Intel® Trust Domain based Virtual TPM

Design Guide

June 2023

Contents

Figures

Tables

1 Introduction

1.1 Background

A Trust Platform Module (TPM) provides the Root-of-Trust for Reporting (RTR) and Root-of-Trust for Storage (RTS) for a computer platform. With a platform-specific Root-of-Trust for Measurement (RTM), usually in the first boot code, the platform can support TPM-based attestation or TPM-based sealing.

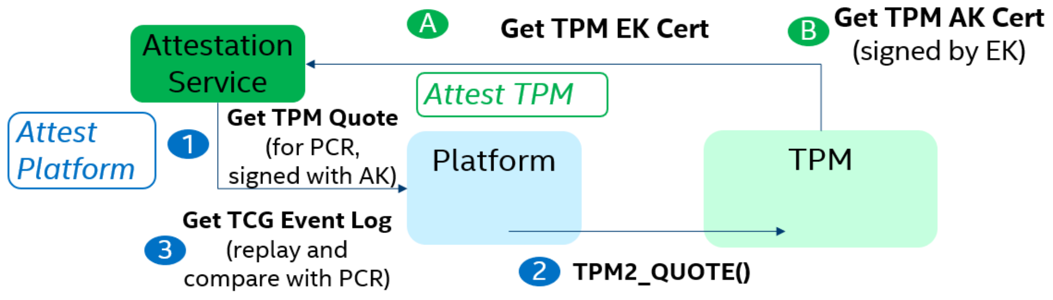

Figure 1: TPM-based Attestation

Figure 1: TPM-based Attestation shows a typical platform attestation with TPM. In a virtual platform, a virtual TPM (vTPM) may be used to support similar TPM-based attestation. It is widely adopted in the hypervisor environment and supported by the virtual machine monitor (VMM) vendor. A VMM may provide virtual TPM services to the guest environment.

However, with the Intel Trust Domain Extension, the VMM is out of Trust Computing Base (TCB) and no longer trusted. As such, a pure VMM-based virtual TPM is not feasible. We need to have another way to support a vTPM-based solution.

1.2 Overview

In this specification, we will describe a Trust Domain (TD) based vTPM solution, which can support vTPM functionality with VMM out of TCB.

Figure 2: A vTPM TD based Attestation shows a vTPM TD-based attestation solution. The vTPM itself is inside of a special TD. As such, the VMM cannot attack the vTPM. The vTPM TD can maintain the confidentiality and integrity of the TPM data.

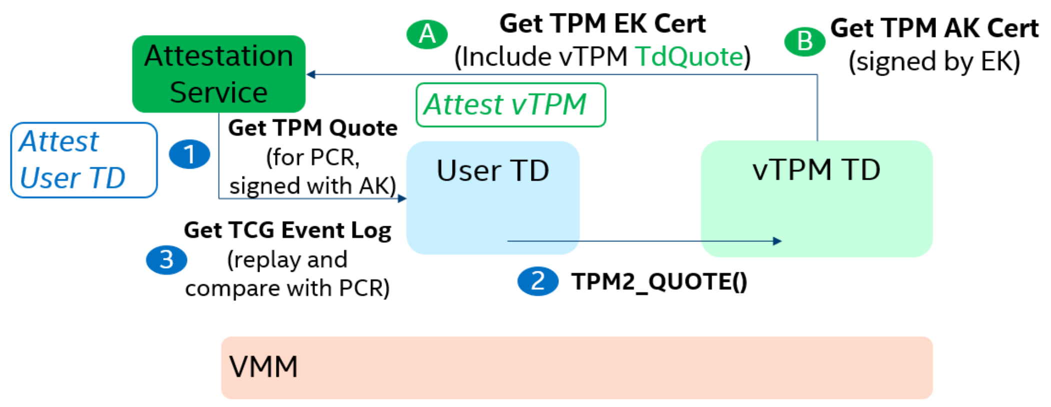
Attestation Service
A Get TPM EK Cert (Include vTPM TdQuote)
B Get TPM AK Cert (signed by EK)
Attest vTPM
Attest User TD
1 Get TPM Quote (for PCR, signed with AK)
User TD
vTPM TD
3 Get TCG Event Log (replay and compare with PCR)
2 TPM2_QUOTE()
VMM

Figure 2: A vTPM TD based Attestation

This document describes the design of the vTPM TD to support a TPM-based attestation use case. The TPM-based sealing use case is not covered in this document, because TDX architecture does not support sealing capability.

1.3 Terminology

Table 1: Terminology

Term	Description
AEAD	Authenticated Encryption with Associated Data
CRB	Command-Response Buffer
DH(E)	Diffie-Hellman (Ephemeral)
DMA	Direct Memory Access
DTLS	Datagram Transport Layer Security
ECDH(E)	Elliptic Curve Diffie-Hellman (Ephemeral)
GHCI	Guest Hypervisor Communication Interface
MR	Measurement Register
MRTD	Measurement Register for TD
NVS	Non-volatile Storage
RA-TLS	Remote Attestation TLS
PCR	Platform Configuration Register
PFP	Platform Firmware Profile
PTP	Platform TPM Profile
RTM	Root-of-Trust for Measurement
RTMR	Runtime Measurement Register
RTR	Root-of-Trust for Reporting
RTS	Root-of-Trust for Storage
SEAM	Secure Arbitration Module
SPDM	Security Protocol and Data Model

Term	Description
SRTM	Static Root-of-Trust for Measurement
TCB	Trust Computing Base
TD	Trust Domain
TDVF	Trust Domain Virtual Firmware
TDX	Trust Domain Extension
TLS	Transport Layer Security
TPM	Trust Platform Module
VMM	Virtual Machine Monitor
vTPM	Virtual TPM

§

2 vTPM Architectural Overview

2.1 vTPM Architecture Solution

A vTPM can be supported in different ways, such as TD-based vTPM or TD Partitioning-based vTPM. See Figure 3: vTPM Solutions.

We do not consider an Intel TDX module as an option for vTPM, because of the following limitations:

- Complexity. It makes TCB larger.
- Long latency to generate a key in TPM.
- NV storage dependency.

This document will focus on the TD-based solution.

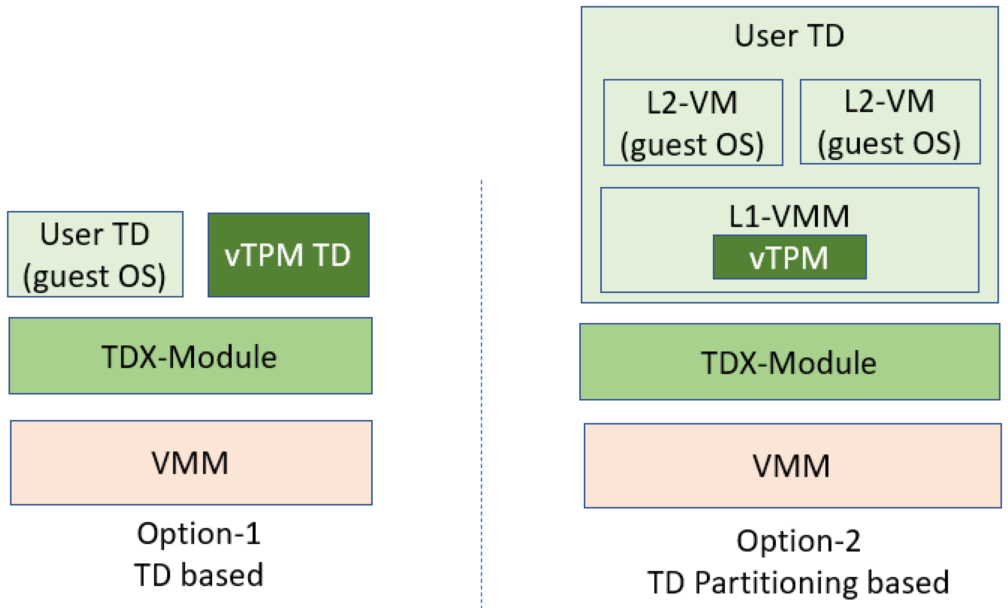

Figure 3: vTPM Solutions

2.1.1 Roles for user TD and L2-VM

The following table shows the important roles for user TD and L2-VM in vTPM solutions.

Table 2: Roles for User TD and L2-VM

Role	vTPM TD (Opt-1)	TD Partitioning vTPM Service (Opt-2)
Virtual Root of Trust for Reporting (vRTR)	**vTPM TD**: TPM software stack.	**vTPM Service**: TPM software stack.
Virtual Root of Trust for Storage (vRTS)	**vTPM TD**: maintain the ephemeral NV storage inside of TD. The NV storage does not exist after the vTPM TD is shutdown.	**vTPM Service**: maintain the ephemeral NV storage inside of TD. The NV storage does not exist after the whole TD is shutdown.
Virtual Root of Trust for Measurement (vRTM)	**TDX-module**: create MRTD. create TDREPORT during mutual authentication. **vTPM TD**: extend TDREPORT to PCR[0] as evidence of initial boot code, after authentication.	**L1-VMM**: extend initial boot block to PCR[0].

NOTE: TPM based attestation information PCR[0] only records the measurement at the system boot time. A TCB update (such as TDX-module update) will not cause the PCR[0] change. This design is aligned with current [PFP]. The TCB update is similar to TPM upgrade, which does not impact TPM EK. Only an explicit TPM2_ChangeEPS() will cause TPM EK change. At that time, the TPM EK will be regenerated.

2.2 vTPM TD requirement

2.2.1 General Requirement

1. The solution SHALL work in existing TDX 1.0. A vTPM TD SHALL not be a TDX 1.5 Service-TD, because there is no need to allow vTPM TD accessing user TD's metadata.

2. The solution SHALL follow TPM 2.0 specification. (Not TPM 1.2)

    a. A vTPM TD MAY provide partial of TPM services, based upon the real use case. (See 2.2.4 vTPM Feature).

3. The solution SHALL not change any TPM Software Stack (TSS).

    a. The solution MAY enlighten every TPM device driver (in TDVF, TD-OS, etc.). But if #VE can be used to make TPM device driver unchanged, the change SHOULD be centralized in #VE handler. For example, the TPM MMIO space access.

2.2.2 vTPM Environment

4. vTPM SHALL be in an isolated environment (such as a TD).

5. One vTPM TD MAY support one vTPM instance for one user TD.

6. vTPM instance SHALL support migration if the user TD is migratable.

2.2.3 vTPM Communication

7. User TD SHALL follow TPM2 specification to send/receive TPM2 command in TPM software stack.
    a. User TD SHOULD use a command-response-buffer (CRB) to send/receive TPM command. The CRB SHALL be used to transfer encrypted-TPM command and response.
8. The communication between vTPM and user TD SHALL be protected in a secure session for confidentiality and integrity with replay protection. The communication SHOULD use industry standard protocol (such as [TLS 1.3] or [SPDM]) with (EC)DHE Key Exchange + AEAD encryption.
    a. The vTPM and user TD SHALL perform integrity verification of the TD Report or Quote from the peer before setup the secure session. For example, TDX Local Attestation or Remote Attestation MAY be used for verifying the vTPM TD and user TD.
9. The vTPM TD and user TD communication SHALL be and only be in 1 secure session.
    a. vTPM TD SHALL ensure the TPM command is from one dedicated user TD and SHALL return the TPM response from the vTPM instance associated with this user TD.
    b. vTPM TD SHALL only communicate the user TD which has not communicated with other vTPM TD yet.
    c. A user TD SHALL only accept the data from one dedicated vTPM TD and one dedicated vTPM instance.

2.2.4 vTPM Feature

10. TPM Crypto primitives SHALL be supported.
11. TPM attestation use case (such as PCR_Extend/Quote) SHALL be supported.
12. Windows BitLocker use case (such as Seal/Unseal) SHALL be supported.
13. TPM defined Non-Volatile Storage (NVS) MAY be ephemeral (NVS disappear after vTPM TD shutdown).

2.3 vTPM Launch Flow

A vTPM may include all TPM specification defined features. Besides a cryptographic algorithm, a TPM should include NVS to store the persistent key. However, a TD cannot provide the secure persistent NVS support. As such, the vTPM TD only maintains the ephemeral NVS inside of TD. See Figure 4: vTPM instance launch Flow and Figure 5: User TD launch Flow.

