# TD-Shim Introduction

## - A lightweight virtual firmware for confidential container

Jiewen Yao, Principal Engineer, Intel Corporation

November, 2021

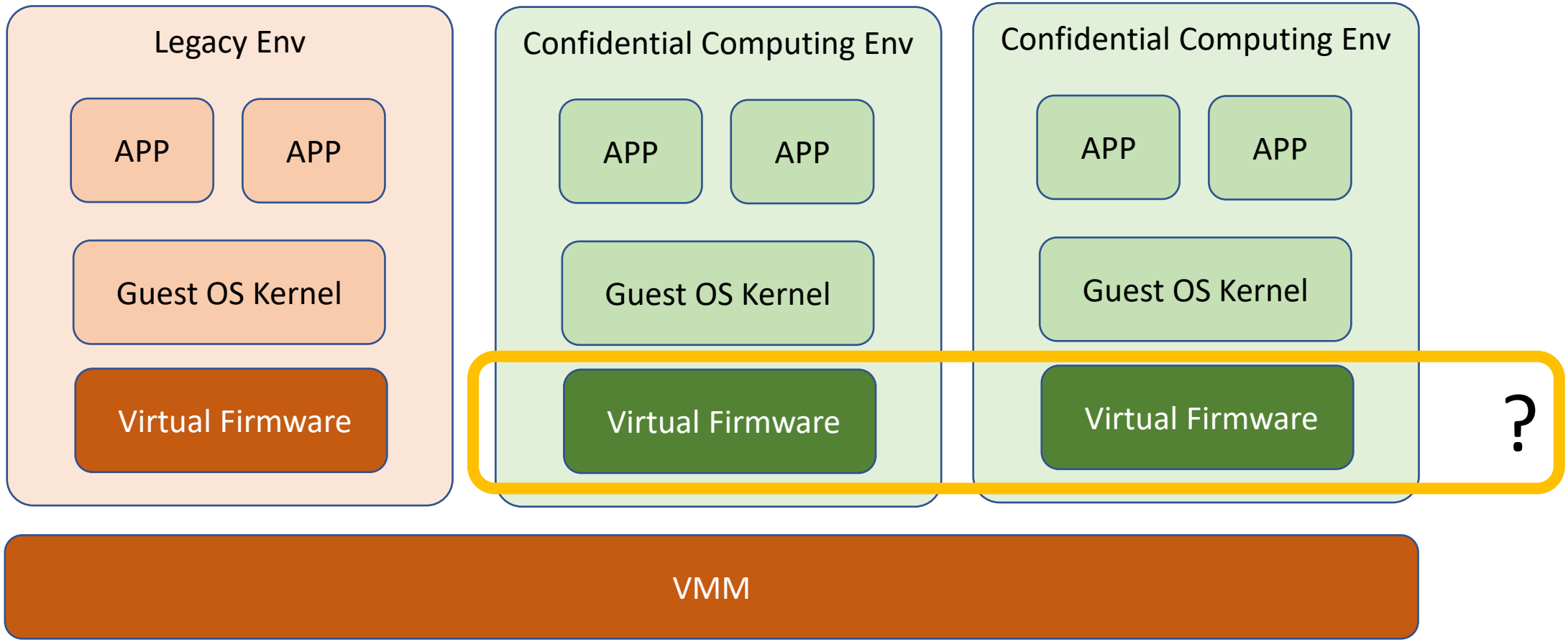
# Agenda

- Background
- Why TD-Shim?
- TD-Shim Internal

# Background

- Trend for Confidential Computing
  - AMD Secure Encrypted Virtualization (SEV)
  - Intel Trust Domain Extension (TDX)
  - ARM Realm Management Extension (RME)

# Need of virtual firmware



# Virtual firmware solution today

| 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** | | | |
| **Missing Feature (TDX)** | 1. Entrypoint – 32bit Reset Vector<br>2. MP Wakeup – special wakeup structure<br>3. Memory Initialize – memory accept before use<br>4. DMA Management – shared/private memory switch<br>5. Measurement – Runtime Measurement Register (RTMR) extend | | |
| **Main Arch Gap (TDX)** | 16 bit code | | ELF Entrypoint<br>No Equivalent Arch |

