Keystone Annual Review 2023

Confidential Computing Consortium

Lily Sturmann lsturman@redhat.com

Dayeol Lee dayeolee@gmail.com

Goals of the Project

- Enable TEE on (almost) **all RISC-V processors**
  - Follow RISC-V standard ISA
  - Standard TEE specification for various RISC-V sub-ISA
- Make TEE **easy to customize** depending on needs
  - Base implementation vs. platform-specific implementation
  - Reuse the implementation across multiple platforms
- **Reduce the cost** of building TEE
  - Reduce hardware integration cost
  - Reduce verification cost
  - Integrate with existing software tools

Remarks

- Code Maintenance
  - Switched to monorepo: for a better developer experience
  - Bump OpenSBI v1.1
- The project have been very slow in 2022
  - Five people from UCB graduated at the same time, and four of them left the project
  - Less momentum from the industry
- Keystone is still a popular option in academia
  - Gained 133 yearly citations (+28% YoY)
  - 100+ forks mostly from researchers

Subproject Status

2022 Projects

Trusted Loader and Dynamic Library
Cathy Lu, Anay Wadhera → Evgeny Pobachienko

Improving Measured Boot and Attestation
Rohit Mittal → Jakob Sorensen

~~Scalable Memory Isolation with RISC-V H-extension~~
~~Aniruddha Alawani~~

~~Preventing Side-Channel Attacks on Dynamic Libraries~~
~~Cathy Lu~~

Churn

New Project

Add Keystone Support to VMWare Certifier Framework (Pending)
Jakob Sorensen, Evgeny Pobachienko

Why is the Project Stuck?

- ❑ Tight Coupling with RISC-V
  - Lack of Development Board
  - Many focused on low-end devices which is not Keystone is aiming for
  - RISC-V specification is still changing; no software standard yet
- ❑ Lack of Industry Contribution
  - Code quality geared toward research (not maintainability)
  - People leave the team after 1-2 years (usually at the same time)
- ❑ Lack of Application Demand
  - RISC-V software ecosystem is still growing, and the application demand is weak

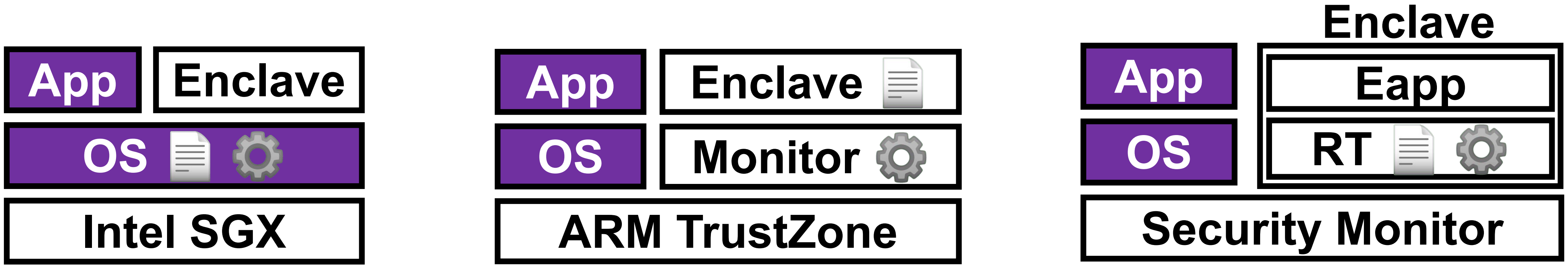
Key Milestones for 2023

- Better application support
  - Dynamic library support
- Parity with industry standards
  - Standard crypto for measured boot / attestation
- Increase dev board accessibility
  - Participate in RISC-V development board program
  - Expecting a supply chain relief in mid 2023
- Work closely with RISC-V AP-TEE working group
  - Not directly relevant, but they are interested in pushing towards server-class RISC-V TEE in the future

Thank You!

Memory Management in Keystone

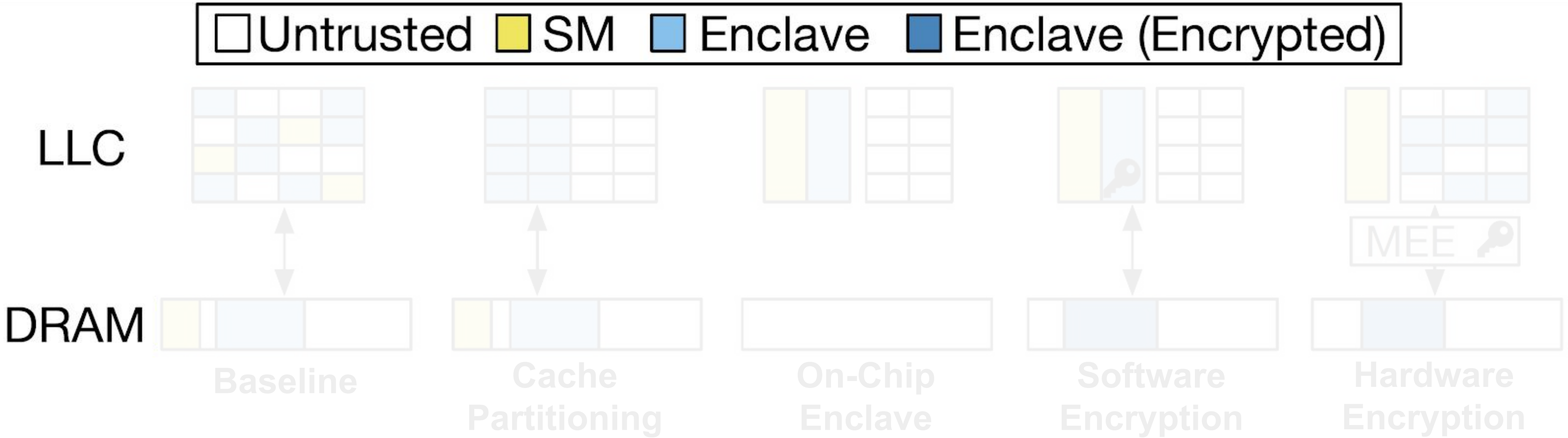
😈 = untrusted 📄 = page table ⚙️ = management



Intel SGX Komodo Keystone

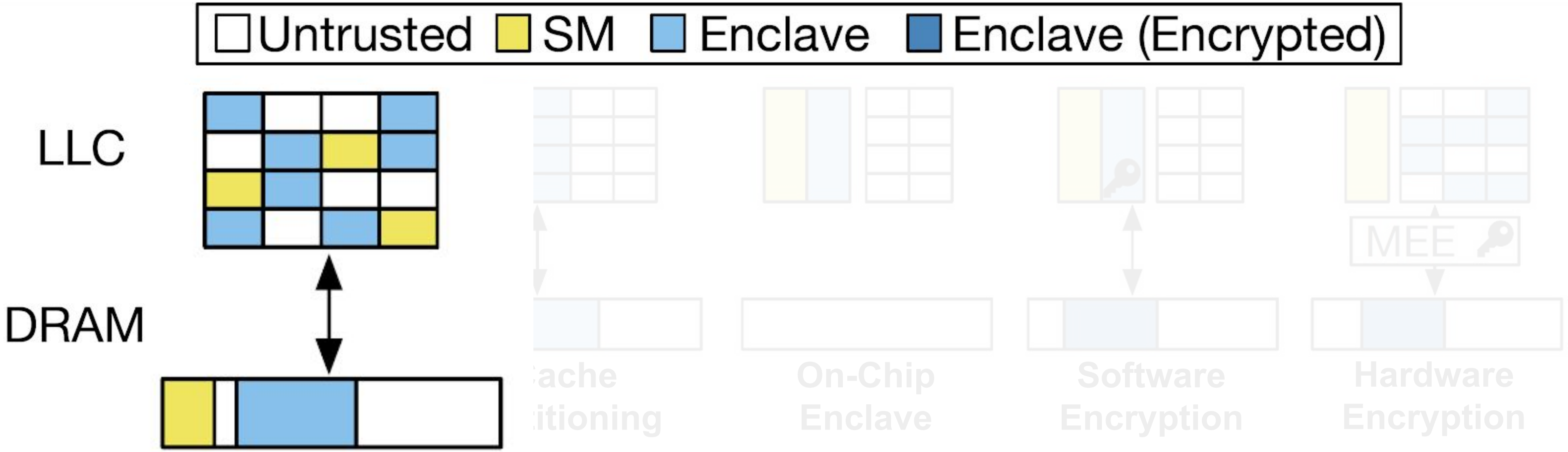
- ❑ Enclave self resource management (e.g., dynamic memory resizing)
- ❑ Various memory protection mechanisms

Various Memory Protection Mechanisms



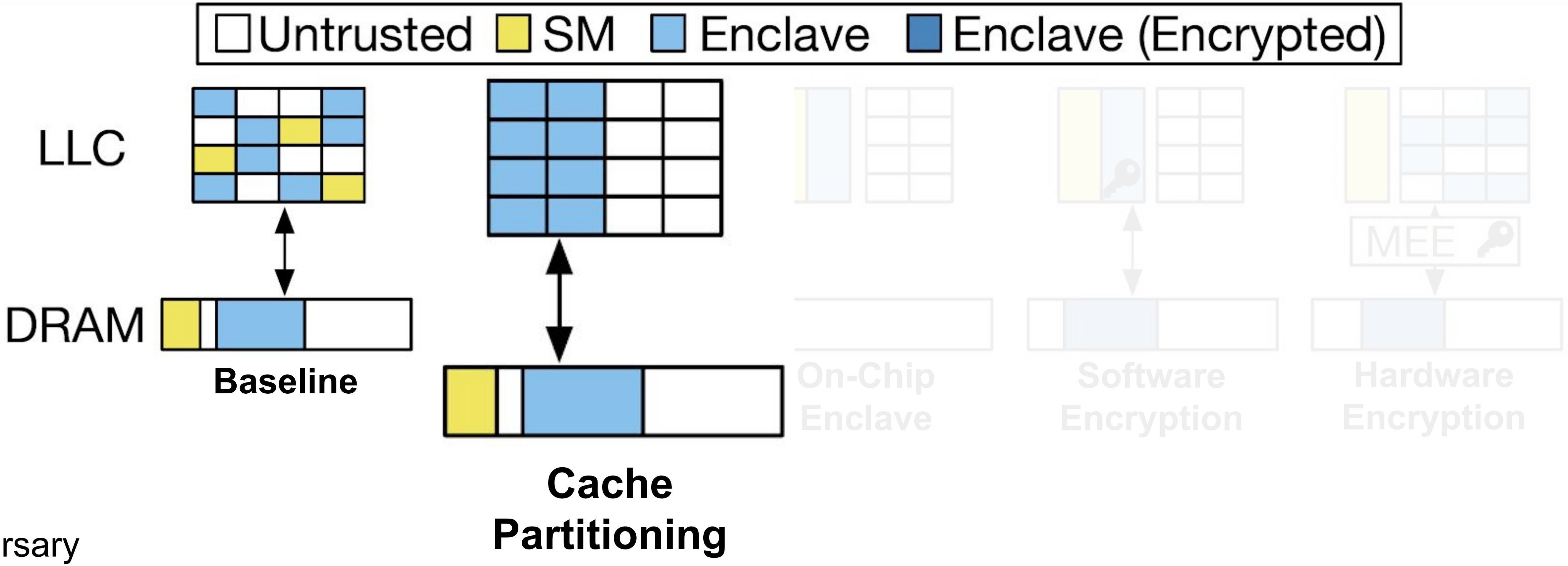
Adversary
SW
Cache SC
HW

Various Memory Protection Mechanisms



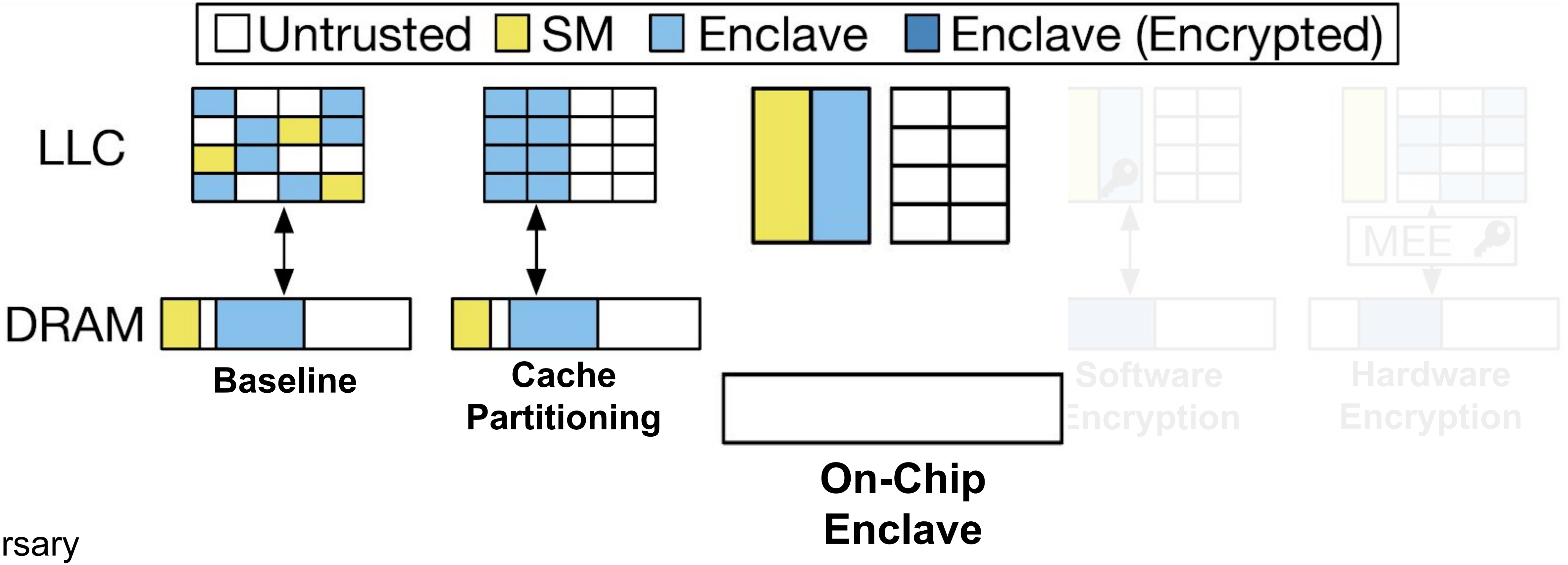
Adversary	Baseline
SW	●
Cache SC	
HW	

Various Memory Protection Mechanisms



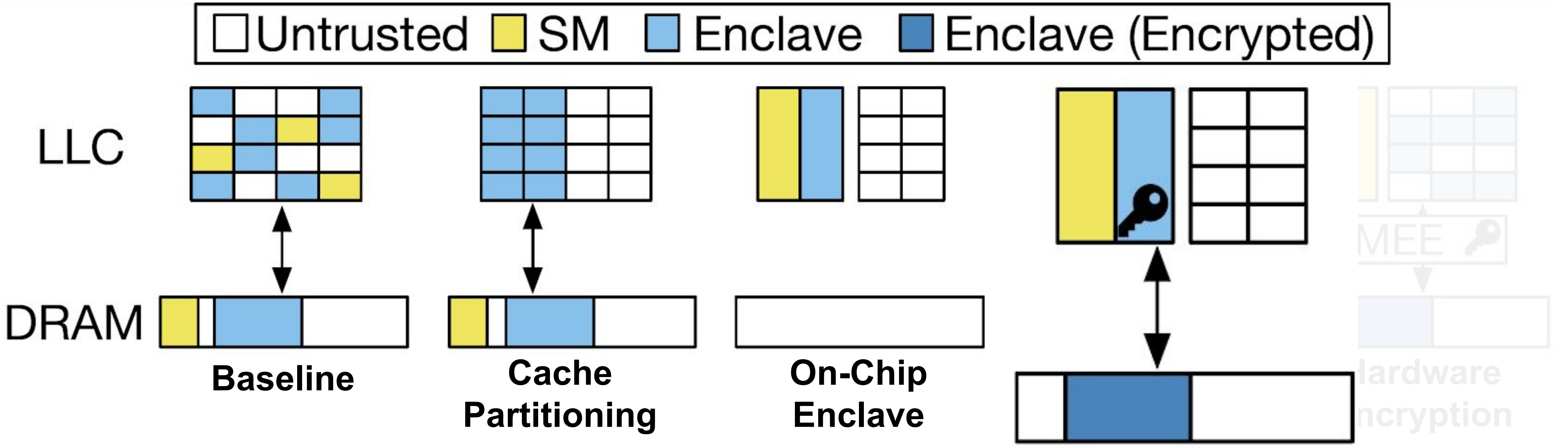
Adversary	Baseline	Cache Partitioning
SW	●	●
Cache SC		●
HW		