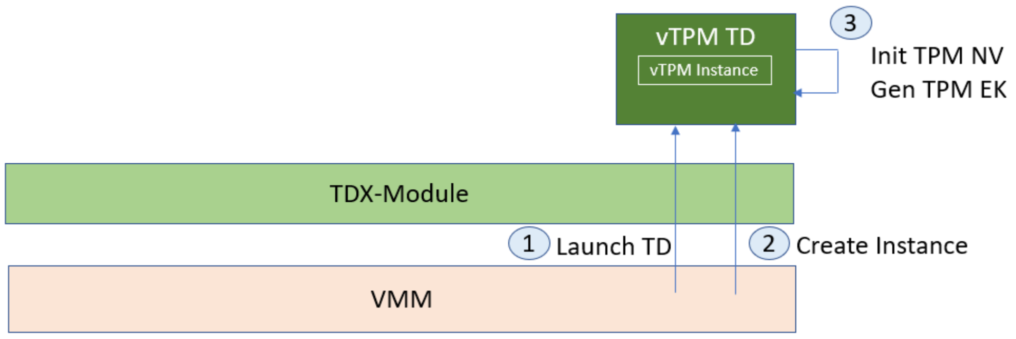

Figure 4: vTPM instance launch Flow

1) A VMM launches the vTPM TD.

2) A VMM creates the vTPM instance.

3) The vTPM TD initializes the TPM NVS inside of vTPM TD for the vTPM instance, and generates TPM EK certificate and saves it to the TPM NVS.

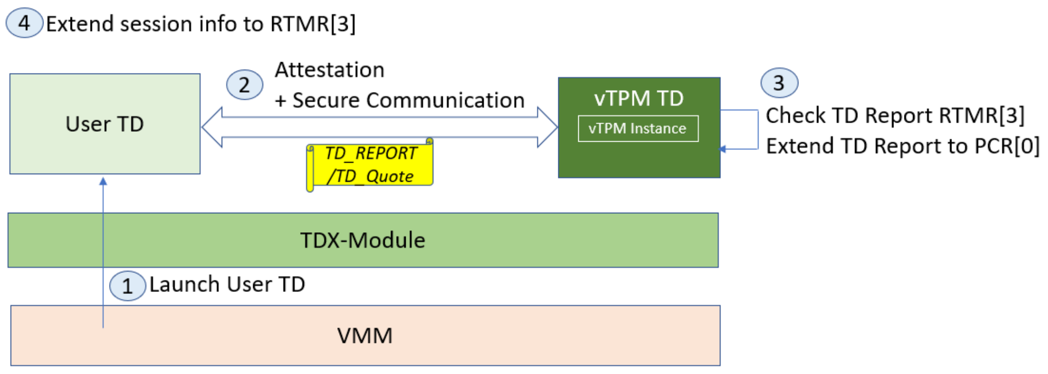

Figure 5: User TD launch Flow

1) A VMM launches the user TD's TDVF and informs the presence of vTPM. The TDVF is extended by the TDX module to MRTD.

2) The TDVF creates a secure session with the vTPM. The TD Report or TD Quote is passed in the secure session and the integrity of TD Report or Quote is verified by the peer.

3) The vTPM TD checks the TD Report RTMR[3] to ensure the user TD is not communicated with vTPM before. Then vTPM TD extends the TD Report to PCR[0].

4) The TDVF extends the session info to RTMR[3] as the evidence that the vTPM is connected. Even if session establishment fails, the RTMR[3] shall still be extended to indicate that the TDVF tried the session establishment.

2.4 Runtime Communication Flow

The following figure shows the vTPM<->User TD communication flow.

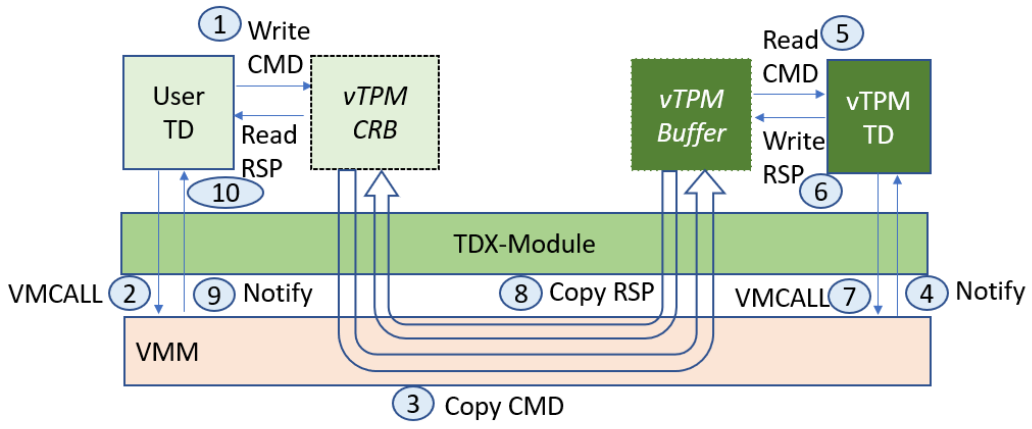

Figure 6: Runtime Communication Flow

Both user TD and vTPM TD have a TPM command-response buffer (CRB) DMA buffer as the GHCI interface between TD and VMM. The CRB DMA buffer is in shared memory, which is accessible by the VMM. As such the TPM command or response in the CRB DMA buffer is encrypted with AEAD algorithm to prevent the attack from the VMM.

The high level TPM communication flow is as follows:

1) User TD prepares the TPM command and encrypts it with AEAD requester key. Then the user TD copies the encrypted vTPM command from its private memory to the TPM CRB.

2) User TD notifies the VMM by using TDG.VP.VMCALL<**Service.vTPM**>.

3) The VMM copies the encrypted vTPM command from the user TD CRB to the vTPM TD CRB.

4) The VMM notifies the vTPM TD the TPM command is ready.

5) Once the vTPM TD gets notification from VMM, the vTPM TD copies the TPM command from the vTPM TD CRB to its private memory. The vTPM TD decrypts the TPM command with AEAD requester key and processes it.

6) The vTPM TD encrypts the TPM response with AEAD responder key and return it back to vTPM TD CRB.

7) The vTPM TD notified the VMM by using TDG.VP.VMCALL<**Service.vTPM**>.

8) The VMM copies the encrypted vTPM response from the vTPM TD CRB to the user TD CRB.

9) Then VMM notifies the user TD the TPM response is ready.

10) Once the user TD gets notification from VMM, the user TD copies the TPM response from the user TD CRB to its private memory. The user TD decrypts the TPM response with AEAD requester key. The decrypted data is the TPM response.

2.5 Trust Relationship

The vTPM TD is the TCB for the user TD.

The trust relationship between user TD and vTPM TD is described in the following sections.

2.5.1 User TD

A user TD trusts a vTPM TD, after the TDVF verifies the vTPM TD REPORT or Quote and creates the secure session. Then the TDVF will extend the SPDM session info to the user TD RTMR[3] after the secure session is established. Even if the session fails to establish, the TDVF shall extend a random nonce value to RTMR[3] to indicate that it tried and failed. NOTE: RTMR[3] is chosen because it is marked as reserved in the Intel TDX Virtual Firmware Design Guide **[TDVF]**, so there should not be any active usage on RTMR[3].

If the vTPM TD is malicious, then the user TD does not know. But the verifier can detect that when it verifies the vTPM TD EK certificate.

2.5.2 vTPM TD

A vTPM TD does not trust a user TD. vTPM TD only trusts the TDX-module.

To prevent TDVF from forging the measurement, vTPM shall record the TDREPORT (received in the secure session) to PCR[0]. The TD REPORT can be trusted because it is from the TDX-module as RTM.

To prevent a guest OS from forging the measurement, vTPM shall ensure it is TDVF that initiates the session by checking the DREPORT.RTMR[3], and rejecting the session request if RTMR[3] is non zero.

If TDVF is malicious and does not extend to RTMR[3], it can be detected by the verifier because TDVF is extended to MRTD by the TDX-module.

2.5.3 vTPM TD <-> User TD binding

To consider vTPM instance NVS* data binding for a user TD: for a physical TPM, there is no binding between HDD and TPM. This is similar for a virtual TPM. There is no binding between virtual HDD (storage) and vTPM.

There is a common request to provide a persistent binding between VM/TD and vTPM. That typically means to bind HDD and vTPM, which is not offered by the vTPM solution.

NOTE: in virtual TPM, we do not provide more security binding properties than physical TPM. This is similar to the following cases:

- Moving an OS disk to another machine.
- Booting another OS on the same machine.

For example, the VMM can launch user TD-1 with vTPM-1, and user TD-2 with vTPM-2. Then VMM shutdowns all and launcesh user TD-2 with vTPM-1.

- The attestation is not impacted, because all PCRs reset.
- The sealed data is not impacted, because we assume user TD shall use the correct sealing policy to seal the data, such as TPM2_PolicyPCR() or TPM2_PolicyPassword() etc.

2.6 vTPM Instance location

The following table shows vTMP Instance location for the same machine versus a different machine.

Table 3: vTPM Instance Location

Impact	Same Machine	Different Machine
vTPM Instance and User TD communication	local attestation	remote attestation
Content in X.509 certificate	**TD Report**	**TD Quote**
User TD PCR0.	User **TD Report** to PCR0	User **TD Quote** to PCR0

2.7 Combined Attestation

Figure 7: Combined Attestation Flow shows the flow of combined attestation.

Figure 8: Combined Attestation shows the components in the combined attestation.

- vTPM TD: the TDX attestation.
- User TD: the standard TPM attestation.

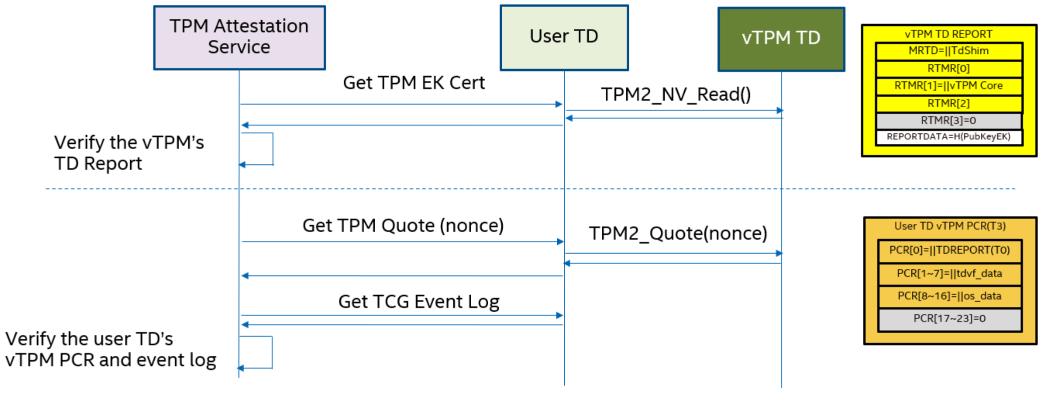

Figure 7: Combined Attestation Flow

NOTE: TPM Quote can provide the freshness of the PCR. [TPM] specification does not provide freshness of the TPM EK by design. A TPM Field Upgrade does not change TPM EK. Only after an explicit TPM2_ChangeEPS() will change TPM EK.

The connection between them is the vTPM Endorsement Key (EK). Every vTPM instance generates a new EK when the instance is created. This TPM EK X.509 certificate will include an Object ID (OID) to indicate the vTPM TD Quote. The REPORTDATA in the TDREPORT is the hash of the vTPM instance EK.