# Confidential Computing Solution (TDX)

| Main Feature | SeaBIOS ->N/A | OVMF -> TDVF | cloud-hypervisor-firmware -> Td-Shim |
|---|---|---|---|
| **Hypervisor** | N/A | KVM, cloud-hypervisor, | cloud-hypervisor, KVM, … |
| **Arch** | N/A | 64bit | 64bit |
| **VMM-BIOS Entrypoint** | N/A | Reset Vector | Reset Vector |
| **BIOS-OS Interface** | N/A | UEFI Specification,<br>ACPI Specification | Linux Boot Protocol,<br>Executable Payload Boot |
| **Feature** | N/A | Support No_CC/TDX/SEV<br>No OVMF arch change | Support TDX only (So far) |

# TD-Shim

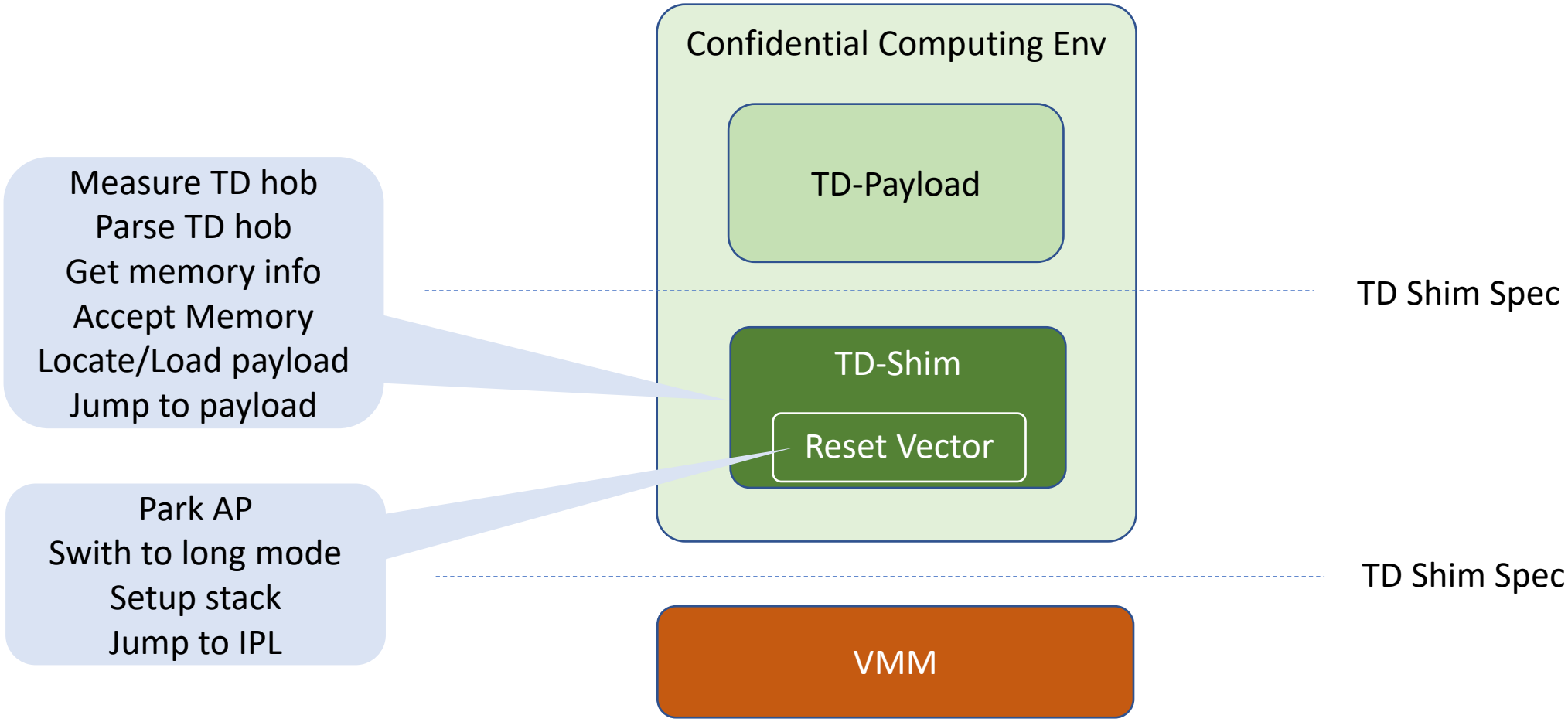
- A lightweight virtual firmware – bridge the gap between VMM and OS.
- no_std build with Rust.

