RISC-V Summit

Keystone Enclave

An Open-Source Secure Enclave for RISC-V

Dayeol Lee[1,2], David Kohlbrenner, Kevin Cheang[1], Cameron Rasmussen[1], Kevin Laeufer[1], Ian Fang, Akash Khosla, Chia-Che Tsai[2], Sanjit Seshia[1], Dawn Song[2,3], and Krste Asanovic[1,2]

University of California, Berkeley ※

Collaborators: Ilia Lebedev[4], and Srinivas Devadas[4]

Berkeley UNIVERSITY OF CALIFORNIA

1 ADEPT

2 riselab UC Berkeley

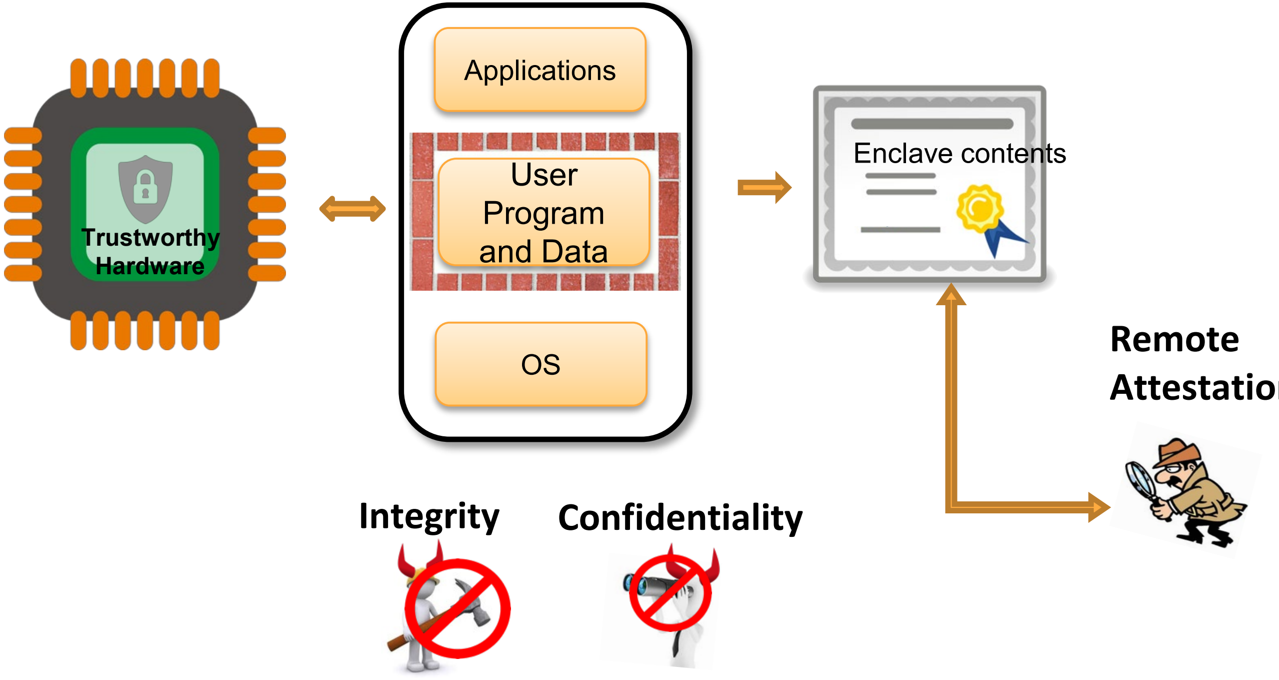
3 OASIS LABS

4 MIT Massachusetts Institute of Technology

※All authors are affiliated with the UCB

Berkeley Architecture Research

What is a Secure Enclave?



Secure Enclave as a Cornerstone Security Primitive

- Strong security capabilities
  - Authenticate itself (device)
  - Authenticate software
  - Guarantee the integrity and privacy of remote execution
- A cornerstone for building new security applications
  - Confidential computing in the cloud (e.g., machine learning)
  - Secure IoT sensor network

Why do we need an Open-Source Enclave?

- Existing enclave systems are proprietary and difficult to experiment with
  - Closed-source commercial hardware (e.g., Intel SGX, ARM TrustZone)
  - Lack of good research infrastructure
- A Lot of Challenges for Enclaves
  - Hardware vulnerabilities: Intel SGX - ForeShadow (USENIX’18), AMD SEV - SEVered (EuroSec’18)
  - Side channel attacks and physical attacks
  - Important questions: do patches really fix the problem? Are there any other issues?

Open Source Design

- **Provides transparency & enables high assurance**
- **Builds a community to help people work on the same problems**

Keystone Enclave

Keystone: Open Framework for Secure Enclaves

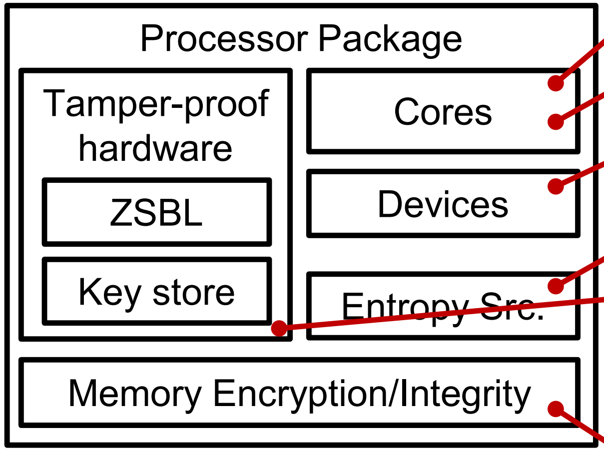
- The First Full-Stack Open-Source Enclave for Minimal Requirements
  - Root of trust, security monitor, device driver, SDK, …
  - Memory isolation, secure bootstrapping, remote attestation, …
- Memory Isolation only with Standard RISC-V Primitives
  - RISC-V Privileged ISA (U-, S-, and M-mode support)
  - Physical Memory Protection (PMP)
  - Demonstrate in unmodified processors
- Open Framework: Built Modular & Portable for Easy Extension
  - Platform-agnostic isolated execution environment
  - Platform-specific threat models (cross-core side channels, untrusted external memory, etc)
  - Use various entropy sources/roots of trust in different platforms

github.com/keystone-enclave

Earlier Work: Sanctum

- The First Enclave Design in RISC-V ISA
  - V. Costan et al., USENIX Security ’16
  - Proof of concept in C++ (https://github.com/pwnall/sanctum)
- Non-standard Hardware Extension
  - PMP was introduced in 2017 (RISC-V Priv. v1.10)
- Keystone and Sanctum
  - Keystone was built from scratch
  - Keystone shares many good practices from prior experiences of Sanctum
  - The primary goal of Keystone is to make an open end-to-end framework

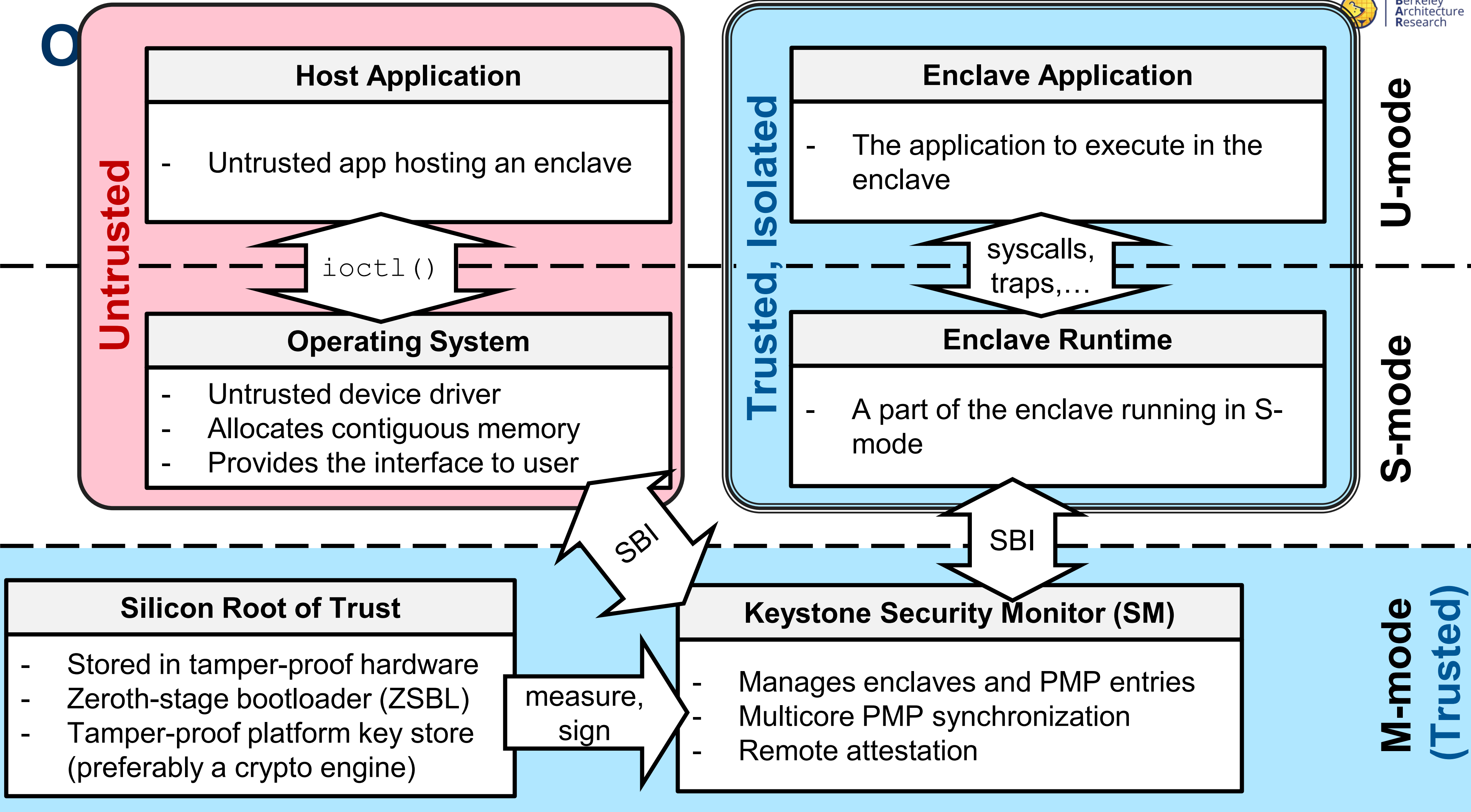
What Hardware Do We Need?



