KAIST School of Computing CASYS Computer Architecture and Systems Lab

Hardware Hardened Sandbox Enclaves for Trusted Serverless Computing

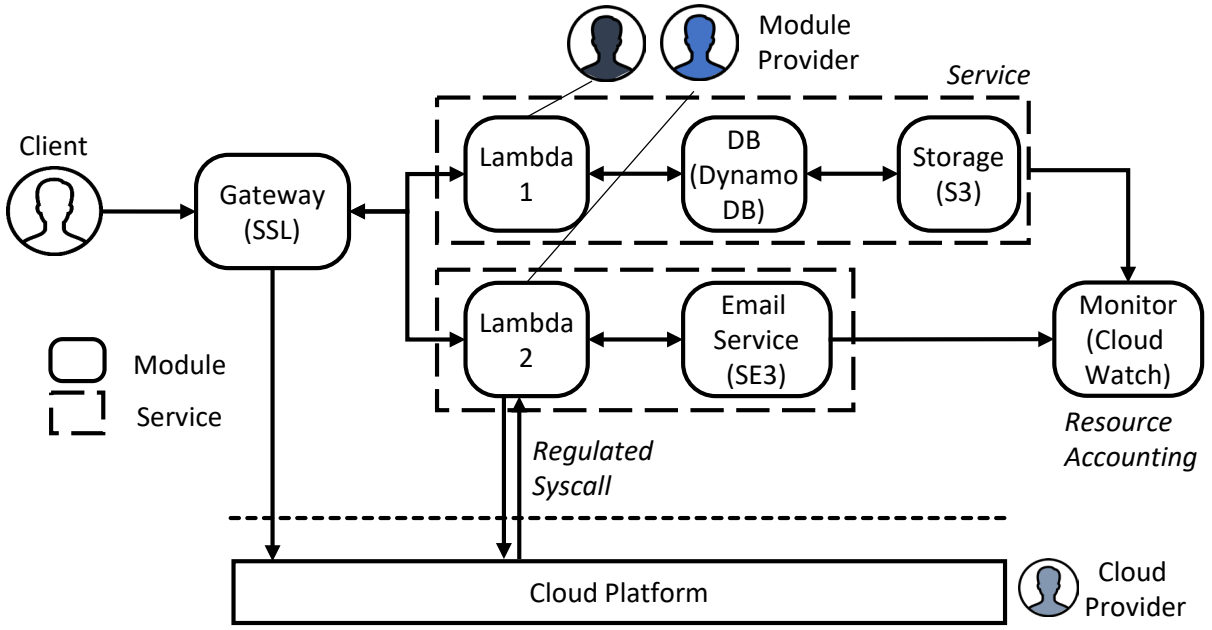
Joongun Park

ACM Transactions on Architecture and Code Optimization 21(1)

Excerpted from the PhD defense slides by Joongun Park

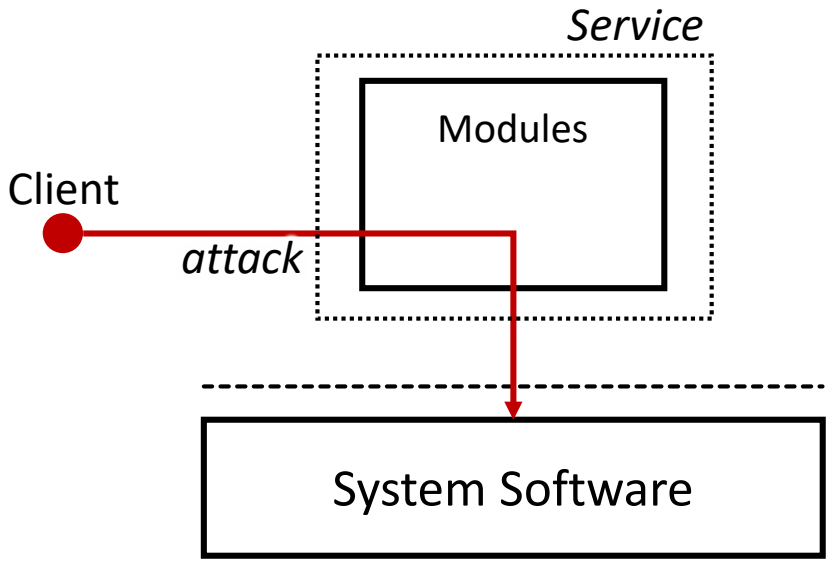
Serverless Computing

- Cloud provider manages the underlying platform instead of a developer
- Services consist of user-level, stand-alone, isolated function modules
- Pay-as-you-go model

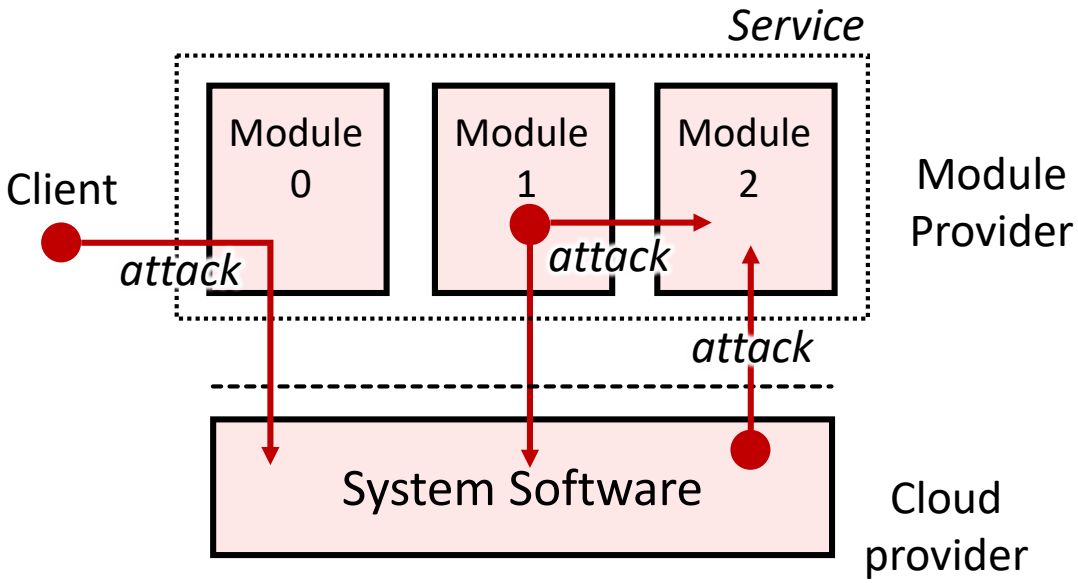


What's different in Cloud security?

- Traditional
  - Protect module/system from a client



- Cloud security
  - Protect each compartment from others

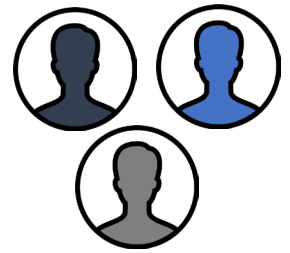


Requirements for Secure Cloud

Cloud Provider

Protect against malicious Client and Module Provider

Protect against malicious Cloud Provider, Clients, and other Module Providers



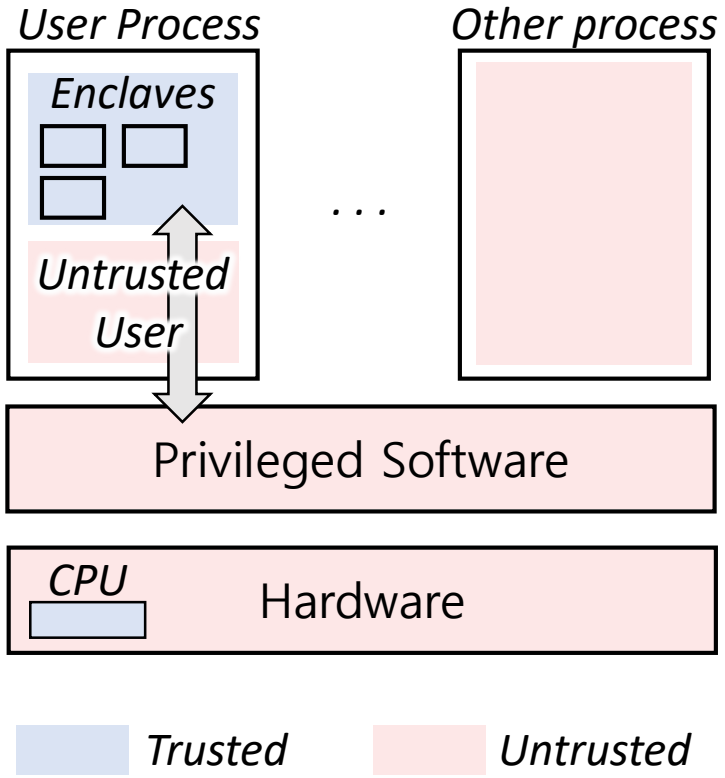
Module Provider

Client

Wants to minimize privacy exposal to
Cloud Provider, Module Provider, and other Clients

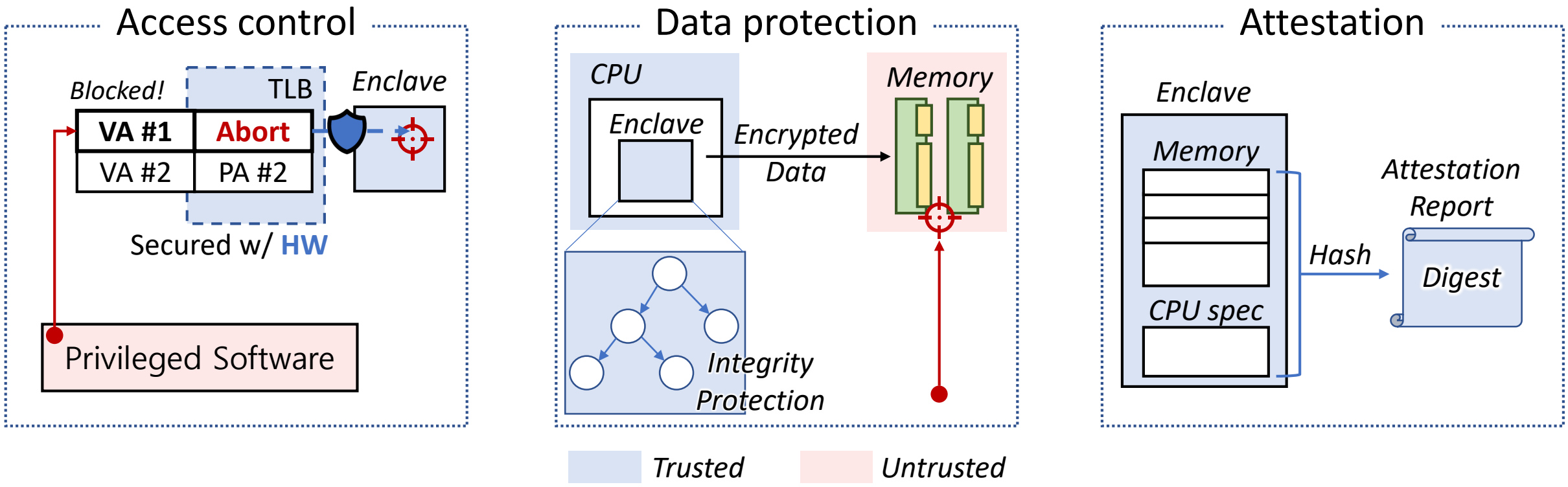
Trusted Execution Environment (TEE)

- **TEE** is an execution environment that provides protection against privileged software and physical attacks
- **Enclave** is one of the TEE models which supports user-level instances
- An Enclave provides,
  - Access control
  - Data protection
  - Attestation

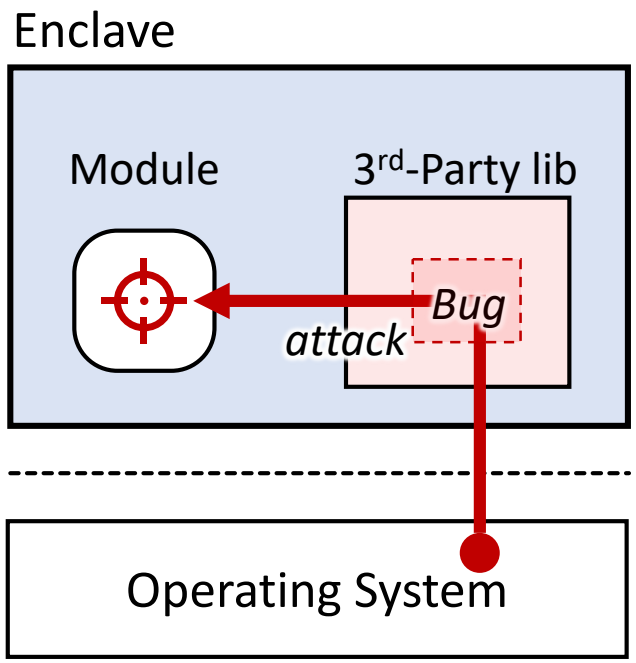


Hardware implementation of Enclave

- Intel's commercialized Trusted Execution Environment called **SGX**



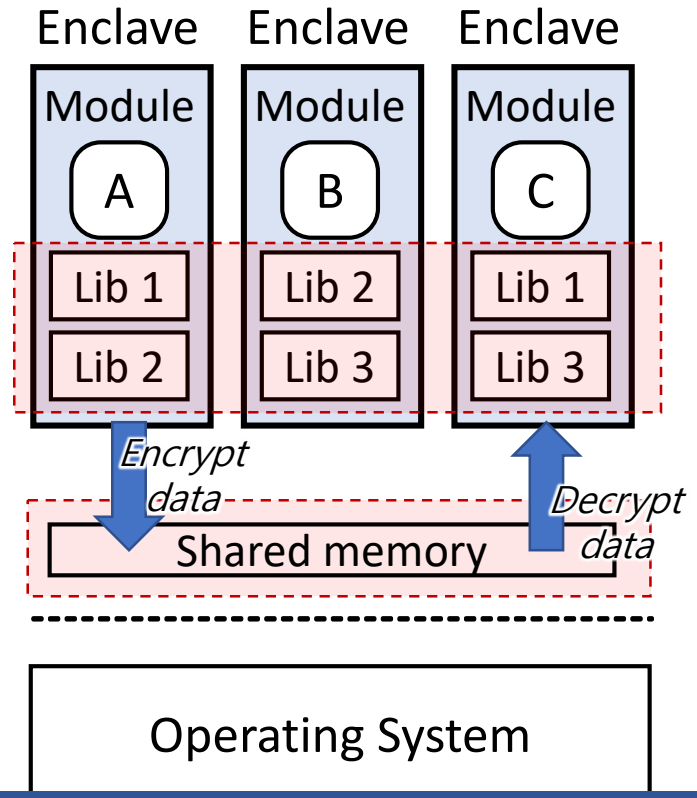
Challenge 1: Monolithic Design



- A service module importing a 3rd-party library in Enclave as its TCB
- Vulnerability in 3rd party endanger entire the enclave
  - E.g., Memory leak (OpenSSL HeartBleed)

