




Confidential Compute for RISC-V Platforms

AP-TEE TG
5/31/2022

Assignee - Security HC



5/31 AP-TEE TG Agenda

Disclosures

Continue AP-TEE sub-topics

- Lifecycle
- Memory Management

Future Agenda



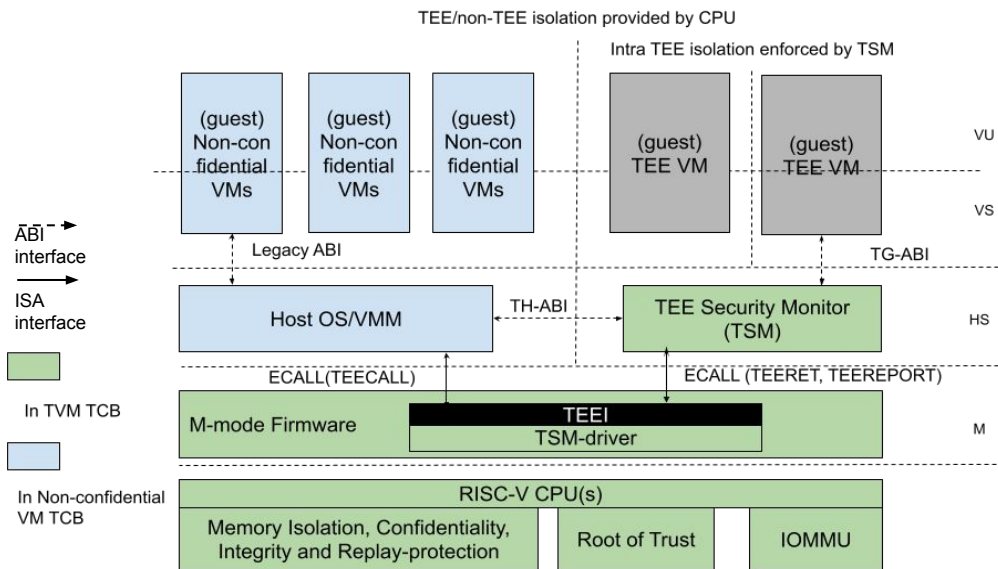
Disclosures



AP-TEE TG Charter

Define

- Reference architecture for confidential compute on RISC-V platforms
- AP-TEE TH/TG-ABIs -- normative non-ISA spec.
- AP-TEE Security Arch/Implementers Guide for RISC-V confidential compute - covers platform recommendations -- informative (living spec).
- AP-TEE ISA extension(s) -- *start with current ISA*; identify ISA gaps in TG -- request FT/TG as needed -- expected to be normative spec.



(AP-TEE TG charter - 5/17 updates)

The RISC-V Application Platform - Trusted Execution Environment Task Group (AP-TEE TG) will collaborate to define the reference architecture for confidential computing on RISC-V **Application Processor-based** platforms **that support multi-tenant virtualized workloads**. The TG will define the ABI required to enable systems software to manage confidential workloads on a multi-tenant platform, while keeping the OS/hypervisor and entities that develop the OS/VMM and/or operate/manage the platform outside the TCB. The TG will design the interfaces to comprehend existing (ratified) privileged ISA and ensure extensibility of the interfaces to new Architectural ISA extensions as required for security or performance of confidential workloads. In addition to the normative specifications mentioned, the TG will produce an AP-TEE-specific security architecture analysis per the confidential computing threat model agreed upon as a living (non-normative) document supporting security recommendations, implementation-specific guidelines and relevant standard protocols for attestation for implementers of the AP-TEE capability on RISC-V platforms. The proposed RISC-V AP-TEE task group will collaborate to define these three specifications:

a. **AP-TEE reference architecture and SBI extension interface (non-ISA, normative)** which specifies the TH-ABI and TG-ABI interfaces to enable the OS/Hypervisor to manage confidential workloads on a multi-tenant platform, while keeping the OS/hypervisor and entities that develop the OS/VMM and/or operate/manage the platform outside the TCB.

The interfaces are defined between:

1. A new **RISC-V AP** platform-specific security service called the Trusted Security Manager (TSM) operating in RISC-V HS-mode and a general-purpose OS/Hypervisor executing in S/HS-mode - called the **TH-ABI**. The TH-ABI should cover aspects of: TVM creation and tear down, TVM measurement and attestation, TVM memory management and protection, TVM virtual-hart state management and protection, TVM execution and IO.