- Initial version:
  - https://github.com/confidential-containers/td-shim/pull/2
  - https://github.com/jyao1/td-shim/tree/init_version
- TD-Shim Spec (0.5):
  - doc/tdshim_spec.md
- TD-Shim Design:
  - doc/design.md

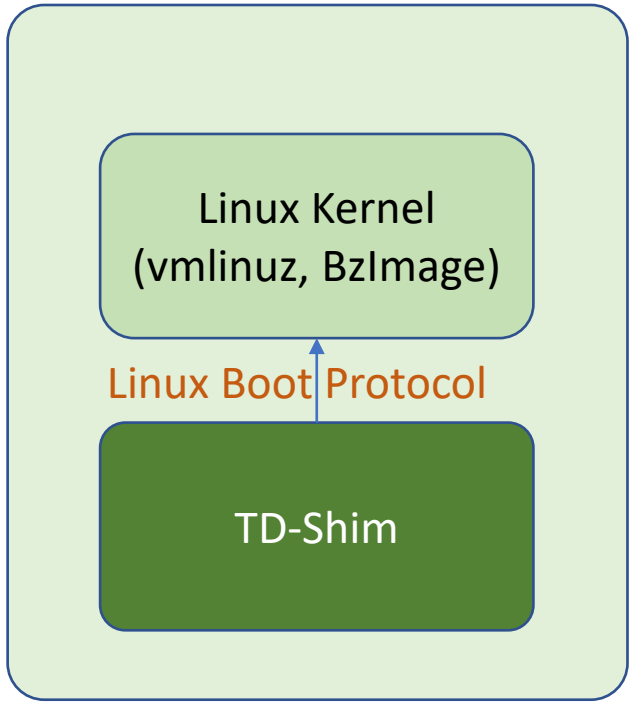
# TD-Shim v.s. TDVF(OVMF)

| | TD Shim | TDVF |
|---|---|---|
| **Reset Vector** | YES | YES |
| **UEFI Service & Adv Features** | NO | YES, 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) |
| **Defense In Depth (DEP, CET)** | Default ON | Default OFF |
| **Language** | RUST + ASM | C + ASM |
| **Image Size (release)** | 100K (w/o crypt), 1M (w/ crypt) | 4M by default. |

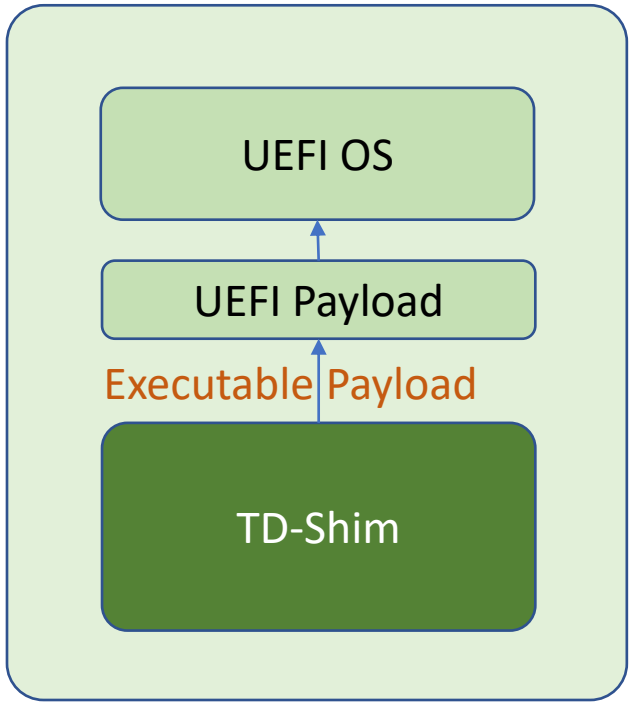
# TD-Shim


Measure TD hob
Parse TD hob
Get memory info
Accept Memory
Locate/Load payload
Jump to payload
Park AP
Swith to long mode
Setup stack
Jump to IPL
Confidential Computing Env
TD-Payload
TD-Shim
Reset Vector
VMM
TD Shim Spec
TD Shim Spec