1. Third-party library should be isolated

Challenge 2: Lack of Sharing Semantic

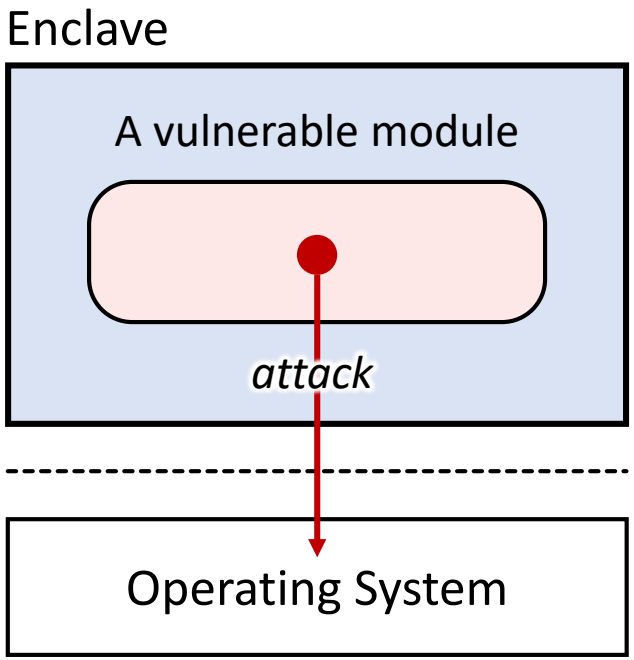


- Memory is wasted due to strictly isolated libraries
- SW encryption has limitations
  - Costly encryption
  - Race condition attack (TOCTOU) [1]
  - Knowledge attack (Silent dropping) [2]

2. Enclave needs to support secured sharing

[2] Panoply: Low-TCB Linux Applications With SGX Enclaves (NDSS' 17)

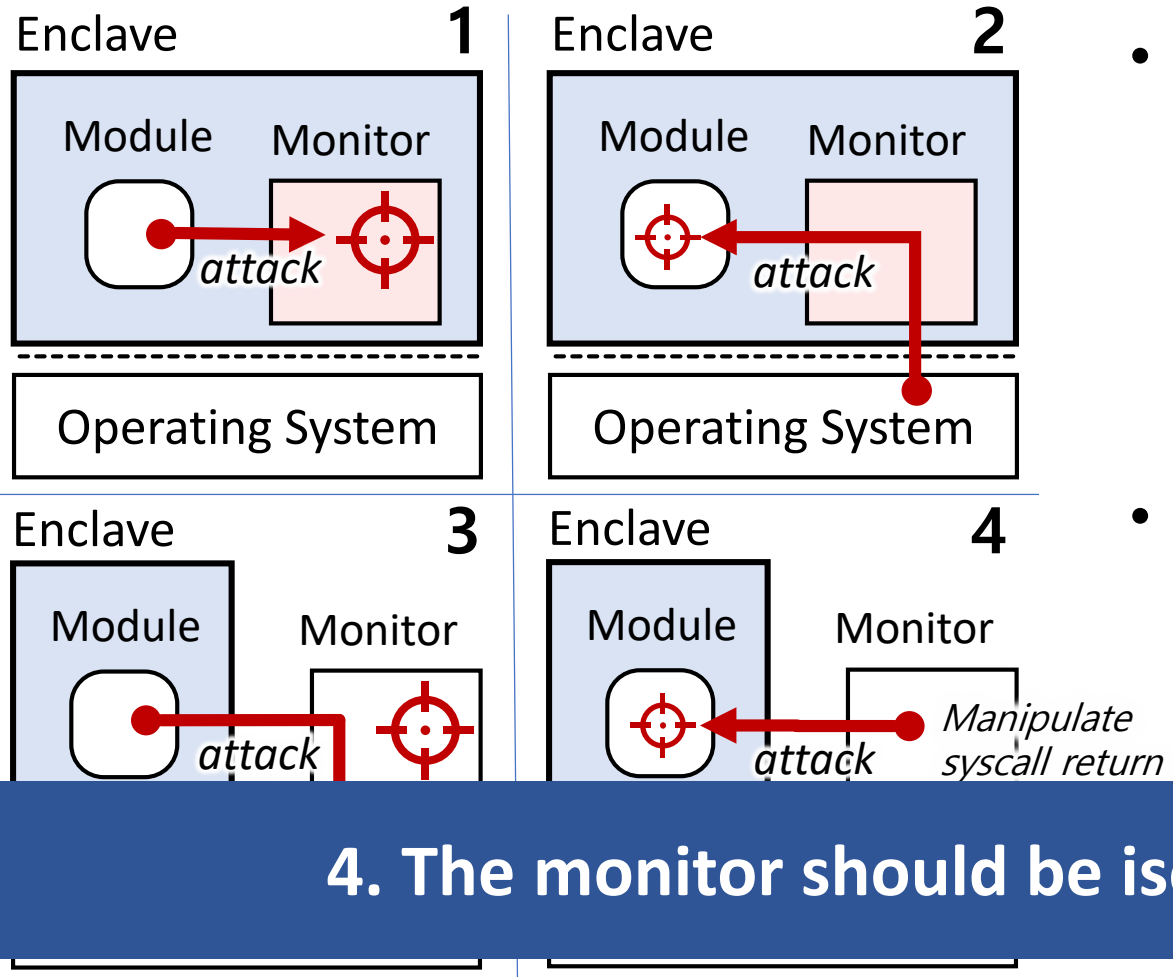
Challenge 3: Potentially vulnerable module



- An enclave may attack the system
  - Malicious module provider
  - Vulnerable module
- Enclave is one-way protection
  - Protect the service from the cloud
  - Sandbox is needed

3. System should be protected from Enclave (Sandboxing)

Challenge 4: Securing monitor



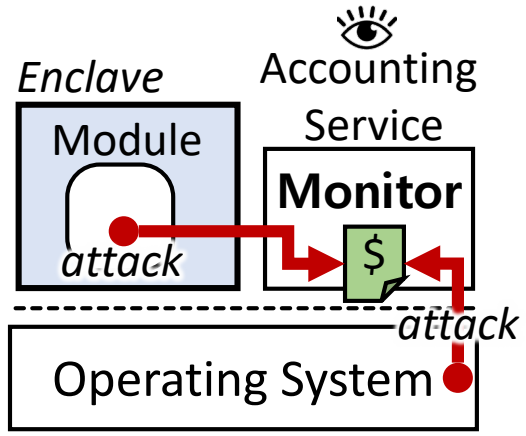
- Sandbox monitor is a mediator between the system and module
  - System call delegation for sandbox
  - 328 vulnerabilities have been reported in 2019-2022 at CVE
- Hard to protect all compartment
  - 1,2: Vulnerable monitor
  - 3: Privilege escalation (e.g, shellshock[1])
  - 4: Iago attack[2]

4. The monitor should be isolated both module and kernel

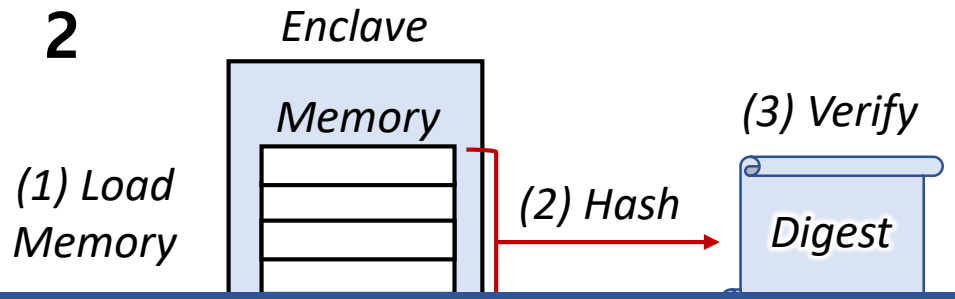
[1] CVE-2014-7169, [2] Iago Attacks: Why the System Call API is a Bad Untrusted RPC Interface (UCSD dissertation)

Challenge 5: Trusted System Level Objective

1



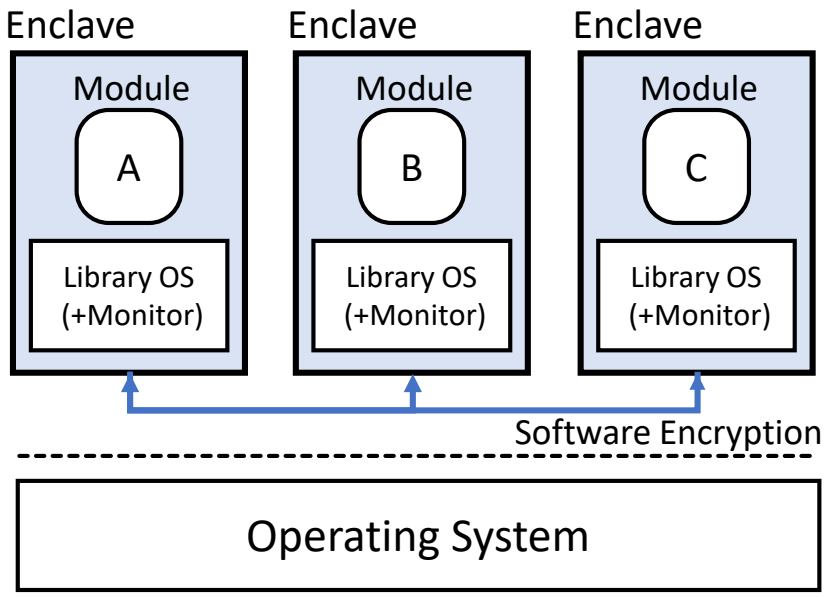
2



- Resource accounting should be protected from both side
  - Bill should be mutually-agreeable
- In order to meet SLO, nimble instance loading is important
  - Verification slows down loading

5. Resource usage should be securely tracked, and Loading should be accelerated

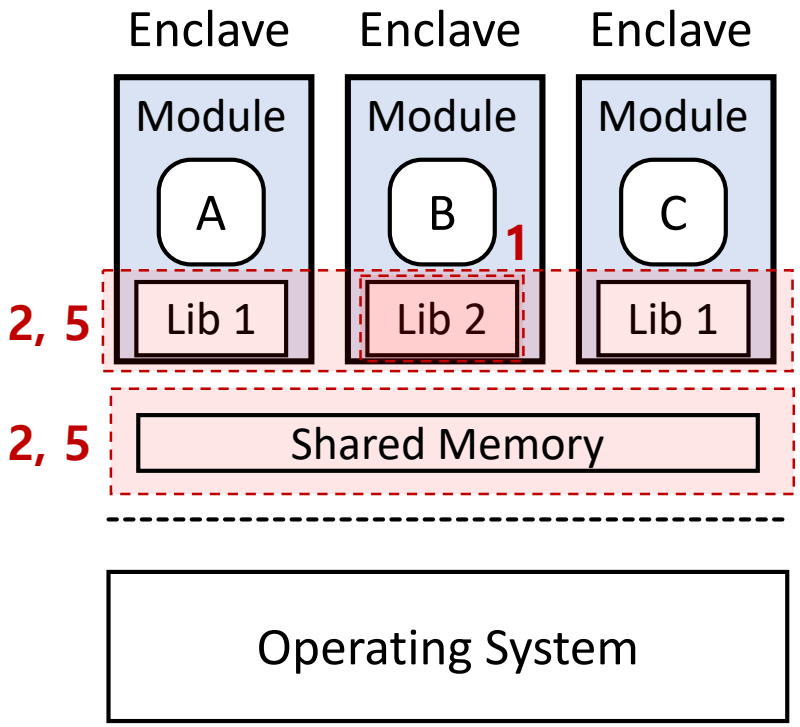
Prior Work



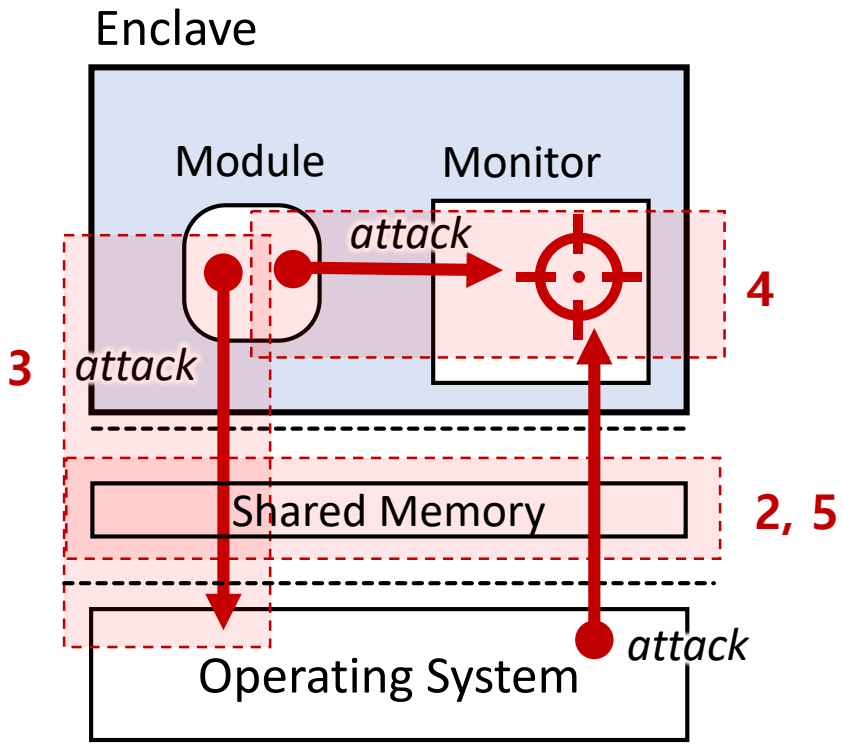
Ryoan

Related Work	Challenge 1	Challenge 2	Challenge 3	Challenge 4	Challenge 5
Ryoan (OSDI '16)	O	X	O	X	X
AccTEE (Middleware '20)	X	X	O	X	O
Occlum (ASPLOS '20)	X	O	O	X	X
CHANCEL (NDSS '21)	O	X	O	X	X
SGXLock (Security '22)	X	X	O	X	X

Challenge Summary



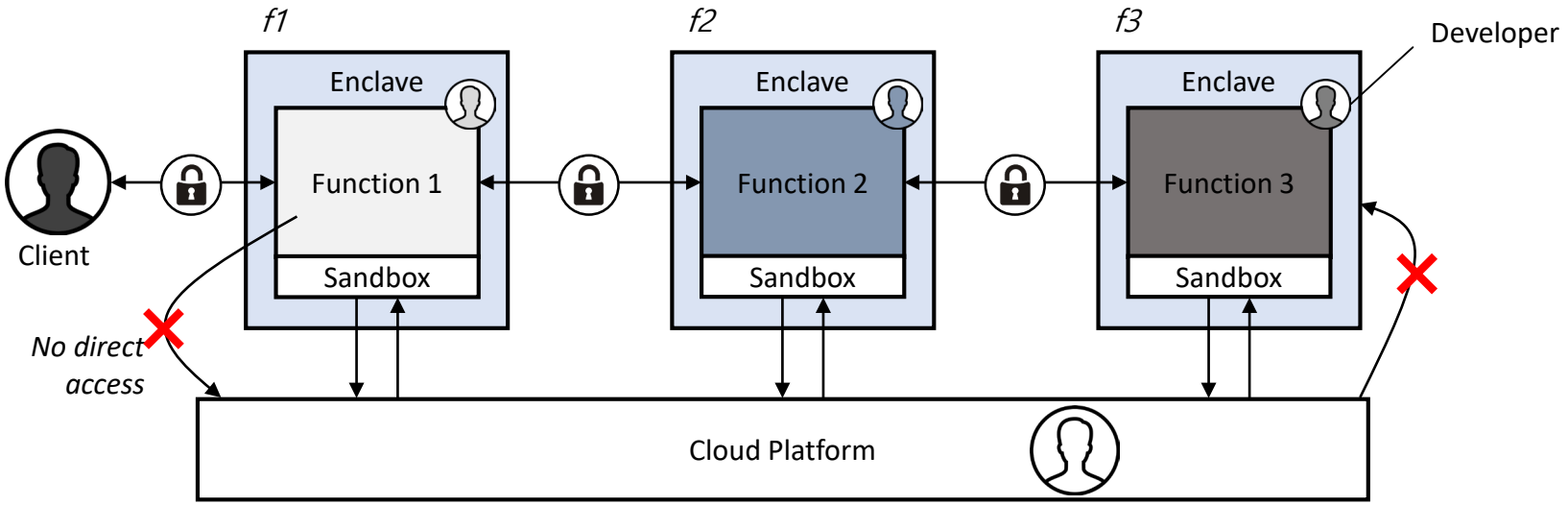
Hierarchical isolation for multilevel security



