Confidential Computing on RISC-V

Nick Kossifidis

Principal Research Engineer @ FORTH

Chair of RISC-V Runtime Integrity SIG

European Commission
Horizon Europe
2021-2027

Quick intro to confidential computing

Trusted Execution Environment

- An execution context that provides confidentiality & integrity guarantees.
  - Outsiders can't access data or code at runtime.
  - Outsiders can't interfere with its control flow at runtime.
  - Outsiders
    - any other untrusted entity (e.g. other processes, devices, the OS, other systems).
    - any other trusted entities that don't belong in the same security / trust domain.
  - Security domain
    - a set of resources managed with a common security policy (e.g. a form of authentication).

Trusted Execution Environment

- **But not availability !**
  - A TEE may be instantiated through an untrusted environment, e.g. an OS that needs to allocate resources for the TEE, load a binary inside etc.

Trusted Computing Base

- The set of mechanisms (hardware and software) required for enforcing security policies (including TEE guarantees).
- TCB usually includes
  - The CPU
    - Bootrom
    - Firmware
    - Secure monitor
  - Other devices
    - Firmware
- Complex OSes, Hypervisors, etc can't be part of TCB since they are very hard to audit and protect.

Trusting the TCB (an example)

- CPU
  - HW Root-of-Trust (RoT)
- Bootrom
  - Signed measurement of itself
- Firmware
  - Measurement signed by Bootrom
- Secure Monitor
  - Measurement signed by firmware

Device Identifier Composition Engine

- A standard way for creating a trust chain
- Each part of the chain (aka Compound Device Identifier) depends on the previous one, and the first one depends on the Unique Device Secret (UDS).
  - $CDI_0 = KDF(UDS_{len}, UDS \,||\, Hash(Measurement(TCB_0)))$
  - $CDI_n = KDF(CDI_{len}, CDI_{n-1} \,||\, Hash(Measurement(TCB_n)))$
- Each CDI never leaves the TCB

https://trustedcomputinggroup.org/work-groups/dice-architectures/

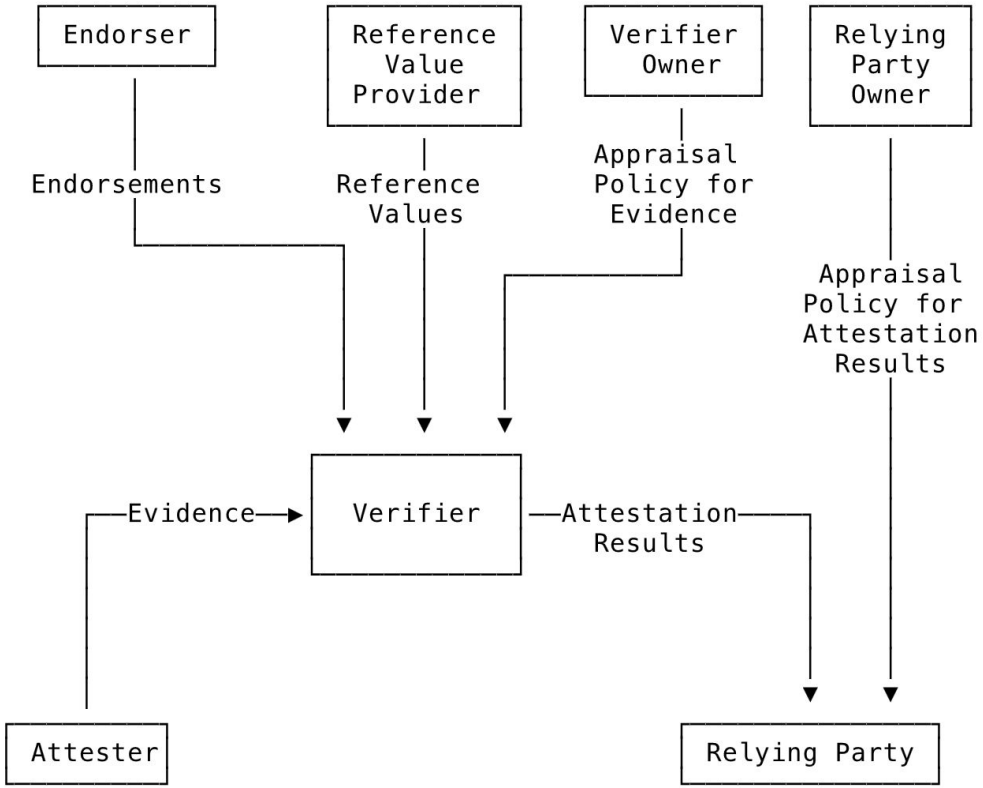
Attestation evidence

- Each TCB layer creates an attestation keypair
  - $(UDS_{priv}, UDS_{pub}) = AsymKDF(UDS)$
  - $(CDI_{priv\_n}, CDI_{pub\_n}) = AsymKDF(CDI_n)$

