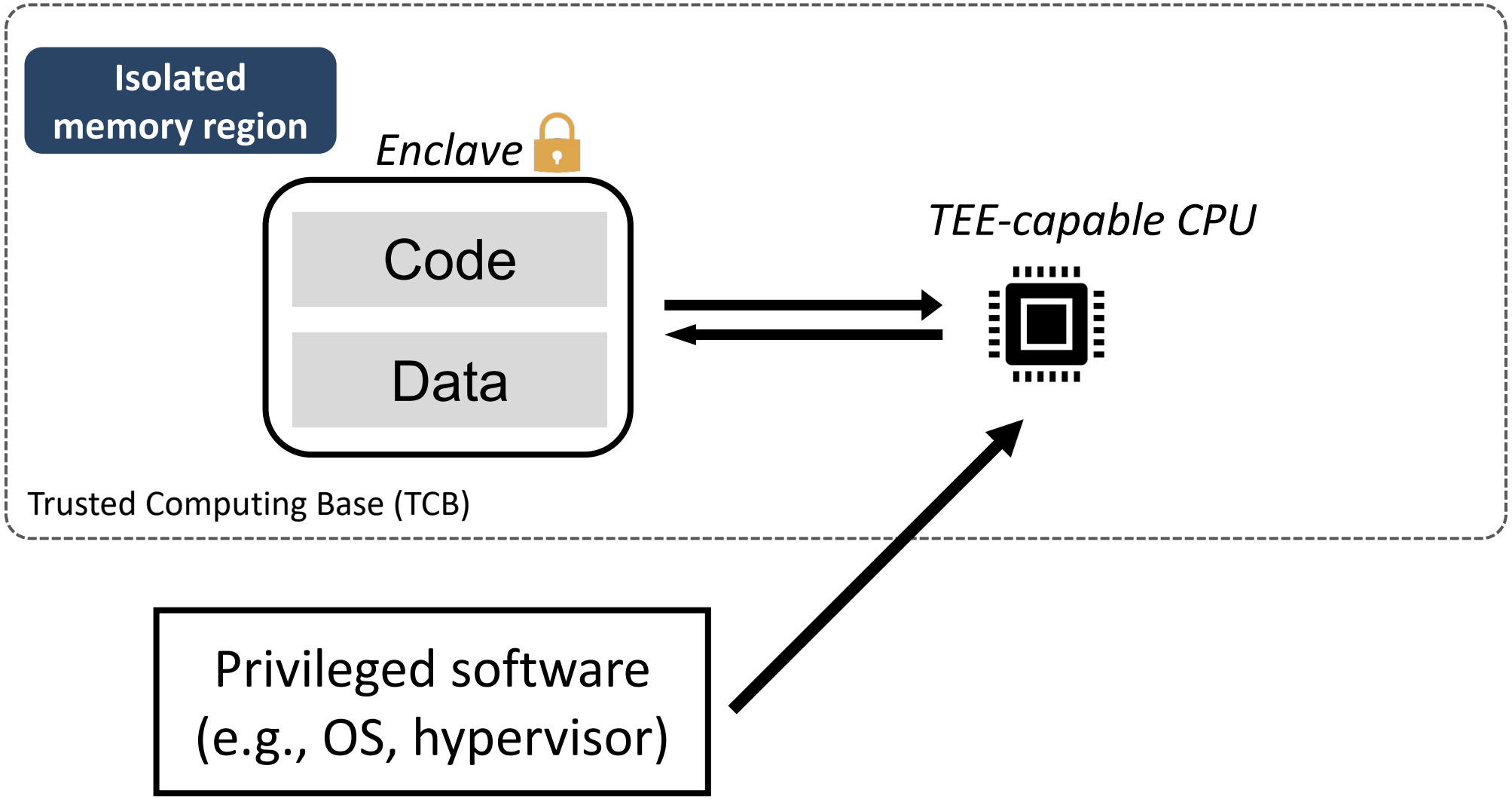
SENSE

Enhancing Microarchitectural Awareness for TEEs via Subscription-Based Notification

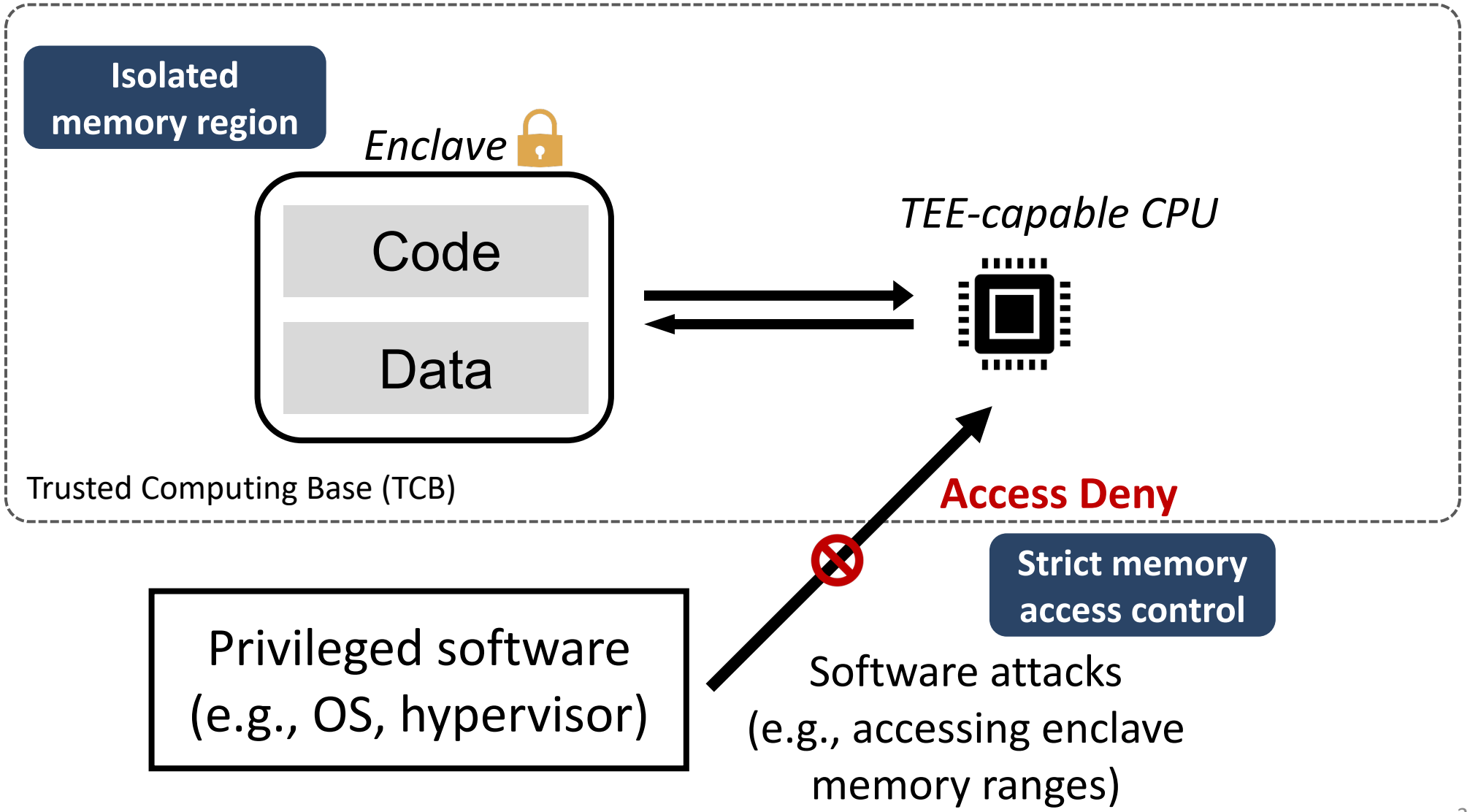
Fan Sang[1], Jaehyuk Lee[1], Xiaokuan Zhang[3], Meng Xu[4],
Scott Constable[2], Yuan Xiao[2], Michael Steiner[2], Mona Vij[2], Taesoo Kim[1]

[1]Georgia Institute of Technology, [2]Intel, [3]George Mason University, [4]University of Waterloo

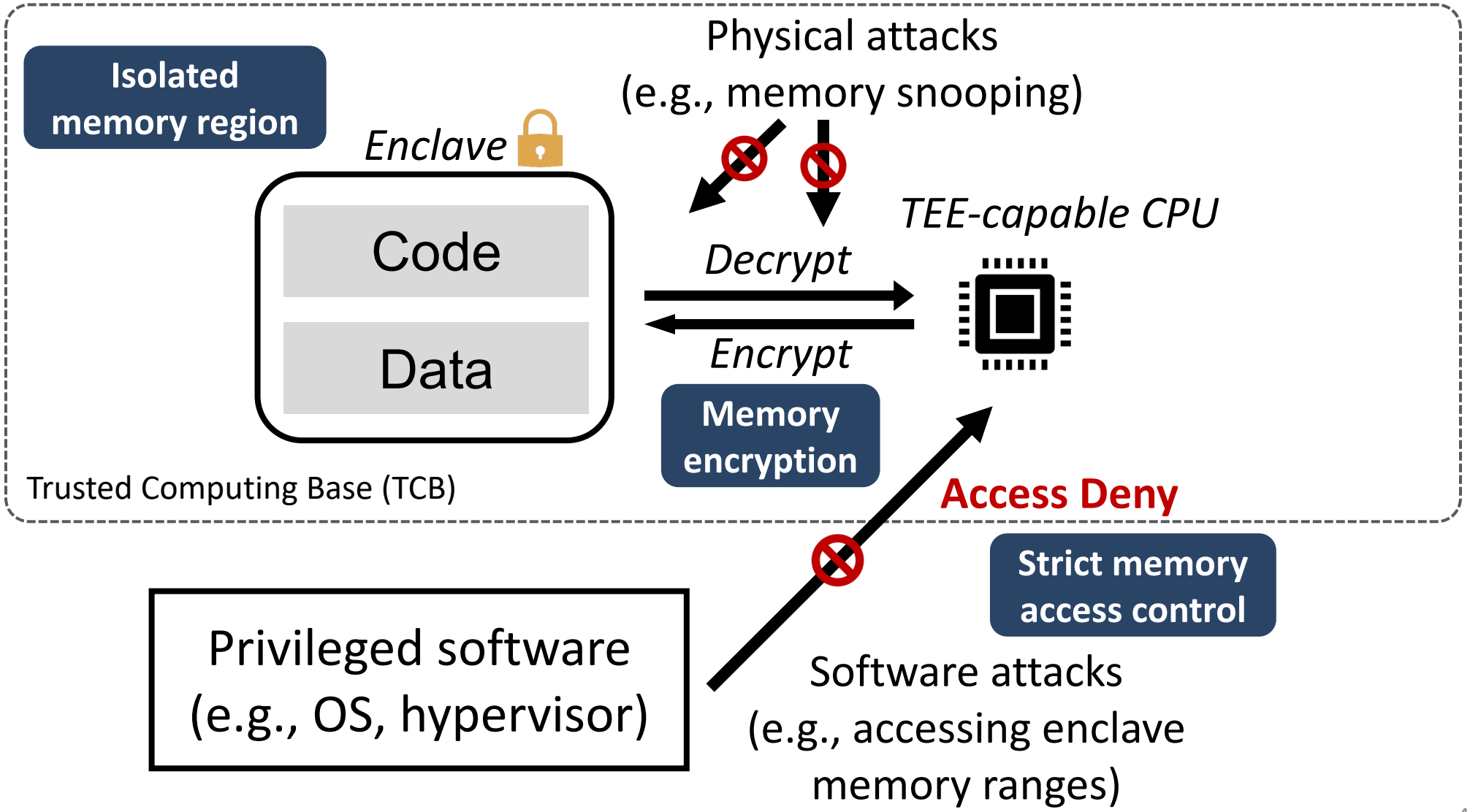
TEE 101


Isolated memory region
Enclave
Code
Data
TEE-capable CPU
Trusted Computing Base (TCB)
Privileged software (e.g., OS, hypervisor)

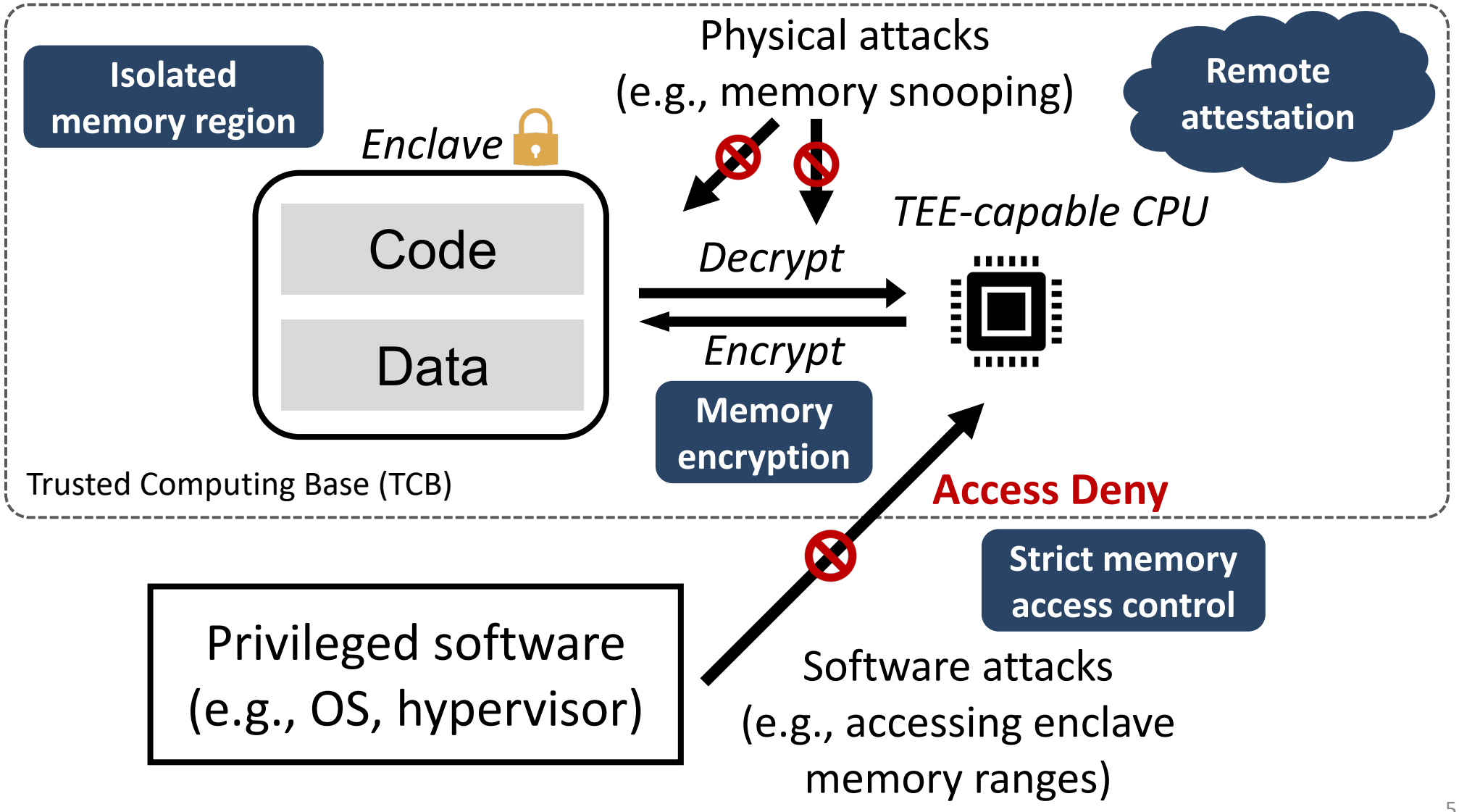
TEE 101


Isolated memory region
Enclave
Code
Data
TEE-capable CPU
Trusted Computing Base (TCB)
Access Deny
Strict memory access control
Privileged software (e.g., OS, hypervisor)
Software attacks (e.g., accessing enclave memory ranges)

