

Intel® TDX Virtual Firmware Design Guide

December 2023



You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.

Intel technologies may require enabled hardware, specific software, or services activation. Check with your system manufacturer or retailer.

The products described may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps

Copies of documents that have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or visit www.intel.com/design/literature.htm.

Intel, the Intel logo, are trademarks of Intel Corporation in the USA and/or other countries.

*Other names and brands may be claimed as the property of others.

© 2020 Intel Corporation. All rights reserved.

Contents

1	Introduction	7
1.1	Background.....	7
1.2	Overview	7
1.3	Terminology	8
2	Architectural Overview	11
2.1	TDVF Features.....	11
2.2	TD Hardware	11
2.3	Boot Flow.....	12
2.3.1	VMM Setup Phase	12
2.3.2	TDVF Launch Phase	12
2.3.3	TDVF OS Boot Phase	13
2.4	TDVF Requirements.....	13
2.5	Reproducibility.....	14
2.5.1	Challenge	14
2.5.2	Solution.....	14
2.6	Security Considerations	15
2.6.1	General Security Practice.....	15
2.6.2	Side Channel Security Practice	16
2.6.3	Confidential Computing Virtual Firmware Security Practice.....	16
2.6.4	TDX Specific Security Practice	17
3	TDVF Binary Image	19
3.1	Boot Firmware Volume (BFV)	19
3.2	Configuration Firmware Volume (CFV)	19
4	TD Launch	21
4.1	TDVF initialization	21
4.1.1	VCPU Init State	21
4.1.2	System Information	21
4.1.3	Long Mode Transition	21
4.1.4	Setup stack to call C function	22
4.1.5	Switch to UEFI environment.....	22
4.2	TD Hand-Off Block (HOB).....	23
4.2.1	PHIT HOB	23
4.2.2	Resource Description HOB	23
4.2.3	CPU HOB	24
4.2.4	GUID Extension HOB.....	24
4.3	TDVF AP handling	24
4.3.1	AP Init State	24
4.3.2	AP Information Reporting from VMM to TDVF	24
4.3.3	AP initialization in TDVF.....	25
4.3.4	AP information reporting from TDVF to OS	26
4.3.5	AP initialization in OS	26
5	TDVF UEFI Secure Boot Support	29
5.1	Provisioning UEFI Secure Boot.....	29
5.2	Variable Driver	29
6	TDVF ACPI Support.....	30

6.1	Source of ACPI Tables	30
6.2	ACPI Support	30
6.3	FADT	30
6.4	DSDT	30
6.5	FACS	30
6.6	MADT	30
7	TD Memory Management	32
7.1	Memory Type	32
7.1.1	Private Memory Indicator in Guest Page Table	32
7.2	Initial State from VMM	32
7.2.1	Memory Type in TD Resource HOB	33
7.3	Memory Information for DXE Core	35
7.3.1	Memory Type in DXE Resource HOB	37
7.4	Memory Map to TD-OS	38
7.5	Convert Shared to Private	38
7.6	Convert Private to Shared	39
7.7	Memory State Transition	39
7.8	Optimization Consideration	40
7.8.1	Partial Memory Initialization in Pre-UEFI	40
7.8.2	Partial Memory Initialization in UEFI	40
7.8.3	Parallelized Memory Initialization	41
7.8.4	Pre-allocating Virtual Device IO Buffer	41
8	TD Measurement	42
8.1	Measurement Register Usage in TD	42
8.2	Fundamental Support	43
8.3	Virtual Firmware Configuration	45
8.3.1	Build-Time Configuration	45
8.3.2	Launch-Time Configuration	45
8.3.3	Runtime Configuration	45
8.3.4	Hypervisor Specific Configuration Interface	45
8.4	Attestation and Quote Support	46
9	TDVF Device Support	48
9.1	Minimal Requirement	48
9.2	VirtIo Requirement	48
9.3	Security Device	48
9.4	HotPlug Device	49
9.5	PCI Device Option ROM	49
10	Exception Handling	50
10.1	Virtualization Exception (#VE)	50
10.2	Instruction Conversion	50
11	TDVF Metadata	51
11.1	TDVF Metadata Location	51
11.2	TDVF descriptor	51
12	OS Direct Boot	56
12.1	Measurement	56
13	Minimal TDVF (TD-Shim) Requirements	58