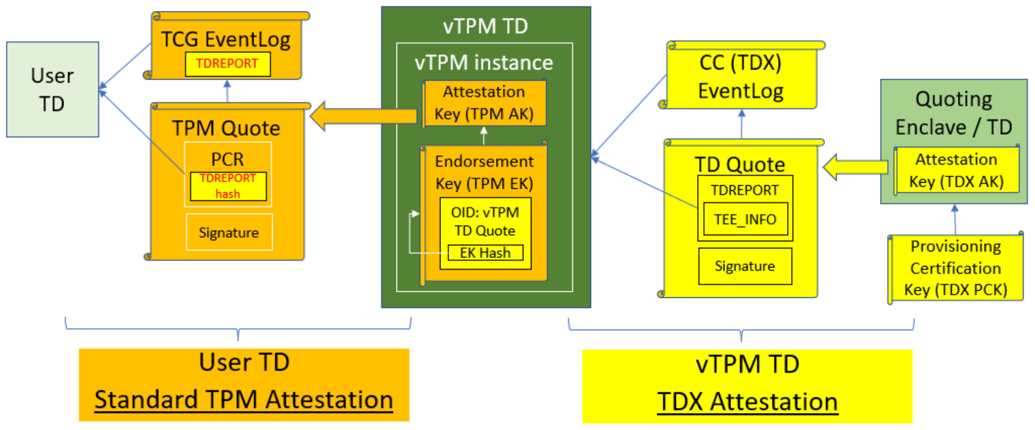

Figure 8: Combined Attestation

NOTE: A TD shall only choose one of RTMR or vTPM based attestation. The selection can happen at build time or runtime. However, a TD shall not support both RTMR and vTPM at runtime, because it may cause a mismatch at the attestation phase. For example, an entity may only be aware of RTMR and extend the measurement to RTMR. However, the verifier may only be aware of vTPM, and try to attest based upon the vTPM. See Table 4: vTPM Attestation Scenario.

Table 4: vTPM Attestation Scenario

Scenario	TDX interface	TPM interface
vTPM is not present	EFI_CC_MEASUREMENT_PROTOCOL CCEL ACPI table (CC event log)	N/A for a good user TD. *If a malicious user TD returns the forged data, it will fail the TPM EK verification.*
vTPM is present and enabled (session established)	N/A for a good user TD. *If a malicious user TD returns the forged data, it will be detected in RTMR verification, because RTMRs are poisoned.*	EFI_TCG2_PROTOCOL TPM2 ACPI table (TCG event log) TPM SSDT ACPI table (TPM instance)
vTPM is present and disabled (session not established)	N/A for a good user TD. *If a malicious user TD returns the forged data, it will be detected in RTMR verification, because RTMRs are poisoned.*	N/A for a good user TD. *If a malicious user TD returns the forged data, it will fail the TPM EK verification.*
vTPM is present and reset in the middle (session reset)	N/A for a good user TD. *If a malicious user TD returns the forged data, it will be detected in RTMR verification, because RTMRs are poisoned.*	N/A for a good user TD. *vTPM TD before reset will close the SPDM session and not return any response.* *vTPM TD after reset will fail accept the SPDM session because RTMR[3] is non-zero.* *If a malicious user TD returns the forged data, it will fail the TPM EK verification or Quote verification.*

2.7.1 User TD measurement register

Figure 9: User TD MR/PCR Change flow shows how user TDs MR and PCR are changed in different phases.

- T0 means just before SPDM session is setup.
- T1 means just after SPDM session is setup.
- T2 means TDVF phase after SPDM session is setup.
- T3 means OS phase after TDVF phase.

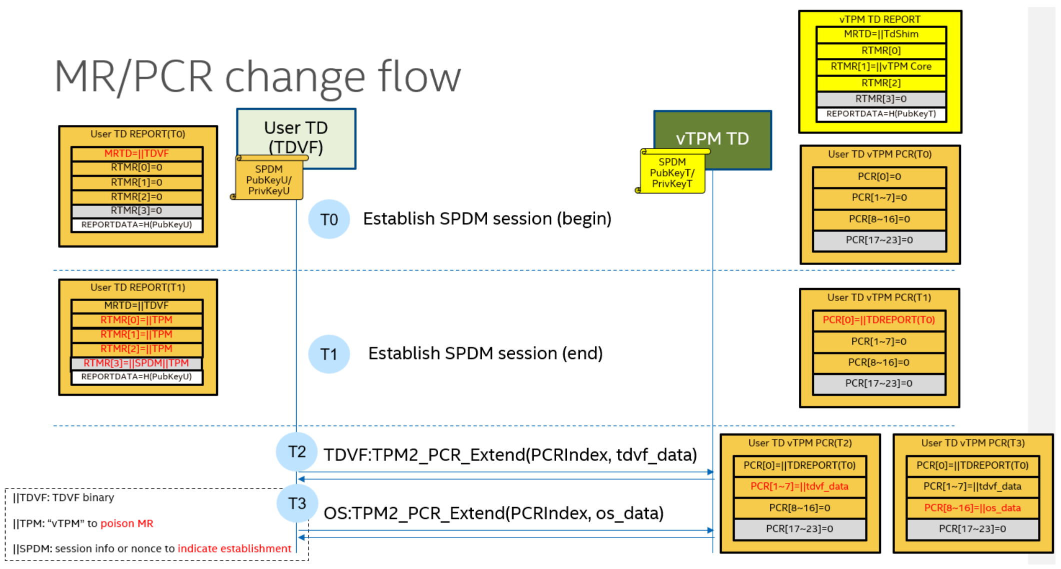

Figure 9: User TD MR/PCR Change flow

Since the user TD will use vTPM PCR, it should not use RTMR. As such we will change the measurement register value as

Table 5: TD Measurement Register.

The TDVF needs to poison all RTMRs to prevent measurement forging.

Table 5: TD Measurement Register

TD Measurement Register	Value
MRTD	No change. **Intel TDX Module extends TDVF BFV to MRTD**.
RTMR[0]	TDVF extends "vTPM" to poison the RTMR after secure session establishment with vTPM.
RTMR[1]	TDVF extends "vTPM" to poison the RTMR after secure session establishment with vTPM.

RTMR[2]	TDVF extends "vTPM" to poison the RTMR after secure session establishment with vTPM.
RTMR[3]	**TDVF BFV extends SPDM session unique info to RTMR[3] after secure session establishment with vTPM**. TDVF extends "vTPM" to poison the RTMR after secure session establishment with vTPM.

2.7.2 User TD vTPM Platform Configuration Register (PCR)

In general, TDVF shall follow TCG PFP specification to extend corresponding entries. See Table 6: vTPM Platform Configuration Register.

Table 6: vTPM Platform Configuration Register (PCR)

TPM PCR	Value
0	Firmware Code. (Follow TCG PFP spec) TDX specific actions: 1. **vTPM TD detects RTMR[3] and rejects if it is non-zero**. 2. **vTPM TD extends user TDREPORT or TD Quote to PCR[0]** after mutual authentication phase. (**NOTE**: To prevent replay attack to the secure session, the user TD needs to generate ephemeral private/public session key pair and put hash of public key as REPORTDATA. The vTPM TD shall NOT include REPORTDATA and MAC to PCR because they are dynamic data. They shall be zeroed to ensure the data is consistent across boot. This follows the TCG PFP specification – no dynamic data in PCR.) 3. TDVF appends it to the event log as the first event during boot.
1	Firmware Configuration Data. (Follow TCG PFP spec)
2	Option ROM Code. (Follow TCG PFP spec)
3	Option ROM Data. (Follow TCG PFP spec)
4	OS Loader Code. (Follow TCG PFP spec)
5	Boot Configuration. (Follow TCG PFP spec)
6	OEM Specific Data. (Follow TCG PFP spec)
7	Secure Boot Configuration. (Follow TCG PFP spec)
8~15	OS application

NOTE: vTPM cannot verify the freshness of the key generation from the user TD. vTPM only verifies the ownership of the private key during session establishment. Assume that each TD shall keep the secret of the private key. But vTPM cannot detect private key leakage. The verifier can verify generation of a fresh ephemeral key by checking the MRTD of the TDVF.

2.8 vTPM EK Certificate

There could be 2 possible vTPM EK certificate modes. See Figure 10: vTPM EK Mode.

- **Self-signed EK**. It is shown in Figure 8: Combined Attestation.
- **CA-signed EK**. If there is vTPM CA, the vTPM TD can send the vTPM TD Quote and vTPM EK PubKey to the vTPM CA and get a vTPM EK Certificate signed by the vTPM CA.

Table 7: vTPM EK Mode

	Self-signed mode	CA-signed mode
vTPM attestation	Required to be enlightened to understand TD Quote	No change required
vTPM TD Quote Verification	Done by verifier	Done by vTPM CA.

The vTPM TD can choose one of the solutions as build time configuration.

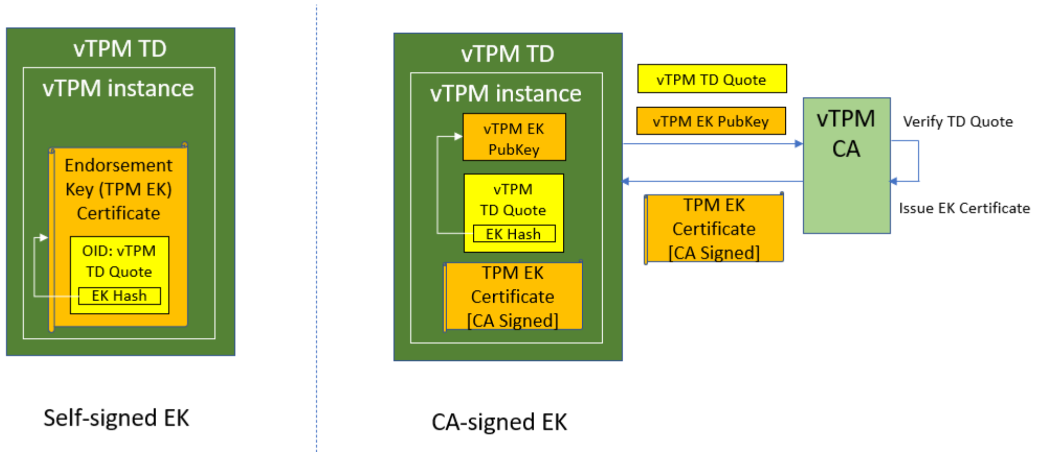

Figure 10: vTPM EK Mode

2.9 vTPM Instance Management

A vTPM TD shall support multiple vTPM instances. Each vTPM instance binds to one specific user TD.

Role/Responsibility:

- **VMM/QEMU**: resource manager – manage vTPM instance.
- **vTPM TD**: Enforce the security property of vTPM instance.
- **User TD**: Consume the vTPM instance.

Normal Flow:

1) VMM/QEMU sends event of VMCALL<Service.vTPM>.[WaitForRequest] with **CreateInstance**.
    a. vTPM TD creates vTPM instance (including volatile storage and non-volatile storage) with the TPM ID.
    b. vTPM TD follows TPM2 spec to provision the vTPM Instance NVS (including EK cert generation)
2) user TD launch.
3) User TD -> setup session
    a. vTPM TD matches user TD and vTPM instance and creates the secure session, adds User TD ID to session list.
4) User TD -> TPM_Start()
    a. vTPM instance -> follows TPM2 specification load NV storage.
5) User TD -> TPM_Shutdown()
    a. vTPM instance -> follows TPM2 specification save NV storage.
    b. vTPM instance will keep the session alive.

6) User TD -> Session termination (SPDM.EndSession)

    a. vTPM TD stops session.

    b. vTPM TD removes the user ID from the session list.

    c. vTPM TD will keep the vTPM instance.

7) User TD shutdown.

8) If VMM/QEMU sends event of VMCALL<Service.vTPM>.[WaitForRequest] with **DestroyInstance**.

    a. vTPM TD destroys vTPM instance (and associated TPM ID)

9) Else (vTPM instance is not destroyed), user TD can start again, go to step 2).

2.10 vTPM Challenge Summary

[TCG VTPM] describes a set of challenges for the virtual TPM implementation. We summarize the solution in Table 8: vTPM Challenges in vTPM Architecture Specification.

Table 8: vTPM Challenges in vTPM Architecture Specification

Challenges	Solution
Protecting Virtual TPM Storage	Use TD to protect vTPM
Protecting vTPM Secrets across Reboots	No persistent storage. The NVS is inside of vTPM TD. See 7 vTPM NV Storage Management.
Attestation	Use Combined Attestation
Supporting Different vTPM Version	N/A. Only support TPM2.0
Field Upgrade of vTPM	vTPM TD teardown/launch. See 13 vTPM Field Upgrade.
vTPM Backup and Restore	Not supported.
Migration	vTPM TD Migration. See 12 vTPM Migration.

Since the vTPM is used for confidential computing environment, we have new challenges summarized in Table 9: vTPM Challenges in Confidential Computing.

Table 9: vTPM Challenges in Confidential Computing

Challenges	Solution
Protecting communication between TEE and vTPM	Use RA-TLS/SPDM for secure session. Use Quote-based mutual attestation.

§

3 vTPM Design Overview

3.1 Design Overview

vTPM TD is a lightweight bare-metal TD. Because vTPM TD is in TCB, the code should be as minimal as possible.

