presented by

Virtual Firmware for Intel® Trust Domain Extensions

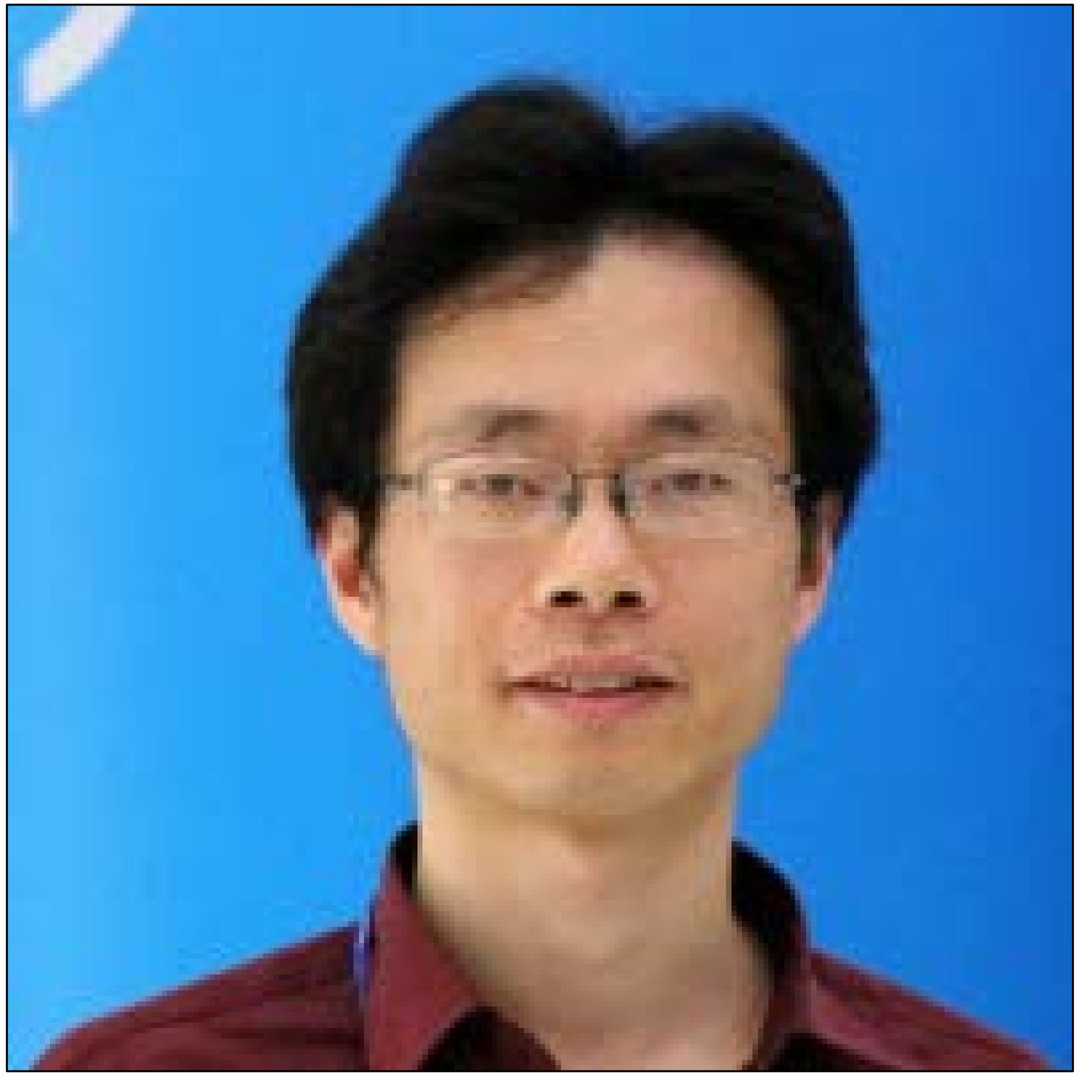
UEFI 2020 Virtual Plugfest

December 15, 2020

Jiewen Yao, Principal Engineer, Intel Corporation

Jiewen Yao

- Principal engineer in Intel Architecture, Graphics, and Software (IAGS).
- Firmware developer for over 15 years.
- Member of the UEFI Security sub team and the TCG PC Client sub working group.
- He is the architect of Intel® TDX Virtual Firmware.



Agenda

- Intel TDX
- TDVF
- TDShim
- Disk Encryption
- Summary

Intel TDX

Why Intel TDX?

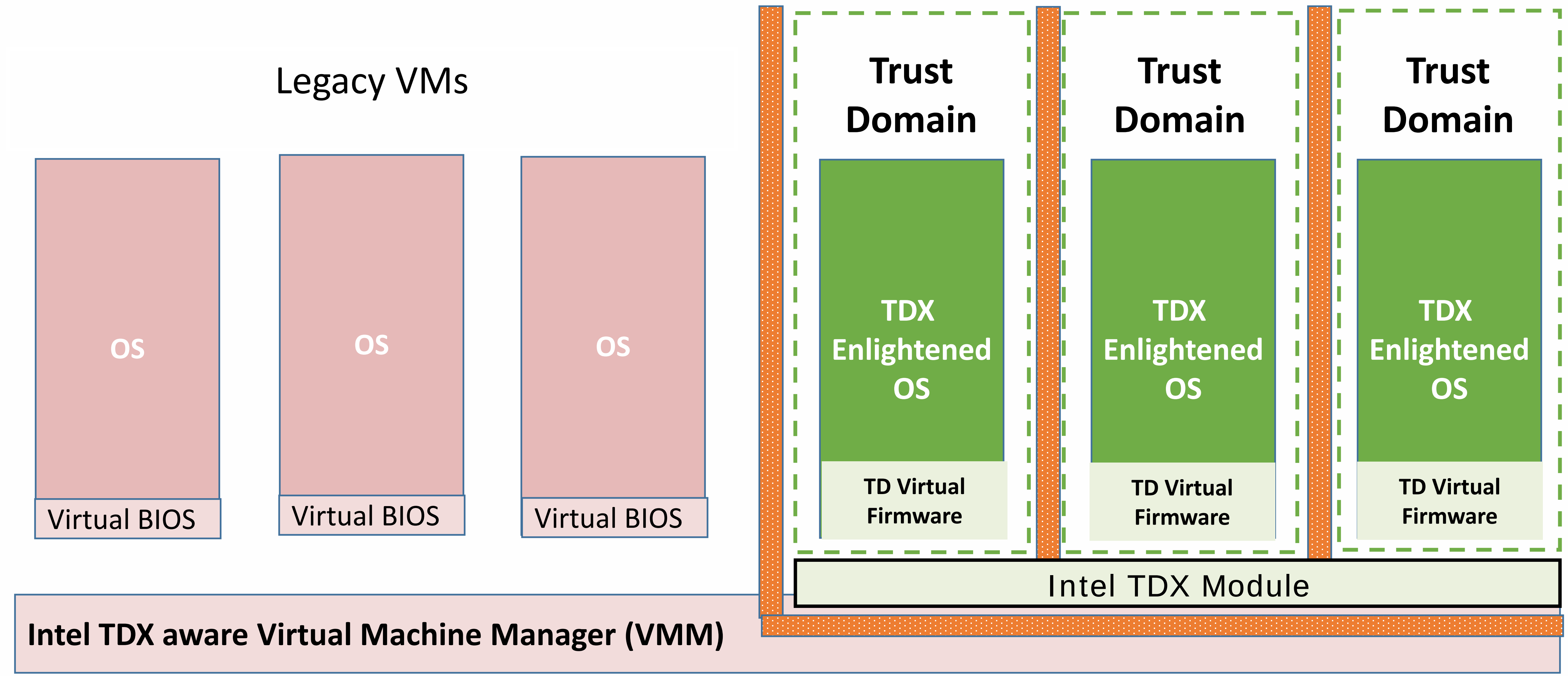
Tranditional virtual machine (VM)/hypervisor

