Device Attestation Model in Confidential Computing Environment

September 2022

Disclaimers

Table of Contents

Overview

Virtualization-based Trusted Execution Environments (TEEs) are used to host confidential computing workloads that are isolated from hosting environments, such as if the virtual machine monitor (VMM) is untrusted. In this document, such TEEs are referred to as Trusted Execution Environment VMs (TVMs) to distinguish them from traditional virtual machines (VMs).

When a portion of a device such as a TEE Device Interface (TDI) is assigned to a virtual machine (VM), the system must establish and maintain a trusted execution environment for the composition. A TDI is the unit of assignment for an IO-virtualization capable device or a Virtual Function (VF) when using single Root IO virtualization (SR-IOV).

In a host environment, we use TEE security manager (TSM) to indicate the logic trust computing base (TCB) component to enforce the security policy. The TSM could be inside of a TVM, or the TSM could be a component outside of the TVM and trusted by the TVM.

Inside the device, we use Device Security Manager (DSM) to indicate the logic TCB component to enforce the security policy. The TSM should set up a secure communication channel with the DSM to get the device information and manage the device interface.

This document describes a framework to establish a trust relationship between the TVM and the device with the help of TSM and DSM. As such, the device becomes a trusted device that can be included in the trust computing base (TCB) of the TVM. Specifically, the document describes mechanisms in the TVM to:

- Obtain the identity of the device, such as the device certificate.
- Obtain the evidence in the device, such as measurements of firmware and configuration.
- Report the verifiable evidence to an appraiser or verifier.
- Verify if the evidence represented by the identity follows the policy, such as matching the latest one.

This white paper focuses on how the verifier is built in the TVM, and how it can obtain information needed from the trusted device to perform the necessary actions described previously.

Note: The definition for “trusted” is different from “secure.” According to the Trusted Computing Group (TCG) definition, an entity can be trusted if it always behaves in the expected manner for the intended purpose. A trusted VM might not be a secure VM:

- If a VM has a known vulnerability that will expose a secret key, this VM is called a trusted VM but not a secure VM.
- Similarly, a trusted device might not be a secure device. If a device is known to use a fixed value when a true random number is required, this device is called a trusted device but not a secure device.

Defining a secure VM or a secure device is outside the scope of this document because the definition is highly dependent upon the security requirement.

The organization of this white paper is as follows:

- Chapter 1 will revisit the general attestation model.
- Chapter 2 will describe the device attestation model in TVM.
- Chapter 3 will describe the remote attestation to a TVM and devices by a third-party.
- Chapter 4 will discuss the mutual attestation flow - a device may want to attest a TVM in some special use case.
- Chapter 5 discusses the device firmware update support.
- Appendix A will map the method described in this white paper to the Intel TDX architecture.

1 General Attestation Model

Many standardization bodies are defining an attestation model, such as the National Institute of Standards and Technology (NIST), the Trusted Computing Group (TCG), and the Internet Engineering Task Force (IETF). Figure 1-1 shows the general attestation data flow defined by the IETF Remote Attestation Procedures (RATS) working group.

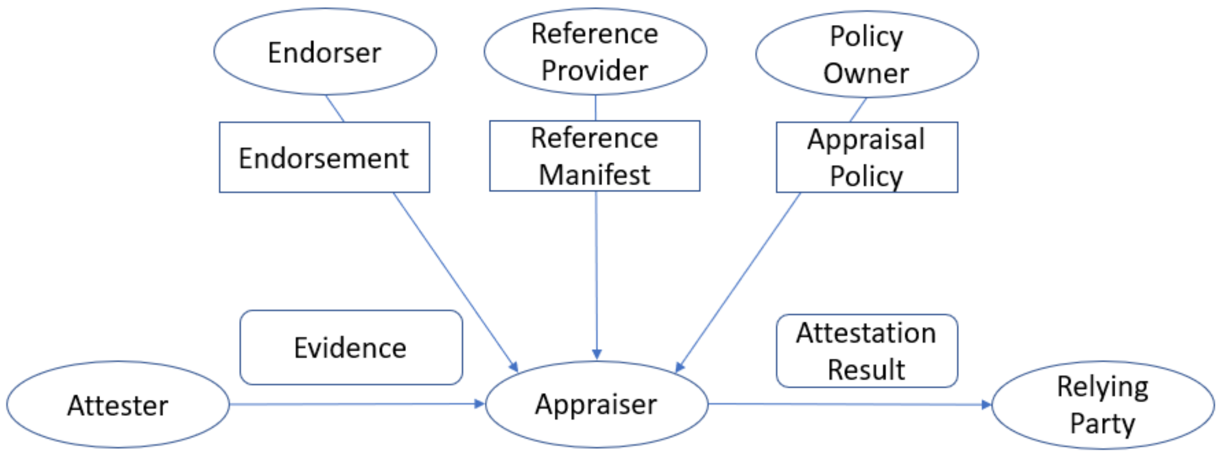

Figure 1-1: IETF RATS Attestation Data Flow

Let's clarify some terms here and give an example of platform attestation.

Role:

1) **Attester**: The entity that provides verifiable evidence to an appraiser. Example: a trusted platform module (TPM) and a platform firmware.
2) **Appraiser/Verifier**: The entity that compares available evidence with the reference manifest according to the appraisal policy. Example: an attestation agent in the operating system.
3) **Endorser**: The entity that provides the endorsement to help the appraiser verify the authenticity of the evidence. Example: a platform manufacturer.
4) **Reference Provider**: The entity that provides the reference manifest to help the appraiser compare with the evidence. Example: a platform manufacturer and a device manufacturer.

Artifact:

1) **Evidence/Measurement**: The property or the state of a component. Example: a set of TPM platform configuration registers (PCRs) and a TCG event log.

2) **Endorsement**: The security statement from the endorser. Example: a platform certificate.
3) **Reference Manifest**: The reference property or state of a component from the reference provider. Example: a set of platform reference measurement for PCRs or the TCG event log.
4) **Appraisal Policy**: A set of rules to inform the appraiser how to verify the evidence. Example: A) a rule that requires all PCR matches; B) a rule that requires a subset of PCR matches; C) a rule that requires a subset of TCG event log matches.

Attestation/Reporting: The presentation of verifiable evidence to an appraiser, which could be local attestation if the attester and appraiser roles are in same entity. It could also be remote attestation if the attester in one entity needs to convey the evidence to an appraiser in a different entity, such as a remote server. Example: using the TPM Quote command to report the platform configuration register (PCR) signed with the TPM attestation key.

2 Device Attestation

Now, we need to resolve the problem: If a device function is assigned to a TEE for the confidential computing workload, how do we attest the physical device? Here the term "assign" means that there is a trusted relationship between the TEE and the physical device, so the TEE can offload any confidential data to the device function - TEE Device Interface (TDI). The TDI could be a physical function (PF), a virtual function (VF), or an Assignable Device Interface (ADI).

Figure 2-1 shows an example of how to map figure 1-1 to the TVM device location attestation model.

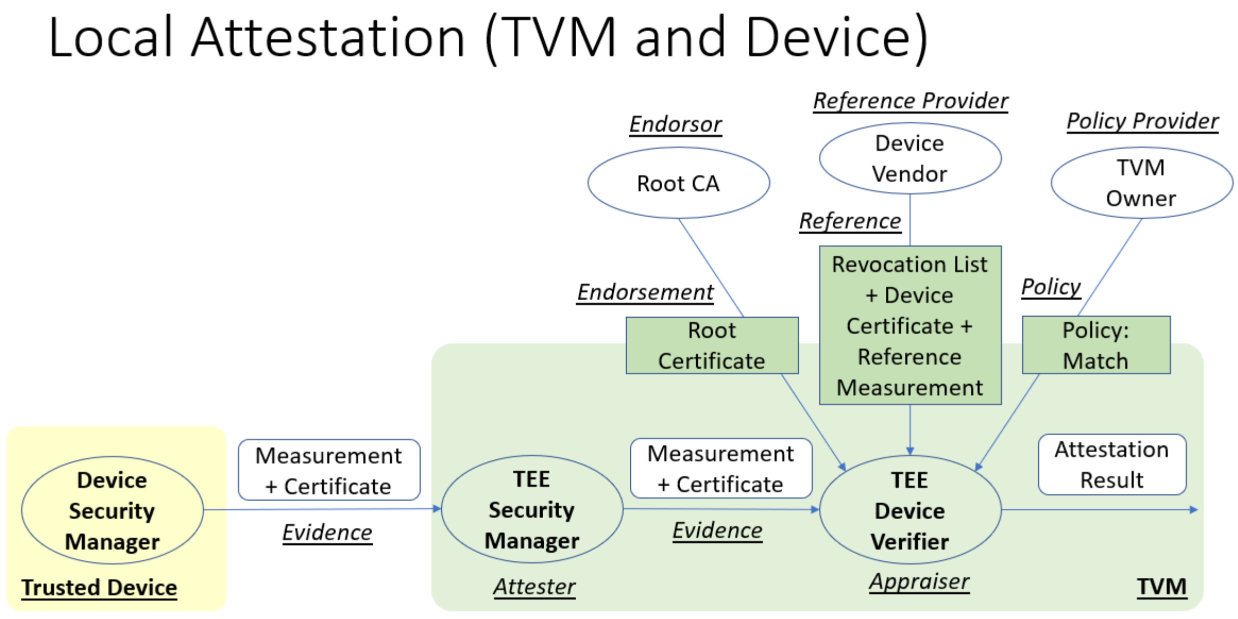

Figure 2-1: Local Attestation Model in TVM

The following are examples of roles and artifacts.

