Computer Architecture Lab.

ShieldStore: Shielded In-memory Key-value Storage with SGX

Taehoon Kim, Joongun Park, Jaewook Woo, Seungheun Jeon, and Jaehyuk Huh

EuroSys 2019

KAIST School of Computing

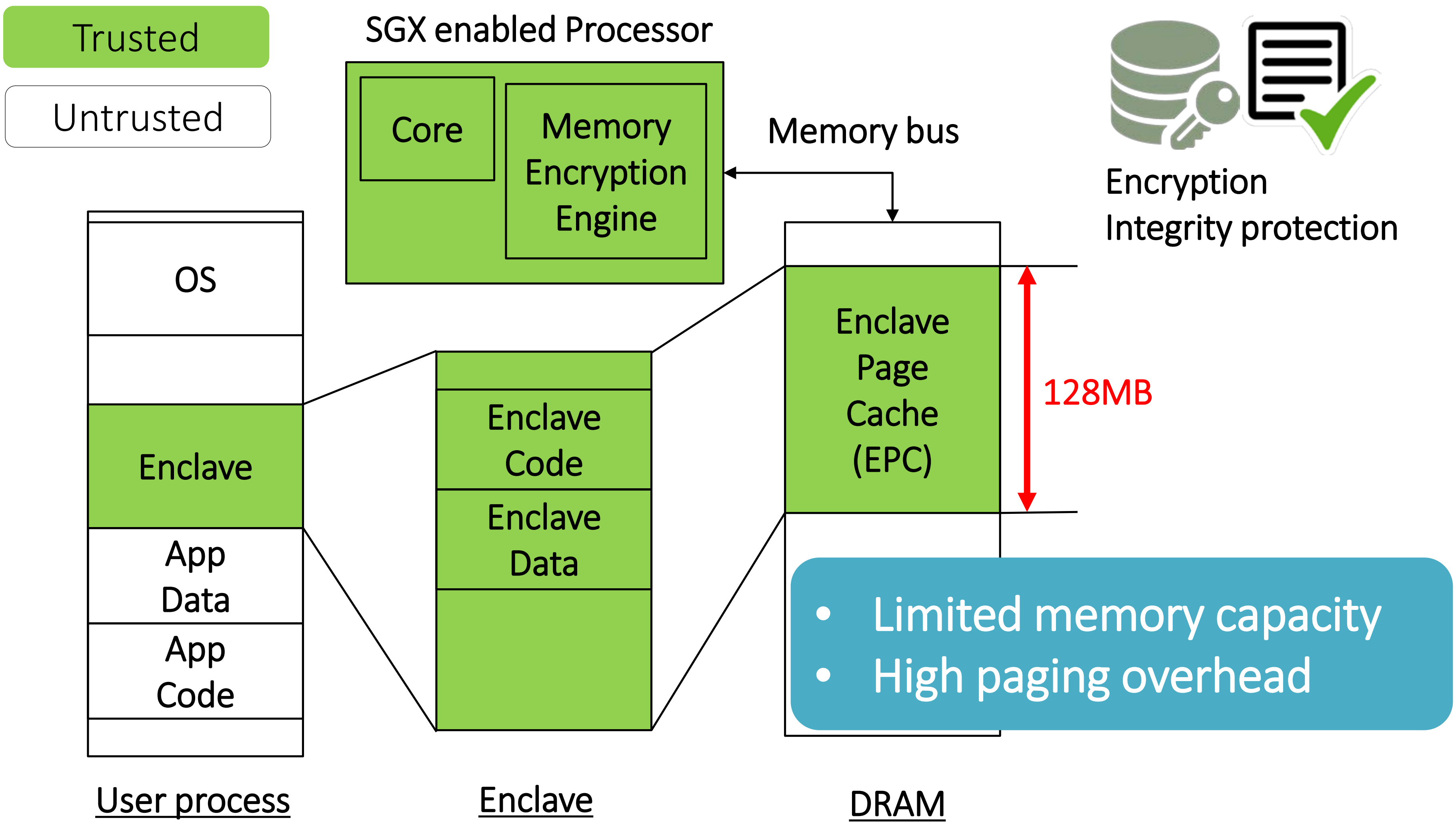
Trusted Key-value Stores

- User data is exposed to malicious attackers in clouds

- Hardware-based security supports
  - Provide *trusted execution environment* for remote server

Intel SGX (Software Guard Extensions)

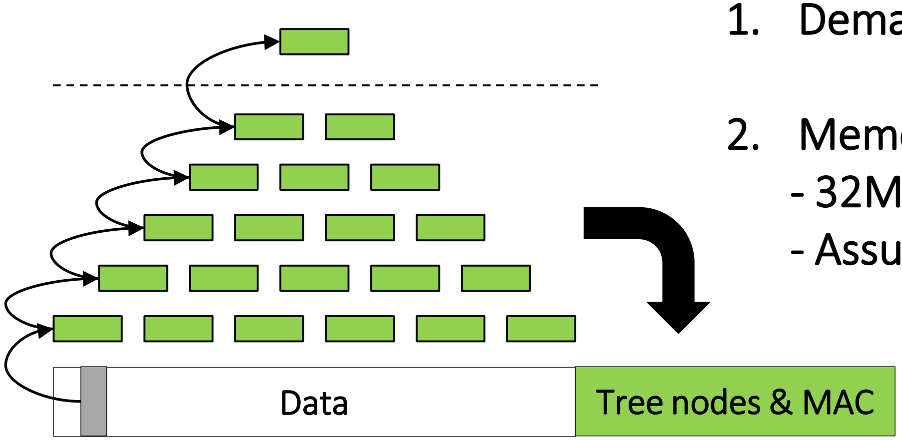
- Support *trusted execution environment* by *enclave* in a process



HW Limitation of EPC

- Several studies assume large protected memory
  – Vault [1], EnclavDB [2]

- With Large protected memory
  – High performance overhead for verifying integrity (Merkle Tree)



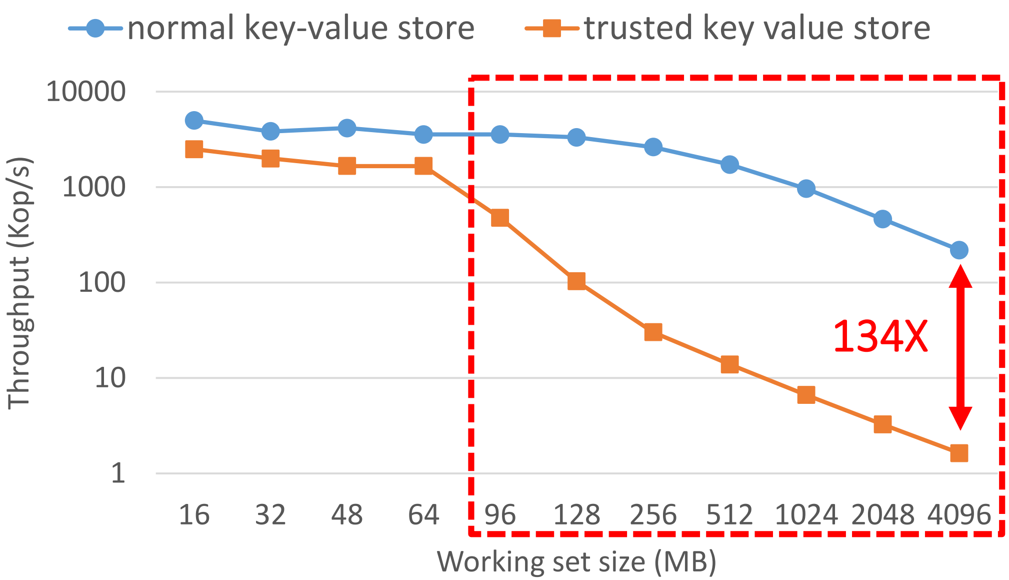
1. Demand high memory bandwidth

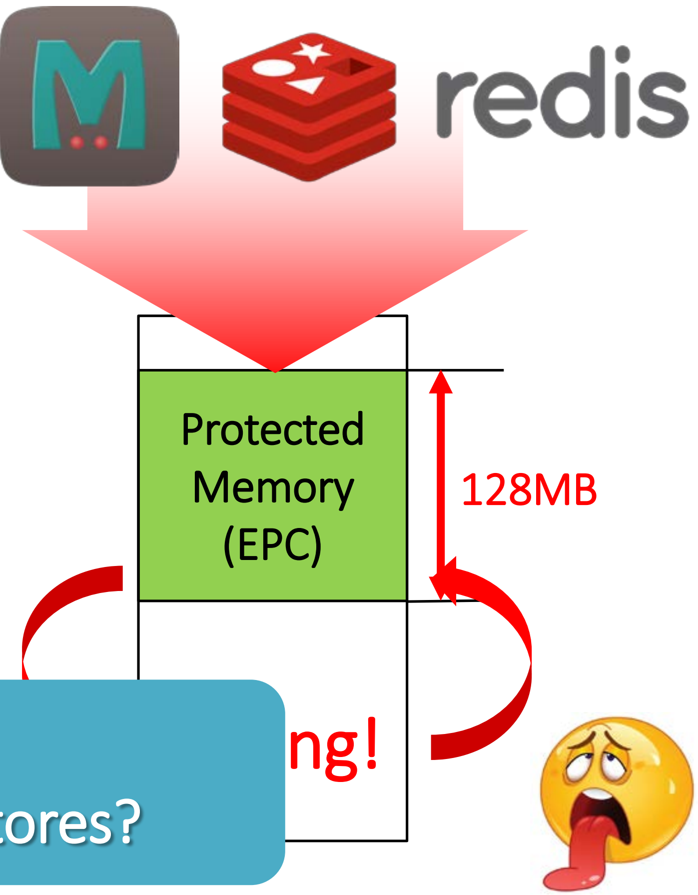
2. Memory capacity overhead for tree nodes & MAC
   - 32MB on 128MB preserved memory [3]
   - Assume 8GB total memory -> 2GB meta-data