- For each TCB layer we get a TCB id from its public key
  - $UDS\_ID = KDF(ID_{len}, UDS_{pub})$
  - $CDI_n\_ID = KDF(ID_{len}, CDI_{pub\_n})$

- We use those to create Entity Attestation Tokens (EAT)
  - https://datatracker.ietf.org/doc/draft-ietf-rats-eat/22/

Remote attestation framework



IETF RATS (RFC9334)

Riser

Quick overview of RISC-V Privilege modes

RISC-V Privilege modes

Machine Mode

- Mandatory
- The most privileged / protected mode visible to the software (there is also Debug mode but it's only accessible / visible to hw debuggers)
- Physical memory addressing
- Physical memory protection
- Trap/Interrupt handling and delegation

User Mode

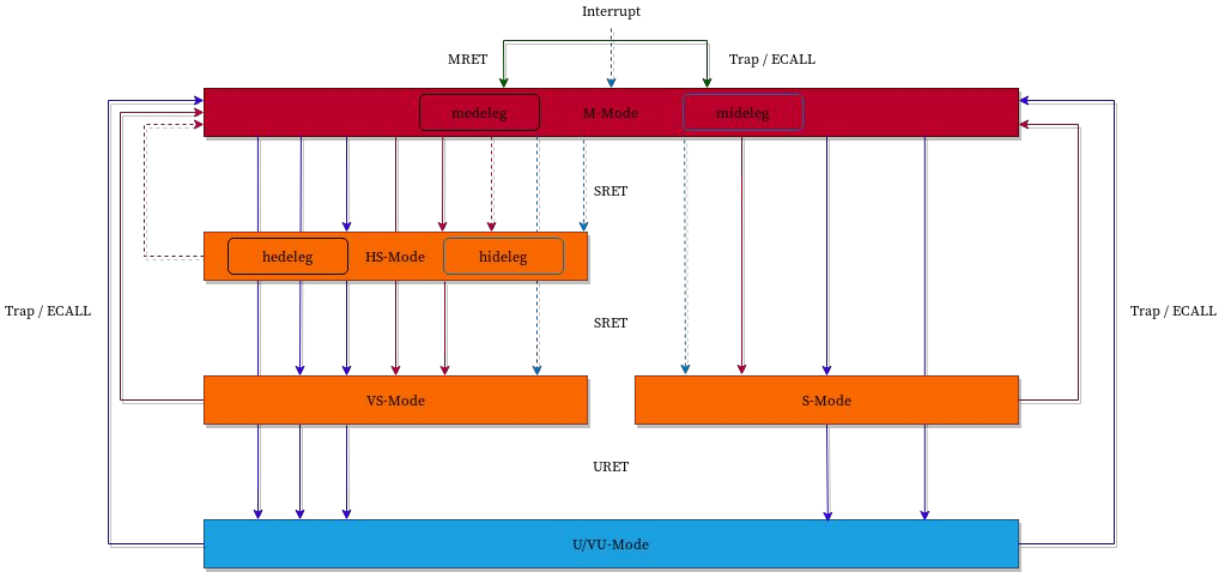
- Optional (depends on M-mode)
- The least privileged / protected mode
- Physical/virtual memory addressing Physical/virtual memory protection
- No trap/interrupt handling

Supervisor Mode

- Optional (depends on M-mode and U-mode)
- Sits between M-mode and U-mode
- Provides virtual memory addressing / protection
- Trap/interrupt handling through delegation, managed by M-mode
- May act as a hypervisor (aka HS-mode) through the use of an extra set of CSRs, also providing a second stage of translation / protection for guests (aka VS-mode instances)