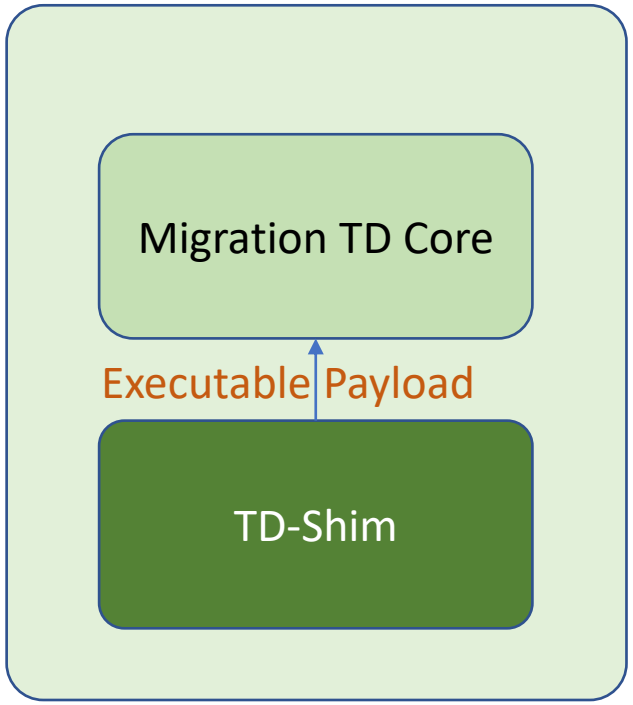
# TD-Shim Use Case




Kernel Direct Boot

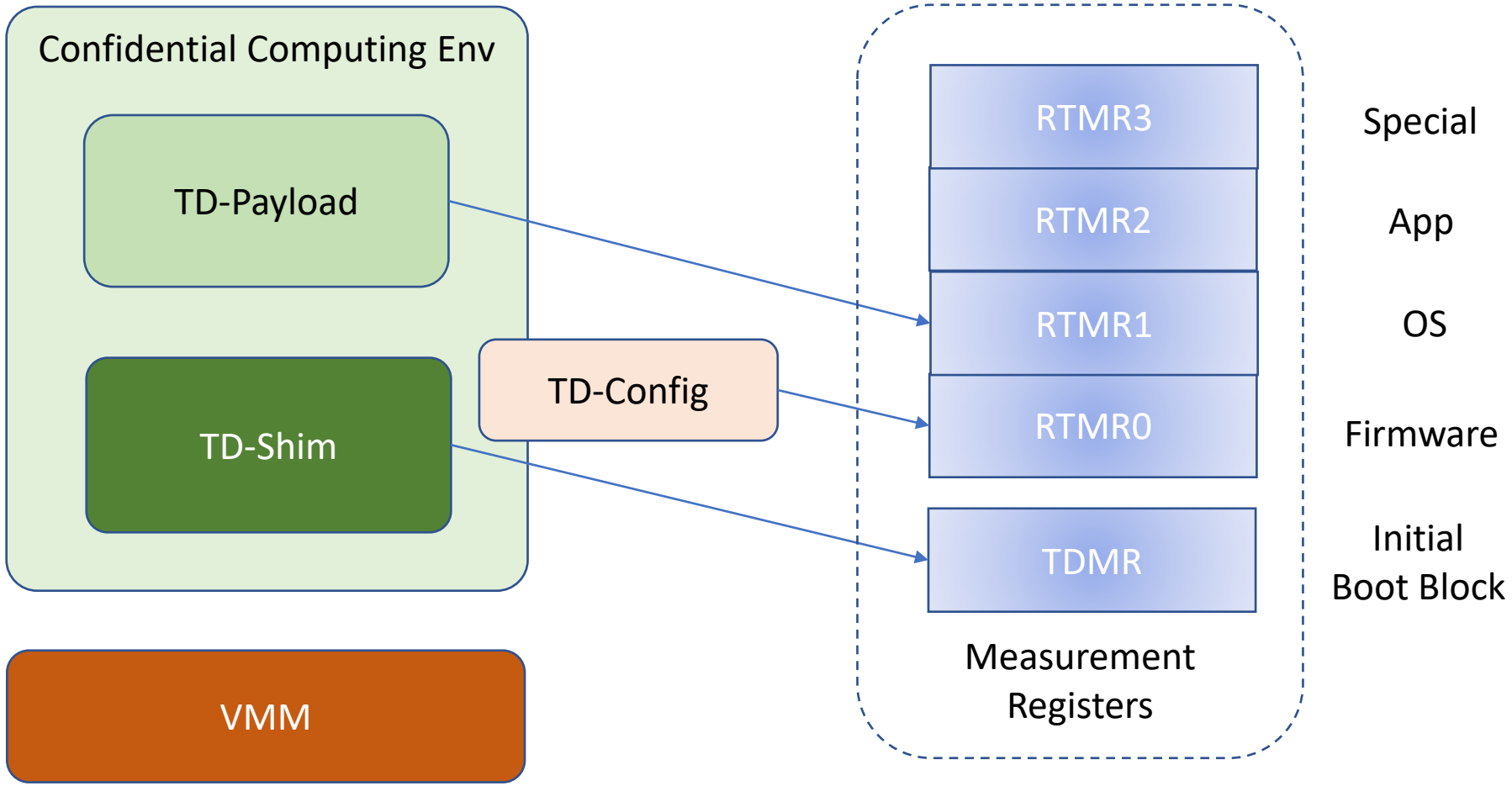



UEFI Boot