Various Memory Protection Mechanisms



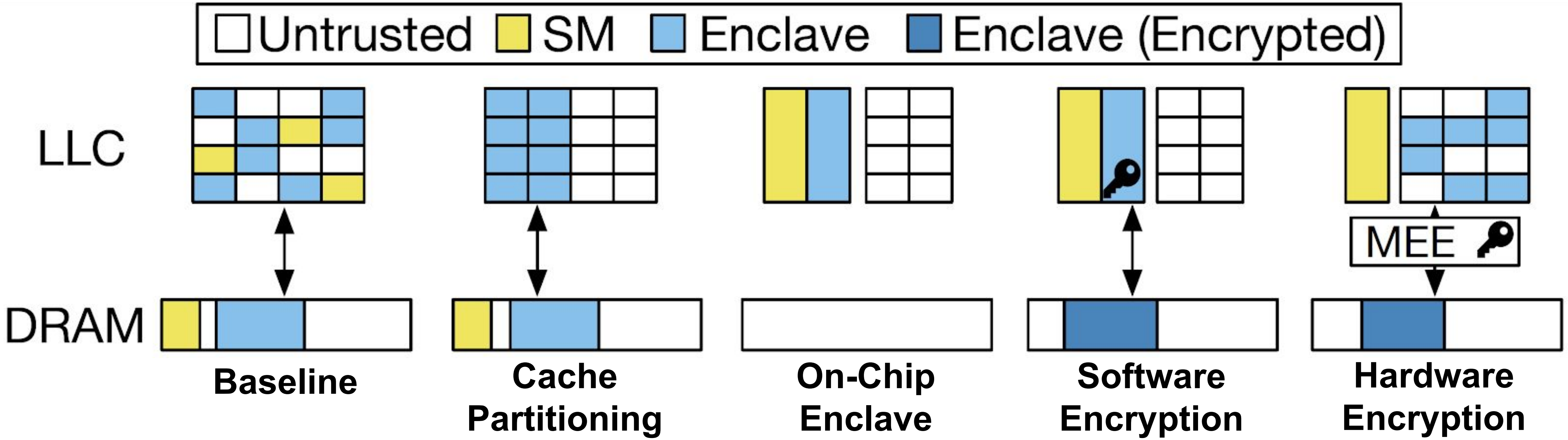
Adversary	Baseline	Cache Partitioning	On-Chip Enclave
SW	●	●	●
Cache SC		●	●
HW			●

Various Memory Protection Mechanisms



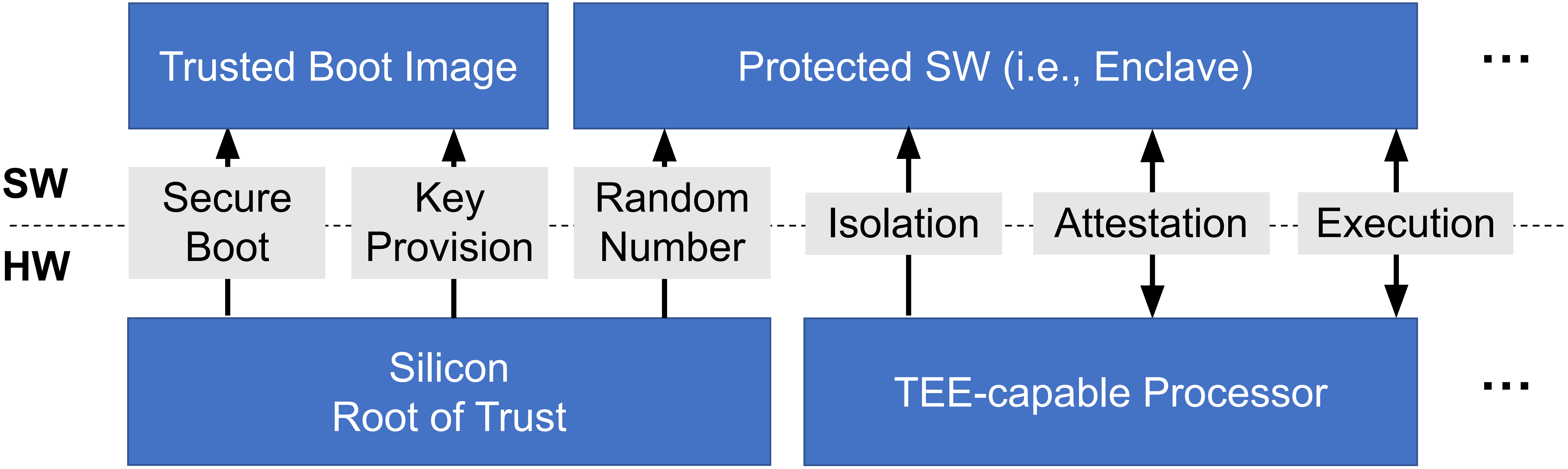
Adversary	Baseline	Cache Partitioning	On-Chip Enclave	Software Encryption
SW	●	●	●	●
Cache SC		●	●	●
HW			●	●

Various Memory Protection Mechanisms



Adversary	Baseline	Cache Partitioning	On-Chip Enclave	Software Encryption	Hardware Encryption
SW	●	●	●	●	●
Cache SC		●	●	●	●
HW			●	●	●

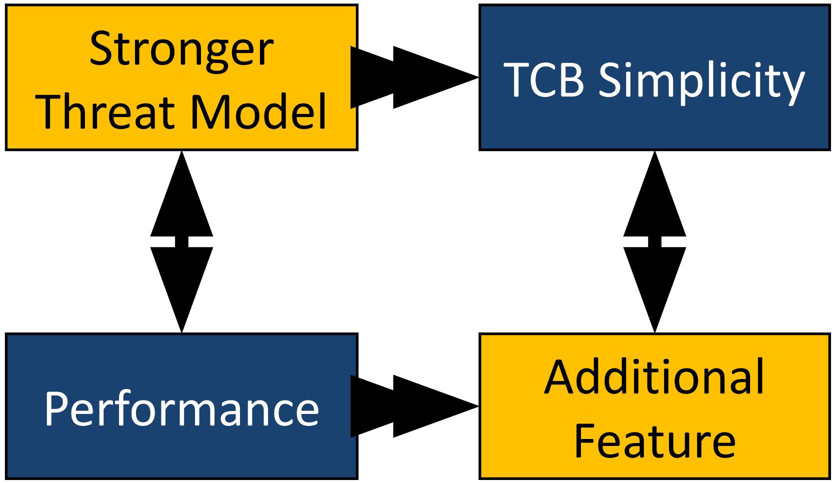
TEE as Software-Hardware Contract



ARM TrustZone intel SGX/TDX AMD SEV

Why Do We Need a Flexible Design?

- Trade-offs in security, functionality, and performance
- A reasonable threat model depends on
  - Different platforms (e.g., mobile vs. desktop)
  - Different applications (e.g., gemm vs. AES)
  - Different trust model (e.g., client vs. server)
- TEE requires a different set of features
  - Resource usage (e.g., memory, I/O)
  - Various constraints (e.g., power, latency)



Evaluation

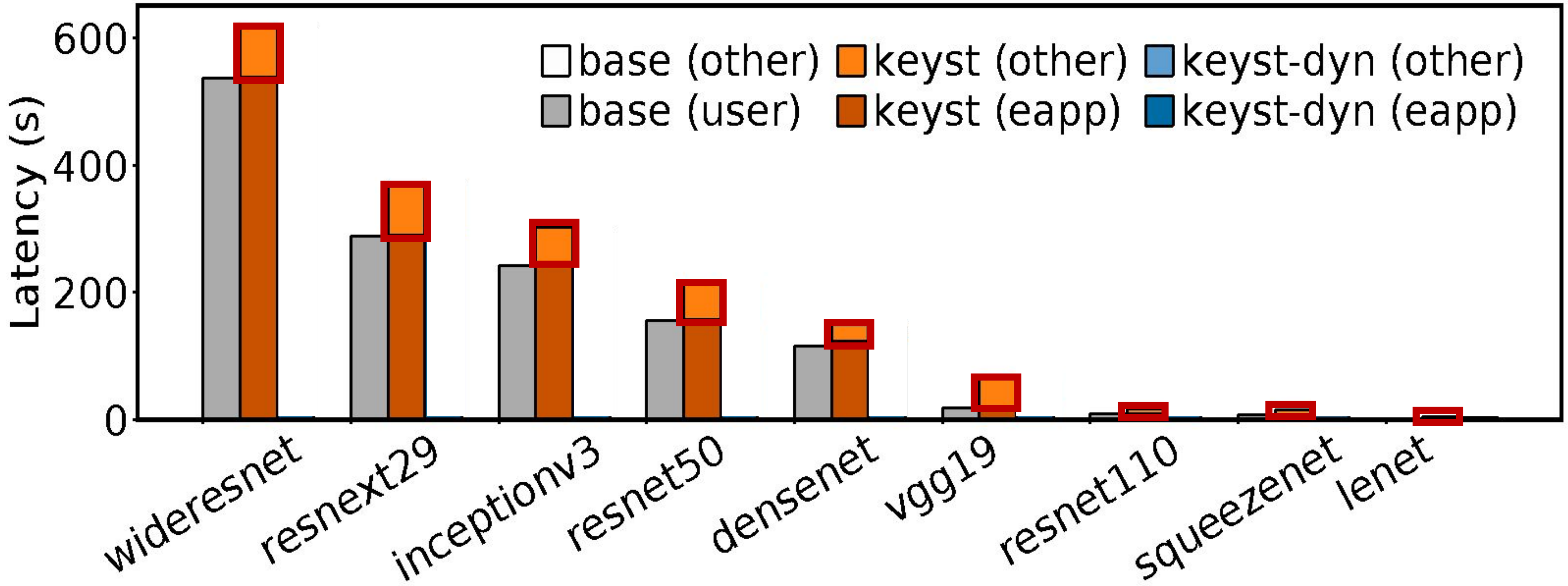
- Security Analysis
  - Keystone enclave defends various adversary models
- Modularity Analysis
  - Keystone supports fine-grained and modular configuration
- Trusted Computing Base Analysis
  - Various of real-world applications with less than thousands of LoC
- Performance Analysis
  - Security Monitor Overhead
  - Runtime Overhead
  - Cost of Memory Protection Mechanisms

Evaluation

- Security Analysis
  - Keystone enclave defends various adversary models
- Modularity Analysis
  - Keystone … ion
- Trusted Computing Base Analysis
  - Various of real-world applications with less than thousands of LoC
- Performance Analysis
  - Security Monitor Overhead
  - Runtime Overhead
  - Cost of Memory Protection Mechanisms

Please check our paper!

Runtime Overhead: Memory Management



- Torch benchmark
  - Unmodified NN inference
- Initialization overhead
  - Enclave measurement (SHA3)
- Execution overhead
  - Min -3.12% (LeNet)
  - Max 7.35% (DenseNet)
- Dynamic memory resizing
  - No noticeable overhead

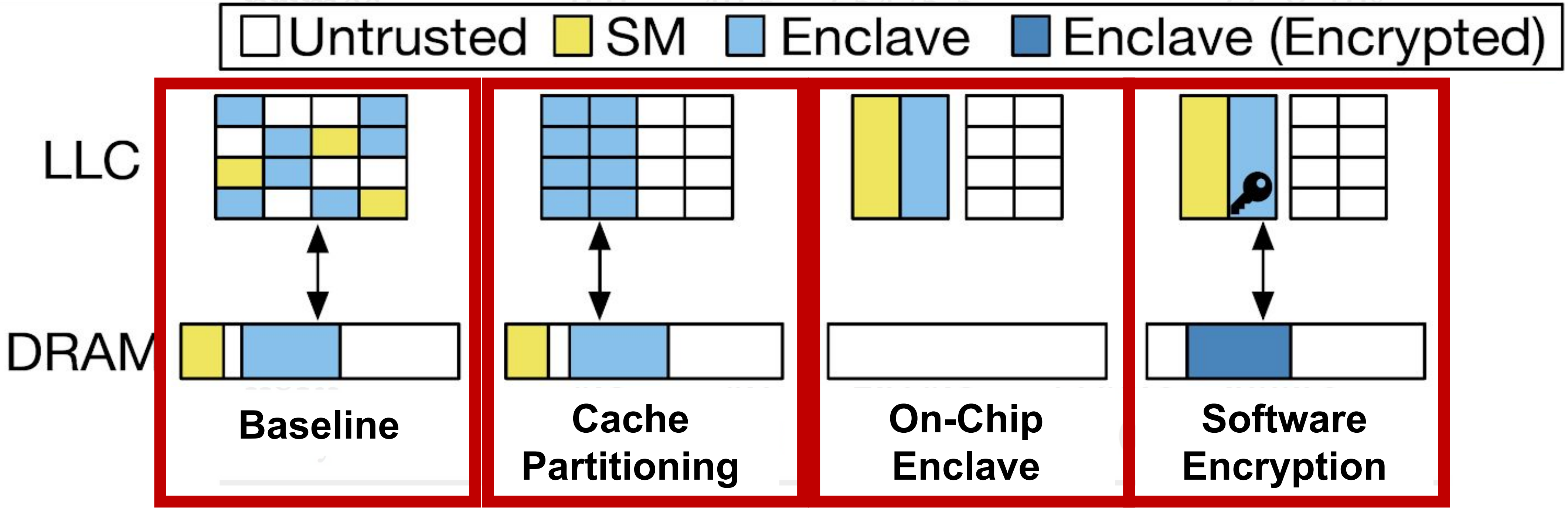
Cost of Memory Protection Mechanisms

Cache Partitioning

On-chip Execution
Self **P**aging

Software **E**ncryption

Benchmark	Overhead (%)				# of Page Faults
	∅	C	O, P	O, P, E	



Cost of Memory Protection Mechanisms

Cache Partitioning

On-chip Execution
Self **P**aging

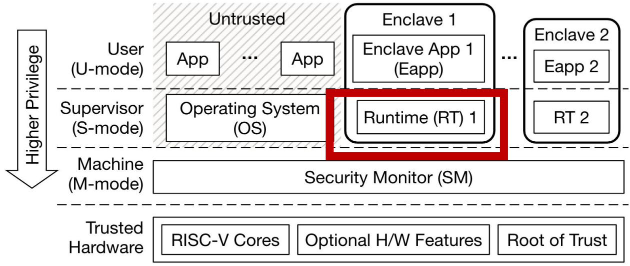
Software **E**ncryption

		Overhead (%)			# of Page
Benchmark	∅	C	O, P	O, P, E	Faults
primes	-0.9	40.5	65475.5	*	66×10^6
miniz	0.1	128.5	80.2	615.5	18341
aes	-1.1	66.3	1471.0	4552.7	59716
bigint	-0.1	1.6	0.4	12.0	168
qsort	-2.8	-1.3	12446.3	26832.3	285147
sha512	-0.1	0.3	-0.1	-0.2	0
norx	0.1	0.9	2590.1	7966.4	58834
dhrystone	-0.2	0.3	-0.2	0.2	0

