Enclave Application Cache for RISC-V Keystone

Takumu Umezawa, Akihiro Saiki, Keiji Kimura
(graduated)
Waseda University, Japan

TEE, Enclave, and Startup Overhead

- Computer systems deal with secret information
- Widely spreading demands for Trusted Execution Environment (TEE)
  - Smartphones
  - IoT, Edge devices
  - Cloud

Enclave Type (Intel SGX, ARM OP-TEE, RISC-V Keystone)

Secure VM Type (AMD SEV, Intel TDX, ARM CCA, RISC-V CoVE)

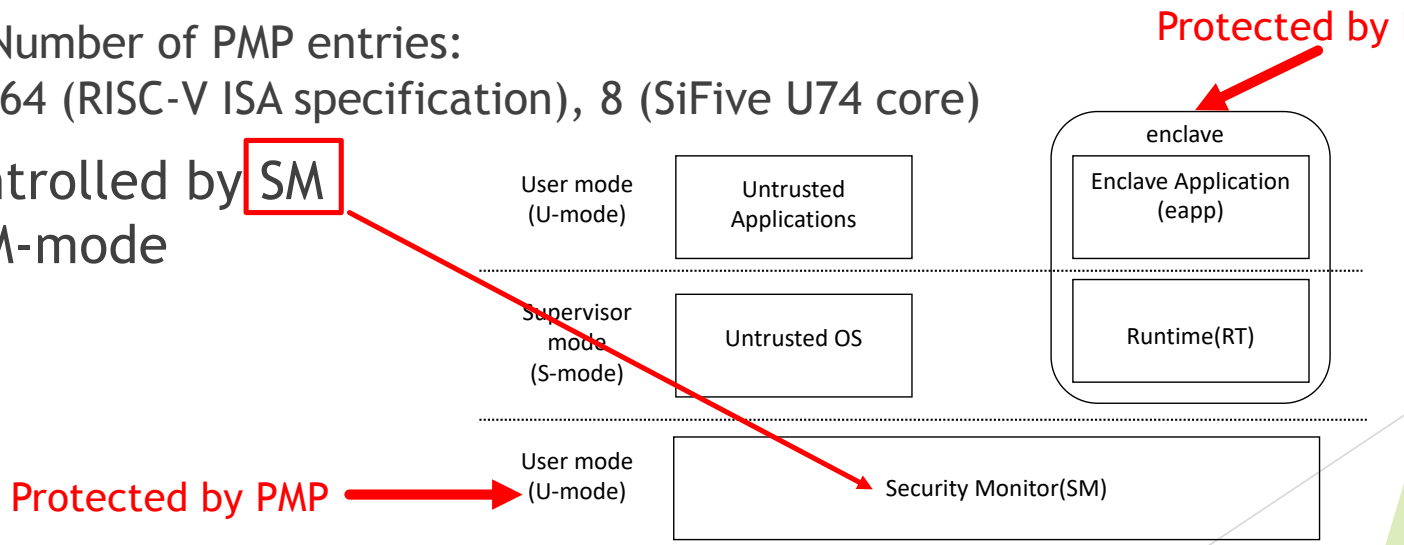
- Need to ensure the trustworthiness of applications/VMs and an execution platform.
  - Measurement of application/VM Binary Image at the startup time
    - Hash calculation ➡ Expensive Startup Cost

Enclave Application Cache

Penglai's Shadow Enclave dealt with a similar target.