3.1.1 vTPM TD design

Figure 11: vTPM TD Design shows the design of vTPM. It includes a shim layer and a core.

- **TdShim** is a generic shim layer to boot any TD, as a transient environment.
    - TdShim includes a **Reset Vector** that is the first instruction when a VMM launches a TD.
- **Core** is the vTPM TD runtime execution environment.
    - **TPM Lib** provides TPM command process capability. It should reuse the existing known good TPM implementation, such as Microsoft TPM2.0 Reference, https://github.com/microsoft/ms-tpm-20-ref.
    - **Crypto** and **TLS/SPDM** are used to establish a secure communication session. It should reuse the existing known good crypto library and TLS or SPDM configuration including version and cipher suite.
    - **Instance Management** is to manage the vTPM instance
    - **Attestation** is to attest the user TD.
    - **vTPM IO** should follow the Guest-Host Communication Interface (GHCI) specification. The communication shall use TLS or SPDM.
    - **vTPM NVS Management** is to manage the ephemeral NVS instance inside of vTPM TD.

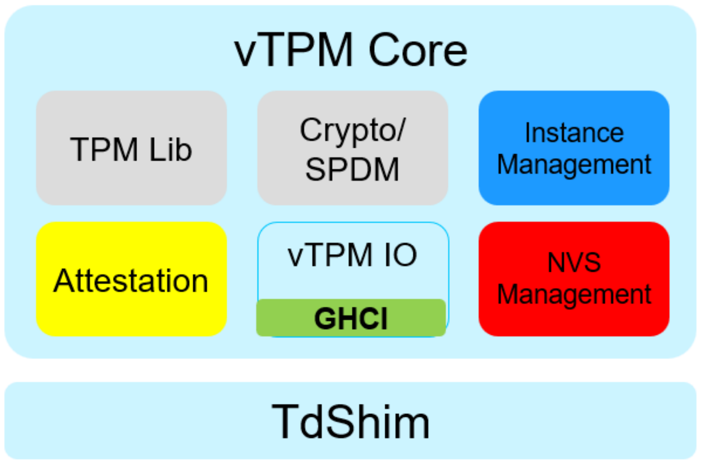

Figure 11: vTPM TD Design

3.1.2 vTPM TD layout

Figure 12: vTPM TD Layout shows the build time layout and runtime layout.

The left-hand side shows the build time layout. The Boot Firmware Volume (BFV) include the TdShim and vTPM core.

The right-handle side shows the runtime layout. The TdShim should be loaded to the top of 4GB memory where the reset vector sits. Then the TdShim will relocate the vTPM core to low memory space and setup x64 long mode environment. It is highly recommended because the top of 4GB is usually mapped to flash device. Executing code in flash device may bring some special restriction even in virtualization environment. All reset additional memory below the top of memory (TOM) can be used as heap, the stack can be allocated from the heap. It is also highly recommended to NOT use the memory below 1MB because the top of 1MB is also mapped to legacy flash ROM.

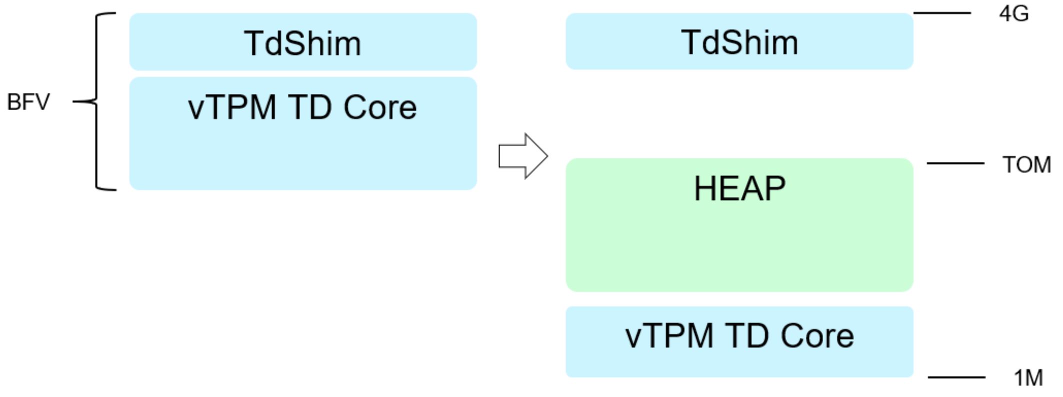

Figure 12: vTPM TD Layout

3.1.3 vTPM TD boot flow

Figure 13: vTPM TD Boot Flow shows the vTPM TD boot flow.

1. **ResetVector in TdShim**. The reset vector code should park all application processors (APs) to a known place and only let bootstrap processor (BSP) initialize the environment. The BSP should switch to long mode, setup stack and jump to the TdShim initial program loader (IPL).

2. **IPL in TdShim**. When TdShim is launched, it should get the memory information internally, such as metadata area. Then it accepts the TD memory (lazy accept or accept all) and sets up all required protection such as data execution prevention (DEP). The final step is to find the vTPM core, load it to low memory and jump to the vTPM core.

3. **Entrypoint of vTPM Core**. The vTPM should initialize the final execution environment including heap and interrupt vector etc. It should also prepare the communication with VMM.

4. **vTPM Core runtime**. The vTPM should use secure communication session as the vTPM IO with a user TD. The user TD should send and receive the TPM command/response via vTPM CRB and let VMM relay the message to vTPM IO GHCI. The vTPM receives TPM command from the vTPM IO, processes it, and sends TPM response to the vTPM IO.

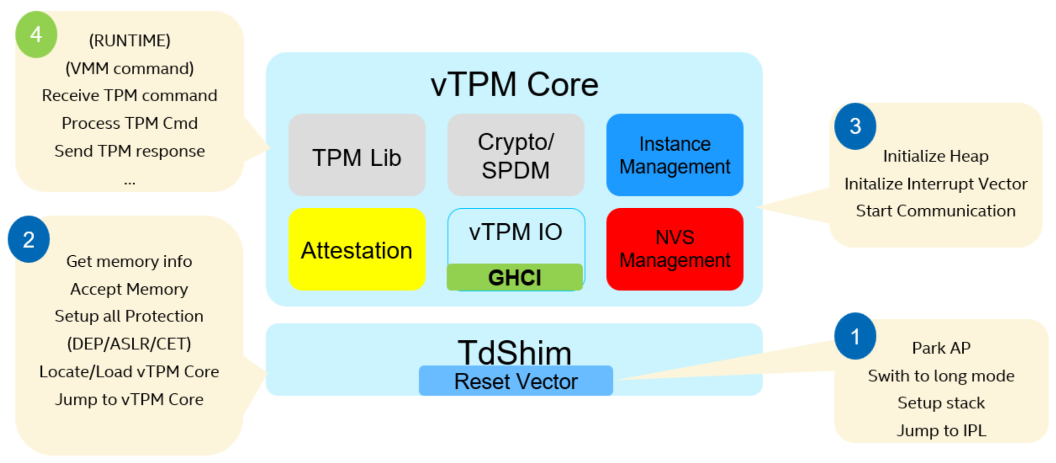

Figure 13: vTPM TD Boot Flow

3.2 Reproducibility

A vTPM TD binary should be reproducible. Please refer to "TDX Virtual Firmware Design Guide" for reproducible support information.

3.3 Security Consideration

vTPM TD should adopt the Security Consideration in the "TDX Virtual Firmware Design Guide".

3.3.1 Type safe language

The vTPM TD should consider using type safe language, such as Rust programming language to eliminate memory safety issues.

4 vTPM IO Interface

When a user TD communicates with a vTPM TD, it should use a standard secure communication protocol to protect the communication.

4.1 vTPM TD detection from user TD

The user TD may send TDG.VP.VMCALL<**Service.Query**> (VMCALL_SERVICE_VTPM_GUID) to detect the presence of vTPM TD.

4.2 User TD/VMM Interface

Generic design:

1) User TD follows the [TCG PTP] to access the vTPM hardware. (Please see "vTPM Profile" section.)

vTPM specific design:

1) User TD needs to setup a TPM secure session with vTPM TD. VMM needs to be in the middle to relay the message between them.
2) User TD needs to issue TDG.VP.VMCALL<**Service.vTPM**> to create a TPM secure session, update the session key, or destroy a TPM secure session.
3) If vTPM TD detects the SPDM protocol failure/attack, vTPM TD shutdowns the vTPM instance. (=TPM2_Shutdown). vTPM instance cannot be used until next TPM2_Start.
4) If user TD detects the SPDM protocol failure/attack, user TD marks vTPM no longer unusable in user TD (set NotPresent in EFI_TCG2_PROTOCOL), but will not try to reestablish an SPDM session. The user TD can still run but the attestation can no longer be supported, either TDX attestation or vTPM attestation cannot be performed.

4.3 User TD internal Interface

4.3.1 User TDVF vTPM Driver

Generic design:

1) User TDVF follows the [TCG PFP] to measure the firmware content into vTPM, with exception for the content measured in MRTD. (See "vTPM solution specific design" section)
    a. The digest extended to PCR shall match the digest extended to MRTD or RTMR. The PCR-TD mapping is defined in [TDVF].
    b. User TDVF shall extend PCR[1~7] (RTMR[0~1]) and create corresponding event log.
    c. User TDOS/loader shall extend PCR[8~15] (RTMR[2]) and create corresponding event log.
2) User TDVF follows the [TCG EFI] to expose EFI_TCG2_PROTOCOL.
3) User TDVF follows the [TCG ACPI] to expose TPM2 ACPI table.

vTPM specific design:

1) TPM command/response in TPM_CRB_DATA_BUFFER is encrypted by "**TPM secure session key**".

    a. Before any TPM communication, user TD need communicate with vTPM TD to setup a secure session with mutual authentication and key exchange. The "**TPM secure session key**" is setup for the secure session. (See section "User TD/VMM Interface")

    b. User TD need pass the "TPM secure session key" expose AEAD key to the TD OS via **TD-vTPM ACPI table** (See section "TD-vTPM ACPI Table").

2) User TDVF need have special handling to establish the chain of trust for the user TD.

    a. vTPM TD shall check user TD_REPORT RTMR[3] and reject the communication request if it is non-zero.

    b. vTPM TD shall help to extend PCR[0] for user TD. When vTPM TD attest the user TD in TPM secure session setup, vTPM TD need record and extend to PCR[0] by using H-CRTM sequence. The input data for _TPM_HASH_DATA is the hash of user TD_REPORT with zeroized REPORTDATA and MAC.

    The flow is:

    ```
    TD_REPORT.REPORTMACSTRUCT.REPORTDATA = 0
    TD_REPORT.REPORTMACSTRUCT.MAC = 0
    _TPM_Hash_Start
    _TPM_Hash_Data (SHA384 (TD_REPORT))
    _TPM_Hash_End
    ```

    c. User TDVF shall create **EV_NO_ACTION/TCG_EfiStartupLocalityEvent** as the first event to indicate the occurrence of H-CRTM sequence.

    The flow is:

    ```
    Event Type = EV_NO_ACTION
    Event = TCG_EfiStartupLocalityEvent {
        Signature = "StartupLocality"
        StartupLocality = 4
    }
    ```

    d. User TDVF shall create **EV_EFI_HCRTM_EVENT** as the second event to indicate the value of the H-CRTM component.

    The flow is:

    ```
    TD_REPORT.REPORTMACSTRUCT.REPORTDATA = 0
    TD_REPORT.REPORTMACSTRUCT.MAC = 0
    PCR Index = 0
    ```

```
Event Type = EV_EFI_HCRTM_EVENT

Digests = Hash (SHA384(TD_REPORT))

Event = "HCRTM"
```

e. User TDVF shall create **EV_NO_ACTION/TCG_HCRTMComponentEvent** as the second event to indicate the value in H-CRTM sequence.

The flow is:

```
TD_REPORT.REPORTMACSTRUCT.REPORTDATA = 0

TD_REPORT.REPORTMACSTRUCT.MAC = 0

Event Type = EV_NO_ACTION

Event = TCG_HCRTMComponentEvent {

    Signature = "H-CRTM CompMeas"

    ComponentDescriptionSize = sizeof("TD REPORT")

    ComponentDescription = "TD REPORT"

    MeasurementFormatType = Digest (0)

    ComponentMeasurementSize = 2 + SHA_384_SIZE (48)

    ComponentMeasurement.hashAlg = TPM_ALG_SHA384 (0xC)

    ComponentMeasurement.digest = SHA384(TD_REPORT)

}
```

f. User TDVF shall extend HASH(SpdmSessionInfo) to the RTMR[3] if the session is established, or a nonce to RTMR[3] if the session is failed.

g. User TDVF shall extend the RTMR[0~3] with HASH("vTPM") to poison the RTMRs.