Conclusion

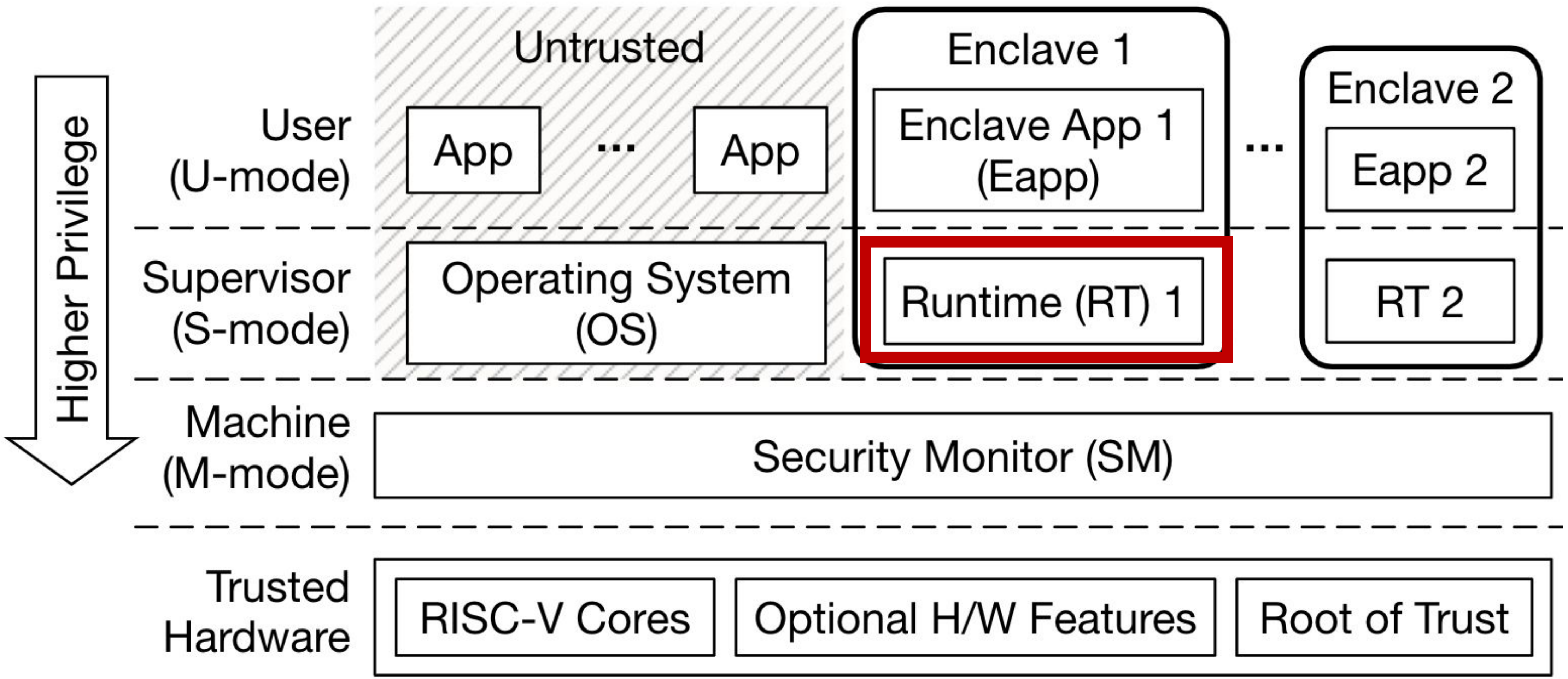
- Introduced Keystone, a *customizable* framework for TEEs
- Modular design with fine-grained customizability
- Useful for building TEEs for different threat models, functionality, and performance requirements
- Keystone is fully open-source under BSD 3-Clause

What does Keystone Runtime Do?

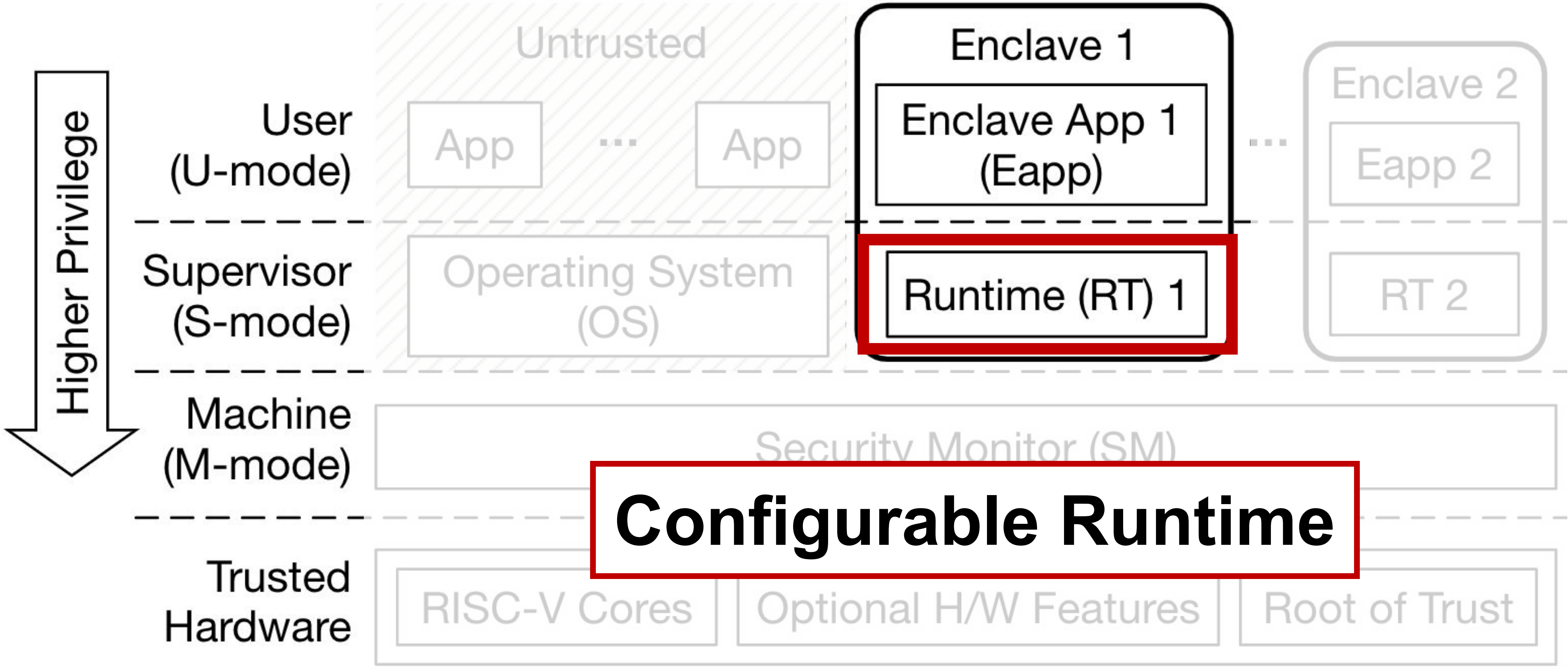


- A kernel-privileged trusted component for each enclave
  - Separation of security & functionality
- Flexible layer of abstraction
  - Minimal interface & functionality for small TCB (<4,000 LoC)
  - Fully featured, formally verified kernel (i.e., seL4)
- Fine-grained customizability for enclaves
  - Memory management: free memory, self-paging, memory encryption
  - Functionality: libraries (e.g., libc, musl-libc) and system calls

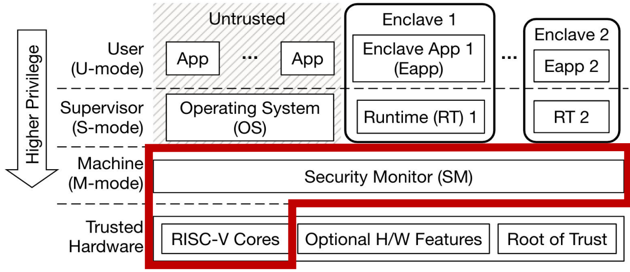
What does Keystone Runtime Do?