2. A Trusted Security Manager (TSM) running in HS-mode and a general-purpose OS executing in VS-mode - called the **TG-ABI**. The TG-ABI should cover aspects the TVM is involved in: TVM measurement extension and attestation, TVM memory conversion, TVM IO and other services used from host

b). AP-TEE architecture security analysis (**non-normative** living document) supporting recommendations and implementation guidelines for e.g. coverage of threat model, attestation protocols, crypto modes

c). AP-TEE **ISA extensions (normative)** to be proposed as needed for enforcing confidential workload security and performance requirements. **The baseline ISA expected for AP-TEE is the RISC-V privileged ISA (M, S/HS, U, VS modes), including the Hypervisor Extension**. The interfaces in item a. will be defined to be extensible to any such future ISA extensions. The TG will start with the definition of the programming interfaces and identify ISA gaps. ISA proposals made will be modeled via tools such as QEMU/Spike.

The goal of the AP-TEE interface specification is to **enable open-source reference implementations of the RISC-V AP-TEE interfaces** for platform-specific TSM implementations that enable confidential compute and trusted execution for different use case scenarios (Server, Automotive, Embedded etc.). To support this goal, a POC is defined that consists of: An SBI extension implementation for AP-TEE will be used as a reference implementation. A TSM implementation will be developed by the community as part of the ratification of the interfaces. The required changes will be made to the Linux/KVM host and guest software to validate the interface specifications.

AP-TEE TG Charter: Interface specs

Specs



Area	Function	Resources
AP-TEE TH-ABI	SBI Extension Interface implemented by the TSM via ECALL for use by OS/VMM to manage TVMs	TG WG members
AP-TEE TG-ABI	SBI Extension Interface implemented by the TSM via ECALL for use by TVM guest workloads	
TEE Security Manager (TSM)	TSM is a RISC-V 64 bit SW module that uses RISC-V H-extension and implements TH and TG-ABI. It is in the TCB for all TVM workloads (Expected to be HW-vendor signed and may be HW-operator signed)	Rivos contributes to start collab.
M-mode FW	Minimal SBI extensions (TCB component) to support TSM initialization, TEECALL, TEERET implementation. It is in the TCB for all TVM workloads (Expected to be HW-vendor signed and may be HW-operator signed) - Collab with OpenSBI	Expecting collaborators on these existing projects from Software HC
Linux, KVM (Host OS/VMM)	<i>Untrusted</i> (enlightened) host OS/VMM that manage resources for TVM-based confidential workloads [TSM enforces security properties] - Collab with Hypervisor SIG	
Linux (TVM Guest OS), Guest Firmware	Enlightened guest OS/runtime (in TCB of TVM workload) - Collab with SW HC	

POCs



AP-TEE TG Charter: Platform & ISA (Scope)

Area	Function	Resources
CPU	Evaluate AP-TEE mode qualifier, Sparse (page-based) confidential memory access-control	TG members
IOMMU	AP-TEE mode qualifier; Sparse (page-based) confidential memory access-control and fabric i/f	w/ IOMMU TG
TLB, Caches	AP-TEE mode qualifier and other micro-architectural structures	TG members
Interconnect, Fabric	Platform-specific cryptographic memory isolation and mode qualifier	TG members to document + Implementation feedback
Memory	Platform-specific cryptographic memory isolation and mode qualifier	
HW Root-of-trust	Platform-specific subsystem to support HW Attestation, Sealing interfaces	
Devices	Device-specific subsystem to support Device attestation, link security	
QoS, RAS, DC	Platform-specific, Domain-specific	w/ SOC Infra

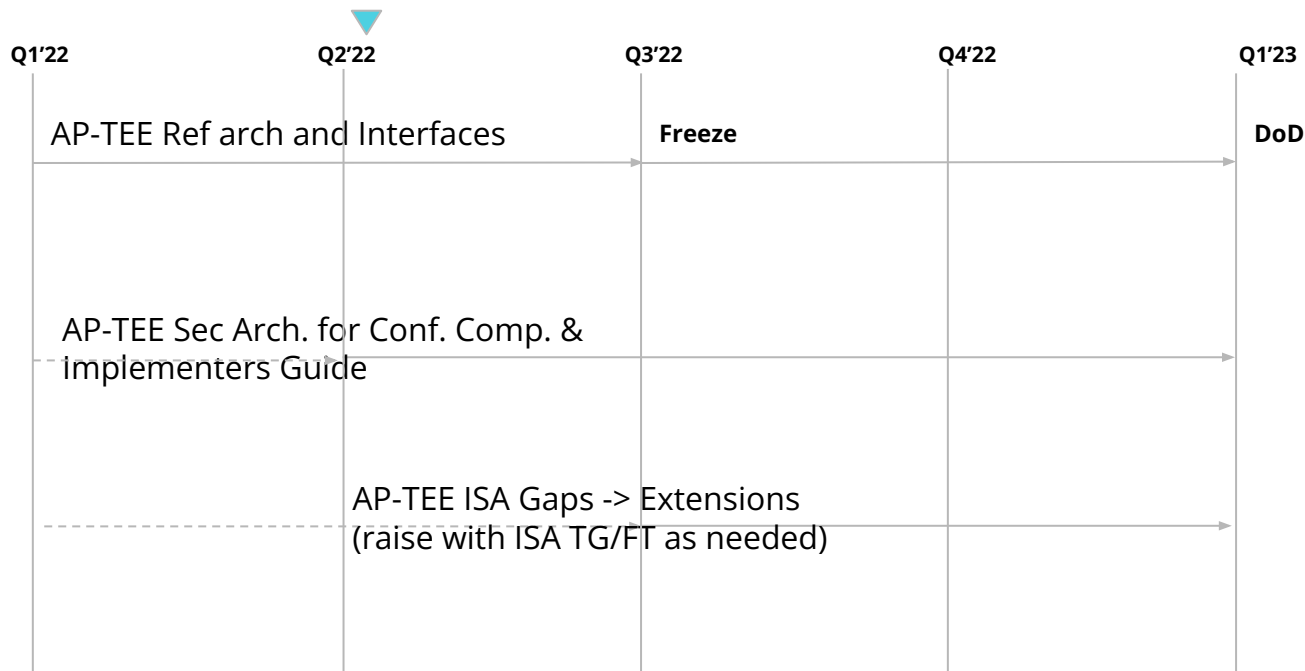
AP-TEE Security Arch for CC and Implementers Guide covers recommendations on:

- Mapping of mitigations to threat model
- Recommendations for crypto modes
- Attestation protocols, formats

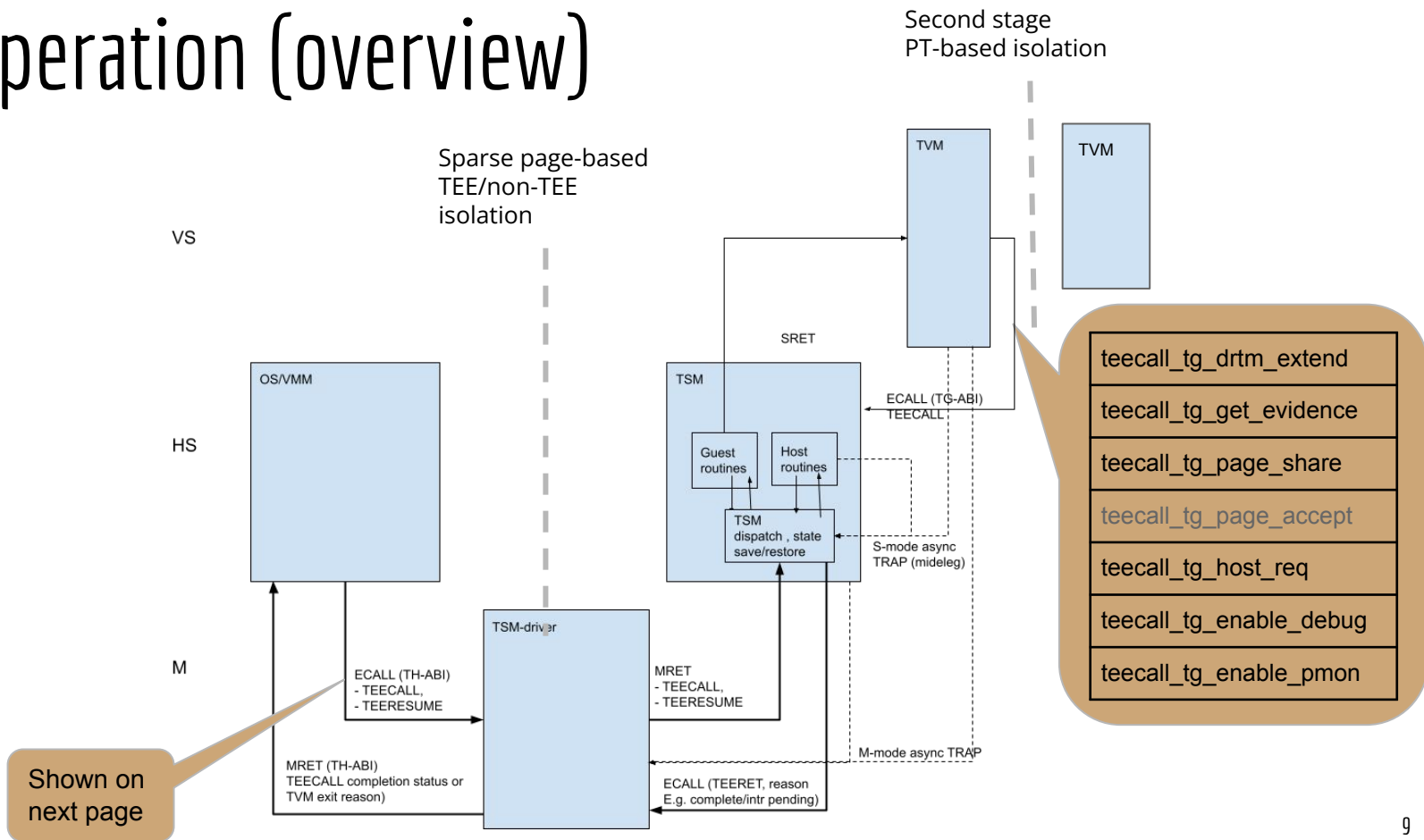
AP-TEE TG workstreams

- [Proposed ratification plan](#)
- [DoD checklist](#)
- [Infra requirements](#)