13.1	Hardware Virtualization-based Containers	58
13.1.1	TD Container Requirements	58
13.2	TD-Shim Launch	58
13.2.1	TD-Shim AP Handling	59
13.3	TD-Shim Secure Boot Support	59
13.4	TD-Shim ACPI Support	59
13.5	TD-Shim Memory Management	59
13.5.1	Memory Type in Initialization	59
13.5.2	Memory Map for OS	60
13.6	TD-Shim Measurement	60
13.6.1	TD Measurement	60
13.6.2	TD Event Log	61
13.7	TD-Shim Device Support.....	61
13.8	TD-Shim Exception Handling	61
14	Disk Encryption	63
14.1	Overview	63
14.2	Attestation Agent.....	65
14.2.1	Network Communication.....	65
14.2.2	Authenticated Secure Session	65
14.2.3	Key Server Information	65
14.2.4	Key transport from Server	66
14.3	Decryption Agent.....	66
14.3.1	Key Passing – SVKL table	66
14.3.2	Storage Volume Key Usage in TDVF	66
14.4	High-level Flow Examples	66
14.4.1	Example on Full Disk Encryption	66
14.4.2	Example on Data Partition Encryption.....	67
Appendix A	- References	68

Figures

Figure 1-1:	Intel TDX Architecture	8
Figure 3-1:	TDVF Configuration Firmware Volume.....	20
Figure 4-1:	TDVF General Flow	23
Figure 7-1:	TD Hob and Initial Memory Layout	33
Figure 7-2:	DXE HOB and Runtime Memory Layout	37
Figure 7-3:	TDVF Memory State Transition.....	39
Figure 8-1:	TDVF Measurement	44
Figure 8-2:	TDVF Measurement Interface.....	45
Figure 8-3:	Attestation and Quote.....	46
Figure 11-1:	TDVF Metadata Layout.....	51
Figure 14-1:	Preboot Disk Decryption Flow.....	64
Figure 14-2:	Early-boot Disk Decryption Flow.....	64

Tables

Table 1-1: Differences between VMX and TDX	8
Table 1-2: Terminology	8
Table 7-1: Memory Type in TD Resource HOB	33
Table 7-2: TDVF memory state from VMM.....	34
Table 7-3: Memory Type in DXE resource HOB	37
Table 8-1: TD Measurement-related Register	42
Table 11-1: TDVF_DESCRIPTOR definition	52
Table 11-2: TDVF_SECTION definition	52
Table 11-3: TDVF_DESCRIPTOR.Type definition	53
Table 11-4: TDVF_DESCRIPTOR.Attributes definition	53
Table 11-5: TD_INFO definition	54
Table 12-1: OS loader measurement.....	56
Table 13-1: TD-Shim memory state from VMM.....	59
Table 13-2: TD Measurement-Related Registers for TD-Shim	60
Table 14-1: Disk Encryption Use Cases.....	63
Table 14-2: Security Property Example.....	64

1 Introduction

1.1 Background

Intel Total Memory Encryption (TME) engine encrypts the platform's entire memory with a single key and provides the ability to specify use of a specific key for a page of memory. The Multi-Key mode of TME (MK-TME) extends TME to support multiple encryption keys. In a virtualization scenario, a Virtual Machine Manager (VMM) or hypervisor will manage keys to transparently support legacy operating systems without any changes (thus, MKTME can also be viewed as TME virtualization in these scenarios). An operating system (OS) may take advantage of MKTME capabilities in a native or virtualized environment. When properly enabled, MKTME is available to each guest OS in a virtualized environment, so both native and guest OS can take advantage of MKTME. In these usages, the VMM is in the Trust Computing Base (TCB).

Intel Trust Domain Extensions (Intel TDX) refers to an Intel technology that extends Virtual Machines Extensions (VMX) and Multi-Key Total Memory Encryption (MKTME) with a new kind of virtual machine guest called a Trust Domain (TD). A TD runs in a CPU mode that protects the confidentiality of its memory contents and its CPU state from any other software, including the hosting Virtual Machine Monitor (VMM), unless explicitly shared by the TD itself.