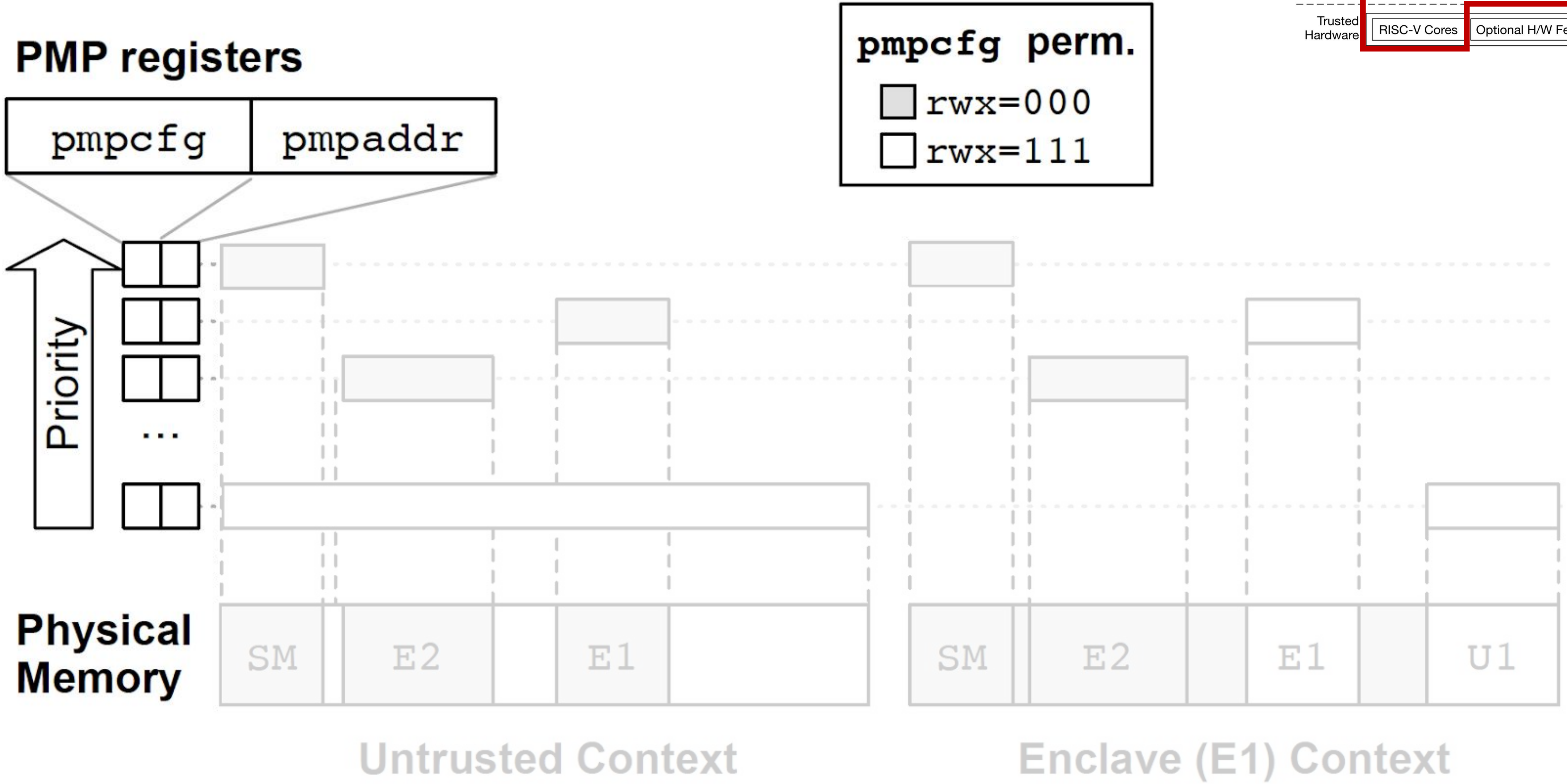


Keystone Architecture and Trust Model

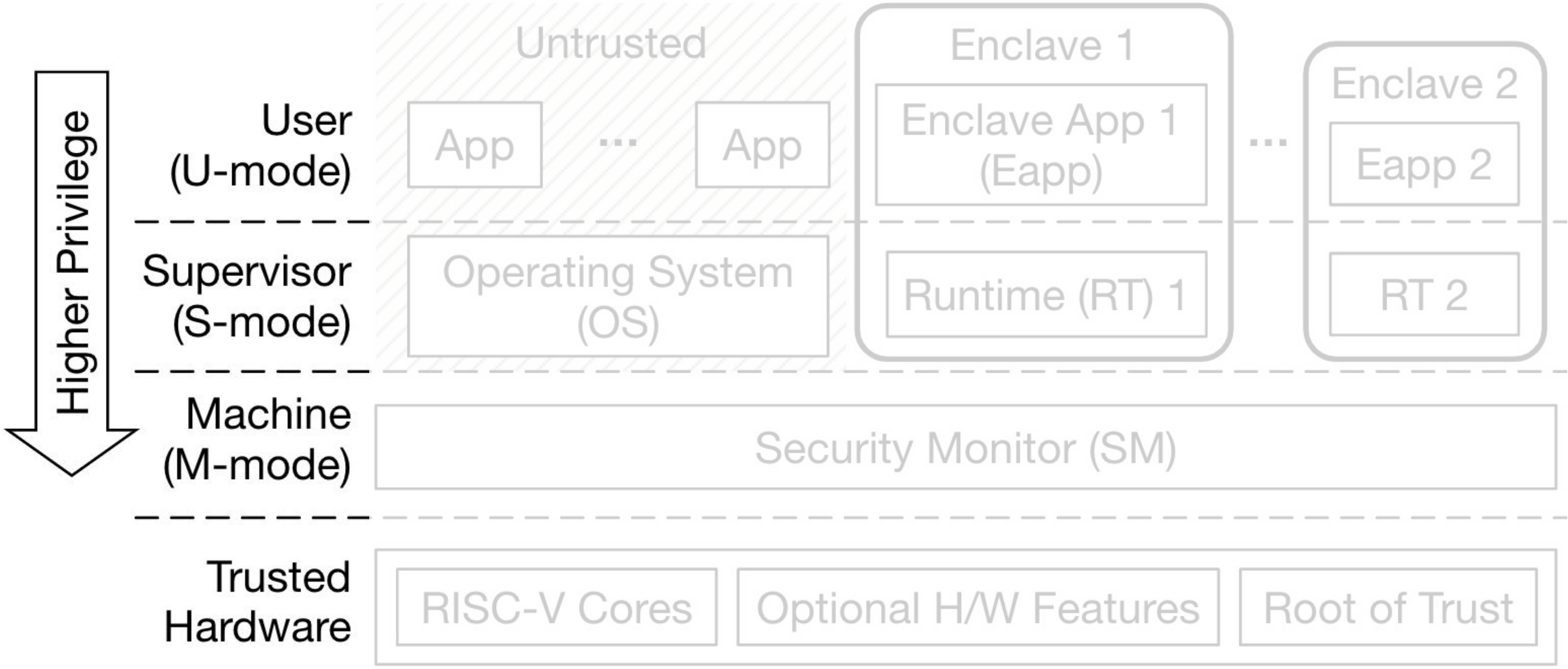


Memory Isolation via RISC-V PMP

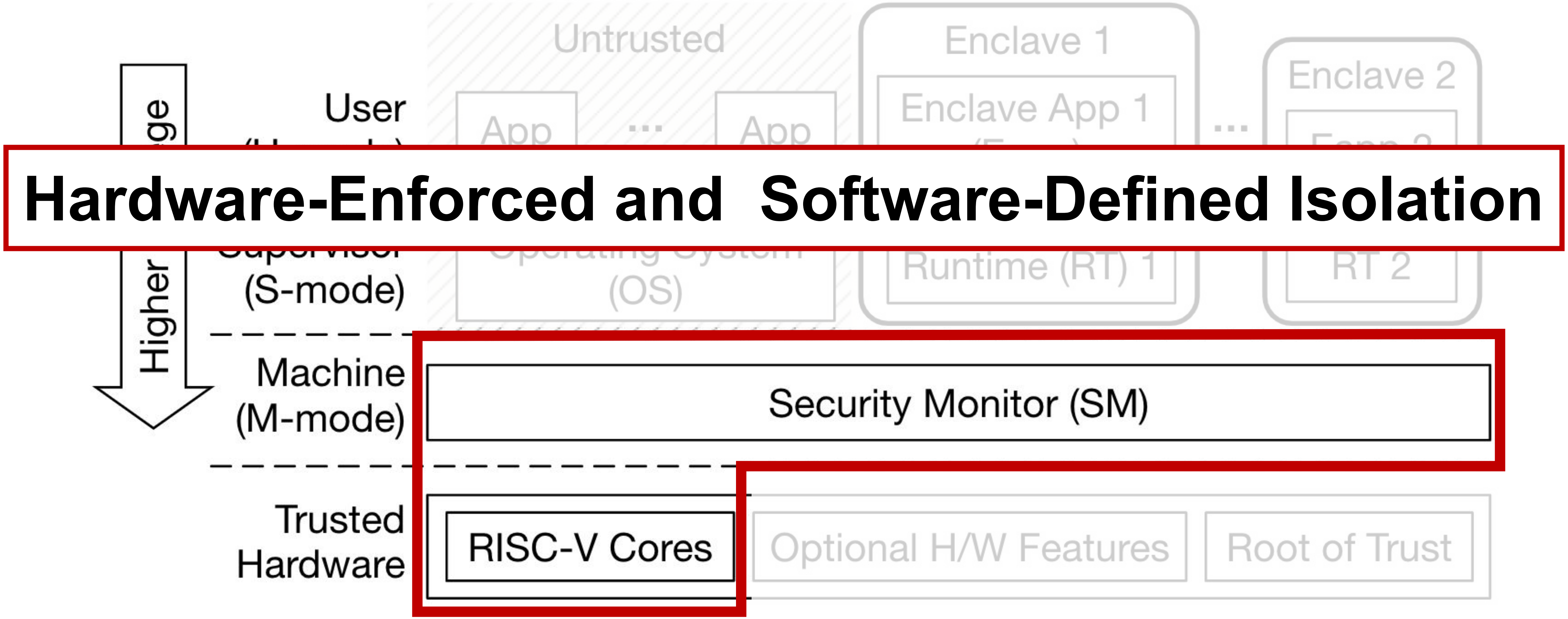




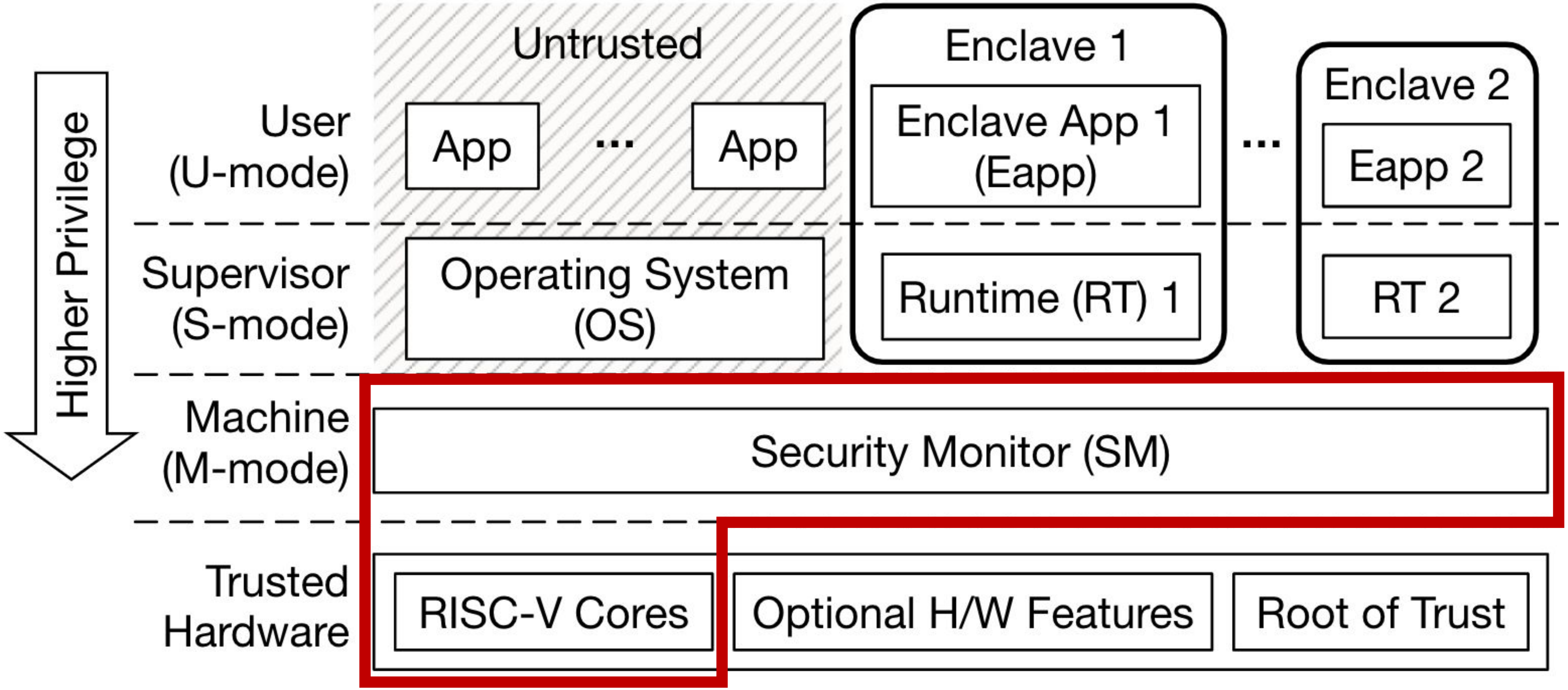
Keystone Architecture and Trust Model



Keystone Architecture and Trust Model

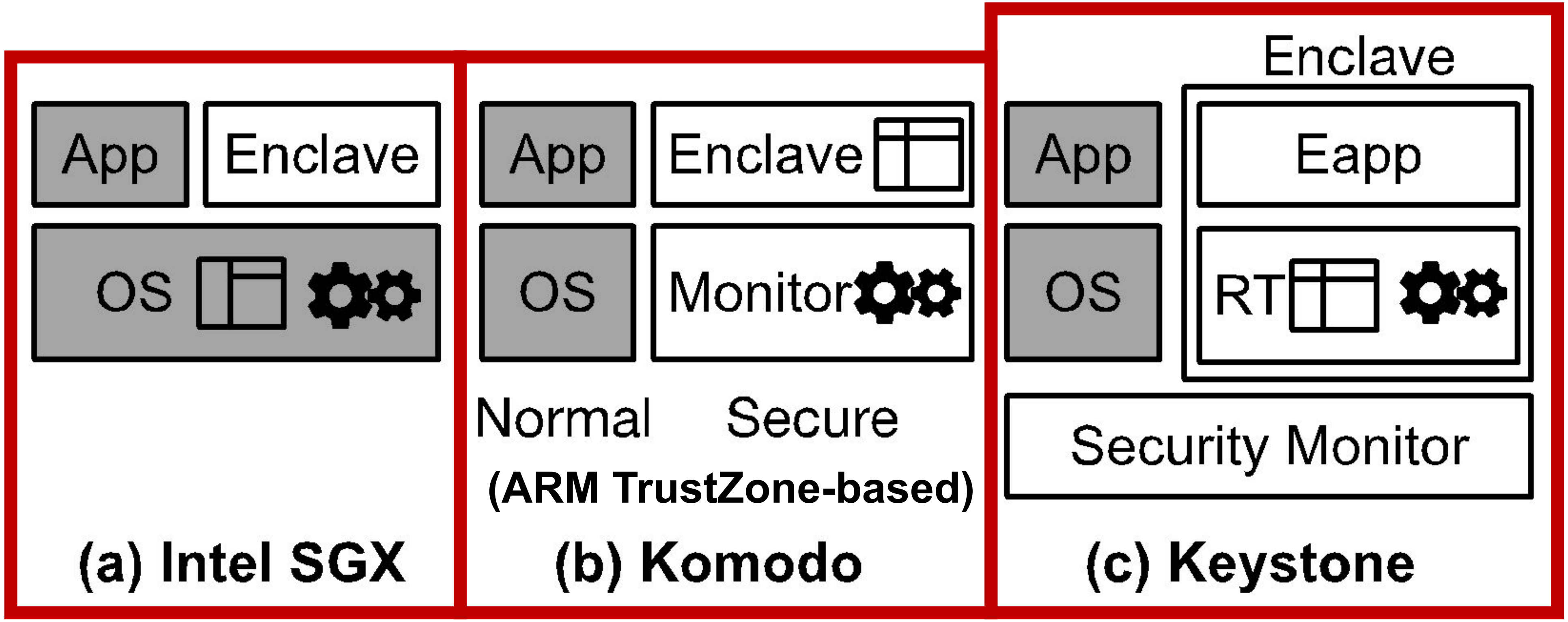


Memory Isolation via RISC-V PMP



Memory Management in Keystone

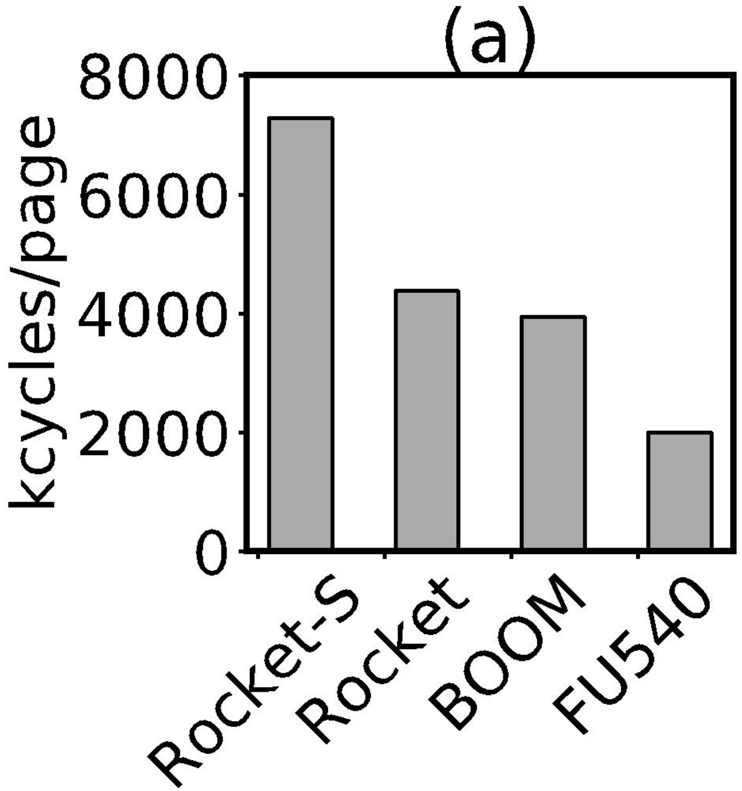
(1) Does the host share virtual addresses with the enclave?
(2) Who owns the memory management unit (MMU)?