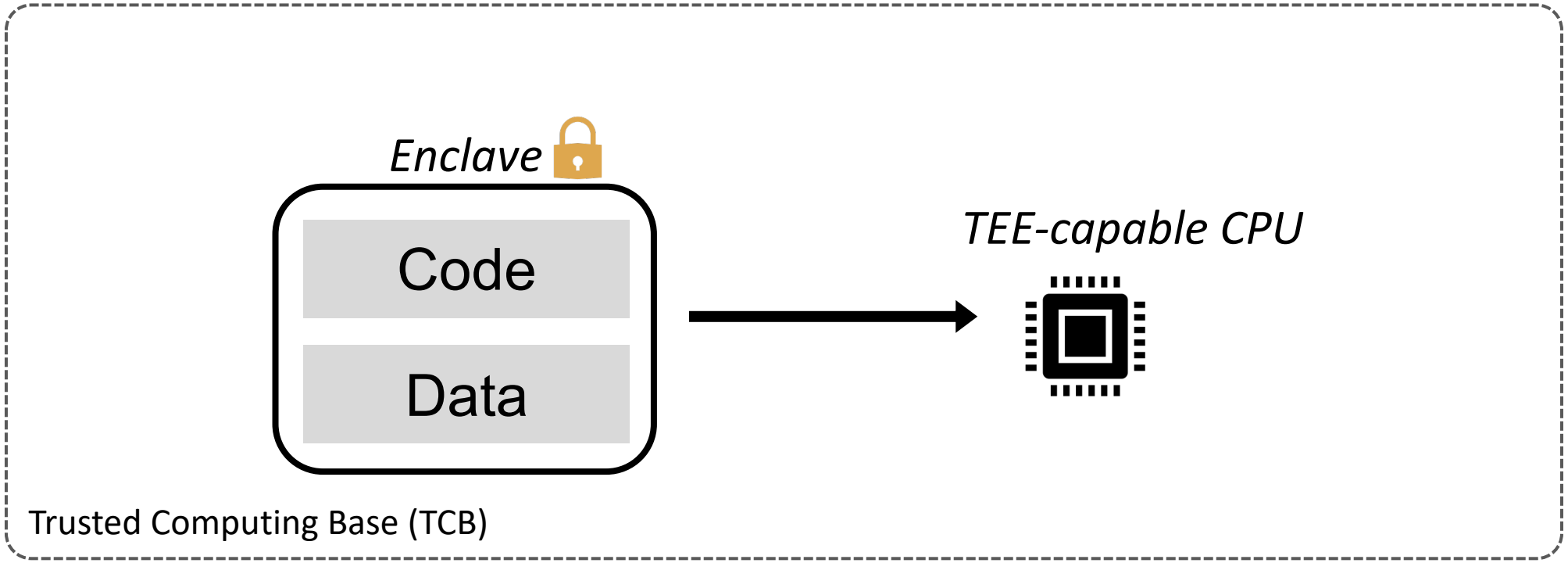
TEE 101


Isolated memory region
Physical attacks
(e.g., memory snooping)
Enclave
Code
Data
TEE-capable CPU
Decrypt
Encrypt
Memory encryption
Trusted Computing Base (TCB)
Access Deny
Strict memory access control
Privileged software
(e.g., OS, hypervisor)
Software attacks
(e.g., accessing enclave memory ranges)

TEE 101

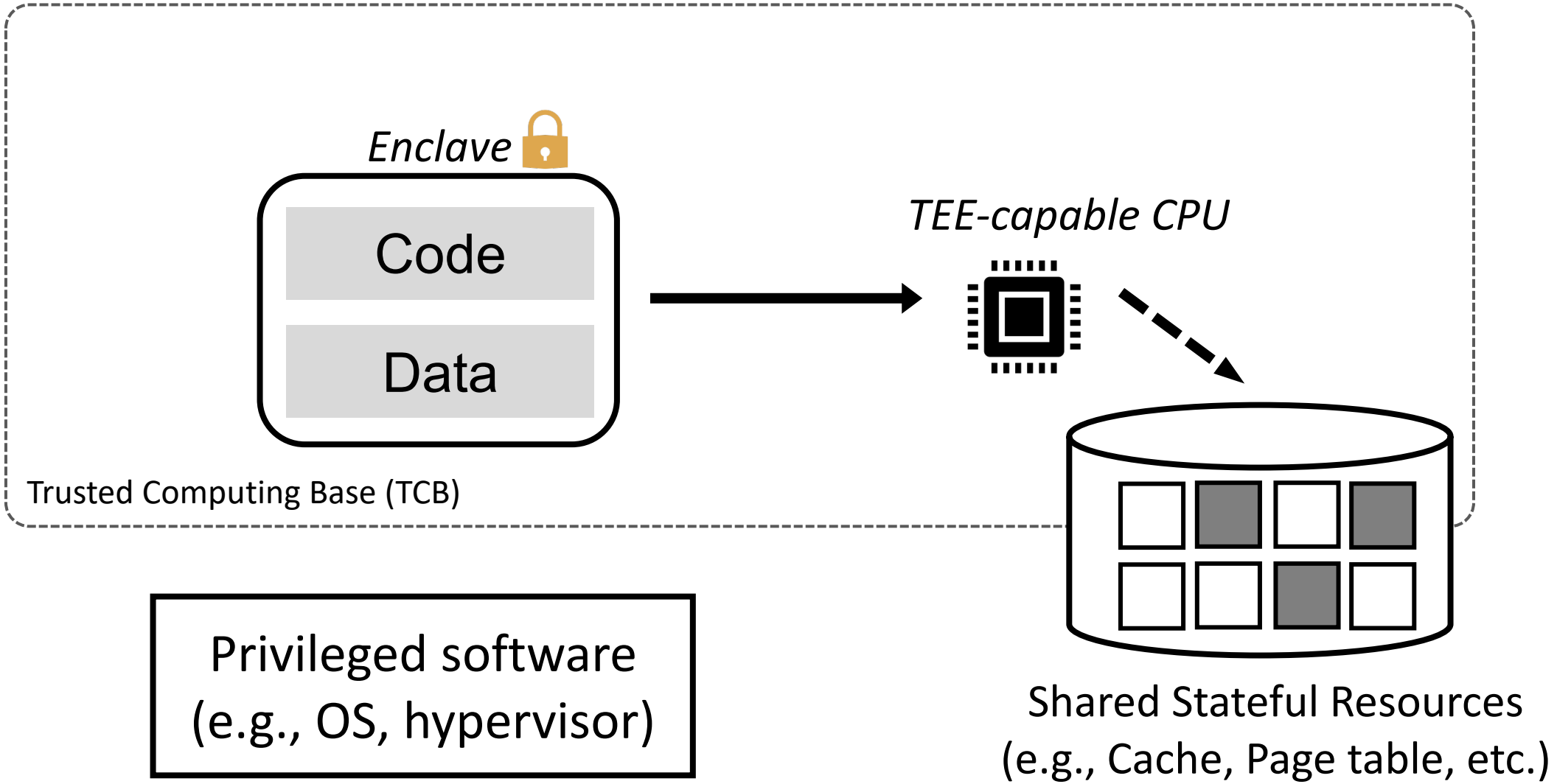

Isolated memory region
Physical attacks (e.g., memory snooping)
Remote attestation
Enclave
Code
Data
TEE-capable CPU
Decrypt
Encrypt
Memory encryption
Trusted Computing Base (TCB)
Access Deny
Strict memory access control
Privileged software (e.g., OS, hypervisor)
Software attacks (e.g., accessing enclave memory ranges)

Achilles' Heel: Side-Channel Attacks (SCAs)

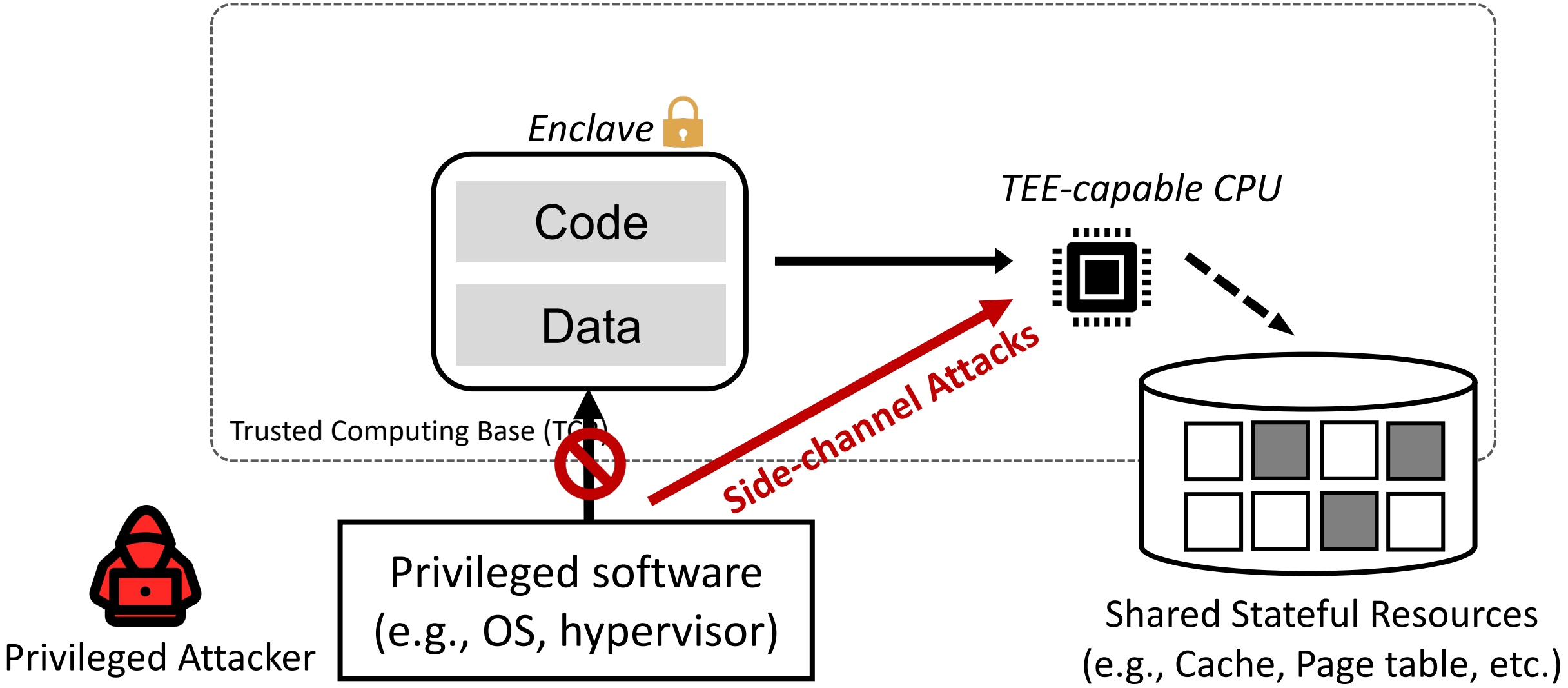


Privileged software
(e.g., OS, hypervisor)