Bi-directional Isolation for trusted sandboxes

Propose Idea

- All enclaves with different privileges are isolated
- Communicate in a secure and effective way
- Hardware extension for trustworthy system level objective



Threat Model

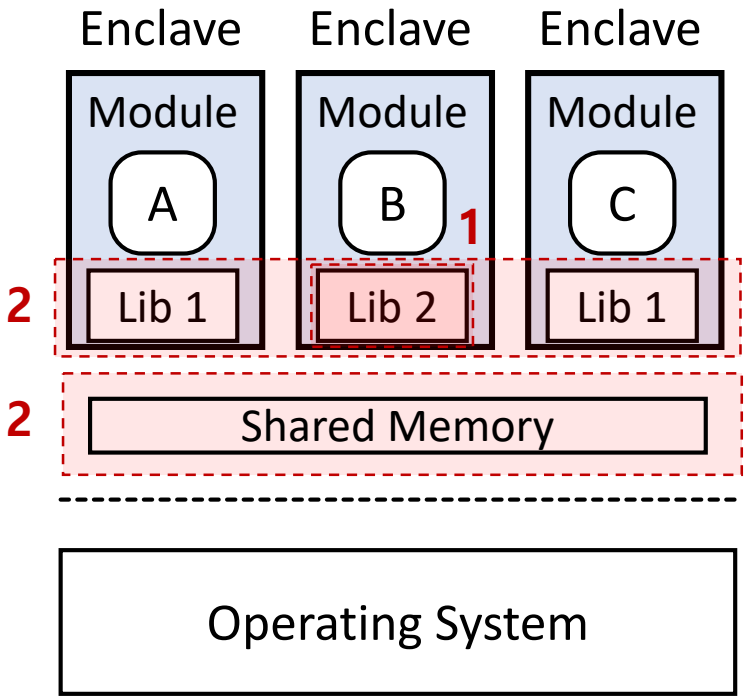
- Trusted Computing Base (TCB)
  - CPU processor
  - Code launched by oneself
  - Code running in monitor
- Protection
  - Confidentiality, Integrity, and freshness of enclave memory
  - Confidentiality, Integrity, and freshness of communication channel, and accounting service
- Attack surface
  - Software attacks from outside the enclave
  - Hardware attacks from outside CPU package
- Out of scope
  - Side channel attacks

Goals

- To propose a new enclave model to have **richer semantics** for **trustworthy cloud service**

- Goal:
  - To support fine-grained compartmentalization and sharing
  - To enhance security for trusted service level objective
  - To improve the performance of trusted cloud computing

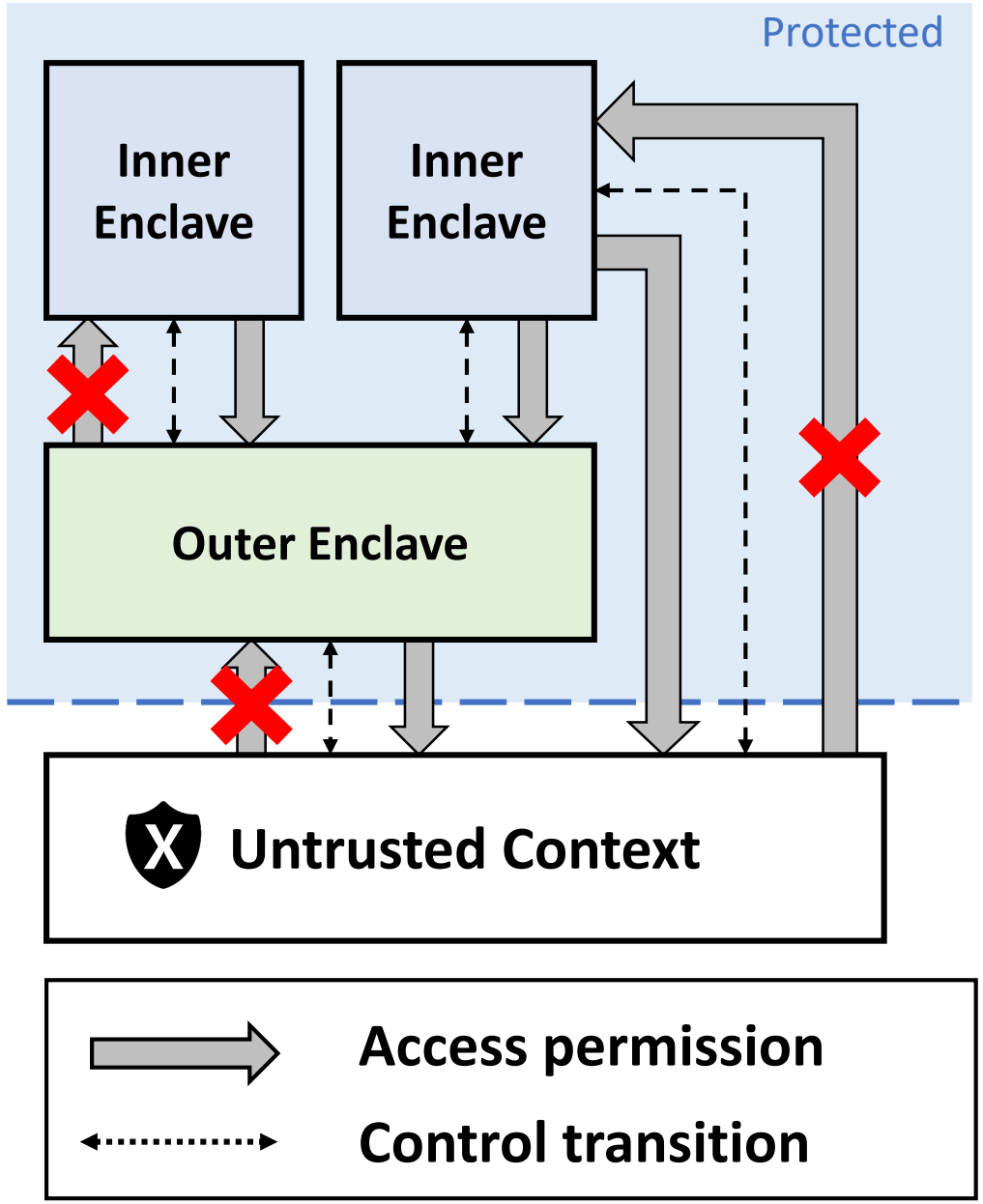
Target Scenario



- Challenge 1: Monolithic Design
  - Potentially malicious third-party libraries
- Challenge 2: Lack of Sharing Semantic
  - Wasted memory by not supporting shared library

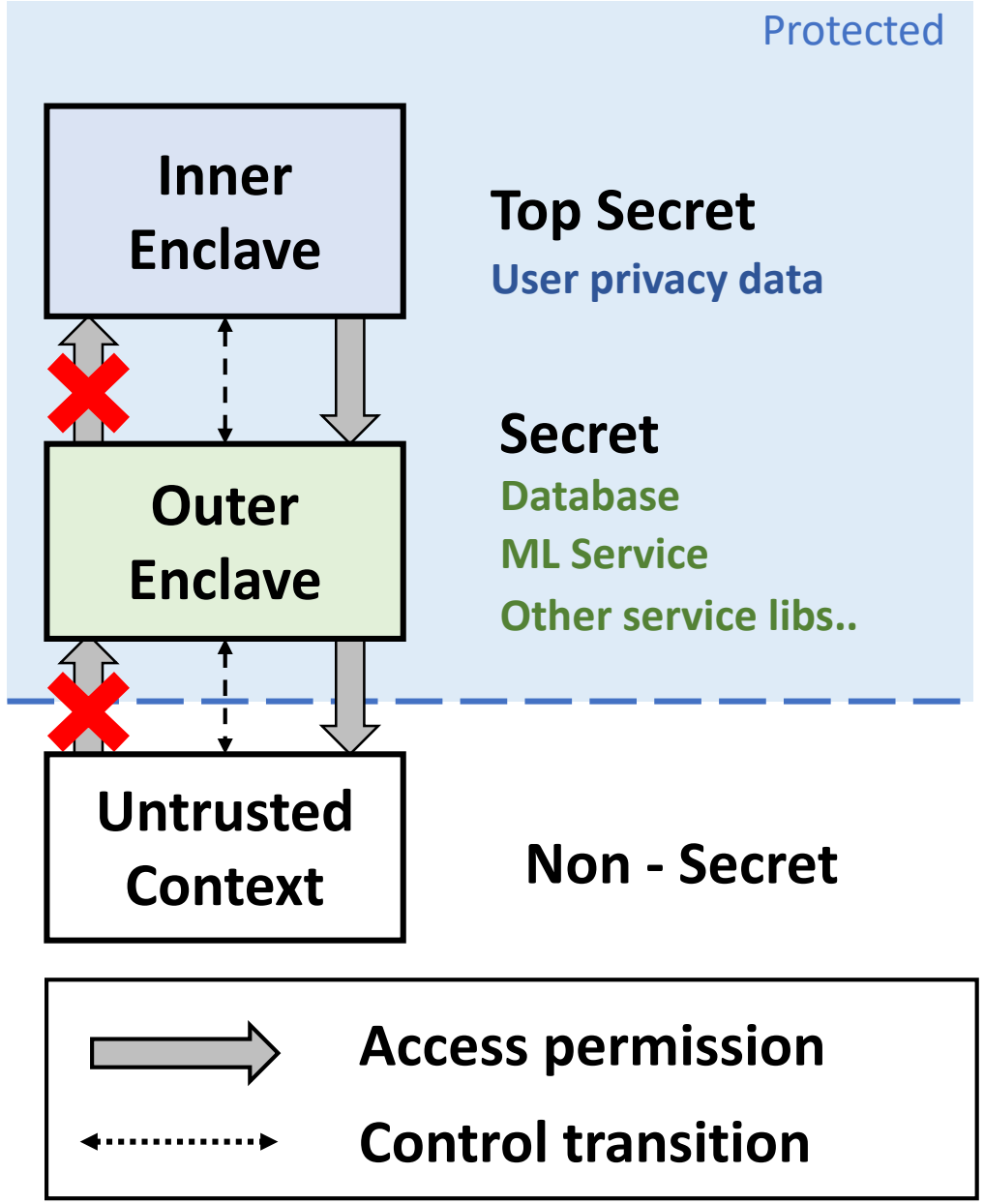
Nested Enclave (ISCA 20)

- Our prior work for Challenge 1 and 2
- Hardware extension to SGX
- Compartmentalization
  - Isolated peer compartments
- Hierarchical Isolation
  - Supports multi-level security
- Sharing lower compartment
  - Shared library
  - Communication channel



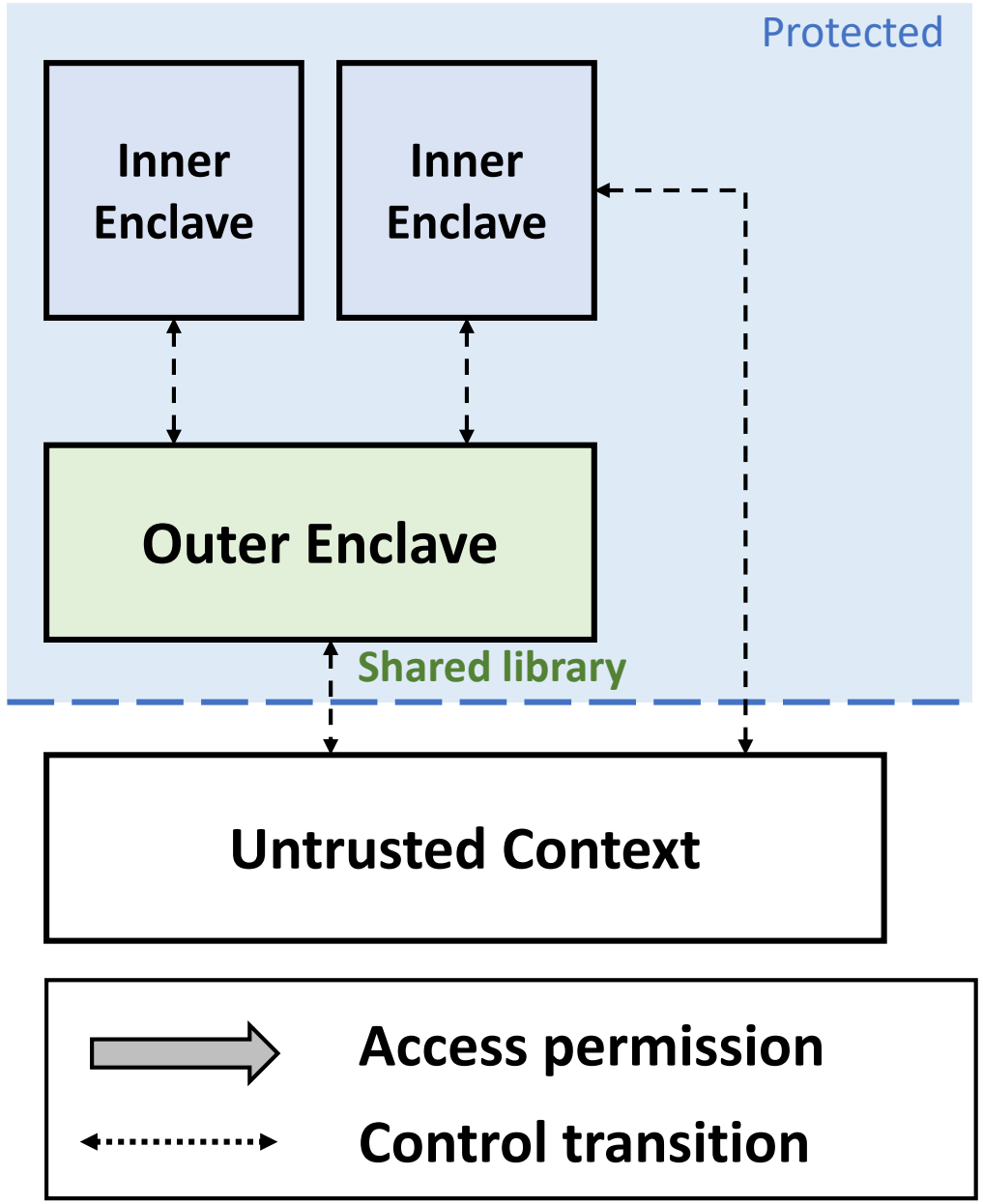
Nested Enclaves (ISCA 20)

- Hierarchical Isolation
  - Outer enclave doesn't have access permission to inner enclaves
  - Inner enclave has access permission to the lower level
- Multi Level Security support
  - Map top secret to inner enclave
  - Map secret to outer enclave