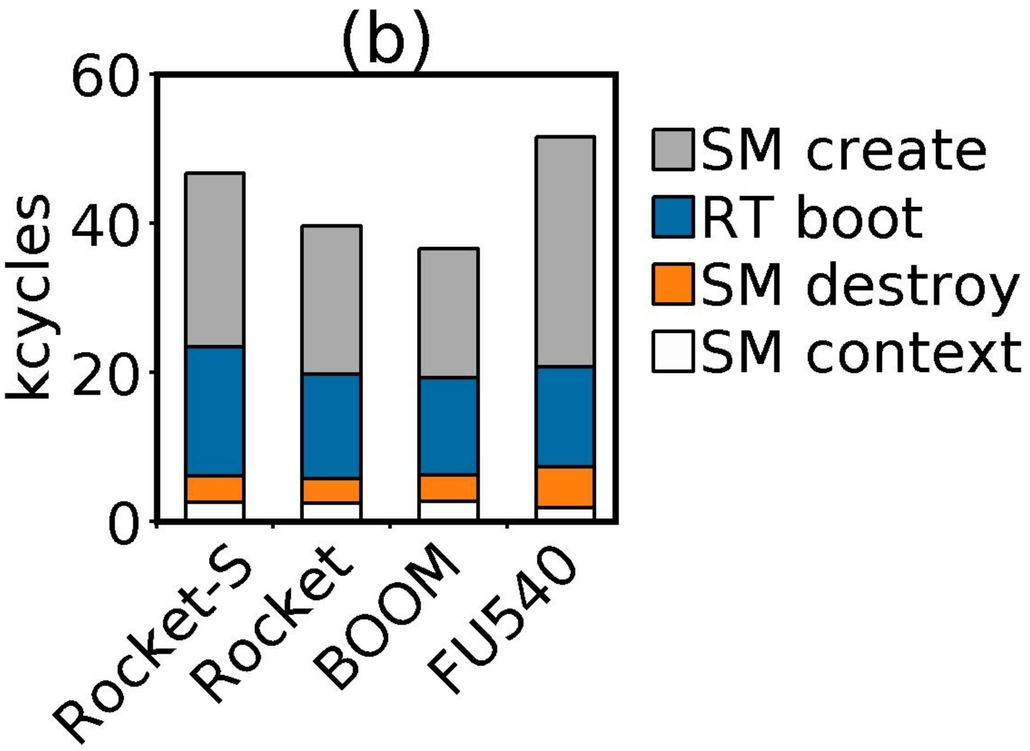


- ❑ Flexible resource management (e.g., dynamic memory resizing)
- ❑ Flexible memory protection mechanisms

Security Monitor Overhead

Enclave Measurement | **Security Monitor Overhead**





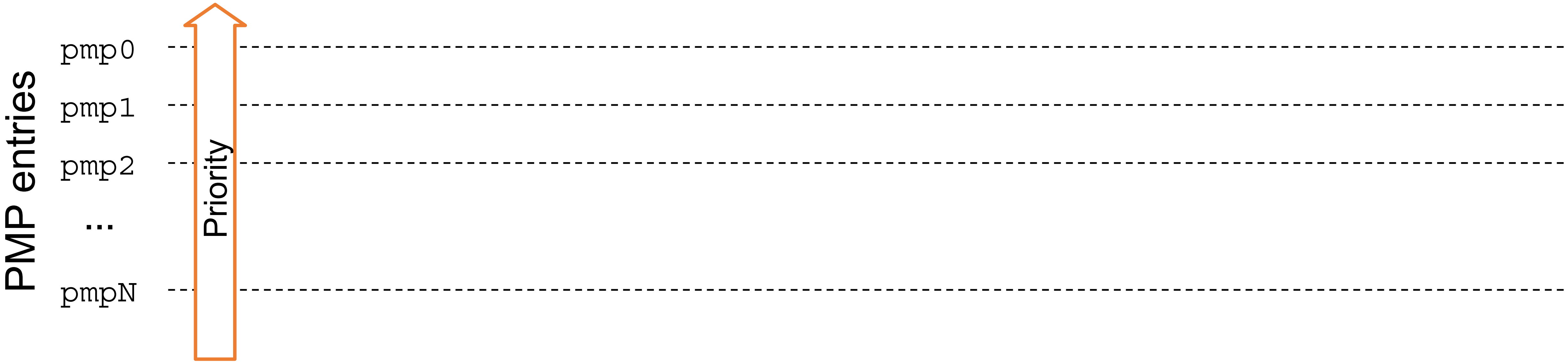
- ❑ Long initialization due to measurement (SHA3)
- ❑ No execution overhead in CPU benchmarks (CoreMark, beebs)

Contributions

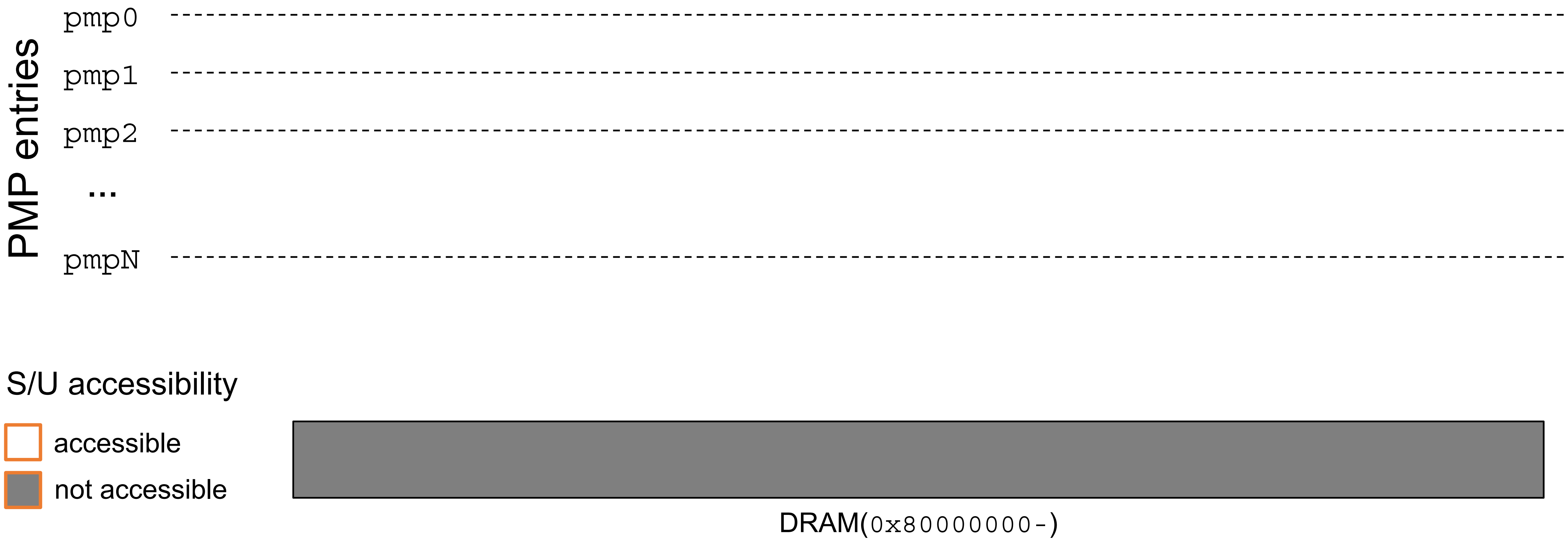
- Provide a **common base** for diverse TEEs
  - Security monitor [1,2]: programmable trusted layer *below kernel/hypervisor*
  - *Hardware-enforced* memory/context isolation and attestation
- A software framework for **customizable TEE**
  - Separation of security and functionality (e.g., resource mgmt.)
  - Fine-grained configuration of modular extensions
- Benchmarking & real-world applications
  - Overhead of various operations (e.g., enclave creation, I/O)
  - Performance trade-offs for various defenses (e.g., cache partitioning)
- An open-source, full-stack implementation for further research