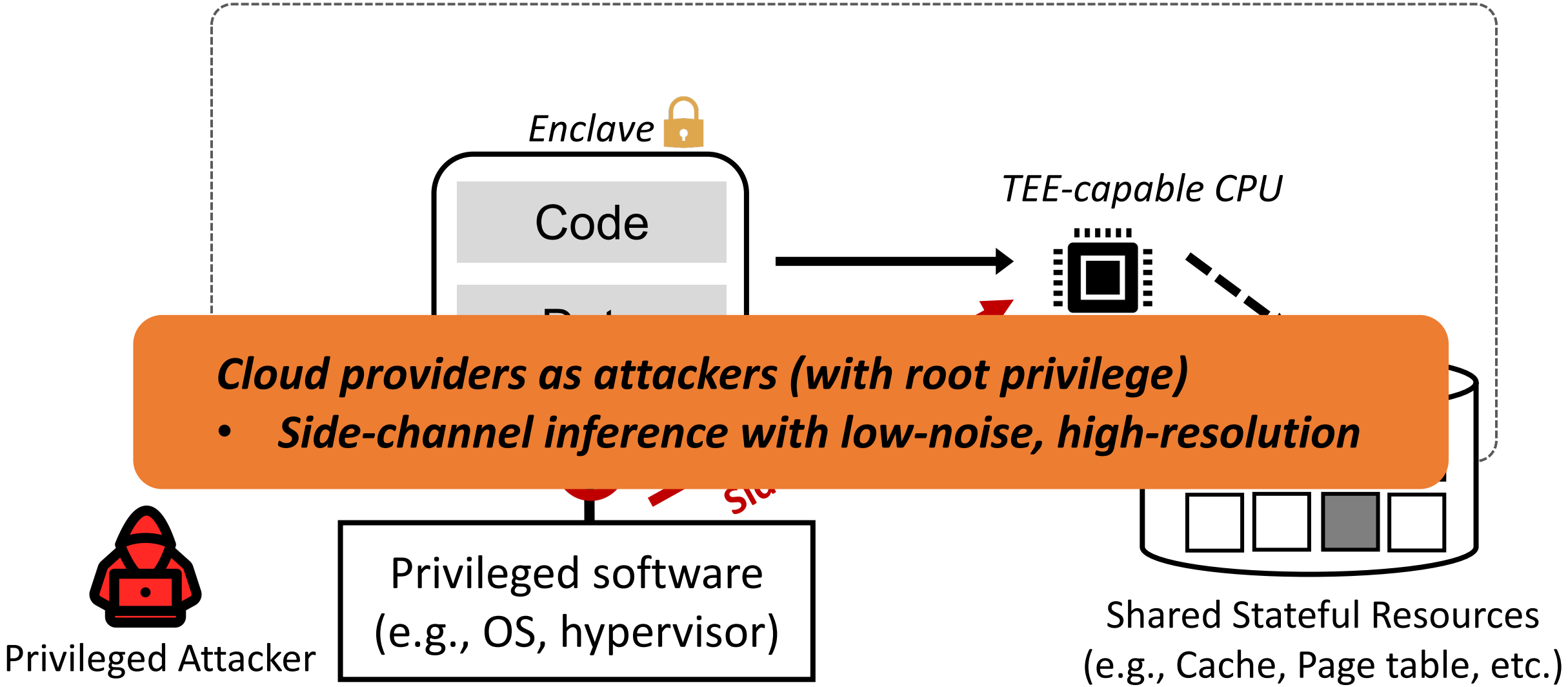
Achilles' Heel: Side-Channel Attacks (SCAs)



Achilles' Heel: Side-Channel Attacks (SCAs)



Achilles' Heel: Side-Channel Attacks (SCAs)



Existing Defenses

- Hardware resource isolation
  - \+ Robust defense that mitigates the root causes of leakage
  - − Software cannot benefit from resource sharing

Existing Defenses

- Hardware resource isolation
  - + Robust defense that mitigates the root causes of leakage
  - − Software cannot benefit from resource sharing
- Detection-based defenses
  - + Permits resource sharing
  - − Leaks at a lower rate, or lacks flexibility

Detection Defenses: Performance Counters

- Assumption: SCAs have impact on victim's performance
  - Detect SCAs by observing such performance characteristics [1-3]
  - Rely on OS interface to collect microarchitectural information
- Limitations:
  - Coarse-grained statistical data
  - Limited victim responses

[1] M. Chiappetta, E. Savas, and C. Yilmaz, "Real time detection of cache-based side-channel attacks using hardware performance counters," Applied Soft Computing, vol. 49, pp. 1162–1174, 2016.

[2] J. Chen and G. Venkataramani, "Cc-hunter: Uncovering covert timing channels on shared processor hardware," in Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Cambridge, UK, Dec. 2014.

[3] Replayconfusion: Detecting cache-based covert channel attacks using record and replay," in Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Taipei, Taiwan, Oct. 2016.

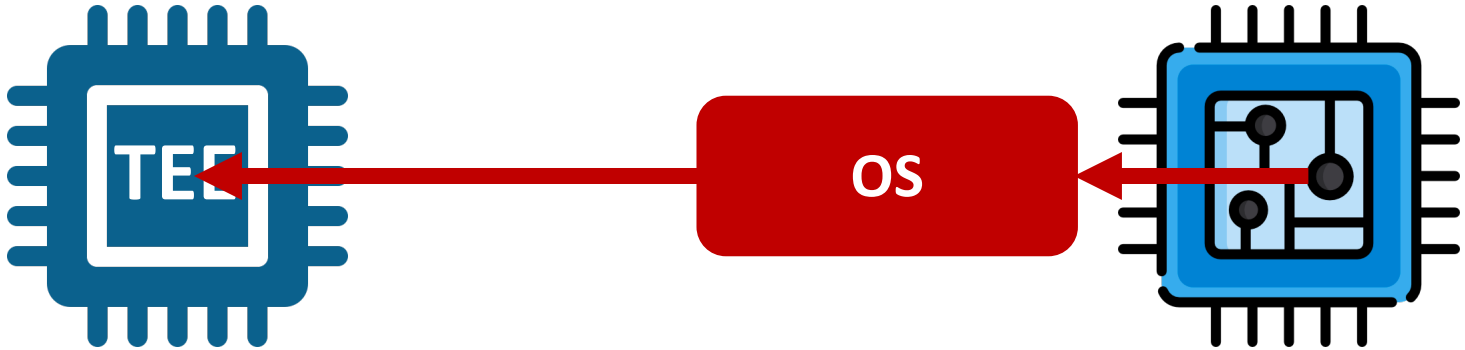
Detection Defenses: HTM

- Hardware Transactional Memory (HTM) provides safe concurrency
- Cloak [4] uses Intel® Transactional Synchronization Extensions (Intel® TSX, an HTM implementation) to mitigate cache side channels

[4] D. Gruss, J. Lettner, F. Schuster, O. Ohrimenko, I. Haller, and M. Costa, “Strong and Efficient Cache Side-Channel Protection using Hardware Transactional Memory,” in Proceedings of the 26th USENIX Security Symposium (Security), Vancouver, Canada, Aug. 2017.

Detection-Based Defenses under TEE

- Missing trusted data sources in TEEs



Detection-Based Defenses under TEE

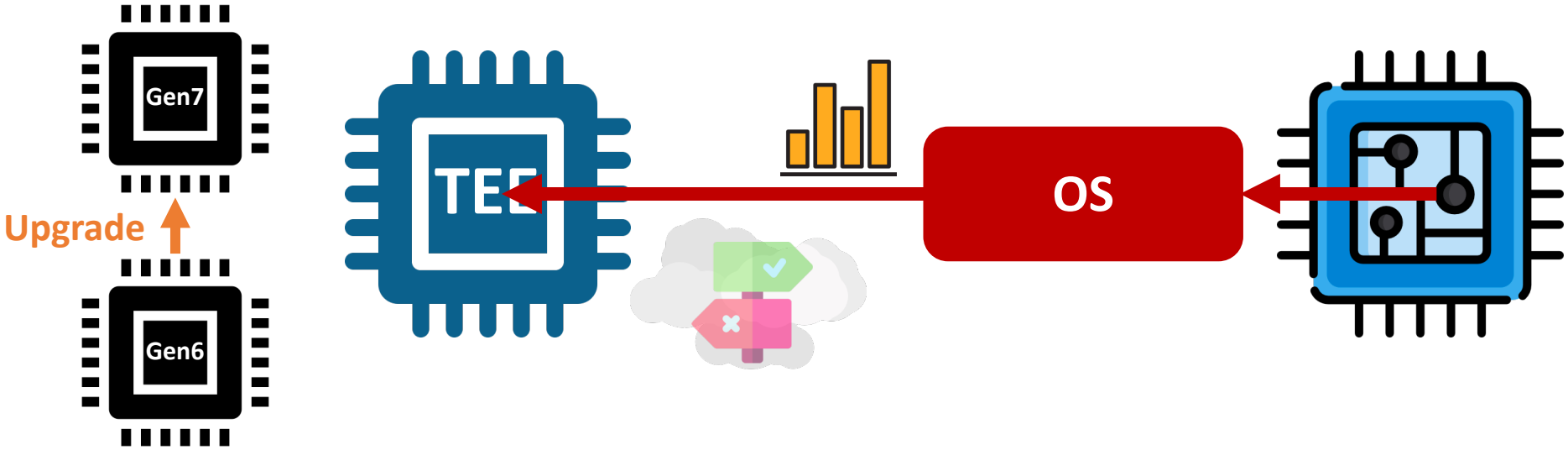
- Missing trusted data sources in TEEs
- Low-quality of available data

Detection-Based Defenses under TEE

- Missing trusted data sources in TEEs
- Low-quality of available data
- How to react to a potential attack?

Detection-Based Defenses under TEE

- Missing trusted data sources in TEEs
- Low-quality of available data
- How to react to a potential attack?
- Platform specific

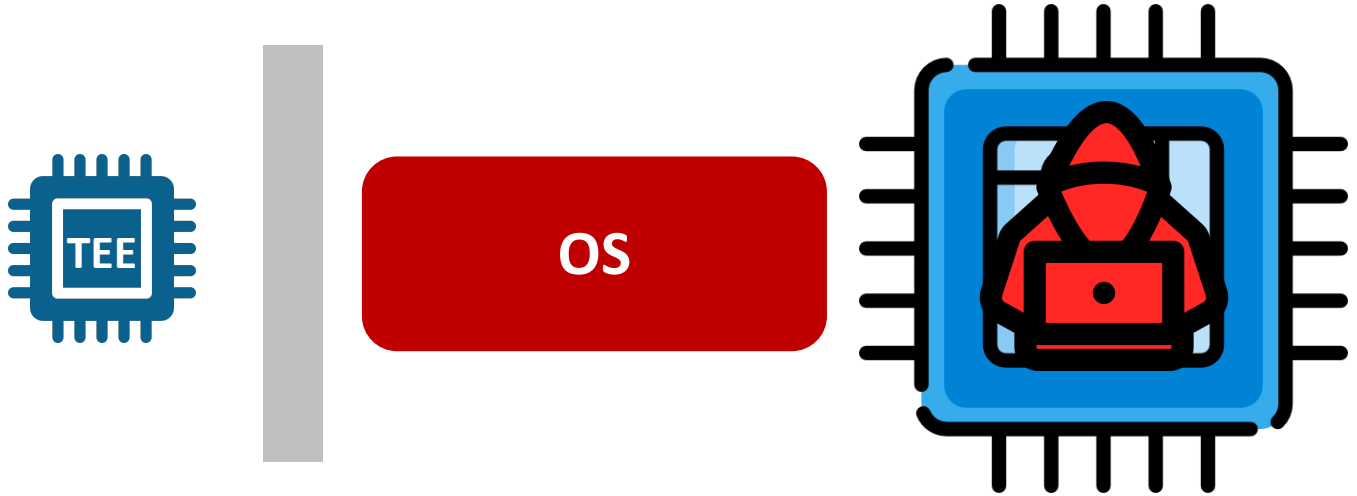


An Irony – Information Asymmetry

- TEE as a security container is constraining victims' ability to reliably gather SCA signals
  - Lack runtime awareness of the microarchitectural context
- **The attacker gathers various microarchitectural signals freely**

An Irony – Information Asymmetry

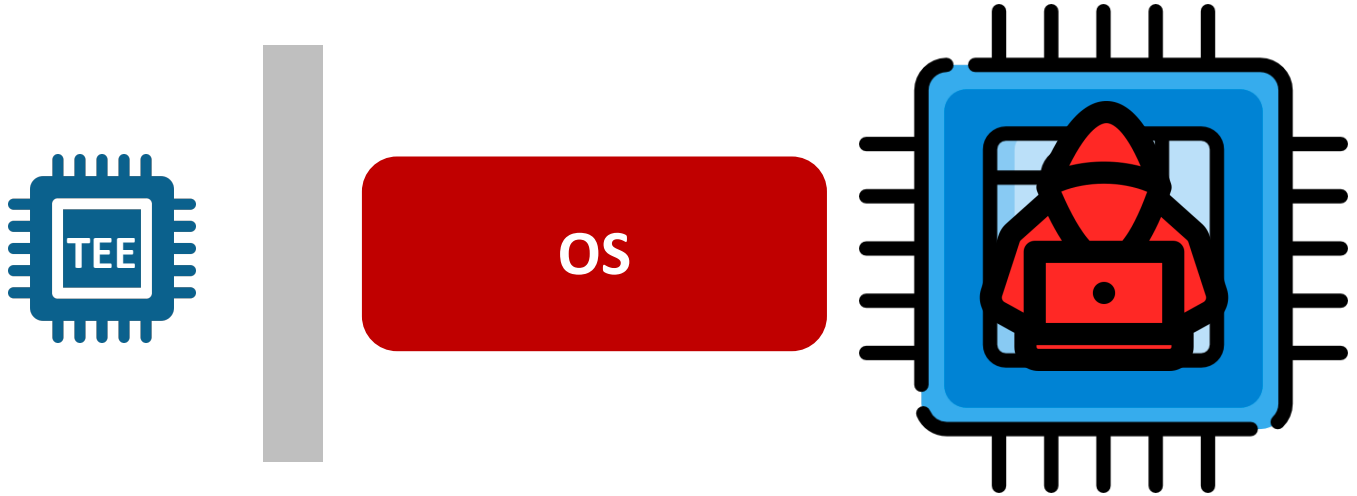
- TEE as a security container is constraining victims' ability to reliably gather SCA signals
  - Lack runtime awareness of the microarchitectural context
- **The attacker gathers various microarchitectural signals freely**



The victim always sits at a disadvantaged position

An Irony – Information Asymmetry

- TEE as a security container is constraining victims' ability to reliably gather SCA signals
  - Lack runtime awareness of the microarchitectural context
- **The attacker gathers various microarchitectural signals freely**



Can we make detection-based defenses more robust?