[1] Taassori, et al. VAULT: Reducing Paging Overheads in SGX with Efficient Integrity Verification Structures [ASPLOS' 18]
[2] Reibe, et al. EnclaveDB: A Secure Database using SGX [S&P' 18]
[3] Shay Gueron A Memory Encryption Engine Suitable for General Purpose Processors [ePrint' 16]

Trusted Key-value Stores with SGX

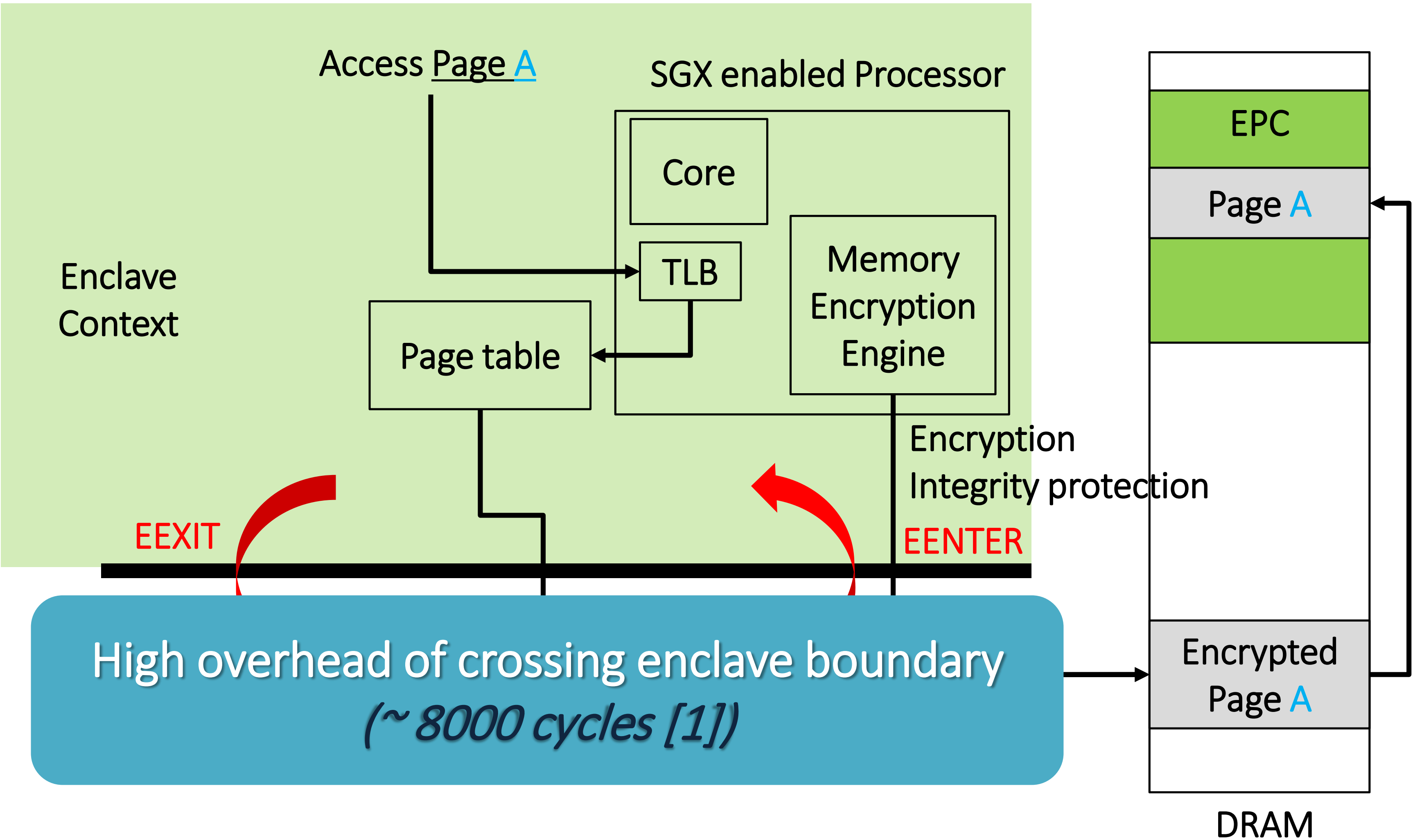
- Protected memory is limited to 128MB
  - Application can use about 92MB





How can we have
efficient trusted key value stores?

Paging Mechanism of SGX



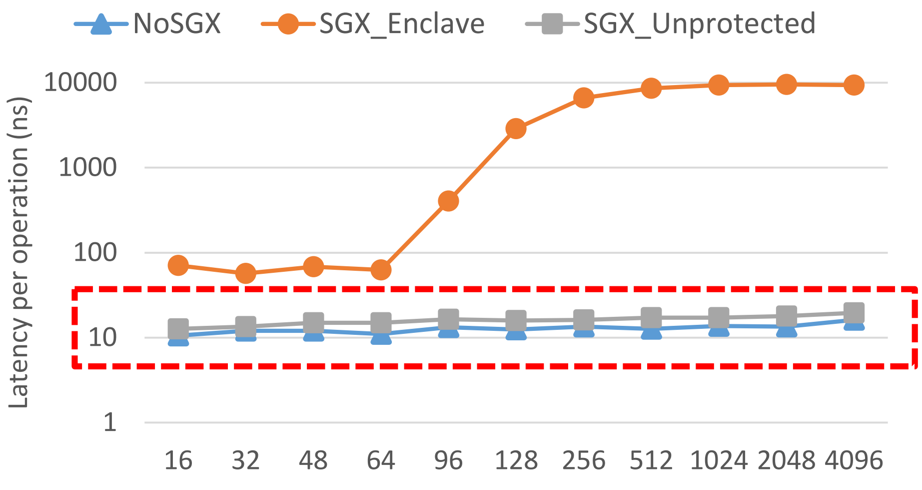
High overhead of crossing enclave boundary

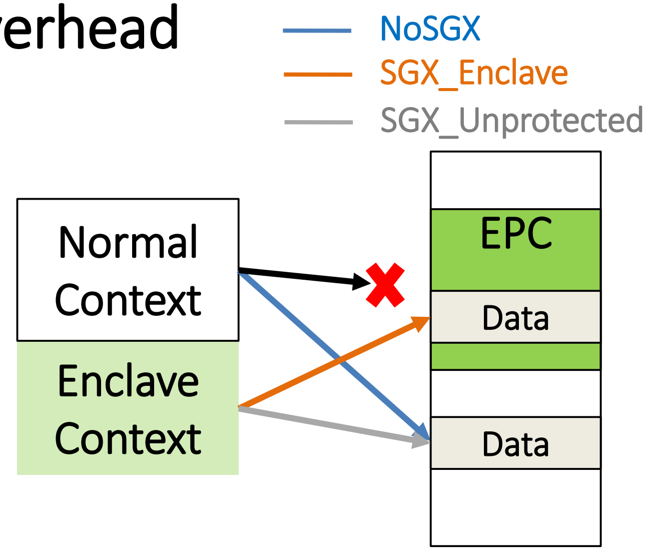
(~ 8000 cycles [1])

[1] Orenbach, et al. Eleos: ExitLess OS Services for SGX Enclaves [EuroSys' 17]

