Confidential Computing Devroom
(formerly Hardware-Aided Trusted Computing Devroom)

FOSDEM'23

Rust based Shim-Firmware for Confidential Container

Jiewen Yao, Principal Engineer, Intel

February, 2023

Notices & Disclaimers

- Intel Corporation (“Intel”) provides these materials as-is, with no express or implied warranties.
- No license (express, implied, by estoppel, or otherwise) to any intellectual-property rights is granted by this document.
-

Speaker

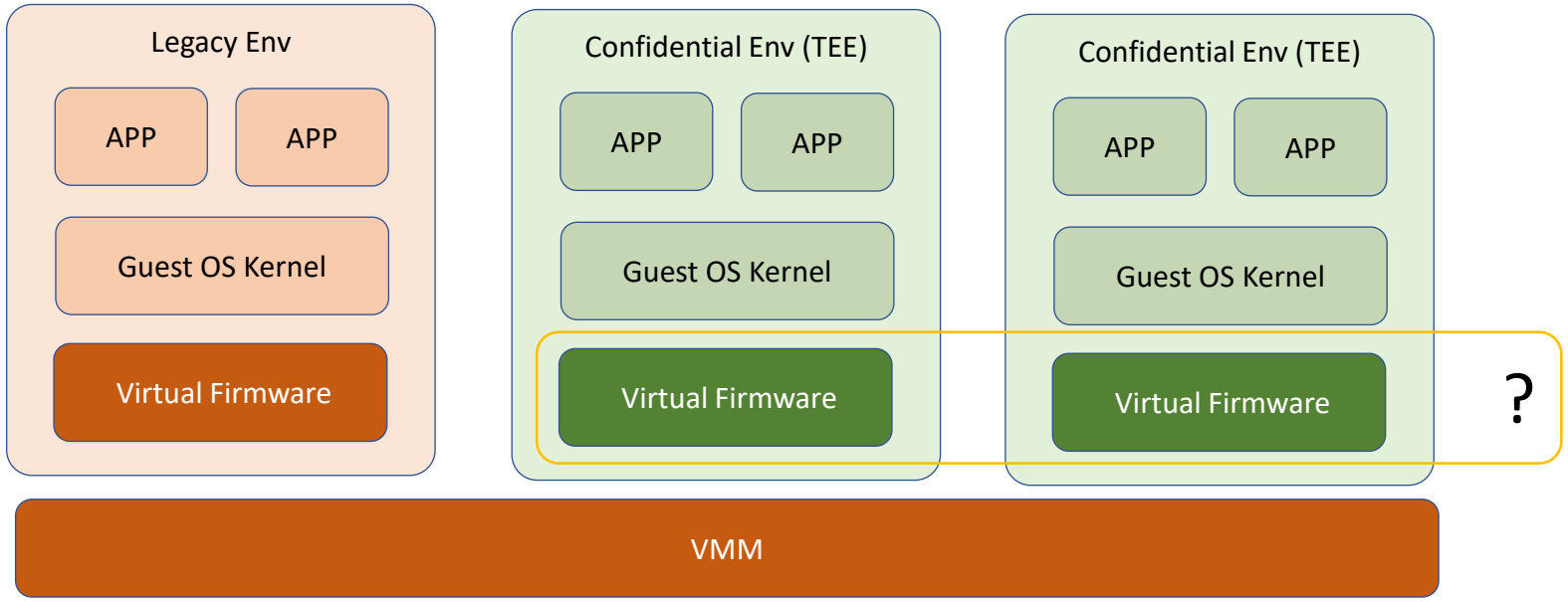
Jiewen Yao is a principal engineer in the Intel Software and Advanced Technology Group. He has been engaged as a firmware developer for about 20 years. He is a member of the UEFI Security sub team, the TCG PC Client working group, and charing DMTF SPDM Code Task Force.

He is the architect for Intel® TDX virtual firmware.