The TDX solution is built using a combination of Intel® Virtual Machine Extensions (VMX) and Multi-Key Total Memory Encryption (MK-TME), as extended by the Intel® Trust Domain Instruction Set Architecture Extensions (TDX ISA). An attested software module called The Intel TDX module implements the TDX architecture.

The platform is managed by a TDX-aware host VMM. A host VMM can launch and manage both guest TDs and legacy guest VMs. The host VMM maintains all legacy functionality from the legacy VMs perspective; it's restricted only with regard to the TDs it manages.

In Summary, Intel TDX:

- 1) Removes the cloud host software and devices from the Trust Computing Base (TCB) of cloud (TD) tenants.
- 2) Provides memory encryption and integrity multi-tenancy for hardware attack protection.
- 3) Supports TD measurement for attestation to a relying party.

1.2 Overview

Intel® Trust Domain Extensions (Intel® TDX) provides the capabilities required to enable an TDX-aware VMM to manage the lifecycle of a TD.

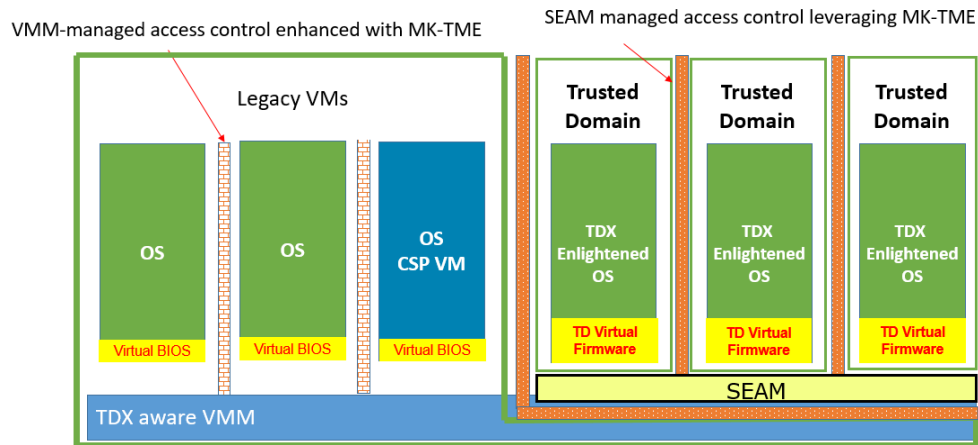


Figure 1-1: Intel TDX Architecture

Fundamental differences between [Virtual Machine Extension](#) (VMX) and Intel TDX are outlined in the following table.

Table 1-1: Differences between VMX and TDX

	Access to Guest State	Transitions for State Save & Restore	Controls
VMX	VMM has full access	VMM SW augments VMX transitions (e.g. general-purpose registers)	VMM has a rich set of controls to interact with the VM in order to de-privilege the VM
TDX	VMM has no direct access; TD may volunteer info	State transitions completed are managed by Intel TDX Module in Secure-Arbitration Mode (SEAM)	VMM has a limited set of controls for resource management

Trust Domain Virtual Firmware (TDVF) is required to provide TD services to the TD Guest OS. This document describes the TDVF architecture.

TDVF reference code is integrated to TianoCore open source repository: <https://github.com/tianocore/edk2/tree/master/OvmfPkg>. A minimal TDVF (td-shim) reference code is at confidential container repository: <https://github.com/confidential-containers/td-shim>.