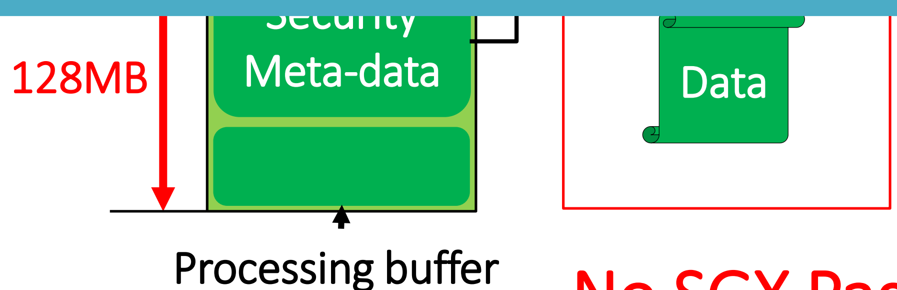
Observations

- Accessing untrusted memory incurs low overhead



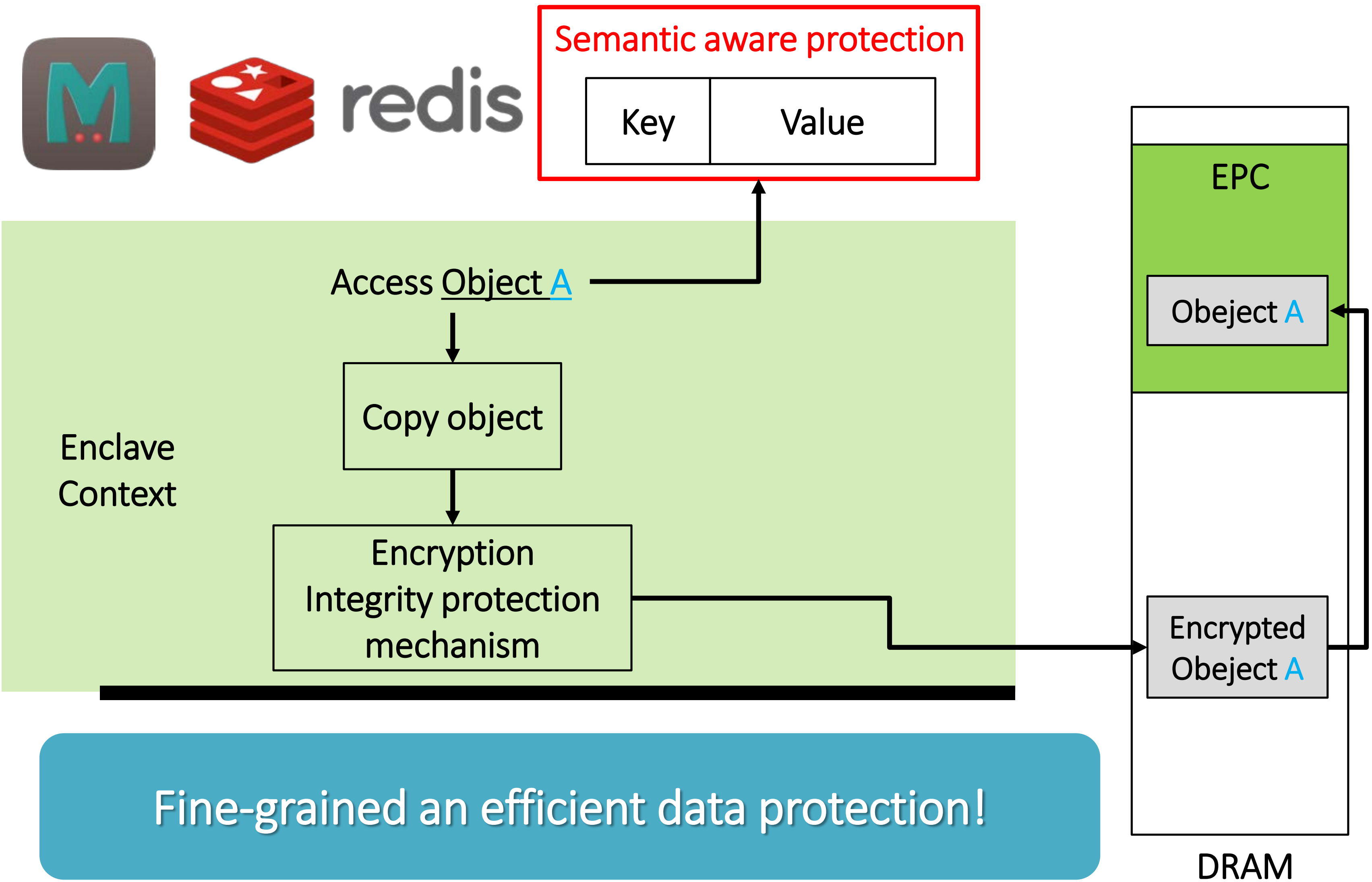


- *Reduce sgx-paging!*
  - Use protected memory as a secure processing buffer



No SGX Paging!

Proposed Design: Semantic Aware Protection



Fine-grained an efficient data protection!

Threat Model

- ShieldStore protects *confidentiality* and *integrity* of key/values

- Trusted Computing Base (TCB) of ShieldStore
  - SGX enabled Processor chip
  - Code & data in *enclave*

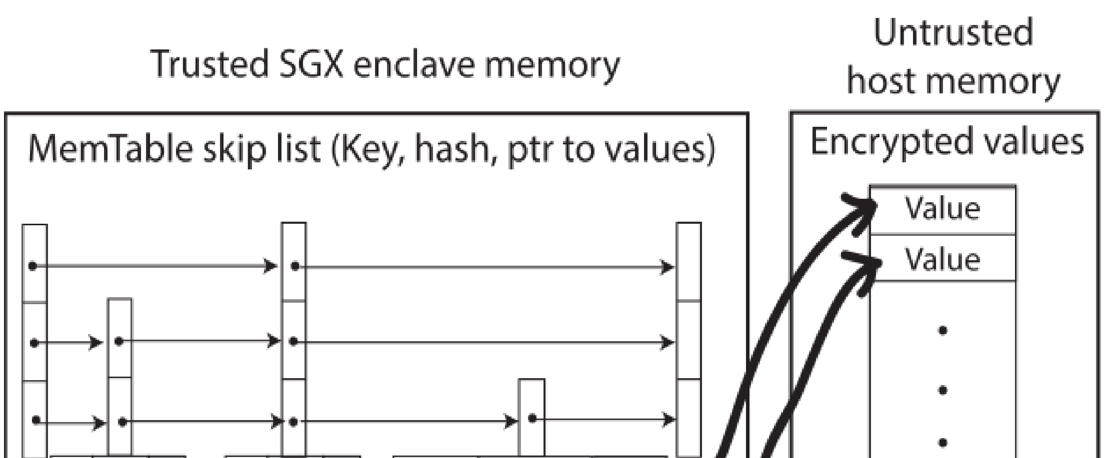
- Out of scope
  - Side channel attacks (ex. Foreshadow, controlled channel attacks)
  - Availability attacks

SPEICHER [2]

- Concurrent work (published in *FAST'19*)
- LSM based trusted key-value store

+ Processing range queries

- Enclave memory overhead (keep all the keys & hashs in EPC)

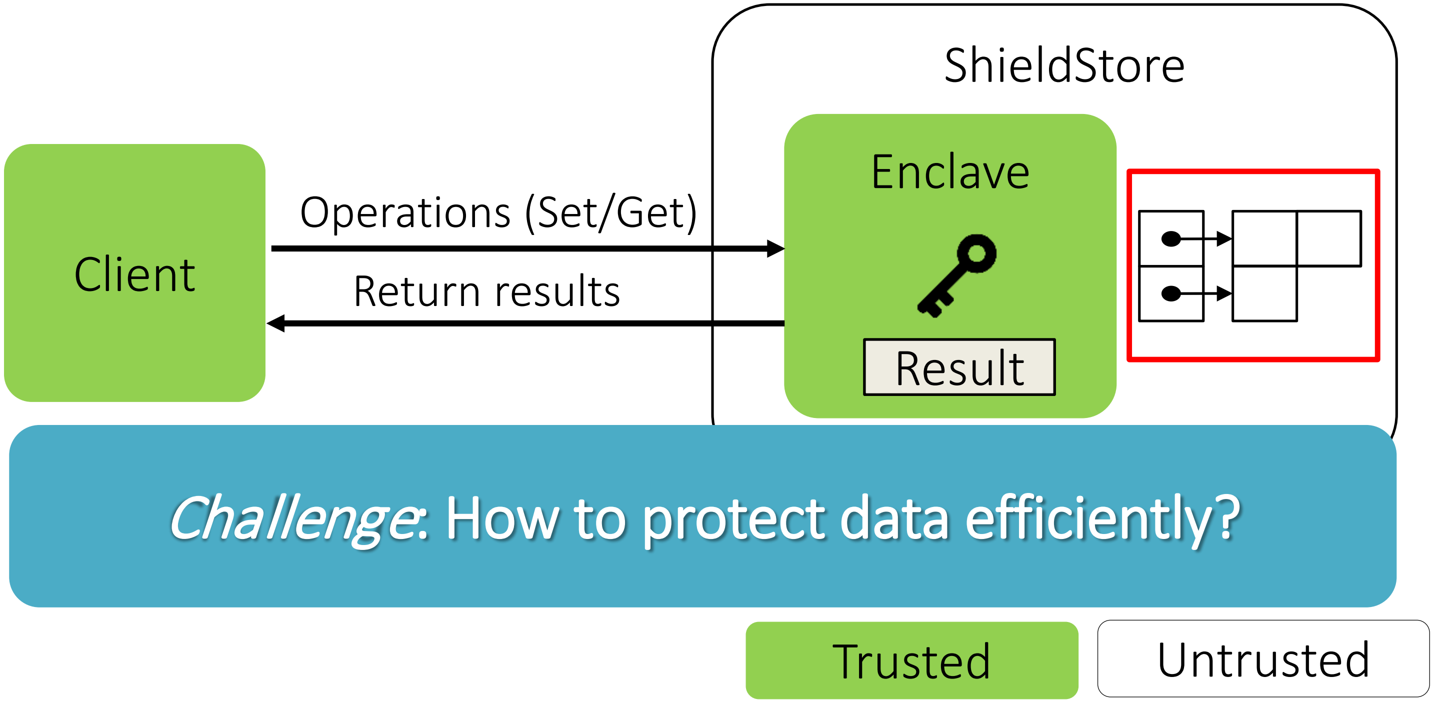


ShieldStore: Efficiently protect large working set data exploiting application specific structures

[2] Bailleu, et al. SPEICHER: Securing LSM-based Key-Value Stores using Shielded Execution [FAST' 19]