Goals of SENSE

- Information symmetry
  - Provide microarchitectural information to TEE and allow proactive response

Goals of SENSE

- Information symmetry
  - Provide microarchitectural information to TEE and allow proactive response
- Security
  - Does not expose new information to a potential attacker

Goals of SENSE

- Information symmetry
  - Provide microarchitectural information to TEE and allow proactive response
- Security
  - Does not expose new information to a potential attacker
- Feasibility
  - Incur minimal hardware modification

Goals of SENSE

- Information symmetry
  - Provide microarchitectural information to TEE and allow proactive response
- Security
  - Does not expose new information to a potential attacker
- Feasibility
  - Incur minimal hardware modification
- Extensibility
  - Ready for new events in the future

Overview of SENSE

- An architectural extension that directly exposes microarchitectural events to userspace TEEs, using three steps:

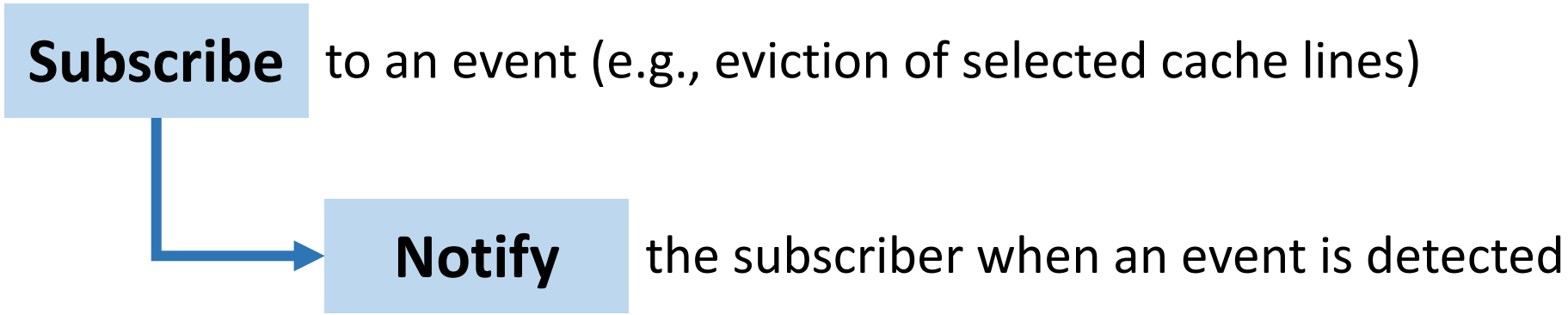
Overview of SENSE

- An architectural extension that directly exposes microarchitectural events to userspace TEEs, using three steps:

Subscribe to an event (e.g., eviction of selected cache lines)

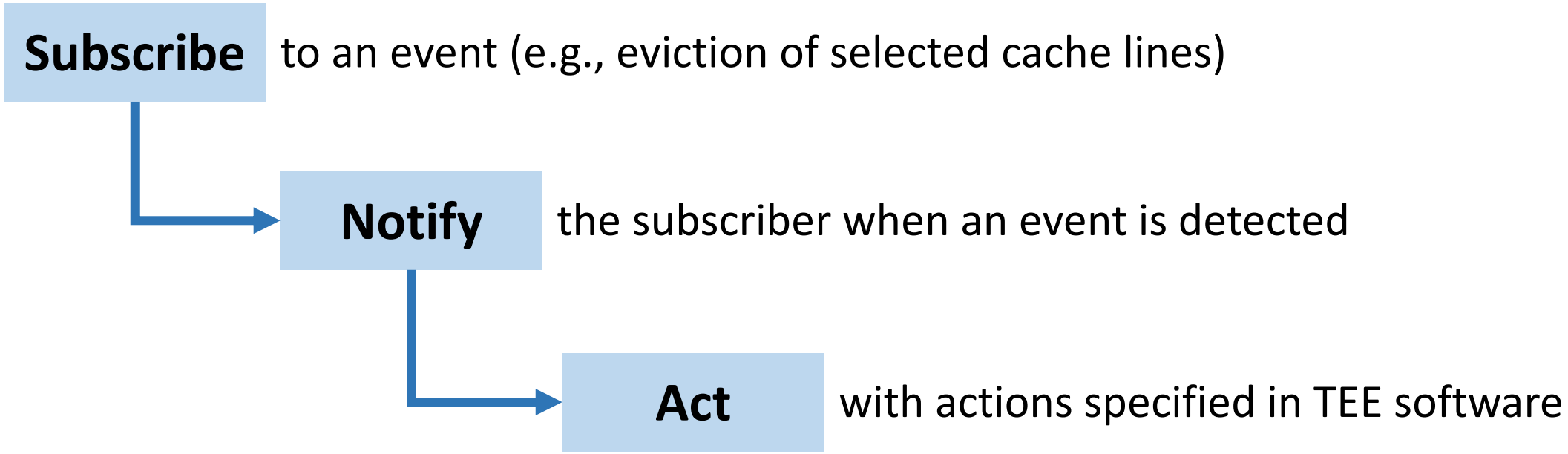
Overview of SENSE

- An architectural extension that directly exposes microarchitectural events to userspace TEEs, using three steps:

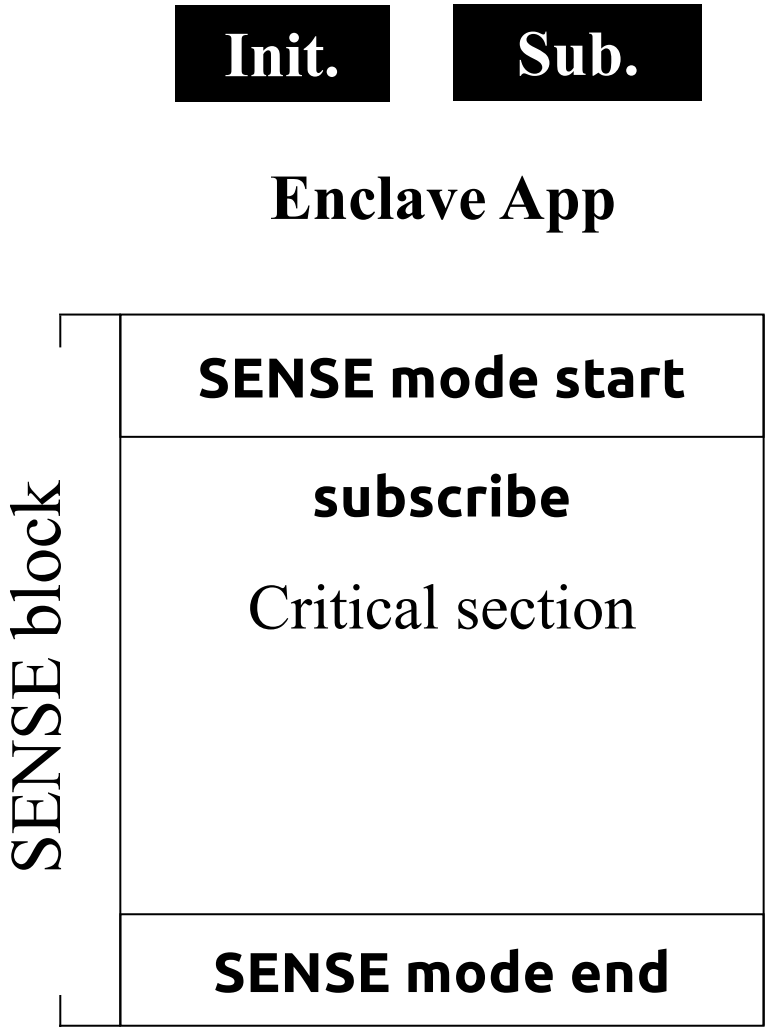


Overview of SENSE

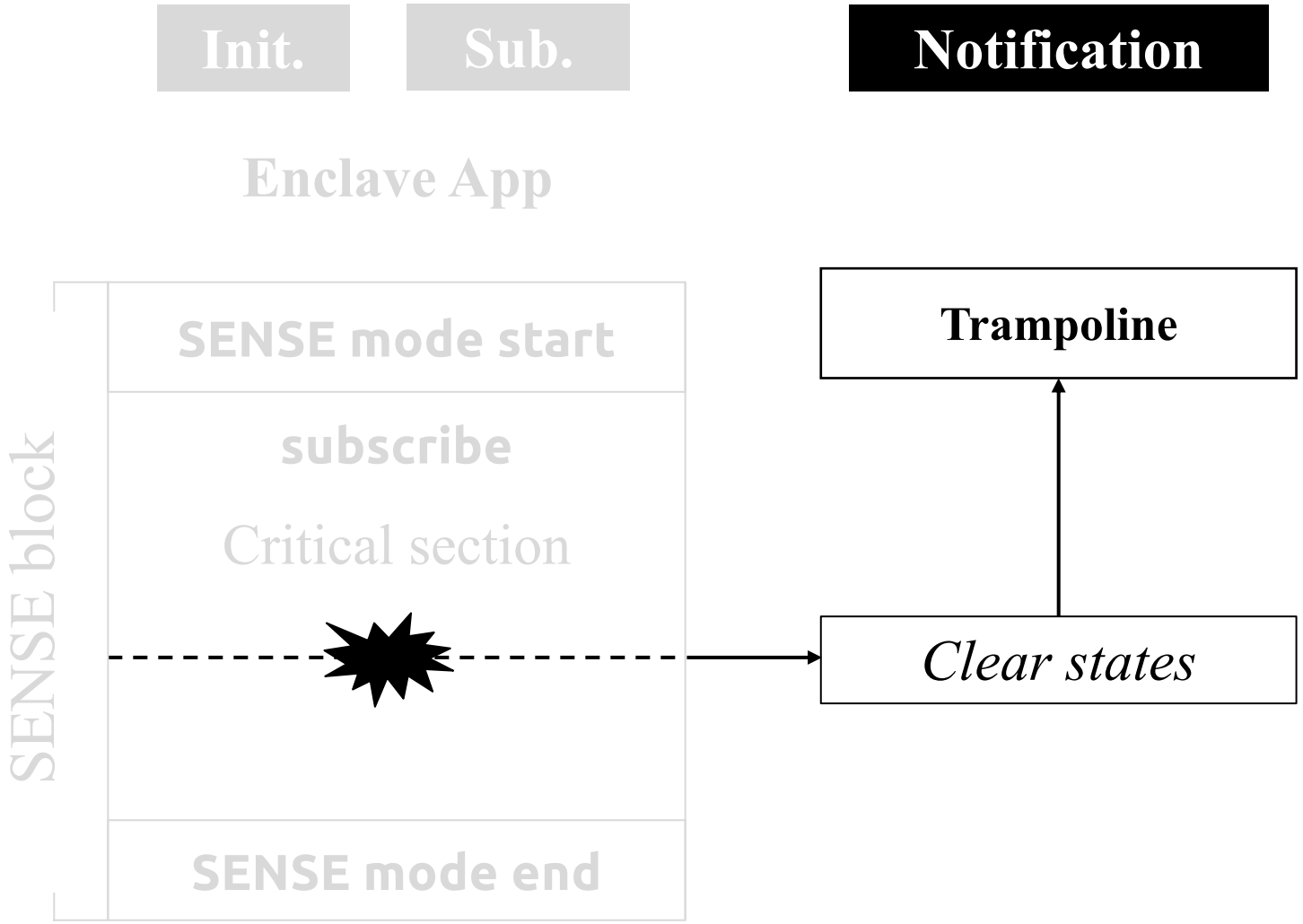
- An architectural extension that directly exposes microarchitectural events to userspace TEEs, using three steps:



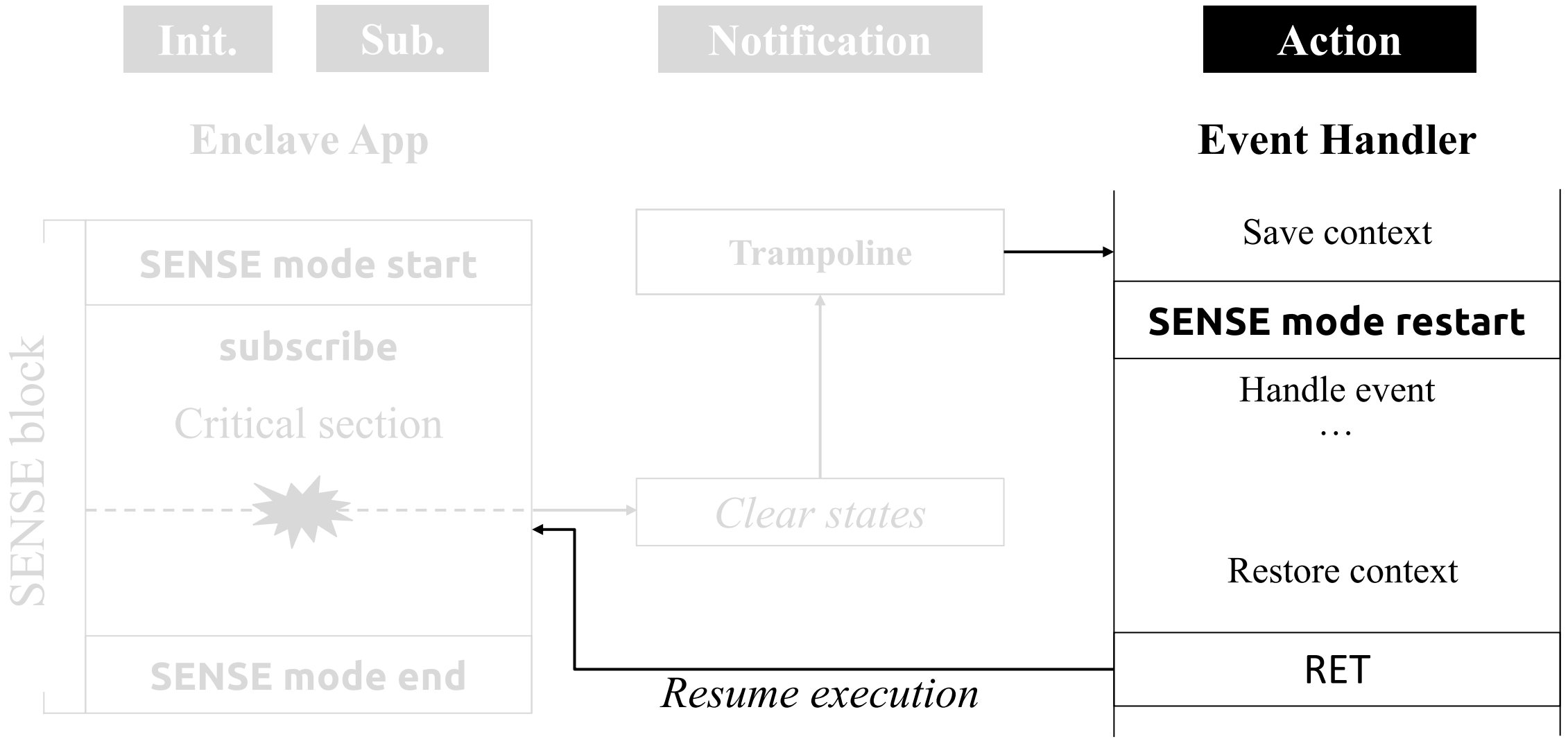
SENSE in Action



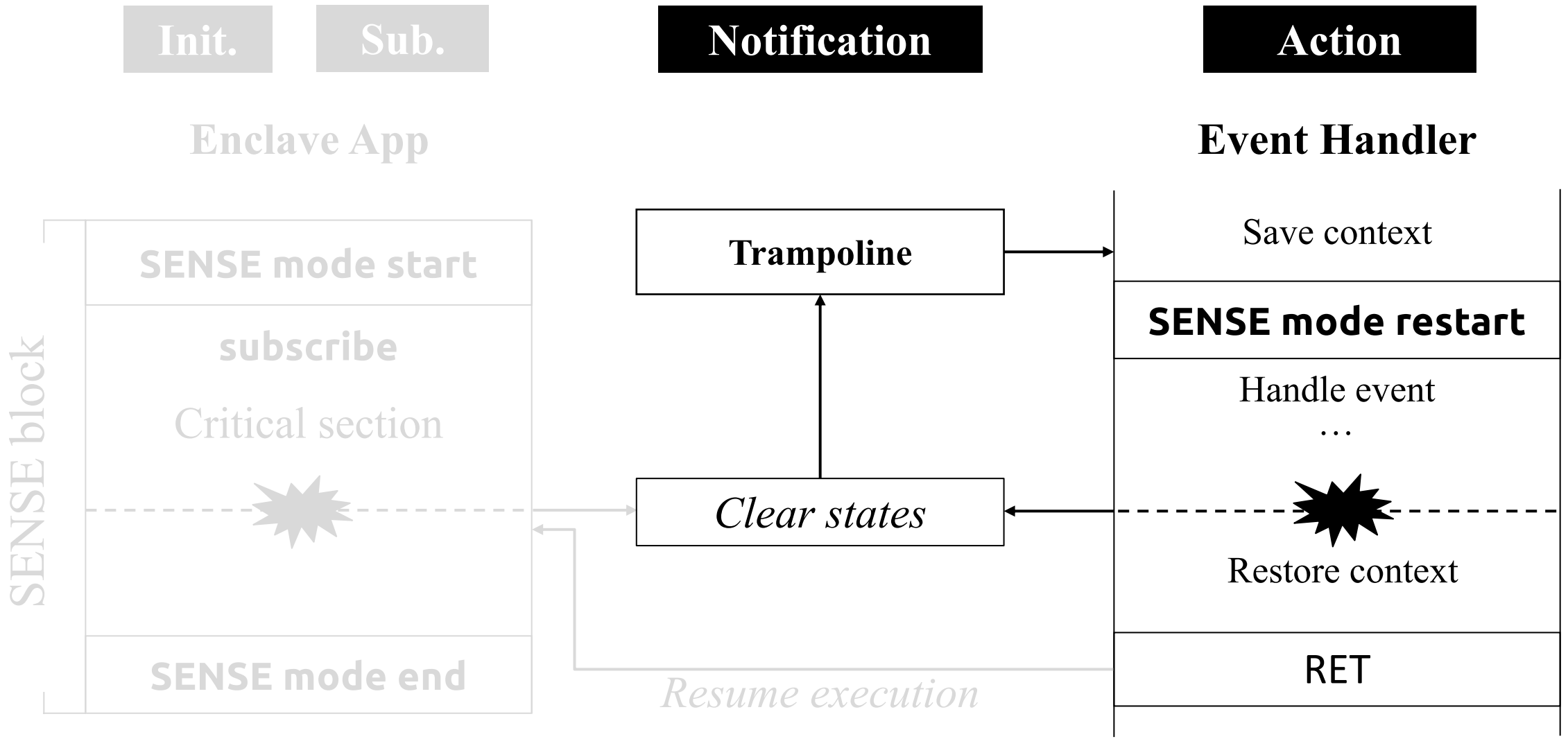
SENSE in Action



SENSE in Action



SENSE in Action

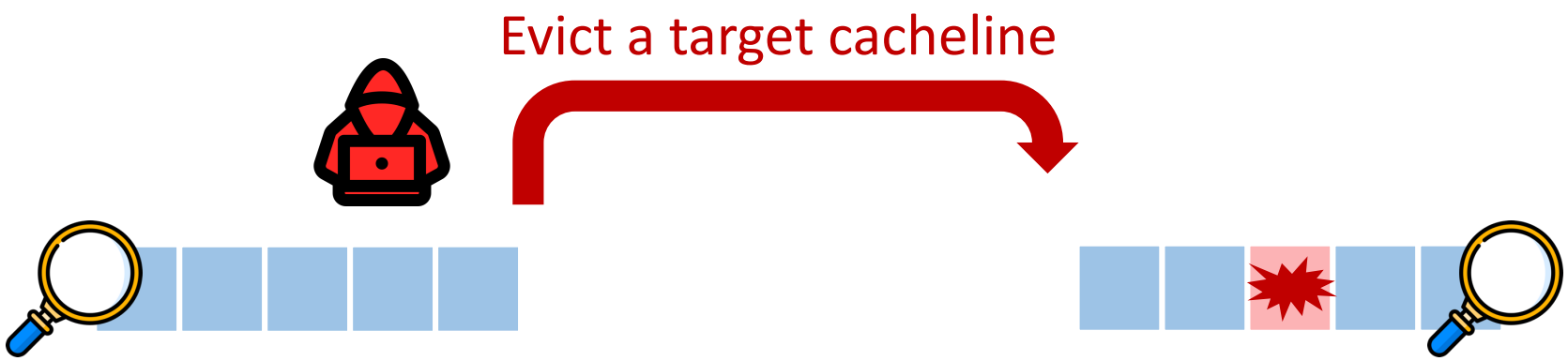


Mitigating Cache SCAs

- Target: code and data that incur secret-dependent cache accesses
  - E.g., cachelines of AES T-table and encryption operations

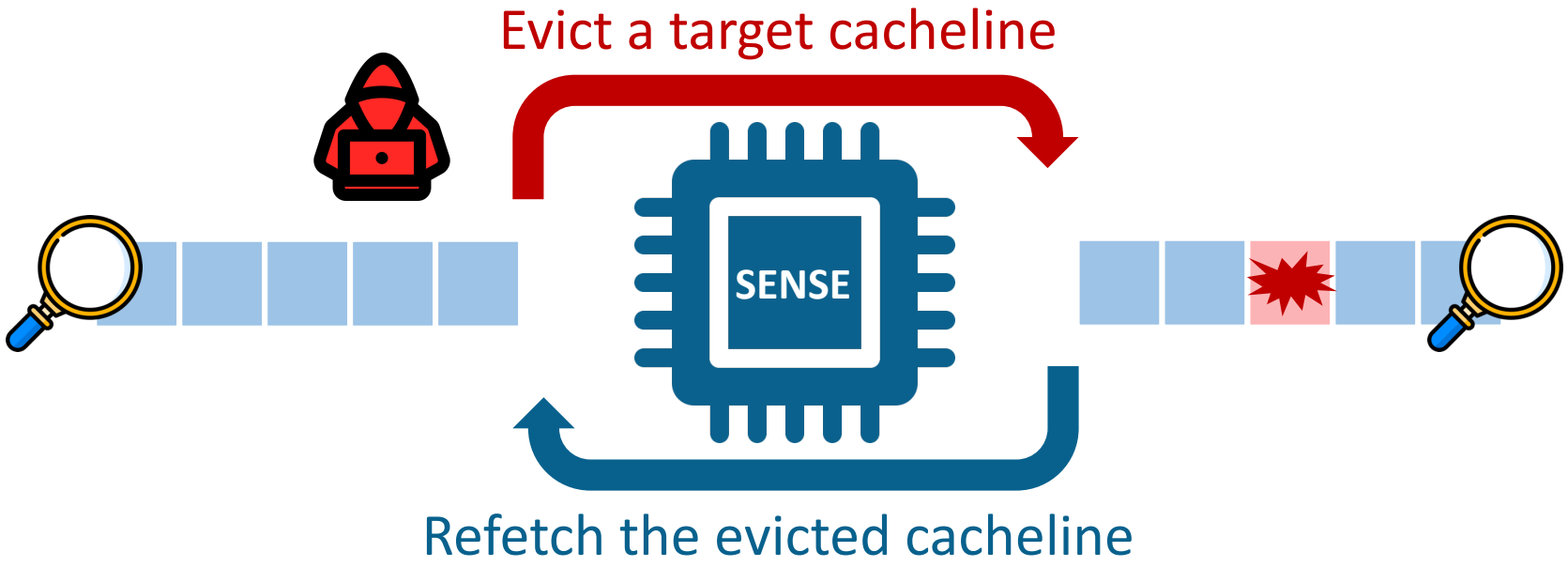
Mitigating Cache SCAs

- Target: code and data that incur secret-dependent cache accesses
  - E.g., cachelines of AES T-table and encryption operations

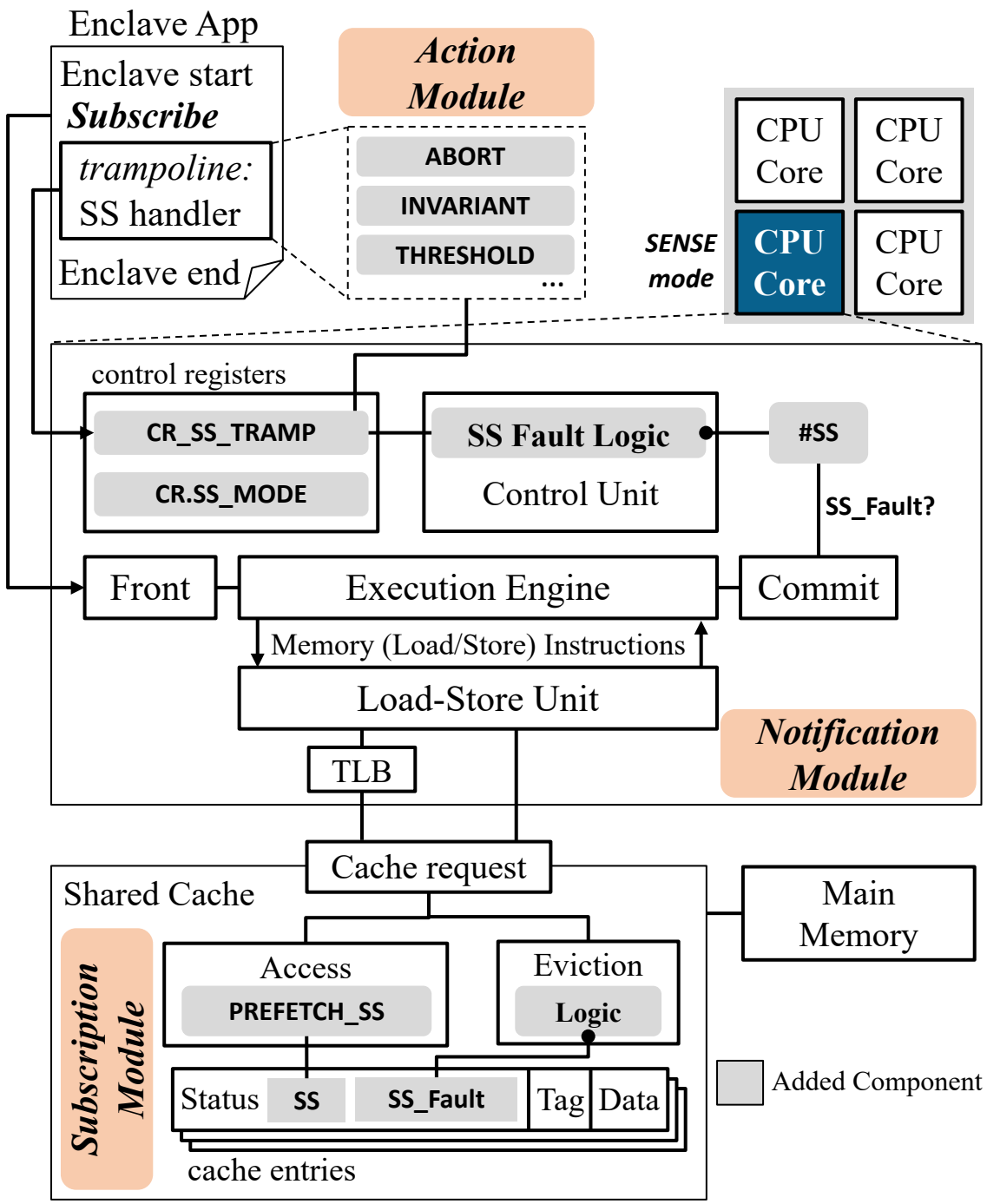


Mitigating Cache SCAs

- Target: code and data that incur secret-dependent cache accesses
  - E.g., cachelines of AES T-table and encryption operations