-
- Initial [AP-TEE spec](#) being discussed at the TC SIG.
 - [Confidential Computing Survey](#)



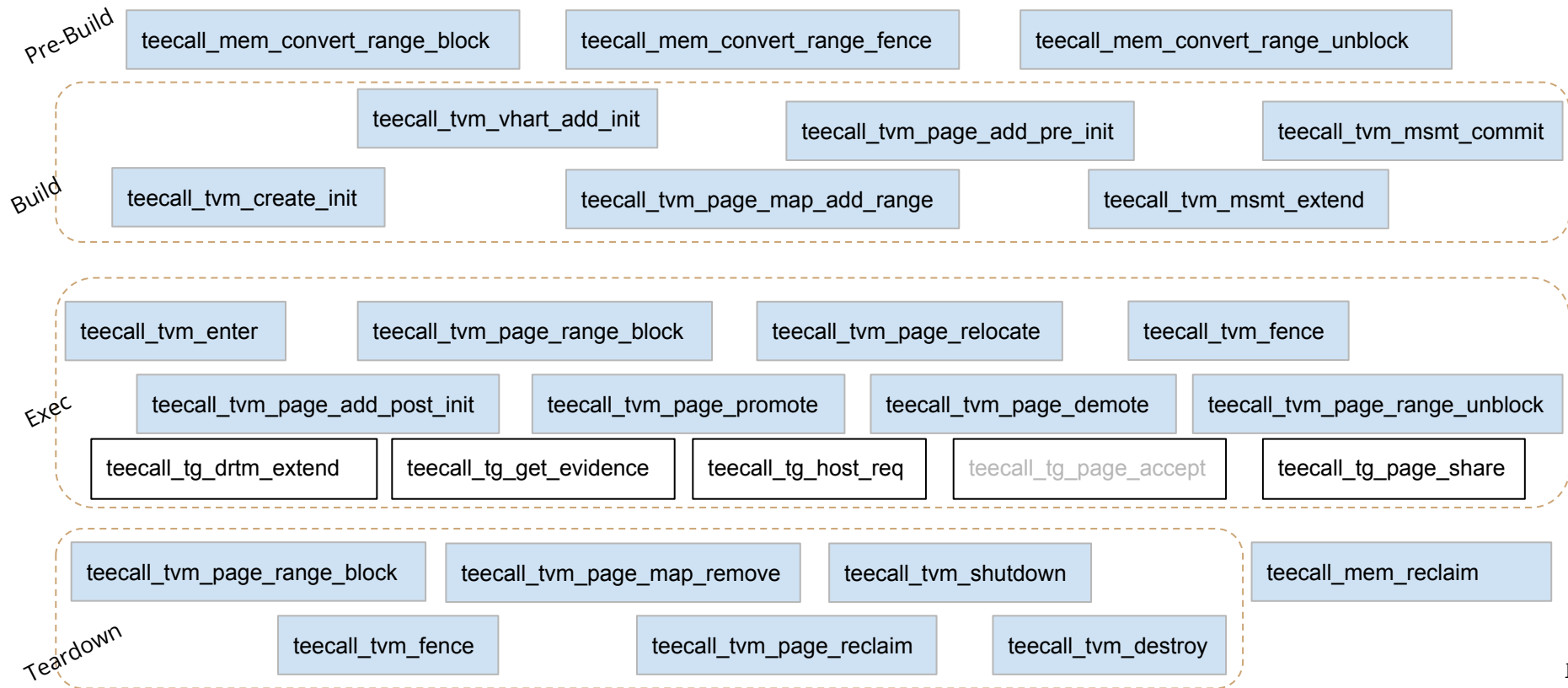
TSM operation (overview)



TVM Lifecycle

TBD:

- Add interrupt mgmt flows
- Add device assignment flows



AP-TEE CPU Mode

- Drives access-control and isolation properties for TVM state stored in
 - Memory Isolation and Encryption
 - CPU caches
 - CPU registers
 - CPU micro-architectural state
- Enabled on a hart via a successful TEECALL <SBI call>
- Disabled on a hart via a successful TEERET <SBI call>
- Specify via new CSR - mapteectrl <priv ISA>