- Hypervisor has highest privilege.
- Hypervisor may tamper VM.

What can Intel Trust Domain Extensions (TDX) offer?

- Trust Domain (TD) can resist the attack from hypervisor.
- A TD can be used for confidential computing.

Intel® Trust Domain Extensions (TDX)



Intel TDX related components

Intel TDX CPU Hardware

Intel TDX Module

- Run in Secure Arbitration Mode (SEAM), protected by SEAM range register (SEAMRR)
- Provide SEAMCALL service to a VMM and TDCALL service to a TD.
- Manage the transition between the VMM and the TD.

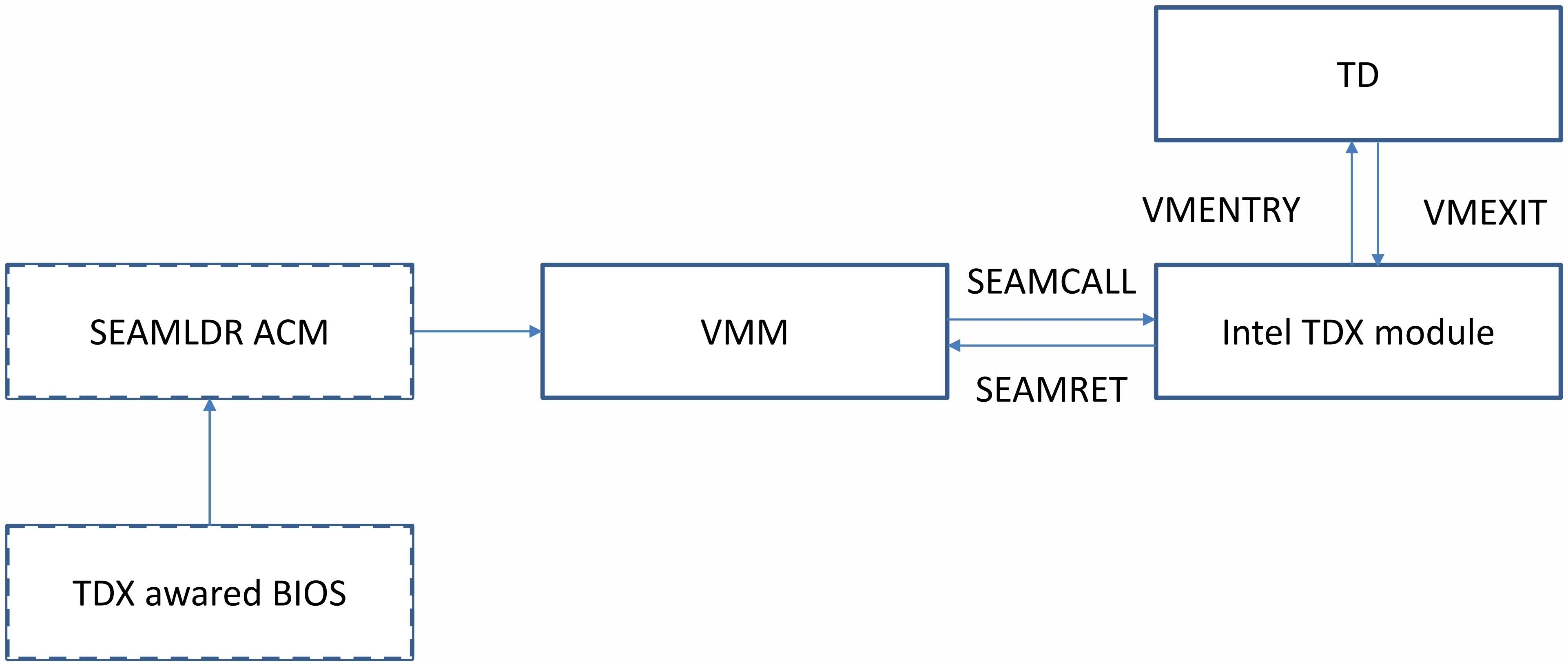
SEAM Loader (SEAMLDR) Authenticated Code Module (ACM)

- Check the hardware configuration
- Load Intel TDX Module to protected memory

TD Quoting Enclave

- Support remote attestation for a TD

System Boot Flow



TDVF

TD Virtual Firmware (TDVF)

Responsibility:

- Own 1st instruction of a trust domain (TD) at reset vector
- Provide service to a TD operating system (TD-OS)
- Build chain-of-trust from Intel TDX Module to TD-OS

Implementation:

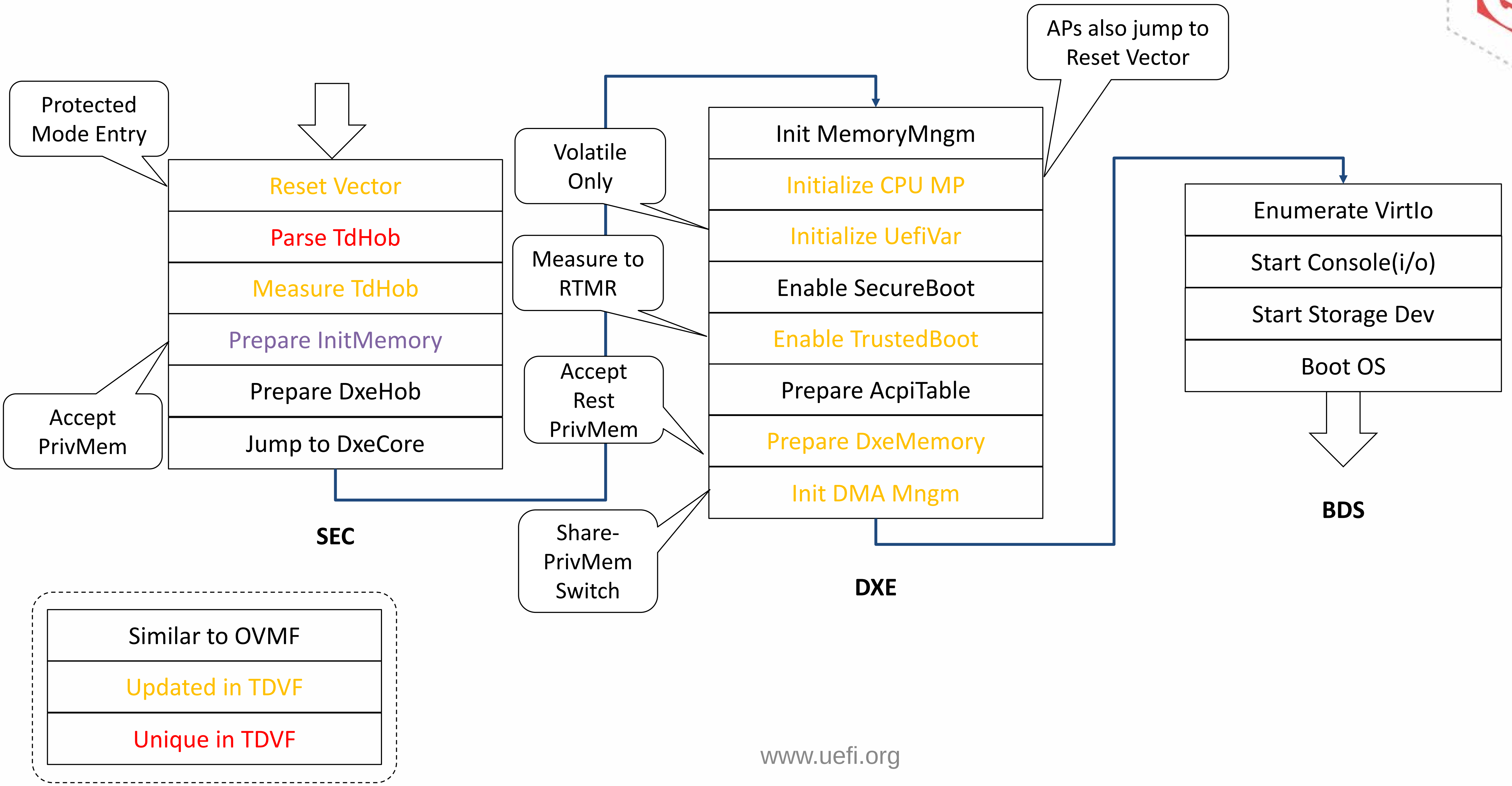
- Based upon EDK II Open Virtual Machine Firmware (OVMF)
- Simplified boot flow (no PEI phase)

Threat Model Difference

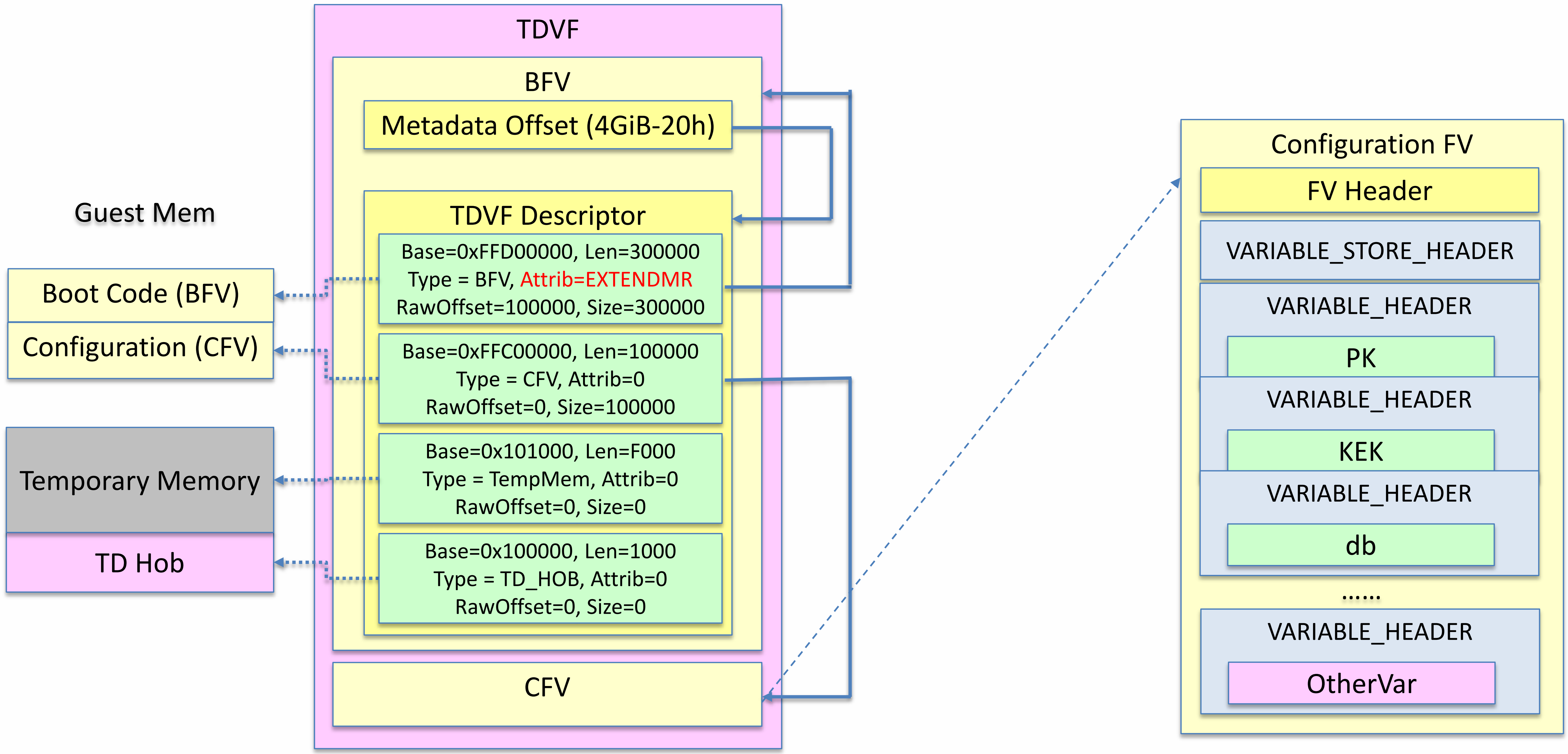
Do not trust any VMM input.

- Always validate the input before use.
- Always measure the input for remote attestation.