Role:

1) **Attester:** a Device Security Manager (DSM) and a trusted execution environment (TEE) security manager (TSM).
2) **Appraiser/Verifier:** a trusted execution environment (TEE) local verifier.
3) **Endorser:** a device root certificate authority (CA), or a device manufacturer.
4) **Reference Provider**: a device manufacturer.

Artifact:

1) **Evidence/Measurement**: a set of measurements from the device, the DICE certificate, and the evidence manifest.

2) **Endorsement**: a device root CA certificate, or a device certificate chain.
3) **Reference**: a device reference measurement and reference manifest.
4) **Appraisal Policy**: a rule that requires all measurements to match so that any device update will cause verification failure.

Attestation/Reporting Example: using the Secure Protocol and Data Model (SPDM) GET_MEASUREMENTS command to report device measurements with a digital signature signed with the device private key. For the Device Identity Composition Engine (DICE) device, the signing key is a device alias key.

The endorsement, reference manifest, and appraisal policy can be built in the TVM/TSM or be input at runtime. If they are input at runtime, these artifacts shall be measured to a TEE measurement register (MR) as part of TEE attestation.

Device Evidence and Evidence Reporting

Distributed Management Task Force (DMTF) defines a Secure Protocol and Data Model (SPDM) that can be used to identify a device, authenticate a device, and collect device measurements.

A TSM may use the SPDM GET_MEASUREMENTS command to collect device measurements. This can include measurements for the immutable ROM, mutable firmware, hardware configuration (e.g. straps), firmware configuration (e.g. configurable policy), debug mode, boot mode, version / secure version number, etc.

The measurement is signed with the SPDM device private key. As such, the TSM uses the device public key to verify the integrity of the measurements record.

If the device does not support an asymmetric cryptography algorithm such as digital signature, the TSM may use the pre-shared key (PSK). The PSK shall be provisioned to the device and TSM with confidentiality and integrity consideration.

For example, only an authorized owner is allowed to provision the device PSK. The TSM should be encrypted at rest and only be decrypted when it is inside of TEE, or when the PSK is derived from secure communication between the verifier and the device.

A TSM may use the SPDM PSK_EXCHANGE command to authenticate the device and get the device measurements. This may be a summary measurement for all components, or a summary measurement for the TCB. The device returns the response data with Hash-based Message Authentication Code (HMAC). As such, the TSM uses its own PSK to verify the integrity of the message data.

If the device supports Device Identity Composition Engine (DICE), the device may also return a DICE certificate in the SPDM GET_CERTIFICATE command, where the DICE certificate includes DICE TCB information. A TSM verifies the digital signature of the DICE certificate with the device certificate.

If a certificate chain includes DICE TCB information, then the verifier shall use the measurement in the TCB information instead of the SPDM measurement. This is because in DICE use cases, the SPDM responder is the firmware layer and is not trusted when firmware reports its own measurement.

Table 2-1. Evidence and Collection (Example)

Evidence	Reporting	Device Capability	Cryptography
SPDM Measurements	SPDM GET_MEASUREMENTS with digital signature	SPDM with Asymmetric	Asymmetric
SPDM summary Measurements	SPDM PSK_EXCHANGE with HMAC	SPDM with Symmetric	Symmetric
DICE TCB Info	SPDM GET_CERTIFICATE with digital signature	DICE and SPDM with Asymmetric	Asymmetric

In order to describe the content of the measurements, the device may report a measurement manifest, which is an evidence manifest. The evidence manifest should be presented to the verifier together with the evidence.

Device Reference Integrity Manifest (RIM)

The device reference integrity manifest (RIM) shall be published by a device vendor, such as:

- a software ID (SWID) tag defined in ISO/IEC 19770-2 and NISTIR 8060;
- a concise SWID (CoSWID) tag defined by the ITEF Security Automation and Continuous Monitoring (SACM) working group; or
- a concise RIM (CoRIM) defined by the IETF Remote Attestation Procedures (RATS) working group.

The SWID-based RIM is defined by TCG in the RIM-Information model (IM), and the CoSWID based RIM, or CoRIM, is defined by RATS. The SWID tag uses Extensible Markup Language (XML) format, while the CoSWID tag, or CoRIM, uses JavaScript Object Notation (JSON) or Concise Binary Object Representation (CBOR) encoding. Both SWID and CoSWID can be used in platform attestation use cases according to the TCG RIM IM specification.

The “CoSWID” tag is designed to represent software components. However, a device may need to present some hardware component information. As such, the device may use a concise module identifier (CoMID) to describe the hardware component and provide a counterpart to CoSWID.

The software stored in a hardware module can be referred to as firmware. The firmware may be presented as CoSWID or a CoMID based upon the implementation. Both CoSWID and CoMID can be included in a CoRIM provided by the device manufacturer.

The device manufacture may choose to support either CoSWID or CoMID, or both. For a device with layered architecture, such as DICE, the CoRIM is a good choice. The support RIM may be a binary presentation, such as an SPDM Measurement Block returned by SPDM GET_MEASUREMENT, an SPDM Measurement Summary Hash returned by SPDM CHALLENGE/KEY_EXCHANGE/PSK_EXCHANGE, or DICE TCB info returned by SPDM GET_CERTIFICATE.

Device Appraisal Policy

Appraisal policy defines how the verifier verifies the evidence against the RIM. Different solutions may use different appraisal policies. Table 2-2 shows example appraisal policies.

Table 2-2. Appraisal Policy (Examples)

Case	Rule	Consequence	Use Case
1	All measurements shall match RIM.	Any device change will cause verification failure. Device change shall be together with appraisal policy update.	No update is allowed. (High assurance platform.)
2	Immutable ROM and mutable firmware measurement match RIM.	Device configuration change will not cause verification failure.	Configuration change is allowed.
3	Device TCB measurements shall match RIM.	A non-TCB device mutable firmware change will not cause verification failure.	Non-TCB change is allowed.
4	Device certificate chain shall match the trust anchor (root CA or intermediate CA).	Any device TCB or mutable firmware change will not cause verification failure.	Device firmware change is allowed.
5	Device root CA certificate shall match the device root CA certificate list.	Any device change or even device certificate reprovision will not cause verification failure, as long as the device certificate is signed by the root CA certificate.	Device identity change is allowed.

Open Policy Agent (OPA) defines a policy language called “Rego” that can be used to describe the appraisal policy. For example, the OPA Rego policy receives a nested JSON object input with three attributes (“evidence,” “endorsement,” and “reference”) as the input, and generates an attestation result in another JSON object as the output.

Device Identification and Authentication. Before attestation, the TSM needs to identify and authenticate a device. The SPDM GET_CERTIFICATE and CHALLENGE commands serve this purpose. Figure 2-2 shows an example.

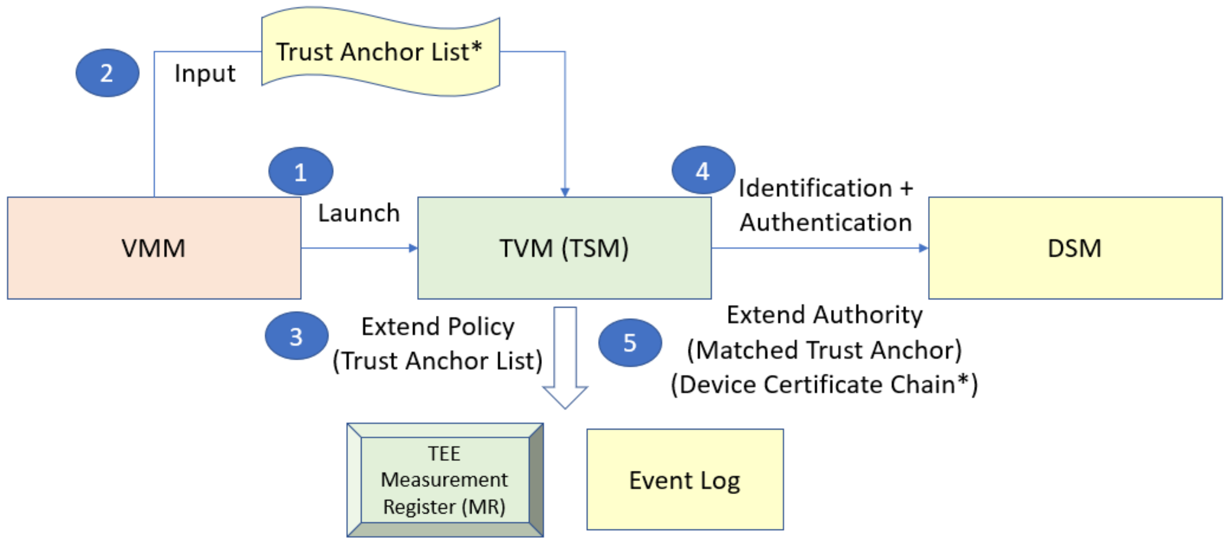

Figure 2-2: Device Authentication

Step 1: A VMM launches a TVM with a TSM. The TVM or TSM may or may not include a trust anchor list as part of endorsement, such as a root certificate list, or an intermediate certificate list. If the trust anchor certificate list is present, it shall be measured together with the TVM or TSM.