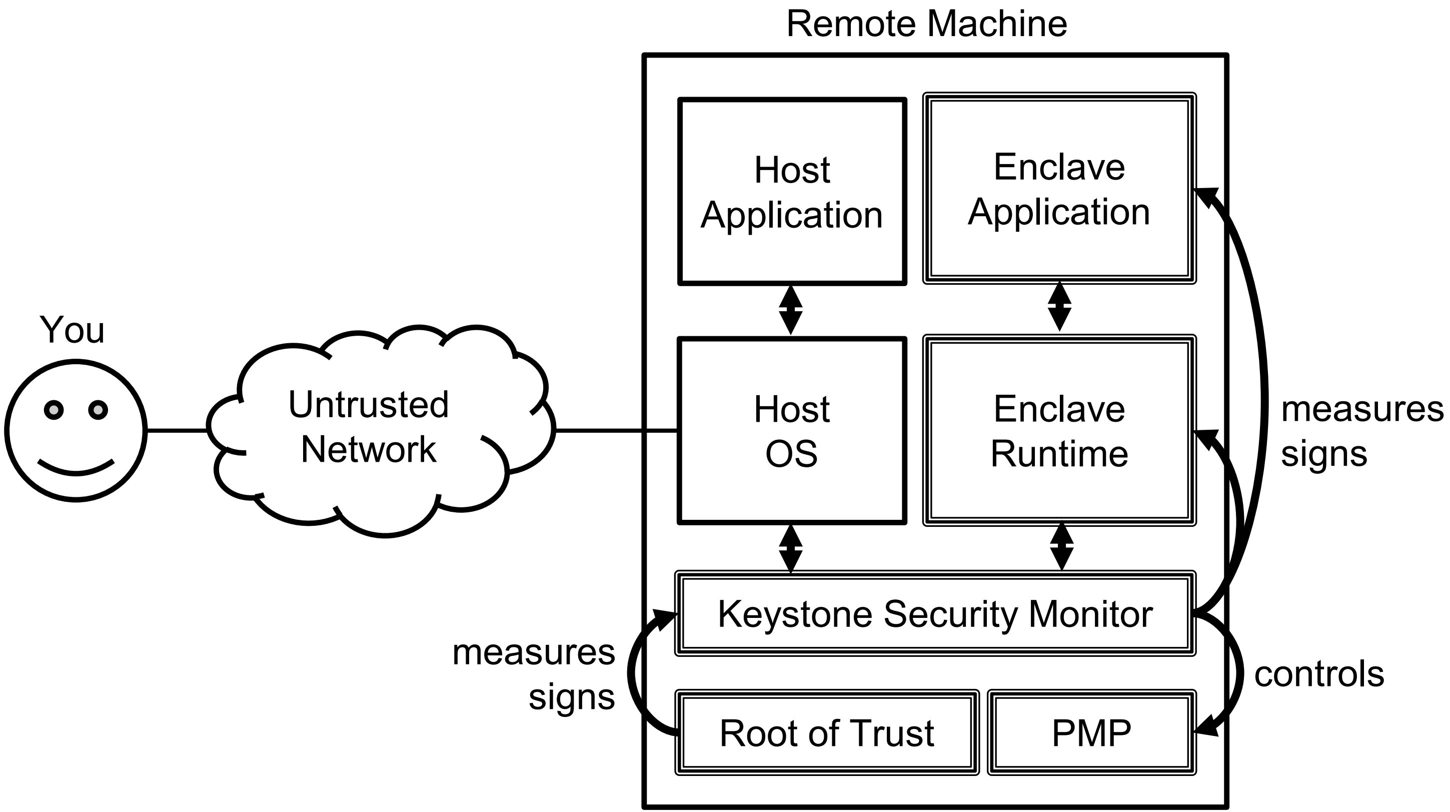
- RISC-V Physical Memory Protection (PMP)
- RISC-V U-, S-, and M-mode
- (RISC-V) Device Gasket PMP (i.e., iopmp)
- An Entropy Source available at boot
- Root of Trust (preferably a crypto engine)
  - Measuring & signing the security monitor
  - Platform key store
- If untrusted/external DRAM – memory encryption/integrity engine (not implemented yet)

O

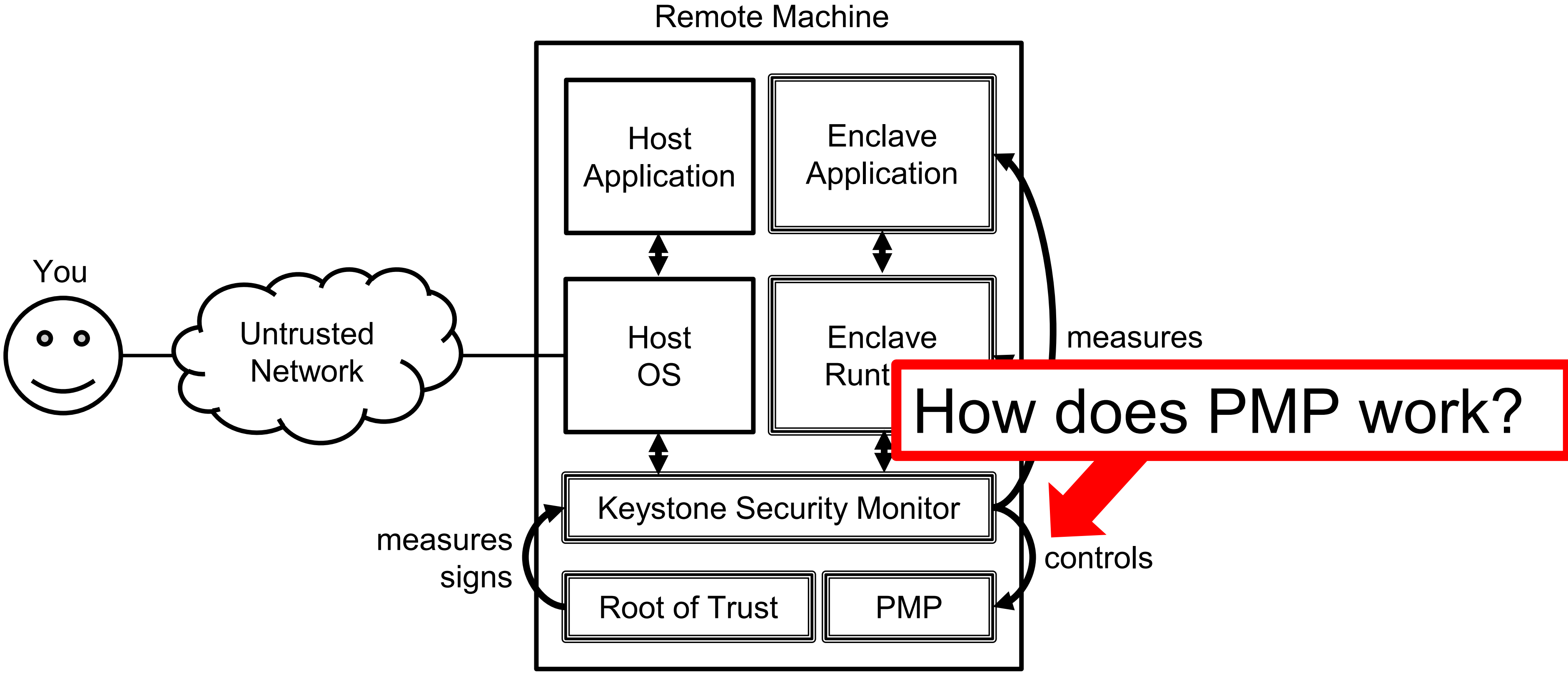
Berkeley Architecture Research



Keystone Overview (Simplified)