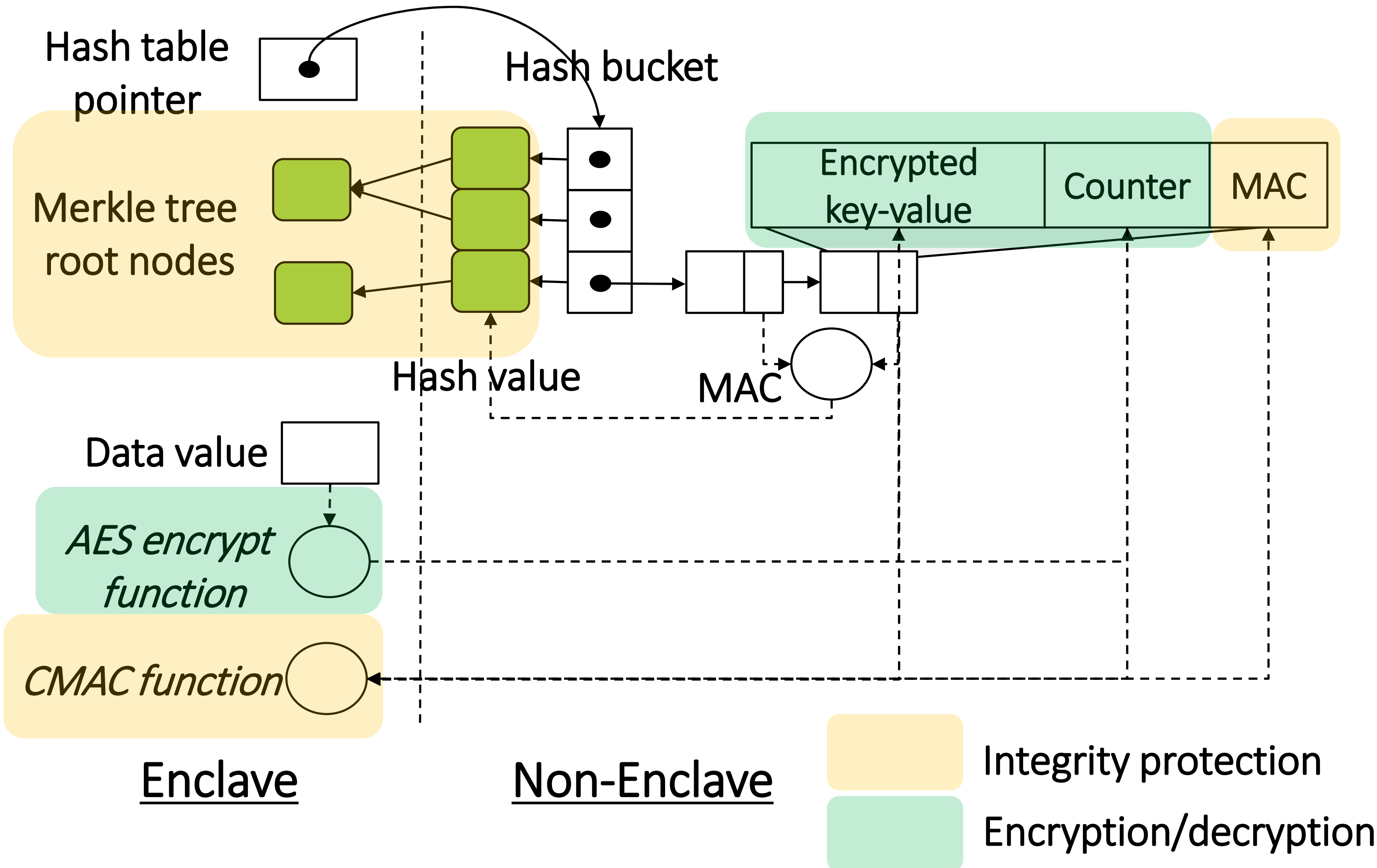
Overall Design of ShieldStore

- Maintain small secure meta-data in trusted memory region
- Store main data structure on untrusted memory region
  – With encrypted and integrity-protected key-value entries



Challenge: How to protect data efficiently?

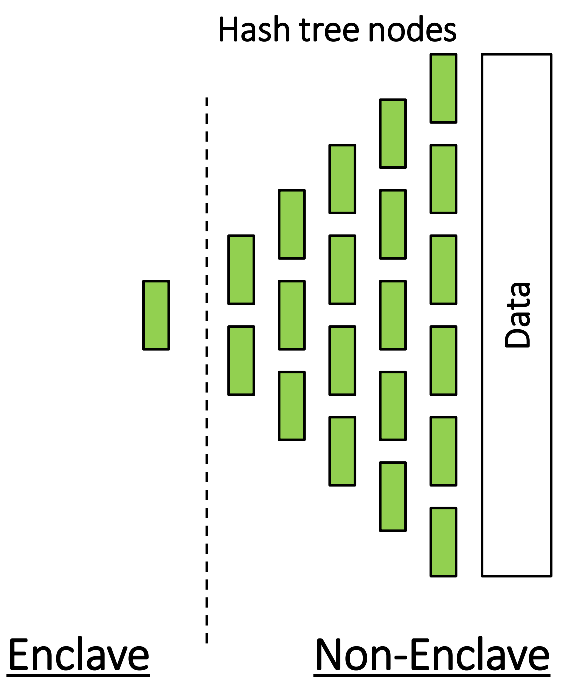
How To Protect Data?



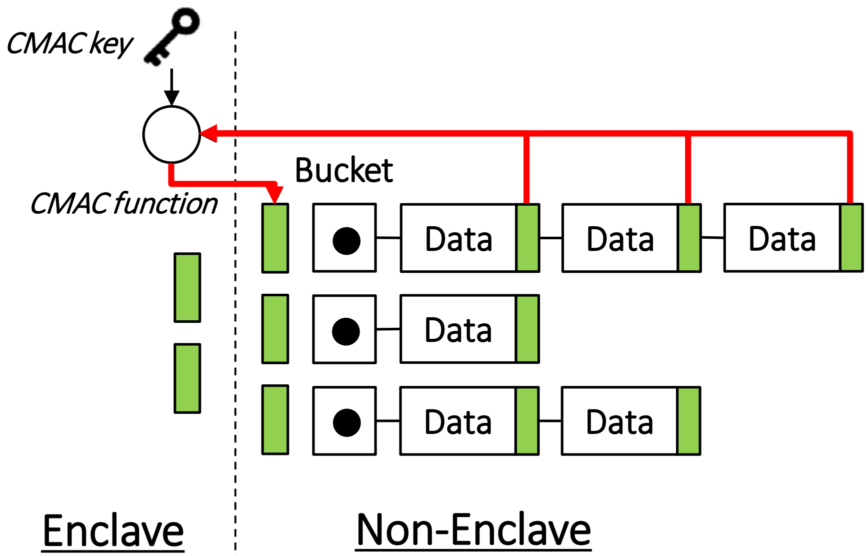
Integrity Protection

- ShieldStore employs *Merkle Tree* mechanism
  - Exploits the hash-based index structure to verify integrity efficiently

Traditional Merkle Tree



Data structure aware Merkle Tree

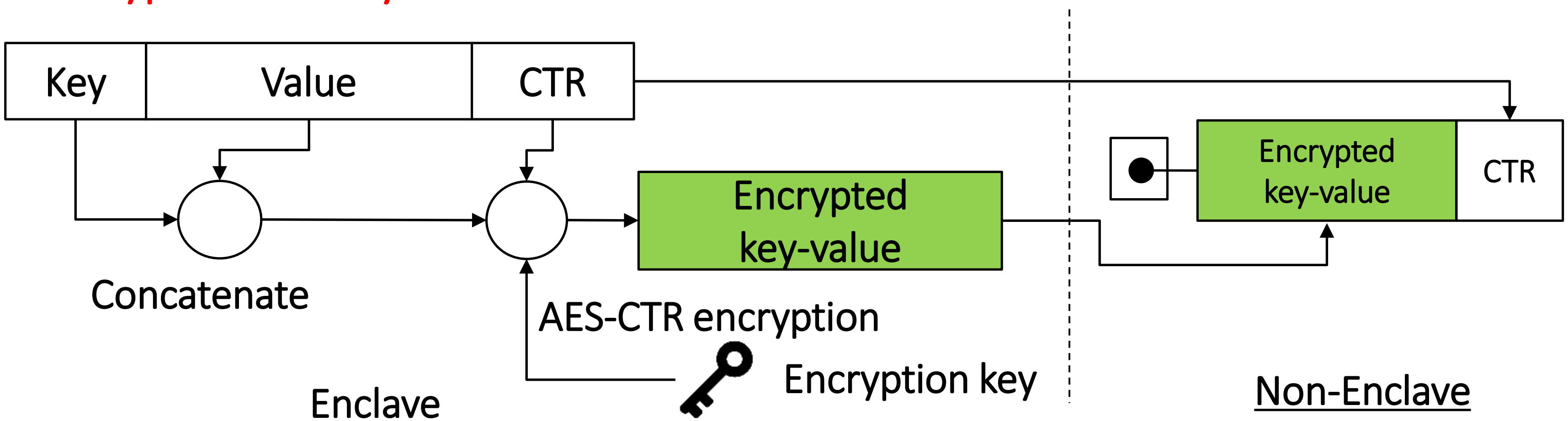


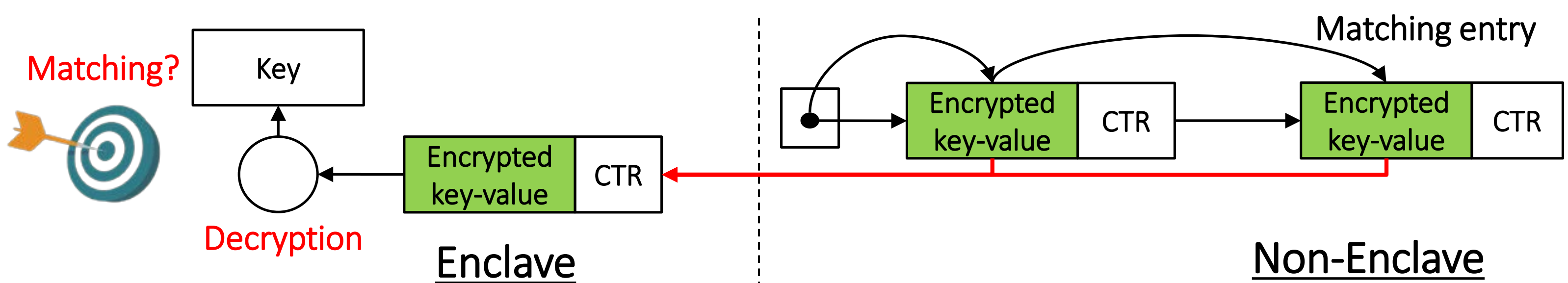
+ Reduce the depth of tree
+ Keep subtree root node on *enclave*
- Traverse all the MAC entries

Encryption

- ShieldStore encrypts both key and value of the entry
  - Alleviate the leakage of information

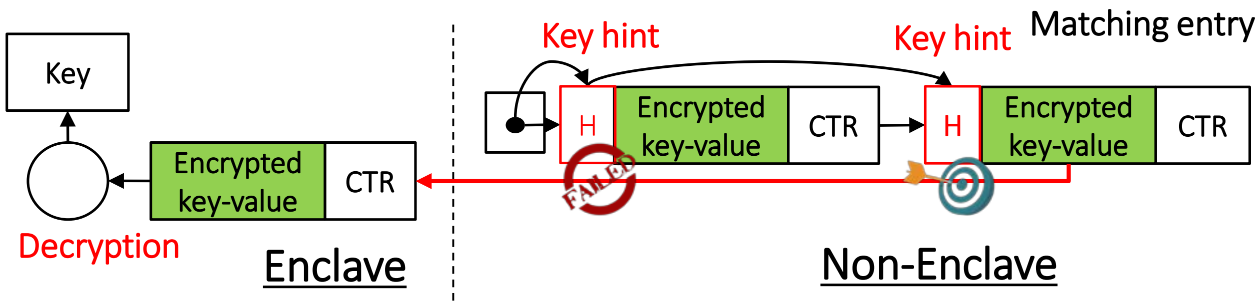
+ Reduce information leaks
- Decrypt all the keys in a same bucket