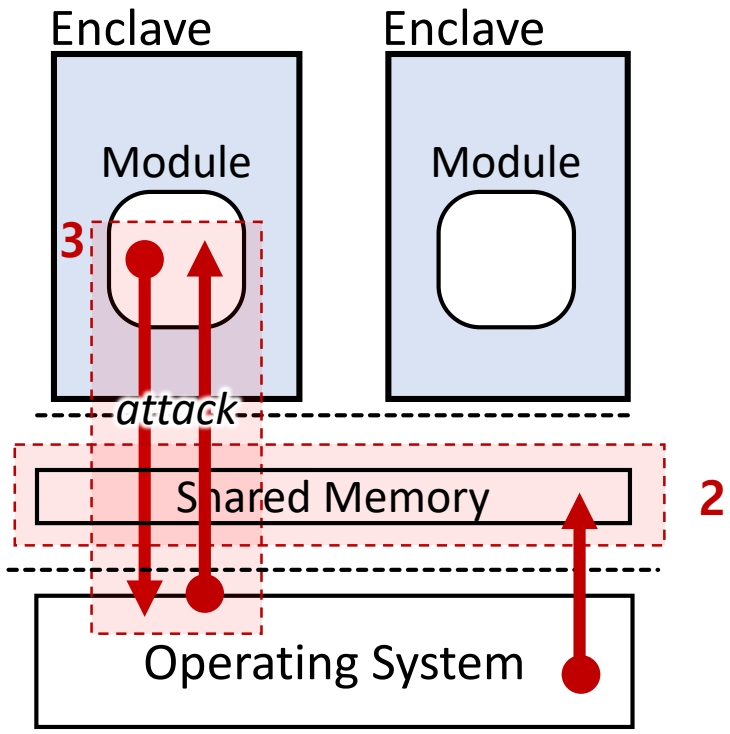


Nested Enclaves (ISCA 20)

- Shared Library
  - Reduced total memory usage
  - Reduced TCB to Inner Enclave
- Secure transition between Enclaves

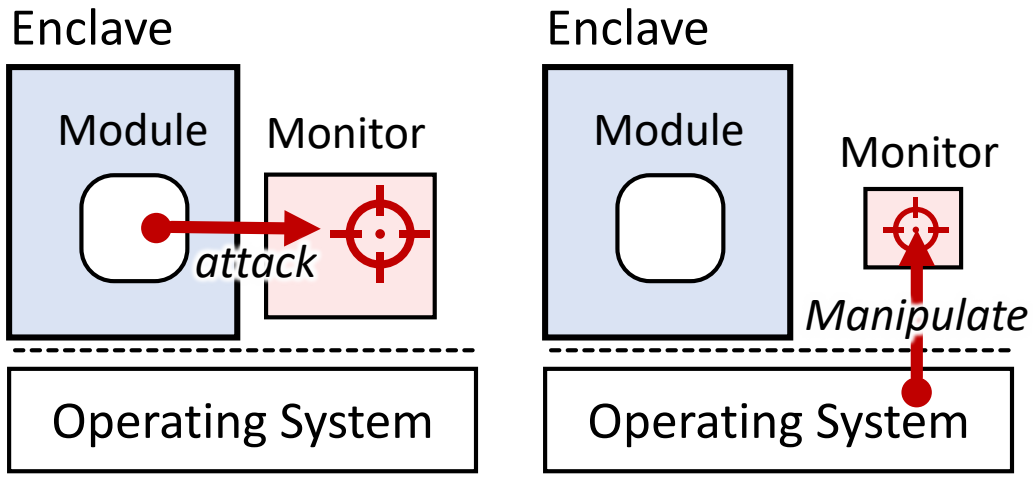


New Target Scenario



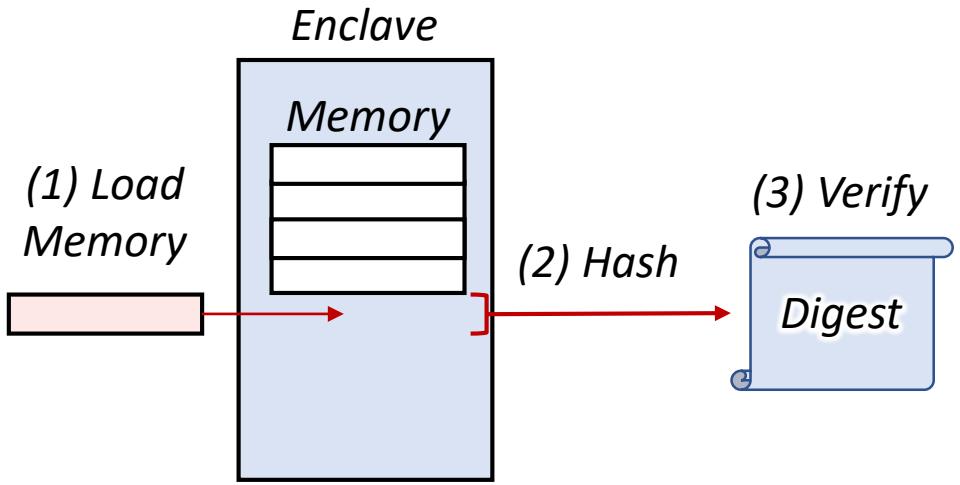
- Challenge 3: Potentially vulnerable module
  - Code running in an enclave can be malicious
- Challenge 2: Lack of Sharing Semantic
  - Sharing through unprotected memory might be vulnerable

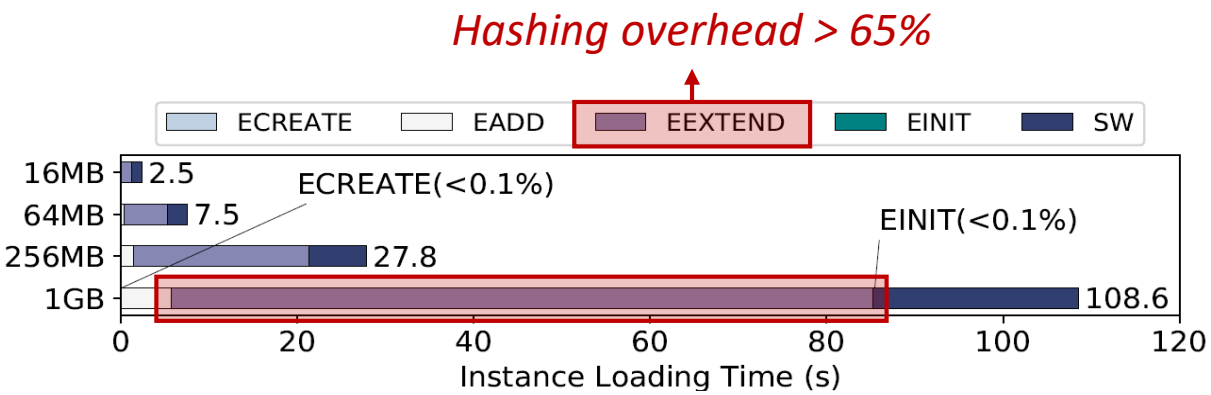
New Target Scenario



- Challenge 4: Securing monitor
  - Sandbox Monitor should be protected by both cloud provider and module provider
- Challenge 5: Trusted Service Level Objective (SLO)
  - Cannot fully trust billing measured by others

New Target Scenario

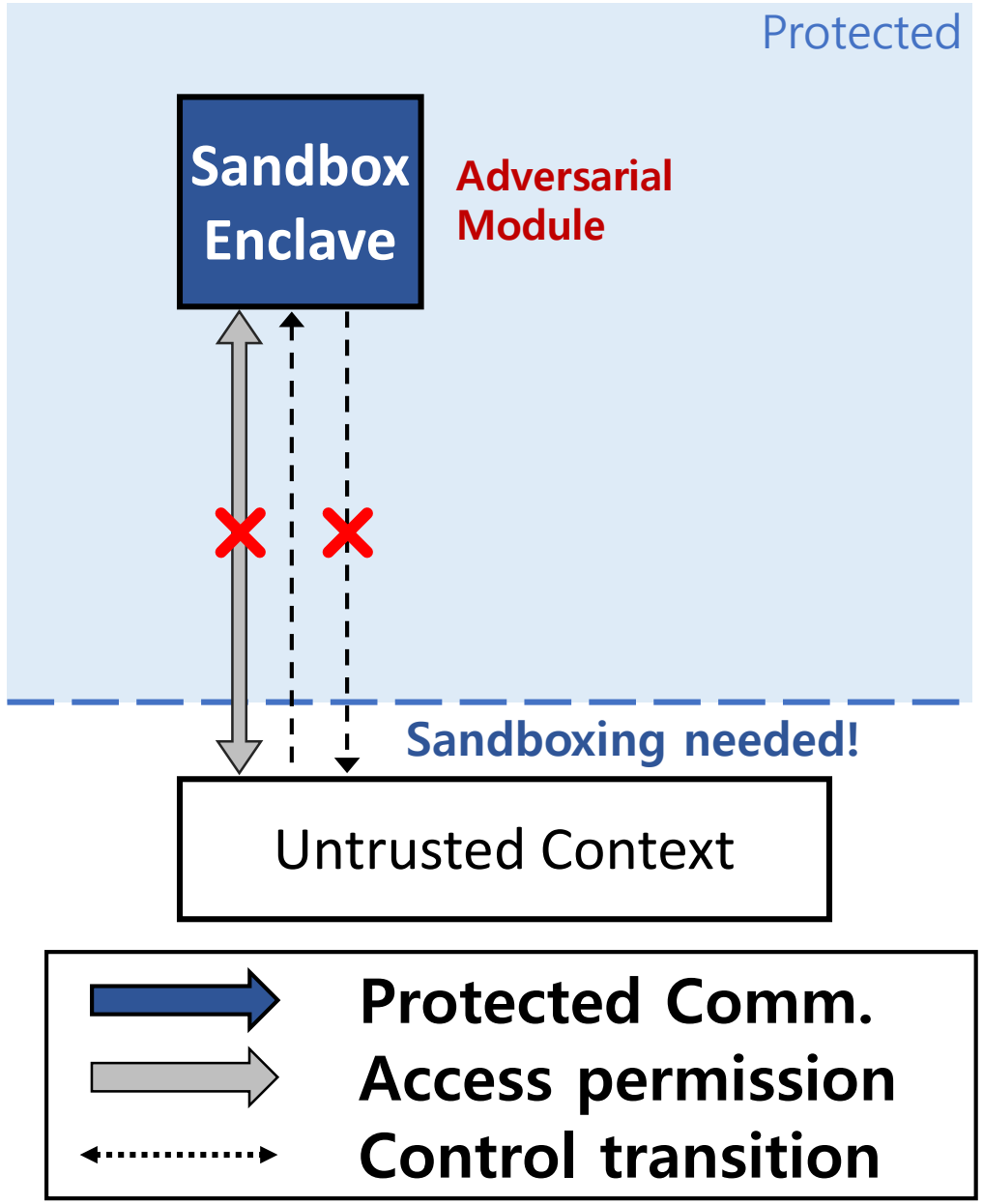




- Challenge 5: Trusted SLO
  - Massive verification slows down enclave instance loading
- Verification is done by hashing the entire memory and comparing with compile time measurement

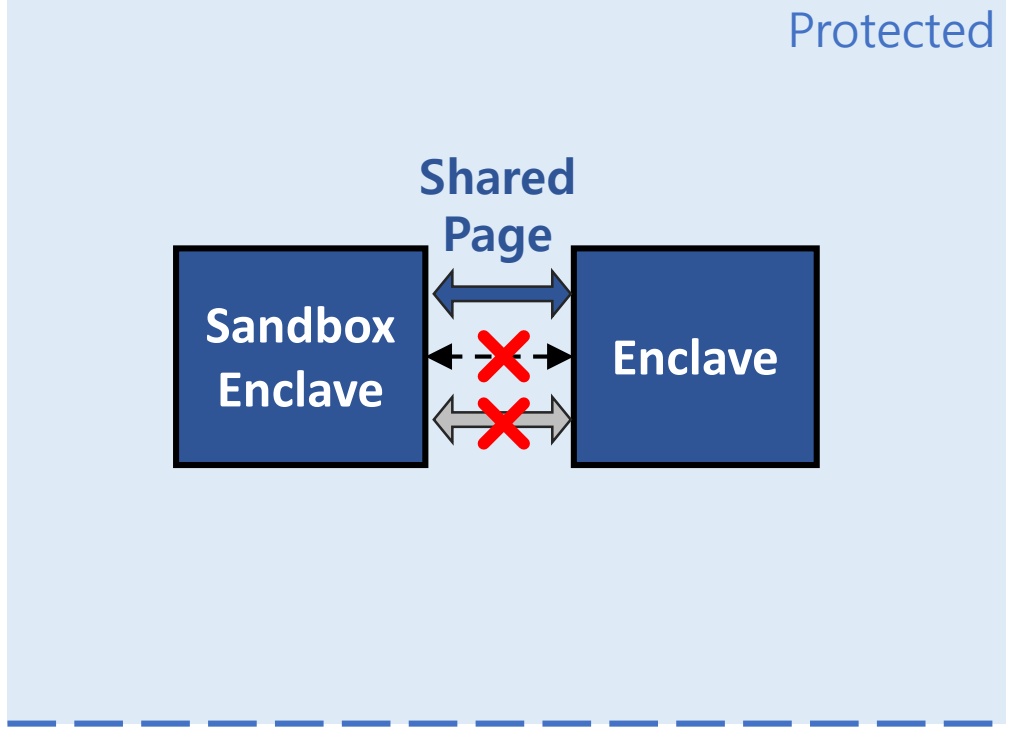
Cloister (This work)

- Sandbox Enclave
  - Mutually isolated Enclave
  - No access permission to untrusted context
  - Cannot perform control transition to untrusted context
  - Hardware-managed signal handler



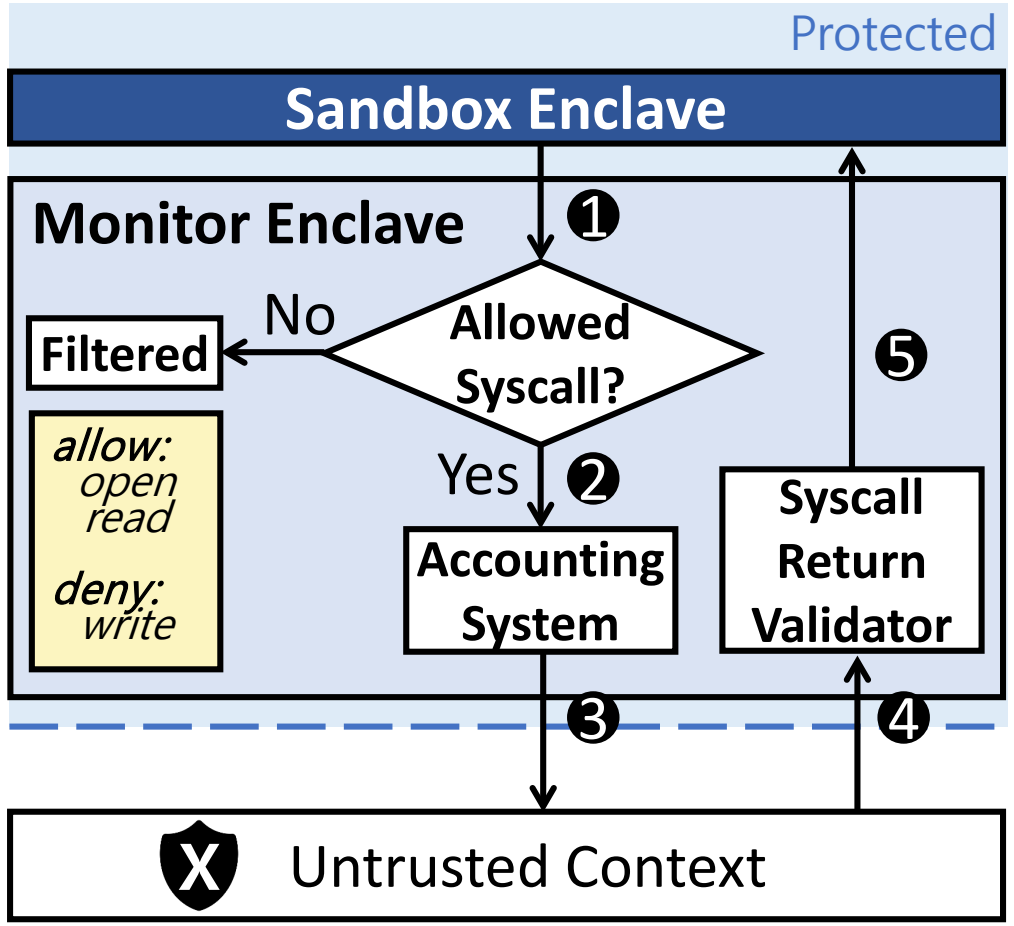
Cloister (This work)