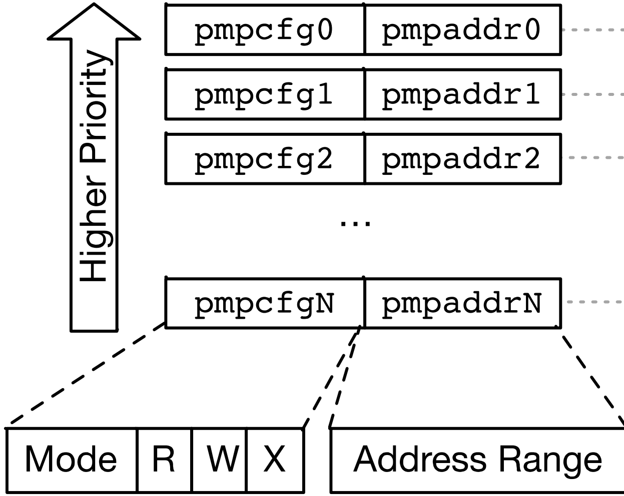


Keystone Overview (Simplified)



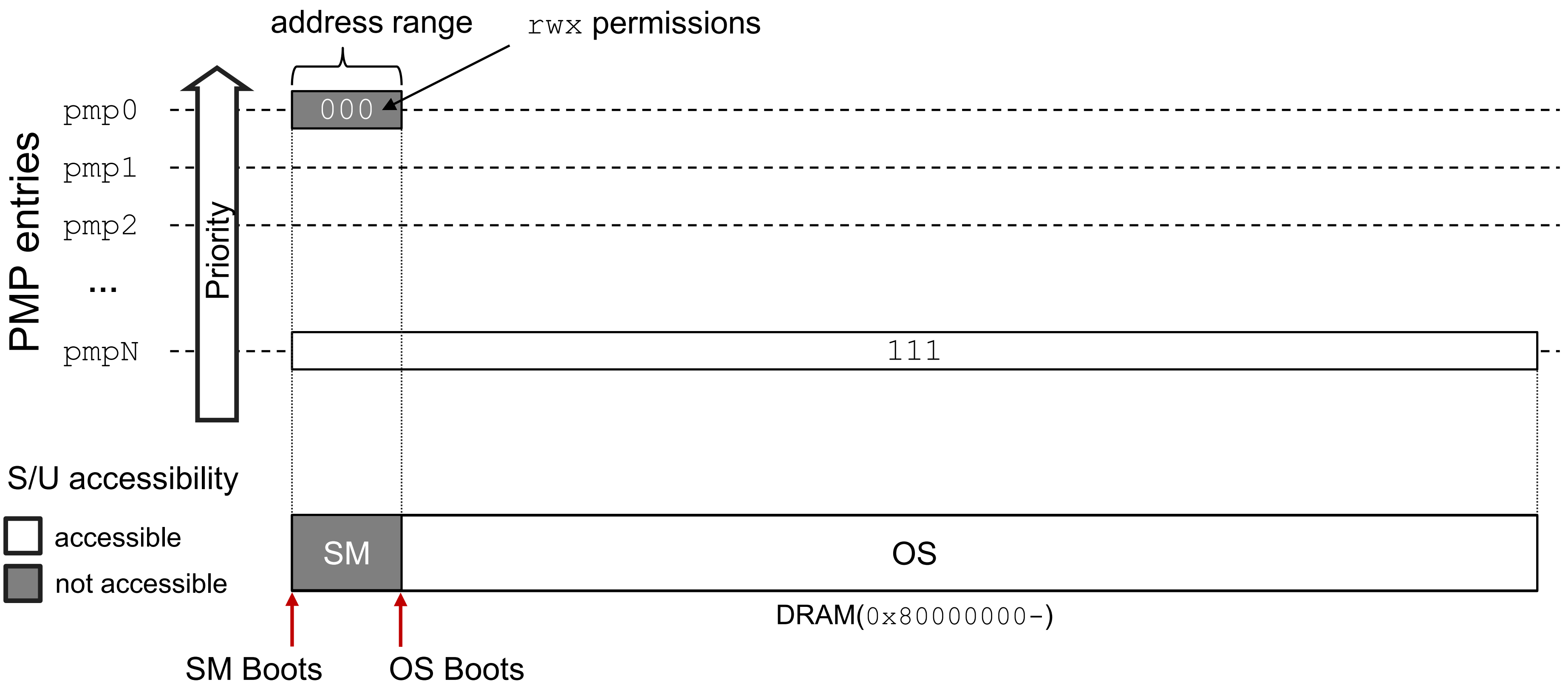
Memory Isolation with RISC-V PMP

- Physical Memory Protection (PMP)
  - Special registers to control permissions of U- and S-mode accesses to a specified memory region
  - # of PMP entries can vary (e.g., default Rocket has 8)
  - Statically prioritized by the order of entry indices
  - Whitelist-based
  - Dynamically configurable by M-mode
  - Addressing modes: NAPOT (>= 4-bytes), Base/Bound
- How Keystone uses PMP
  - Top/bottom PMP entries are reserved for SM/OS
  - 1 PMP entry for each “active” enclave
  - NAPOT > 4KB (fragmentation / Linux buddy allocation)