The RISC-V Privileged Spec
https://github.com/riscv/riscv-isa-manual/releases

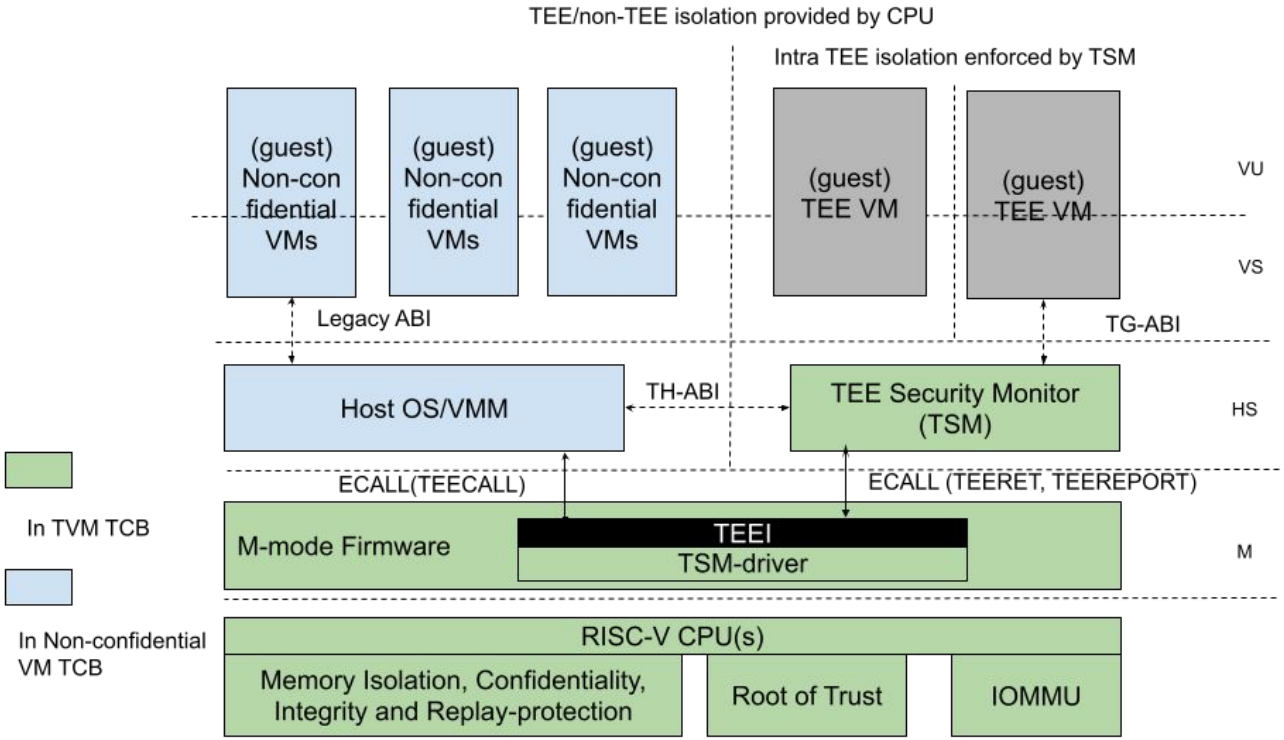
Trap and interrupt delegation



Interrupt	Exception Code	Description
1	0	*Reserved*
1	1	Supervisor software interrupt
1	2	Virtual supervisor software interrupt
1	3	Machine software interrupt
1	4	*Reserved*
1	5	Supervisor timer interrupt
1	6	Virtual supervisor timer interrupt
1	7	Machine timer interrupt
1	8	*Reserved*
1	9	Supervisor external interrupt
1	10	Virtual supervisor external interrupt
1	11	Machine external interrupt
1	12	Supervisor guest external interrupt
1	13–15	*Reserved*
1	≥16	*Designated for platform or custom use*
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode or VU-mode
0	9	Environment call from HS-mode
0	10	Environment call from VS-mode
0	11	Environment call from M-mode
0	12	Instruction page fault
0	13	Load page fault
0	14	*Reserved*
0	15	Store/AMO page fault
0	16–19	*Reserved*
0	20	Instruction guest-page fault
0	21	Load guest-page fault
0	22	Virtual instruction
0	23	Store/AMO guest-page fault
0	24–31	*Designated for custom use*
0	32–47	*Reserved*
0	48–63	*Designated for custom use*
0	≥64	*Reserved*

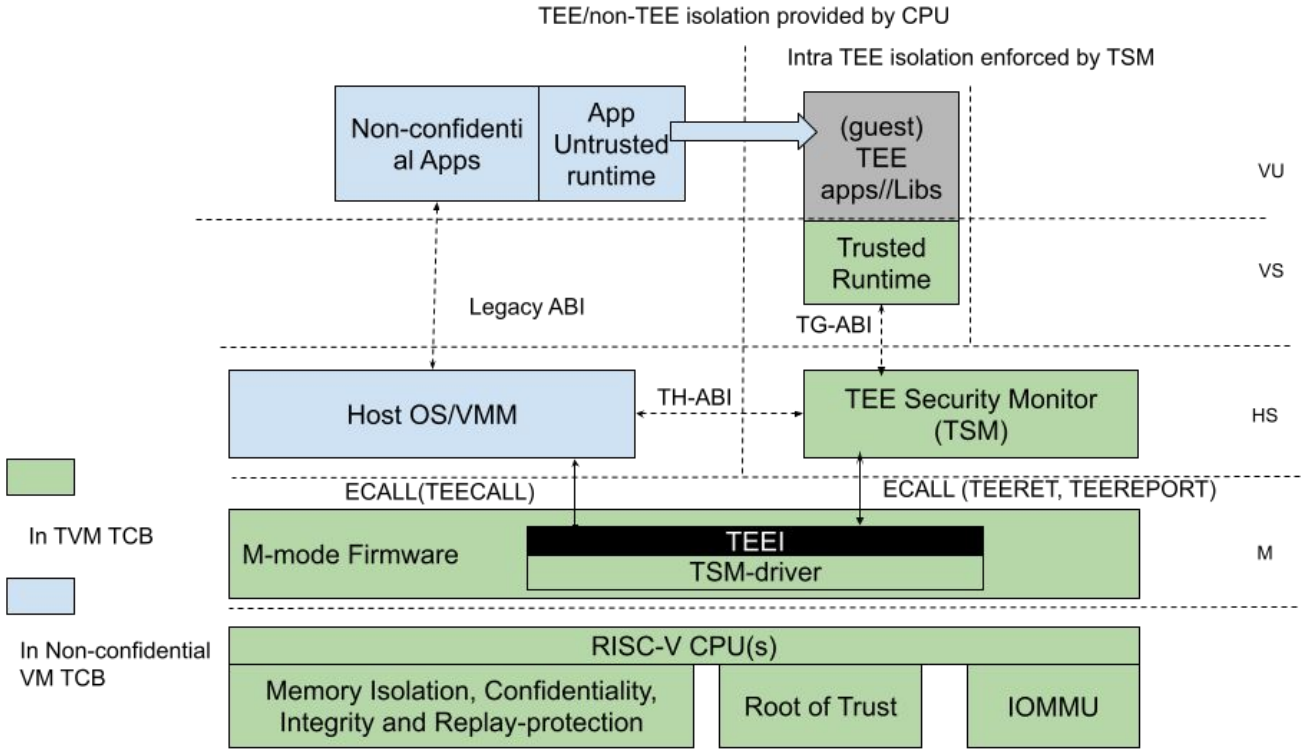
RISC-V CoVe (Confidential VM Extension)