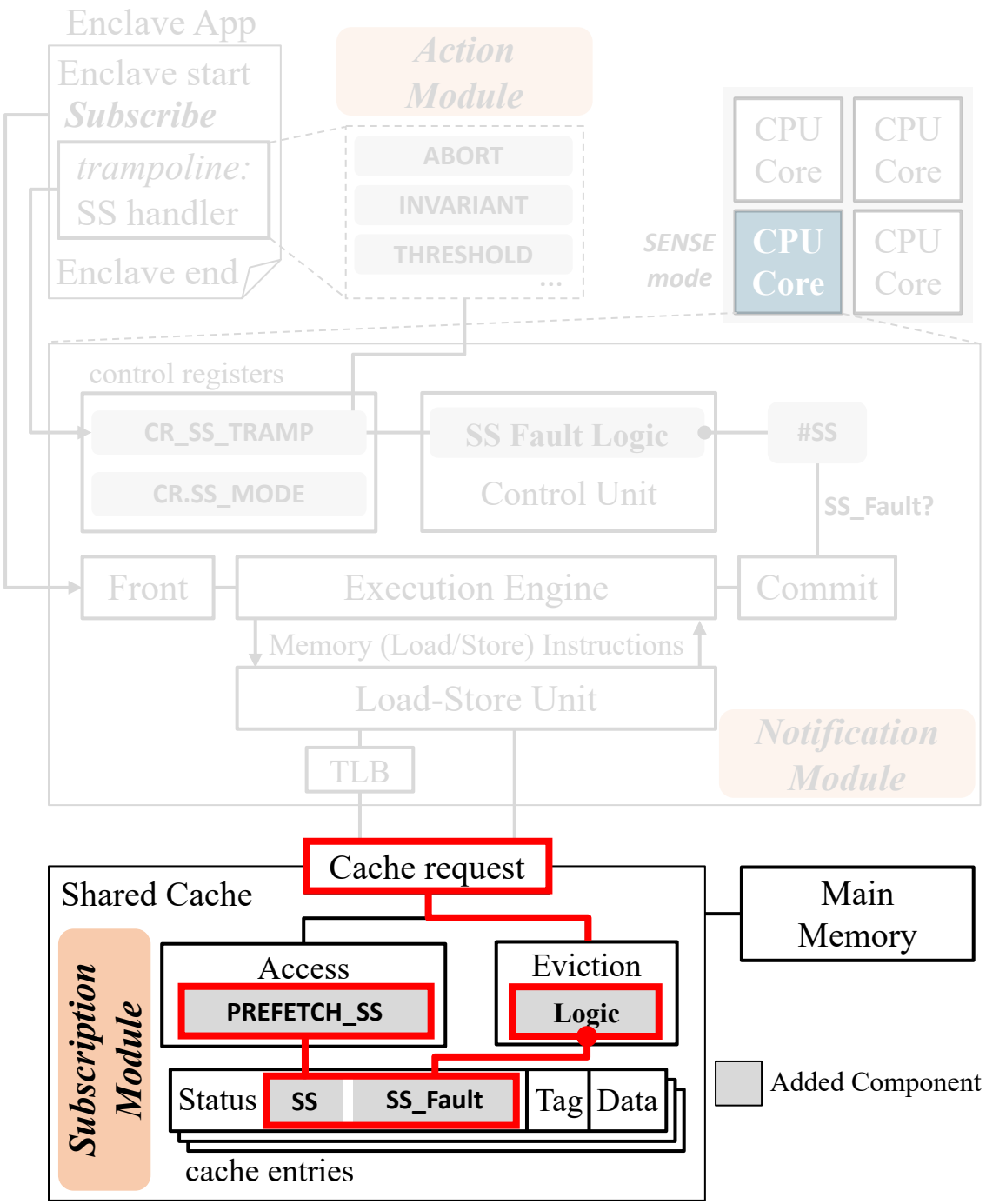


Architecture



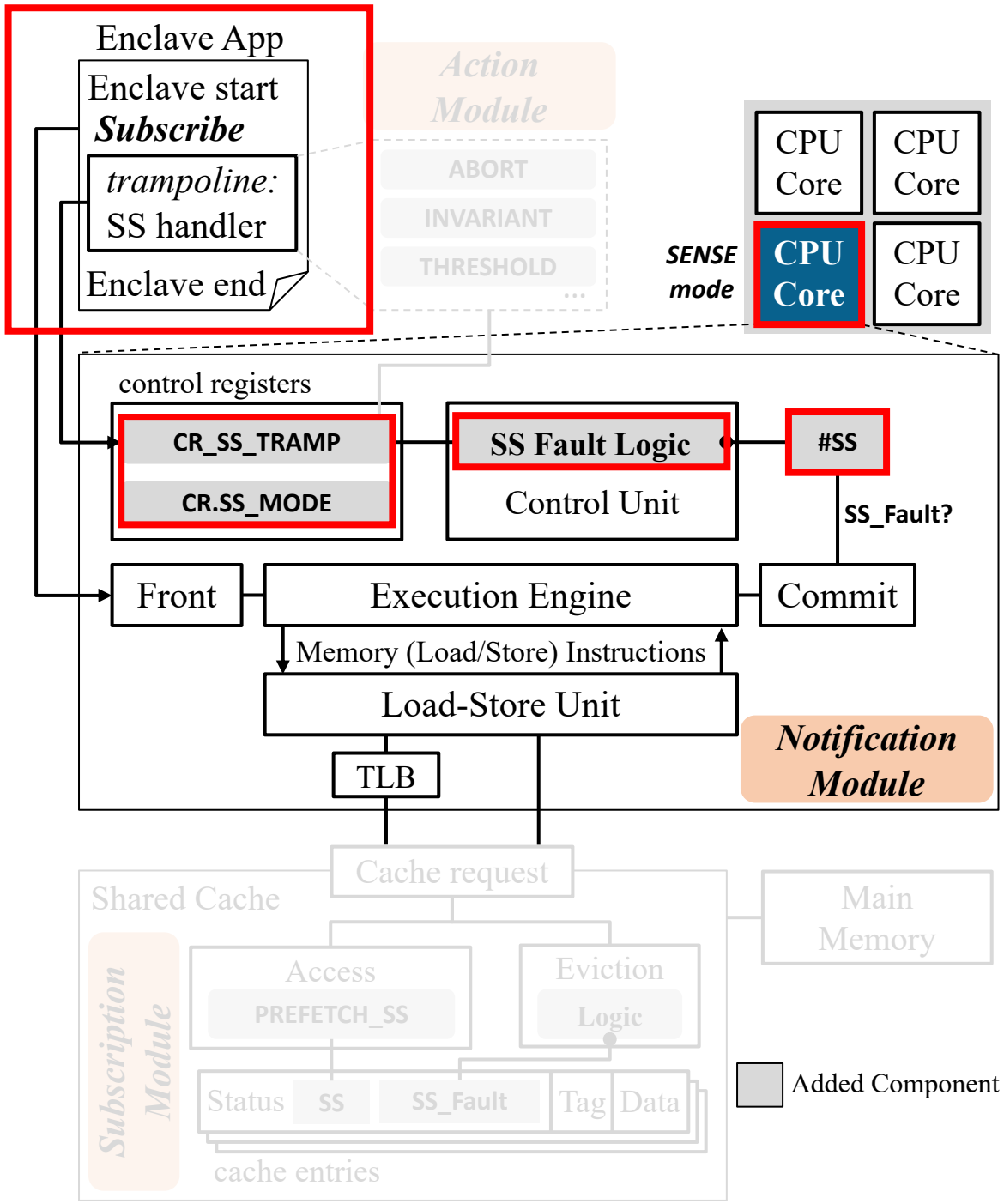
Subscription Module

- Subscription request as a specialized memory request
- SENSE status bits in each cache entry for status monitoring
- Check the monitoring status during eviction
- Inform the notification module in the cache request response



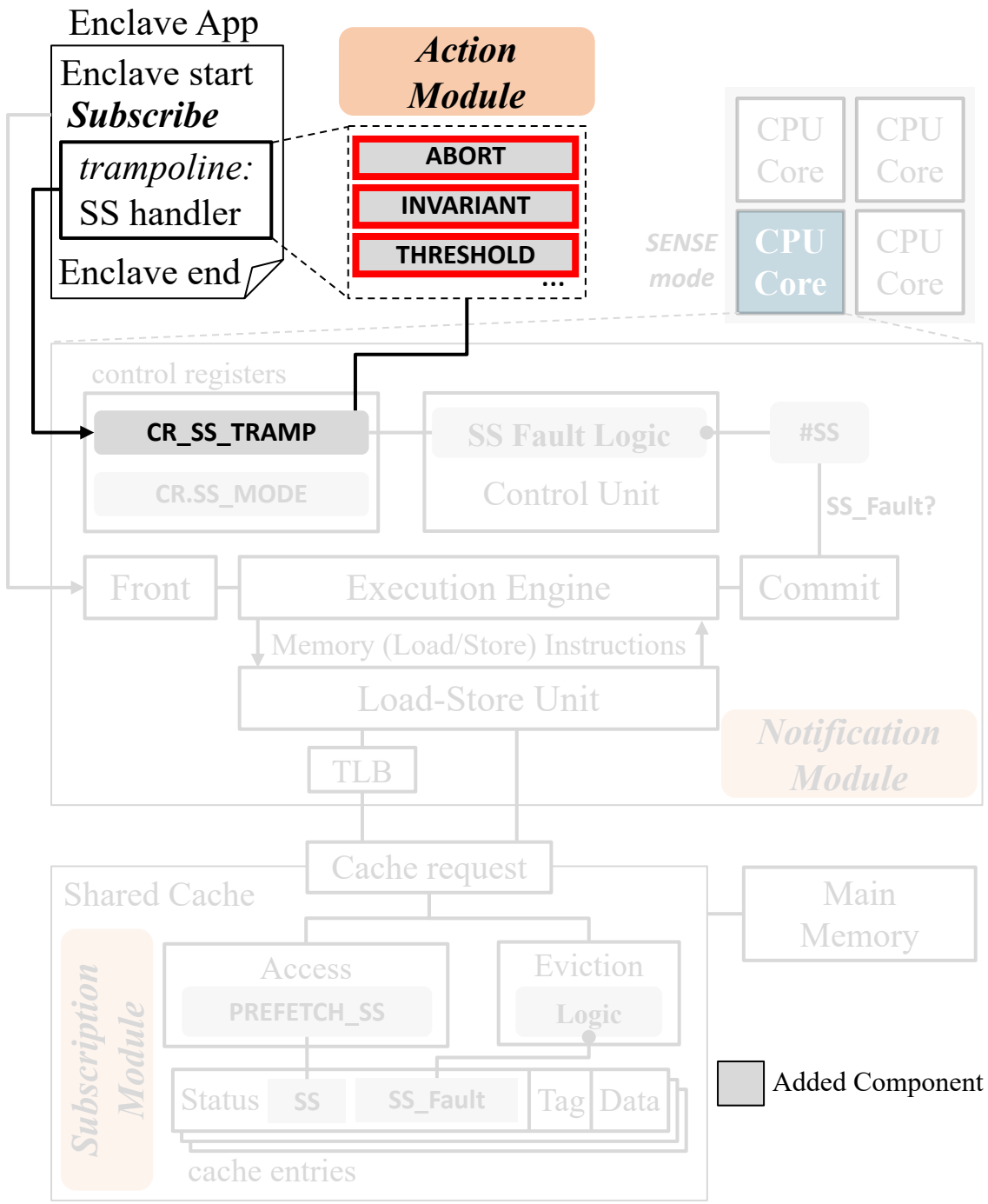
Notification Module

- SENSE control registers
- New ISA instructions
- SENSE hardware fault
- SENSE fault control logic to enter the trampoline



Action Module

- ABORT
  - Abort the process
- INVARIANT
  - Preserve a safety property
- THRESHOLD
  - Local counter for event quota
  - A policy when the threshold is exceeded (e.g., terminate)

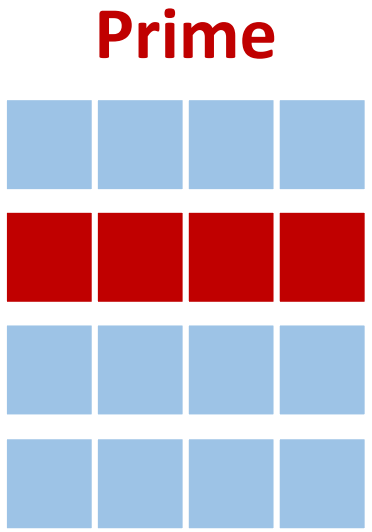


Attacker's Exploitation of SENSE

- Prime+Probe without coarse-grained timing
  - SENSE promptly delivers notifications to the attacker

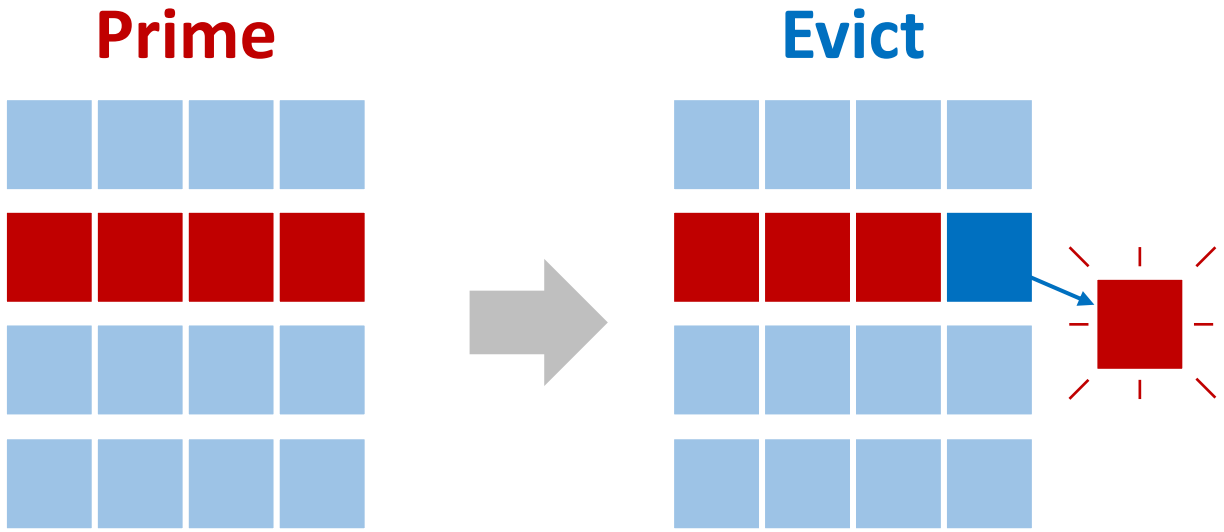
Attacker's Exploitation of SENSE

- Prime+Probe without coarse-grained timing
  - SENSE promptly delivers notifications to the attacker