AP-TEE Memory Isolation

A platform that supports AP-TEE requires the CPU to support new PMAs: **Confidential** and **Non-Confidential** [dynamic/programmable memory attributes]

A TVM needs to access two types of memory

- **Confidential memory** - has Confidential PMA - trusted used for TVM code, data
- **Non-Confidential memory** - has NC PMA - untrusted used for communication between TVM and untrusted entities

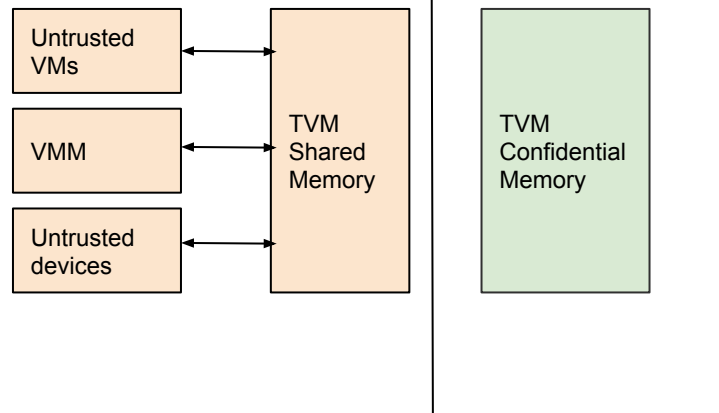
TSM provides TH-ABI primitives to the VMM to convert memory to Confidential [Assign] and vice-versa [Reclaim]

TVM memory is by default assigned from Confidential memory regions [enforced by the TSM]

TVM may indicate to convert memory assigned to Non-Confidential via TG-ABI (and back)

=> CPU AP-TEE mode =1 is allowed to access both Confidential and Non-Confidential memory

=> CPU AP-TEE mode =0 is allowed to access only Non-confidential Memory



AP-TEE Confidential PMA Restrictions

Confidential PMA required for **implicit accesses** including:

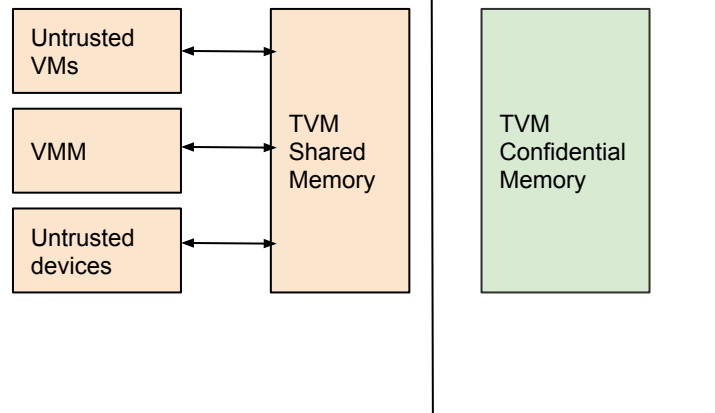
- Instruction fetch - security property: TVM cannot fetch code from untrusted shared memory
- First level paging structure walk - security property: TEE cannot locate page tables in untrusted shared memory

Memory with Confidential PMA may be associated with a unique memory encryption key (and similar IO/fabric protection policies)

Non-confidential memory is assigned by the VMM - the TSM and TSM-driver update the Confidentiality PMA by programming a