Agenda

- Background
- Why shim-firmware?
- TD-Shim Internal

Need of virtual firmware



Virtual Firmware Solution

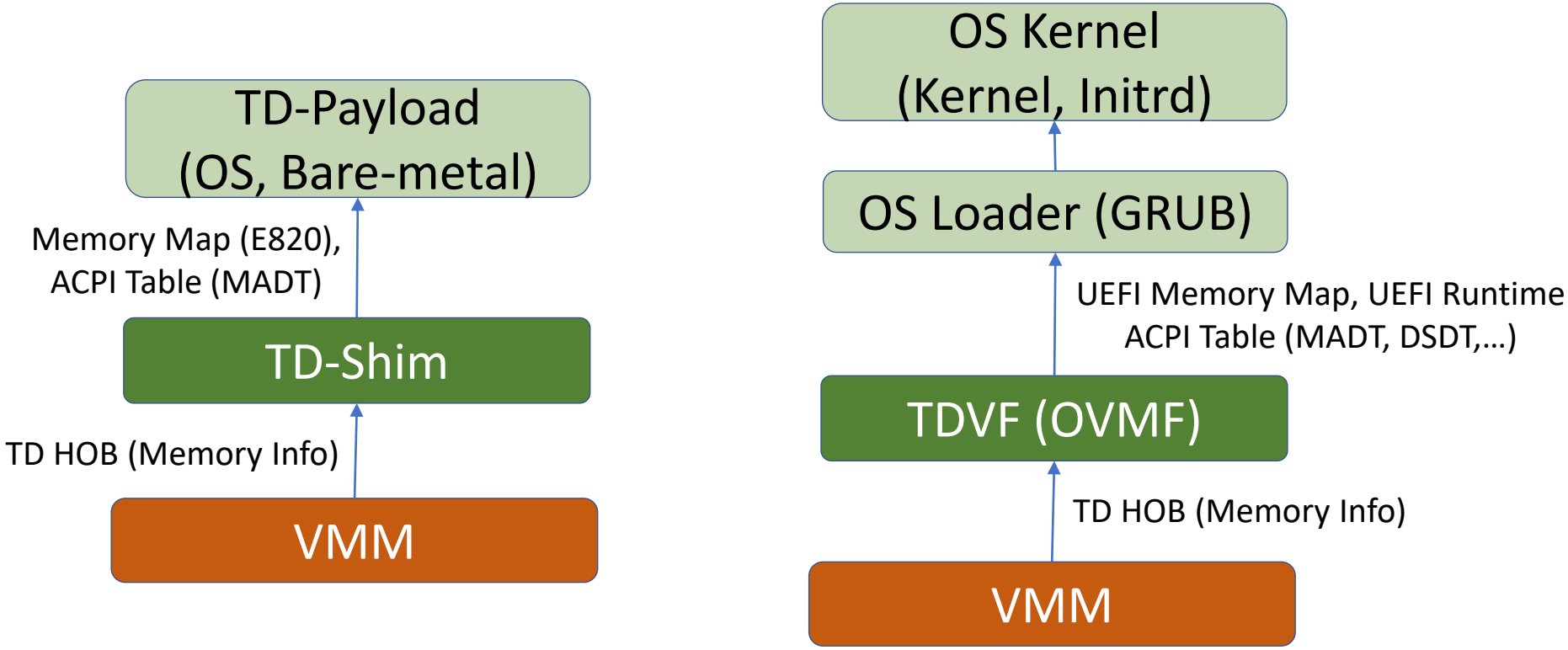
Main Feature	SeaBIOS	OVMF	cloud-hypervisor-firmware
Hypervisor	XEN, KVM, …	XEN, KVM, …	cloud-hypervisor, …
Arch	16 bit	32bit/64bit	64bit
VMM-BIOS Entrypoint	16bit Reset Vector	16bit Reset Vector	ELF Entrypoint
BIOS-OS Interface	Legacy 16bit INT	UEFI Specification	Linux Boot Protocol
Gap Analysis (TDX)	1. Entrypoint – 32bit Reset Vector 2. MP Wakeup – special wakeup structure 3. Memory Initialize – memory accept before use 4. DMA Management – shared/private memory switch 5. Measurement – Runtime Measurement Register (RTMR) extend		
Solution	N/A	TDVF (upstreamed)	TD-SHIM

td-shim

- A **lightweight** virtual firmware for **confidential container** environment.
- Written in Rust
- Support Intel ® TDX
  - https://github.com/confidential-containers/td-shim