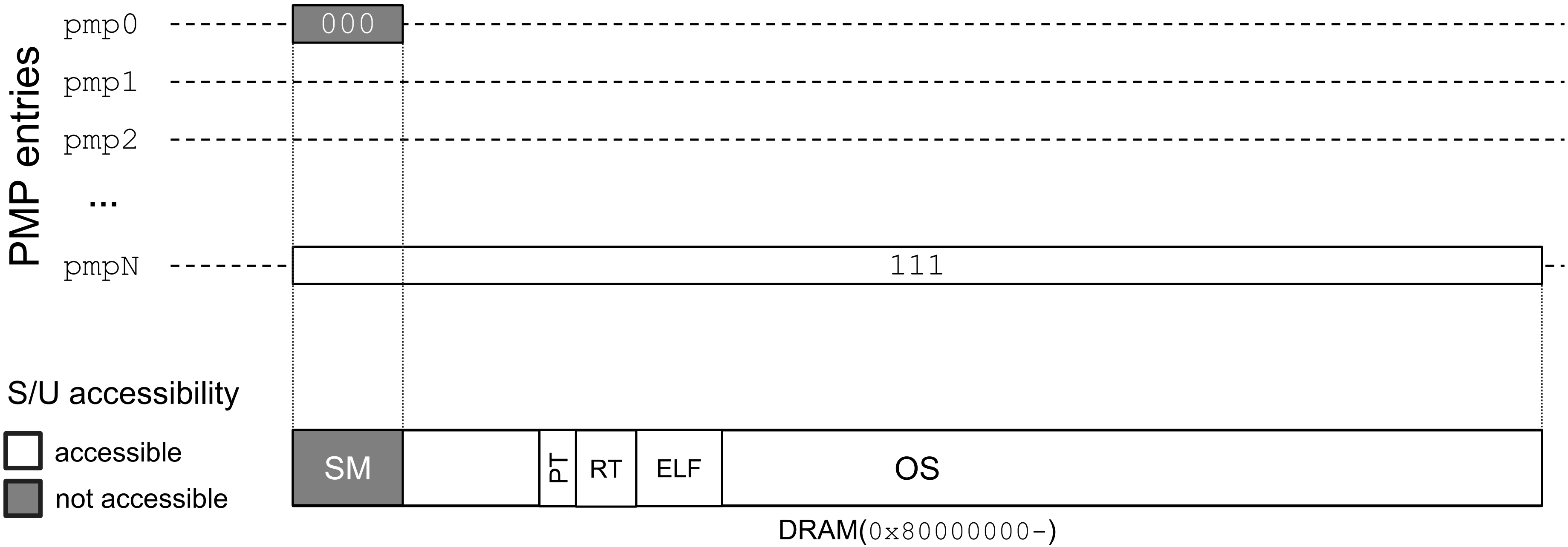


PMP registers

Isolation via Switching PMP Permission Bits



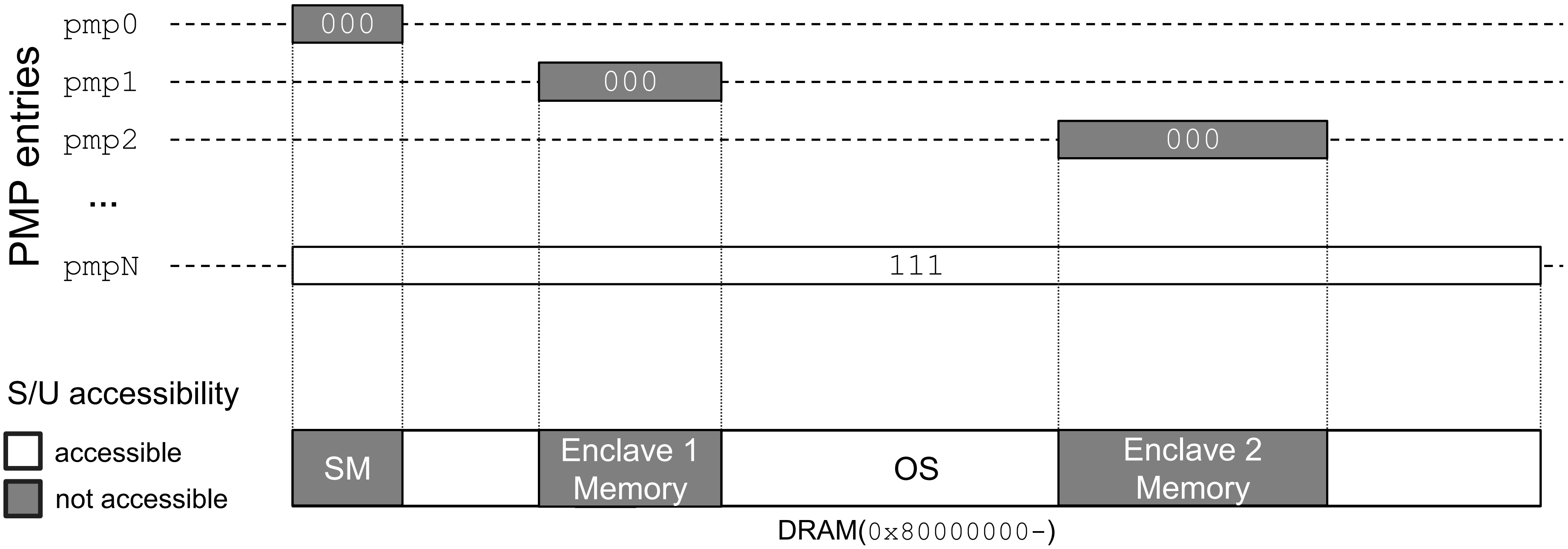
Creating an Isolated Enclave



OS allocates a contiguous chunk of memory using `__get_free_pages()` and initializes the free pages with the enclave page table, and the enclave program (runtime + enclave application)