General Boot Flow – TDVF



TDVF Binary Layout



Launch State

Reset Vector:

- Protected mode reset vector (0xFFFFFFF0)

General Purpose Register:

- **RBX**: Guest Physical Address Width (GPAW), 48 or 52
- **RCX/R8**: hold a pointer of TD Hob. TD Hob contains the TD information, such as memory information, MMIO/IO information, which is passed from VMM.
- **RSI**: Virtual CPU (VCPU) Index (0 ~ N-1)

Launch State

- Multi Processor Support
  - All CPUs jump to reset vector at same time.
  - VCPU 0 is selected as Bootstrap Processor (BSP).
  - VCPUs (1~N-1) are Application Processors (APs), parking and waiting to be waken up by BSP.

- TDCALL[TDG.VP.INFO]
  - R8: NUM_VCPUS

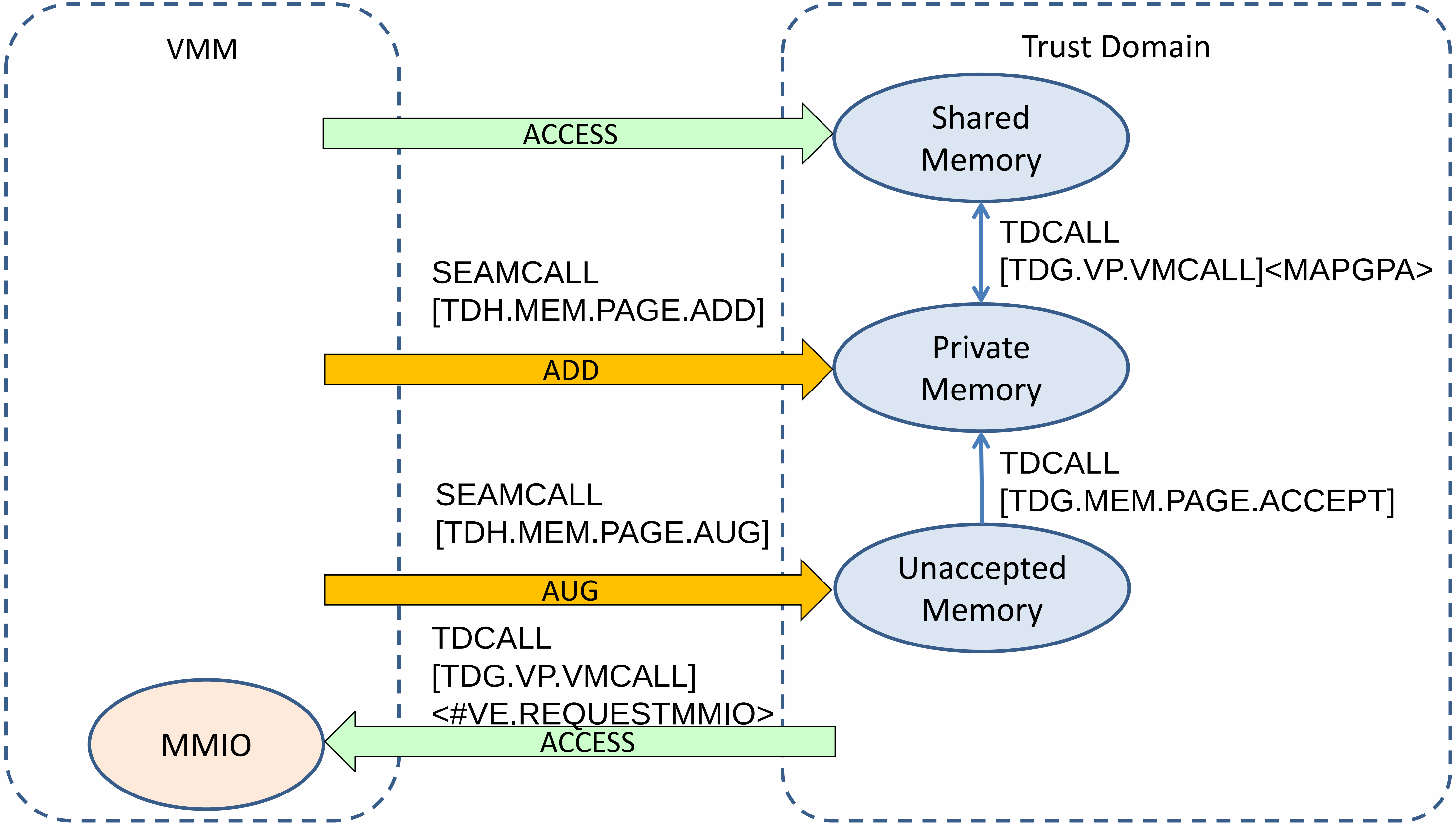
TDCALL

TDCALL	Usage	Comment
TDG.VP.VMCALL	Invoke service from the VMM	(See next page)
TDG.VP.INFO	Get TD information	GPAW, NUM_CPUS
TDG.MR.RTMR.EXTEND	Extend to TD runtime measurement register (RTMR)	SHA384 hash
TDG.VP.VEINFO.GET	Get #VE information	Exit Reason, Instruction Information
TDG.MR.REPORT	Get TD_REPORT	Measurement of the TD, TD configuration, Intel TDX module, etc.
TDG.VP.CPUIDVE.SET	Control unconditional #VE on CPUID	Supervisor mode, user mode.
TDG.MEM.PAGE.ACCEPT	Accept a pending, private page.	Guest physical address/size

TDCALL[TDG.VP.VMCALL]

TDG.VP.VMCALL	Usage
GetTdVmCallInfo	Enumerate VMCALL capabilities
MapGPA	Request VMM to map a GPA range as private or shared memory
GetQuote	Request a Quote-Enclave to sign the TD_REPORT to a TD_QUOTE
ReportFatalError	Report fatal error in TD.
SetupEventNotifyInterrupt	Request VMM specify which interrupt vector to use as an event notify vector.
Instruction.CPUID	Request VMM to emulate CPUID instruction
#VE.RequestMMIO	Request VMM to emulate the MMIO access
Instruction.HLT	Request VMM to emulate HLT instruction
Instruction.IO	Request VMM to emulate IO instruction
Instruction.RDMSR	Request VMM to emulate RDMSR instruction
Instruction.WRMSR	Request VMM to emulate WRMSR instruction
Instruction.PCONFIG	Request VMM to emulate PCONFIG instruction