- Responsibility
  - Own the 1st instruction (reset vector) of a TD
  - Provide the required boot information (memory map, CPU info) to the next phase (payload)
  - Build the chain-of-trust from Intel ® TDX-module to the next phase

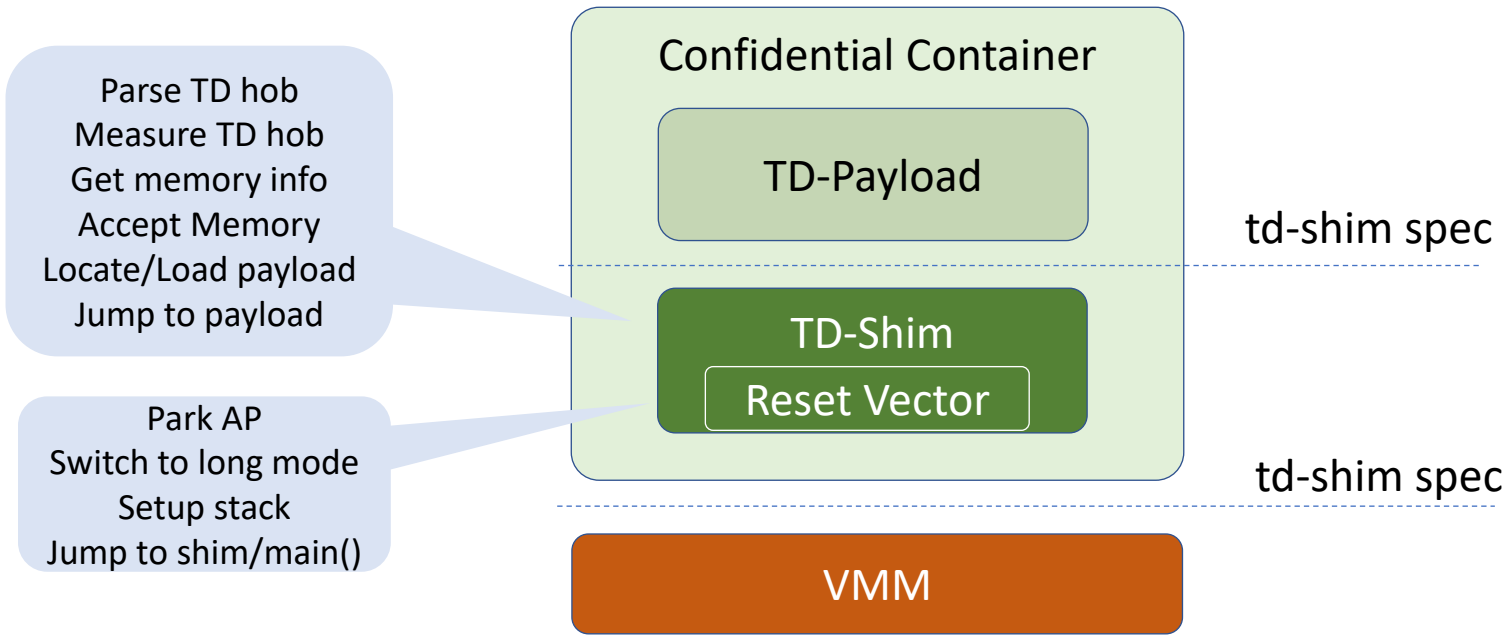
td-shim boot vs. TDVF boot



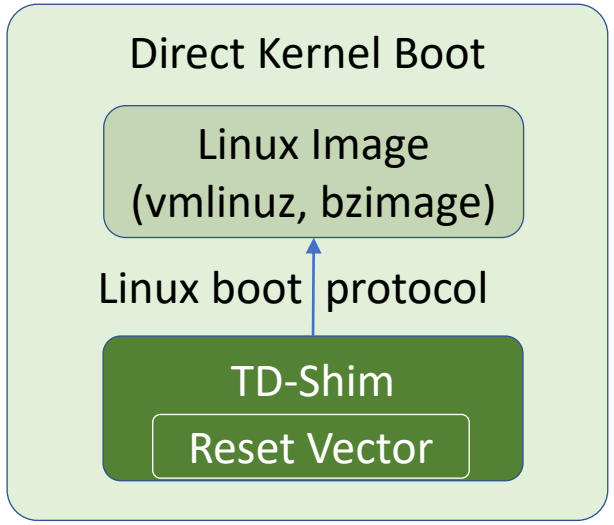
td-shim vs. TDVF

	TD Shim	TDVF
Use Case	Confidential Container, Small Service TD	Confidential VM, Rich Service TD
Language	RUST (no-std) + ASM	C + ASM
UEFI Service & Features	NO	Network, File System, etc
OS Runtime	NO	UEFI RT, ACPI ASL
Device Driver	NO	Virtio, PCI, etc