4.3.2 User TDOS vTPM Driver

Generic design:

1) User TDOS follows the [TCG EFI] to consume EFI_TCG2_PROTOCOL.

2) User TDOS follows the [TCG ACPI] to consume TPM2 ACPI table and get TPM_CRB_DATA_BUFFER address.

vTPM specific design:

1) TPM command/response in TPM_CRB_DATA_BUFFER is encrypted by "**TPM secure session key**".

2) User TDOS consumes **TD-vTPM ACPI table** to get the "TPM secure session key".

4.4 vTPM TD/VMM Interface

vTPM specific design:

1) vTPM TD need setup a TPM secure session with user TD. VMM need be in the middle to rely the message between them.

2) vTPM TD need issue TDG.VP.VMCALL<**Service.vTPM**> to create a TPM secure session, update the session key, or destroy a TPM secure session.

3) vTPM TD need issue TDG.VP.VMCALL<**Service.vTPM**> to receive TPM command and send TPM response.

vTPM NVS server specific design:

1) vTPM TD need communicate with vTPM NVS server to manage the NVS. VMM need be in the middle to relay the message between them.

2) vTPM TD need issue TDG.VP.VMCALL<**Service.vTPM**> to manage the NVS in the vTPM NVS server.

§

5 vTPM Secured Message

5.1 User TD TDG.VP.VMCALL<Service.VTPM>

This function is used to allow user TD to send and receive vTPM messages.

```
#define VMCALL_SERVICE_VTPM_GUID \
{0x64590793, 0x7852, 0x4e52, 0xbe, 0x45, 0xcd, 0xbb, 0x11, 0x6f, 0x20, 0xf3}
```

5.1.1 TDG.VP.VMCALL <Service.VTPM.SendMessage>

Table 10: User TD TPM <Service.VTPM.SendMessage> Command

Field	Offset (Bytes)	Length (bytes)	Description
Version	0	1	0: for this data structure
Command	1	1	1: **User TD TPM SendMessage**
Reserved	2	2	Reserved
Secure TPM Message	4	N	The secure session protocol (TLS or SPDM) used for TPM message. It includes the handshake phase message and application phase message. See Table 20: Secure Session Message format for SPDM.

Table 11: User TD TPM <Service.VTPM.SendMessage> Response

Field	Offset (Bytes)	Length (bytes)	Description
Version	0	1	0: for this data structure
Command	1	1	1: **User TD TPM SendMessage**
Status	2	1	Status of the response
Reserved	3	1	Reserved

5.1.2 TDG.VP.VMCALL <Service.VTPM.ReceiveMessage>

Table 12: User TD TPM <Service.VTPM.ReceiveMessage> Command

Field	Offset (Bytes)	Length (bytes)	Description
Version	0	1	0: for this data structure
Command	1	1	2: **User TD TPM ReceiveMessage**
Reserved	2	2	Reserved

Table 13: User TD TPM <Service.VTPM.ReceiveMessage> Response

Field	Offset (Bytes)	Length (bytes)	Description
Version	0	1	0: for this data structure
Command	1	1	2: **User TD TPM ReceiveMessage**
Status	2	1	Status of the response
Reserved	3	1	Reserved
Secure TPM Message	4	N	The secure session protocol (TLS or SPDM) used for TPM message. It includes the handshake phase message and application phase message. See Table 20: Secure Session Message format for SPDM.

5.2 vTPM TD VMCALL<Service.VTPMTD>

This is used to allow vTPM TD to get the instruction from VMM.

```
#define VMCALL_SERVICE_VTPM_TD_GUID \
{0xc3c87a08, 0x3b4a, 0x41ad, 0xa5, 0x2d, 0x96, 0xf1, 0x3c, 0xf8, 0x9a, 0x66}
```

5.2.1 TDG.VP.VMCALL <Service.VTPMTD.WaitForRequest>

Table 14: vTPM TD <Service.VTPMTD.WaitForRequest> Command

Field	Offset (Bytes)	Length (bytes)	Description
Version	0	1	0: for this data structure
Command	1	1	0x1: **vTPM TD WaitForRequest**
Reserved	2	2	Reserved
TPM ID	4	16	Optional field. All 0 means broadcast.

Table 15: vTPM TD <Service.VTPMTD.WaitForRequest> Response

Field	Offset (Bytes)	Length (bytes)	Description
Version	0	1	0: for this data structure
Command	1	1	0x1: **vTPM TD WaitForRequest**
Operation	2	1	0: No-op 1: Communicate 2: Create Instance 3: Destroy Instance
Reserved	3	1	Reserved
TPM ID	4	16	
Payload	20	N	If Operation is Communicate, it is the secure session protocol (TLS or SPDM) used for TPM message. It includes the handshake phase message and application phase message. See Table 20: Secure Session Message format for SPDM. If Operation is Create Instance or Destroy Instance, it is absent.

5.2.2 TDG.VP.VMCALL <Service.VTPMTD.ReportStatus>

Table 16: vTPM TD <Service.VTPMTD.ReportStatus> Command

Field	Offset (Bytes)	Length (bytes)	Description
Version	0	1	0: for this data structure
Command	1	1	0x2: **vTPM TD ReportStatus**
Operation	2	1	0: No-op 1: Communicate 2: Create Instance 3: Destroy Instance
Status	3	1	0: SUCCESS 1: INVALID_PARAMETER 2: UNSUPPORTED 3: OUT_OF_RESOURCE 4: Reserved 5: NETWORK_ERROR 6: SECURE_SESSION_ERROR 7: MUTUAL_ATTESTATION_ERROR 8: VTPM_MIGPOLICY_ERROR 9: VTPM_INSTANCE_ALREADY_STARTED 0xA: VTPM_INSTANCE_NOT_STARTED 0xFF: VTPMTD_INTERNAL_ERROR 0xB~0xFE: Reserved
TPM ID	4	16	
Payload	20	N	If Operation is Communicate, it is the secure session protocol (TLS or SPDM) used for TPM message. It includes the handshake phase message and application phase message. See Table 20: Secure Session Message format for SPDM. If Operation is Create Instance or Destroy Instance, it is absent.

Table 17: vTPM TD <Service.VTPMTD.ReportStatus> Response

Field	Offset (Bytes)	Length (bytes)	Description
Version	0	1	0: for this data structure
Command	1	1	0x2: **vTPM TD ReportStatus**
Reserved	2	2	Reserved

5.3 Secure Session Message

The user TD shall send to secure session messages to vTPM TD and receive the responses from vTPM TD.

The secure session message includes the handshake message to create the session and the application message to pass the data in the secure session. Here the data means TPM command or TPM response in [TPM2].

5.3.1 SPDM Protocol Configuration

Secure session message uses [SPDM].

Table 18: SPDM Protocol Configuration

	Supported Configuration
Version	1.2

Capability	KEY_EX_CAP = 1
	ENCRYPT_CAP = 1
	MAC_CAP = 1

Algorithm	Hash: SHA384 AsymSig: ECDSA-NIST_P384 DHE: ECDHE-SECP_384r1 AEAD: AES_256_GCM

5.3.2 Mutual Authentication

vTPM solution uses attestation-based authentication – Remote Attestation TLS or SPDM (RA-TLS/SPDM). The attestation could be local attestation by using TD_REPORT, or remote attestation by using TD_Quote.

The certificate is an ephemeral certificate to pass the TD_REPORT or TD_Quote to the peer. The flow is:

1) User TD and vTPM TD generate an ephemeral private/public key-pair for this session.
2) User TD and vTPM TD include the hash of public key in the TD_REPORT or TD_Quote.
3) User TD and vTPM TD generate the X.509 certificate for the public key, including OID for TD_REPORT or TD_Quote, CC event log (vTPM only).

Table 19: Certificate Field

Field	Description	Required
Version	Version of the encoded certificate shall be present and shall be version 3 (**value 0x2**)	Mandatory
Serial Number	Serial number shall be present with a positive integer value. For example: **Serial Number: 1**	Mandatory
Signature Algorithm	Signature algorithm shall be present. For example: **sha384WithRSAEncryption**	Mandatory
Issuer	Issuer distinguished name shall be specified.	Mandatory
Subject Name	Subject name shall be present and shall represent the distinguished name associated with the certificate. It shall be same as Issuer.	Mandatory
Validity	Certificate may include this attribute. If the validity attribute is present, the value for **notBefore** field should be assigned the generalized **19700101000000Z** time value and **notAfter** field should be assigned the generalized **99991231235959Z** time value.	Mandatory
Subject Public Key Info	Device public key and the algorithm shall be present. For example: **Public Key Algorithm: rsaEncryption** **Modulus: ...** **Exponent: 65537 (0x10001)**	Mandatory
X509v3 Extension: Basic Constraints	**CA: FALSE.**	Optional
X509v3 Extension:	Subject Key Identifier	Optional

Field	Description	Required
Subject Key Identifier		
X509v3 Extension: Authority Key Identifier	It should be same as Subject Key Identifier.	Optional
X509v3 Extension: Extended Key Usage	Ephemeral Certificate indicator vTPM TD: "2.16.840.1.113741.1.5.5.2.1" User TD: "2.16.840.1.113741.1.5.5.3.1"	Mandatory
OID:TD_REPORT	TD_REPORT vTPM TD: "2.16.840.1.113741.1.5.5.2.4" User TD: "2.16.840.1.113741.1.5.5.3.4"	Mandatory (in local attestation)
OID:TD_Quote	TD_Quote vTPM TD: "2.16.840.1.113741.1.5.5.2.2" User TD: "2.16.840.1.113741.1.5.5.3.2"	Mandatory (in remote attestation)
OID:Event_Log	CC Event Log vTPM TD: "2.16.840.1.113741.1.5.5.2.3"	Mandatory for vTPM only.

5.3.3 SPDM Secure Session Message Format

The Secure Session Message for SPDM is defined in the following table.

Table 20: Secure Session Message format for SPDM

Field	Byte Length	Byte Offset	Description
Message Length (L)	2	0	Length of remaining fields, including Version (V), Message Type (T) and Message.
Version (V)	1	2	0x01
Message Type (T)	1	3	1 – DSP0274 SPDM message 2 – DSP0277 Secured SPDM message
Message	Message Length	4	For message type 1, it is DSP0274 SPDM message, defined in [SPDM]. It starts from "SPDMVersion" field. For message type 2, it is DSP0277 Secured SDPM message, defined in [Secure SPDM] and Table 21: DSP0277 Secure Session Message format for SPDM as binding. It starts from "Session ID" field.

Table 21: DSP0277 Secure Session Message format for SPDM

Field	Byte Length	Byte Offset	Description
Session ID (SID)	4	0	SPDM Session ID
Sequence Number (Seq)	8	4	Sequence Number.
Total Length (Len)	2	12	Length of remaining fields, including Application Data Length, Application Data, Random Data, MAC.
Application Data Length (App Len)	2	14	Length of Data
Application Data	Application Data Length	16	The application data form is defined in Table 22: Application data format for SPDM.
Random Data (Rand)	R [0, 16]	16 + Application Data Length	Random byte: Min 0, Max 16.
MAC	MacLen	16 + R + Application Data Length	AEAD Tag (For AES-256-GCM, MacLen is 16.)

AEAD requirement is as follows (Refer to [Secured SPDM])

AEAD Associated Data:

1. Session ID
2. Sequence Number
3. Total Length

AEAD Plaintext/Ciphertext:

1. Application Data Length
2. Application Data
3. Random Data

AEAD Tag:

1. MAC

AEAD nonce derivation per message:

1. Zero extend 64-bit sequence number to IV length.
2. Perform a bitwise XOR of the zero extended sequence number with IV.
3. Increase sequence number by one.

Table 22: Application data format for SPDM

Field	Byte Length	Byte Offset	Description
Message Type (T)	1	0	1 – DSP0274 SPDM message 3 – TPM message
Message	Application Data Length - 1	4	For message type 1, it is DSP0274 SPDM message, defined in [SPDM]. It starts from "SPDMVersion" field. For message type 3, it is TPM Command or Response, defined in [TPM2]. It starts from "tag" field.