- Page-granularity Sharing
  - Share a page between Enclaves
  - The shared page only allows Read/Write permission
  - Faster IPC through cache
    - Skipping encryption



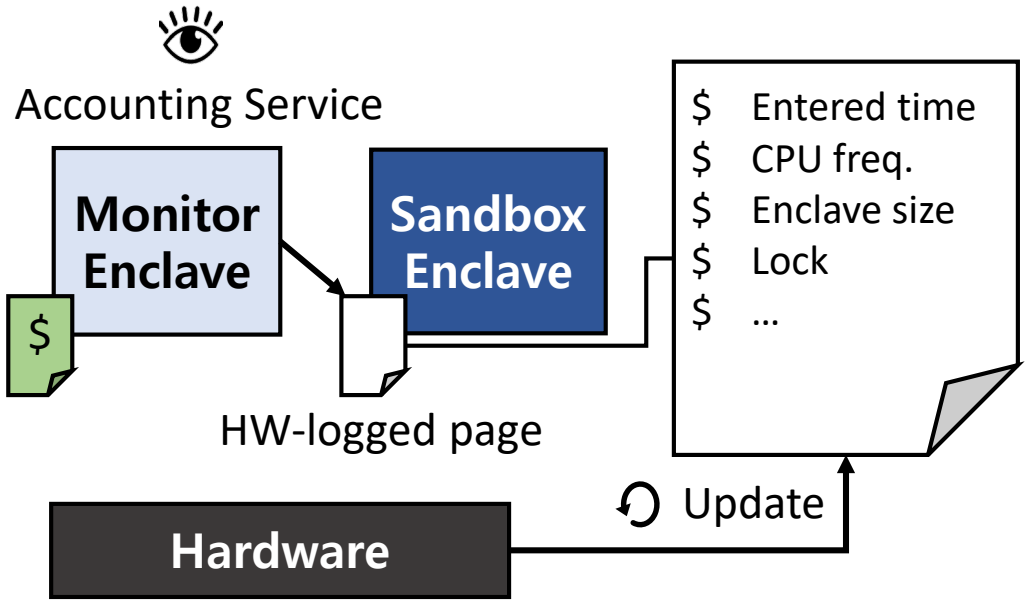
Cloister (This work)

- Monitor Enclave
  - Mutually trusted by attestation
  - Filters system call requests
  - Accounting system
  - Validates system call returns



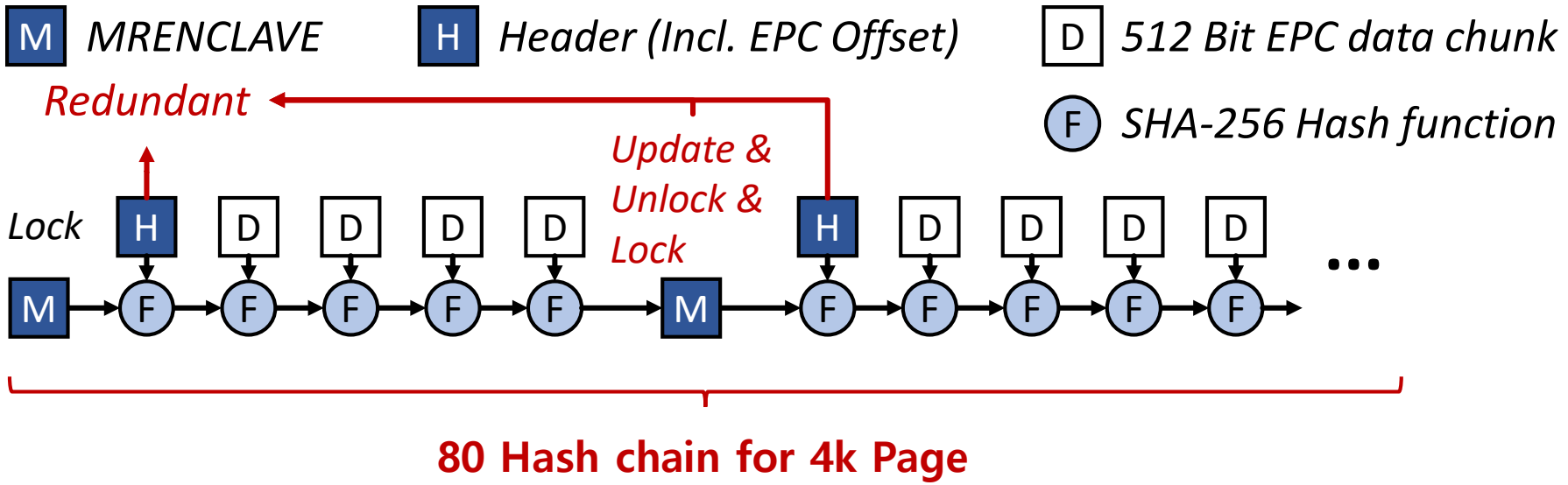
Design – Trusted Resource Accounting

- Tracking CPU/MEM usage
  - Measure in-enclave CPU time
    - Time Stamp Counter (TSC)
    - Frequency (Power state)
  - Track enclave memory allocation
    - SGX Instructions



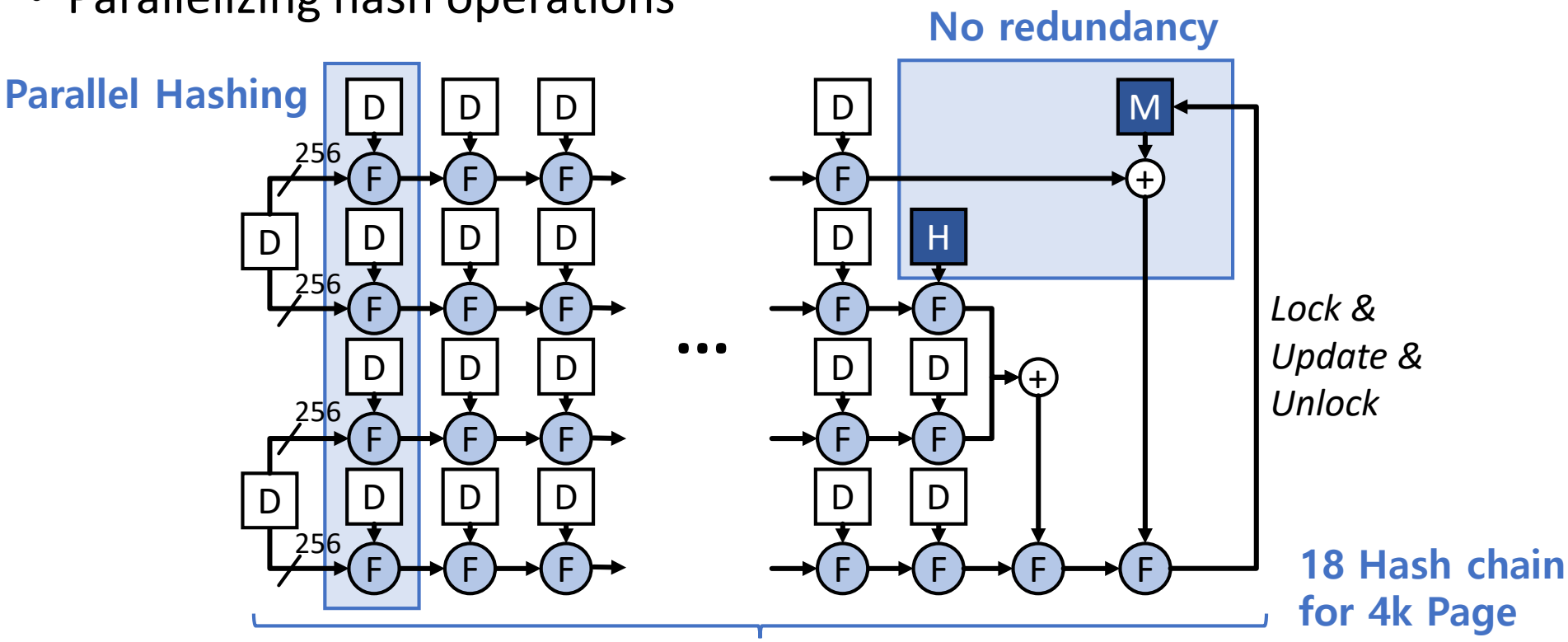
Design (1/2) - Fast Measurement

- Original SGX measurement for a page (4KB)
  - Redundant operations
  - Serialized hash operations



Design (2/2) - Fast Measurement

- Suggested SGX measurement for a page (4KB)
  - Eliminating redundant operations
  - Parallelizing hash operations



Evaluation Questions

- Security benefits from isolation
  - CASE 1: Heart Bleed Attack
- Performance in serverless computing scenario
  - CASE 2: Query Server
- Latency distribution with trustworthy resource accounting
  - CASE 3: Secure Accounting with FTPS Server
- Loading time acceleration from hardware support
  - CASE 4: Launching Enclave Instances

Methodology

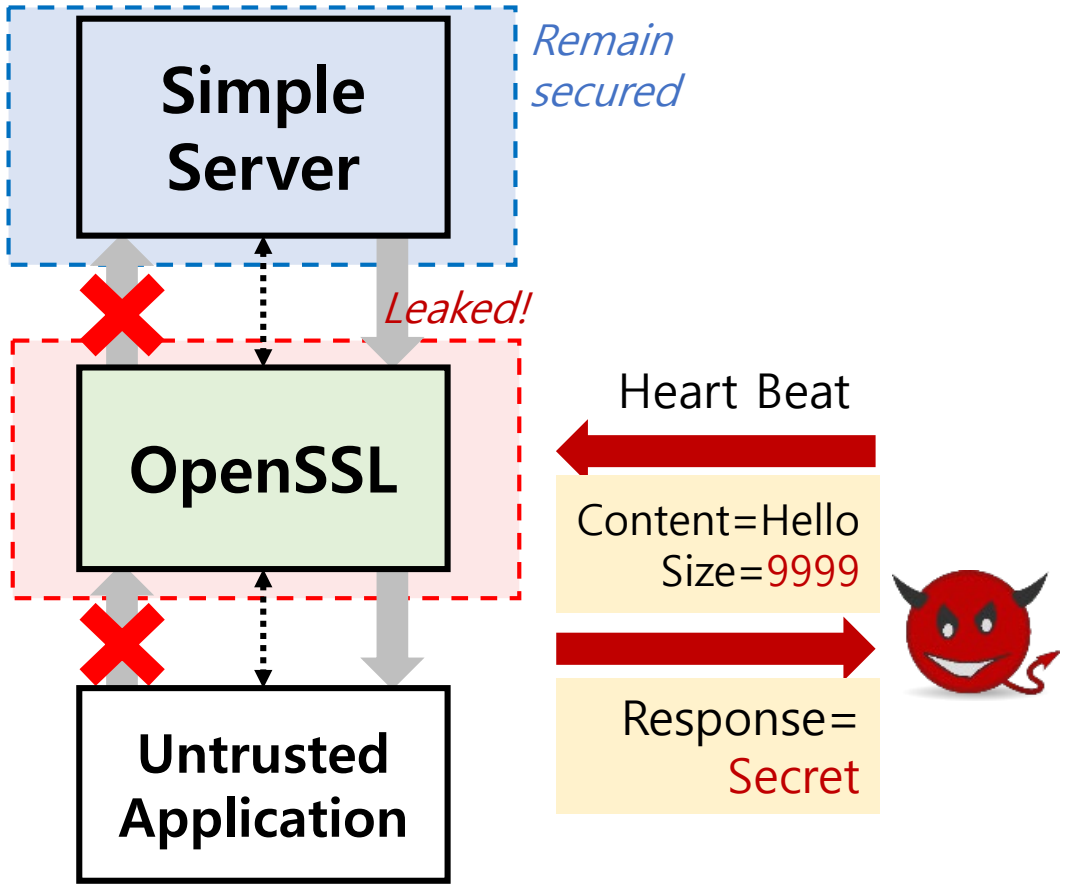
Evaluation Environment

- Intel i7-7700, 64 GB DRAM
- Ubuntu 18.04
- Intel SGX Linux SDK/Driver v2.2
- ZSim / Dramsim2

Implementation

- New instructions in SDK / Driver for emulation
- Porting applications to use new instructions
- Micro architecture simulation

CASE 1: Heartbleed Attack



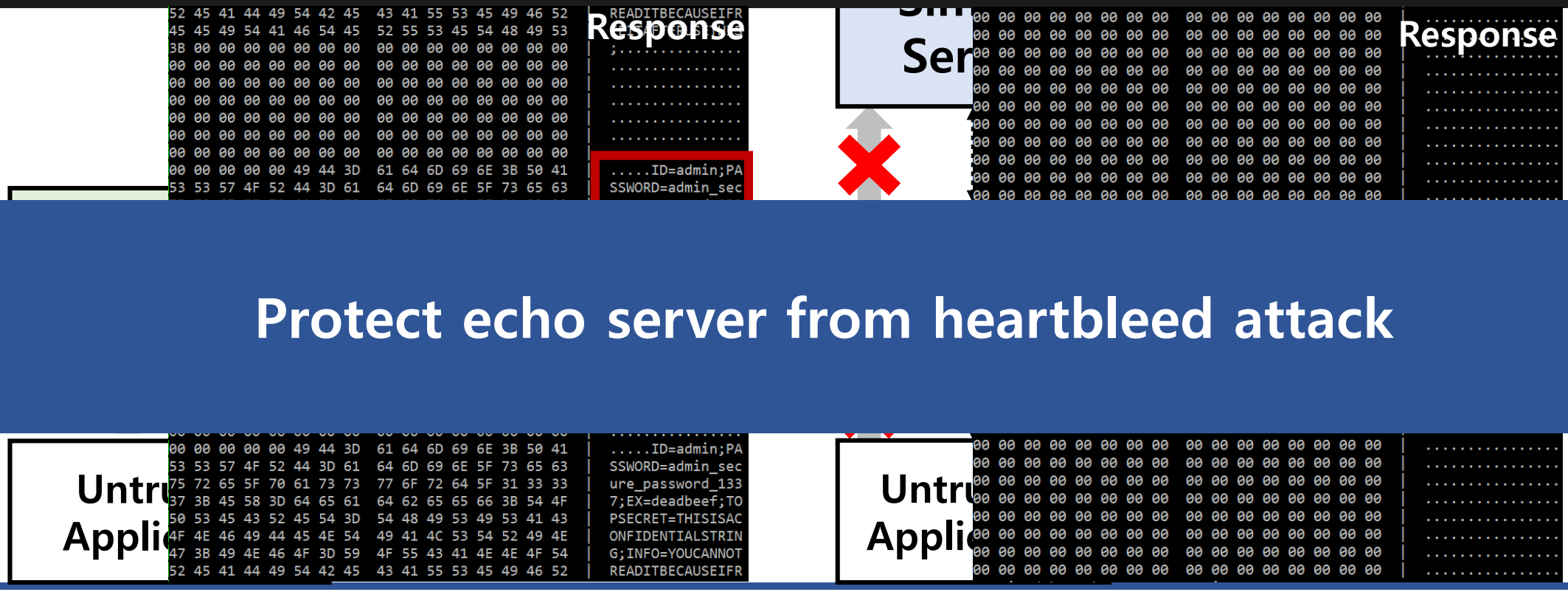
- Goal:
  - To show **security benefits** from isolation
- Attacking Echo server with vulnerability in OpenSSL