ACPI Table (MP Support)	Static table only (MADT,..). No DSDT.	All (MADT, DSDT, …)
IRQ Info	Other (Boot Param, …)	ACPI DSDT
Memory Map	E820 table	UEFI Memory Map
Trusted Boot	YES (RTMR + EventLog)	YES (RTMR + EventLog)
Secure Boot	Optional	Optional (UEFI Secure Boot)
Image Size (release)	140K (w/o SecureBoot) 270K (full feature, w/ SecureBoot)	4M by default.

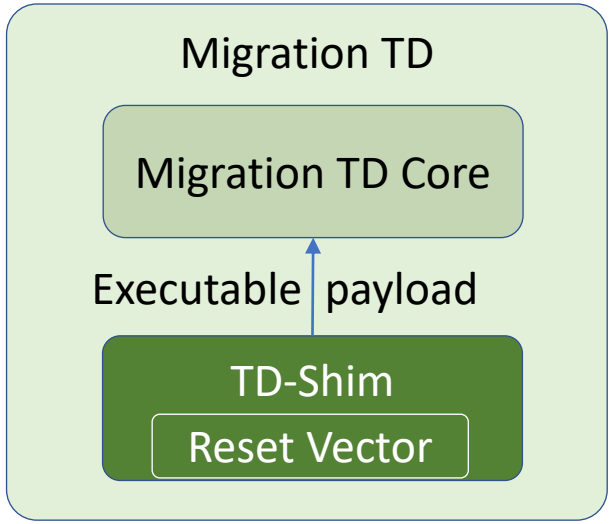
td-shim boot


Parse TD hob
Measure TD hob
Get memory info
Accept Memory
Locate/Load payload
Jump to payload
Park AP
Switch to long mode
Setup stack
Jump to shim/main()
Confidential Container
TD-Payload
TD-Shim
Reset Vector
td-shim spec
td-shim spec
VMM