Memory Management



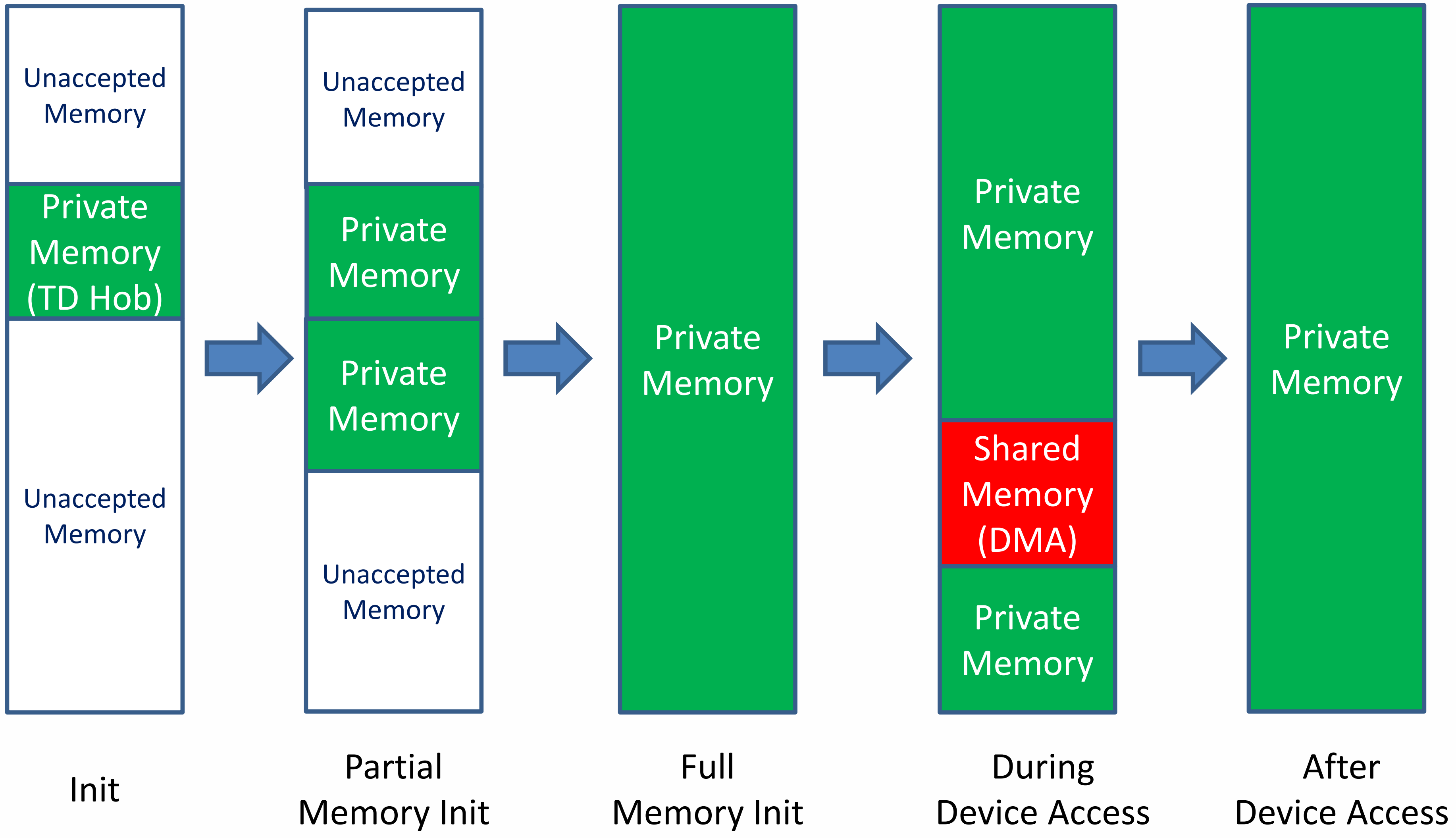
Memory Type

Type	Usage	Setup	Guest Page Table	Access
Private Memory	Default	1)**VMM:**SEAMCALL[TDH.MEM.PAGE.ADD] 2)**VMM:**SEAMCALL[TDH.MEM.PAGE.AUG] **TD:**TDCALL[TDG.MEM.PAGE.ACCEPT]	S-bit cleared	Direct Access
Shared Memory	Hypervisor communication buffer, Virtual device DMA buffer	Same as private memory	S-bit set	Direct Access
Unaccepted Memory	Private memory, not accepted yet.	**VMM:**SEAMCALL[TDH.MEM.PAGE.AUG]	N/A	N/A
MMIO	MMIO emulation	N/A	N/A	TDCALL[TDG.VP.VMCALL]<#VE.REQUEST MMIO>

Memory State Transition

Transition	Usage	Action
Unaccepted -> Private	Lazy memory init	TDCALL[TDG.MEM.PAGE.ACCEPT]
Private -> Shared	Communication buffer setup	Set S-bit in page table. TDCALL[TDG.VP.VMCALL]<MAPGPA>
Shared -> Private	Communication buffer reclaim	Clear S-bit in page table. TDCALL[TDG.VP.VMCALL]<MAPGPA> TDCALL[TDG.MEM.PAGE.ACCEPT]

Memory State Transition



UEFI/PI Memory Indicator

Type	UEFI Memory Map	PI GCD	PI Hob	ACPI E820	ACPI ASL
Private Memory	Normal UEFI Memory Type	EfiGcdMemoryTypeSystemMemory	EFI_RESOURCE_SYSTEM_MEMORY (EFI_RESOURCE_ATTRIBUYTE_ENCRYPTED)	Normal Memory Range	N/A
Shared Memory	Normal UEFI Memory Type	EfiGcdMemoryTypeSystemMemory	EFI_RESOURCE_SYSTEM_MEMORY	Normal Memory Range	N/A
Unaccepted Memory	EfiUnacceptedMemory (*)	EfiGcdMemoryTypeUnaccepted (*)	EFI_RESOURCE_SYSTEM_MEMORY (EFI_RESOURCE_ATTRIBUYTE_UNACCEPTED) (*)	AddressRangeUnaccepted (*)	N/A
MMIO	N/A	EfiGcdMemoryTypeMemoryMappedIo	EFI_RESOURCE_MEMORY_MAPPED_IO	N/A	Memory