Step 2: The VMM may optionally input a trust anchor list to the TVM or TSM, as part of additional endorsement. This additional trust anchor list is not trusted and shall be measured to a TEE dynamic Measurement Register (MR).

Step 3: The TVM or TSM should extend the policy--trust anchor list into a TEE measurement register (MR) and record an event log. The MR could be a static MR to record the launching time state, or a dynamic MR to record TEE runtime state, based on when the policy is input. The MR and event log are presented as part of evidence for the whole TEE.

Step 4: The TSM uses the SPDM GET_CERTIFICATE command to retrieve the device certificate chain and verifies the certificate chain with the trust anchor list. If the verification passes, then the TSM uses the SPDM CHALLENGE command to authenticate the device.

The TSM generates a nonce and sends it to the device, then the device signs the data with the nonce using its private key and returns the data back. The TSM verifies the digital signature with the leaf certificate in the device certificate chain.

Step 5: The TVM or TSM should extend the authority-matched trust anchor and optional device certificate chain into the TEE MR and record an event log. Care must be taken that the trust anchor is the device "class" information, with all devices in this class having the same class information. And the device certificate chain may be the device "instance" information, with each device having its Unique Data Secret (UDS) information there. The "class" information and "instance" information should be put into different MRs.

PSK Based Authentication

A device may only support a symmetric cryptography algorithm, but not support asymmetric cryptography algorithms such as digital signature. In such case, the TEE may use pre-shared key (PSK) based authentication. Figure 2-3 shows an example.

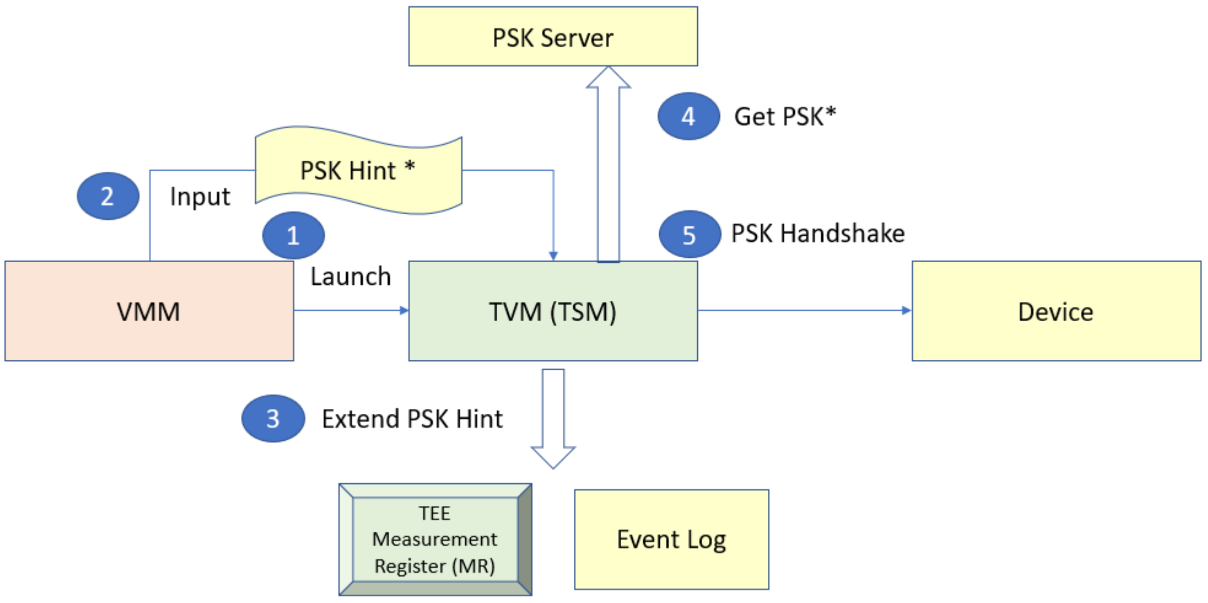

Figure 2-3: PSK based Authentication and Attestation

Step 1: A VMM launches a TVM with a TSM. The TVM or TSM may or may not include a PSK hint. If the PSK is stored in the device secure storage, the PSK hint may be used as an identifier to specify which PSK will be used. If the PSK is not stored in the device, the PSK hint may also be used to derive the PSK, for example, together with the unique device secret (UDS).

Step 2: The VMM may optionally input the PSK hint to the TVM or TSM.

Step 3: The TVM or TSM should extend the PSK hint into a TEE measurement register (MR) and record an event log.

Step 4: The TSM may have a PSK provisioned inside of the TSM. If the PSK is not provisioned, then the TSM may communicate with a PSK server to retrieve the PSK in a secure communication channel. Significantly, the PSK server shall do TEE authentication/attestation to ensure that the TSM is trusted before passing the PSK to the TSM.

Step 5: Now that the TSM has retrieved the PSK, it uses the SPDM PSK_EXCHANGE command to authenticate the device. The TSM generates a random number and sends it to the device, which performs HMAC on the random number with the device PSK and returns the data back. Then the TSM verifies the HMAC with the TSM PSK.

Device Local Attestation Flow

Now the TVM can perform local attestation for the device. Figure 2-4 shows an example.

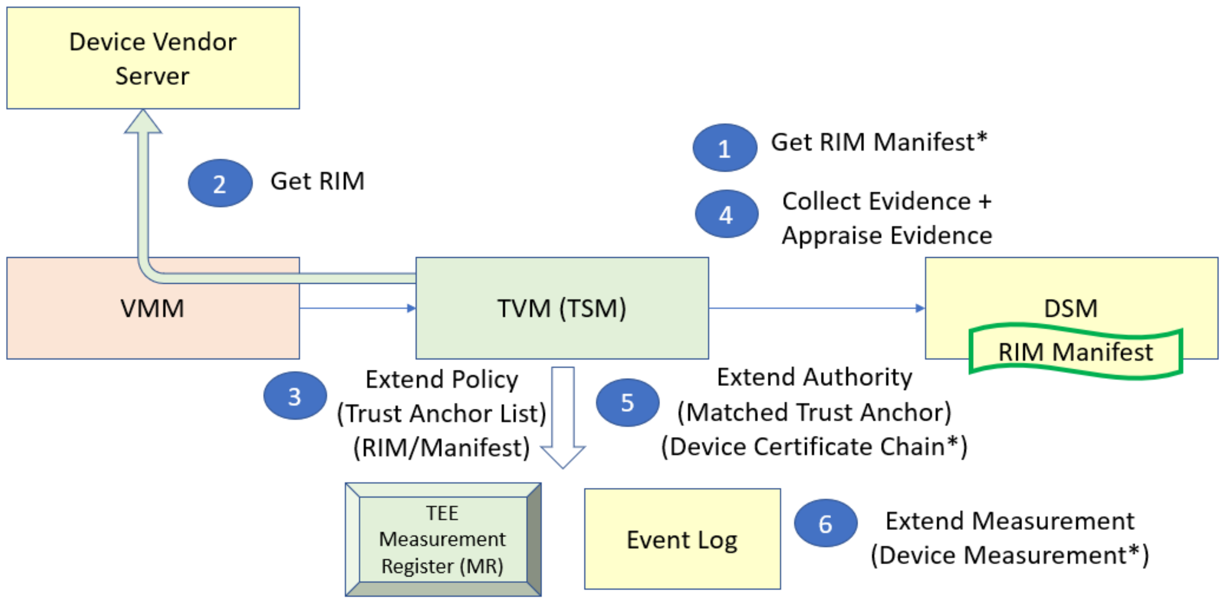
Device Vendor Server
2 Get RIM
1 Get RIM Manifest*
4 Collect Evidence + Appraise Evidence
VMM
TVM (TSM)
DSM
RIM Manifest
3 Extend Policy (Trust Anchor List) (RIM/Manifest)
5 Extend Authority (Matched Trust Anchor) (Device Certificate Chain*)
TEE Measurement Register (MR)
Event Log
6 Extend Measurement (Device Measurement*)

Figure 2-4: Device Local Attestation

Step 1: The TVM gets a RIM manifest. The RIM manifest may be included in a device, passed from the VMM, or embedded in the TVM.

Step 2: The TVM gets a device RIM according to the RIM manifest. Again, there are multiple possibilities. The device RIM may be received from the device vendor server at runtime, passed from the VMM, or embedded in the TVM.

Step 3: The TVM should extend the policy – device RIM into TEE MR and record an event log. If the appraisal policy is input, it shall also be extended. For example, a list of object identifiers (OID) to check with.

Step 4: Now the TSM collects the evidence from the device as discussed in the previous section. Such as the SPDM GET_MEASUREMENTS command for devices with asymmetric cryptography support, or the SPDM PSK_EXCHANGE command for devices with symmetric cryptography support only. TSM shall perform necessary integrity checks on the evidence (such as digital signature), to ensure that data is fresh and has not been tampered with. The TSM then passes the information to the verifier in the TVM. The verifier can appraise the evidence with the RIM collected with the appraisal policy we discussed in the previous section.

Step 5: When the verifier in the TVM checks the device identity, it should extend the authority, such as the device root certificate as a trust anchor to the TEE MR, and record it in the event log.

Step 6: The verifier in the TVM may extend the device runtime measurement to TEE MR and record the event log, together with device information such as device type, device location, vendor ID, device ID, class code, etc.