The final supported message types are shown in the following figure.

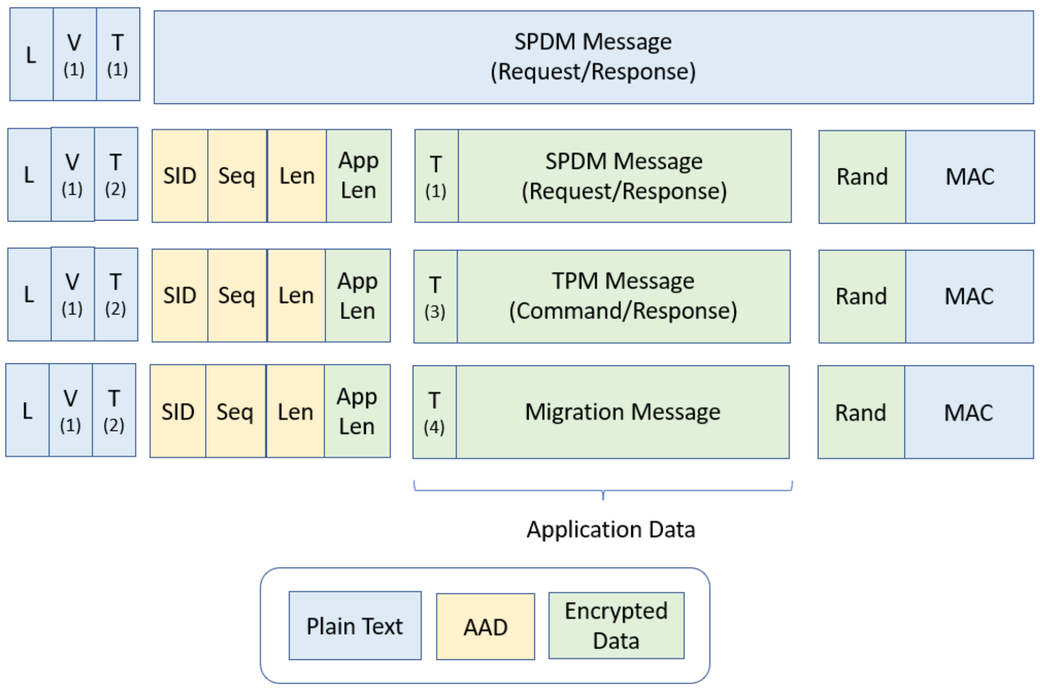

Figure 14: Supported Message

5.4 TD vTPM Key ACPI Table

TDVF needs to set up an ACPI table to pass the TD-vTPM session information.

Table 23: Intel TD vTPM Key ACPI Table

Field	Byte Length	Byte Offset	Description
Header			
Signature	4	0	'TDTK' Signature.
Length	4	4	Length, in bytes, of the entire Table
Revision	1	8	1
Checksum	1	9	Entire table must sum to zero.
OEMID	6	10	Standard ACPI header
OEM Table ID	8	16	Standard ACPI header
OEM Revision	4	24	Standard ACPI header
Creator ID	4	28	Standard ACPI header
Creator Revision	4	32	Standard ACPI header
Reserved	4	36	Reserved
vTPM Secure Session Info Header	16	40	vTPM Secure Session Info Header (See Table 24: vTPM Secure Session Info Header)

Table 24: vTPM Secure Session Info Header

Field	Byte Length	Byte Offset	Description
Version	2	0	0x0100
Protocol	1	2	0x00: SPDM 0x80: TLS 1.2 -- DROP 0x81: TLS 1.3 -- DROP
Reserved	1	3	Reserved to 0
Length	4	4	Length of Secure Session Info Table.
Address	8	8	Address of Secure Session Info Table. The Secure Session Info table must be allocated in ACPI NVS memory, because the information (such as sequence number) may be updated after the table is created. The Secure Session Info Table (It depends on Protocol field). See Table 25: vTPM Session Info Table for SPDM.

5.4.1 SPDM based vTPM Session Information

SPDM session key is passed to OS directly via below table.

Table 25: vTPM Session Info Table for SPDM

Field	Byte Length	Byte Offset	Description
Transport Binding Version	2	0	0x1000 (version 1.0) This version needs to be used KEY_EXCHANGE and KEY_EXCHANGE_RSP.
AEAD Algorithm	2	2	0x1 – AES-256-GCM (KeyLen is 32, IvLen is 12)
Session ID	4	4	SPDM Session ID
Request Direction Key	KeyLen	8	Key to encrypt the TPM command by user TD.
Request Direction IV	IvLen	8+KeyLen	IV associated with the "Request Direction Key".
Request Direction Sequence Number	8	8+KeyLen + IvLen	Sequence Number associated with the "Request Direction Key".
Response Direction Key	KeyLen	16+ KeyLen + IvLen	Key to encrypt the TPM response by vTPM TD.
Response Direction IV	IvLen	16 + KeyLen*2 + IvLen	IV associated with the "Response Direction Key".
Response Direction Sequence Number	8	16 + KeyLen*2 + IvLen*2	Sequence Number associated with the "Response Direction Key".

§

6 vTPM Profile

vTPM TD follows [TCG PTP] specification, "TPM Attributes" chapter and "TPM Capabilities and Commands" chapter.

6.1 vTPM Attributes

6.1.1 vTPM Algorithms

vTPM TD follows [TCG PTP] specification, "PC Client Algorithms" section and "PC Client Curves" section.

6.1.2 vTPM NVS

vTPM TD follows [TCG PTP] specification, "NV Storage Requirement" section.

The vTPM TD will maintain the ephemeral vTPM NVS inside of vTPM TD.

6.1.3 vTPM EK Certificate

vTPM TD follows [TCG PTP] specification, "Endorsement Key Certificate" section.

The vTPM instance shall be provisioned with EK certificates, following [TPM2 EK] specification. For example, NV Index 0x01c00002 is for RSA 2048 EK Certificate. NV Index 0x01c0000a is for ECC NIST P256 EK Certificate.

In self-signed mode, the vTPM instance EK certificate shall be issued by the vTPM TD certificate. The flow is:

1) vTPM TD generates a private/public key-pair for this vTPM instance.

2) vTPM TD includes the hash of public key in the TD_Quote.

3) vTPM TD generates the X.509 certificate for the public key, including OID for TD_Quote and CC event log.

See Table 26: EK Certificate Field for more details.

Table 26: EK Certificate Field

Field	Description	Required
Version	Version of the encoded certificate shall be present and shall be version 3 (**value 0x2**)	Mandatory
Serial Number	Serial number shall be present with a positive integer value. For example: **Serial Number: 1**	Mandatory
Signature Algorithm	Signature algorithm shall be present. For example: **sha384WithRSAEncryption**	Mandatory
Issuer	Issuer distinguished name shall be specified.	Mandatory

Field	Description	Required
Subject Name	Subject name shall be present and shall represent the distinguished name associated with the certificate. It shall be same as Issuer.	Mandatory
Validity	Certificate may include this attribute. If the validity attribute is present, the value for **notBefore** field should be assigned the generalized **19700101000000Z** time value and **notAfter** field should be assigned the generalized **99991231235959Z** time value.	Mandatory
Subject Public Key Info	Device public key and the algorithm shall be present. For example: **Public Key Algorithm: rsaEncryption** **Modulus: ...** **Exponent: 65537 (0x10001)**	Mandatory
X509v3 Extension: Basic Constraints	CA: FALSE.	Optional
X509v3 Extension: Subject Key Identifier	Subject Key Identifier	Optional
X509v3 Extension: Authority Key Identifier	It should be same as Subject Key Identifier.	Optional
X509v3 Extension: Extended Key Usage	vTPM TD issued EK Certificate indicator "2.16.840.1.113741.1.5.5.2.5"	Mandatory
OID:TD_Quote	TD_Quote "2.16.840.1.113741.1.5.5.2.2"	Mandatory
OID:Event_Log	CC Event Log "2.16.840.1.113741.1.5.5.2.3"	Mandatory

6.1.4 vTPM PCR

vTPM TD follows [TCG PTP] specification, "PCR Requirements" section.

vTPM only supports SRTM PCR[0~16]

6.1.5 vTPM AK Certificate

vTPM TD follows [TPM2 KEY] specification to generate Attestation Key (AK). The detail setup is in "Identity Provisioning" section of [TPM2 KEY].

NOTE: OEM creates Initial AK (IAK) while owner creates Local AK (LAK).

6.2 vTPM Capabilities and Commands

6.2.1 vTPM Command

vTPM TD follows [TCG PTP] specification, "Command Table" section.

6.2.2 vTPM Locality

vTPM only supports locality 0. Other localities (1~4) are unsupported.

6.2.3 vTPM Timeout

vTPM uses a different interface. As such, it does not follow [TCG PTP] specification, "**Interface Timeouts**" section, including TIMEOUT_A, TIMEOUT_B, TIMEOUT_C, TIMEOUT_D. The vTPM deriver in TD may wait longer.

6.3 vTPM Software Interface

6.3.1 vTPM Interface Type

User TD and vTPM TD use TDG.VP.VMCALL<**Service.vTPM**> to transmit messages.

§

7 vTPM NV Storage Management

vTPM-TD only provides ephemeral NV Storage support inside of the vTPM-TD. That means the TPM Instance NVS can be reused as long as vTPM Instance is not deleted and vTPM TD is not shutdown. After the VMM terminates the vTPM-TD, the NV Storage does not exist. When VMM launches the vTPM-TD again, the vTPM-TD need reprovision the vTPM instance. There is no persistent NV storage, because the TDX architecture does not have sealing capability.

The vTPM-TD will management the NV Storage in the TD for the vTPM instance. When a vTPM instance is created, the vTPM NVS instance is created. When a vTPM instance is destroyed, the vTPM NVS instance is destroyed.

An OSV may add extension to support persistent NV storage for the vTPM TD. For example, using a NVS server to provide secure storage. That is out of scope of this document.

§

8 vTPM TD Binary Image

8.1 Boot Firmware Volume (BFV)

The vTPM includes one Firmware Volumes (FV) - Boot Firmware Volume. The format of FV is defined in PI specification.

The Boot Firmware Volume includes all component required during boot.

The file system GUID must be EFI_FIRMWARE_FILE_SYSTEM2_GUID or EFI_FIRMWARE_FILE_SYSTEM3_GUID, which is defined in PI specification.

1) TD Shim

   a) **ResetVector** – this component provides the entrypoint for vTPM, switch to long mode, and jumps to the ShimIpl. The FFS GUID must be EFI_FFS_VOLUME_TOP_FILE_GUID, which is defined in PI specification.

   b) **ShimIpl** – This component prepares the required parameter for vTPM Core and jump to the vTPM Core. Module type is EFI_FV_FILETYPE_SECURITY_CORE, which is defined in PI specification.

2) **vTPM Core** – This is main vTPM module. It finishes all its work described in chapter 2 then teardown itself. vTPM core includes key exchange cryptography capability and networking capability via VMM interface. File type is EFI_FV_FILETYPE_DXE_CORE, which is defined in PI specification.

The BFV may include an initial static page table to assist the ResetVector switch from 32bit mode to 64bit mode.

8.2 Configuration Firmware Volume (CFV)

vTPM does not includes a Configuration Firmware Volume (CFV).

§

9 vTPM TD Launch

A vTPM TD launch is similar to a TDVF launch.

9.1 vTPM initialization

The vTPM TD initialization is same as TDVF initialization flow. It starts from 32bit protected mode with paging disabled, then switch to 64bit long mode.

9.2 vTPM Hand-Off Block (HOB)

vTPM TD shall ignore the TD HOB passed from VMM. The TD memory information shall be indicated by vTPM TD in the metadata area.

9.3 vTPM teardown

To provide vTPM service to the user TD, the vTPM TD shall be alive if the user TD is alive.

To support vTPM NV storage, the vTPM TD should be alive even if the user TD is shutdown, but the user TD may be started later.

The vTPM TD can be teardown if the user TD is no longer needed, or a vTPM TD update is required.

9.4 vTPM AP handling

For simplicity, a vTPM only supports one processor.

§

10 vTPM TD Measurement

10.1 Non-Secure Boot Mode

TD-Shim is treated as firmware code and extended to MRTD. The vTPM Core is treated as OS code and extended to RTMR[1]. See Table 27: TD Measurement Registers in Non-Secure Boot Mode.