[1] Costan et al., USENIX Sec'16 [2] Ferraiuolo et al., SOSP'17

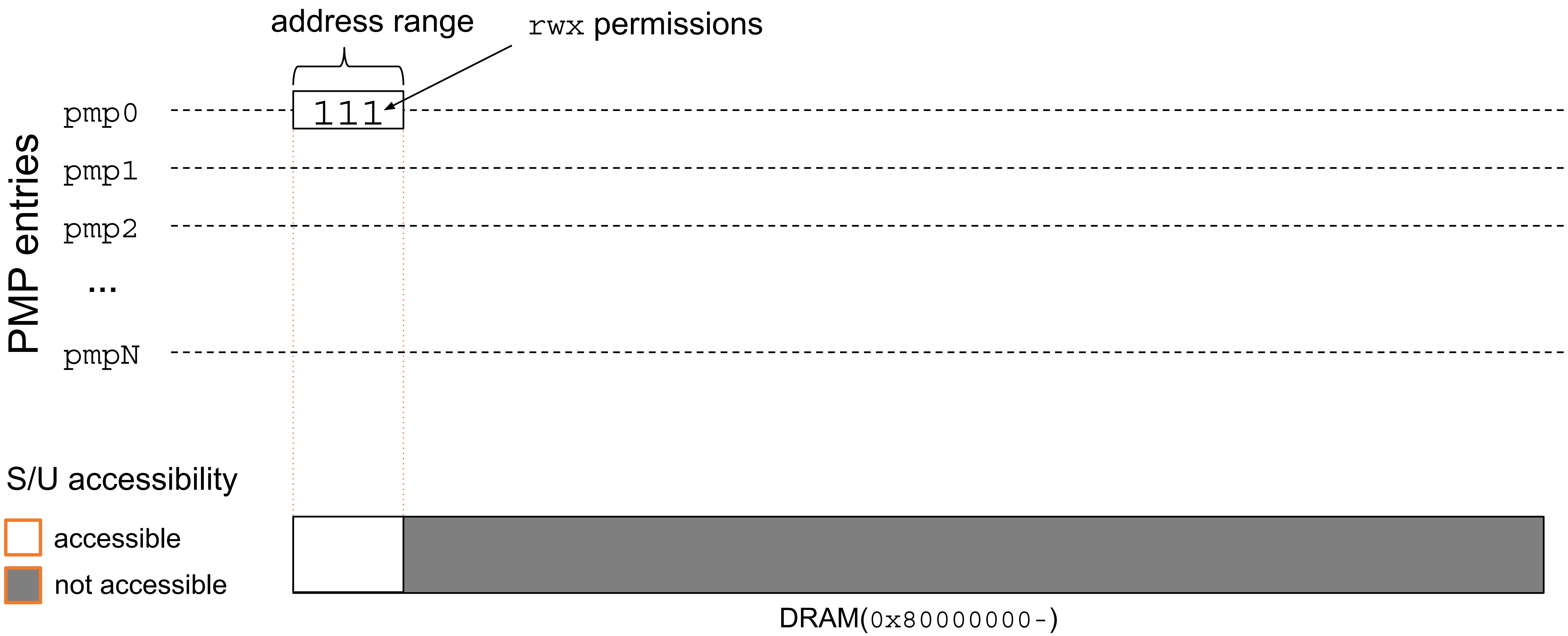
Isolation with RISC-V PMP



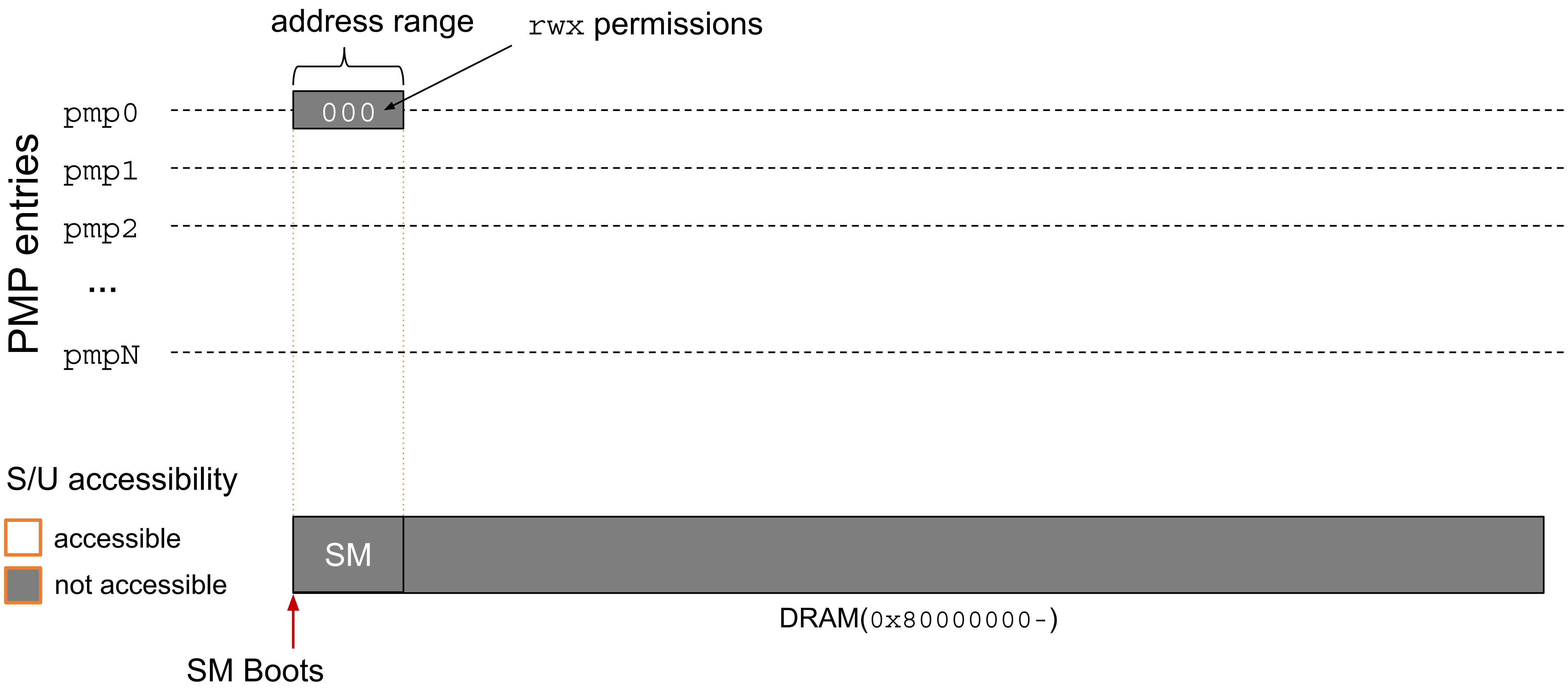
Isolation with RISC-V PMP



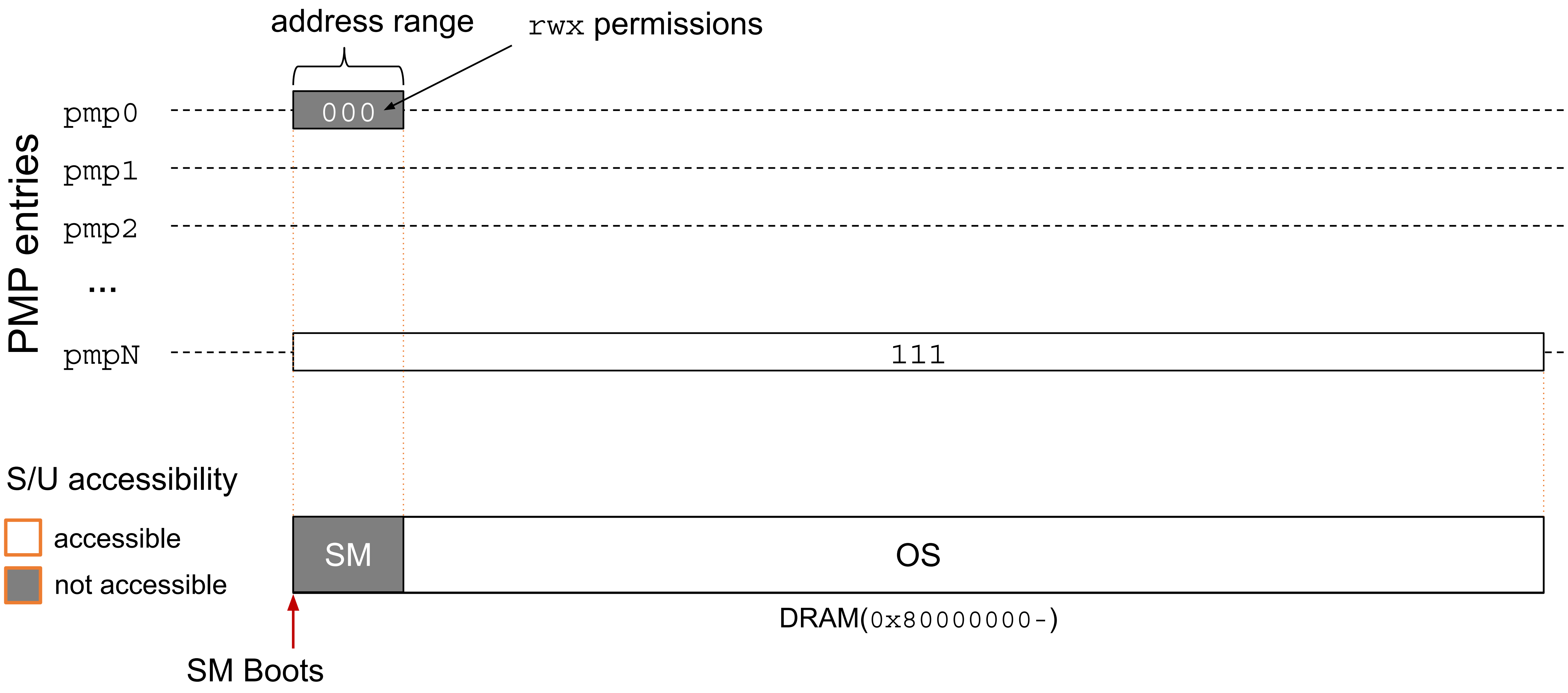
Isolation with RISC-V PMP



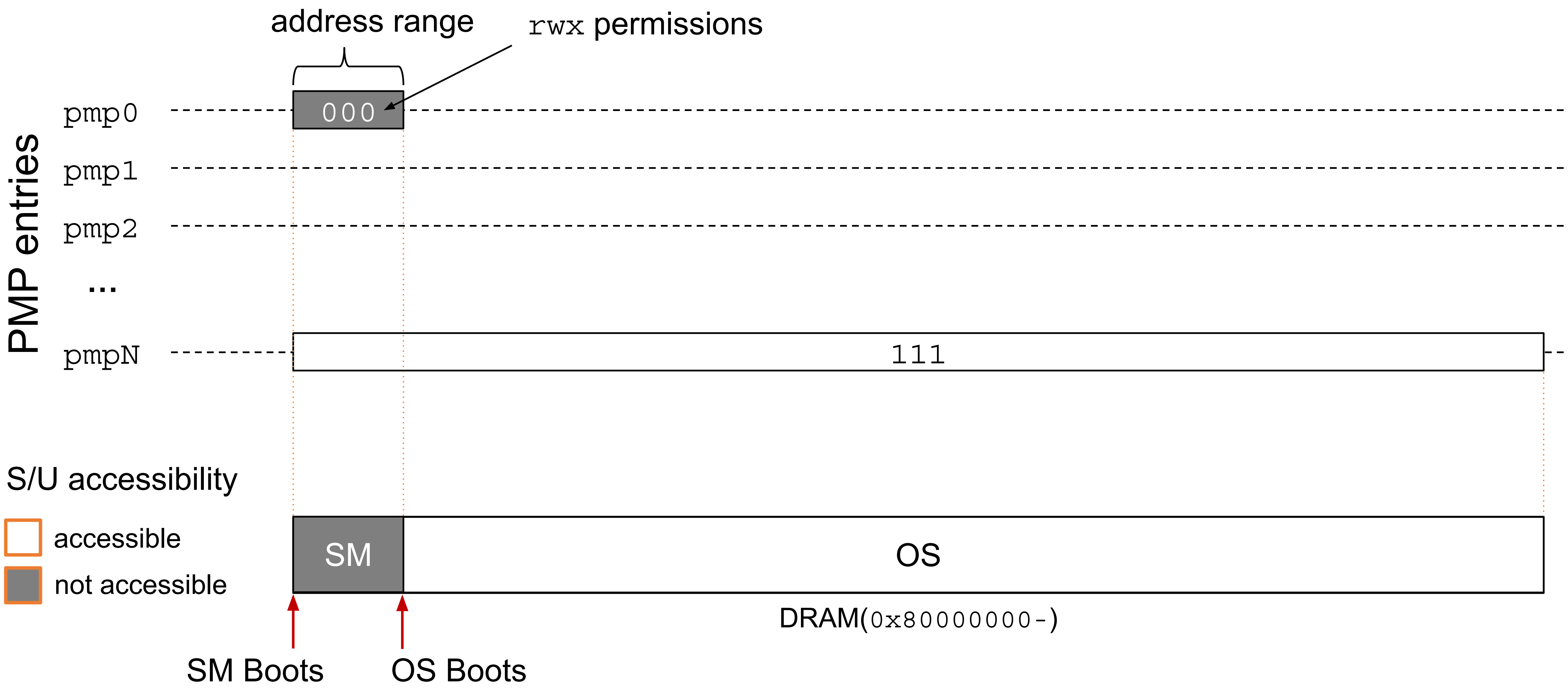
Isolation with RISC-V PMP



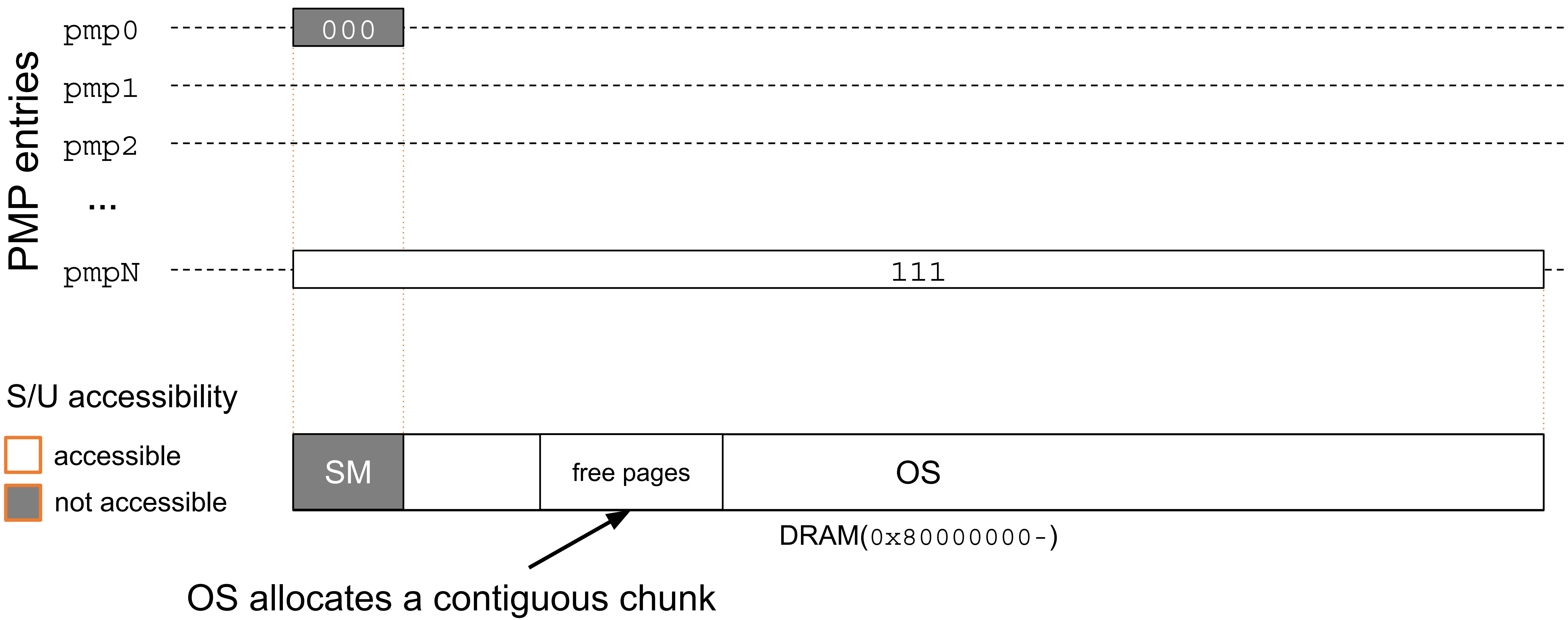
Isolation with RISC-V PMP