Trusted VMs

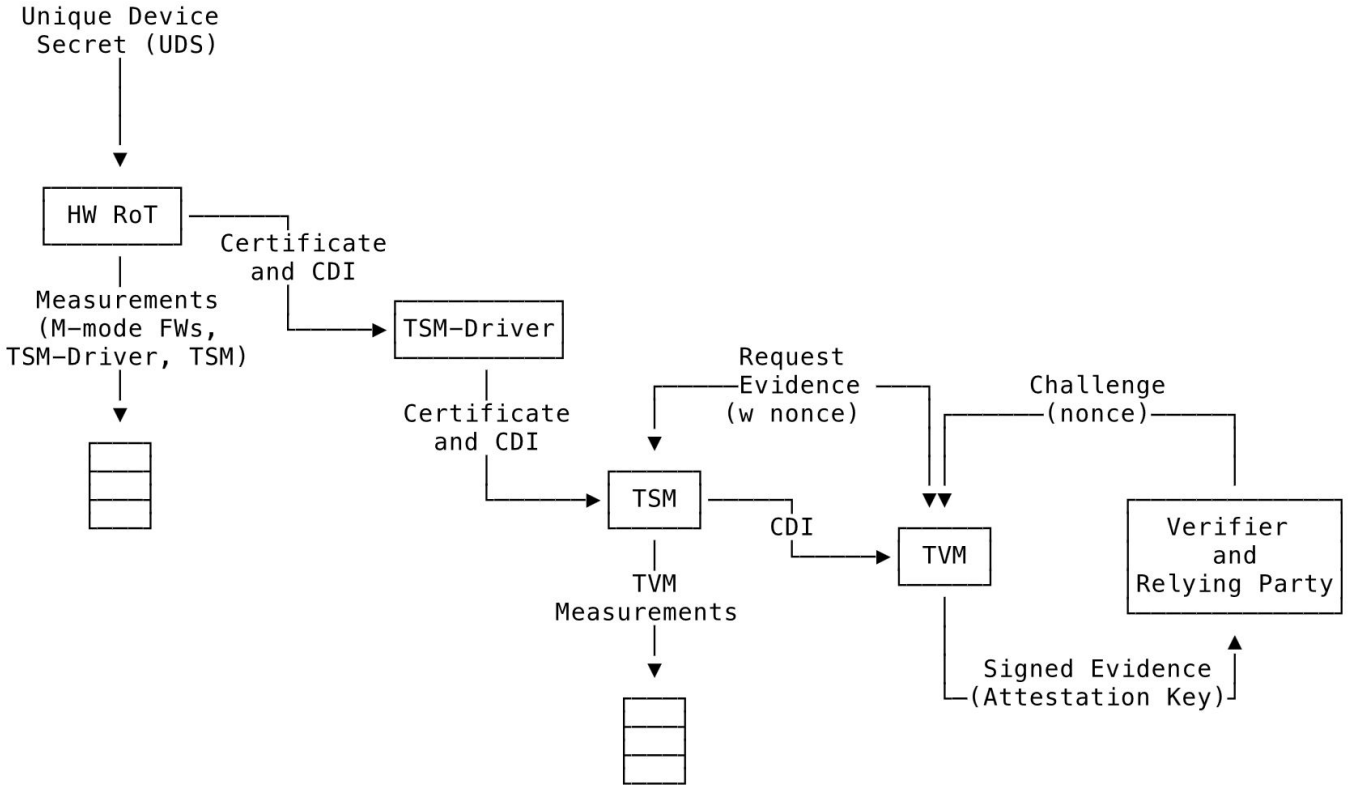


https://github.com/riscv-non-isa/riscv-ap-tee

Application workloads



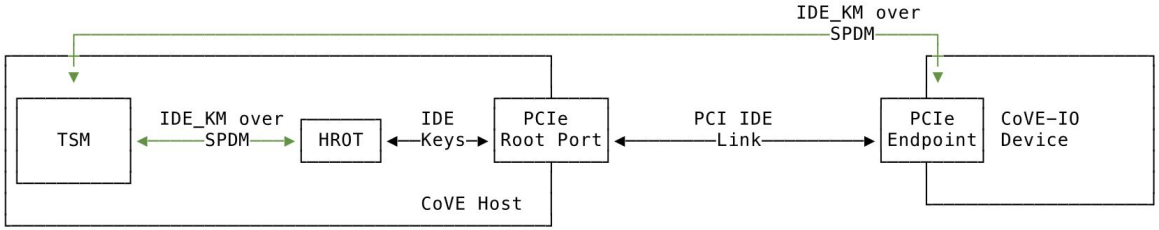
Chain of trust

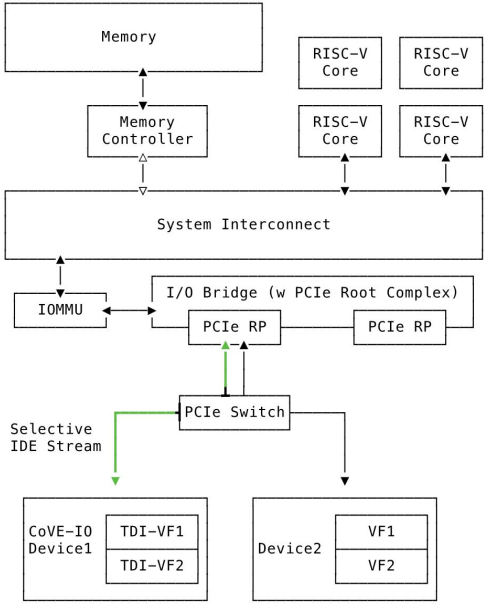


Cove-IO

Work in progress

Focuses on PCI-E devices





https://github.com/riscv-non-isa/riscv-ap-tee-io

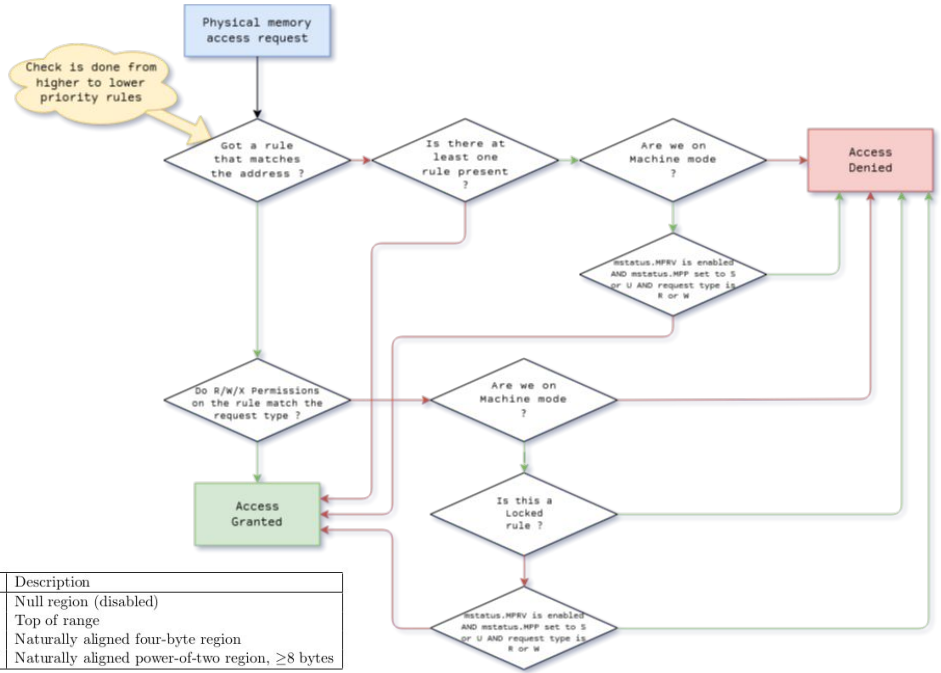
Riser

Under the hood

(Part 1: CPU-level Memory isolation)

Physical Memory Protection (PMP)

- Basic isolation between M-mode and S/U-modes.
- Normal rules apply to S/U, Locked rules (impossible to edit after adding them) apply to all modes. PMP gives access to S/U (locked down by default) and restricts M (full access by default).
- Up to 64 entries for defining physical memory regions and their permissions.
- Support for three different addressing modes (TOR, NA4, NAPOT).
- Priority matching from lower to higher indexed entries.



