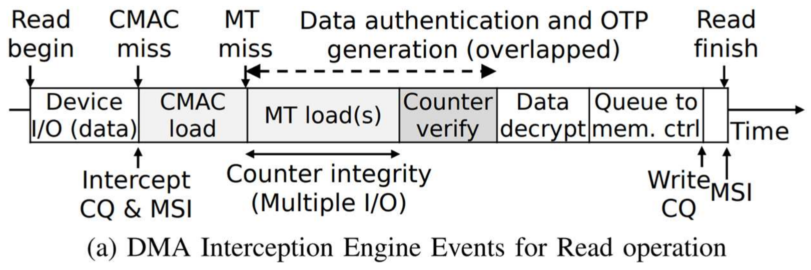
NVMe Read/Write path in D-Shield



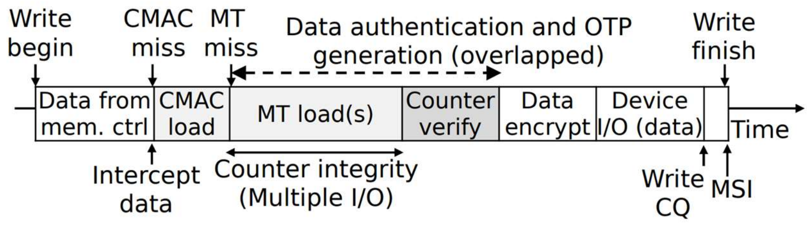
D-Shield-Hybrid metadata storage

More on Paper

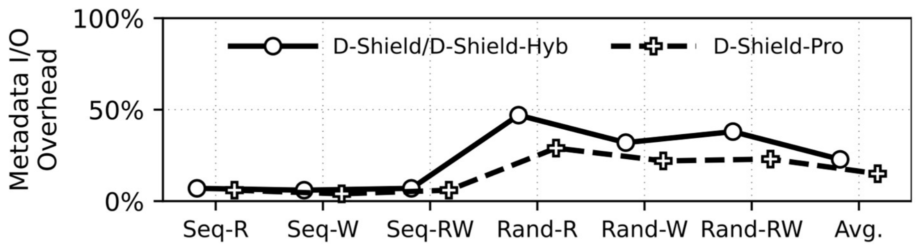
- **Architectural design space explorations**
- **Additional details on D-Shield designs:**
  - Complete R/W paths
  - Metadata arrangements and maintenance
- **D-Shield overhead analysis:**
  - Additional I/O overheads
  - Logic and storage overheads



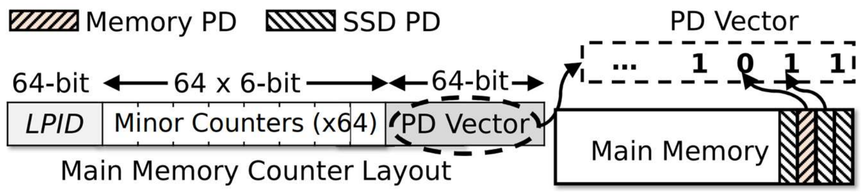
(a) DMA Interception Engine Events for Read operation



NVMe Read/Write path in D-Shield



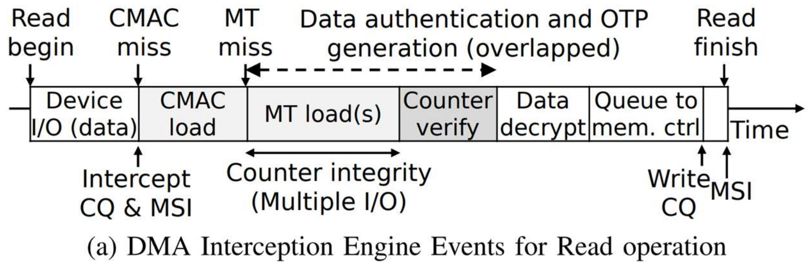
D-Shield I/O overheads



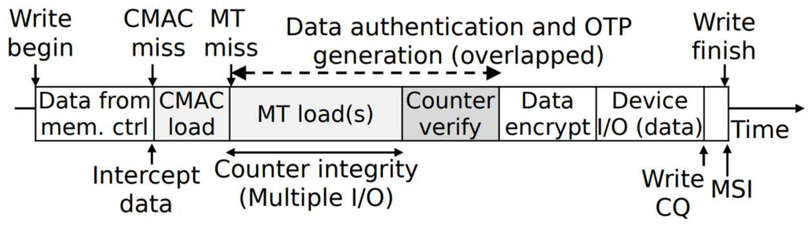
D-Shield-Hybrid metadata storage

More on Paper

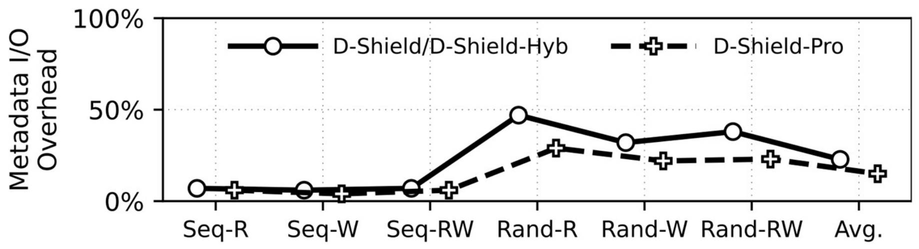
- **Architectural design space explorations**
- **Additional details on D-Shield designs:**
  - Complete R/W paths
  - Metadata arrangements and maintenance
- **D-Shield overhead analysis:**
  - Additional I/O overheads
  - Logic and storage overheads
- **Sensitivity analysis of D-Shield schemes**
- **And more…**



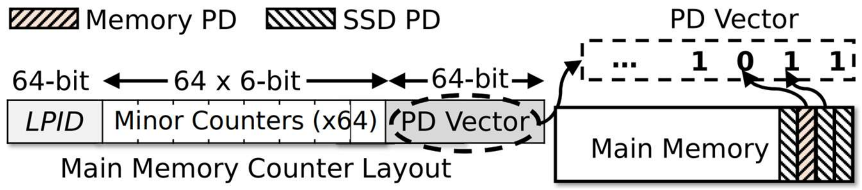
(a) DMA Interception Engine Events for Read operation



NVMe Read/Write path in D-Shield



D-Shield I/O overheads



D-Shield-Hybrid metadata storage

Thanks! Questions?

Md Hafizul Islam Chowdhuryy
CASR Lab (https://casr.ece.ucf.edu)
Email: reyad@knights.ucf.edu