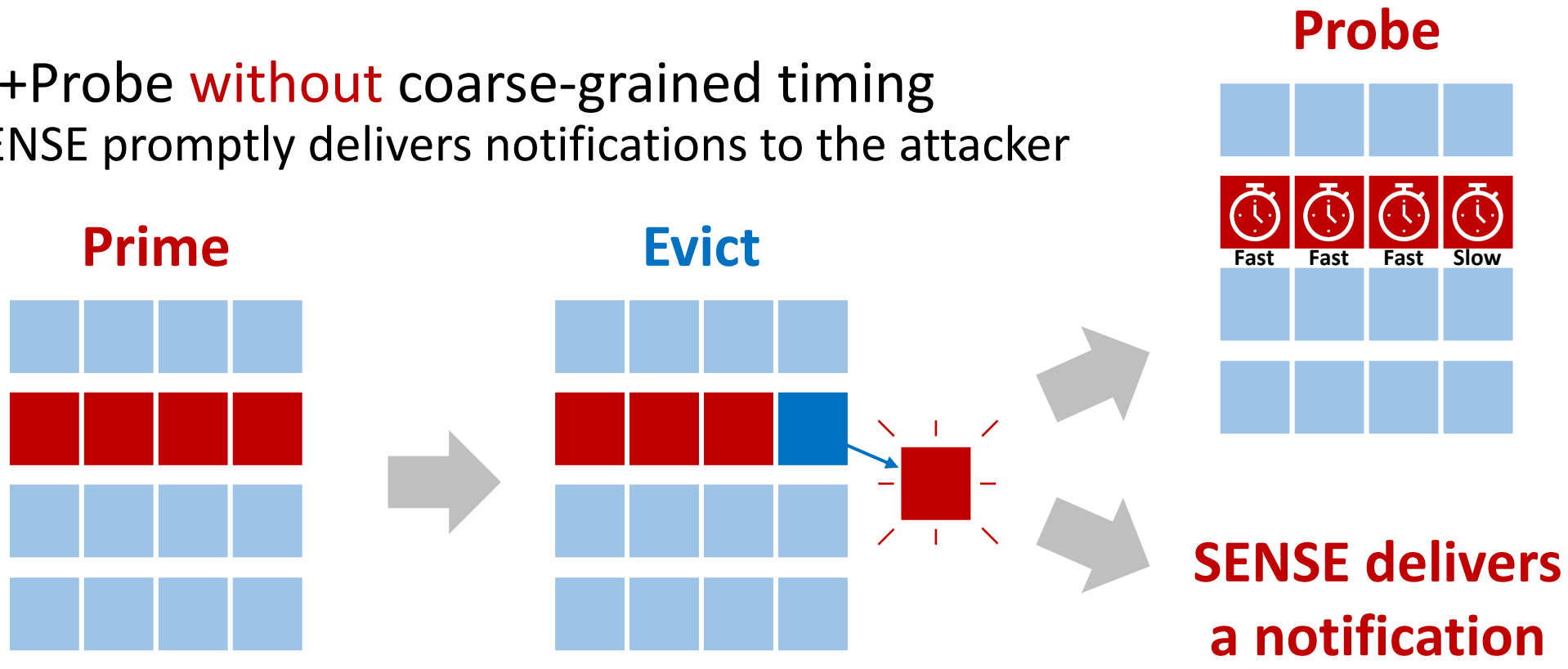
Attacker's Exploitation of SENSE

- Prime+Probe without coarse-grained timing
  - SENSE promptly delivers notifications to the attacker



Attacker's Exploitation of SENSE

- Prime+Probe without coarse-grained timing
  - SENSE promptly delivers notifications to the attacker



- SENSE does not expose new information to a malicious TEE thread
  - SENSE provides the same eviction information as Prime+Probe, with less noise

Other Use Case: Verifying OS Contracts

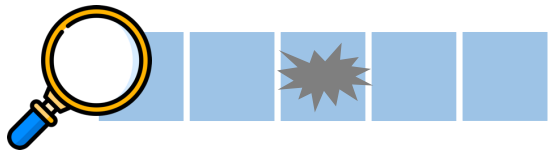
- The OS is responsible for management tasks
  - OS-application contract
- Malicious OS does not honor such contracts
- Use SENSE to implement a verification logic
- Example: Cache coloring

Other Use Case: Verifying OS Contracts

- A colored physical address is only accessible by threads with the same color
- Used to isolate cache accesses in SCA mitigations

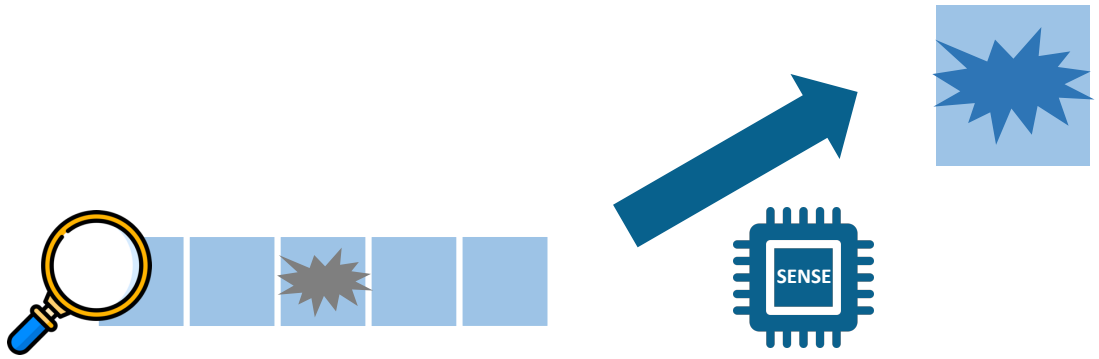
Other Use Case: Verifying OS Contracts

- A colored physical address is only accessible by threads with the same color
- Used to isolate cache accesses in SCA mitigations



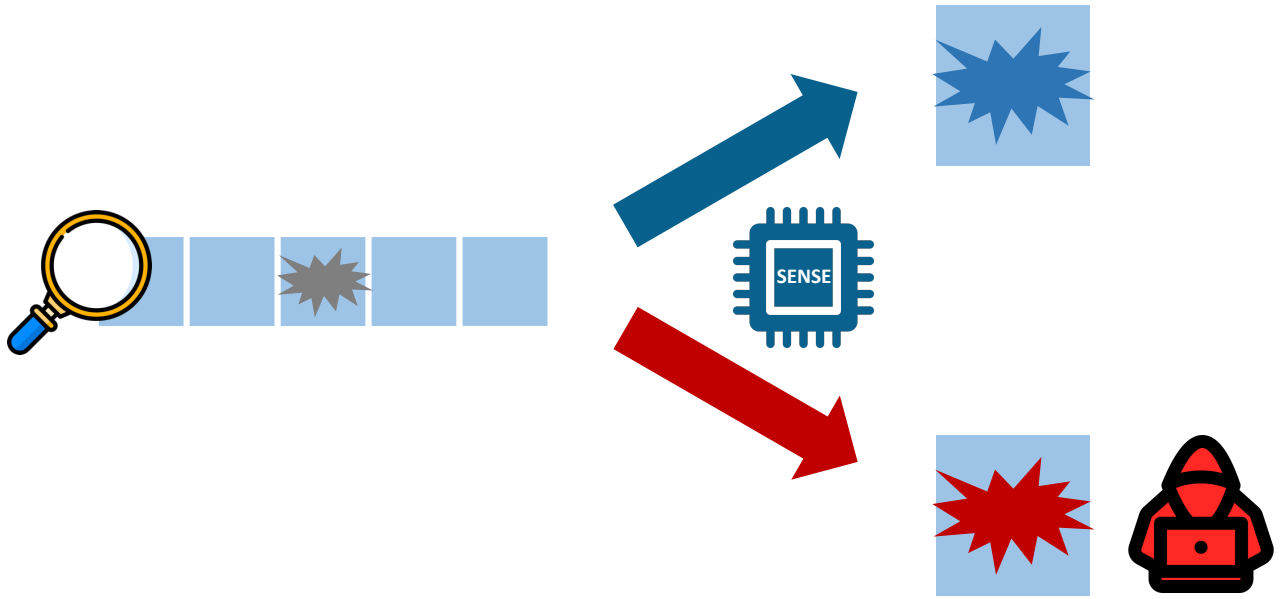
Other Use Case: Verifying OS Contracts

- A colored physical address is only accessible by threads with the same color
- Used to isolate cache accesses in SCA mitigations



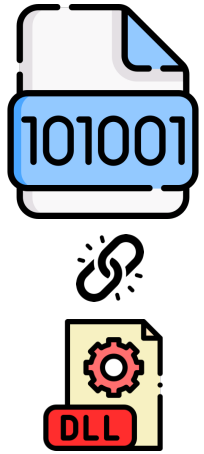
Other Use Case: Verifying OS Contracts

- A colored physical address is only accessible by threads with the same color
- Used to isolate cache accesses in SCA mitigations



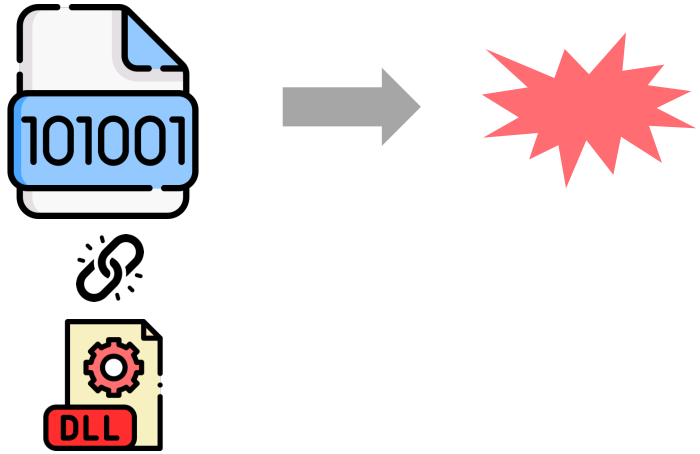
Other Use Case: On-demand Library Loading

- Software-based SCA mitigations have heavy performance overhead
- The overhead is paid regardless of the presence of SCAs
- Ideally, only pay the overhead when necessary
  - Application starts using an optimized library
  - Load the secure version of library if there is a sign of SCAs



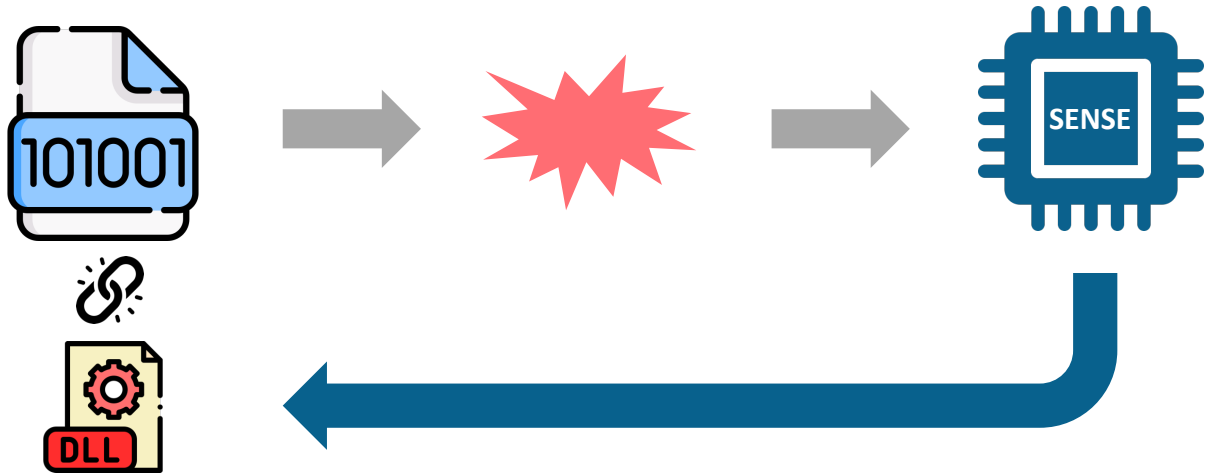
Other Use Case: On-demand Library Loading

- Software-based SCA mitigations have heavy performance overhead
- The overhead is paid regardless of the presence of SCAs
- Ideally, only pay the overhead when necessary
  - Application starts using an optimized library
  - Load the secure version of library if there is a sign of SCAs