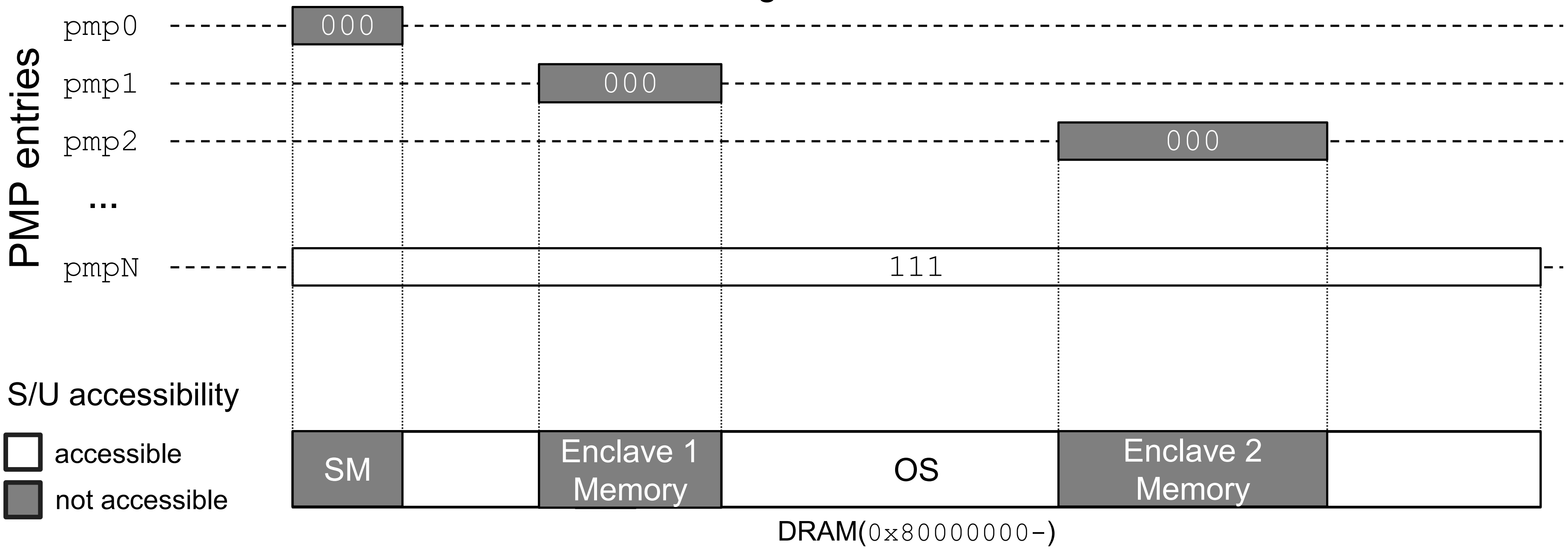
Creating an Isolated Enclave

SM sets PMP entry and finalizes the enclave hash

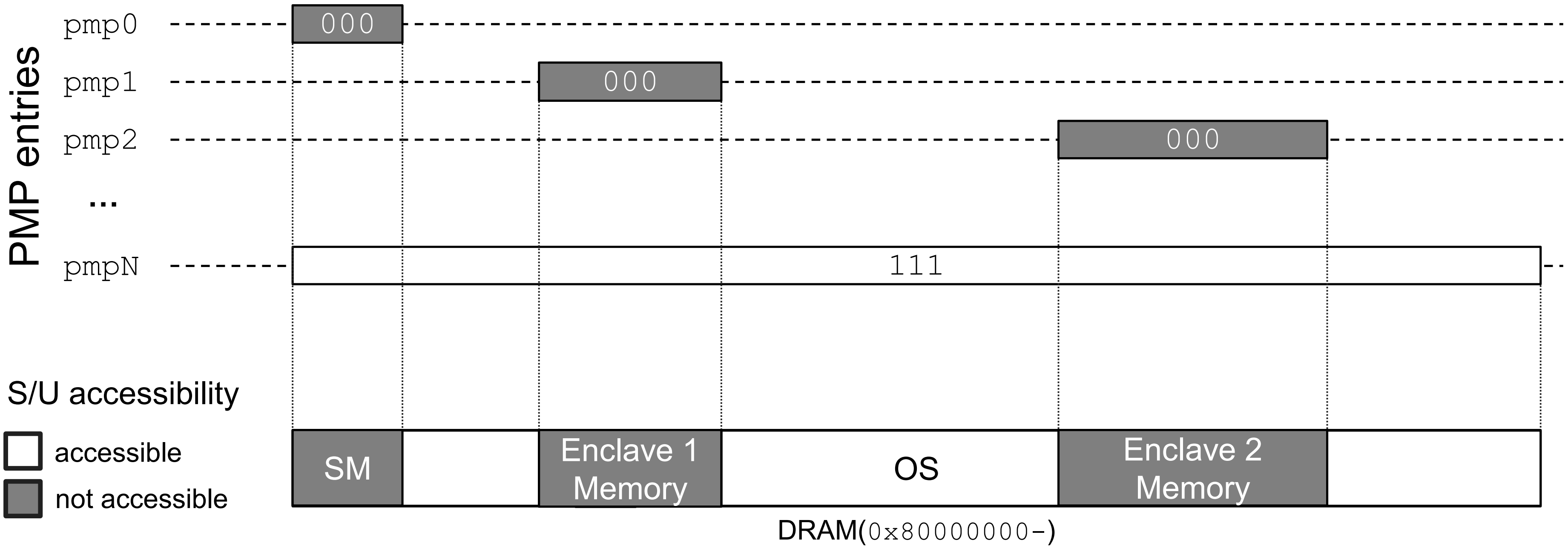


Creating an Isolated Enclave

OS can ask SM to create as many enclaves as the number of remaining PMP entries

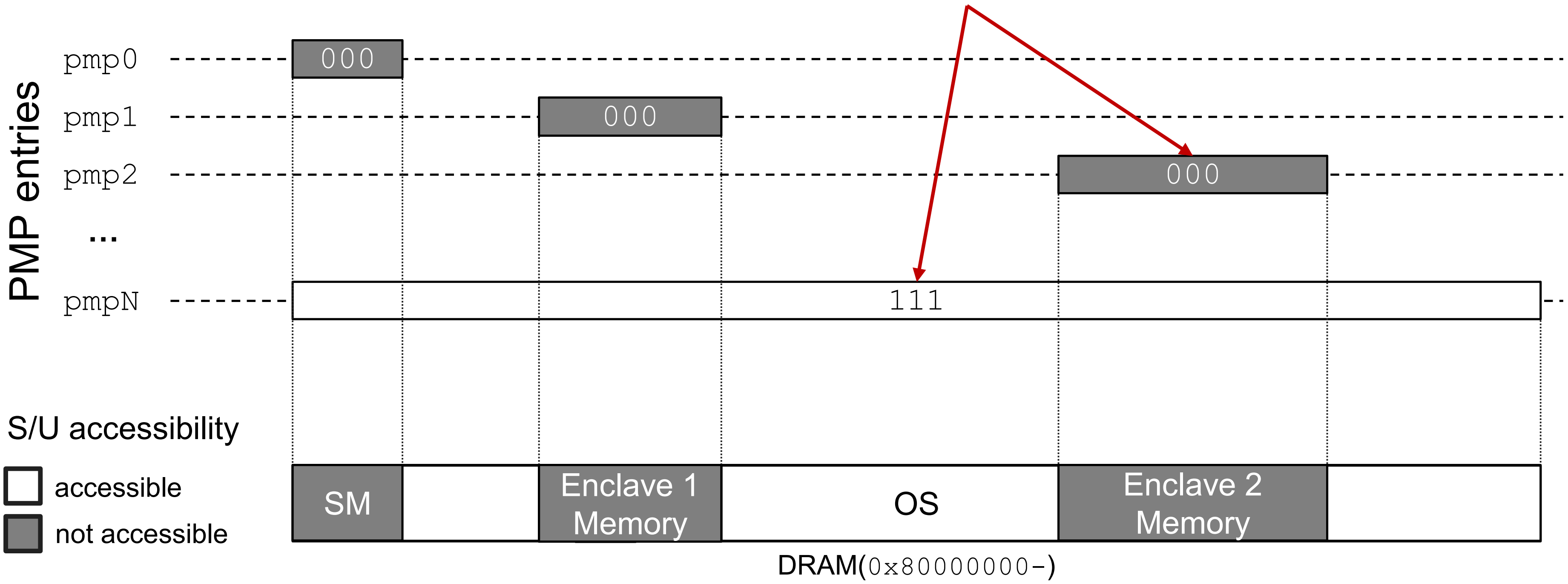


Executing an Enclave



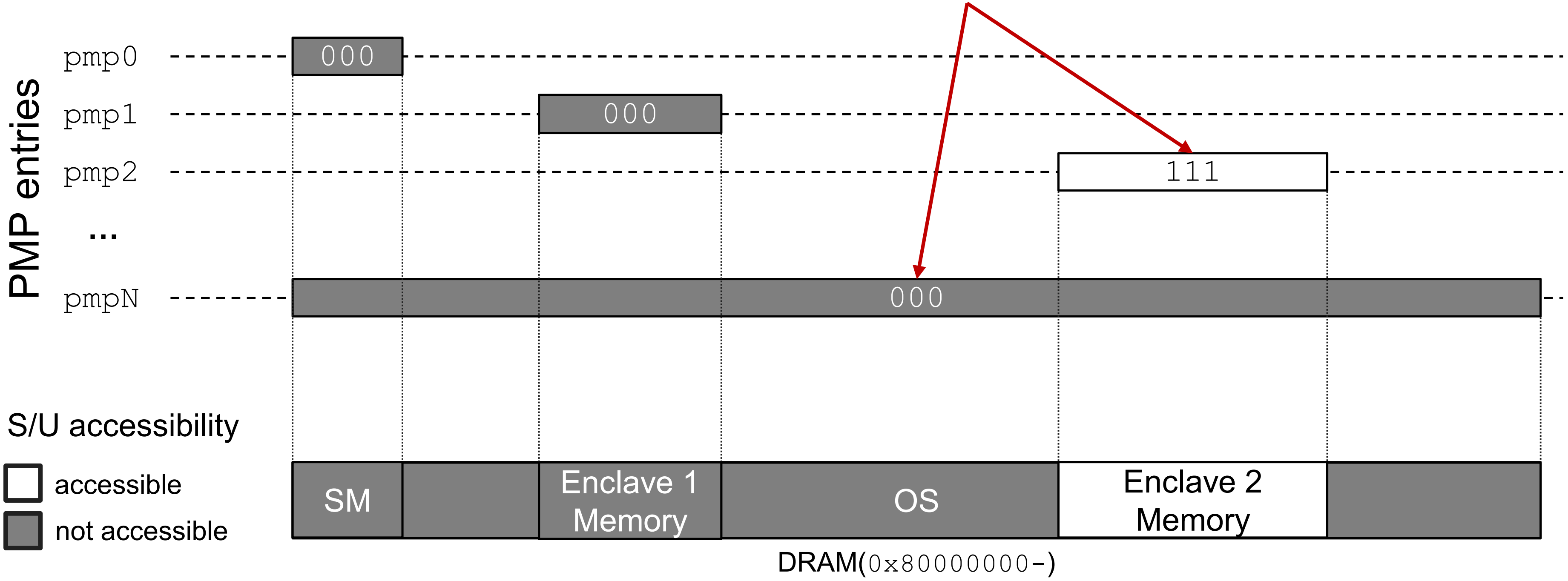
Executing an Enclave

SM flips the PMP permission bits of `pmp2` and `pmpN` to execute Enclave 2



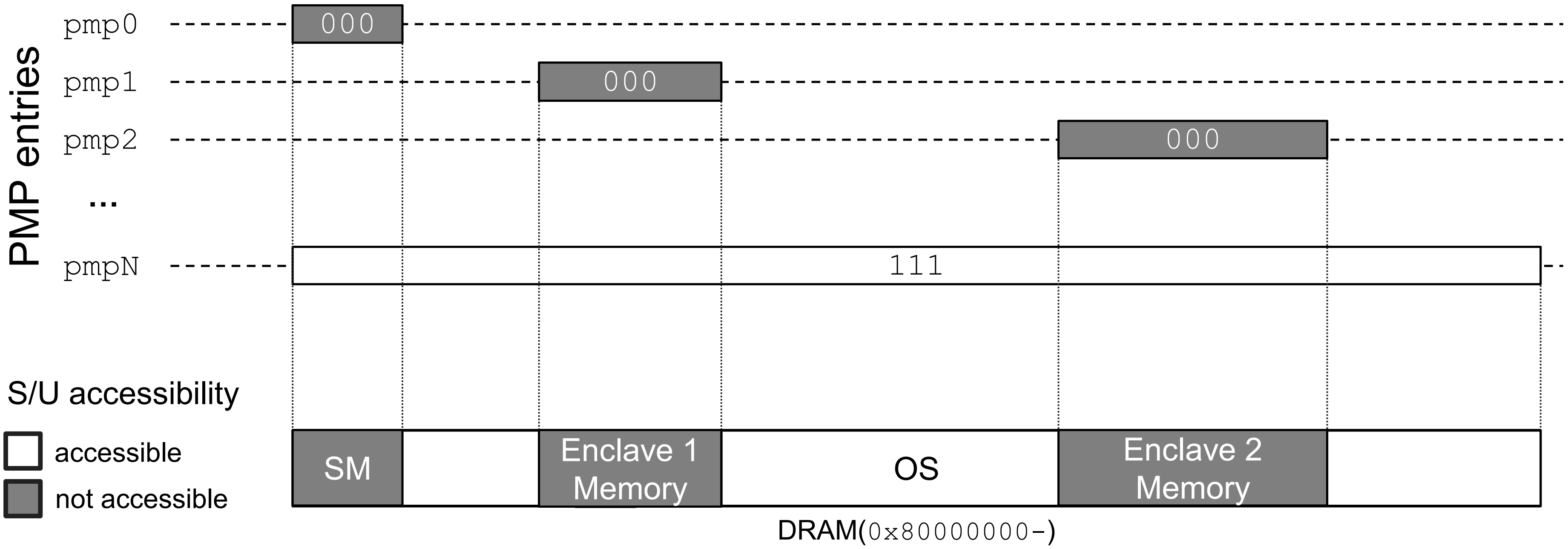