Our Target: RISC-V Keystone

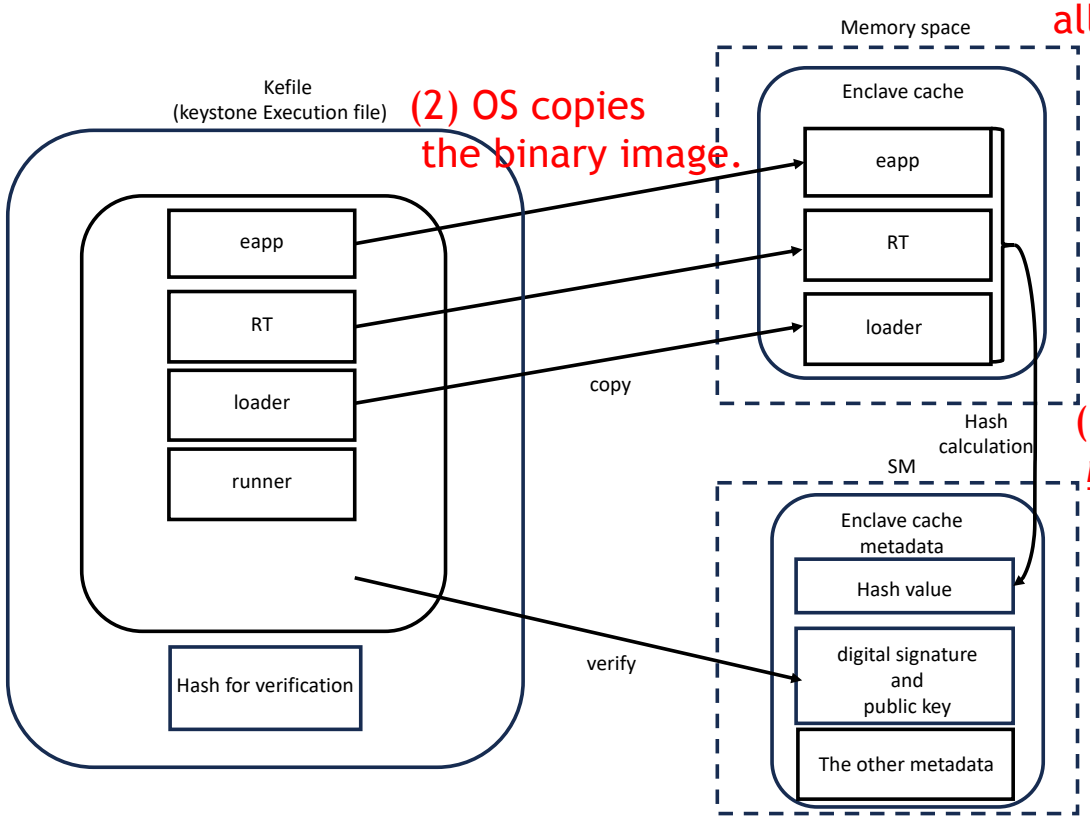
- Open Source, Enclave type TEE
- Utilizing RISC-V PMP (Physical Memory Protection) for memory isolation
  - Setting physical address information to be isolated in PMP registers
  - Number of PMP entries:
    64 (RISC-V ISA specification), 8 (SiFive U74 core)
- Controlled by SM in M-mode



(from RISC-V Keystone paper [6])

Keystone Eapp Startup Process

(1) Keystone Device Driver allocates memory space. (contiguous physical address region)