3 Device Remote Attestation

For a given TVM, a third-party remote verifier may also perform remote attestation for the TVM, TSM, and device, to verify if the TVM performed a trusted boot and verified the device. Figure 3-1 shows an example.

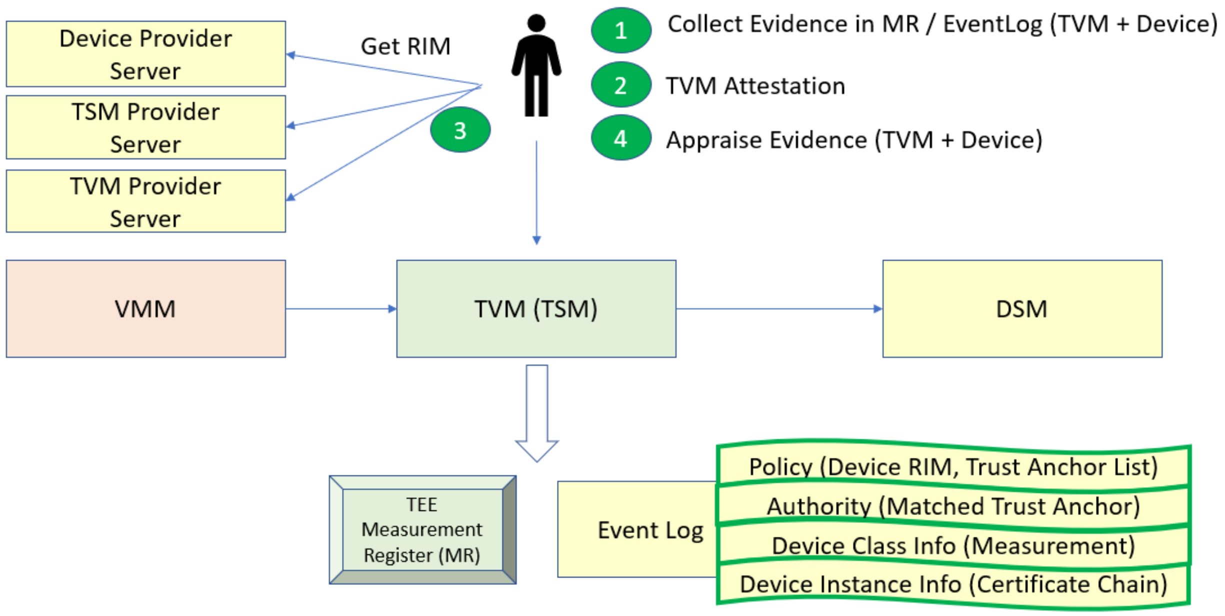
Device Provider Server
TSM Provider Server
TVM Provider Server
Get RIM
3
1
Collect Evidence in MR / EventLog (TVM + Device)
2
TVM Attestation
4
Appraise Evidence (TVM + Device)
VMM
TVM (TSM)
DSM
TEE Measurement Register (MR)
Event Log
Policy (Device RIM, Trust Anchor List)
Authority (Matched Trust Anchor)
Device Class Info (Measurement)
Device Instance Info (Certificate Chain)

Figure 3-1: Device Remove Attestation

Step 1: The remote verifier collects the evidence in the TEE measurement register (MR) and event log from the TEE attester. The MR and event log include the information on the TVM itself, the TSM, and the trusted device used in the TVM. This can include the policy (device RIM, trusted anchor list), the authority (matched trust anchor), device class information (measurement), and device instance information (certificate chain).

Step 2: The remote verifier performs a remote attestation for the MR and event log. This step is TEE-specific. For example, this could be a TEE quote verification.

Step 3: The remote verifier gets the reference integrity manifest (RIM) from TVM provider, TSM provider, and the device provider. The TVM RIM, TSM RIM, and device RIM are provided by different vendors. The TSM RIM and device RIM may be referred to in the TVM RIM, based on the TVM policy.

Step 4: The remote verifier may have its own appraisal policy for the TVM, TSM, and the trusted device.

4 Device Mutual Attestation

Attestation is performed because a TEE needs to verify that a device before the TEE decides to offload the workload to the device. In most cases, this is good enough. However, a special device may want to verify the TEE or the host platform to decide if the device wants to accept the offloading.

For example, a Cloud Service Provider (CSP) can build a special accelerator and only accept the workload from a set of special CSP specific service TDs or host platform. This can be achieved by mutual authentication.

Mutual Authentication in Secure Session

Mutual authentication (also known as two-way authentication) means that both entities need to authenticate each other at the same time in an authentication protocol before the secure session is established. For example, the network transport layer security (TLS) protocol or the SPDM protocol support mutual authentication.

When a TVM authenticates the device, the device can also authenticate the TVM. The device needs a way to identify the TVM and verify the TVM information, such as a certificate, TEE measurement, or TEE quote. The SPDM Key Exchange process (KEY_EXCHANGE + FINISH command) with encapsulated GET_CERTIFICATE can be used to support the mutual authentication. Figure 4-1 shows an example.

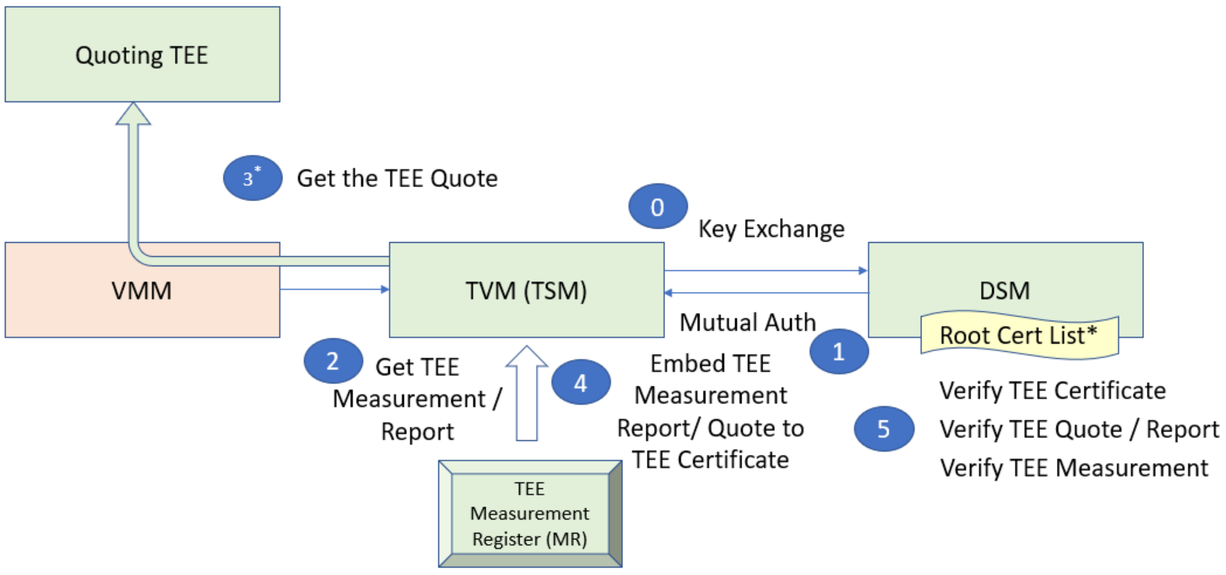
Quoting TEE
3*
Get the TEE Quote
0
Key Exchange
VMM
TVM (TSM)
DSM
Mutual Auth
Root Cert List*
2
Get TEE Measurement / Report
4
Embed TEE Measurement Report/ Quote to TEE Certificate
1
Verify TEE Certificate
5
Verify TEE Quote / Report
Verify TEE Measurement
TEE Measurement Register (MR)

Figure 4-1: TEE Authentication

Step 0: A TSM can initiate the SPDM Key Exchange process to set up a secure session with the device.

Step 1: The device returns SPDM Key Exchange with a mutual authentication flag to indicate that mutual authentication is required to set up the secure session. Then the DSM sends the SPDM GET_CERTIFICATE command to collect the TVM information. The TSM will start the Attested SPDM flow. We will discuss the general Attested SPDM in more detail in the next section.

Step 2: The TSM collects the TEE measurement register (MR) in a TEE report from the CPU. The TEE report may include both TEE and host platform information, such as CPU version or secure version number (SVN).

Step 3: The TSM may optionally get the TEE quote based on the TEE MR and report. A special quoting TEE can help do that work. In some scenarios, a TEE quote must be used because the TEE report may only be verified in the local machine. Only a TEE quote can support remote attestation.

Step 4: Once the TSM gets the TEE quote, the TSM can put the information into a TEE certificate with a special Object ID (OID). The TEE certificate can be a self-signed ephemeral certificate generated at runtime. Then the TSM returns the TEE certificate with the embedded TEE quote to the device. Technically, the TSM can embed any data that is required from the device based on the policy, such as the TEE event log.

Step 5: After the device gets the TEE certificate, the device obtains the TEE quote from the TEE certificate. The device should verify the TEE certificate, TEE quote, TEE report, TEE measurement, CPU SVN, or any TEE specific data or platform specific data based on its policy.

Only after the verification pass can the device continue the secure session setup flow with the TSM.

Attested SPDM

The flow outlined previously is not a traditional SPDM mutual authentication flow. A variant called Attested SPDM, which is similar to Attested TLS, is an SPDM session that integrates the remote attestation validation as part of the SPDM session establishment. Once established, the Attested SPDM guarantees that an attested connecting party is running inside a TEE with expected identity.