td-shim use cases



Confidential Container



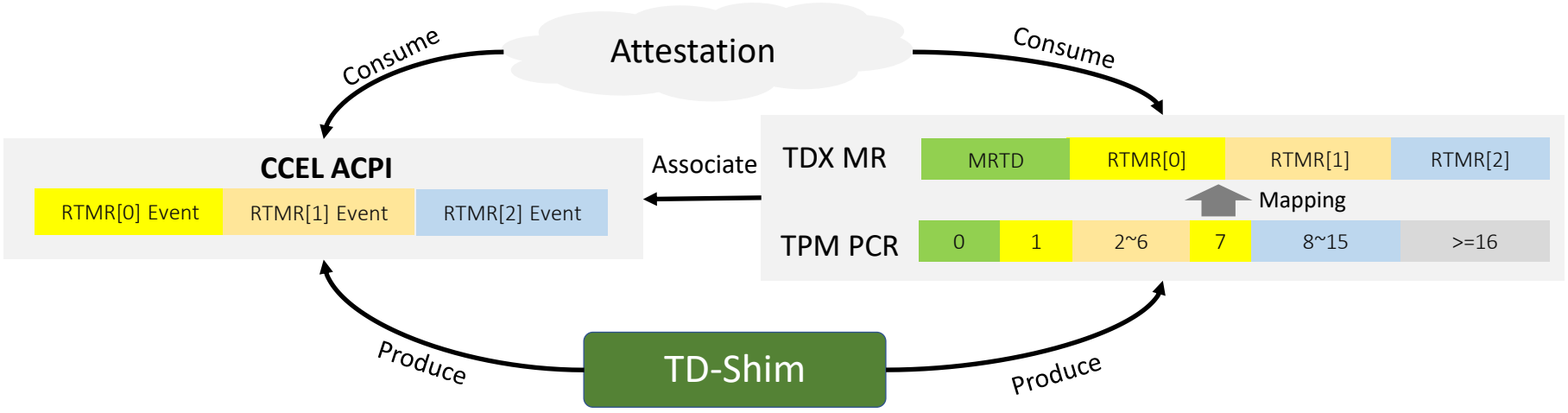
Service TD

td-shim feature

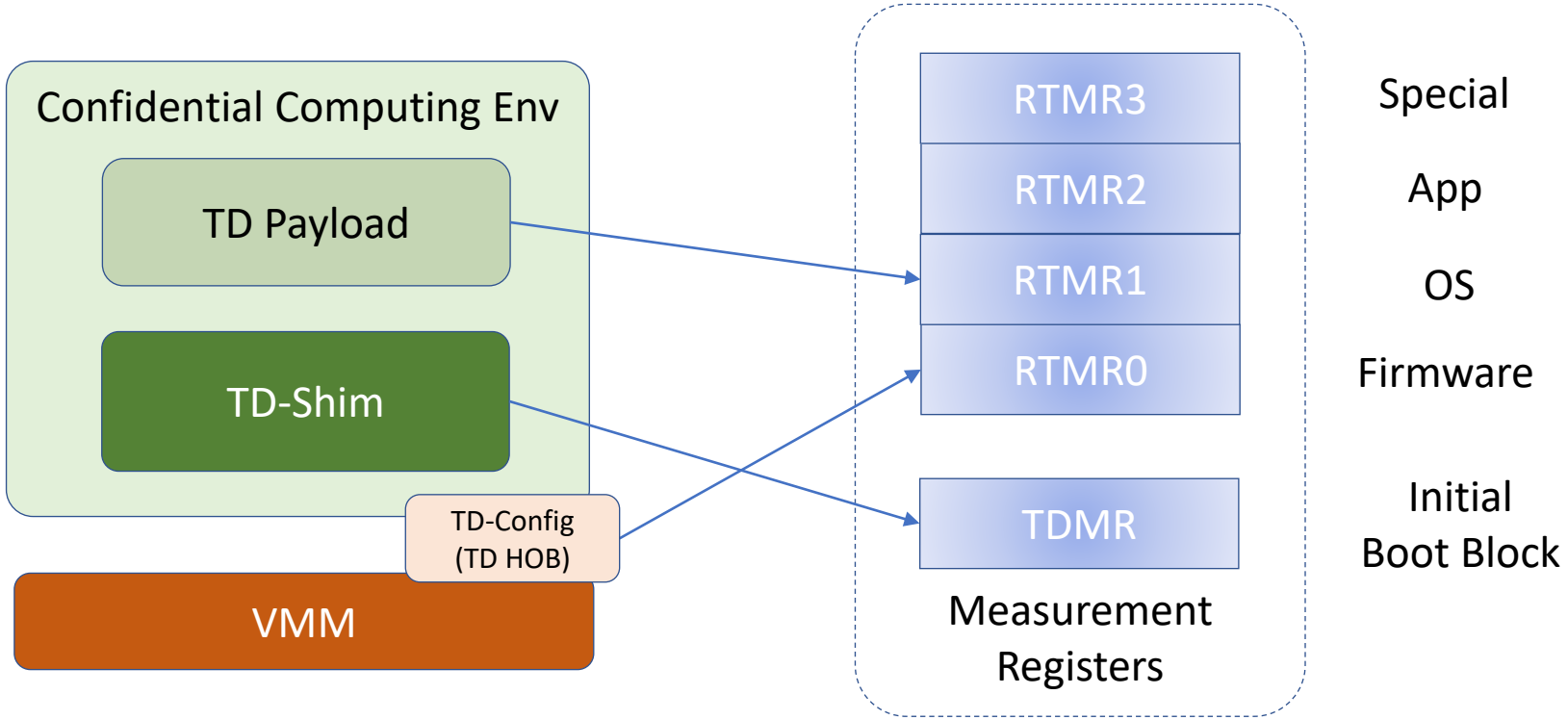
- Trusted Boot
  - https://github.com/confidential-containers/td-shim/blob/main/doc/tdshim_spec.md#guideline
- Secure Boot
  - https://github.com/confidential-containers/td-shim/blob/main/doc/secure_boot.md

Trusted Boot

- td-shim extends measurement to TD runtime measurement register (RTMR)
- td-shim provides event log (CCEL) to reproduce the value in RTMR.
- Attestation can be based upon MR register or event log.