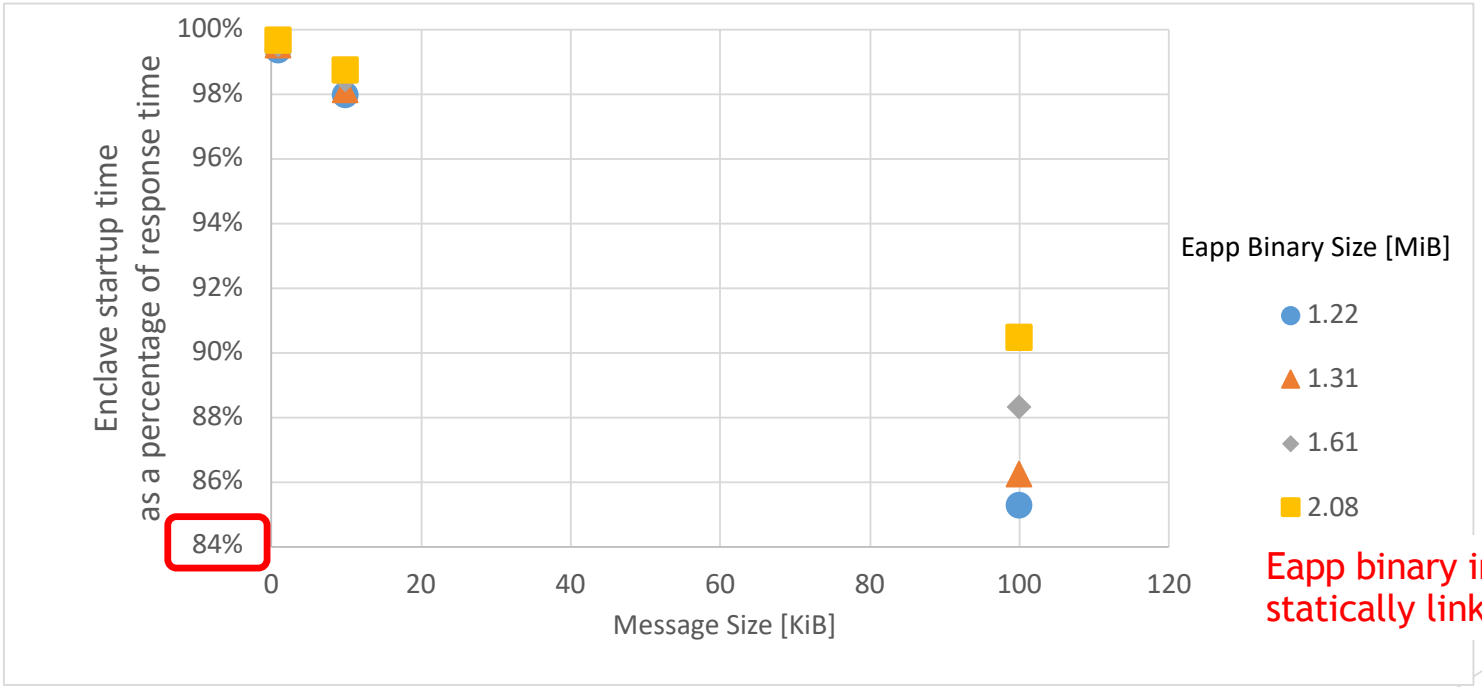


(3) SM sets PMP and *measures the memory image*.

Preliminary Evaluation (Startup Overhead)
Evaluation Environment: HiFive Unmatched

Core	U74 (1.2GHz)	S7 (1.2GHz)
# of Cores	4	1
L1I-Cache	32KiB	16KiB
L1D-Cache	32KiB	N/A
L2-Cache	2MiB	
Main Memory	DDR4 16GiB	
Storage	Samsung SSD 970 EVO Plus 250GB (MZ-V7S250BW)	
Evaluation Program	Digital Signature Server Program using ED25519 and SHA3-512	