In the traditional SPDM flow, both entities use an X.509 certificate chain as their identity information. This certificate chain is transferred to the peer via an SPDM CERTIFICATE response. One entity can then verify the digital signature from the other one by using the public key in the leaf certificate. Figure 4-4 shows the SPDM mutual authentication flow.

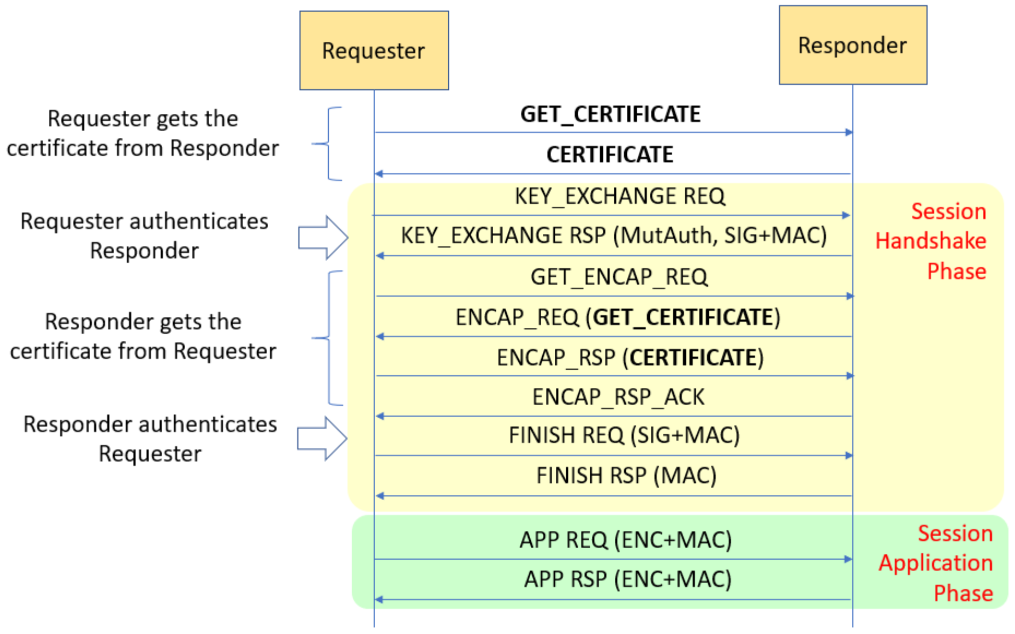
Requester
Responder
Requester gets the certificate from Responder
GET_CERTIFICATE
CERTIFICATE
Requester authenticates Responder
KEY_EXCHANGE REQ
KEY_EXCHANGE RSP (MutAuth, SIG+MAC)
Session Handshake Phase
GET_ENCAP_REQ
Responder gets the certificate from Requester
ENCAP_REQ (GET_CERTIFICATE)
ENCAP_RSP (CERTIFICATE)
ENCAP_RSP_ACK
Responder authenticates Requester
FINISH REQ (SIG+MAC)
FINISH RSP (MAC)
APP REQ (ENC+MAC)
APP RSP (ENC+MAC)
Session Application Phase

Figure 4-4: SPDM Mutual Authentication Flow

The Attested SPDM uses a transient self-signed X.509 certificate that represents the identity. This certificate is bound to the specific TEE via TEE-specific information such as a TEE attestation quote. Figure 4-5 shows one example.

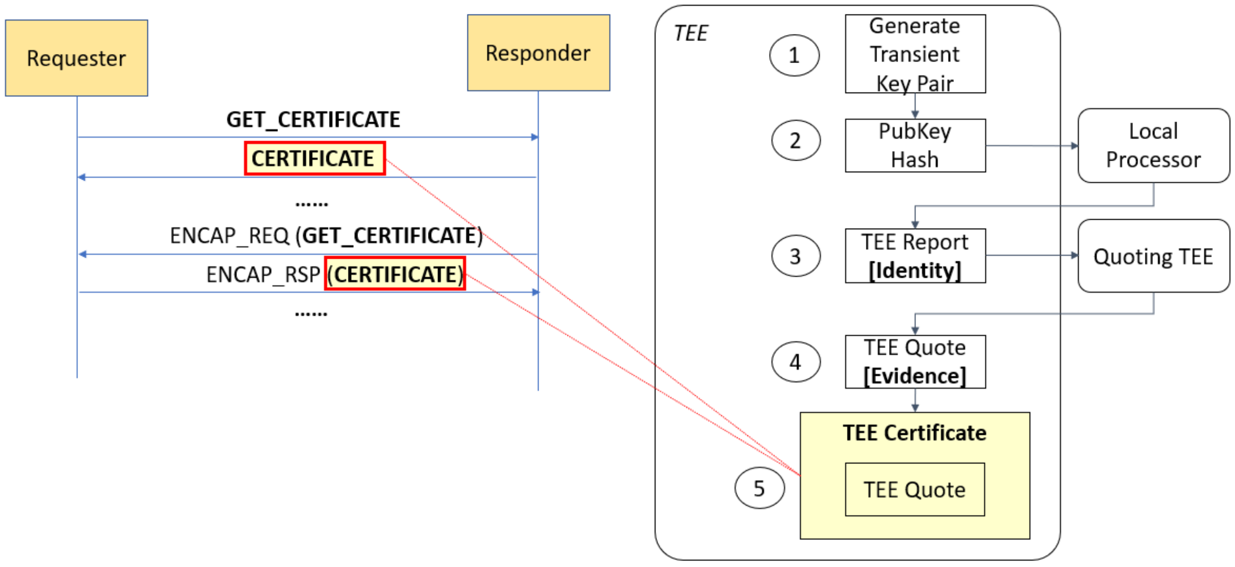

Figure 4-5: Attested SPDM Mutual Authentication Flow

The TEE certificate can be generated with the following flow outline:

Step 1: When a TEE needs to provide a TEE certificate, the TEE generates a transient self-signed public/private keypair.

Step 2: The TEE calculates the hash of the public key and uses the public key hash as input to get the TEE report. The TEE report includes the measurement of the TEE, and host platform information, such as the CPU SVN. The TEE report can be used as TEE identity.

Step 3: The TEE sends the TEE report to a quoting TEE to get the TEE quote. The TEE quote is used as TEE evidence.

Step 4: The TEE embeds the TEE evidence (TEE quote) into the TEE certificate with a specific Object ID (OID).

Step 5: The TEE can send the TEE certificate to the peer.

The peer should verify the TEE certificate per the following:

Certificate Validation

The verifier checks the signature of the self-signed TEE certificate to confirm whether the attestation TEE evidence (TEE quote) is genuine and unmodified.

Evidence Validation

The verifier extracts the TEE evidence from the TEE certificate and performs TEE evidence validation, such as TEE quote verification based upon an evidence verification policy.

Identity Validation

The verifier validates the TEE identity (TEE measurement, SVN) based upon an identity verification policy, such as the reference measurement or SVN.

Host Platform Validation

The verifier may validate the host platform information, based on a platform verification policy, such as the CPU SVN, or platform certificate.

Only after all verifications are passed can both entities know that the peer is the intended party.

Two-Phase Authentications

In the mutual authentication model, the device only authenticates the TSM because it is the TSM that sets up the communication with the device. The device cannot authenticate a full TEE, because the full TEE might not be ready at that moment.

If a device wants to authenticate a full TEE, then we may use two-phase authentications.

1) Phase 1: The TSM attests the device and provides the evidence to the TEE. Then the TEE relies on a policy to decide if it can accept the device. This is discussed in Chapter 2 and Chapter 3.
2) Phase 2: After a TEE is fully launched, the device collects evidence from the TEE and attests it. Or the TEE can collect the evidence by itself and present that to the device.

The device relies on its own appraisal policy to determine if the device accepts the TEE. This could be a normal TEE attestation flow such as a TEE Quote, or another mechanism such as password-based authentication, token-based authentication, etc.

Care must be taken that the two-phase authentication is bound in one secure session. For example, phase 1 is the one-way authentication secure session, and phase 2 happens inside of this secure session.

5 Device Firmware Update

A trusted device may support a firmware update. The device root-of-trust for update (RTU) shall follow NIST SP800-193 to ensure that the new firmware image is always authenticated, and the new image has an equal or higher security version number (SVN) to prevent a rollback attack.

A typical device includes hardware immutable read-only memory (ROM) and mutable firmware on the non-volatile (NV) storage, such as NOR flash or NAND flash, and runs firmware code in the system random-access memory (RAM), such as static RAM (SRAM) or dynamic RAM (DRAM). The immutable ROM is not updatable. The mutable firmware in NV storage and system RAM is updatable. Let's clarify some terms for device updates.

- **Firmware NV Storage Update**: the device updates the mutable firmware on the NV storage. This will not touch the runtime firmware code in the RAM. The NV storage change might not take effect until the device performs a reset.
- **Firmware Update Activation**: the device updates the runtime firmware code in the RAM after firmware NV storage is updated.
    - **Activation After Reset**: the device triggers a reset to finish activation, such as a device function reset, or a device system reset. This is usually with rolling firmware support.
    - **Runtime Activation**: the device does not need to trigger a reset to finish activation. The system running state is kept during activation.
- **Firmware Live Patch**: the device updates the runtime firmware code in the RAM without updating NV storage. However, a device reset will discard those patches and roll back the firmware into the old state.
    - **Live Patch with Activation**: the device is required to trigger an activation to make the new patched image take effect.
    - **Live Patch without Activation**: the device uses the new patched image immediately.
- **Firmware Update Lock**: the device is configured to reject any firmware update request until it is unlocked.