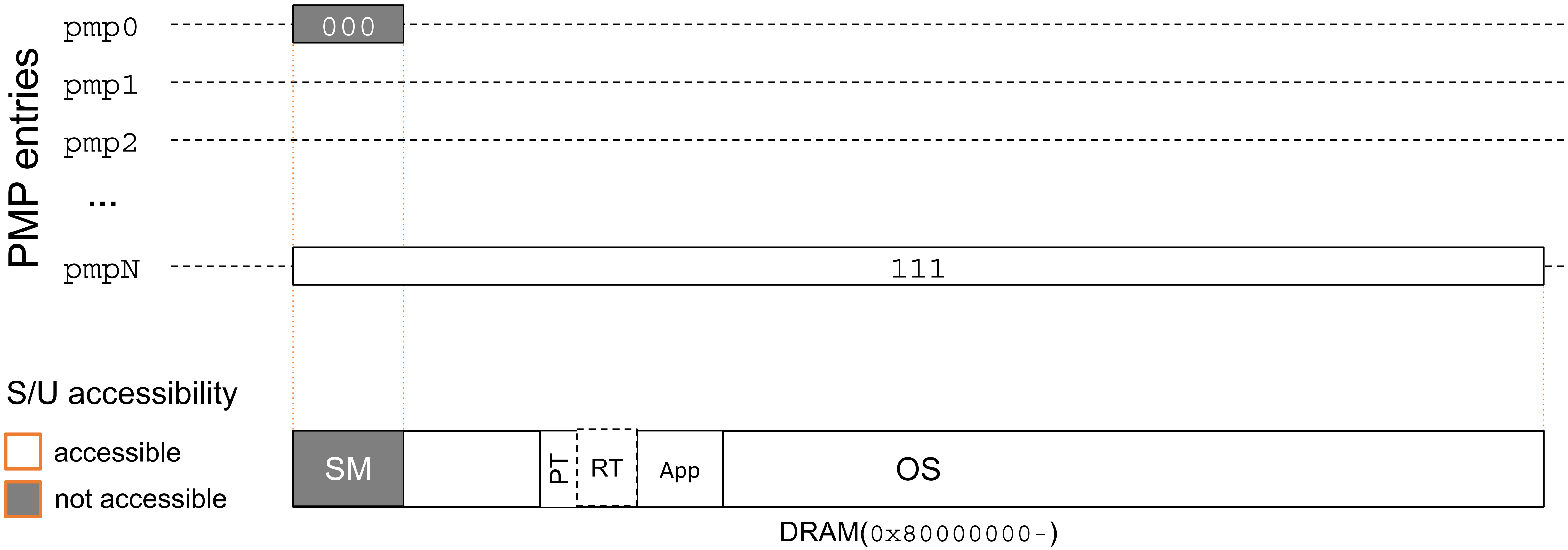
Isolation with RISC-V PMP



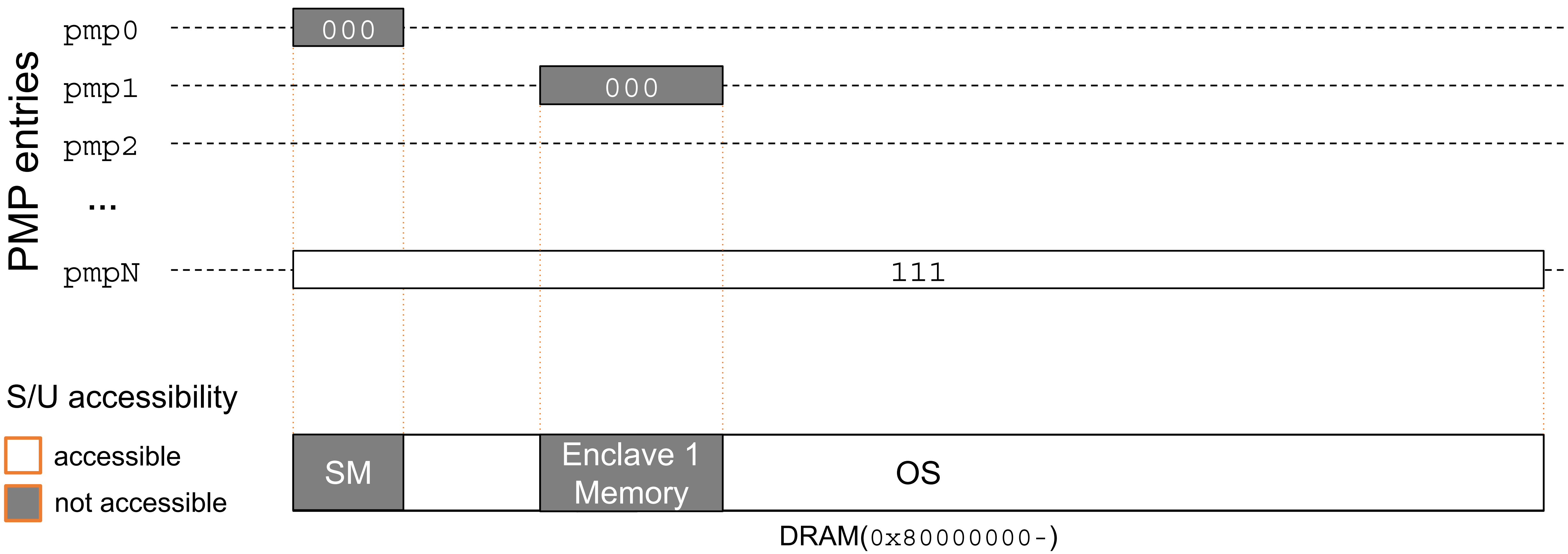
Creating an Isolated Enclave



Creating an Isolated Enclave

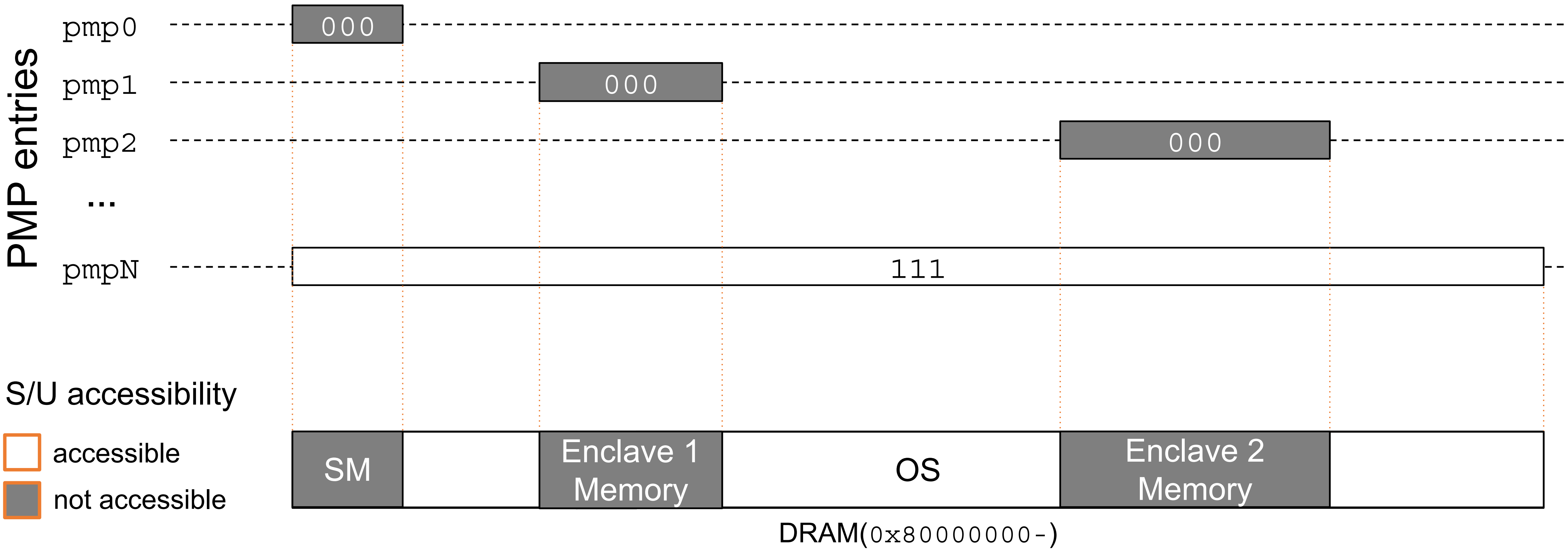


Creating an Isolated Enclave

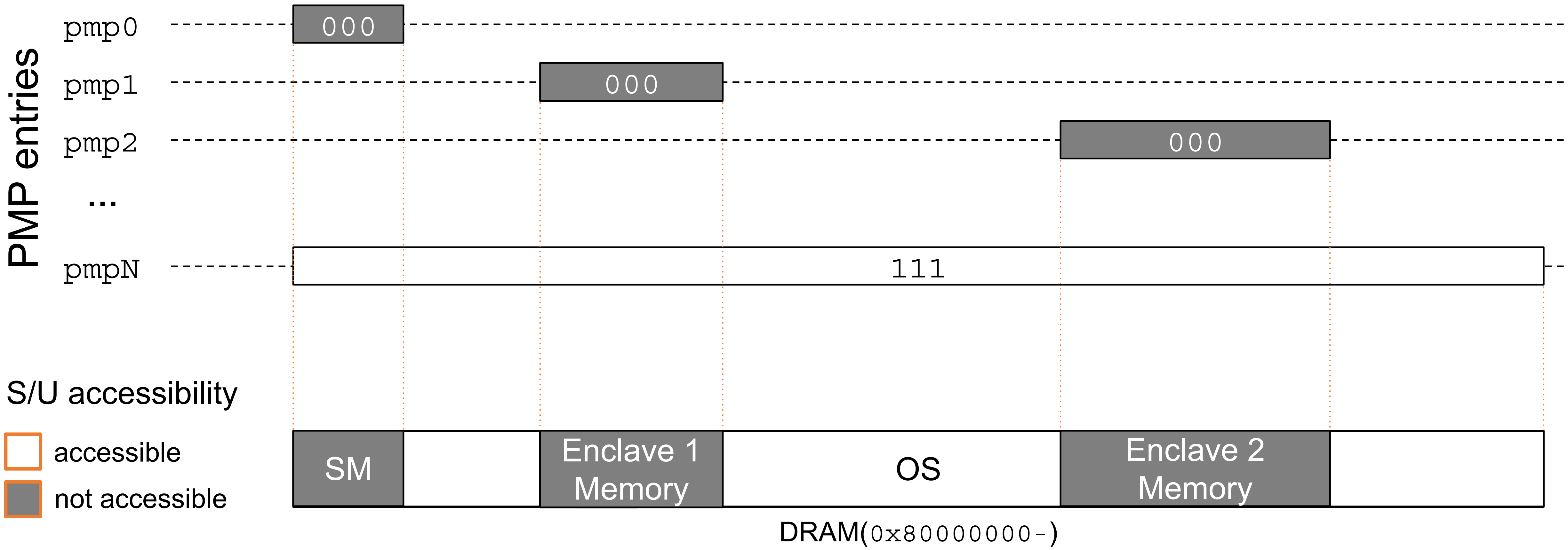


Creating an Isolated Enclave

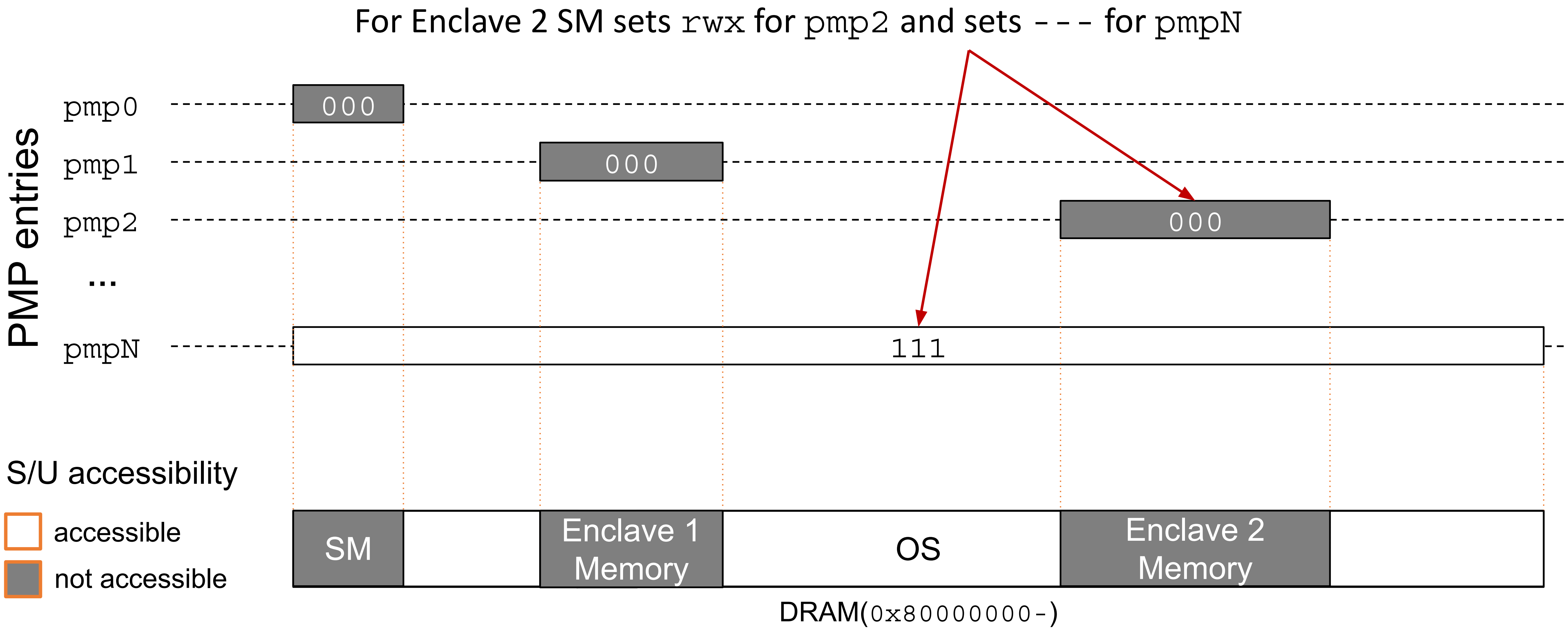
OS can ask the SM to create multiple enclaves