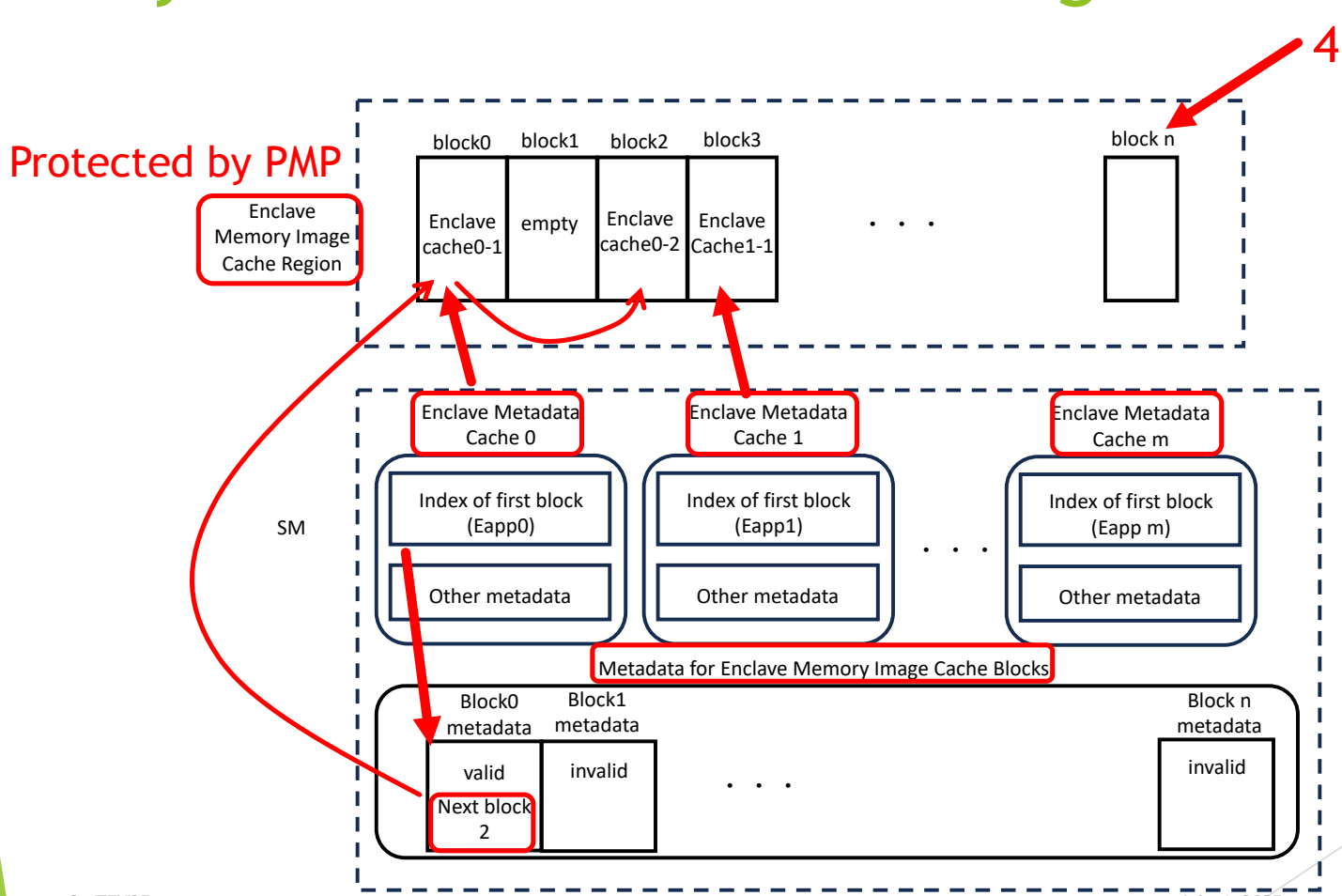
Preliminary Evaluation (Startup Overhead) Result