Memory Tracking Table

TSM manages finer-granular (page-based) allocation from Confidential memory

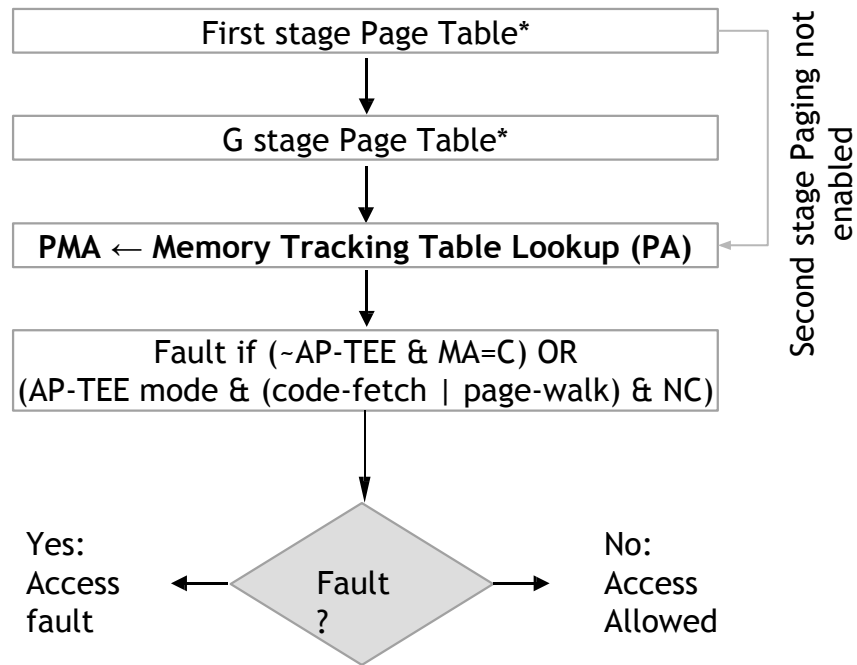


AP-TEE TVM Memory Access

MTT provides PMA → Confidential or Not-Confidential attribute

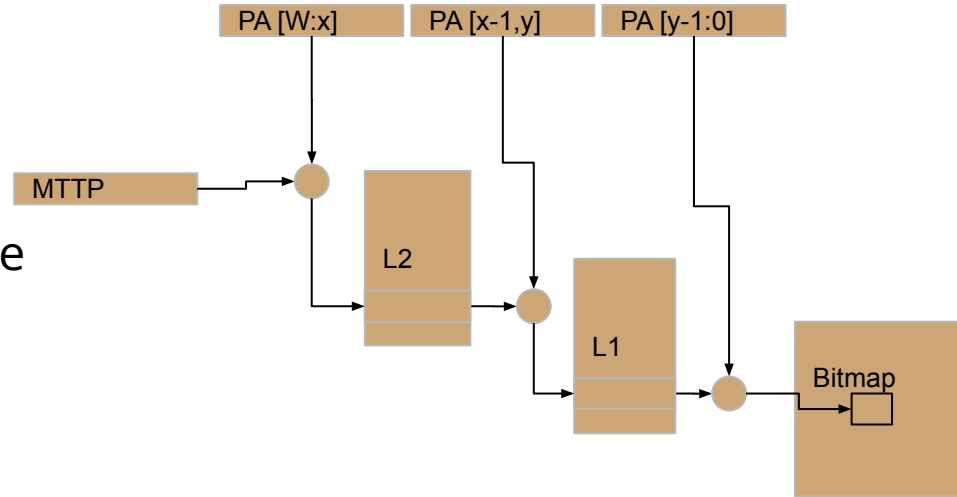
TSM maps memory pages to specific TVMs (via G-stage translation) - pages may belong to C|NC PMA

*See RISC-V [priv-isa spec](#) for paging modes



An Example Memory Tracking Structure

- Options: Flat or Hierarchical structure
- MTTP CSR read/write accessible from TCB.
- When enabled, MTT looked up to determine confidentiality attribute for implicit and explicit accesses
- Implementation may choose to cache MTT to optimize



Page Conversion (TH-ABI)

teecall_mem_convert_range_block

teecall_mem_convert_range_fence

teecall_mem_convert_range_unblock

Post measured-boot, TSM must know the addressable Confidential memory on the platform

- Memory region used for the TSM are setup in the MTT by the TSM-driver
- the TSM manages the MTT from init onwards.

Page conversion involves the following steps by the TSM:

- Verify page(s) donated by the VMM is/are Non-Confidential page(s)
- Create **blocked MTT entries** (synchronized) for the requested page(s) and size as Confidential pages
- TSM enforces a **TLB version scheme** and enforces that the VMM perform the **invalidation** of the hart TLBs (via IPIs) to remove any cached mappings
 - Pages may be scrubbed later since page contents may be initialized before assignment to a TVM
- VMM completes the donation of selected pages by **unblocking MTT entries** - TSM verifies that TLB fence (c) was completed for the pages selected for conversion
- At this point non-TEE mode software cannot create new TLB entries to donated pages - since non-TEE mode accesses to MTT-tracked Confidential pages will fault (including implicit accesses)

Page Assignment (TH-ABI)

```
teecall_tvm_page_map_add_range
```

The following are the security requirements/invariants for enforcement of the memory access-control for memory assigned to the TVMs. These rules are enforced by the TSM and the HW:

1. Contents of a TVM page assigned (statically measured or lazy-initialized) to the TVM is bound to the Guest PA assigned to the TVM during TVM operation. [TSM-enforced]
2. Single owner - a TVM page can only be assigned to a single TVM, and mapped via a single GPA unless aliases are allowed in which case, such aliases must be tracked by the TSM). Aliases in the virtual address space are under the purview of the TVM OS. [TSM-enforced]
3. 1st stage address translation - A TVM page mapping must be translated only via first stage translation structures which are contained in confidential type pages, and assigned to the same TVM [HW, TSM-enforced].
4. 2nd stage address translation [TSM-enforced]:
 - a. A TVM page guest physical address mapping must be translated only via the TSM-managed second stage translation structures for that TVM.
 - b. 2nd stage structures may not be shared between TVMs, and must not refer to any other TVMs pages.
 - c. The OS/VMM has no access to TVM second stage paging structures (or ciphertext) [HW-enforced]
 - d. Circular mappings in the second stage paging structures are disallowed.
5. Access to shared memory pages must be explicitly signaled by the TVM and enforced for memory ownership by the CPU (via TSM).