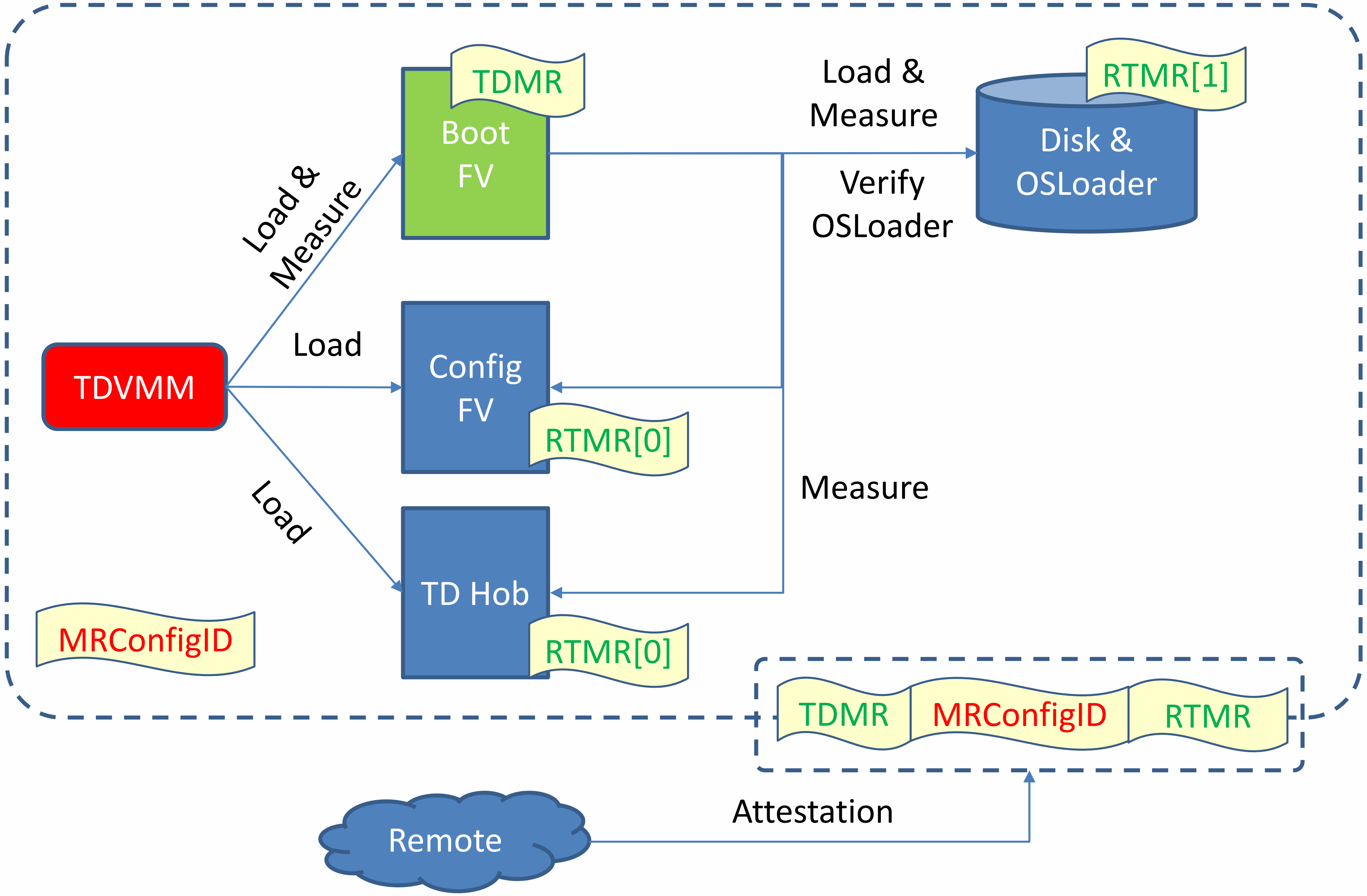
ACPI – MP Wakeup

- AP init in OS
  - All APs are reported via MADT ACPI table.
  - A new MPWK structure is defined to describe a 4KiB mailbox.
    - APs loop to check the vector in the mailbox.
    - OS fills the AP Wakeup vector, then AP jumps to the Wakeup vector.

```
typedef struct {
  UINT8       Type;
  UINT8       Length;
  UINT16      MailBoxVersion;
  UINT32      Reserved2;
  UINT64      MailBoxAddress;
} ACPI_MADT_MPWK_STRUCT;

typedef struct {
  UINT16      Command;
  UINT16      Reserved;
  UINT32      ApicId;
  UINT64      WakeupVector;
  UINT8       OsReserved[SIZT_TO_2K];
  UINT8       FirmwareReserved[SIZE_TO_4K];
} ACPI_MPWK_MAIL_BOX;
```

TD Trusted Boot



TD Trusted Boot (TDMR + 4 RTMR)

PCR	Typical Usage	TD Register	TD Reg Index	Extended by	Comment
0	Firmware Code	TDMR	0	**VMM**:SEAMCALL[TDH.MR.EXTEND]	VF code (BFV, initial page table)
1	Firmware Data	RTMR [0]	1	**TDVF**:TDCALL[TDG.MR.RTMR.EXTEND]	Dynamic Configuration Data (TD HOB, ACPI) Data from FW_CFG_IO_SELECTOR/ FW_CFG_IO_DATA
2	Option ROM code	N/A		N/A	
3	Option ROM data	N/A		N/A	
4	OS loader code	RTMR [1]	2	**TDVF**:TDCALL[TDG.MR.RTMR.EXTEND]	OS loader
5	Boot Configuration	RTMR [1]	2	**TDVF**:TDCALL[TDG.MR.RTMR.EXTEND]	GPT, Boot Variable
6	N/A	N/A		N/A	
7	Secure Boot Configuration	RTMR [0]	1	**TDVF**:TDCALL[TDG.MR.RTMR.EXTEND]	SecureBootConfig (CFV)
	TD OS App measurement	RTMR [2]	3	**TDOS**:TDCALL[TDG.MR.RTMR.EXTEND]	TD OS App. Done by TD OS.

TD Trusted Boot Interface

- EFI_TD_PROTOCOL
  - Similar to TCG: EFI_TCG2_PROTOCOL
  - Provide measurement services for OS loader and OS kernel
- TD EventLog 'TDEL' ACPI table
  - Similar to TCG: TPM2 ACPI table
  - Provide TD Event Log – similar to TCG2 event log

UEFI Secure Boot

- Image Verification is same
- UEFI Auth Variable
  - *Volatile only*
  - All keys be provisioned at TD build time
  - All keys are measured into RTMR

TDVF Design Principle

- Keep it Simple
  - Do remove unncessarey feature (e.g. PEI)
  - Don’t expose unrequired external interface (e.g. network)
- Apply best Security Practice
  - Do validate all input before use.
  - Don’t trust VMM (new threat model)
- Build Chain-of-Trust
  - Do measure all input before use.
  - Don’t use mutable non-volatile storage (causing MR change)