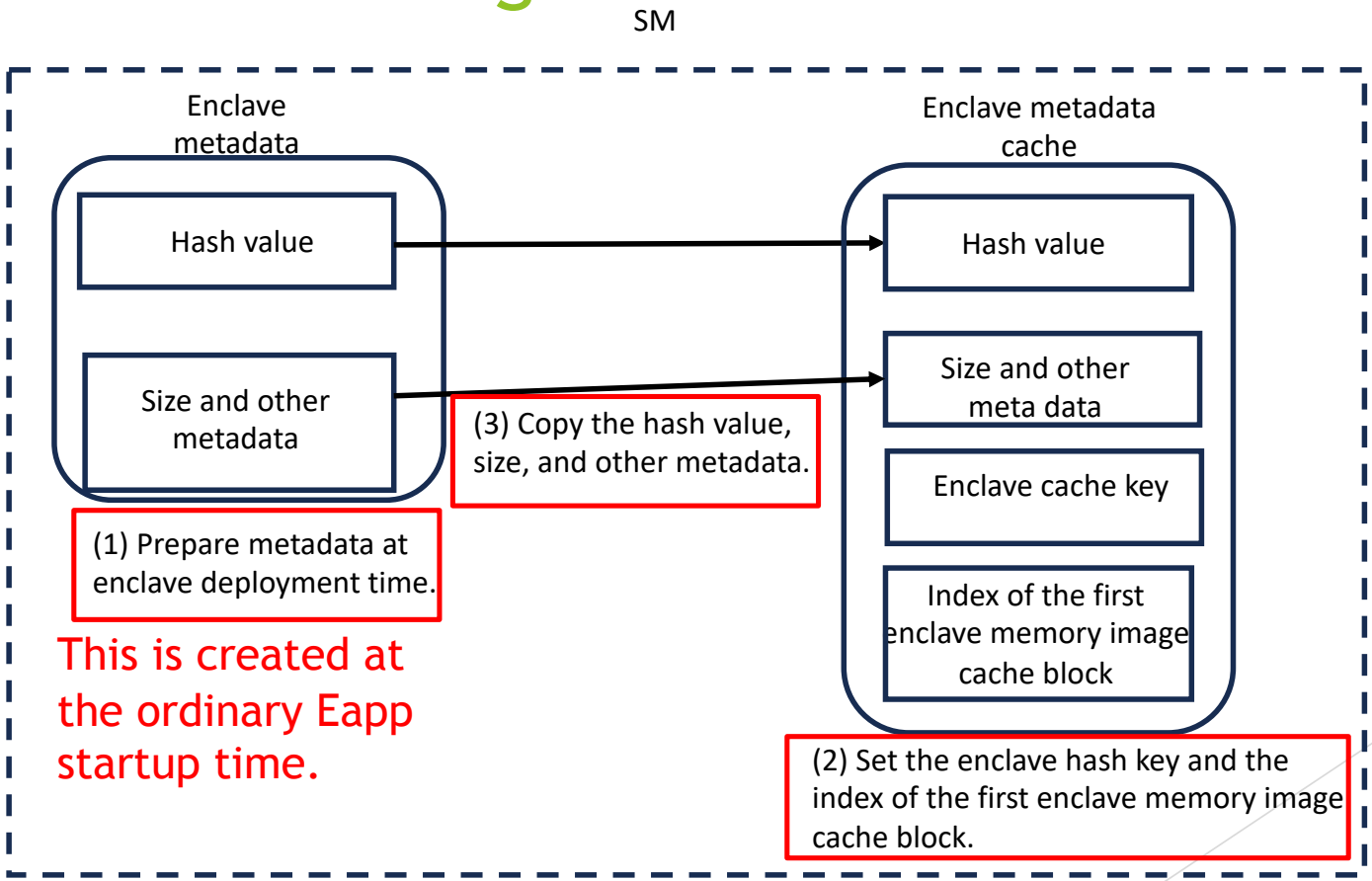
Eapp binary includes Eapp (with statically linked library), RT, Loader.

> 85% of execution time is spent on Eapp startup.