TVM TLB mgmt (TH-ABI)

Similar model as conversion applied for TVM pages - If the VMM requires to make any change to an assigned and present GPA mapping (e.g. remove, relocate, promote etc.) - then it must execute the following sequence (enforced by TSM) to affect that chance:

teecall_tvm_page_range_block

teecall_tvm_fence

teecall_tvm_page_<op>

...

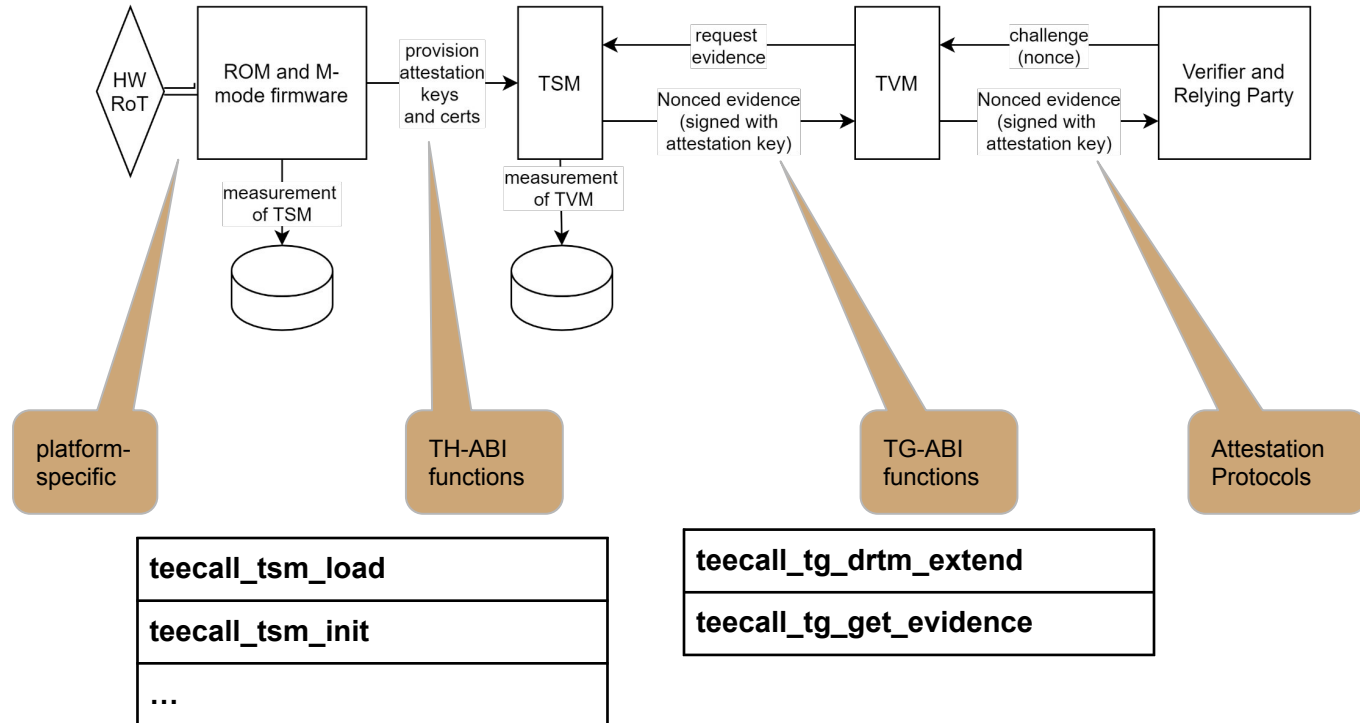
teecall_tvm_page_range_unblock

1. **Block the mapping it wants to modify (page or range of pages)**
 - a. Prevents new mappings from being populated in the TLB (and requires unblock)
 - b. in the PA metadata maintained by the TSM, captures the tvml_tlb_tracker version into per page pg_tvm_tlb_tracker
2. **Initiate Fence/increment the tvml_tlb_tracker version for the TVM (on any one hart)**
 - a. Checks that the previous TLB synchronization was completed [refcount corresponding to tvml_tlb_tracker is 0], and then increments the tvml_tlb_tracker counter (Only previous and current refcount has to be maintained)
3. **VMM must issue an IPI to each TVM virtual-hart executing to trap to the TSM -- enables the TSM to perform a Hfence.GVMA**
 - a. TVM exit/trap decrements the refcount for active TVM virtual-harts [reported to VMM as normal]
 - b. This step prevents pre-existing (stale) mappings from being utilized
 - c. A (subsequent) virtual-hart being entered compares the tvml_tlb_tracker with the vhart_tlb_tracker version and on a mismatch is subject to a flush during TVM Enter (and brings vhart_tvm_tlb_tracker up to date with the tvml_tlb_tracker)
4. **-----No active/usable translations for affected TVM 2nd stage mappings after this point-----**