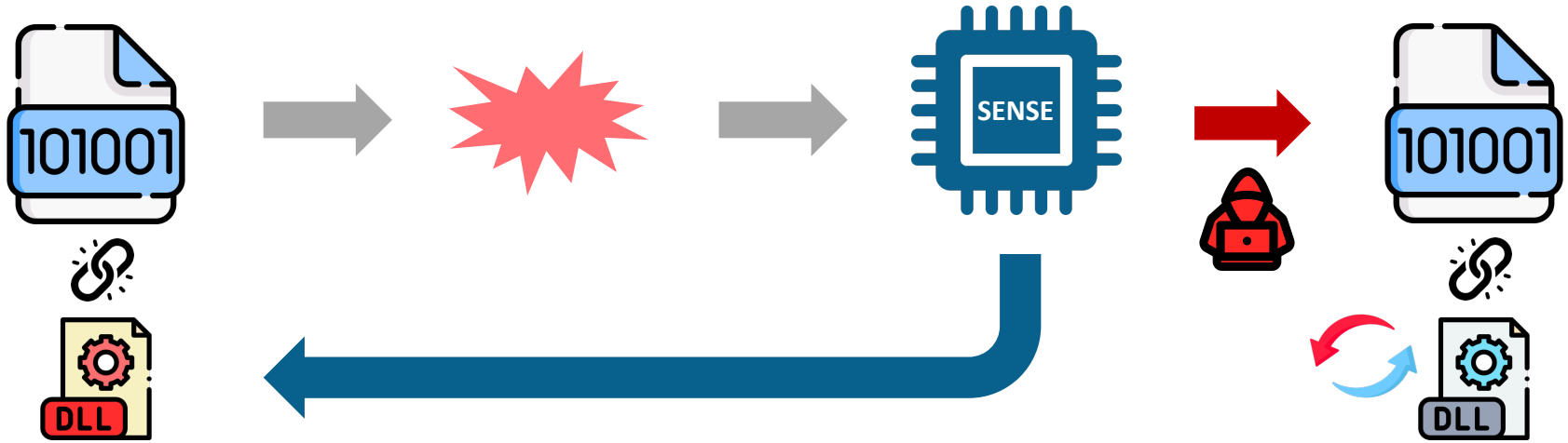
Other Use Case: On-demand Library Loading

- Software-based SCA mitigations have heavy performance overhead
- The overhead is paid regardless of the presence of SCAs
- Ideally, only pay the overhead when necessary
  - Application starts using an optimized library
  - Load the secure version of library if there is a sign of SCAs



Other Use Case: On-demand Library Loading

- Software-based SCA mitigations have heavy performance overhead
- The overhead is paid regardless of the presence of SCAs
- Ideally, only pay the overhead when necessary
  - Application starts using an optimized library
  - Load the secure version of library if there is a sign of SCAs



Implementation

- Gem5 simulation
  - CPU SENSE mode
  - CPU micro-code for event notification
  - Enabling SENSE on cache eviction events
- Basic event handlers for cache evictions
  - ABORT (exit(0))
  - INVARIANT (Refetch the evicted cacheline)
  - THRESHOLD (Number of allowed cache. Terminate if exceeded)

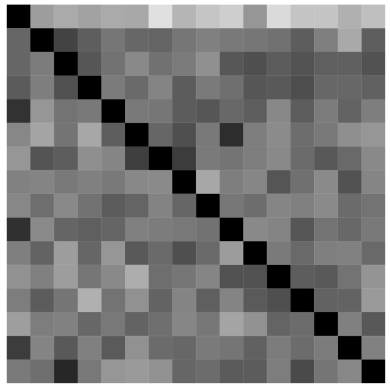
Security Evaluation

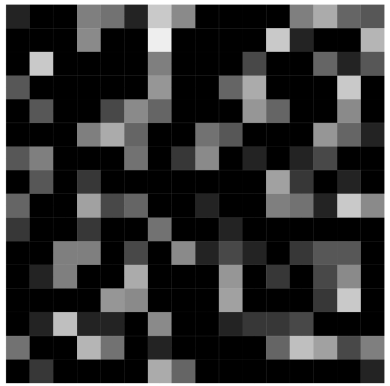
- Harden AES T-table against Prime+Probe
  - Monitors the T-table for cache eviction events
  - INVARIANT event handler

Security Evaluation

- Harden AES T-table against Prime+Probe
  - Monitors the T-table for cache eviction events
  - INVARIANT event handler

Without SENSE





With SENSE + cache INVARIANT handler

Cache hit patterns of the first AES T-Table (Te0) under Prime+Probe attack.

Security Evaluation

- Attacker's exploitation of SENSE
  - Probe via SENSE is ~8× faster than probing via timing channel
  - Probe via SENSE has a reduced false positive rate compared to timing channel

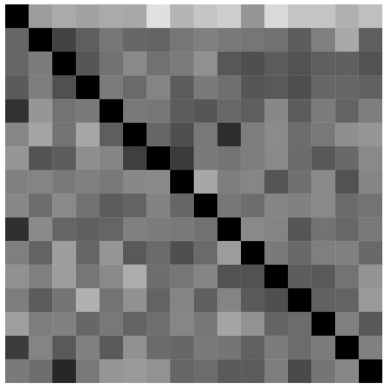
  → Efficiency benefits during an actual Prime+Probe attack

Security Evaluation

- Attacker's exploitation of SENSE
  - Probe via SENSE is ~8× faster than probing via timing channel
  - Probe via SENSE has a reduced false positive rate compared to timing channel

  → Efficiency benefits during an actual Prime+Probe attack

Without SENSE



With SENSE

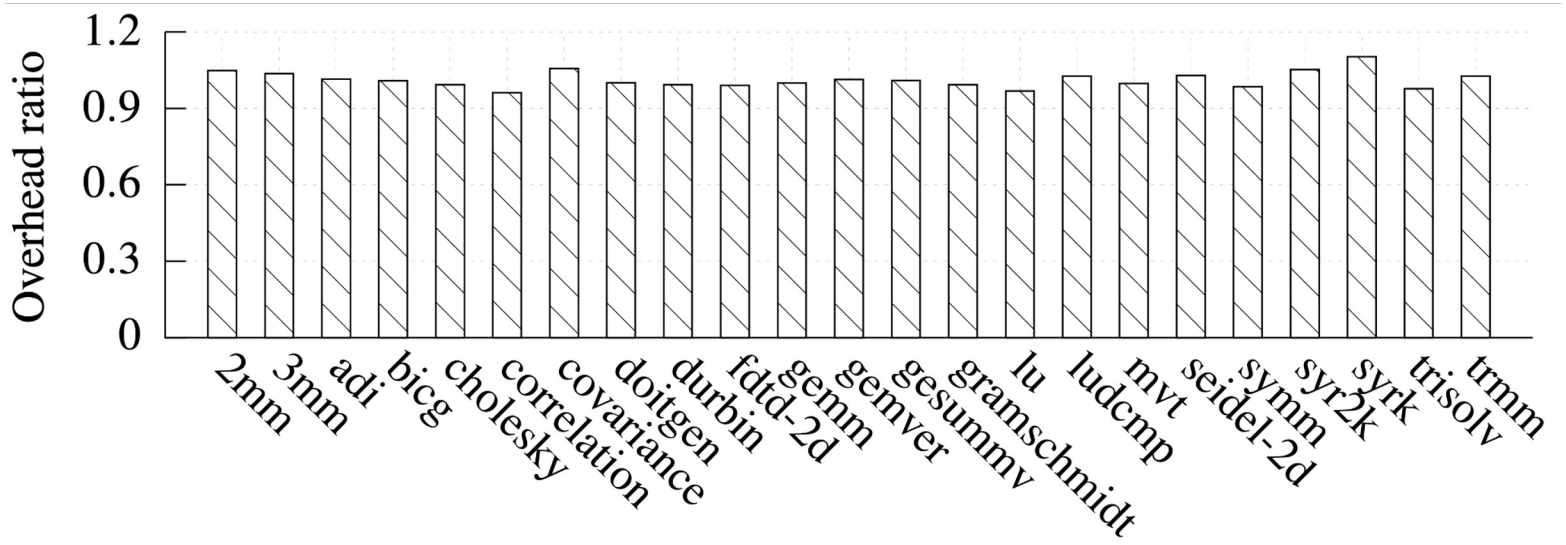
Cache hit patterns of the first AES T-Table (Te0) under Prime+Probe attack.

Performance Evaluation

- The performance of SENSE is evaluated on each module:
  - Subscription Module
  - Notification Module
  - Action Module
- PolyBenchC
  - Kernel functions that perform mathematical operations using matrices
  - Kernel functions as critical sections in TEEs monitored by SENSE
  - One matrix is monitored for cache eviction events

Performance Evaluation (Subscription Module)

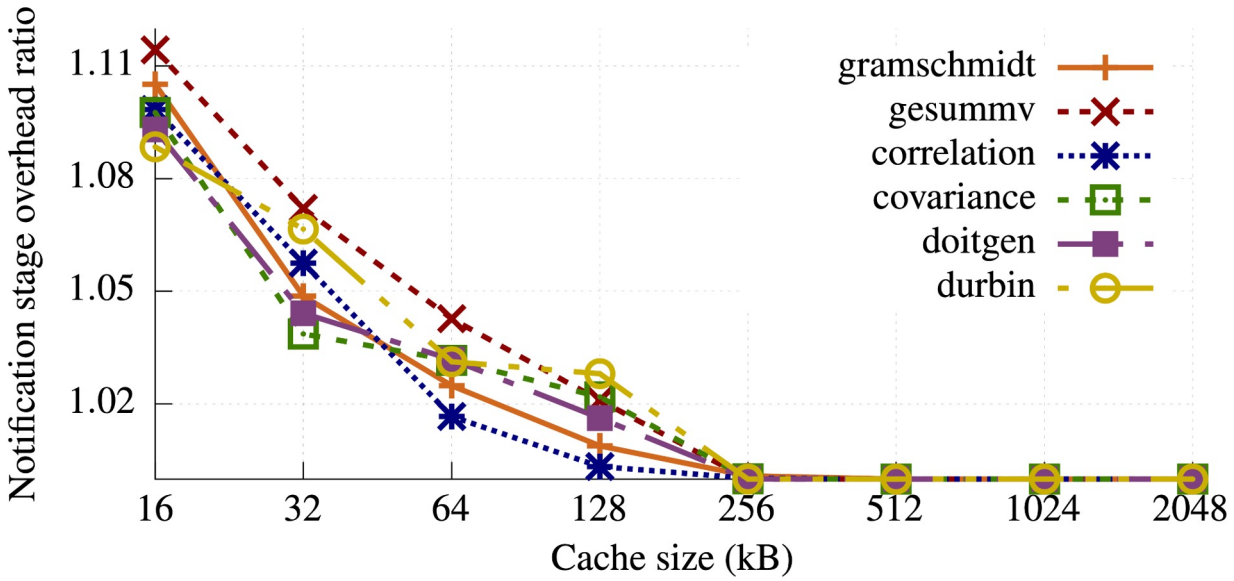
- Initialization: registers trampoline and turns on CPU SENSE mode
- Preparation: prefetching the secret data and mark as monitored
- Average overhead is **1.2%**



Performance overhead of Subscription Module

Performance Evaluation (Notification Module)

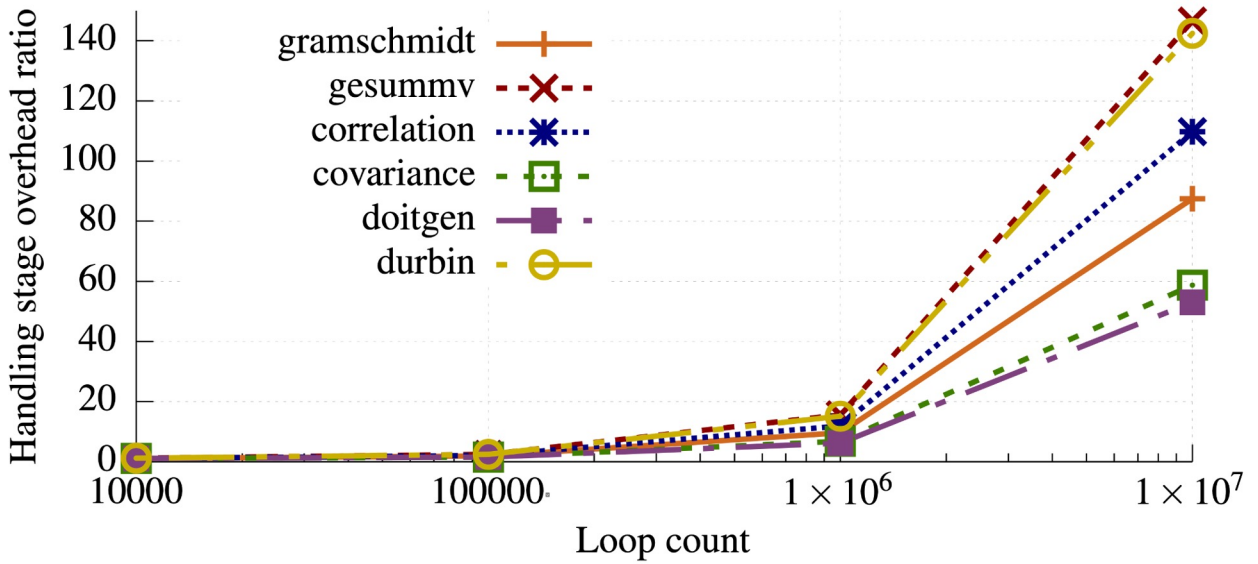
- CPU micro-code for state cleaning and control transfer
- Few events with a large cache size
- Decrease cache size to magnify the behavior



Performance overhead of Notification Module

Performance Evaluation (Action Module)

- Dummy handlers that performs simple operations inside a loop
- Increase the loop variable to simulate the increase of handler complexity
- Cache size of 32 kB (effective to show behaviors)
- Not surprisingly, the performance of the handler dominates



Performance overhead of Action Module

Conclusion

- Addressing **information asymmetry** can allow TEEs to proactively defend themselves against SCAs
- SENSE to turn a side channel exploited by attackers into a direct channel dedicated to users of TEEs
- Performance overhead of 1.2% under benign situations
- Does not degrade security of the TEE

Conclusion

- Addressing **information asymmetry** can allow TEEs to proactively defend themselves against SCAs
- SENSE to turn a side channel exploited by attackers into a direct channel dedicated to users of TEEs
- Performance overhead of 1.2% under benign situations
- Does not degrade security of the TEE

https://github.com/sslab-gatech/Sense

Thank You! (And Q&A)

Fan Sang[*,1], Jaehyuk Lee[1], Xiaokuan Zhang[3], Meng Xu[4],
Scott Constable[†,2], Yuan Xiao[2], Michael Steiner[2], Mona Vij[2], Taesoo Kim[1]

[1]Georgia Institute of Technology, [2]Intel, [3]George Mason University, [4]University of Waterloo

[*]fsang@gatech.edu [†]scott.d.constable@intel.com