A traditional device may only support the firmware NV storage update and firmware update activation after a device reset. Some devices may be allowed to activate the new image at runtime without a device reset. Some devices may support live patching, where part of the firmware may be patched without reset. This no-reset-update feature may be used in data center cloud VMM environment, where the TVM may keep a secure connection to a virtual device current when the physical device is patched.

Runtime Update Control Policy

A device may choose how to support a runtime update with or without reset, based on device capability and policy.

- **SPDM - Measurement Freshness Capability (MEAS_FRESH_CAP)** is defined in the SPDM 1.0 specification. This indicates whether the device is able to return a fresh measurement as follows.
  - **No Fresh Measurement**: the device only returns the measurements computed during last time reset.
  - **Fresh Measurement**: the device can recompute all measurements and always return the fresh measurements.
- **SPDM - Session Policy (TerminationPolicy)** is defined in the SPDM 1.2 specification. When a secure session is created between the device and the TSM, the TSM can indicate that a session termination policy after the firmware is updated:
  - **Unconditional Termination**: the device shall unconditionally terminate the session when the runtime update has taken effect.
  - **Device Controlled Termination**: the device is allowed to decide whether to keep the session or terminate it based on the device policy. For example, a session is kept when SVN is the same or terminated when SVN is updated.
- **Trusted Device Interface Security Protocol (TDISP) - Interface Lock Policy (NO_FW_UPDATE)** is defined in the PCI-SIG TDISP specification. When a TSM requests the device to lock the interface, the TSM can indicate the firmware update policy:
  - **No Firmware Update**: the device is not allowed to update firmware. The TSM shall stop the interface, unlock the device, then perform the firmware update.
  - **Firmware Update**: the device is allowed to update firmware at runtime.

In a usual confidential computing environment with device support, a VMM is the resource manager and decides the final update control policy (allow runtime update or not) for the device. The TSM is policy enforcement and reports the final decision to the TVM honestly. The TVM verifies the final update control policy for the device, and decides whether it is acceptable or not.

Ideally, the TVM should provide the preferred update control policy to the VMM as a hint, such as in a TEE manifest. If there are multiple TVMs requiring different update control policies, the VMM may carefully isolate them and let them attach to different devices.

If a TVM wants to support a runtime update without SPDM session termination, then the TVM shall let the TSM or TSM-RoT choose “Firmware Update” in the TDISP Lock Policy and “Device Controlled Termination” in the SPDM session policy. See Table 5-1.

Table 5-1. Runtime Update Control Policy and Impact

SPDM Session TerminationPolicy	TDISP NO_FW_UPDATE	Result

0 (Unconditional Termination)	0 (Firmware Update)	No runtime update support. An update will cause session termination.
0 (Unconditional Termination)	1 (No Firmware Update)	No runtime update support. An update will cause session termination before device interface lock. Any update is not allowed after device interface lock.
1 (Device Controlled Termination)	0 (Firmware Update)	Runtime update is allowed. An update will cause session termination or session alive. This is controlled by the device.
1 (Device Controlled Termination)	1 (No Firmware Update)	Runtime update is allowed before device interface lock. An update will cause session termination or session alive before device interface lock. This is controlled by the device. Any update is not allowed after device interface lock.

Secure Update Policy

NIST SP800-193 defines the Platform Firmware Resiliency Guidelines. The firmware update should prevent a rollback attack. Usually this is achieved by verifying a secure version number (SVN). Please note the following term definitions.

- **Secure Version Number (SVN)** is a special version to indicate the security property of the firmware. See Figure 5-1.
    - **Birth SVN** is the SVN in the device at launch. This is a fixed value and cannot be changed even after a runtime firmware update. It may only be changed after a device reset.
    - **Current SVN** is the SVN for the current running firmware. The current SVN shall always be bigger than or equal to the birth SVN. This is a dynamic value. The current SVN becomes the update SVN after a runtime update.
    - **Update SVN** is the SVN in the firmware update package. This is a fixed value for the update package.

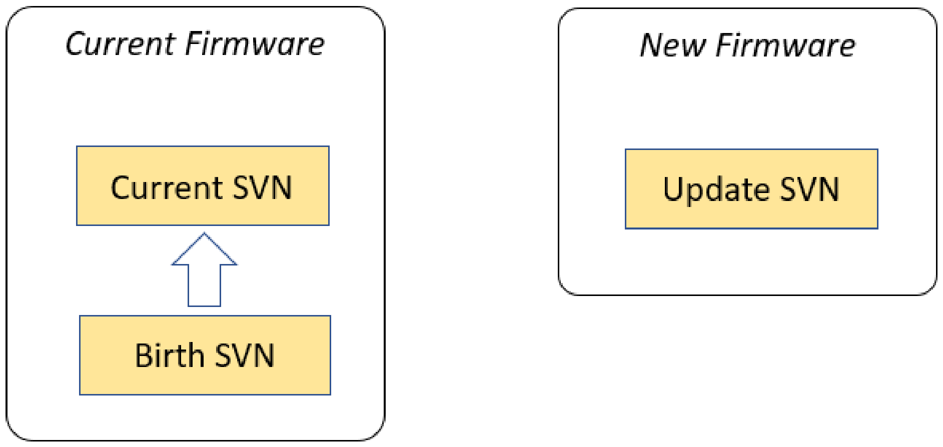

Figure 5-1: Device Firmware SVN

- **Rollback**: Ideally, a new firmware version should be higher than the current firmware version. In some special cases, if the latest firmware version is found to be having stability issues, then the owner of the platform or the device may choose to roll back the firmware to an older version.
    - The best practice is that firmware should only be allowed to perform a functional rollback, but not a security rollback. Rollback protection means that the attacker cannot install an old firmware with known vulnerabilities. As such, it is of great importance to protect the current SVN storage. Usually, it is in tamper-proof non-volatile storage. Only an authorized entity is allowed to update the SVN.
- **SVN Commit** is a special step to update the current SVN to the update SVN after a runtime firmware update. There are two ways to commit SVN.
    - **Immediate SVN Commit**: after runtime firmware update, the current SVN is changed to the update SVN immediately. In this mode, the firmware may be updated to an older version, but it cannot be updated to an older SVN.
    - **Delayed SVN Commit**: the current SVN is NOT changed to the update SVN immediately after a firmware update, as the firmware may be in a middle state to do some device-specific function self-test to ensure that the new firmware image is good enough. After the self-test, the update SVN is committed to the current SVN. This mode can help the firmware rollback in case of functional issues in any new firmware image, even if the SVN is higher in the new image.

With the definitions above, we can define the **SVN Update Policy**, which describes how to control the SVN checking.

- **Birth SVN Mode**: the device uses the policy “**Update SVN >= Birth SVN**” to verify the update package. As such, partial rollback (update SVN < current SVN) is acceptable. With this mode, a device can roll back to an old image if the new update image has function issues. However, it might not be a secure solution.

- **Incremental SVN Mode**: the device uses the policy "**Update SVN >= Current SVN**" to verify the update package. As such, no rollback is acceptable. This is the best practice. If the device has function issues after a runtime update, it needs another way to roll back to an old image – for example, delaying the SVN commit until after function verification.

People should understand the Pros and Cons for the two SVN Update Policies.

Evidence Reporting Policy

After the device is updated, the device may need to report the new identity (certificate) and the new measurement information. The device may choose an **evidence reporting policy** that describes how the device reports the new identity and new measurement.

- **Birth Mode:** the device does not generate a new identity or measurement. It still reports the initial identity and measurement at launch. The initial measurement could be the Secure Version Number (SVN). This is known as create time SVN.
- **Refresh Mode:** the device generates the current identity and measurement based on the current firmware image. It reports the current identity and measurement. The current SVN is known as report time SVN.
- **History Mode (Ledger Mode)**: the device generates a new identity and measurement based upon the new firmware image, and it also keeps a copy of the old data. It reports the new identity and measurement, as well as the old measurements as the history data, which is also known as a ledger. A simple ledger may only include the initial measurement and latest measurement. A full ledger may include the measurements of all updated firmware versions.

This evidence reporting policy is similar to the SPDM Measurement **Freshness Capability** (No Fresh Measurement vs. Fresh Measurement), and the concept is extended to the device certificate.

Some devices may implement Device Identity Composition Engine (DICE). In this case, the leaf certificate is an alias that contains the device identity information and the firmware measurement (such as firmware hash). If a layer (N) firmware is updated, then its alias certificate shall be regenerated by layer (N-1) firmware. A reset may be required in this case because the layer (N-1) firmware might not exist anymore. This is the **Refresh Mode**.

The TSM shall also use SPDM GET_CERTIFICATE to get the latest device certificate chain and SPDM GET_MEASUREMENT to get the latest device measurement. Then the TSM verifies them against the appraisal policy. Care needs to be taken that the device reset happens in a very short time period (such as 10 microseconds) to keep the hardware bus communication alive. Otherwise, the device might be considered in an error state and the hardware communication is broken.

The new device measurement record may keep adding the hash of the new firmware as the new record and keep the old record as is. As such, the update history is maintained. This is the **History Mode (Ledger Mode)**.