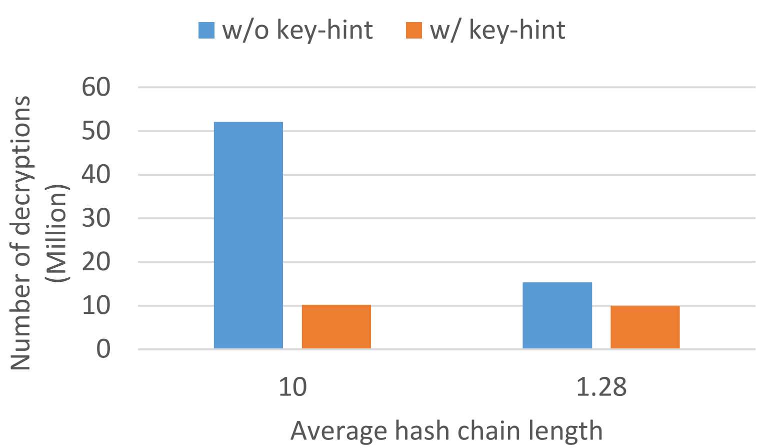




Optimization: Searching encrypted key

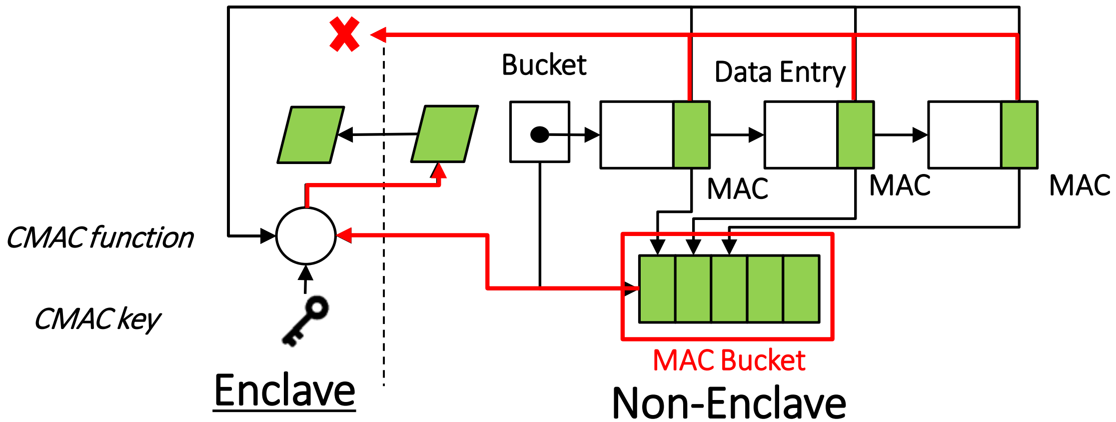
- Searching encrypted key
  - 1 byte key hint on data field





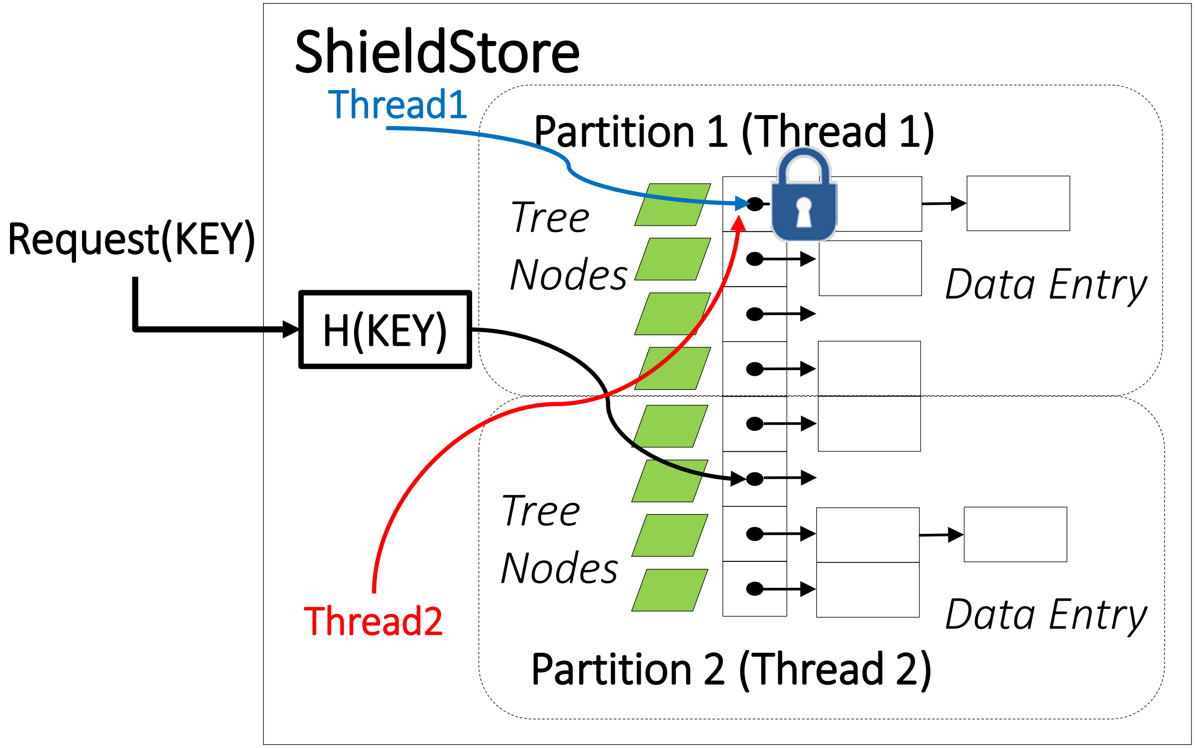
Optimizations: MAC Bucketing

- MAC bucketing
  - Maintain the MAC buffer per a hash bucket



Optimization: Multi-threading

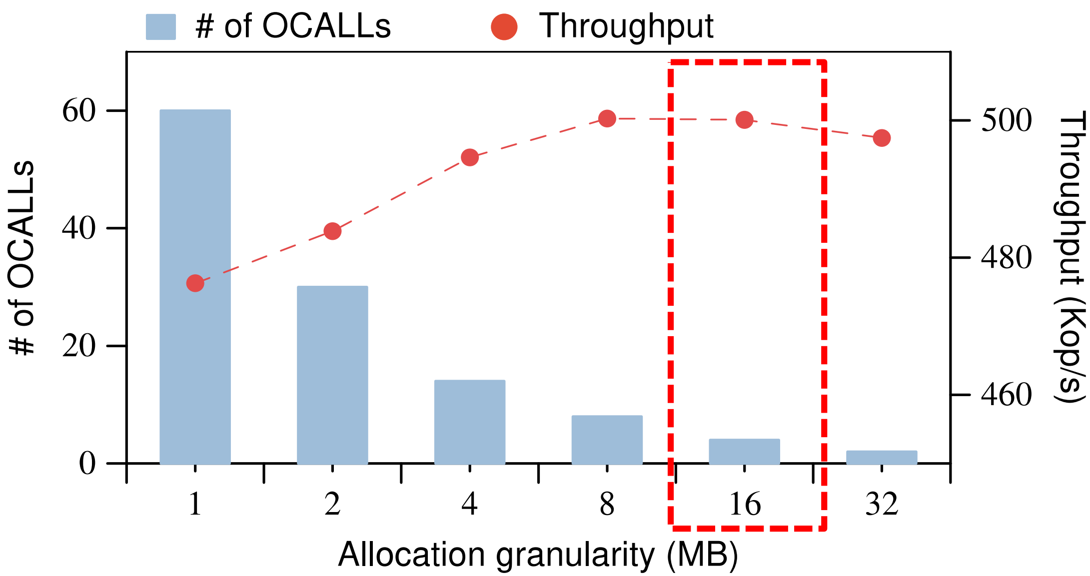
- Partition hash buckets with key distribution
  - Exploit the parallelism
  - Remove the overhead of synchronization across multiple threads



Partition(KEY) = H(KEY) / Total Threads

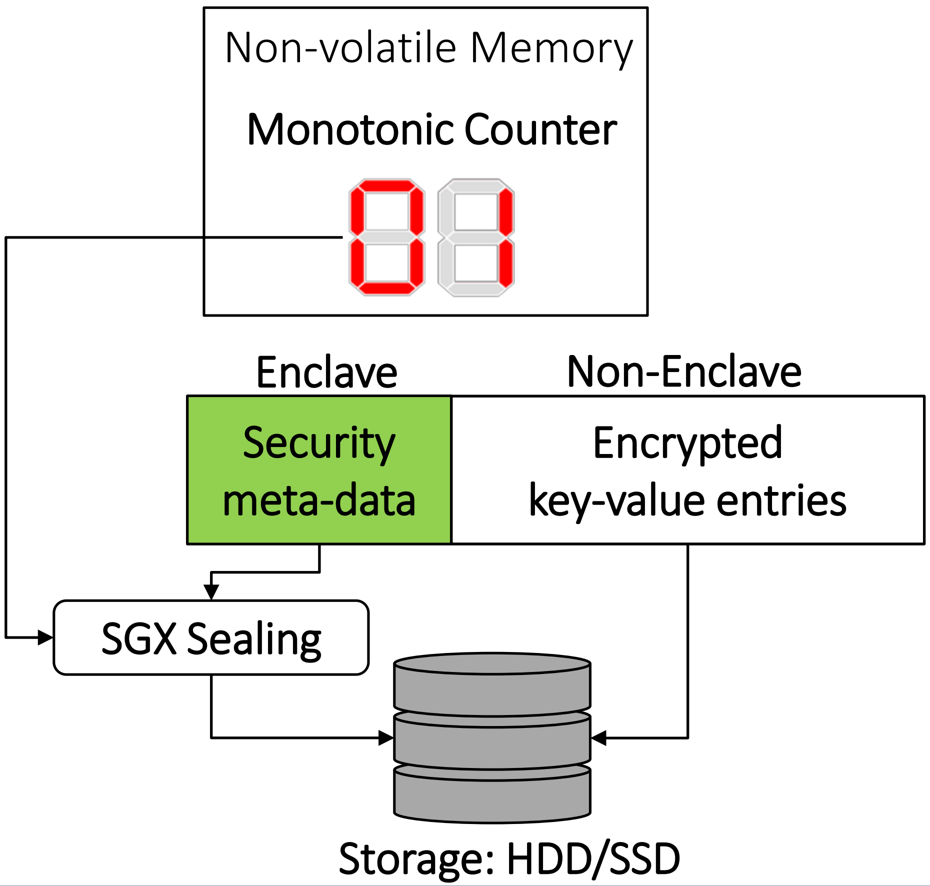
Optimization: Custom Heap Allocator

- Allocate untrusted memory on *enclave*
  - Reduce the EEXIT occurs in the calls out of *enclave* (OCALL)
  - OCALL only calls *sbrk()*