pmpaddr	pmpcfg.A	Match type and size
yyyy...yyyy	NA4	4-byte NAPOT range
yyyy...yyy0	NAPOT	8-byte NAPOT range
yyyy...yy01	NAPOT	16-byte NAPOT range
yyyy...y011	NAPOT	32-byte NAPOT range
...	...	...
yy01...1111	NAPOT	2^{XLEN}-byte NAPOT range
y011...1111	NAPOT	2^{XLEN+1}-byte NAPOT range
0111...1111	NAPOT	2^{XLEN+2}-byte NAPOT range
1111...1111	NAPOT	2^{XLEN+3}-byte NAPOT range

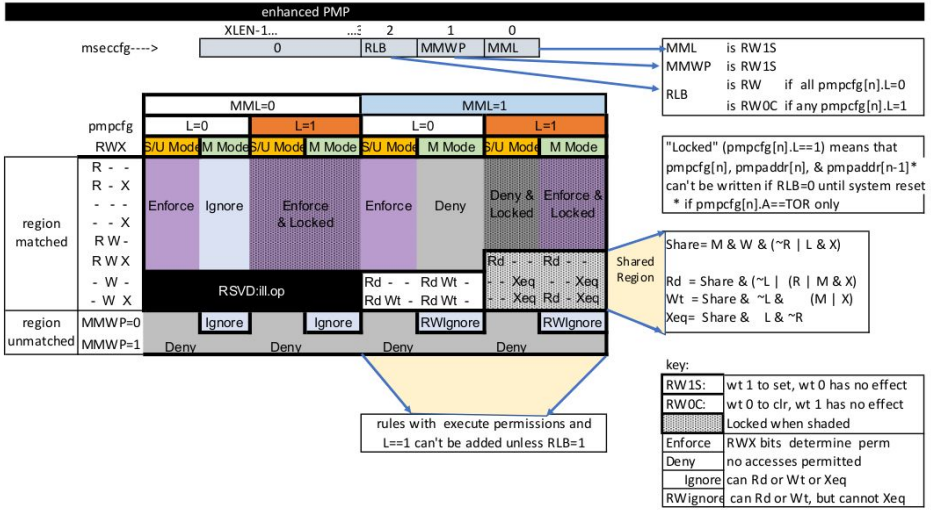
A	Name	Description
0	OFF	Null region (disabled)
1	TOR	Top of range
2	NA4	Naturally aligned four-byte region
3	NAPOT	Naturally aligned power-of-two region, ≥8 bytes

Table 3.8: Encoding of A field in PMP configuration registers.

RISC-V Privilege Spec

Enhanced Physical Memory Protection (ePMP)

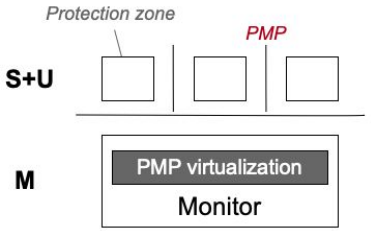
- Locked rules that apply only to M mode.
- Access/execution prevention from M-mode to S/U-mode.
- Ability to switch policy from blacklist to whitelist for M-mode.
- Ability to prevent adding new executable regions on M-mode.
- Shared regions with reduced privileges between M-mode and S/U-modes.
- Allow for greater flexibility to support more use cases.
- Ratified.



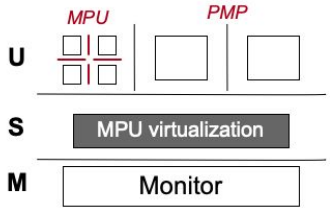
https://github.com/riscv/riscv-tee/blob/main/Smepmp/Smepmp.pdf

Supervisor PMP (sPMP)

- May be used instead of the MMU on S/HS/VS-modes.
- Similar encodings to ePMP.
- Fast switching between sets of rules.
- Useful for:
  - Supporting small trusted hypervisors on HS-mode (VS to VS and HS to VS isolation)
  - TEEs on S/U/VS-mode
  - Small IoT devices without MMU
- Under development, goal is to freeze by the end of 2024.