History Mode is the best way to provide all information to a verifier, because the verifier can determine whether an intermediate vulnerable version has been used and updated later. Without the history, this information is missing.

A device may only have a limited measurement record index. For example, the TPM specification only defines 24 platform configuration registers (PCRs) and the SPDM specification defines 254 measurement indexes. As such, the device may use the hash extend function to measure all data into one dedicated runtime update measurement slot, and use an event log to record all history data.

If a device is not DICE compatible and it does not support SPDM Fresh Measurement capability, then it may use the **Birth Mode** to support runtime update. However, the TSM needs be aware that the collected evidence might NOT match the running firmware.

The TSM needs to trust that the DSM always updates the device firmware with better or at least equal security by performing an SVN check. However, the verifier has no way to know the current value or history.

Besides evidence, the TSM and TVM may collect other device information. For example:

- **TDISP – CAPABILITIES** includes the capabilities supported by the device, such as DSM capability flags, type of TDISP request message, lock interface flags, device address bits, or number of outstanding TDISP requests.
- **TDISP – DEVICE_INTERFACE_REPORT** includes the runtime capabilities and settings.
    - **Capability** includes the MMIO range attribute IS_MEM_ATTR_UPDATABLE.
    - **Runtime settings** include firmware update permission, Address Translation Service (ATS) state, Page Request Service (PRS) state, Message Signaled Interrupt Extension (MSI_X) control, Lightweight Notification Requester (LNR) control, Transaction Layer Packet (TLP) Processing Hints (TPH) control, and MMIO range report.

The TVM may verify these before accepting the device. A device firmware runtime update should not impact the runtime setting in the DEVICE_INTERFACE_REPORT. Since the TSM needs to send LOCK_DEVICE_INTERFACE to the DSM, the locked device interface should not be updated. See Table 5-2.

Table 5-2. Updatable Device Information Summary

Category	Example	Runtime Updatable	Info Report

SPDM Certificate	X.509 Certificate (DICE or non-DICE)	Updatable. Equal or better security.	SPDM CERTIFICATE
SPDM Measurement Code	Hash of ROM, Hash of firmware, Firmware Version, Firmware SVN	Updatable. Equal or better security.	SPDM MEASUREMENTS
SPDM Measurement Data	Hardware Fuse, Firmware Policy Config, Debug Mode, Device State (Normal, Recovery)	Updatable. Equal or better security.	SPDM MEASUREMENTS
TDISP Capability	DSM capability flags, Lock interface flags	Updatable. Equal or more capability.	TDISP CAPABILITIES
TDISP Device Interface Report Capability	MMIO range attribute	Updatable. Equal or more capability.	TDISP INTERFACE_REPORT
TDISP Device Interface Report Runtime Setting	Firmware update permission, MSI_X control, MMIO range report	Not updatable.	TDISP INTERFACE_REPORT

Evidence Collection Time

The TSM needs to collect the device evidence and let the TVM verify. If a runtime update is allowed, the device evidence may be changed at any time. Basically, the TSM may perform the evidence collection at the following points:

- **Device Connection Time**: after the TSM sets up SPDM connection with the device (SPDM GET_VERSION), it can collect the SPDM certificate and measurements. This is called **connection evidence** info.
- **Device Secure Session Creation Time:** when the TSM sets up a new SPDM session with the device (SPDM KEY_EXCHANGE/PSK_EXCHANGE), it can collect the SPDM certificate and measurements inside of the session. This is **session evidence** info. Different sessions may have different session evidence, if the TSM allows the device runtime update.
- **Device Interface Creation Time**: inside of an SPDM session, a VMM may create multiple device interfaces and use the TDISP protocol to manage the interface. Once an interface is created (TDISP GET_VERSION), the VMM may ask the TSM to collect the evidence info. This is called **interface creation evidence**. Even in one SPDM session,

different interfaces may be associated with different evidence, if the TSM allows the deice runtime update in a session.

- **Device Interface Lock Time**: a TVM or VMM may ask the TSM to collect the evidence after the device interface is locked (TDISP LOCK_DEVICE_INTERTFACE). We call this **interface lock evidence**, which is very similar to interface creation evidence. The only difference is that the TSM needs to ensure the interface is locked.
- **Device Update Time:** if a TVM or VMM knows that the device is updated, they may ask the TSM to collect the evidence again. This evidence is called **update fresh evidence**. The collection action might not be immediately triggered after the device update. There might be a delay based upon the refresh policy. Care must be taken that the TVM has a way to guarantee the **freshness** of the collected evidence, such as a nonce. The TSM shall collect and return both the evidence and the nonce to the TVM.

If a TVM needs to offload the work to a device after device interface lock and device verification, then the best way is to get the **interface lock evidence** for the device. This is when the TVM can ensure that the device configuration and security guarantee are locked. The TVM may also collect the **update fresh evidence** based on policy.

One policy example is that a cloud orchestrator may trigger a device firmware update and ask every TVM to refresh their device evidence. Another example is when the TVM creates a weekly task every Sunday morning to trigger the evidence collection, and verifies it based on the latest reference integrity manifest (RIM).

In case the device verification fails after collecting the refreshed device evidence, the TVM may perform a device failure action, such as stopping and detaching the device interface, or teardown of itself immediately.

Figure 5-2 shows an example of device evidence (SVN) collected at different times. The device is updated periodically and its SVN is increased, so the SVN collected at different times is different. The TVM can make the final decision about which SVN is to be used for verification.

Note: Each device interface may be associated with a different device SVN, even for the same device. In figure 5-2, interface-A has locked evidence SVN-3, while interface-B has SVN-5 and interface-C has SVN-6. After refresh, the interface-A may see SVN-{3,4,5}, interface-B can only see SVN-5, and interface-C can see SVN-{6,7} in refresh with History Mode.

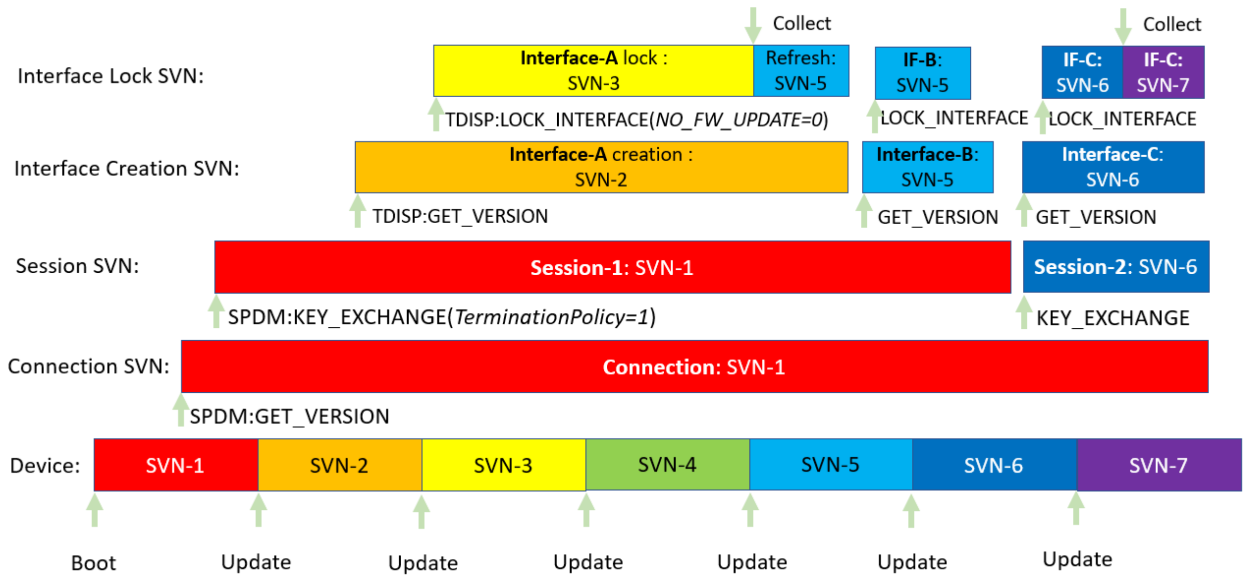

Figure 5-2: Device SVN collection example

Update Notification Policy

In case of device support for Refresh Mode, the device needs have a way to notify the TSM that the device update happened. The device may choose **Update Notification Policy**. The device may notify the TSM that a new update image is present and gives the TSM a chance to send the collected evidence again or even pre-authorize the update. See Figure 5-3.

- **No Notification / Silence Mode**: the device does not send any notification to the TSM.
- **Passive Notification / Poll Mode:** the device does not actively notify the TSM. However, if the TSM starts communicating with the device, the device will return the firmware update notification.
- **Active Notification / Interrupt Mode**: the device actively notifies the TSM after update, e.g., an event notification mechanism.
- **Pre-Authorization Notification / Interrupt Mode**: the device actively notifies the TSM before update and asks the host to authorize the new device firmware image. This gives the host-side software a chance to evaluate the new image based upon the host-side device update policy instead of the device-side device update policy. The host software needs to collect new image information (such as SVN or measurement) and send the pre-authentication result (accept or reject) back to the device. Only after all host software allows the new image and the device allows the new image will the update happen.
- **Out of band (OOB) Pre-Authorization**: instead of the device notifying the TSM, the host-side software may get the new device firmware image information from the OOB channel (not in the SPDM session) and evaluate the new image. If host-side software

decides to accept the new image, the host software can communicate with the TSM and let the TSM send an approval message to the device.