Persistent Support

- Intel SGX supports *sealing mechanism*
  - Using *monotonic counter* stored in non-volatile memory
  - Protect data from rollback attacks



Experimental Setup

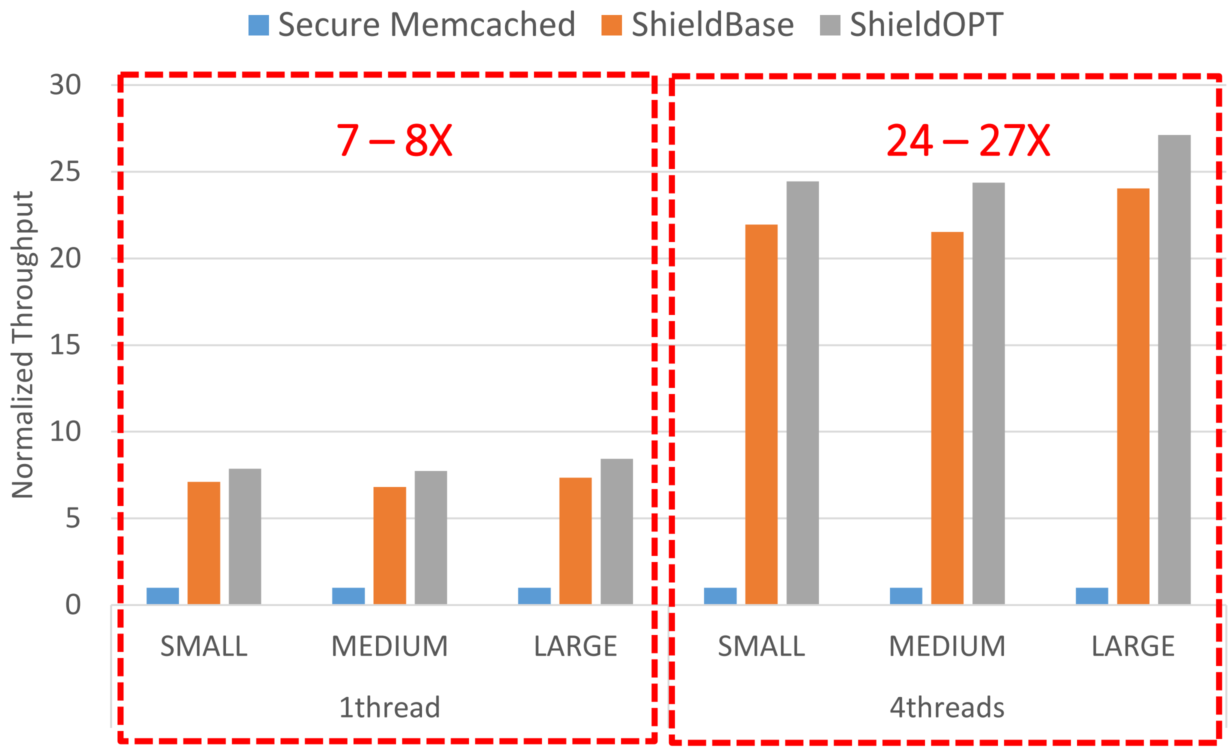
- Evaluation
  - Standalone: Focus on data store aspect without network
  - Network: Socket interface with a 10Gb NIC and 256 concurrent clients
- Metrics
  - *Secure Memcached*: memcached with grapheneSGX [3]
  - *ShieldBase*: ShieldStore without optimizations
  - *ShieldOPT*: ShieldStore with optimizations

Data Set	Key Size(B)	Value Size(B)	Working set(MB)
Small	16	16	305
Medium	16	128	1,373
Large	16	512	5,035

[3] Tsai, et al. Grahpene-SGX: A Practical Library OS for Unmodified Applications on SGX [USENIX ATC' 17]

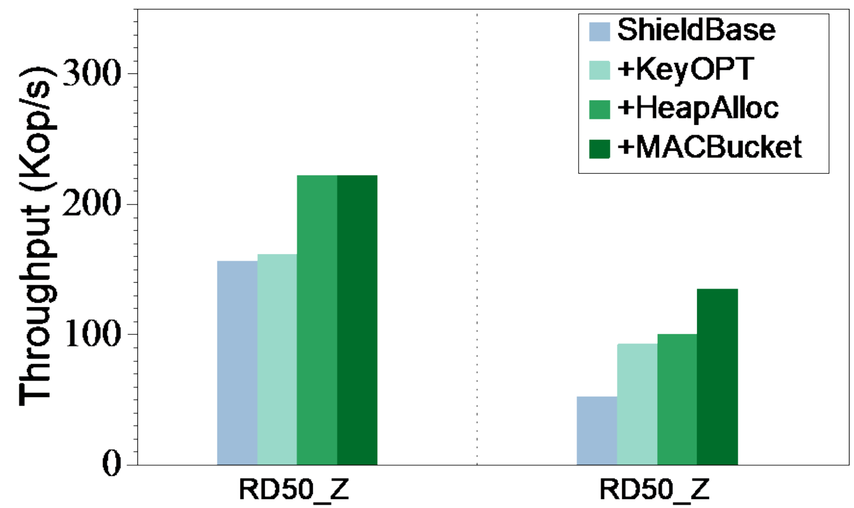
Standalone Evaluation

- ShieldStore performs
  - 7 – 8 times better than *Secure Memcached* on 1 thread
  - 24 – 27 times better than *Secure Memcached* on 4 threads



Effects of Optimizations

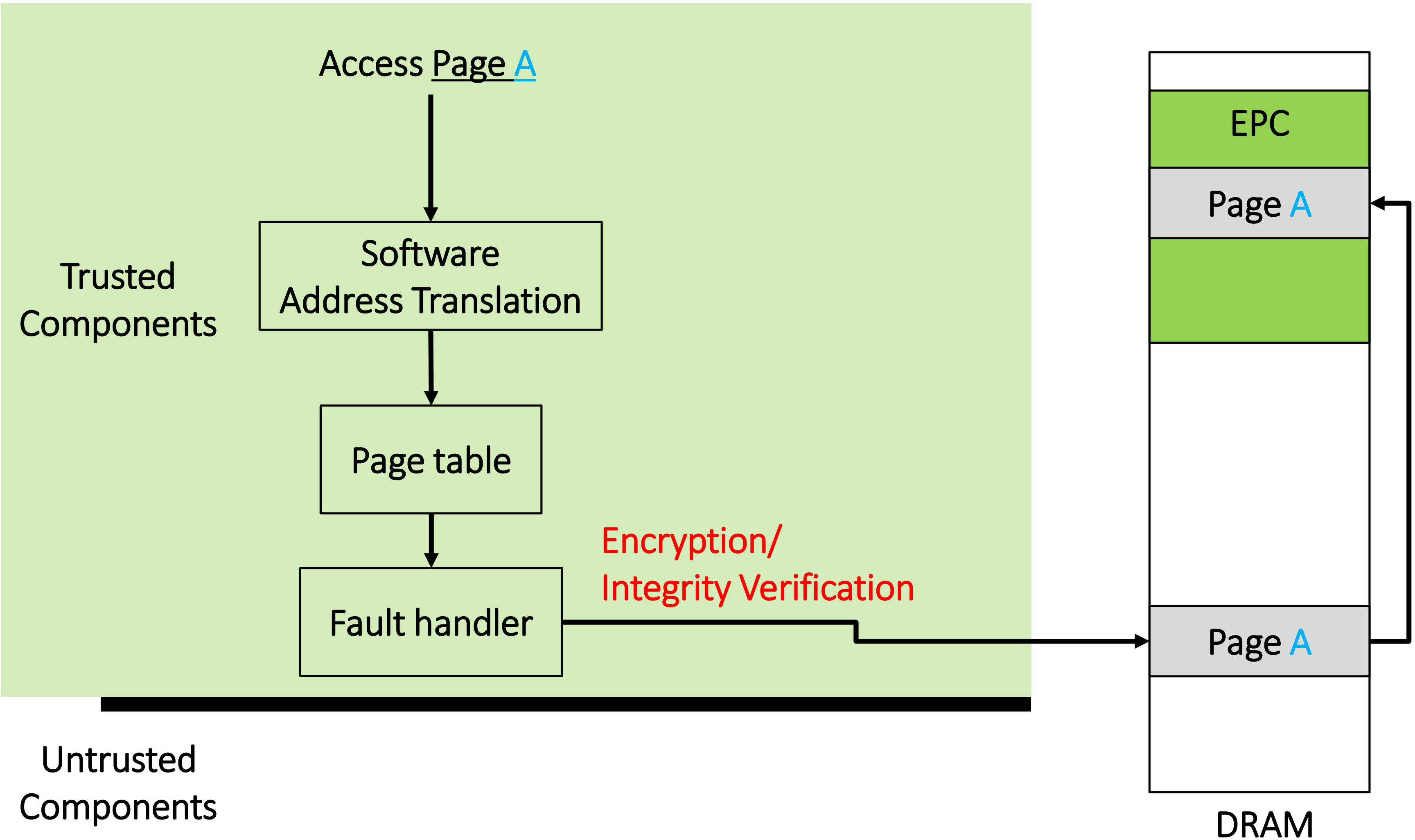
- The optimizations of ShieldStore increase performance
- Key hint & MAC bucket
  - Large affect on large hash chain length
- Custom heap allocation
  - Performance improvement on SET