However, if we choose this approach, there is no way to know a secure version number (SVN) of a vTPM. If we want to attest the vTPM identified with SVN, then we need a different way – Secure Boot Mode.

Table 27: TD Measurement Registers in Non-Secure Boot Mode

Typical Usage	Register	Event Log	Extended by	Content
Firmware Code	MRTD	NO	VMM: SEAMCALL [TDH.MR.EXTEND]	TD-Shim
Firmware Config	RTMR [0]	YES	TD-Shim: TDCALL [TDG.MR.RTMR.EXTEND]	N/A
OS code / Config	RTMR [1]	YES	TD-Shim: TDCALL [TDG.MR.RTMR.EXTEND]	vTPM Core
APP code/ Config	RTMR [2]	YES	vTPM Core: TDCALL [TDG.MR.RTMR.EXTEND]	N/A

NOTE: Putting SVN to RTMR is not useful in this case. If SVN is not part of vTPM Core, then SVN might be modified by the malicious entity. If SVN is part of vTPM Core, then the verifier must verify vTPM Core before SVN to ensure SVN is unmodified. It might be a hint, but SVN cannot be used alone without vTPM Core.

10.2 Secure Boot Mode

In Secure Boot Mode, TD-Shim is extended to MRTD. TD-Shim can follow secure boot policy to verify the signature of vTPM core and its SVN. The secure boot key is treated as firmware config and extended to RTMR[0]. The vTPM Core and its SVN are treated as OS code and config. They are extended to RTMR[1]. See **Error! Reference source not found.**.

With this approach, the verifier can get the vTPM core SVN. If TD-Shim and secure boot key are same and vTPM Core and SVN are updated, the verifier can verify the SVN.

Table 28: TD Measurement Registers in Secure Boot Mode

Typical Usage	Register	Even t Log	Extended by	Content
Firmware Code	MRTD	NO	VMM: SEAMCALL [TDH.MR.EXTEND]	TD-Shim
Firmware Config	RTMR [0]	YES	TD-Shim: TDCALL [TDG.MR.RTMR.EXTEND]	Secure Boot Key
OS code / Config	RTMR [1]	YES	TD-Shim: TDCALL [TDG.MR.RTMR.EXTEND]	vTPM Core, vTPM Core SVN
APP code/ Config	RTMR [2]	YES	vTPM Core: TDCALL [TDG.MR.RTMR.EXTEND]	N/A

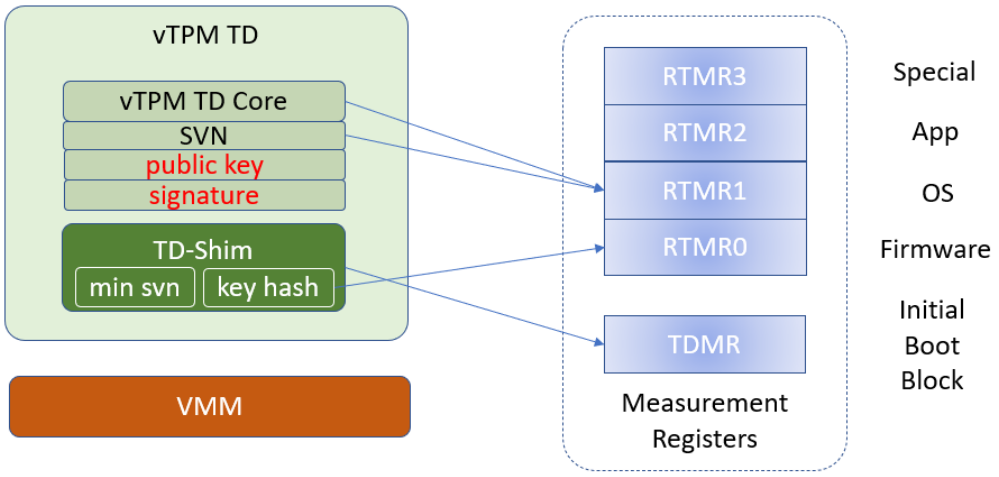

Figure 15: TD Measurement in Secure Boot Mode

§

11 vTPM TD Metadata

vTPM uses the same metadata defined as required for TDVF. BFV and payload are required to indicate the vTPM image. The temp memory (stack / heap) and permanent memory are required to indicate the memory to be used by vTPM. TD HOB is NOT required in the vTPM. VMM shall follow the entries in TDVF descriptor to create the vTPM memory one by one.

§

12 vTPM Migration

If the user TD is migratable, a vTPM TD shall be bound to the same Migration Service TD to support TDX live migration in TDX 1.5.

The orchestration layer can coordinate the live migration process for both user TD and vTPM TD. See below figures for a migration flow:

1. The orchestration layer finds the migration destination platform.

2. The orchestration layer migrates the user TD to the destination platform with the help of the Migration service TD.

3. The orchestration layer migrates the vTPM TD to the destination platform with the help of the Migration service TD.

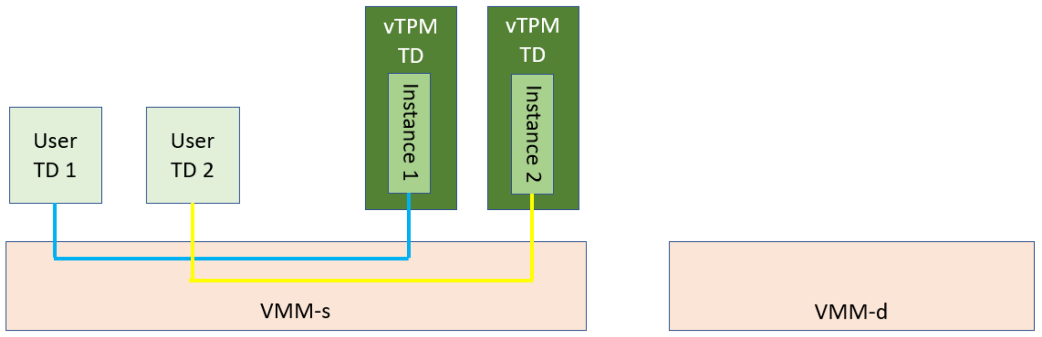

Figure 16: Migration - Initial

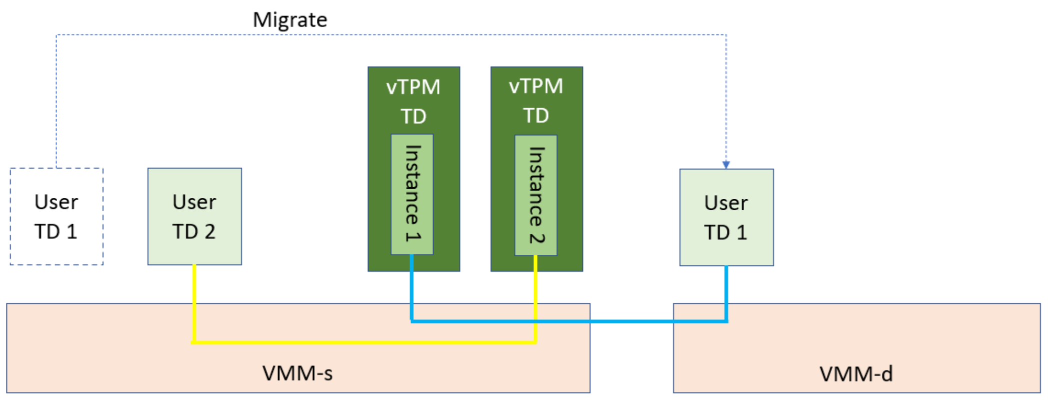

Figure 17: Migration - Middle

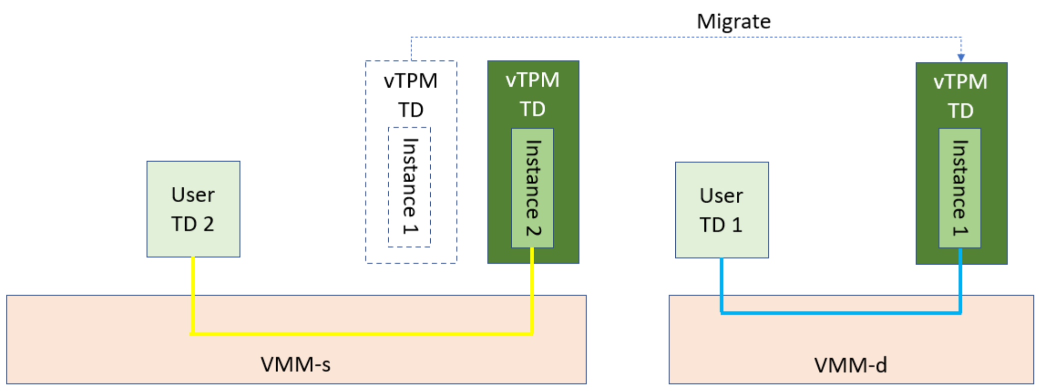

Figure 18: Migration - Final

§

13 vTPM Field Upgrade

A vTPM TD may support TPM field upgrade. In virtual TPM case, it means the vTPM TD is updated. The VMM needs to tear down the old vTPM TD and starts a new vTPM TD.

13.1 vTPM instance EK Cert update

vTPM instance EK includes the vTPM TD Quote.

If the vTPM TD restarts, the vTPM instance in the new vTPM TD will recreate a new EK Cert based upon the new vTPM TD Quote.

§

Appendix A Reference

A.1 Standards

[TCG VTPM] TCG Virtualized Trusted Platform Architecture Specification, https://trustedcomputinggroup.org/resource/virtualized-trusted-platform-architecture-specification/

[TPM2] TPM2 Library Specification, https://trustedcomputinggroup.org/resource/tpm-library-specification/

[TPM2 PP] TCG Protection Profile for PC Client Specification TPM2.0, https://trustedcomputinggroup.org/resource/pc-client-protection-profile-for-tpm-2-0/

[TPM2 EK] TCG EK Credential Profile for TPM2.0, https://trustedcomputinggroup.org/resource/tcg-ek-credential-profile-for-tpm-family-2-0/

[TPM2 PROVISION] TCG TPM2.0 Provisioning Guidance, https://trustedcomputinggroup.org/resource/tcg-tpm-v2-0-provisioning-guidance/

[TPM2 KEY] TPM2.0 Keys for Device Identity and Attestation, https://trustedcomputinggroup.org/resource/tpm-2-0-keys-for-device-identity-and-attestation/

[TCG PTP] TCG PC Client Platform TPM Profile Specification, https://trustedcomputinggroup.org/resource/pc-client-platform-tpm-profile-ptp-specification/

[TCG PFP] TCG PC Client Platform Firmware Profile Specification, https://trustedcomputinggroup.org/resource/pc-client-specific-platform-firmware-profile-specification/

[TCG EFI] TCG EFI Protocol Specification, https://trustedcomputinggroup.org/resource/tcg-efi-protocol-specification/

[TCG ACPI] TCG ACPI Specification, https://trustedcomputinggroup.org/resource/tcg-acpi-specification/

[TCTI] TCG TSS 2.0 TPM Command Transmission Interface (TCTI) API Specification, https://trustedcomputinggroup.org/resource/tss-tcti-specification/

[TCG TAP] TCG TAP Information Model, https://trustedcomputinggroup.org/resource/tcg-tap-information-model/

[TLS 1.2] The Transport Layer Security (TLS) Protocol Version 1.2, https://datatracker.ietf.org/doc/rfc5246/

[TLS 1.3] The Transport Layer Security (TLS) Protocol Version 1.3, https://datatracker.ietf.org/doc/rfc8446/

[TLS Session Resumption] Transport Layer Security (TLS) Session Resumption without Server-Side State, https://datatracker.ietf.org/doc/rfc5077/

[DTLS 1.2] The Datagram Transport Layer Security Version 1.2, https://datatracker.ietf.org/doc/rfc6347/

[DTLS 1.3] The Datagram Transport Layer Security Version 1.3, https://datatracker.ietf.org/doc/rfc9147/

[SPDM] Security Protocol and Data Model, https://www.dmtf.org/dsp/DSP0274

[Secure SPDM] Secured Message using SPDM specification, https://www.dmtf.org/dsp/DSP0277

[TDVF] Intel TDX Virtual Firmware Design Guide, https://software.intel.com/content/www/us/en/develop/articles/intel-trust-domain-extensions.html