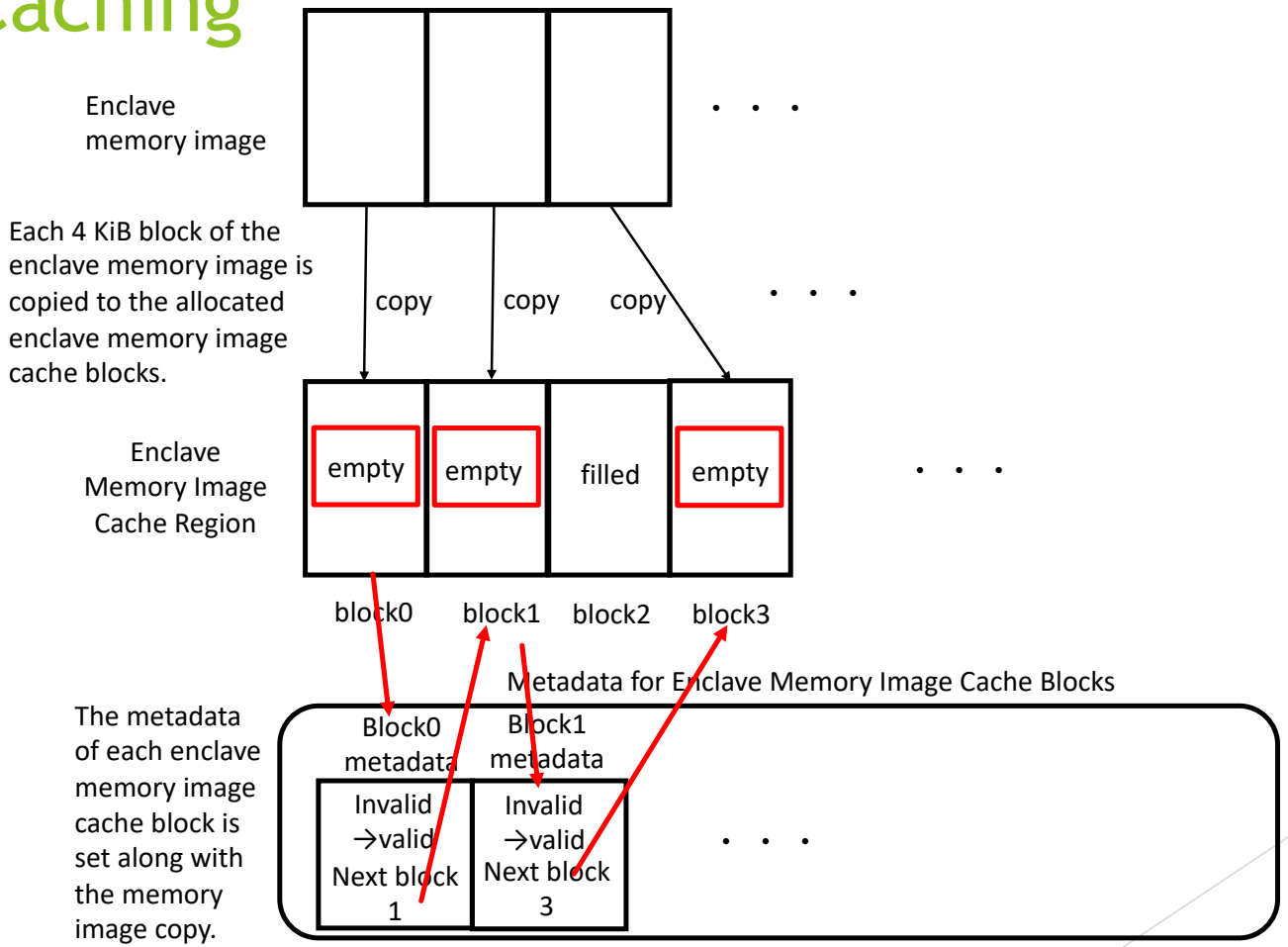
Keystone Enclave Cache Design



Caching Eapp (Cache miss): Metadata Setting after Enclave Creation



Caching Eapp (Cache miss): Memory Image Caching



Launching Enclave from Cache (Cache Hit)

- Inverting the caching process.
- Creating the eapp memory image from the memory image cache blocks.
- Setting the eapp metadata from the cached metadata in SM.