(a) PMP-based isolation

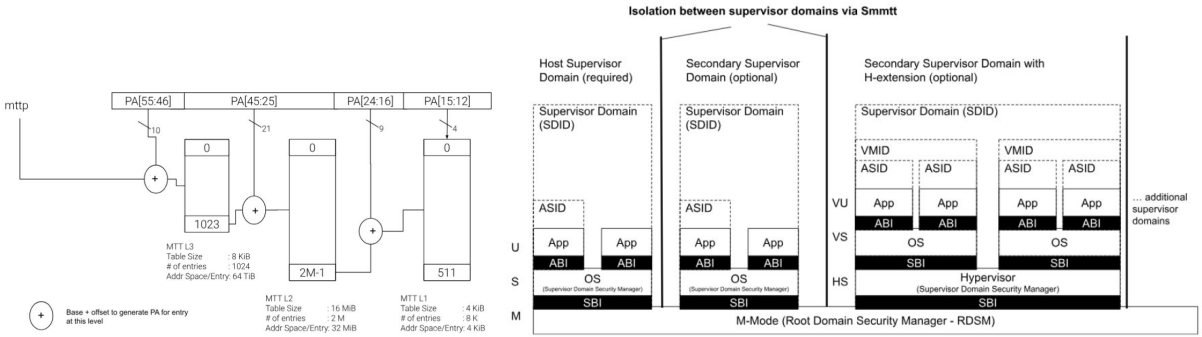


(b) MPU

https://github.com/riscv/riscv-spmp/blob/main/rv-spmp-spec.pdf

Supervisor Domains

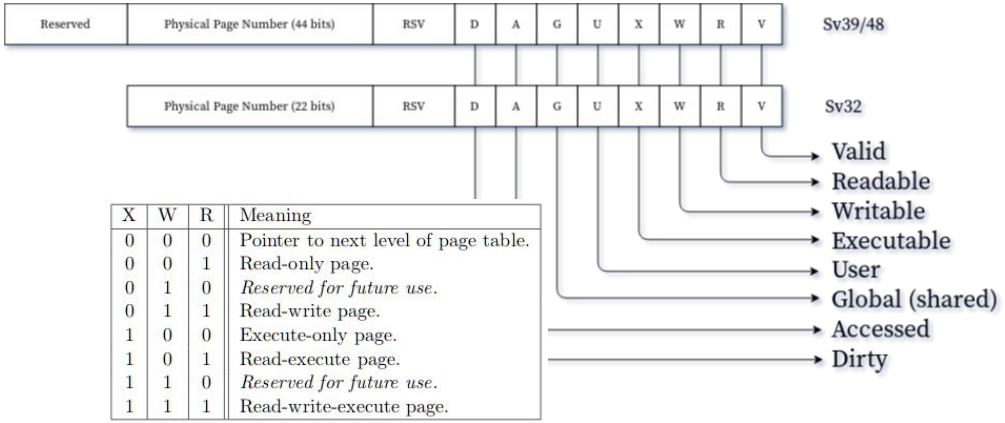
- A more flexible/extensible approach to Physical Memory Protection
- System is split in Supervisor Domains, managed by trusted M-mode software (Root Domain Security Manager - RDSM)
- Each domain is associated with a series of physical memory regions, through a table called Memory Tracking Table (MTT), held in memory.
- RDSM switches between Supervisor Domains by re-setting mtt pointer register (mttp)
- Better suited for systems with complex VMs/TEEs on S/VS-mode
- Required for confidential computing



https://github.com/riscv/riscv-smmtt

Virtual memory

- Isolation between S and U mode and between tasks on U mode.
- Also used for isolation between guests (VS - VS), and between the guests and the host (HS - VS), using a 2nd translation stage.
- SMEP is always in place, there is no way for S mode to execute pages marked with the U bit.
- SMAP is on by default but can be disabled temporarily (through sstatus.SUM) so that S-mode can read/write data from U-mode on specific code paths (e.g. copy_to/from_user() on Linux).



X	W	R	Meaning
0	0	0	Pointer to next level of page table.
0	0	1	Read-only page.
0	1	0	*Reserved for future use.*
0	1	1	Read-write page.
1	0	0	Execute-only page.
1	0	1	Read-execute page.
1	1	0	*Reserved for future use.*
1	1	1	Read-write-execute page.

RISC-V Privilege Spec

Work in progress…

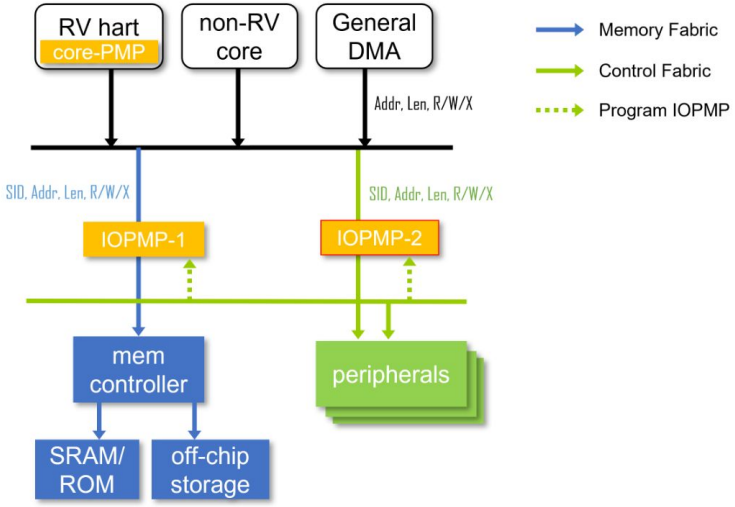
- Hardware Fault Injection (HFI)
  - https://cseweb.ucsd.edu/~dstefan/pubs/narayan:2023:hfi.pdf
- Memory Tagging
  - Task group in place, in early stages of development
- Capability-based protections (CHERI/Capstone)
  - There is another SIG for that…
- M-mode isolation / lightweight TEE
  - Most of us prefer to just privilege things on S-mode
  - But there are some use cases where we need isolation on M-mode