CASE 1: Heartbleed Attack

In Simple Server

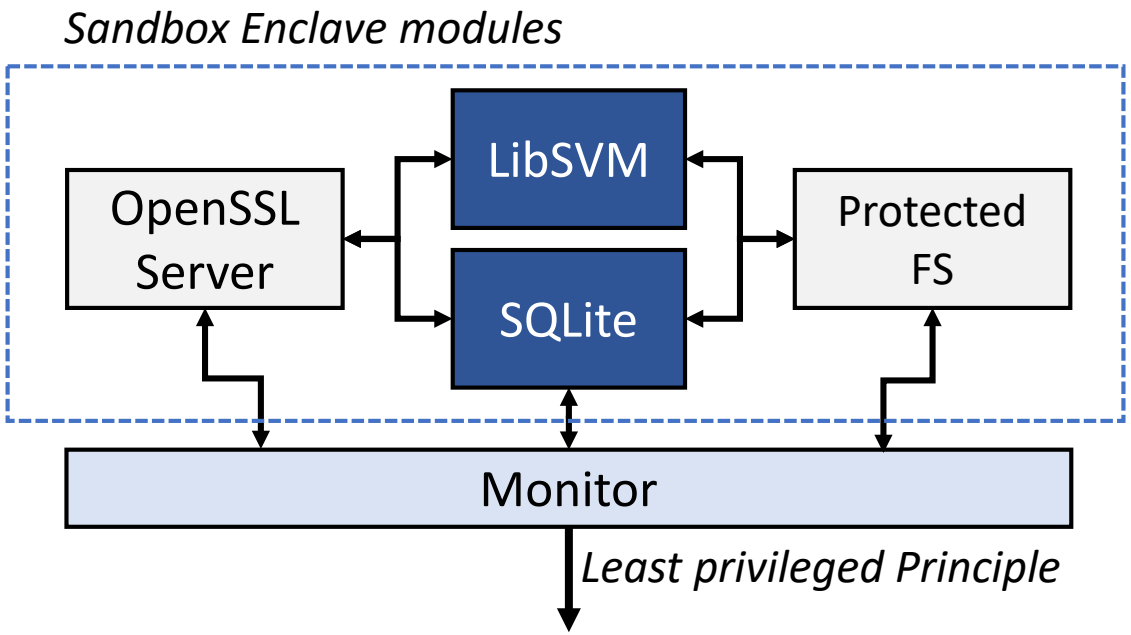
```
void info_leak()
{
    char *secret;
    int size = 0x8000;

    secret = (char *) malloc (size);
    strlcpy (secret + i * 0x100, "ID=admin;PASSWORD=admin_secure_password_1337;EX=deadbeef;TOPSECRET=THISISACONFIDENTIALSTRING;INFO=YOUCANNOTREADITBECAUSEIFREEITAFTERUSETHIS;", 0x100);
    free(secret);
}
```



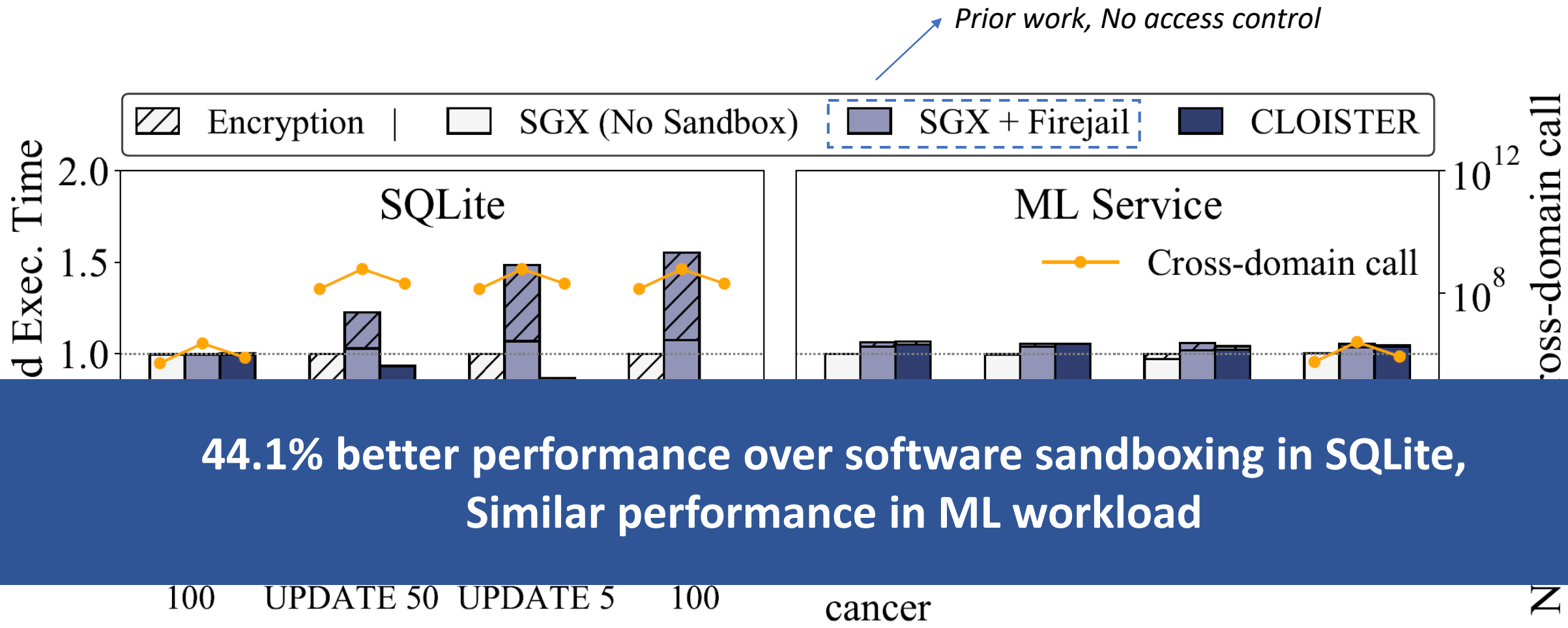
Protect echo server from heartbleed attack

CASE 2: Query Server



- Goal:
  - To show **performance** implications in serverless computing scenario
- A query server scenario with SQLite / LibSVM
- Least privilege support
  - E.g., SQLite cannot perform file I/O

CASE 2: Query Server

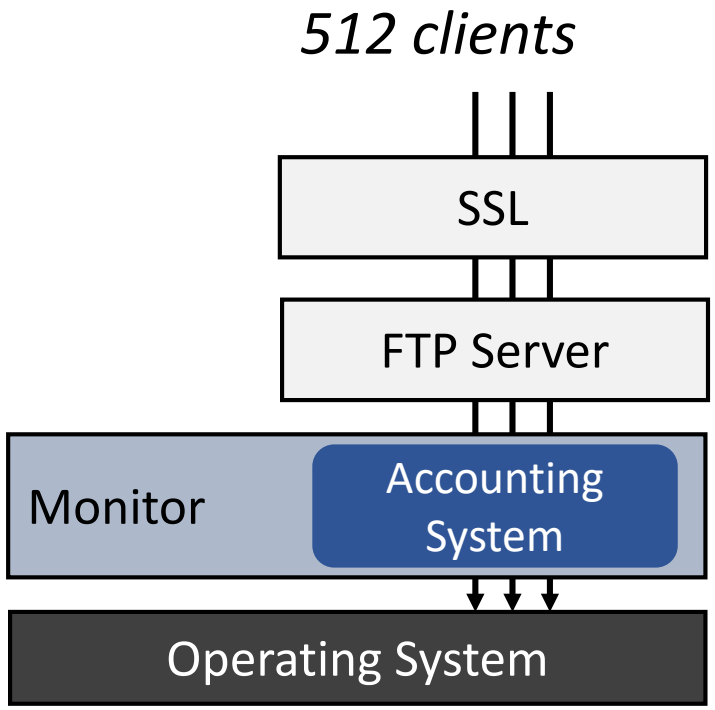


44.1% better performance over software sandboxing in SQLite,
Similar performance in ML workload

YCSB 10000 Queries

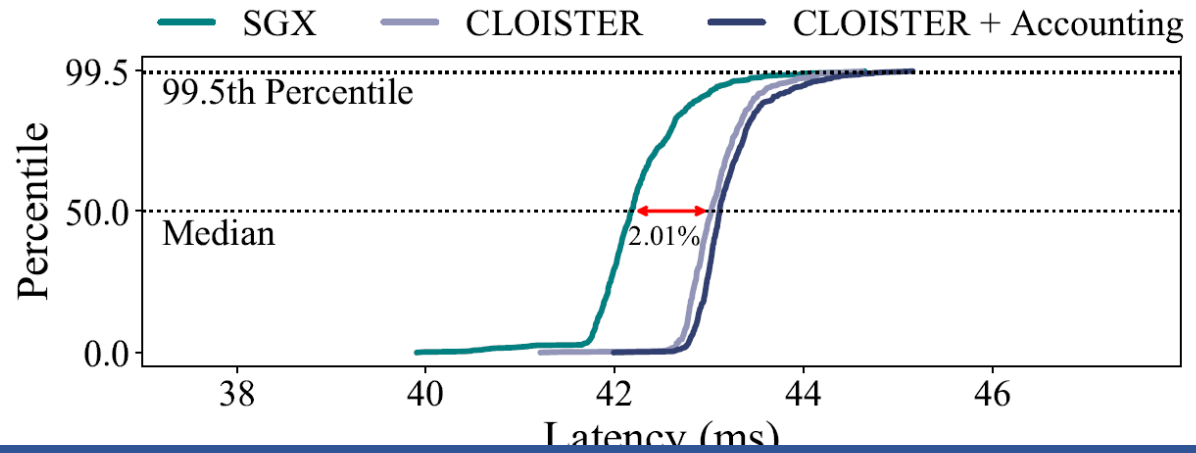
Supported vector machine datasets, 10000 Queries

CASE 3: Secure Accounting with FTPS Server

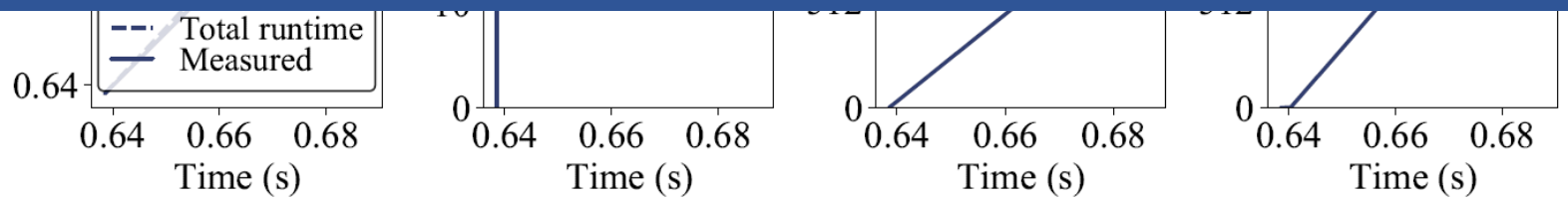


- Goal:
  - To show **latency distribution** along with trustworthy resource accounting
- FTPS Server with secure accounting
- 512 Clients send requests to the server for a 1MB encrypted file

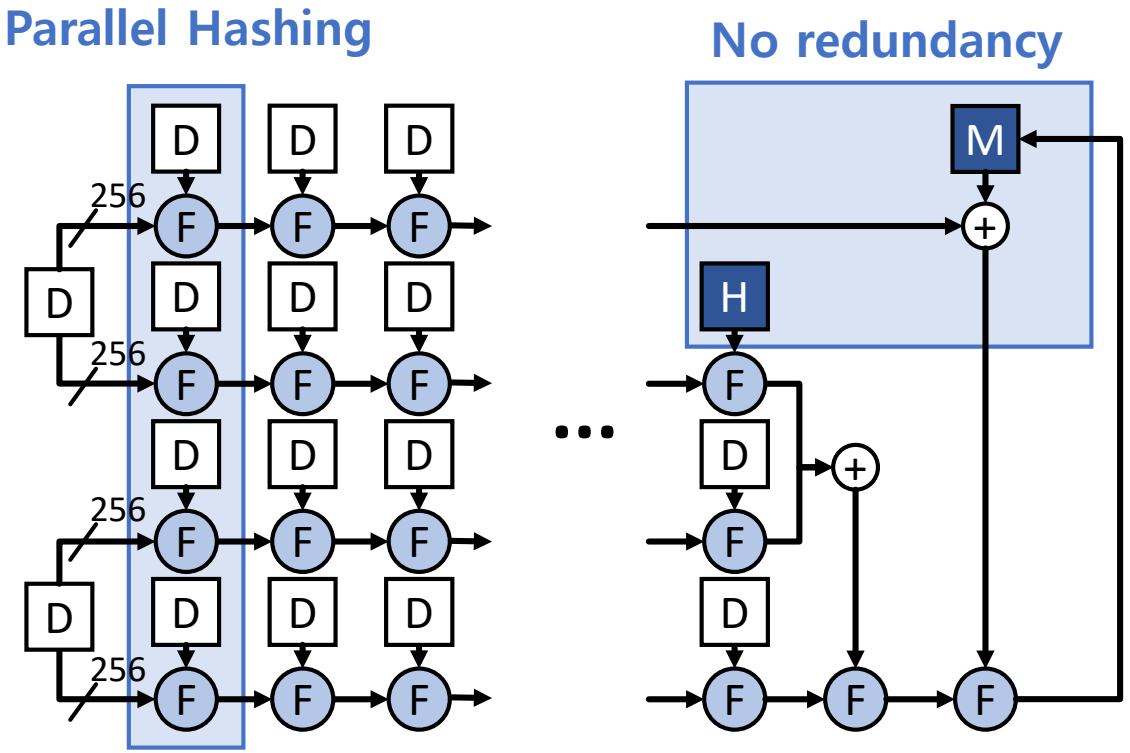
CASE 3: Secure Accounting with FTPS Server



Overhead is small (<2%)

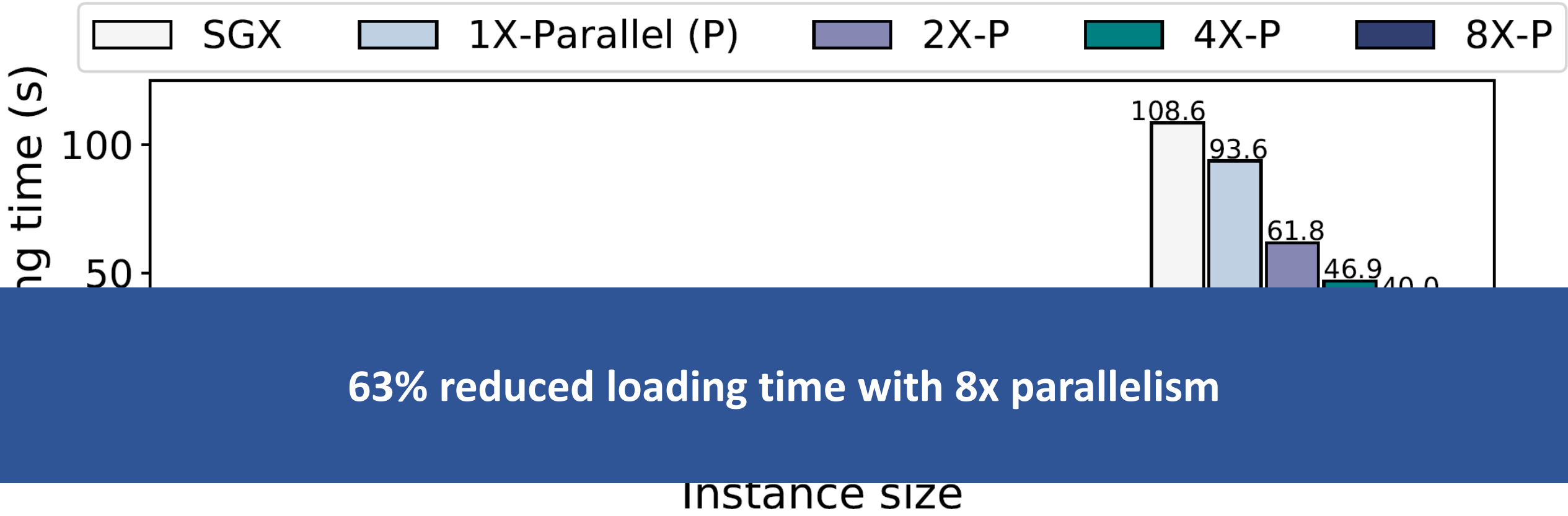


CASE 4: Launching Enclave Instances



- Goal:
  - To show **instance loading** acceleration with hardware support
- Launching enclave instances with various sizes and levels of parallelism