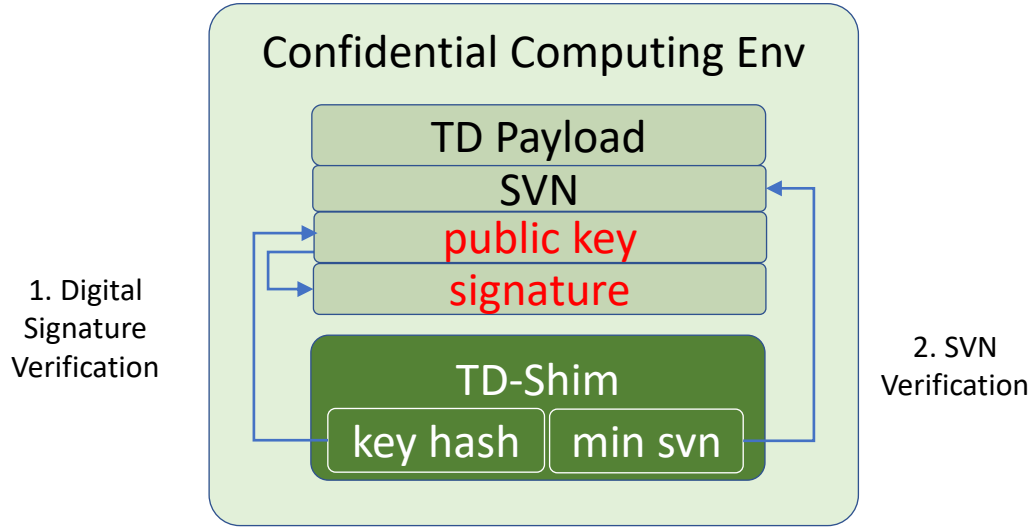


Trusted Boot

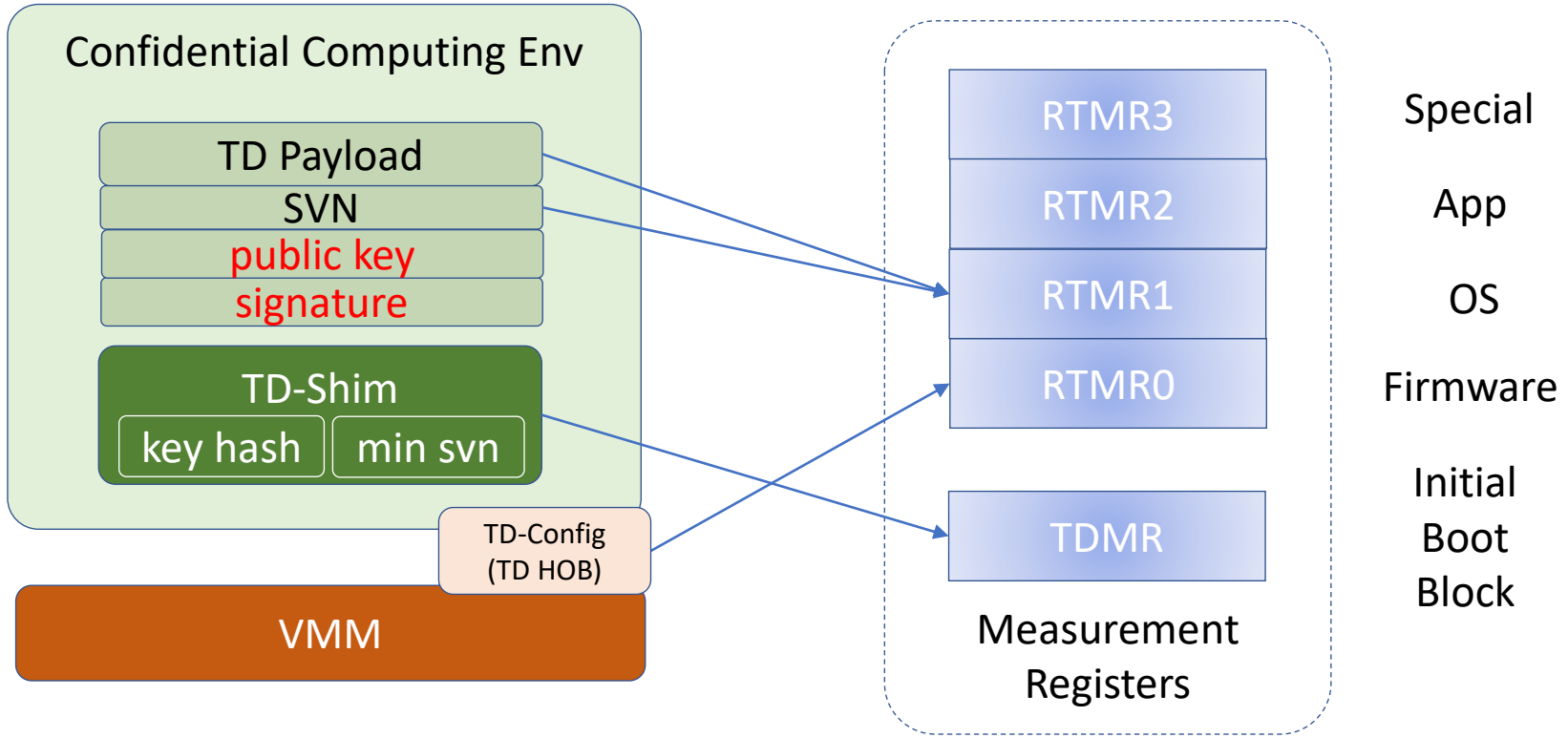


Secure Boot

- Verify the next component before launch
- Need to provision the known good public key and secure version number (SVN)
- Payload attestation can be based upon SVN value, not image hash.



Secure Boot


Confidential Computing Env
TD Payload
SVN
public key
signature
TD-Shim
key hash
min svn
TD-Config
(TD HOB)
VMM
RTMR3
RTMR2
RTMR1
RTMR0
TDMR
Measurement
Registers
Special
App
OS
Firmware
Initial
Boot
Block

Other Features

- Data Execution Protection (DEP)
  - Page table based enforcement.
  - DataPage = Non-Executable
  - CodePage = Read-Only

- Control Flow Enforcement (Intel ® CET)
  - Backward-Edge control flow - Shadow Stack (SS)
  - Forward-Edge control flow – Indirect Branch Tracking (IBT)
    - Depend upon compiler (TBD)

Tools

- tee_info_hash tool
  - https://github.com/confidential-containers/td-shim/tree/main/td-shim-tools/src/bin/td-shim-tee-info-hash
- Payload reference calculator
  - https://github.com/confidential-containers/td-shim/tree/main/td-shim-tools/src/bin/td-payload-reference-calculator