Service TD - Migration

# Feature – Trusted Boot & Attestation



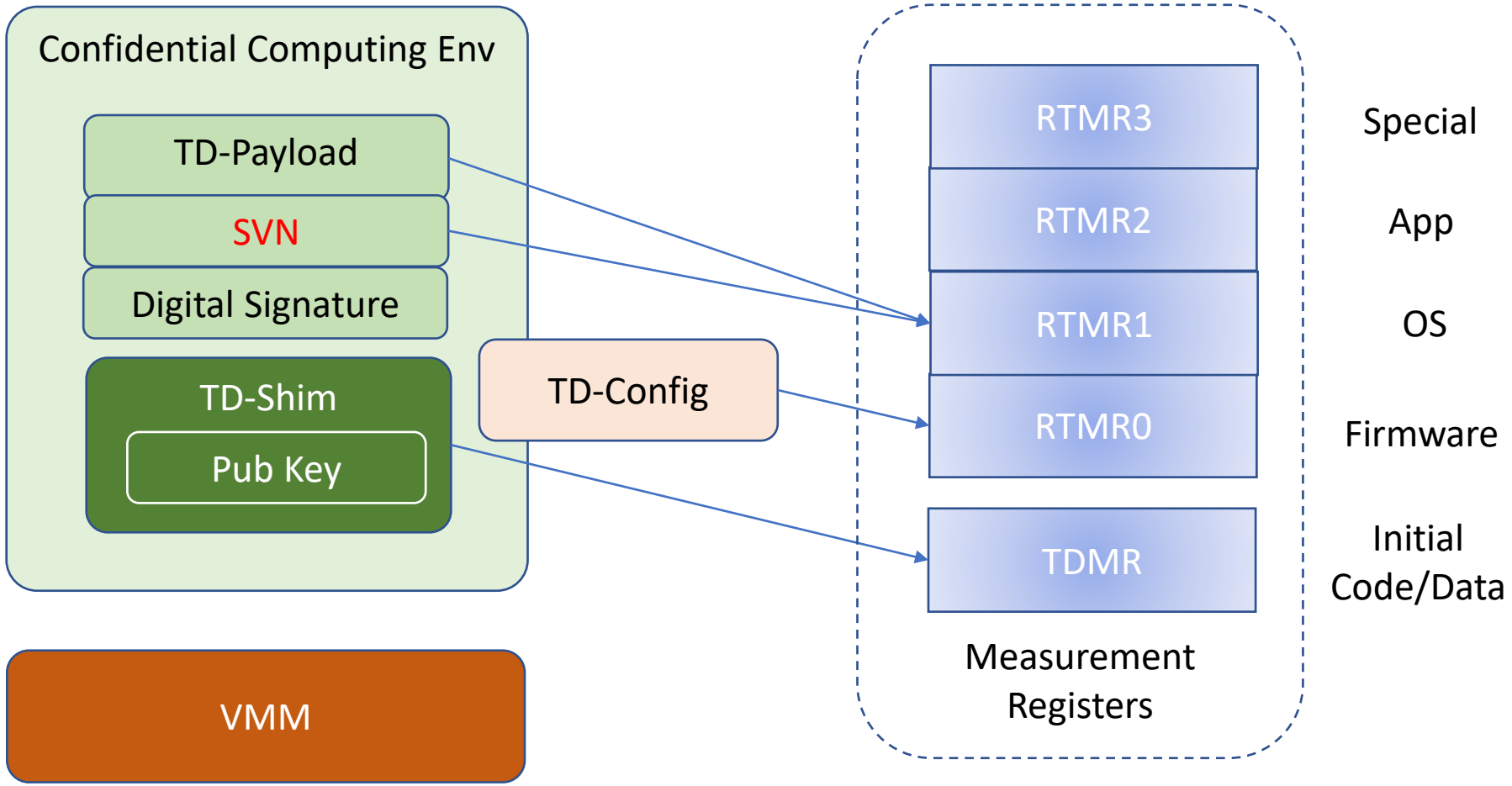
# Feature – Secure Boot (Optional)