TD Shim

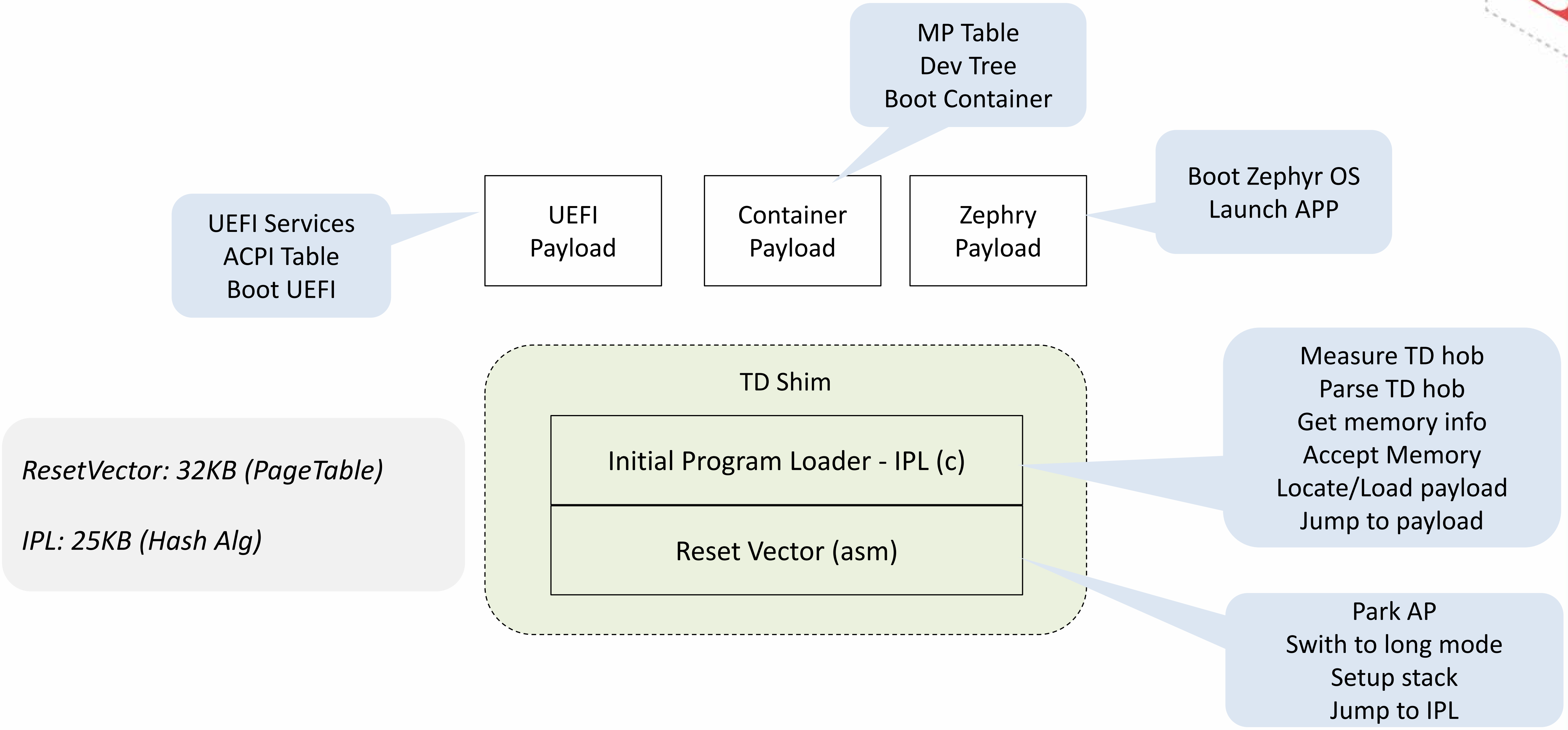
TD Shim - Motivation

- Non-UEFI OS
  - No UEFI services required.
- Container OS
  - Special OS interface. (no UEFI)
- A small service TD
  - Bare metal environment. (no UEFI)

TD Shim – A tiny TDVF

	TDVF	TD Shim
Reset Vector	YES	YES
SEC (Initial Program loader - IPL)	IPL to boot a UEFI Core	IPL to boot a payload (UEFI, Container, Zephyr, etc)
UEFI Core	UEFI Services	NO
Device Driver	Virtio, PCI, etc	NO
ACPI Table (MultiProcessor)	MADT / DSDT	Static ACPI table only. Or MP table extension
Memory Map	UEFI Memory Map	E820 table
Trusted Boot	TD Measurement + TD Event Log (ACPI)	TD Measurement + TD Event Log Table
Secure Boot	UEFI Secure Boot	NO