Riser

Under the hood

(Part 2: Platform-level Memory isolation)

I/O PMP Overview



- Provides per-device memory protection based on incoming Source ID.
- Assignment of Source IDs are out of scope.
- Number of Source IDs supported is implementation-defined.
- Still work in progress
- We try to make it compatible with WorldGuard

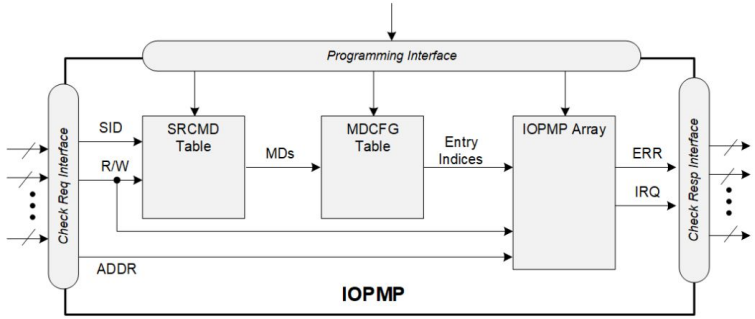
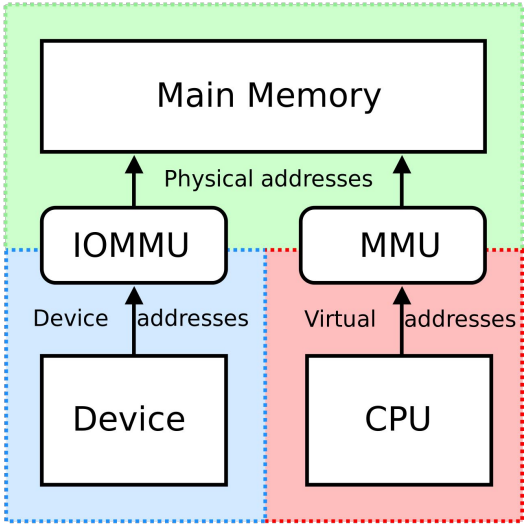


Figure 2. IOPMP Block Diagram.

https://github.com/riscv-non-isa/iopmp-spec

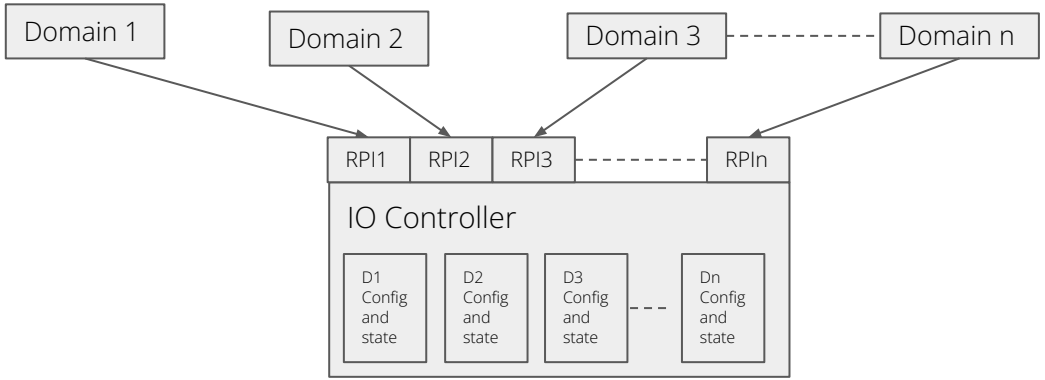
I/O Memory Management Unit (IOMMU)

- DMA Remapping service
  - Present a sparse physical region as a continuous virtual
  - Allow devices that can only access 32bit addressable memory to overcome that limitation
- Virtual memory protection like the MMU
  - Page tables have the same format as the MMU
  - Applications may share page tables with devices
- Other useful functionality
  - Interrupt remapping
  - Memory management service to peripherals
  - …
- Recently ratified !



https://github.com/riscv-non-isa/riscv-iommu

Multi-domain I/O Controller



- **Provides per-domain Register Programming Interface based on incoming Domain ID.**
- **Number of domains is implementation-defined.**
- **Domain initialization may be fixed (e.g. during system design) or dynamic (programmable e.g. through a root secure monitor / root of trust)**
- **In early stages of development…**

Riser

Questions ?

Thank you

Contact infos:

E-Mail: mick@ics.forth.gr

https://www.ics.forth.gr/carv

https://riser-project.eu