Executing an Enclave

SM flips the PMP permission bits of `pmp2` and `pmpN` to execute Enclave 2

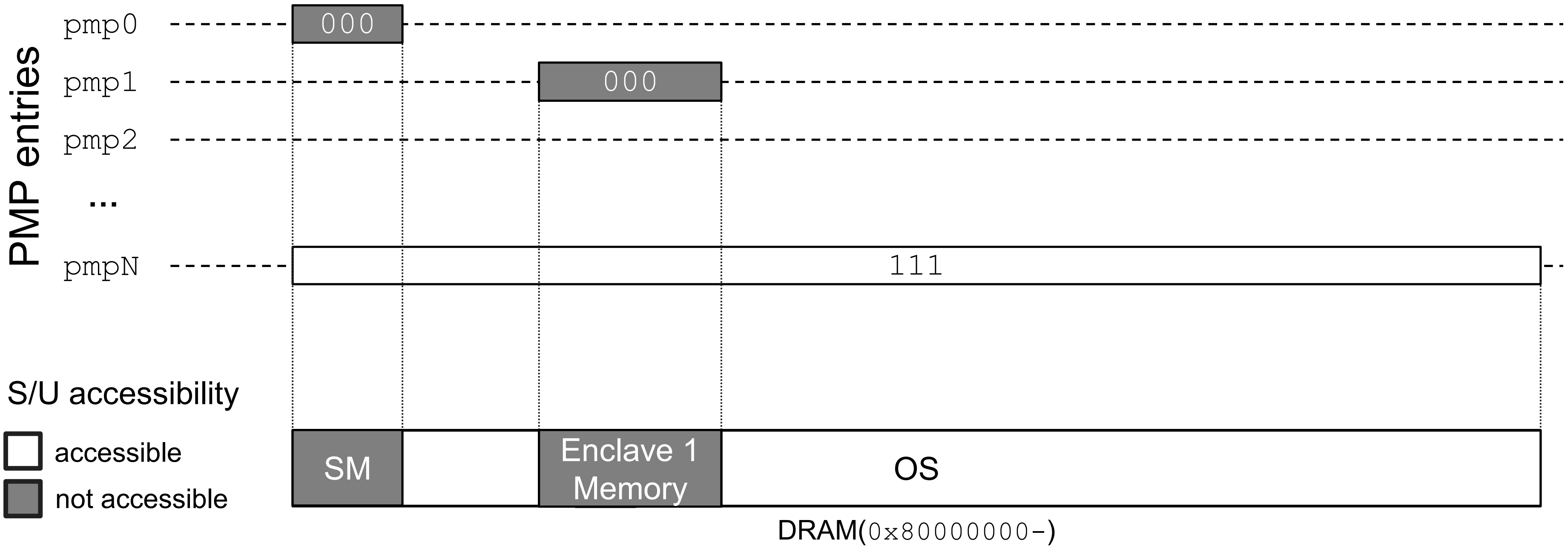


(Asynchronous) Exit and Resume

The enclave can only exit by an SM SBI call.
The SM flips the permissions before entering the untrusted context.



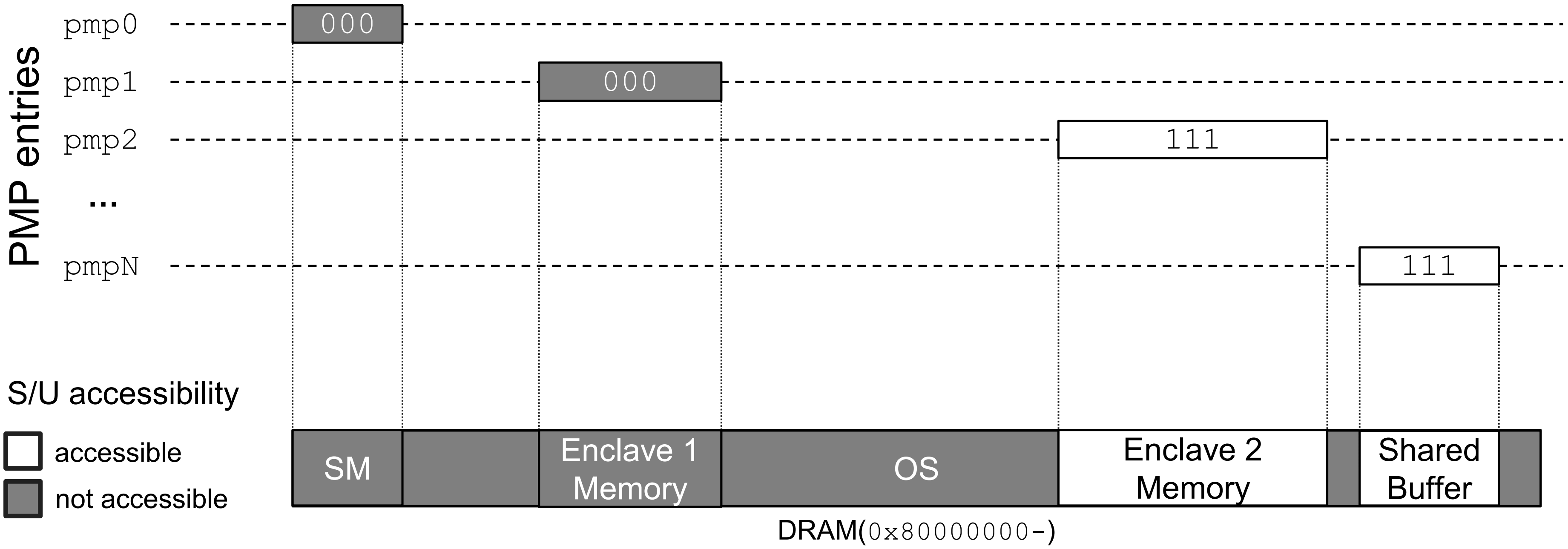
Destroying an Enclave



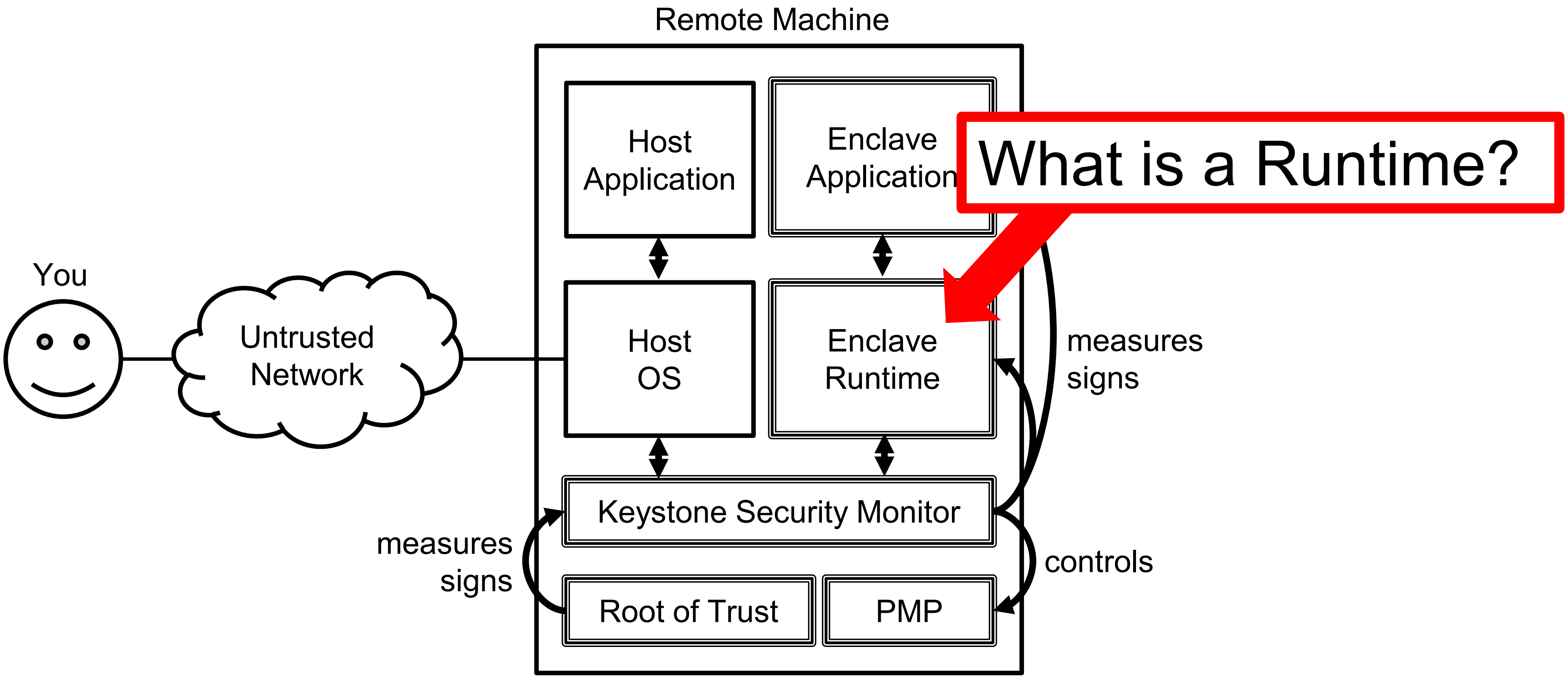
Untrusted Shared Buffer

The OS can allocate a shared buffer in OS memory

The SM uses the last PMP entry to allow the enclave to access the buffer.