	8M buckets 10M entries	1M buckets 40M entries
Average hash chain length	1.28	40

Eleos [1]: Exitless Software Paging

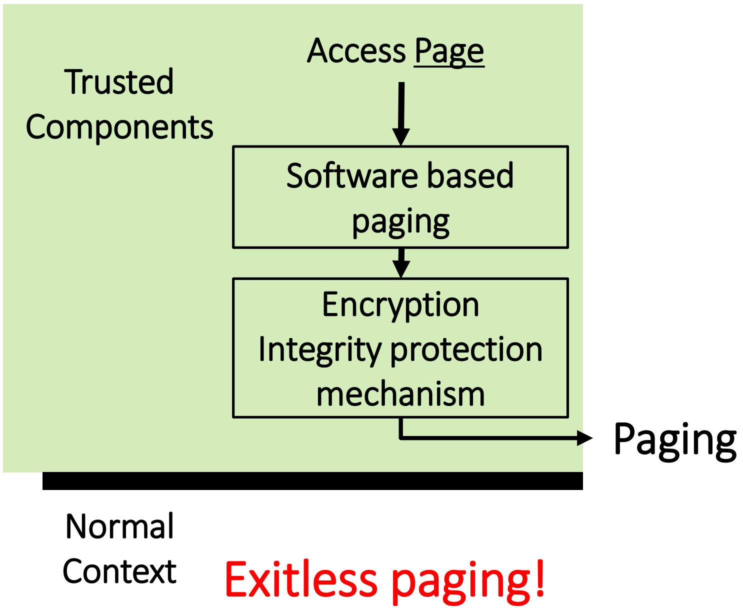
- Provide coarse-grained user space memory paging

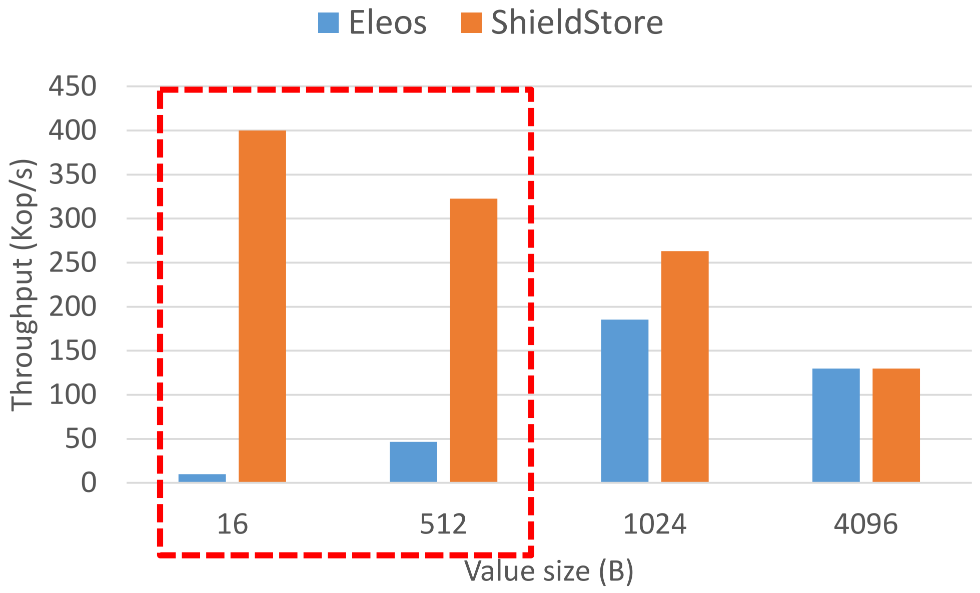


[1] Orenbach, et al. Eleos: ExitLess OS Services for SGX Enclaves [EuroSys' 17]

Comparison to Key-value Store on Eleos [1/2]

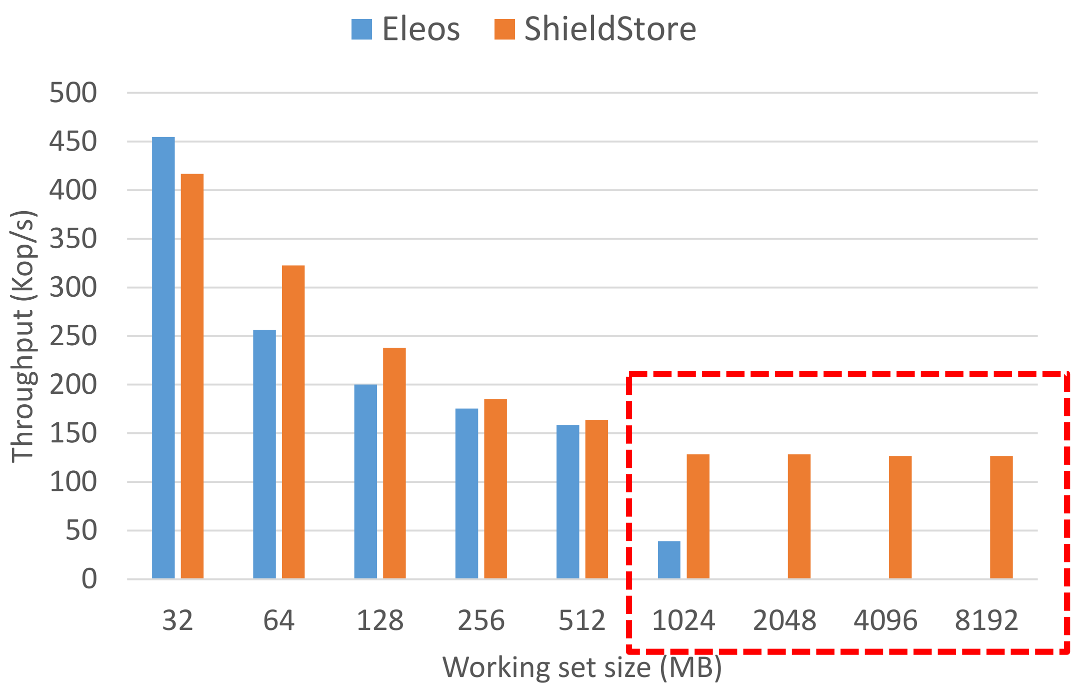
- Eleos provides coarse-grained user space memory paging
  - Eleos provides 1KB/4KB page-grained protection
  - ShieldStore provides fine-grained data protection





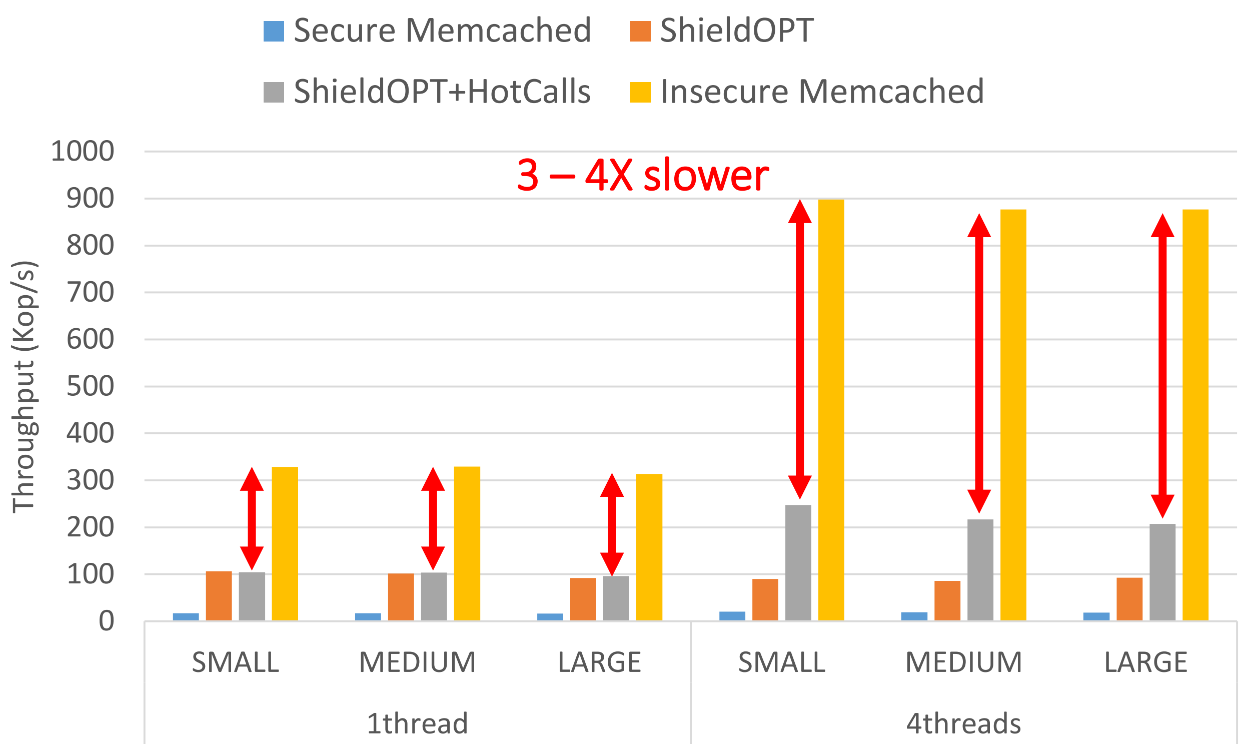
Comparison to Key-value Store on Eleos [2/2]

- ShieldStore performs better than Eleos even with 4KB value
  - Efficient data protection improves the performance of ShieldStore



Network Evaluation

- ShieldStore with HotCalls [4] performs
  - 6 – 11 times better than *Secure Memcached* on 1 thread and 4 threads
  - 3 – 4 times slower than *Insecure Memcached* on 1 thread and 4 threads