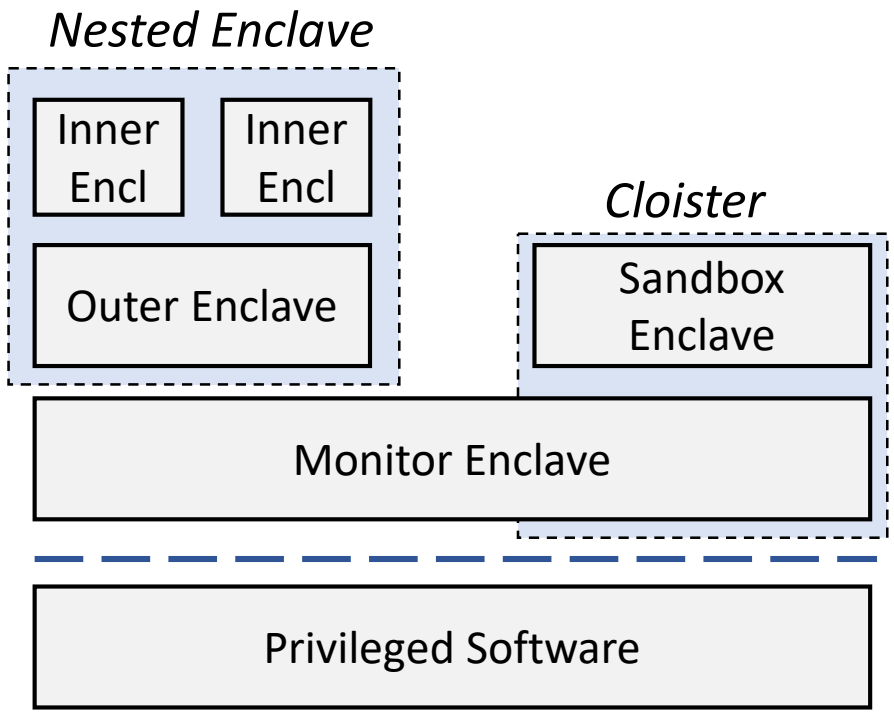
CASE 4: Launching Enclave Instances



63% reduced loading time with 8x parallelism

Summary

- Nested Enclave [ISCA'20]
  - Hierarchical isolation
  - Control Transition
  - 1:N shared Enclave
- Cloister [On review]
  - Bi-direction isolation
  - Message
  - 1:1 shared page
  - Accounting
  - Fast loading



Conclusion

- Investigate the limitations of the current monolithic enclave design
- To propose a new enclave model to have **richer semantics** for **trustworthy cloud service**
  - Support fine-grained compartmentalization and sharing
  - Enhance security for trusted service level objective
  - Improve the performance of trusted cloud computing
- Evaluate the performance benefits from emulation and simulation

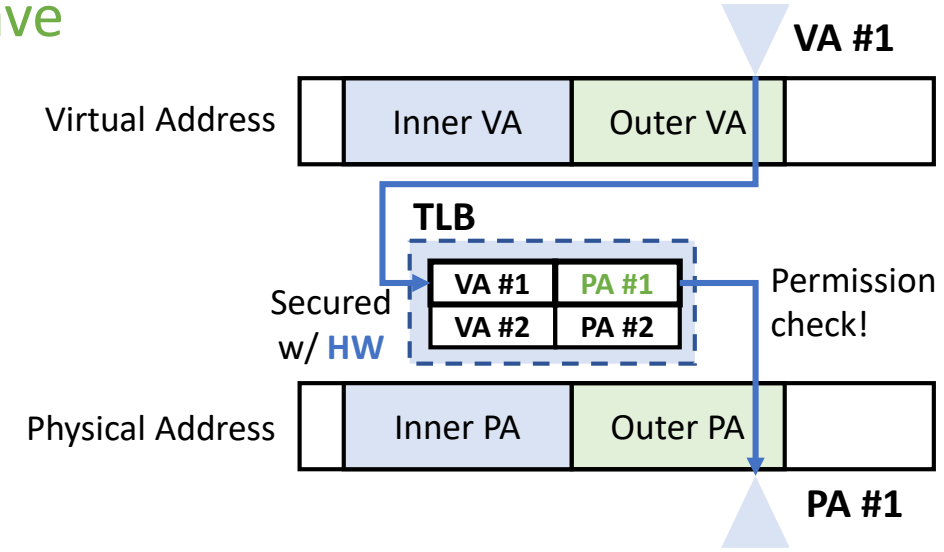
Appendix

- Detailed Access control
- Case study
  - Shared library
- Comparison to prior sandbox

Design (1/3) - Memory access control

- Validate memory access during TLB miss handling
- Invariant for security: TLB must contain only validated translation

- Memory access cases
  - Inner enclave accesses its outer enclave

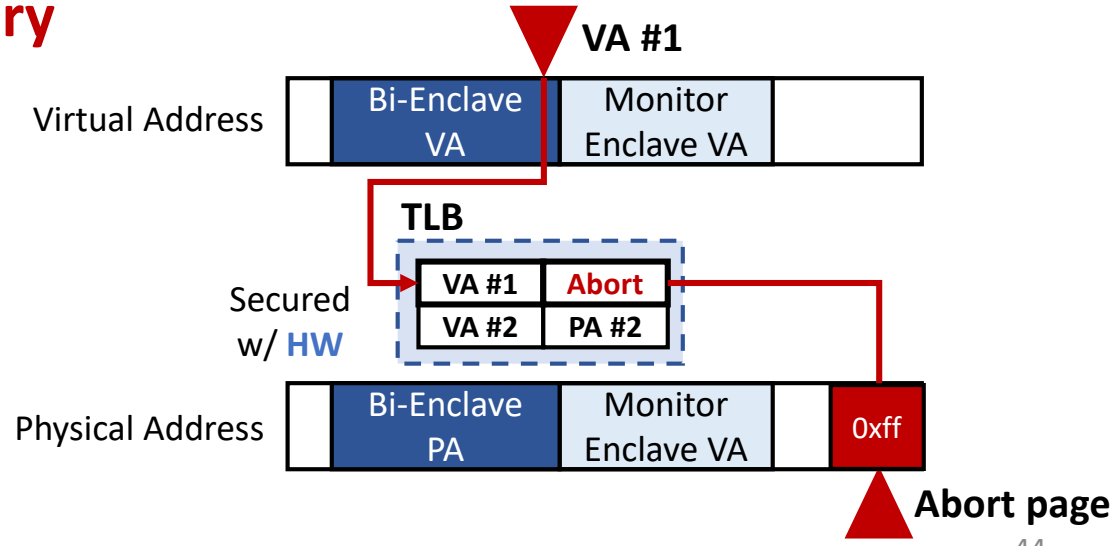


Modification

Design (2/3) - Memory access control

- Validate memory access during TLB miss handling
- Invariant for security: TLB must contain only validated translation

- Memory access cases
  - Bi-Enclave accesses **outside its memory**

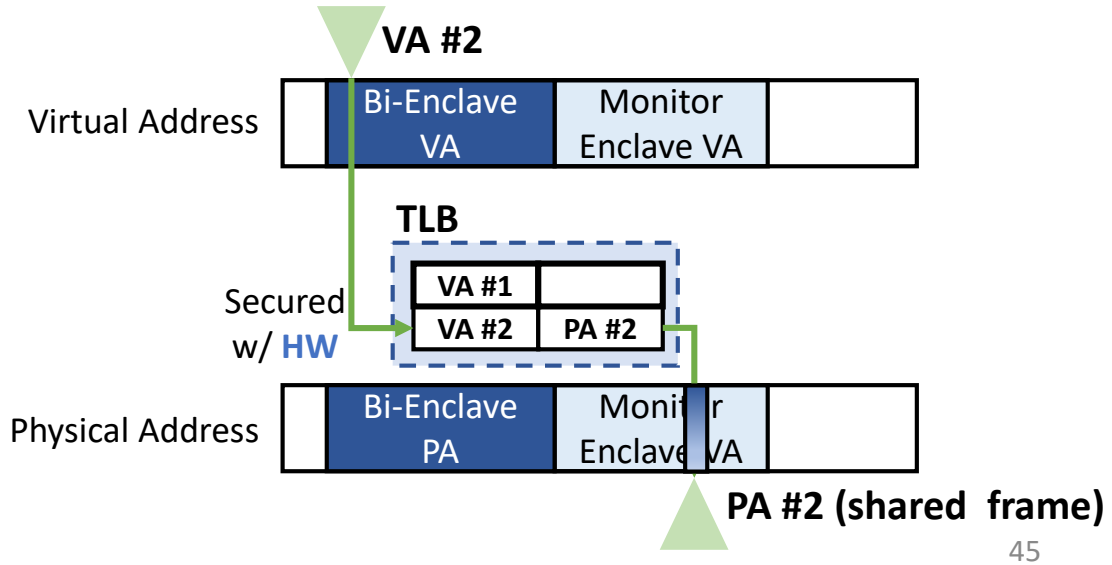


Modification

Design (3/3) - Memory access control

- Validate memory access during TLB miss handling
- Invariant for security: TLB must contain only validated translation

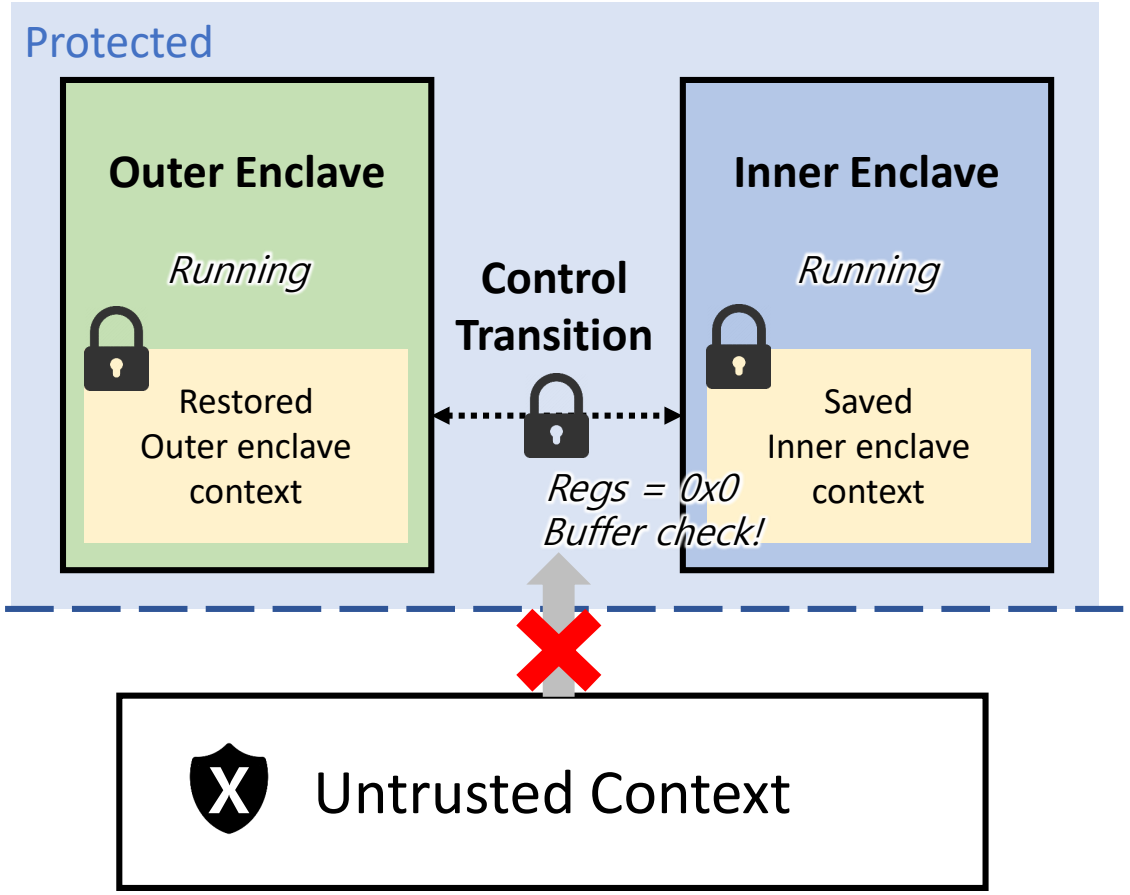
- Memory access cases
  - Bi-Enclave accesses shared frame



Modification

Design (1/1) - Secure control transition

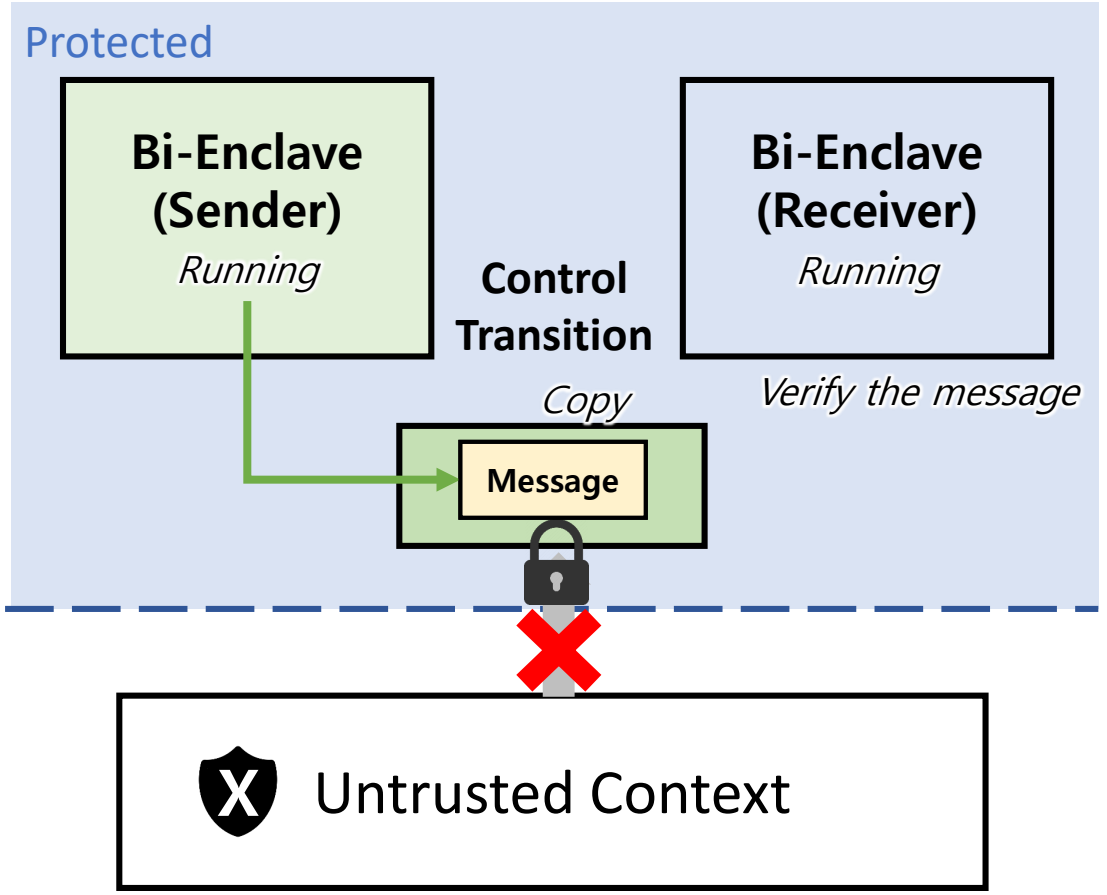
- Transition between Inner and outer enclave
  - Save running context
  - Flush flags, register, and TLB
  - Check & sanitize parameters
  - Restore target context if exists
  - Done with 2 new instructions