1.3 Terminology

Table 1-2: Terminology

Term	Description
ACPI	Advanced Configuration and Power Interface

AP	Application Processor
APIC	Advanced Programmable Interrupt Controller
BFV	Boot Firmware Volume
BSP	Bootstrap Processor
CFV	Configuration Firmware Volume
CSM	Compatibility Support Module
DXE	Driver Execution Environment
EPT	Extended Page Table
GPA	Guest Physical Address
GPAW	Guest Physical Address Width
HOB	Hand Off Block
HPA	Host Physical Address
MADT	Multiple APIC Description Table
MKTME	Multi-Key Total Memory Encryption
MPWK	Multiple Processor Wakeup
MRCONFIGID	Measurement Register of configuration data
MRTD	TD Measurement Register
OVMF	Open Virtual Machine Firmware
PEI	Pre-EFI Initialization
RA-TLS	Remote Attestation-TLS
RTMR	Runtime Extendable Measurement Register
SEAM	Secure-Arbitration Mode
SEC	Security Phase
SMM	System Management Mode
SMX	Safer Mode Extensions
TCB	Trust Computing Base
TCG	Trusted Computing Group
TD	Trust Domain
TDVF	Trust Domain Virtual Firmware
TDX	(Intel) Trust Domain Extensions
TLS	Transport Layer Security
TME	Total Memory Encryption
UEFI	Unified Extensible Firmware Interface
VCPU	Virtual CPU
VE	Virtualization Exception
VM	Virtual Machine

VMCS	Virtual Machine Control State
VMM	Virtual Machine Monitor
VMX	Virtual Machine Extension

2 Architectural Overview

2.1 TDVF Features

TDVF has the following features:

- 1) Use Unified Extensible Firmware Interface (UEFI) Secure Boot as base with extensions for TD launch.
- 2) Use Trusted Computing Group (TCG) Trusted Boot to perform a measured and verified launch of a guest OS loader or kernel.
- 3) Simplify firmware by removing features found in traditional UEFI implementations:
 - a) SEC, PEI, SMM (DXE Only)
 - b) CSM (UEFI Class 3 OS only)
 - c) Setup UI
 - d) Recovery
 - e) Capsule-based Firmware Update
 - f) ACPI S3 (not supported by TDX guests)

2.2 TD Hardware

A TD is based on the following hardware:

- 1) CPU: x2APIC
- 2) CPU: xFLUSH, STI, CLI, LIDT, LGDT instruction are allowed.
- 3) VMM-specific Virtual Device (block device, console, network). This is highly dependent on the hypervisor configuration (e.g. VirtIo device for KVM/XEN, Vmbus device for Microsoft Hyper-V).
- 4) Hot Plug – CPU and memory hotplug are not supported now. Device hotplug is out of scope of this document.
- 5) TD may or may not support below feature. If it is supported, the device must be access via TDCALL [TDG.VP.VMCALL] interface instead of direct hardware access. For example, the IO access needs to be replaced by TDCALL [TDG.VP.VMCALL] <INSTRUCTION.IO>.
 - a) Persistent NV storage.
 - b) The emulated physical device. (Graphic, Keyboard, Storage, etc.)

- c) I/O Subsystems (PCI, USB, ISA, DMA, IOMMU, PIC, PIT, etc.). For example, the PCI might only be used to emulate the VirtIo device.
- d) MMIO (APIC, HPET)
- e) vTPM

2.3 Boot Flow

A TD launch takes below steps:

- 1) VMM sets up TDVF, calls Intel TDX module to create the initial measurement, then calls Intel TDX module to launch TDVF.
- 2) TDVF boots and enables UEFI Secure Boot.
- 3) TDVF prepares CC event log and launches the OS loader.

2.3.1 VMM Setup Phase

The TDVF image includes firmware code, which is measured into TD Measurement Register (MRTD). The TDVF image may also include static configuration data, including UEFI Secure Boot certificates (PK/KEK/db/dbx). The VMM provides dynamic configuration data, including the hand-off block (HOB) list as a parameter for the entrypoint.

The VMM calls Intel TDX module to initialize TD memory. This includes firmware code and UEFI Secure Boot configuration read-only data captured by the tenant:

- Intel TDX module associates memory via Extended Page Table (EPT) with the TD guest.
- Intel TDX module associates logical processors with TD guest via TD-Virtual Machine Control State (VMCS).
- Intel TDX module performs TDENTER instruction on all processors, including Bootstrap Processor (BSP) and Application Processors (APs).

2.3.2 TDVF Launch Phase

TDVF is launched on all processors:

- All processors start in 32-bit protected mode with flat descriptors (paging disabled).
- The CPU with VCPU_INDEX 0 is elected as BSP, the other CPUs are APs.
- Startup code switches to 4-level paging enabled (0-4GB). Option for startup code to switch to 5-level paging enabled (0-4GB).
- BSP performs Virtual Firmware initialization and determines how many APs to wake via TDCALL [TDG.VP.INFO].

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/916121150243010210>