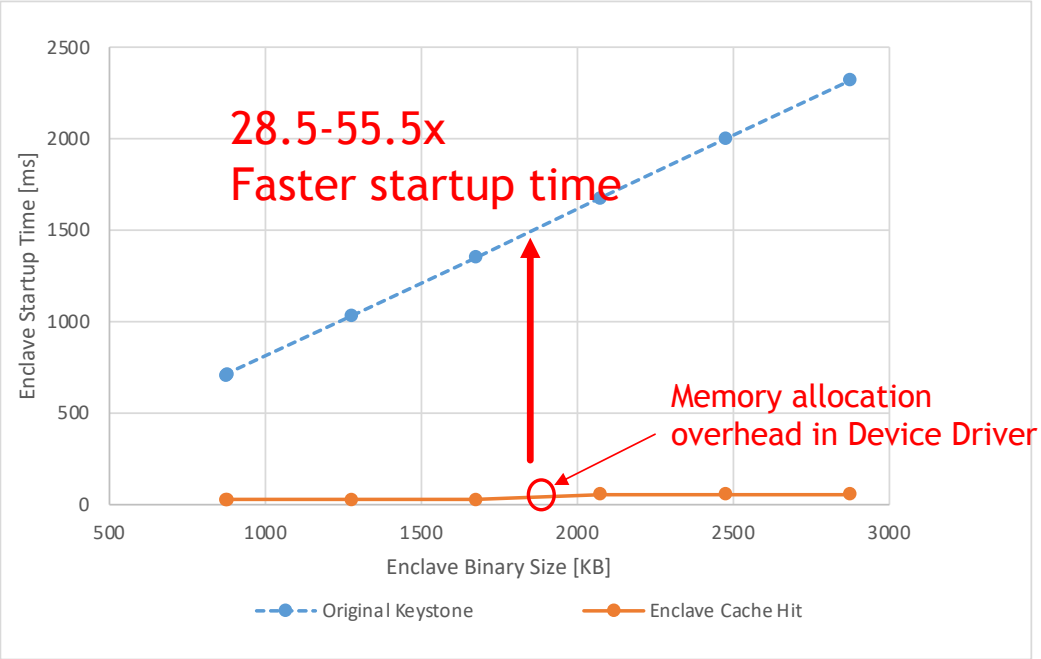
Security Analysis

- Ensuring the same security level as Keystone
  - Protecting Memory Regions by PMP
    - Enclave Memory Image Cache Region
    - Cache Metadata in SM
  - Cache Management Processes in SM
    - Caching Process, including measurement
    - Launching Eapp from Cache

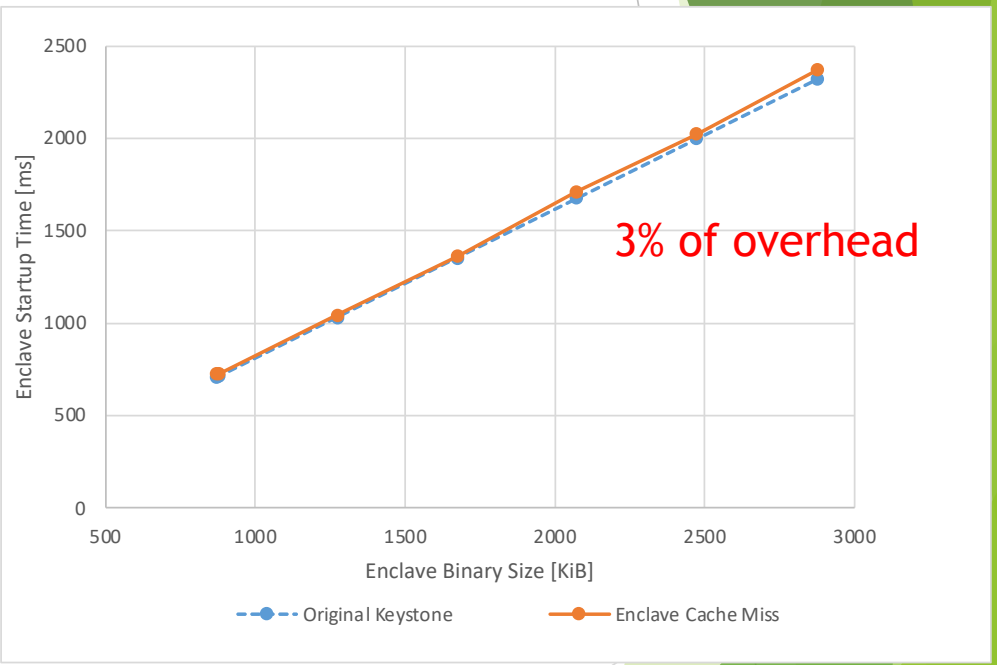
Experimental Evaluation

- Evaluation Platform
  - HiFive Unmatched (also used for the preliminary evaluation)
- Primitive Performance
  - Eapp launching speedup by Enclave Cache
  - Cache-miss Overhead introduced by Enclave Cache
- Evaluation on Digital Signature Program