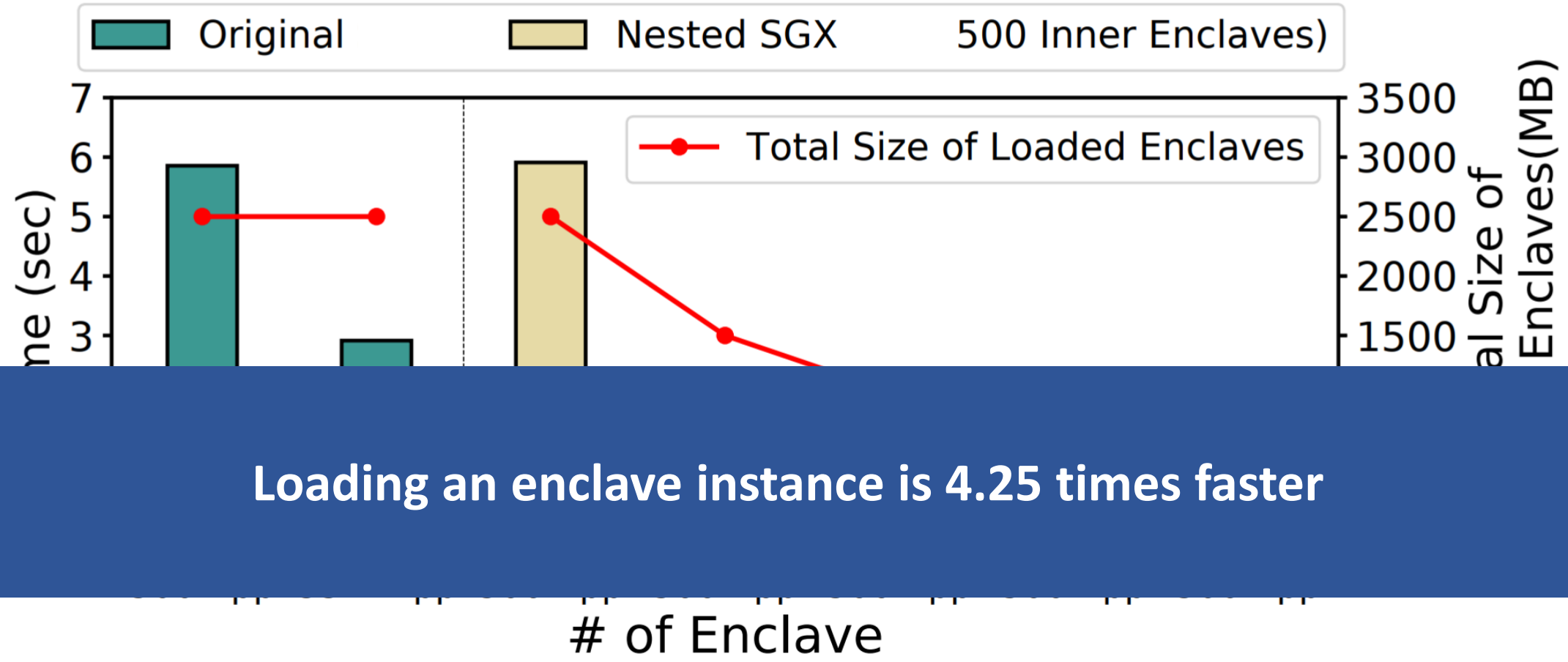
Modification

Design (1/1) - Communication API

- Message based calling communication
  - Copy into private memory
  - Call functions after verification
    - To prevent race attack
    - E.g., Time-of-check-time-of-use (TOCTOU)



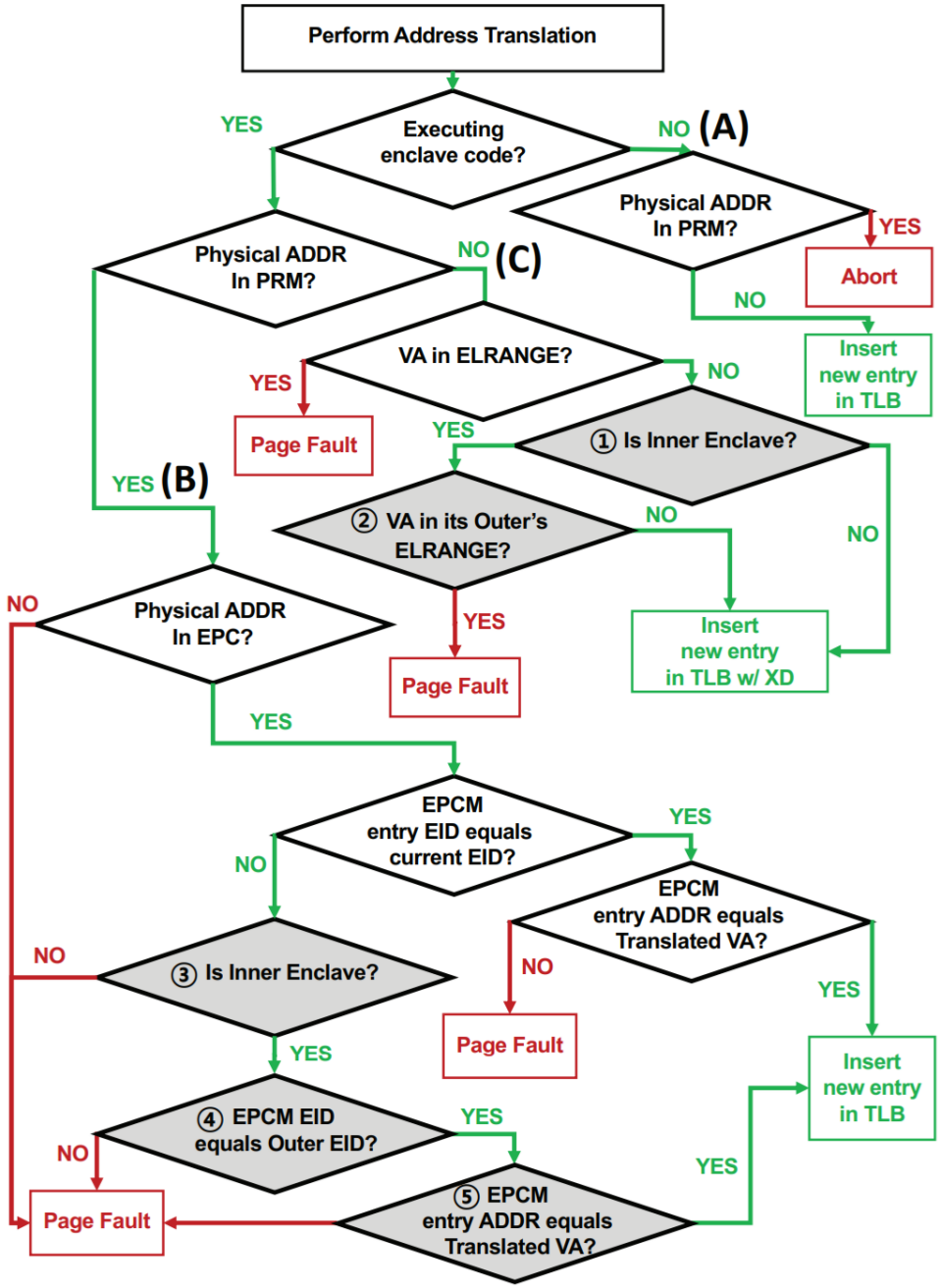
Shared library



Loading an enclave instance is 4.25 times faster

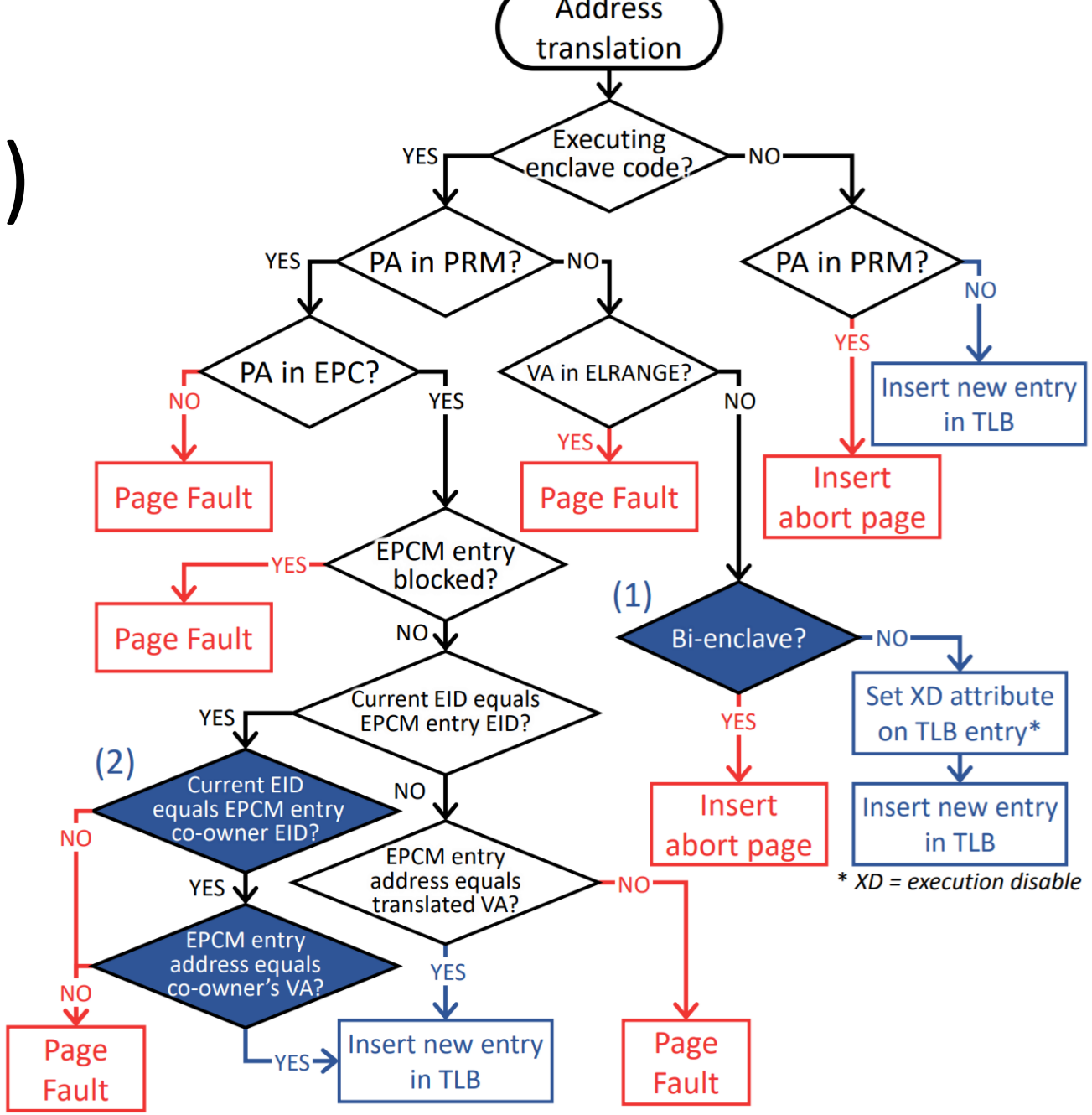
Access Control

- **Modifications are marked grey**

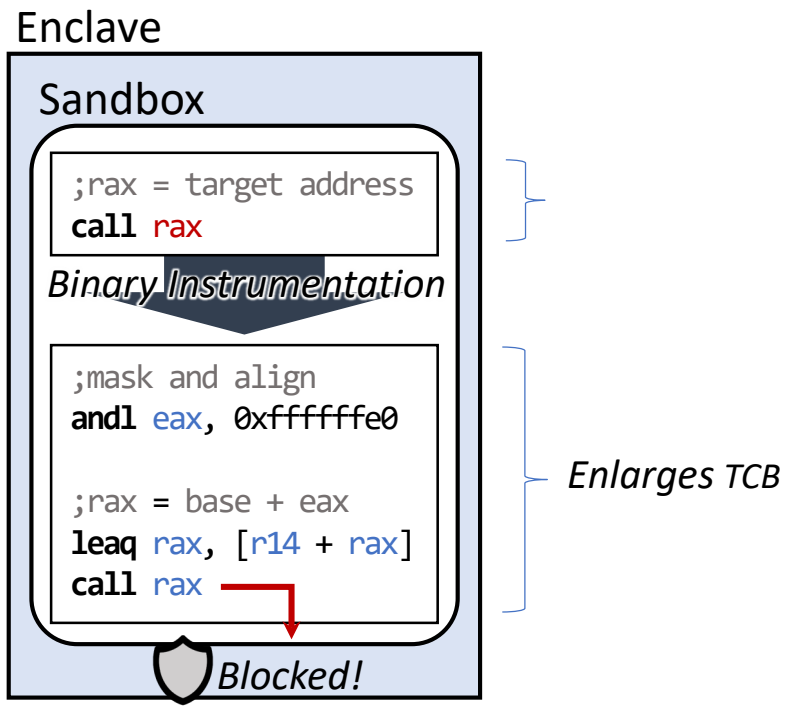


Access Control (detail)

- **Modifications are marked blue**

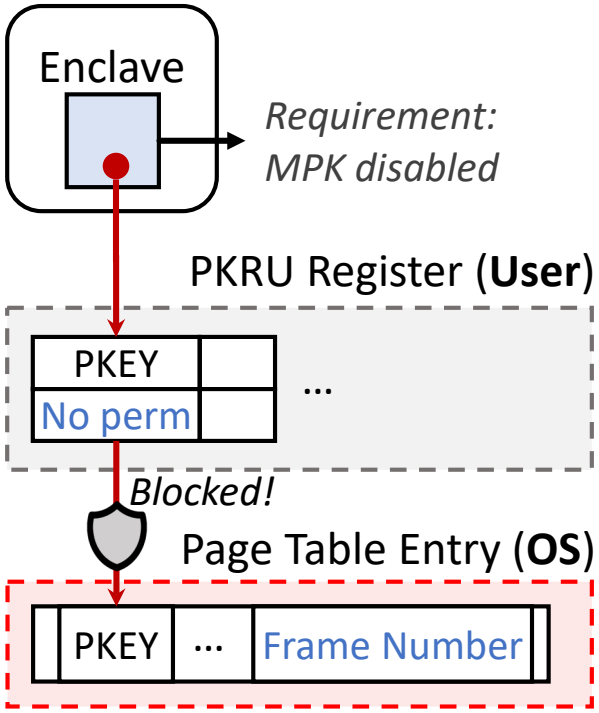


Problem - Access control



- Goal : Preventing to access outside sandbox
- A software based approach
- Pros:
  - No need to change hardware
- Cons:
  - Enlarge TCB
  - Slowdown every memory access
    - ~24%
  - Vulnerable to attacks
    - Bugs, Spectre, Rowhammer,

Problem - Access control



(b) HW+OS confinement

- Goal : Preventing to access outside sandbox
- A hardware+OS approach
- Pros:
  - Fast
  - Smaller TCB
- Cons:
  - Limited number of domains
  - Vulnerable to Rowhammer attacks
  - Still, HW need to be changed