- Use Case - Secure Version Number (SVN) based Attestation

- Problem Statement:
  - How can we do payload attestation based upon SVN, instead of measurement?
  - TD-Shim cannot guarantee the SVN is unmodified.

- Solution:
  - Sign the {payload + SVN}.
  - TD-Shim extends the SVN to RTMR, and creates a dedicate event log.
  - If TD-Shim (in MRTD) is trusted, then SVN (in RTMR) is unmodified.

# Feature – Secure Boot with Trusted Boot



# Feature – Defense in depth

- Data Execution Protection (DEP)
  - Page table based enforcement.
  - DataPage = Non-Executable
  - CodePage = Read-Only
- Control Flow Enforcement (CET)
  - Backward-Edge control flow - Shadow Stack (SS)
  - Forward-Edge control flow – Indirect Branch Tracking (IBT)
    - Depend upon compiler (TBD)
- Address Space Layout Randomization (ASLR)
  - Non-Fixed Data Address (Entropy problem, TBD)

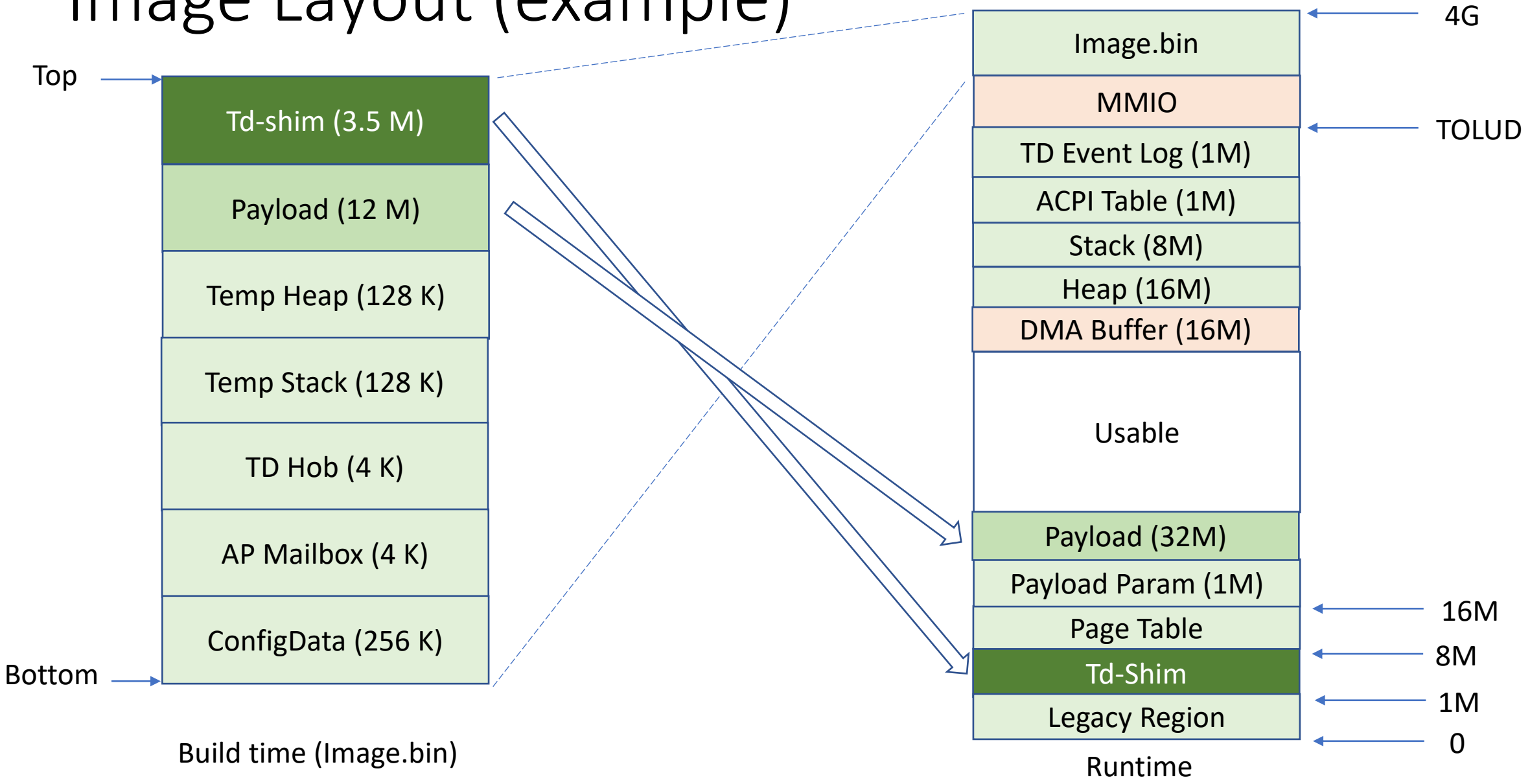
# Tool

- Everything is self-contained. (No EDKII dependency.)

- **rust-td-tool**: create final binary (reset_vector + td-shim + payload)

- **td-shim**: use x86_64-unknown-uefi (PE)
- **payload**: use x86_64-unknown-uefi (PE) or x86_64-unknown-none (ELF)

# Image Layout (example)



# Modules

- Td-shim
  - rust-tdshim
  - ResetVector
- TDX related lib
  - tdx-exception
  - tdx-logger
  - tdx-tdcall
- External Dependency
  - crypto - ring (upstream WIP)