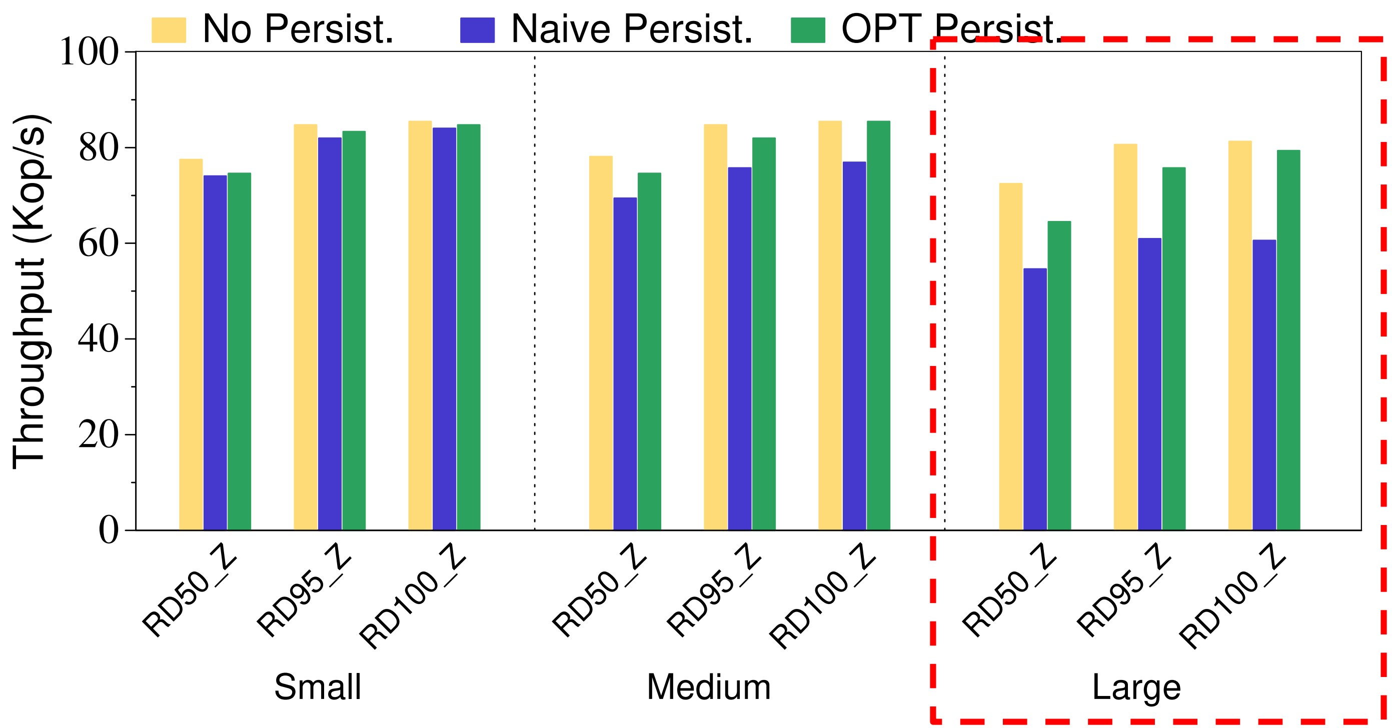


[4] Weisse, et al. Regaining Lost Cycles with HotCalls: A Fast Interface for SGX Secure Enclaves [ISCA' 17]

Persistent Support Evaluation

- The overhead of naïve approach becomes higher with large data

- Optimized persistent approach
  - Degrade 2.1 – 6.5% of performance on average



Summary of Paging Principles

	SGX paging (*Secure Memcached*)	User space paging (*Eleos*)	Semantic aware protection (*ShieldStore*)
Security Granularity	Page	Page/Sub page	Key-value
	Fixed/coarse-grained		Variable/fine-grained
Crossing Enclave Overhead	High	Small	Small
		More efficient	

ShieldStore code will be available at
https://github.com/cocoppang/ShieldStore

Computer Architecture Lab.

ShieldStore: Shielded In-memory Key-value Storage with SGX

Taehoon Kim, Joongun Park, Jaewook Woo,
Seungheun Jeon, and Jaehyuk Huh

School of Computing

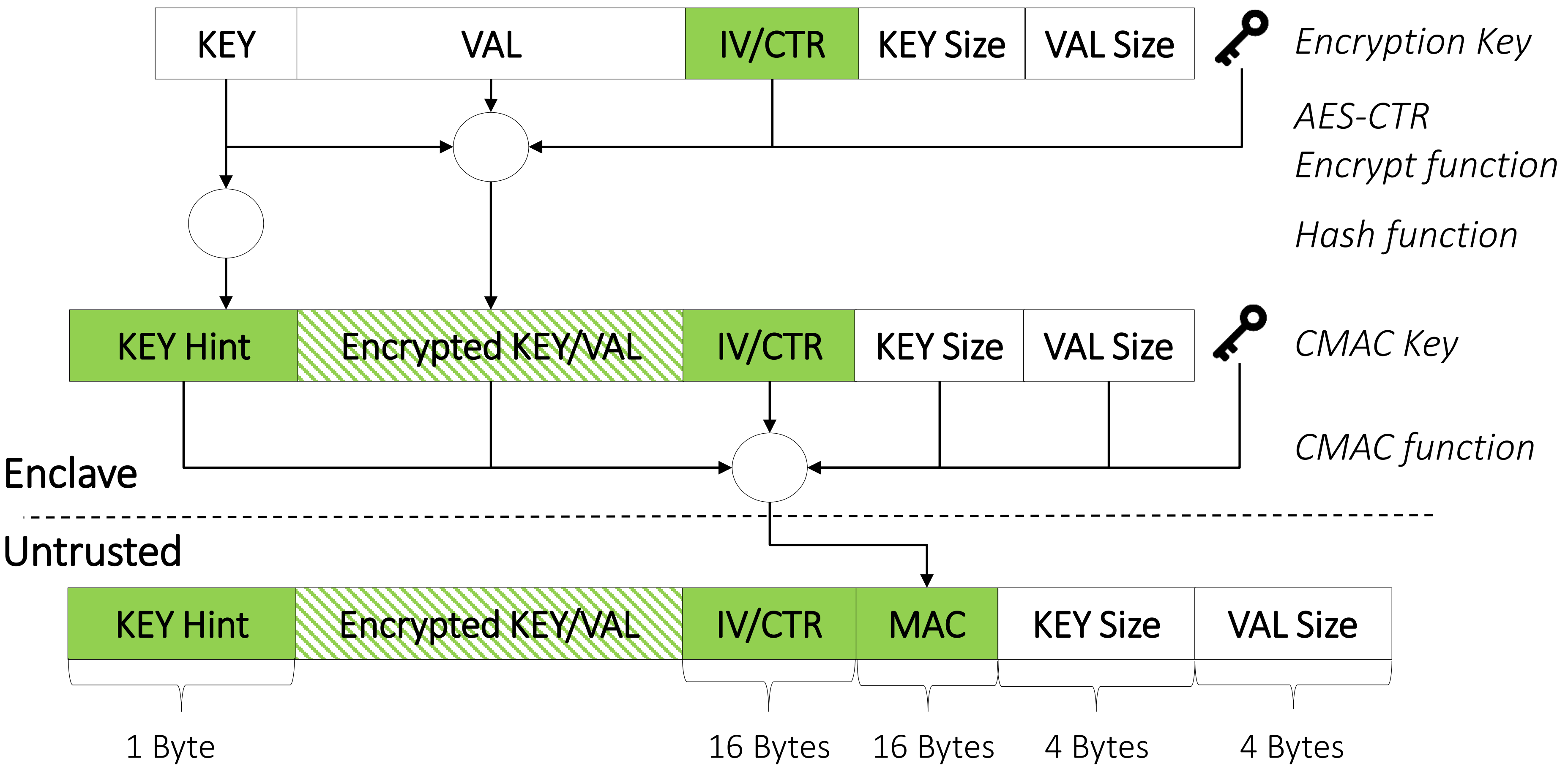
Limitations & Future Work

- Hash based key-value stores
  - Does not support range queries
- Vulnerability of SGX
  - Foreshadow[1] can make the *enclave* region vulnerable.
  - Micro-code update reduces performance
- Weak persistent support
  - Need fine-grained log-based persistent support

[1] Bulck, et al. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution [USENIX Security’ 18]

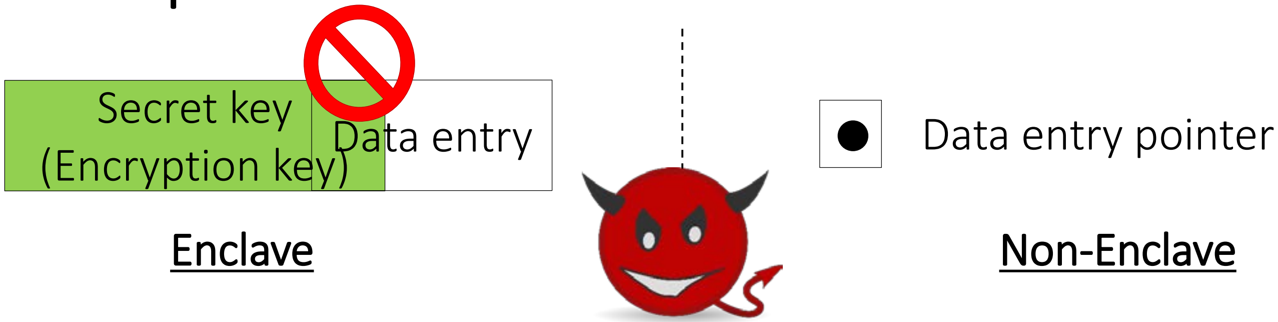
Data Entry of ShieldStore

- The data entry needs more fields for the protection mechanism



Security Consideration

- Untrusted pointers



- Untrusted meta-data of custom heap allocation
  - Attacker can maliciously manipulate allocator's meta-data (free lists, synchronization primitives)

- Key-hint