TVM TLB mgmt (TH-ABI) /2

5. Invoke the specific **mapping change** operation (remove, relocate, promote, migrate etc.)
 - a. This step checks that the affected mapping(s) are blocked in 2nd stage mapping
 - b. and, uses the recorded PA metadata `pg_tvm_tlb_tracker` to check that the TLB synch in step 3 was completed (`refcount=0`)
6. **Unblock** the mapping(s)
 - a. Checks that TLB flushing step 3 has completed for TVM virtual-harts, then unblocks the 2nd stage mapping (& clears PA blocking meta-data)

TVM Attestation

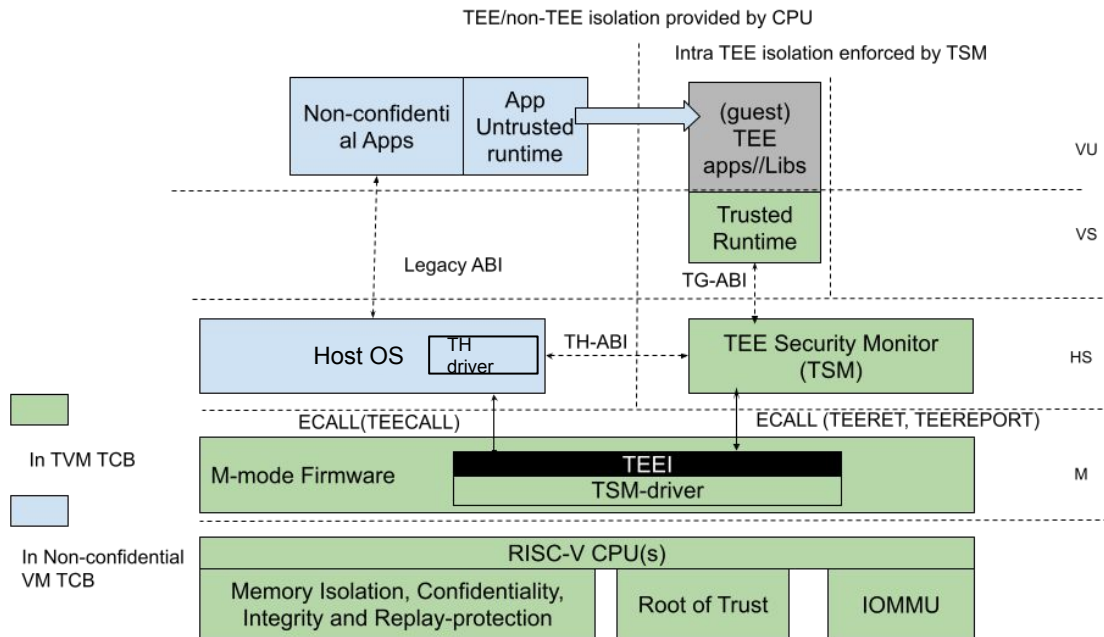


TG discussion areas

- SW Lifecycle
- Confidential Memory Mgmt
- Interrupt Mgmt
- Attestation
- Physical Protection - Encryption
- Confidential process (SW model)
- Direct IO assignment to TVMs
 - IOMMU discussions ongoing to establish direction
- Domain partitioning
- Stolen time
- TVM Live Migration
- TVM Checkpointing
- Others?

Extra Slides

Ref. Arch use for confidential applications



Role of M-mode

- M-mode firmware hosts the TSM-driver that programs HW to provide the isolation of the TSM memory via following mechanisms:
 - Cryptographic mechanisms (confidentiality, integrity) and/or
 - Access-control mechanisms (page ownership tracking or sequestered memory)
 - Owns context switch from host OS/VMM to TSM and vice versa (TEE ABI functions invoked via ECALL to M-mode (TEECALL, TEERET, TEEREPOROT)
 - Activates “AP-TEE mode” on the hart
- M-mode programs HW engines for access-control, memory confidentiality, and RoT for TSM reporting.

Role of the TSM

TSM enforces isolation of TVM memory (amongst TVMs) via 2nd stage translation

TSM enforces isolation of TVM hart state when invoked by the host VMM to schedule TVM virtual-hart on a physical hart.

TSM maintains TVM measurement for attestation

TSM itself measured by the TSM-driver and HW RoT.