Executing an Enclave

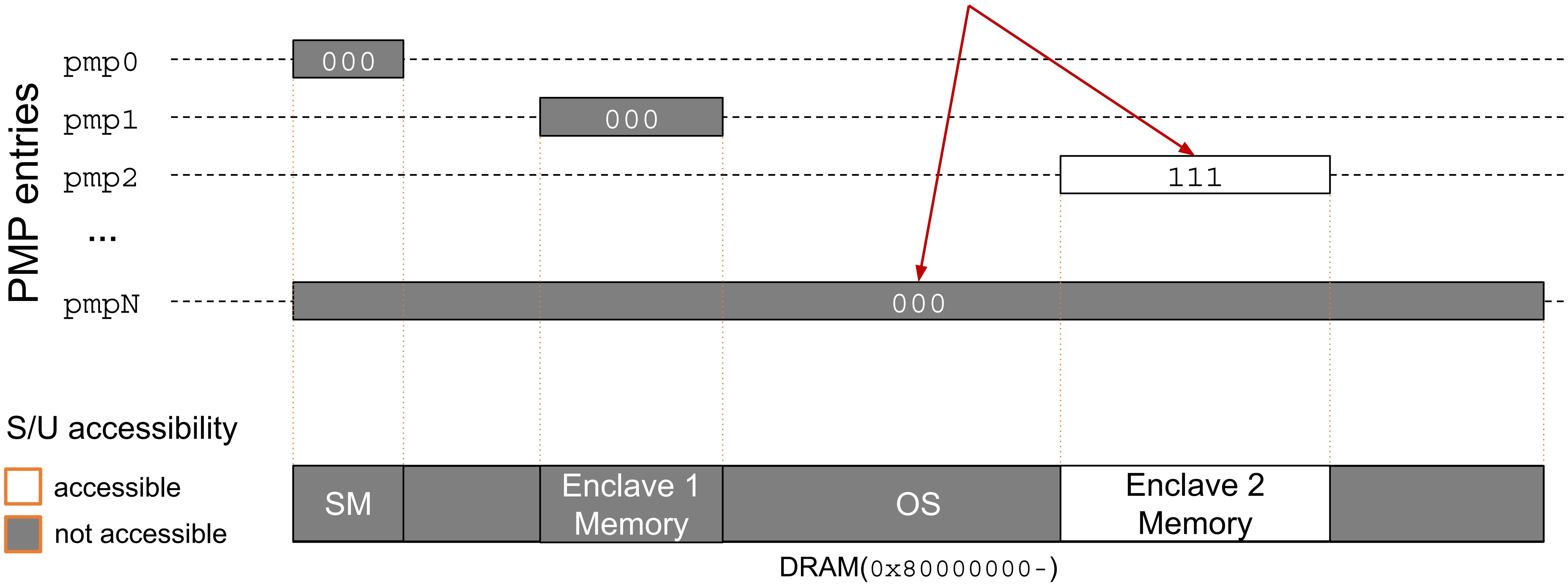


Executing an Enclave



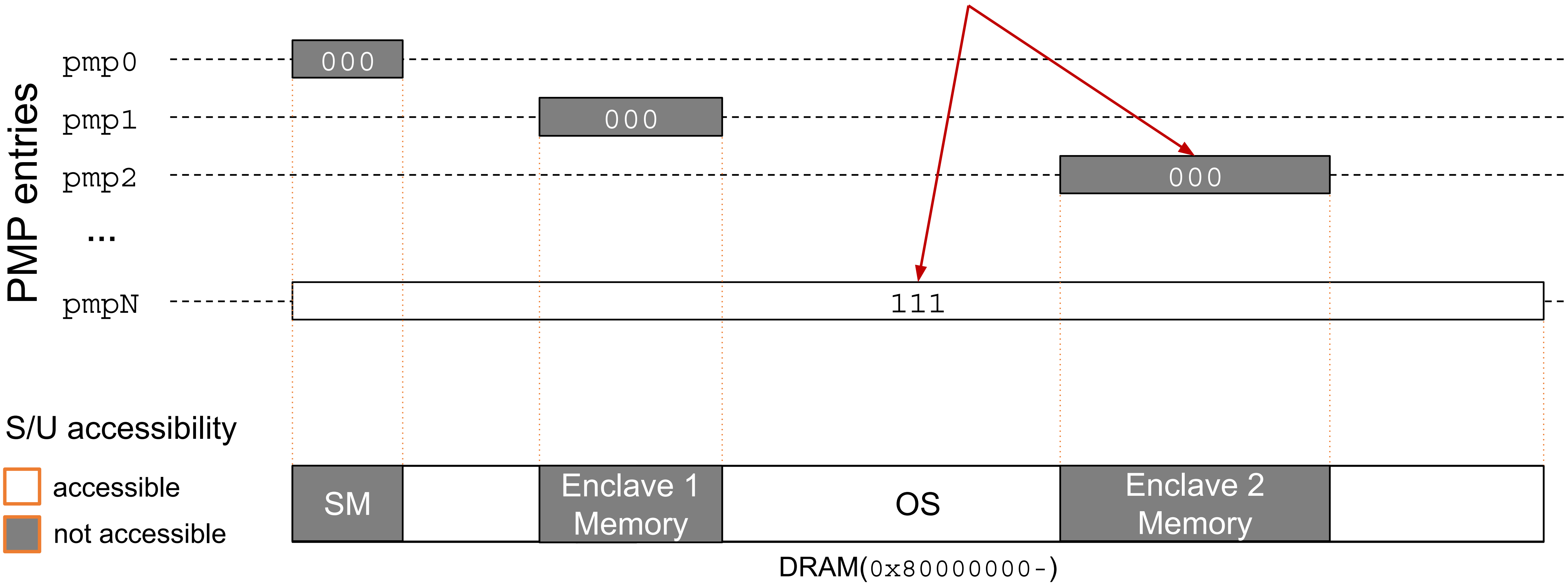
Executing an Enclave

For Enclave 2 SM sets `rwx` for `pmp2` and sets `---` for `pmpN`

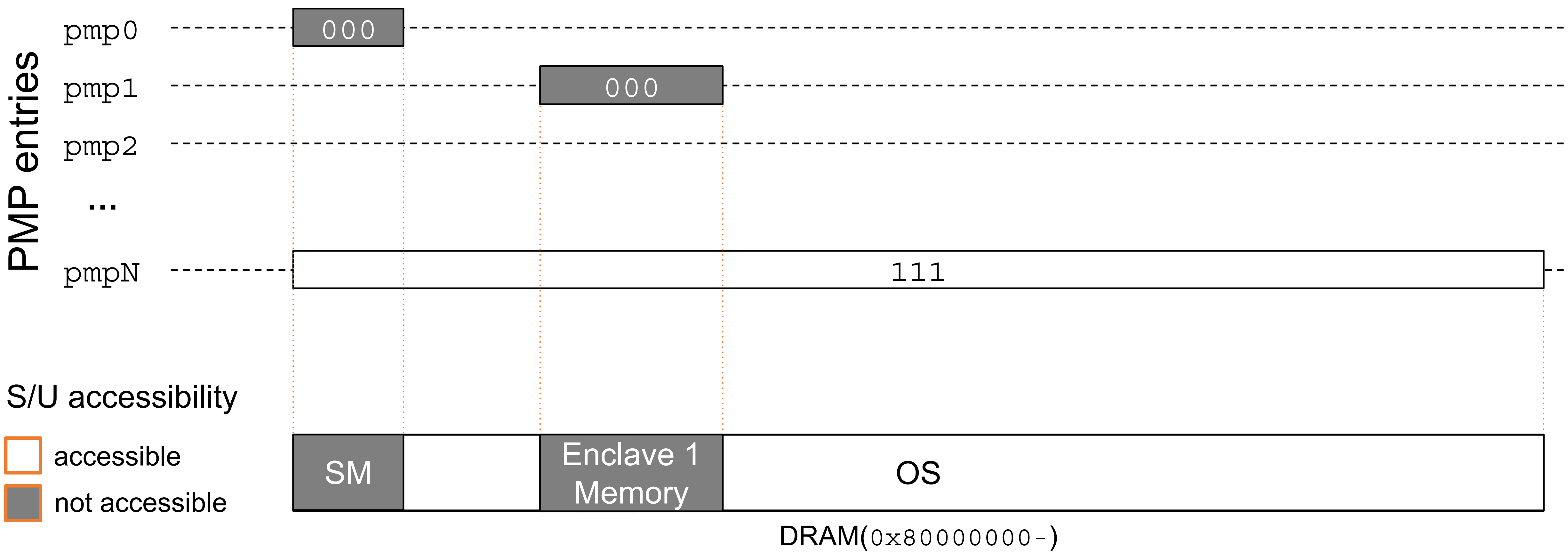


Exit an Enclave

Switch back to defaults for the OS



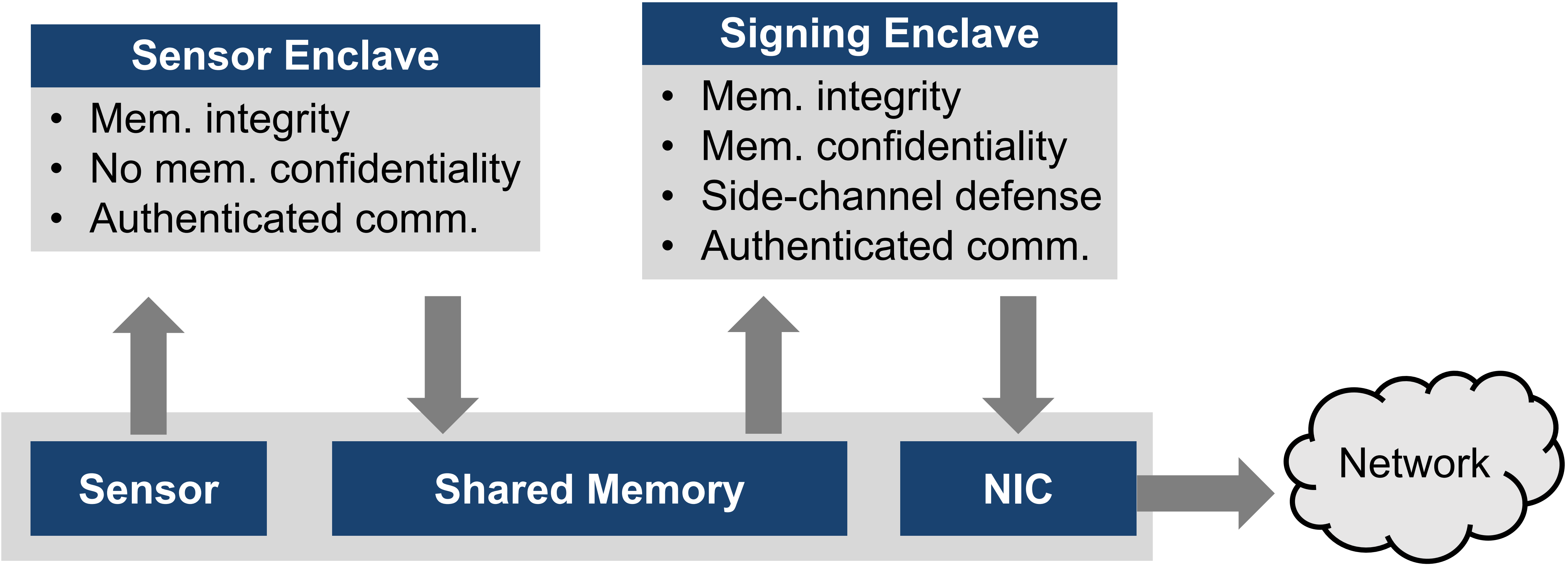
Destroying an Enclave



Keystone

Varying Threat Models in the Same Platform

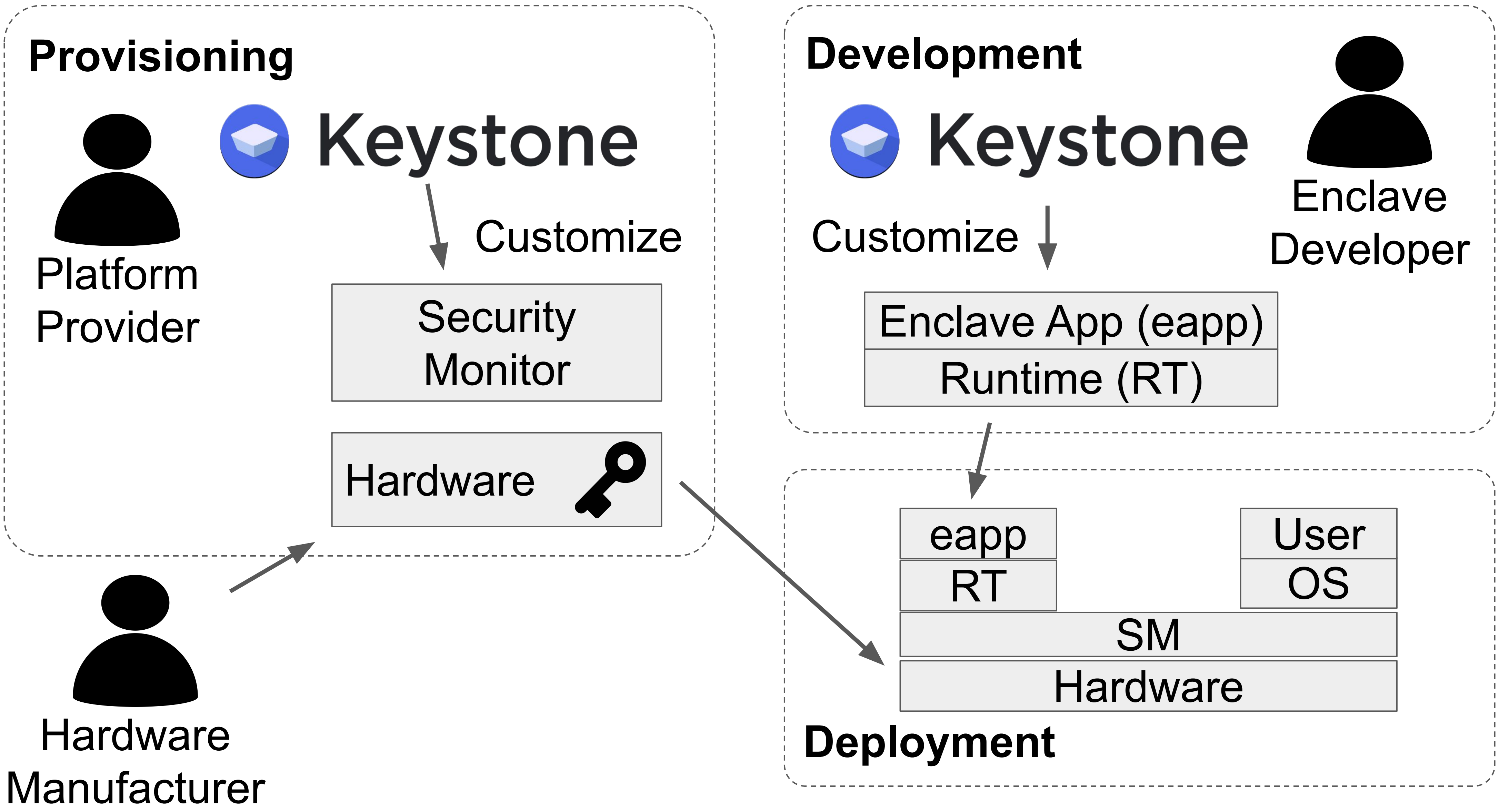
e.g., IoT Sensor Platform



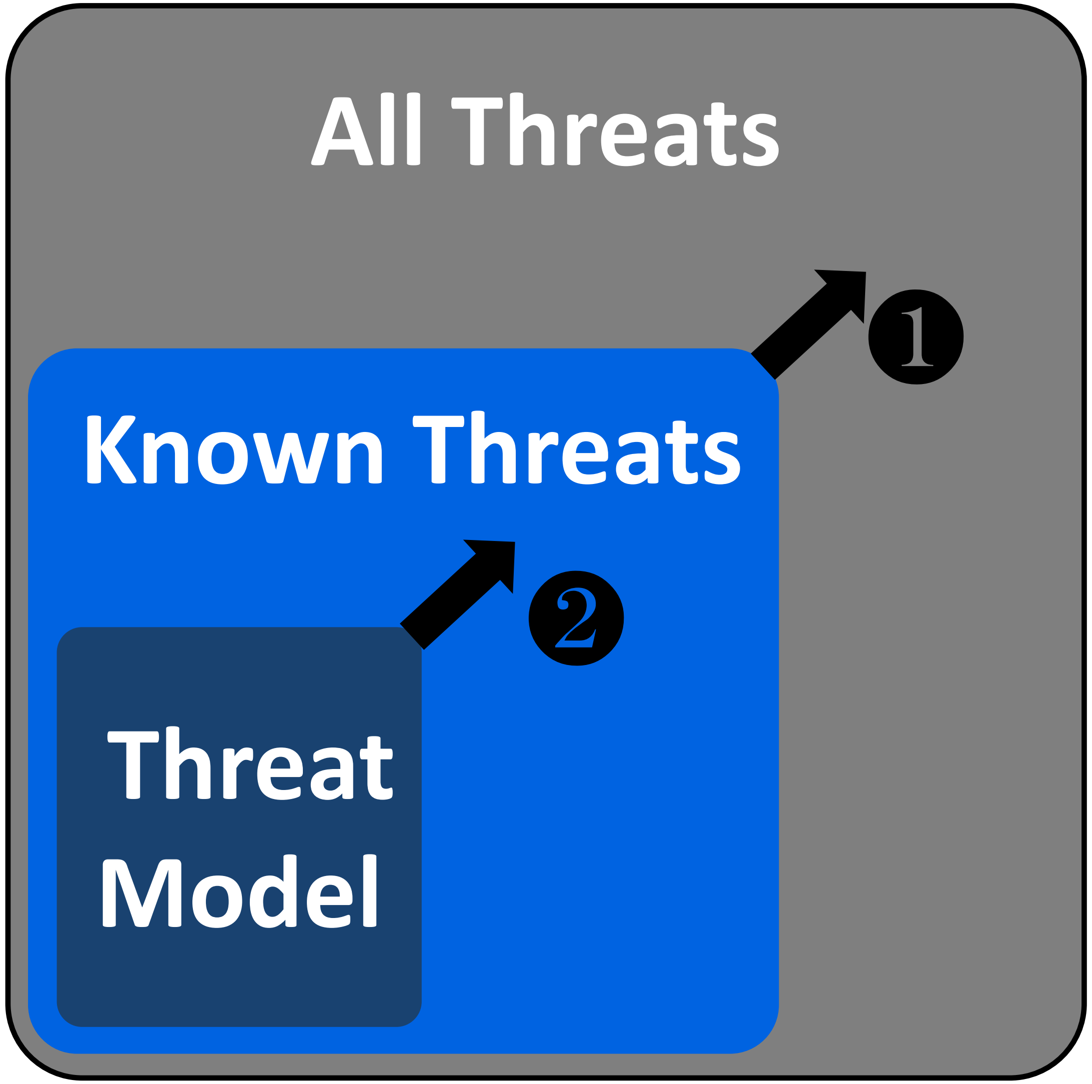
Why TEEs are Promising for Security?

- Increasing needs for secure computation
  - Big data and machine learning (e.g., coopetitive learning)
  - Trends in IoT, mobile, and cloud computing
  - Requirements in new applications (e.g., blockchain, contact tracing)
- Security in system software is getting harder
  - Increasing attack surface in the system software (e.g., Linux kernel)
  - SW boundaries are often broken by SW/HW attackers
- TEE as a cornerstone for secure computation
  - Minimizing trusted computing base (TCB)
  - Efficient HW-enforced isolation and authentication

How TEE Customization Work?



TEE Threat Model Evolves Over Time



Unknown threat is newly discovered (e.g., vulnerability in speculative execution)

Known threats become substantial (e.g., cache side-channel attacks)

Inflexible Design and Implementation

- TEEs in commercial hardware: Intel SGX, ARM TZ, AMD SEV
- Designs and threat models depend too much on their business
  - Intel SGX – small server/desktop apps (e.g., DRM, cryptography, etc)
  - ARM TZ – vendor-provisioned mobile apps (e.g., fingerprint, ledger)
  - AMD SEV – full VM isolation only (targeting cloud market?)
- Implemented on closed-source hardware
  - Slow iteration dictated by a company; researchers can’t step forward
  - Any additional features/defenses need significant workaround

Vendor-Locked Threat Models

ISA/Arch	System	SW Attacks	HW Attacks	Side Channel	Controlled Channel
Intel	SGX	●	●	○	○
	Haven	●	●	○	○
	Graphene	●	●	○	○
ARM	TrustZone	●	○	○	○
	Komodo	●	●	○	●
	OPTEE	●	●	○	○
AMD	SEV	◉	◉	○	○
	SEV-ES	●	●	○	○
RISC-V	Sanctum	●	○	●	●
	Keystone	●	●	●	●

Runtime Overhead: Untrusted I/O