TD Shim - Boot Flow

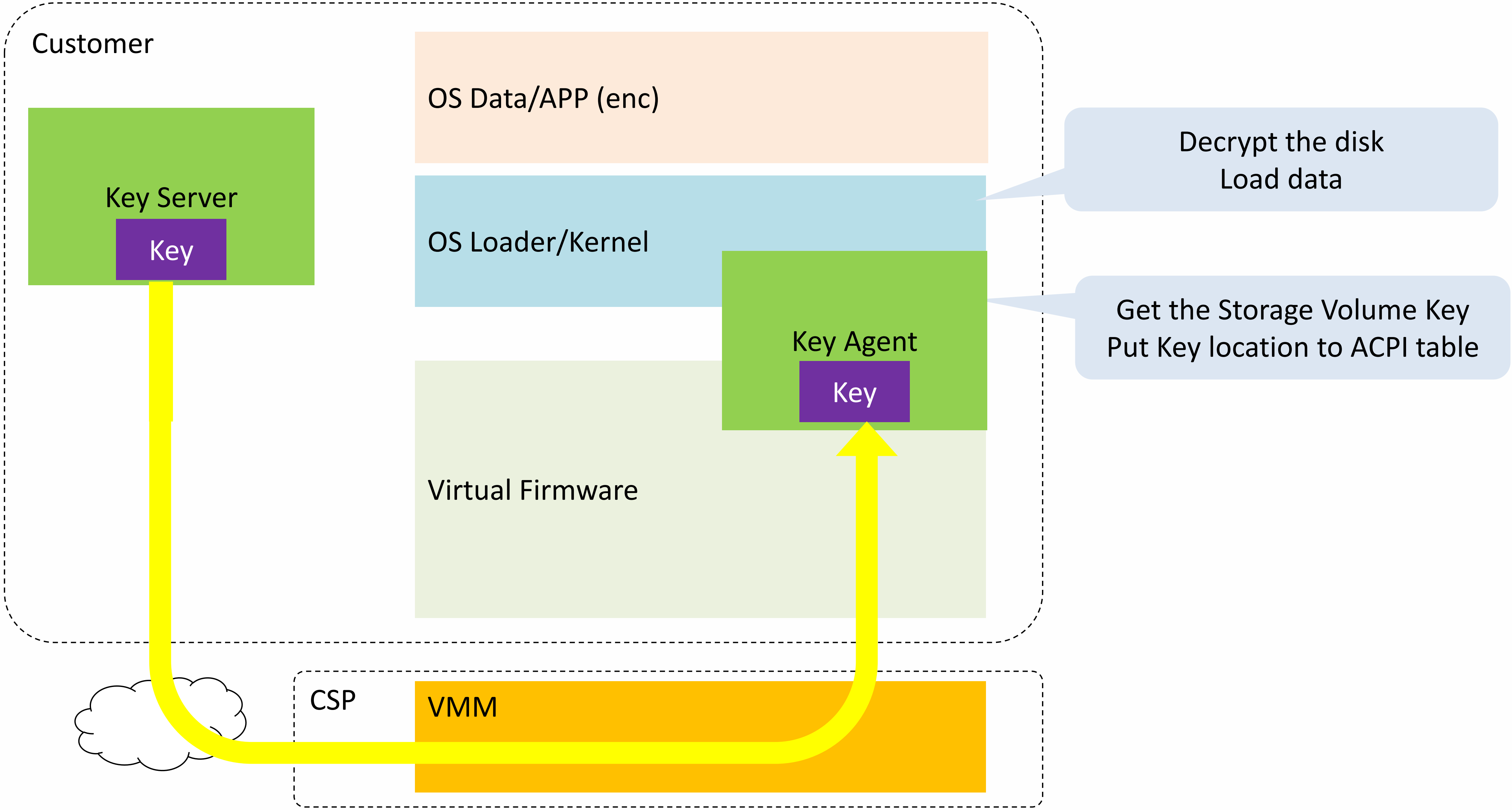


Disk Encryption

Storage Volume Key

- Today, the VMM/OS gets the “storage volume key” and decrypts the VM disk.
- In TDX, the VMM/OS is not trusted.
- The TD need get the “storage volume key” and decrypt the VM disk.

High Level Component Layout

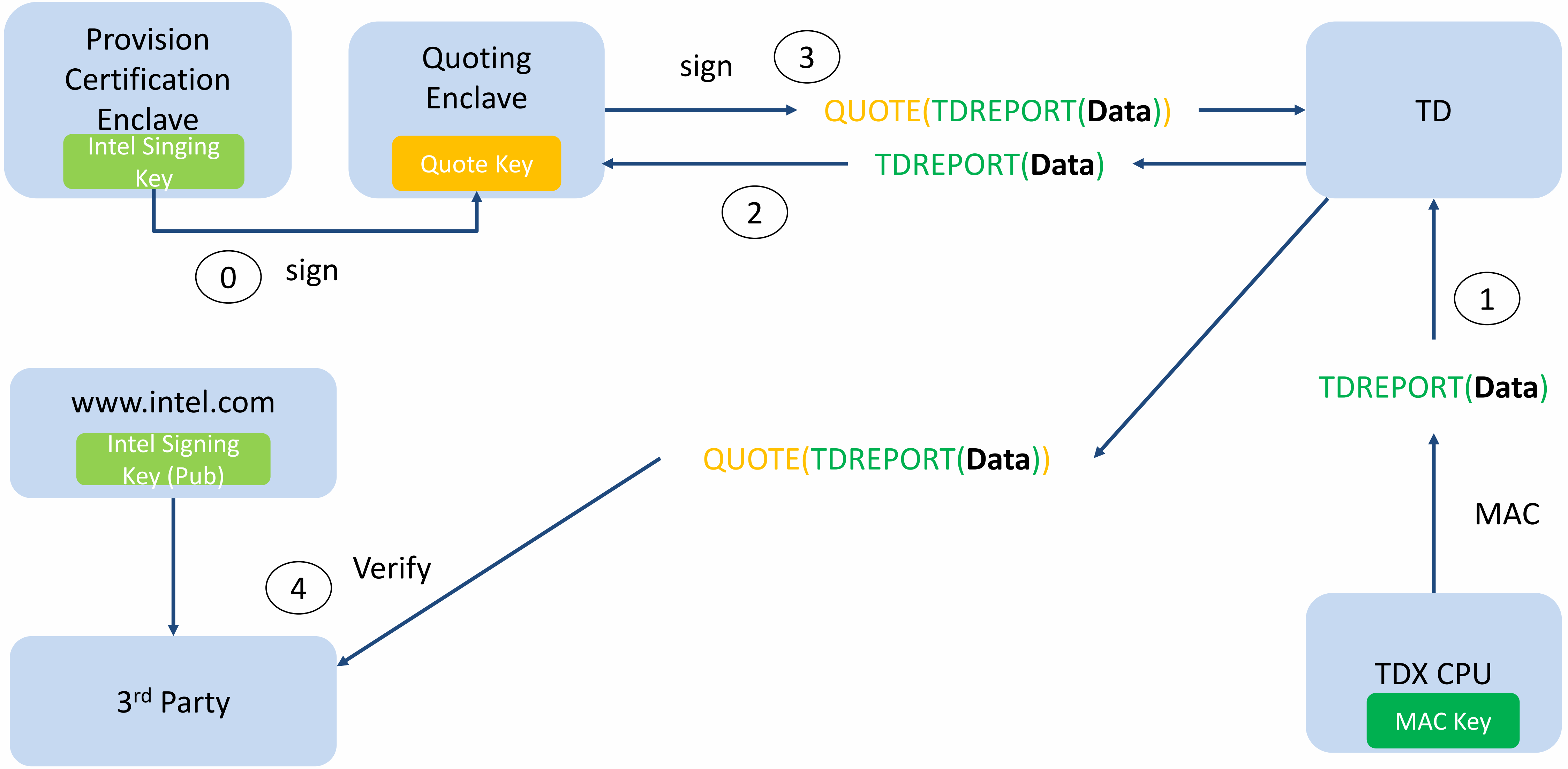


Secure Communication Channel

- Confidentiality/Integrity:
  - Transport Layer Security (TLS)

- Mutual Authentication
  - Client TD -> Key Server: Server certificate verification
  - Key Server -> Client TD: TD Attestation

TD Attestation



Disk Encryption Key Passing

Remote Key Server -> KeyAgent in TD

- TLS + mutual authentication

Key Agent in TD -> OS loader/kernel in TD

- Storage Volume Key Location ‘SVKL’ ACPI Table

Key Agent Location (Use case specific)

- TDVF – PROs: A common OS
- OS – PROs: Keep TDVF simple

Summary

Summary

Intel® TDX

- Supports memory and CPU state confidentiality and integrity
- Supports measurement and remote attestation

TD Virtual Firmware (TDVF)

- TDVF: Build the chain of trust & Launch a TD-OS
- TDShim: A tiny TD to boot a payload.

Disk Encryption

- KeyAgent: Get disk encryption key from remote key after TD attestation
- OsLoader/Kernel: Get key from KeyAgent and decrypt the disk

Reference

Intel® TDX Specification and Whitepaper

- Intel® Trust Domain Extensions (Intel® TDX)
- Intel® TDX Virtual Firmware Design Guide
- Intel® TDX Guest-Hypervisor Communication Interface

TDVF Pre-Production Code

- TdvfPkg at TianoCore (edk2-staging)
- TdShimPkg POC

Questions?

More Questions?

Following today's webinar, join the live, interactive WebEx Q&A for the opportunity to chat with the presenter

Visit this link to attend: https://bit.ly/3oW5SdD
Meeting number (access code): 126 544 0541
Meeting password: q2TPMRqMw36 (72876776 from phones and video systems)

Thanks for attending the UEFI 2020 Virtual Plugfest

For more information on UEFI Forum and UEFI Specifications, visit http://www.uefi.org

presented by

intel