- Refer to doc/design.md

- Generic Lib
  - elf-loader
  - pe-loader
  - rust-paging
  - r-uefi-pi (data structure)
  - uefi-pi (fv/hob parsing)
  - fw-pci (for payload)
  - fw-virtio (for payload)
  - fw-vsock (for payload)

# Test

- benchmark
  - collect benchmark information, such as stack usage, heap usage, execution time.
- fuzzing-test
  - use afl-fuzz, e.g. pe_loader. (doc/fuzzing.md)
- static code scan
  - run clippy in CI
  - use rudra  (doc/rudra.md)
- vulnerable crate scan
  - use cargo-deny (.github/workflows/deny.yml)
- unit test
  - run test in no_std environment. (doc/test_in_no_std.md)
- CI basic test
  - Run test in CI – e.g. clippy (.github/workflows)

# Future Plan

- Fix bug & improve quality
  - Improve memory accept performance
  - Feedback is welcome!
- Enable more {hypervisor, guest payload}
  - Feedback is welcome!
- Remove private dependency: crypto – ring
  - Upstream WIP - https://github.com/briansmith/ring/pull/1406
- Add std lib support for no_std env (WIP)
  - Evaluating how to add std support to uefi target in rust lang.
  - Important to use other libraries.

# Open

- Code Structure
  - Currently, it is flat.
  - Maybe we create some sub folder, to group features?

- Naming
  - https://github.com/confidential-containers/td-shim/issues/1
  - Currently, it is **td-shim**.
  - Maybe we rename it to **cc-shim** to allow other arch?