- metadata checker
  - https://github.com/confidential-containers/td-shim/tree/main/td-shim-tools/src/bin/td-shim-checker

Test

- fuzzing-test: afl-fuzz, cargo-fuzz
  - https://github.com/confidential-containers/td-shim/blob/main/doc/fuzzing.md
- static code scan: cargo-clippy, rudra, Prusti, MIRAI
  - https://github.com/confidential-containers/td-shim/blob/main/doc/static_analyzer.md
- vulnerable crate scan: cargo-deny
  - https://github.com/confidential-containers/td-shim/blob/main/doc/cargo-deny.md
- general test:
  - unit test coverage: https://github.com/confidential-containers/td-shim/blob/main/doc/unit_test_coverage.md
  - no_std test: https://github.com/confidential-containers/td-shim/blob/main/doc/test_in_no_std.md
  - payload test: https://github.com/confidential-containers/td-shim/blob/main/doc/test_with_td_payload.md

Reference

- Intel ® TDX
  - https://software.intel.com/content/www/us/en/develop/articles/intel-trust-domain-extensions.html
- Virtual Firmware for Intel® Trust Domain Extensions
  - https://cfp.osfc.io/osfc2020/talk/CRKZB8/
- Enabling Rust for UEFI firmware
  - https://cfp.osfc.io/osfc2020/talk/SLFJTN/