Primitive Performance

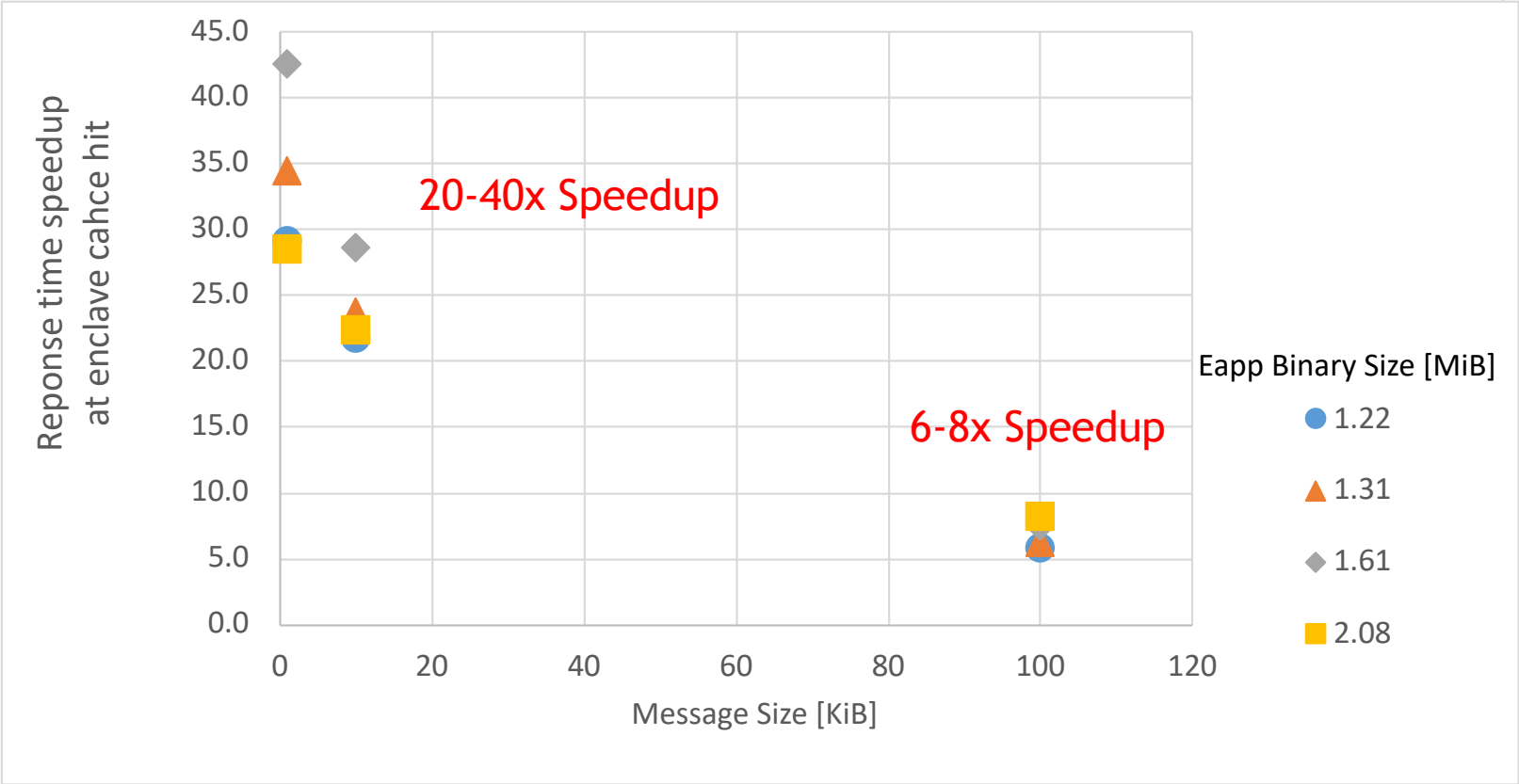


Cache Hit



Cache Miss
(Launching + Caching)

Digital Signature Eapp Performance



Comparison with Penglai's Shadow Enclave

- Shadow Enclave: Launching Enclave before execution
  - User explicitly creates it.
- Comparison with another type of cache mimicking Shadow Enclave.
  - Caching an Eapp in a dedicated Enclave.
  - Consuming one PMP entry for one Eapp cache.
- Shadow Enclave type cache obtains 10-15 [ms] shorter startup time at Cache Hit.
  - Our Enclave Cache creates the memory image by traversing Enclave Memory Image Cache Blocks.
- Our cache needs no hardware extension.
- Only consuming one PMP entry.

Conclusion

- Enclave Application Cache for RISC-V Keystone
- Caching Eapp binary images in the protected memory regions.
  - To reduce the expensive Eapp launching cost.
- Performance evaluation:
  - 28.5-55.5x faster startup time at Cache hit.
  - 6-40x speedup for a digital signature program.