[GHCI] Guest Hypervisor Communication Interface Spec, https://software.intel.com/content/www/us/en/develop/articles/intel-trust-domain-extensions.html

[MigTD] Intel TDX Migration TD Design Guide, https://software.intel.com/content/www/us/en/develop/articles/intel-trust-domain-extensions.html

A.2 Web Resources

[TPM2 ms-tpm-20-ref] Microsoft TPM2.0 Reference, https://github.com/microsoft/ms-tpm-20-ref

[TPM2 libtpms] Linux TPM2.0 Reference, https://github.com/stefanberger/libtpms

[TPM2 Software] TPM2.0 Software Community, https://tpm2-software.github.io/

[TPM2 Tutorials] TPM2.0 Software Tutorials, https://tpm2-software.github.io/tutorials/

[TPM2 Remote Attestation] https://tpm2-software.github.io/tpm2-tss/getting-started/2019/12/18/Remote-Attestation.html

[TPM2 Remote Attestation tool] https://tpm2-software.github.io/2020/06/12/Remote-Attestation-With-tpm2-tools.html

[TPM Remote Attestation] https://safeboot.dev/attestation/

[Intel RA-TLS] Intel Remote Attestation TLS, https://github.com/cloud-security-research/sgx-ra-tls

[Open Enclave RA-TLS] open enclave Remote Attestation TLS, https://github.com/openenclave/openenclave/tree/master/samples/attested_tls, https://github.com/openenclave/openenclave/blob/master/include/openenclave/attestation/attester.h, https://github.com/openenclave/openenclave/blob/master/include/openenclave/attestation/verifier.h

[Illustrated TLS Connection] https://tls12.xargs.org/, https://tls13.xargs.org/

[Linux Kernel TLS] https://docs.kernel.org/networking/tls.html, https://docs.kernel.org/networking/tls-offload.html

[Linux in-kernel TLS] https://lwn.net/Articles/892216/, https://lwn.net/Articles/896746/

[Keylime] https://next.redhat.com/project/keylime/

[SVSM-vTPM] https://github.com/svsm-vtpm

[Occlum-eTPM] https://github.com/inclavare-containers/confidential-vtpm

§

Appendix B vTPM Solution in Orchestrator

B.1 Overview

The cloud orchestrator needs to manage the vTPM instance lifecycle, such as vTPM instance creation, assignment, and destruction.

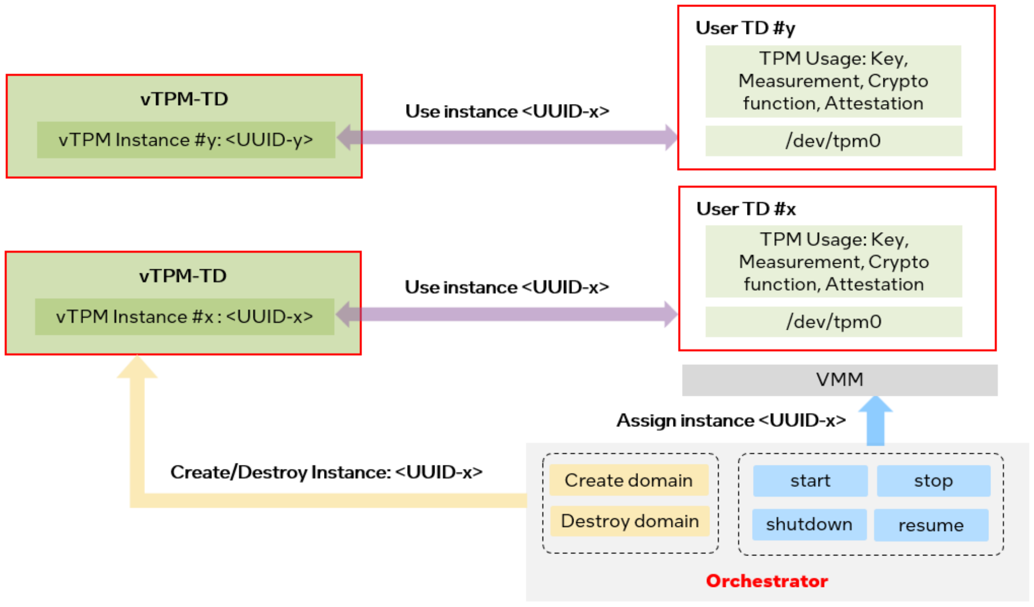
User TD #y
TPM Usage: Key, Measurement, Crypto function, Attestation
/dev/tpm0
vTPM-TD
vTPM Instance #y: <UUID-y>
Use instance <UUID-x>
User TD #x
TPM Usage: Key, Measurement, Crypto function, Attestation
/dev/tpm0
vTPM-TD
vTPM Instance #x : <UUID-x>
Use instance <UUID-x>
VMM
Assign instance <UUID-x>
Create/Destroy Instance: <UUID-x>
Create domain
Destroy domain
start
stop
shutdown
resume
Orchestrator

Figure 19: vTPM with Cloud Orchestrator

B.2 QEMU Command Line Interface

The following properties are added to QEMU's "tdx-guest" object:

Object	User TD	vTPM TD
vtpm-type	String: "client"	String: "server"
vtpm-id	2 types of vTPM ID are supported: 1. UUID, format is "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" 2. Normal string, maximum length 16.	Ignore.
vtpm-path	The vTPM TD socket connect to.	The socket listening on.

B.3 QMP Command for vTPM Instance

The following commands are added to manage the vTPM instance creation/destroy in QEMU Machine Protocol (QMP, https://wiki.qemu.org/Documentation/QMP).

B.3.1 Create vTPM instance

Command:

{ "execute": "tdx-vtpm-create-instance", "arguments": { "user-id": "test_id_str" } }

user-id: UUID in "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" formant or plain string

with maximum length 16.

Response:

```
{ "timestamp": {"seconds": 1680154289, "microseconds": 374070},
 "event": "TDX_VTPM_OPERATION_RESULT",
 "data": {
           "state": int64_t value,
           "user-id": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" } }
```

The response is standard QMP event response, plus the following additional fields:

- event: Fixed string "TDX_VTPM_OPERATION_RESULT"
- state: The error code, refer the table below for more details.
- user-id: UUID or plain string, same as the "user-id" field in Command.

B.3.2 Destroy vTPM instance

Command:

{ "execute": "tdx-vtpm-destroy-instance", "arguments": { "user-id": "test_id_str" } }

user-id: UUID in "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" format or plain string

with maximum length 16.

Response:

SAME as Response of Create vTPM instance

B.3.3 Error Code Table

Table 29: Error Code Table

Value	Description
0~0xFF	Following state field of "vTPM TD <Service.VTPMTD.ReportStatus> Command"
1001	QMP Command Invalid parameter For example, "user_id is not UUID" && "length > 16"
1002	QMP Communication layer error e.g: QEMU received the QMP command but failed to dispatch to vTPM TD, due to no enough memory or other issues.
1003	QMP Not supported e.g: Send the QMP command to QEMU which does NOT running vTPM TD

B.4 Basic Workflow

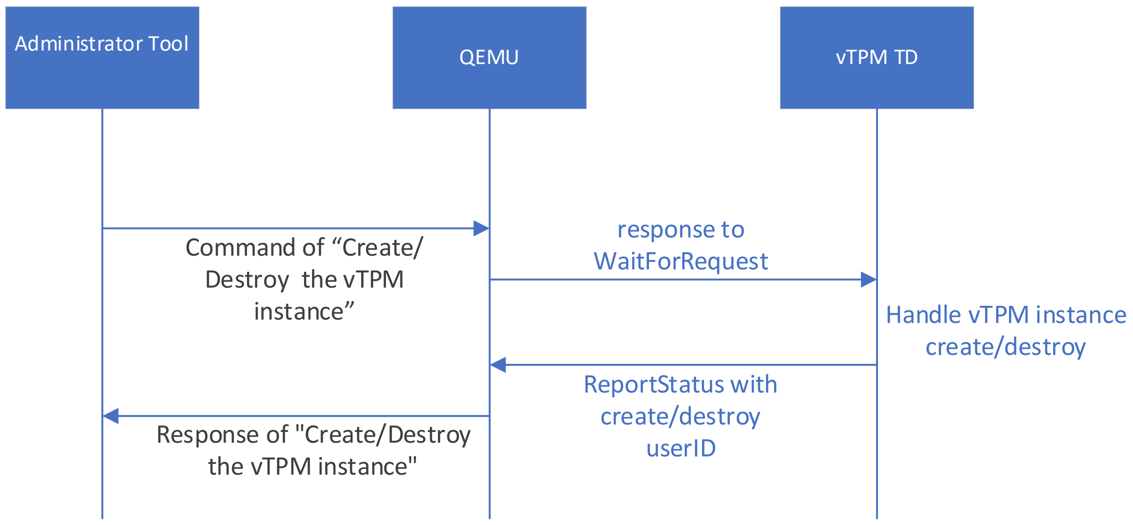
Administrator Tool
QEMU
vTPM TD
Command of "Create/ Destroy the vTPM instance"
response to WaitForRequest
Handle vTPM instance create/destroy
ReportStatus with create/destroy userID
Response of "Create/Destroy the vTPM instance"

Figure 20: vTPM Instance Management Flow

The Response of "Create/Destroy the vTPM instance" won't be sent until vTPM TD reported the result of "create/destroy vTPM instance" to QEMU, the response may be delay if vTPM TD/QEMU is in heavy stress scenario.

§

Appendix C vTPM Solution with Keylime

C.1 Overview

Keylime (https://next.redhat.com/project/keylime/) is a highly scalable remote boot attestation and runtime integrity measurement solution. The TDX vTPM solution can be integrated with keylime for guest TD attestation.

Below figure shows high level picture of keylime usage and connection in vTPM solution. The green part is the vTPM TD specific part.

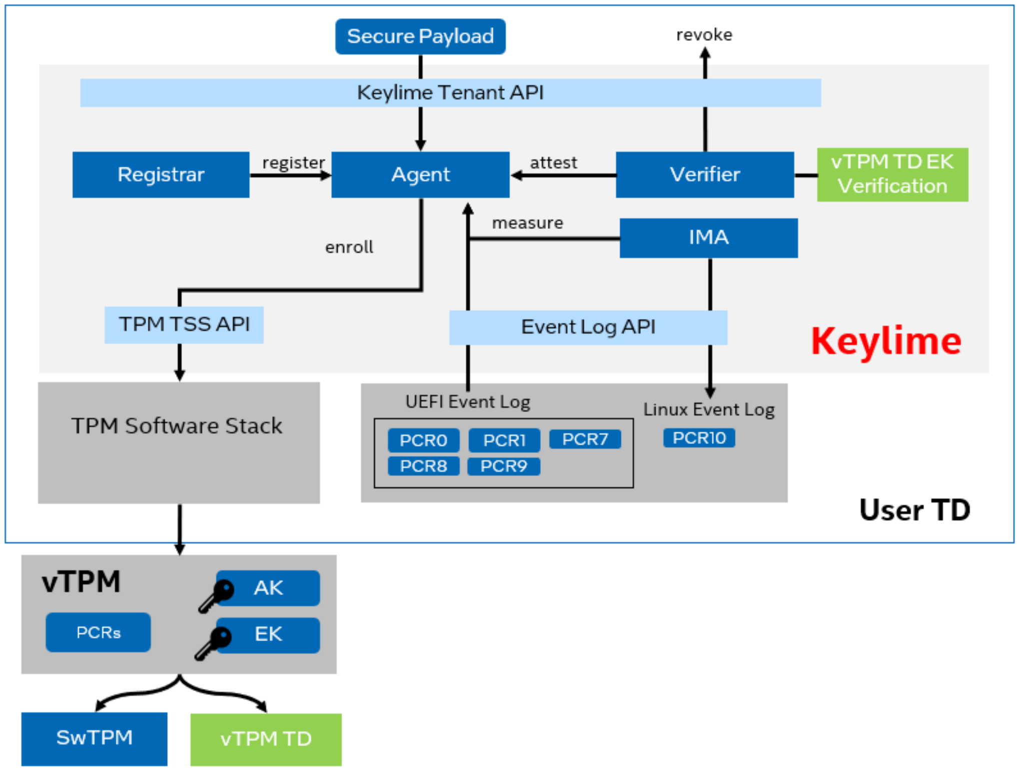

Figure 21: Keylime with vTPM Architecture

§