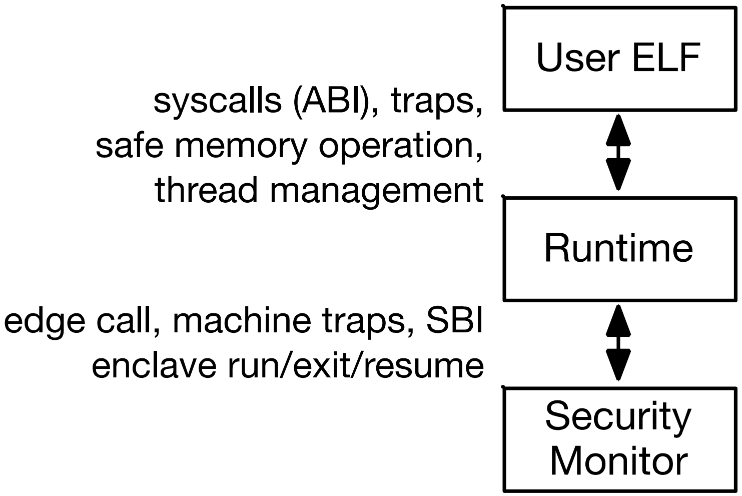


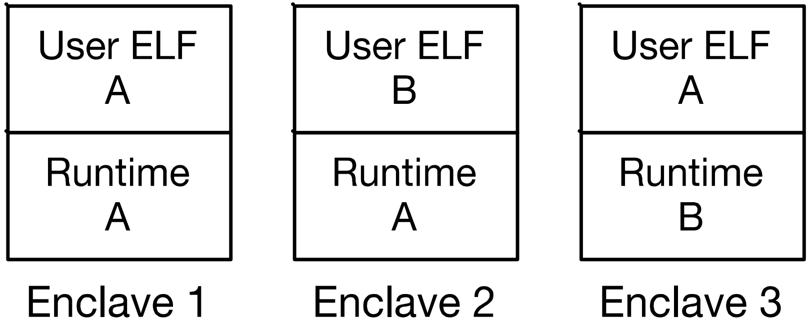
Keystone Overview Revisited



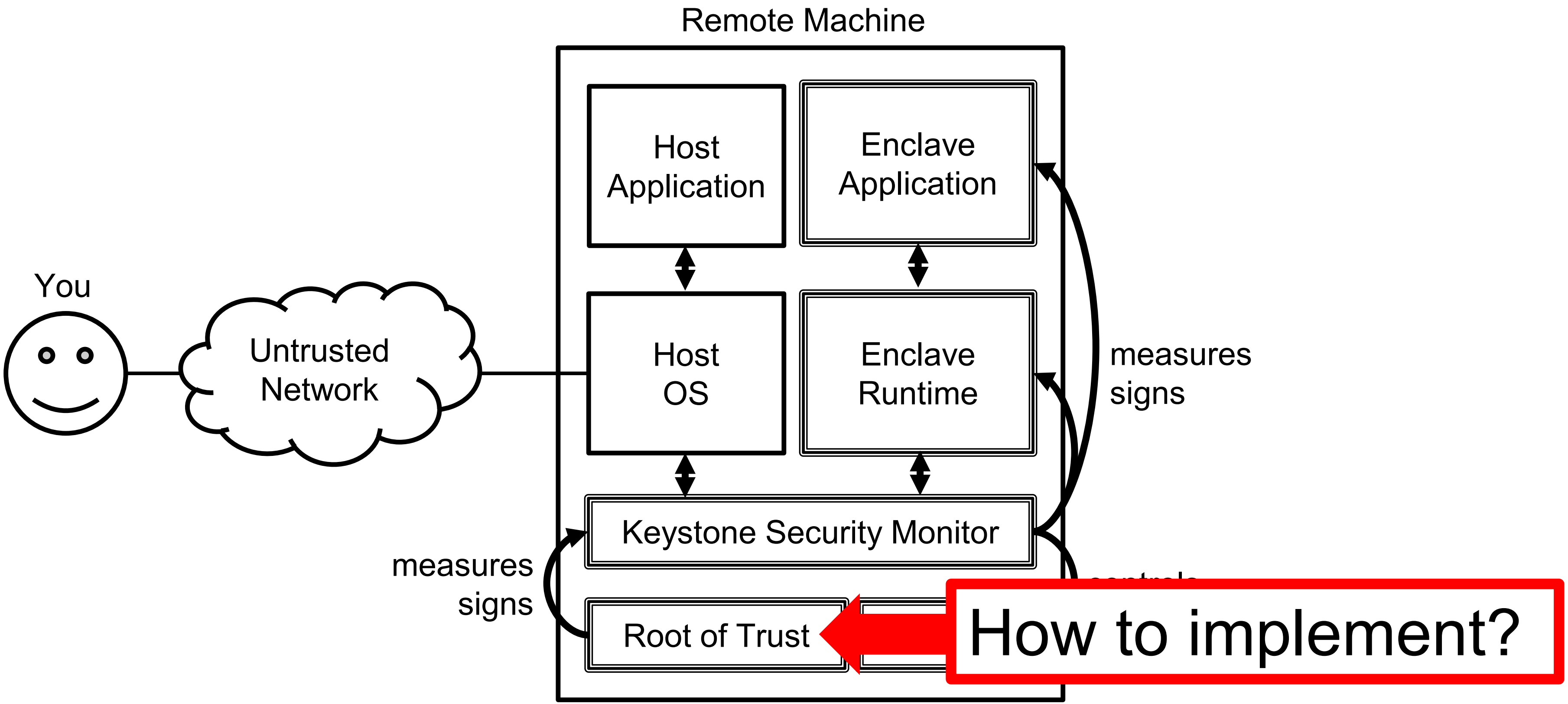
S-Mode Enclave Runtime

- Provides Kernel-like Functionality
  - Syscalls, traps
  - thread and page table management
- Useful Layer of Abstraction
  - Least privilege of U-mode code
  - Additional functionality without complicating the SM
  - SM < 2K LoC + 5K LoC crypto lib.
- Reusability
  - Compatible with multiple user programs
  - Can act as a shield system (e.g., Haven, Graphene) in SGX





Keystone Overview Revisited



Silicon Root of Trust

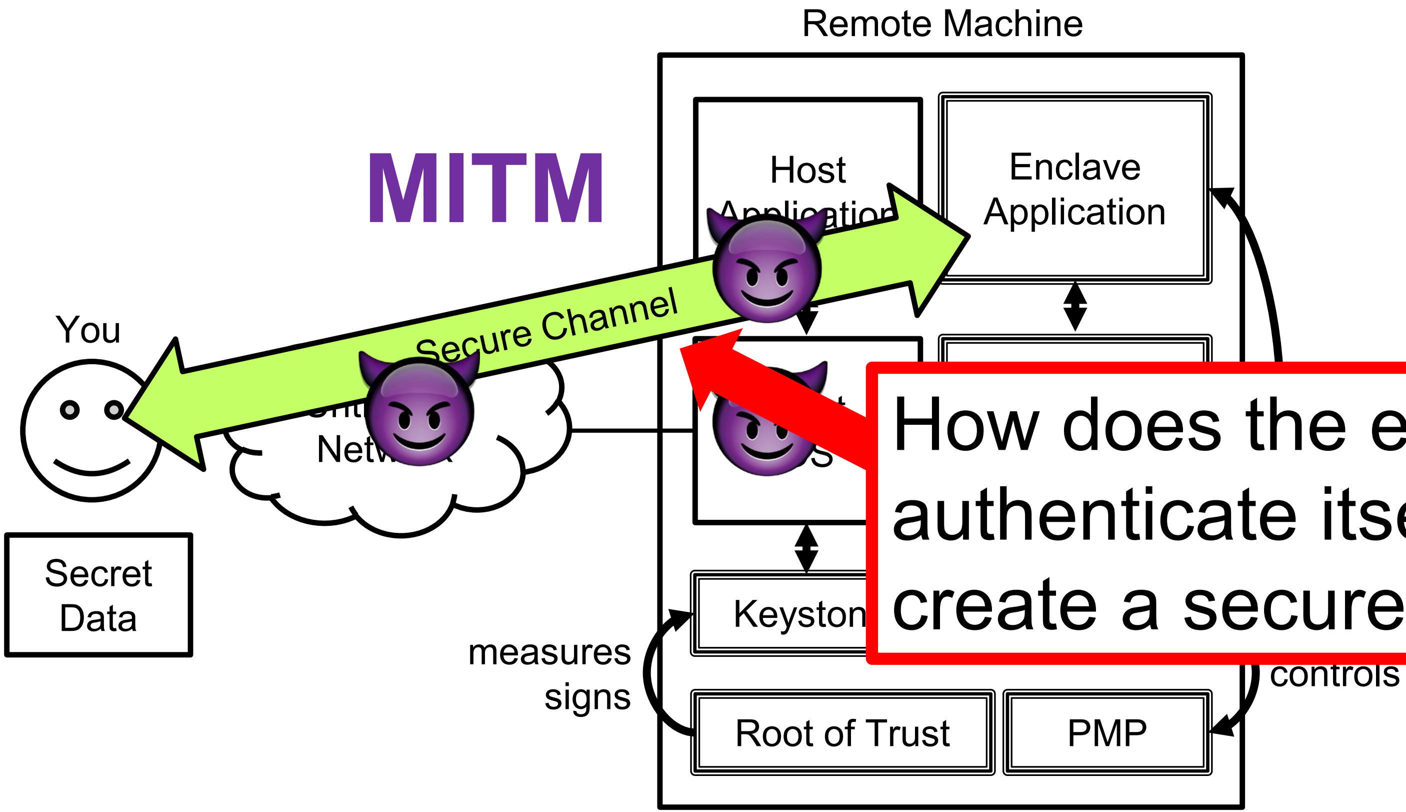
- Tamper-proof hardware that cryptographically hashes the security monitor, provisions an attestation key, and signs them with device’s secret key.
- Various ways to implement the root of trust
  - Various entropy sources, various platform key store, and implementation of the crypto engine
- Keystone uses Sanctum’s root of trust which uses ECDSA and SHA-3

4:30pm - 5:00pm 30 mins Info ⓘ
Secure RISC-V

Secure Bootstrapping of Trusted Software in RISC-V

Ilia Lebedev - Graduate Student, Massachusetts Institute of Technology

Keystone Overview Revisited

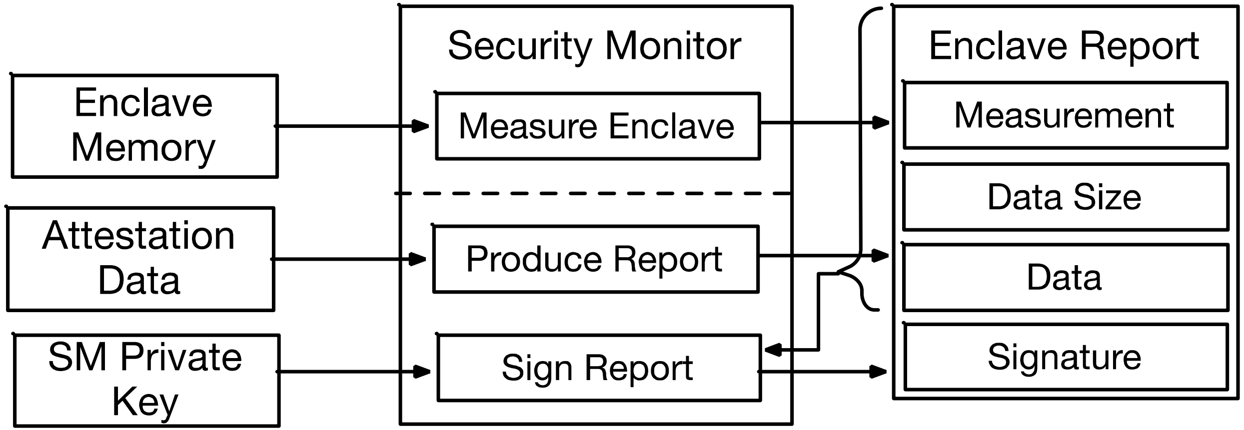


How does the enclave authenticate itself and create a secure channel?

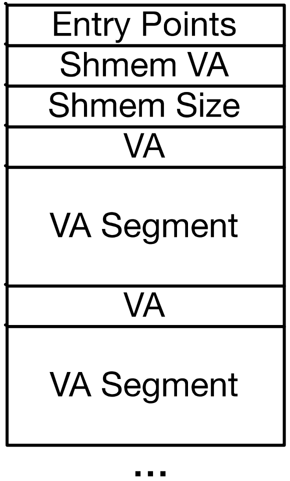
Remote Attestation

- SM measures the enclave upon enclave creation
- Enclave may bind a key to the enclave report
- SM signs the enclave report and hands it (+ SM report) to the user

The Full Process of Attestation



Measurement Layout



Project Status

- Testable in Various Platforms
  - Latest RISC-V QEMU: functionality test, development
  - Latest FireSim (v1.4.0): performance analysis, hardware modification
  - SiFive Unleashed: runs on a real quadcore in-order processor!
- Ongoing Efforts
  - Formal verification of PMP-based security monitor
  - Mitigating cache side-channel attacks using platform features
- Contributions Needed!
  - Building software stack: more use cases, libraries, edge compiler, …
  - Adding software/hardware extensions
    e.g., demand paging, memory encryption/integrity, multithreading, CMA integration, …

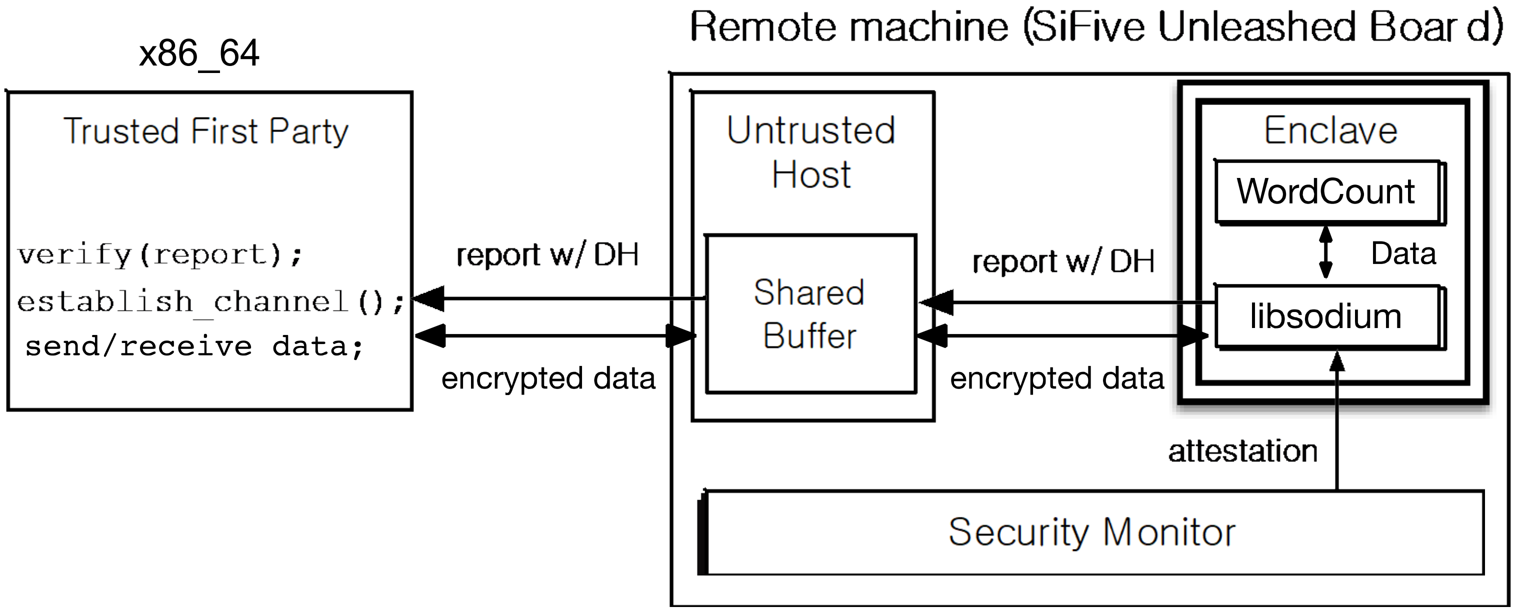
Project Links

- Deployment:
  - **QEMU**: **https://github.com/keystone-enclave/keystone**
  - FireSim: https://github.com/keystone-enclave/keystone-firesim
  - SiFive Unleashed: https://github.com/keystone-enclave/keystone-hifive-unleashed
- Keystone Repository:
  - Keystone-SDK: https://github.com/keystone-enclave/keystone-sdk
  - Device Driver: https://github.com/keystone-enclave/riscv-linux
  - Security Monitor: https://github.com/keystone-enclave/riscv-pk
  - A Simple Runtime: https://github.com/keystone-enclave/keystone-runtime
  - Demo: https://github.com/keystone-enclave/keystone-demo
- Documentation (more coming):
  - Website/Blog: https://keystone-enclave.org
  - Development Docs: https://docs.keystone-enclave.org

Demo

A Remote Enclave with Secure Channel

- SiFive Unleashed board + simulated non-standard hardware
  - Root of trust: Modified FU540 FSBL with hard-coded device key
- Successfully ported libsodium for ECDH Key Exchange



Conclusion

keystone-enclave.org

- Keystone: an Open-Source Full-Stack Enclave for RISC-V
  - Runs on standard RISC-V cores
  - Modular design for better extensibility & portability
- Use Cases
  - Secure hardware research (e.g., LLC side-channel defense w/ way partitioning + PMP)
  - Building secure systems (e.g., Secure IoT network)
- Opens up Research Opportunities around Hardware Security
  - Formal Verification of PMP and Security Monitor Implementation
  - Performance Analysis
  - Defending Side Channels & Physical Attacks
  - Multi-level Security (MLS) for Sensitive Data Analytics

Thank You!

Dayeol Lee (dayeol@Berkeley.edu)
David Kohlbrenner (dkohlbre@berkeley.edu)
Forum (keystone-enclave@googlegroups.com)