- The approval message shall include the new device firmware information, such as a hash of the new firmware image, new firmware measurement and/or the new device certificate. As such, the device RTU knows that only the particular new firmware image is authorized by the host.

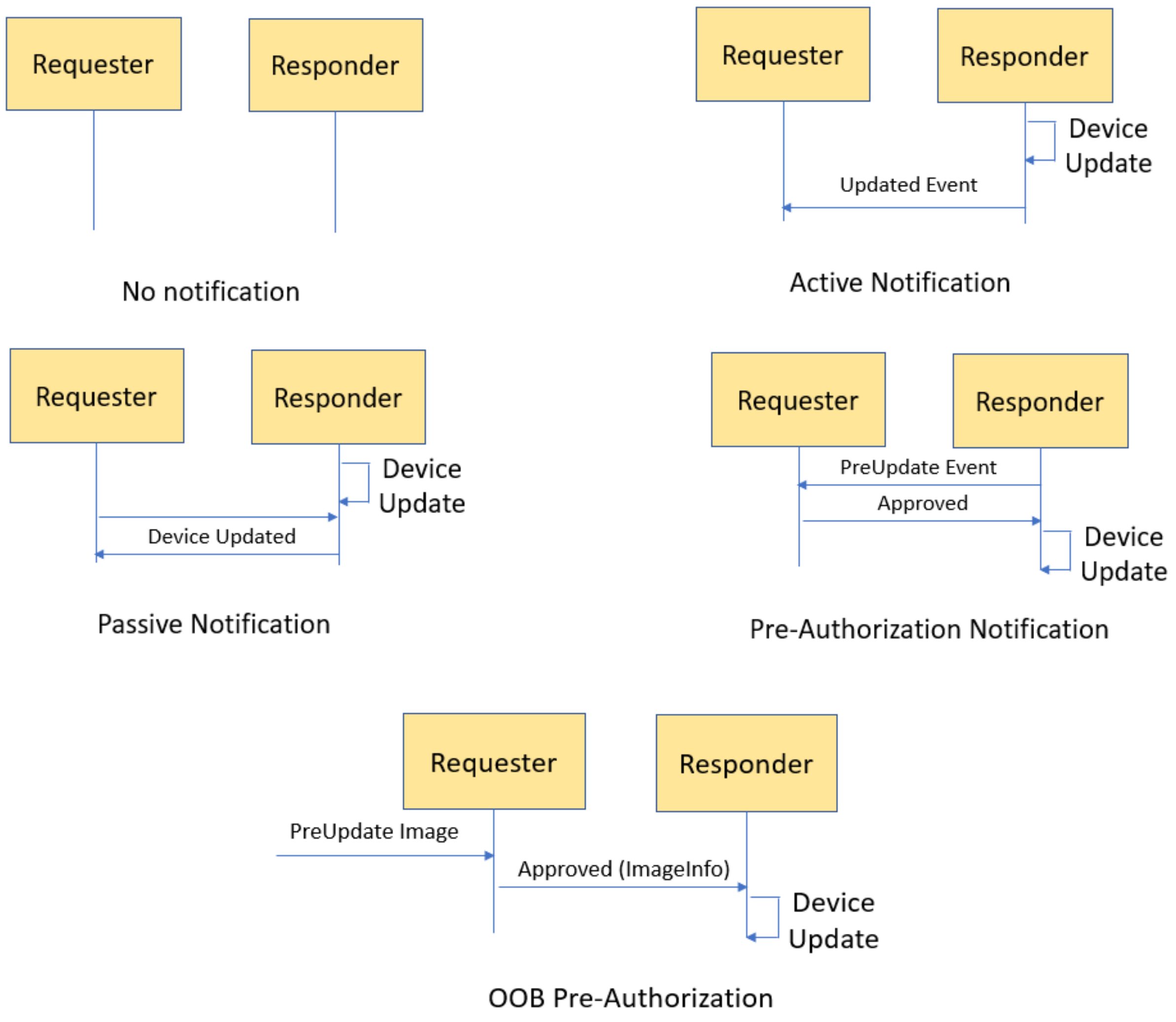

Figure 5-3: Update Notification Policy

Care must be taken that the notification happens in a secure session. As such, the TSM can have a way to detect if the attacker in the middle drops the notification packet.

Channel Reset Policy

After a TSM establishes a secure connection with a DSM, the TSM may use this channel to manage the device. For example, the TSM should use the integrity and data encryption (IDE) key management (IDE_KM) protocol to set up the IDE key for a Peripheral Component Interconnection (PCI) or Compute Express Link (CXL) device to secure the device physical link.

The TSM should use the Trusted Device Interface Security Protocol (TDISP) to manage a TDI of a physical device that supports scalable IO virtualization. These IDE_KM protocol or TDISPs are application protocols based upon the SPDM secure session.

A TSM may use TDISP to attach, detach, or lock a device interface to a TVM in a trusted manner. The TSM may also use TDISP to start or stop the device DMA or lock the firmware update. Again, let's clarify some terms for device communication. See Figure 5-4.

- **Management Channel** is a secure communication between the TSM and DSM to allow the TSM to manage the DSM, such as TDISP protocol on top of an SPDM secure session.
- **Data Channel** is the data transfer between the TVM and device interface, such as memory mapped input/output (MMIO) and Direct Memory Access (DMA).

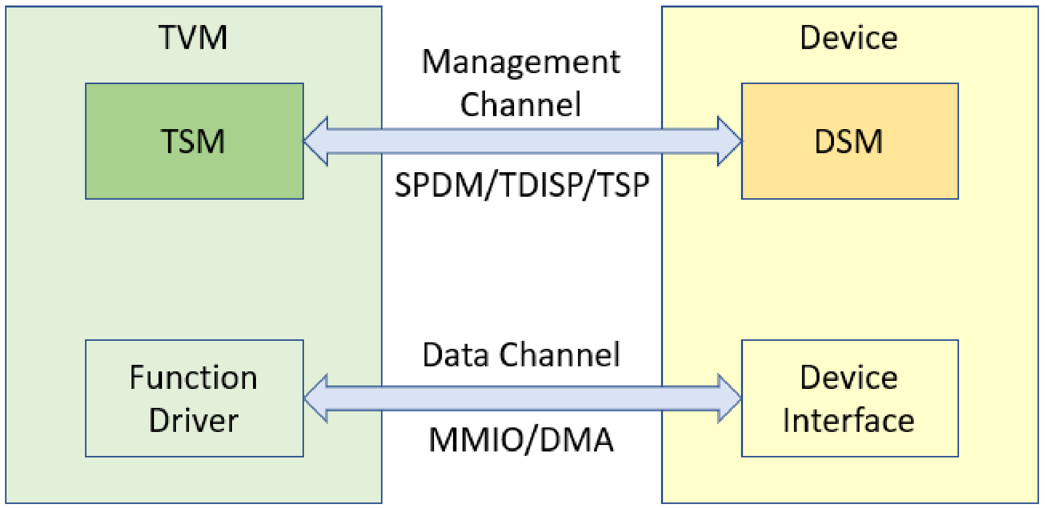

Figure 5-4: Communication Channel between TVM and Device

Based on different device update mechanisms, we may use a different level reset. See table 5-3.

Table 5-3: Communication Channel Reset in Device Update

Level	Channel Reset	Description	Example
0	No Channel Reset	The device is not reset. The management channel between the TSM and DSM is kept.	Firmware Live Path

		The data transfer between the TVM and device interface is also kept.	
1	Data Channel Reset	The device is not reset. The management channel between the TSM and DSM is kept. The data transfer needs to be stopped and started again.	Firmware NV Storage Update + Runtime Activation
2	Management and Data Channel Reset	The device is reset after update. The management channel between the TSM and DSM needs to be reestablished. The data transfer needs to be stopped and started again.	Firmware NV Storage Update + Activation After Reset

The TEE environment may need to perform the attestation again once the device update happens.

Runtime Update Attestation

Device Update with Management and Data Channel Reset

If the TVM knows that the device is reset, and it needs to reestablish the management channel such as SPDM/TDISP and the data channel for MMIO and DMA.

The TSM shall collect the evidence again (such as via SPDM GET_CERTIFICATE and GET_MEASUREMENT) and perform the verification according to the appraisal policy. These steps are same as in the normal device startup.

Device Update with Data Channel Reset

If a device update does not need a reset, the management channel can be kept. The SPDM session and TDISP protocol are alive after the update.

If a TSM still wants to perform attestation for the device after update, the TSM shall set NO_FW_UPDATE policy in TDISP LOCK_INTERFACE_REQUEST. The device firmware update request will be rejected from the device interface in CONFIG_LOCKED state. The VMM shall detach this device interface from TVM and move it to CONFIG_UNLOCKED state to perform a device firmware update.

The VMM will then attach the device again, and the TSM shall follow normal flow to collect the evidence and perform the verification according to its appraisal policy. The SPDM session is kept, and the TSM may skip the step to reestablish the SPDM secure session.

In some rare cases, a device update may only involve a configuration change that does not have a secure version number (SVN) associated with it. In such cases, the device shall set the device interface to ERROR state.

The device interface will then be detached from the TVM. Again, the VMM may attack the device and the TSM shall follow the normal verification process mentioned previously. The configuration change shall be reflected in the new SPDM measurement.

Device Update without Channel Reset

In an aggressive device live patching, the device may choose to keep both the management channel and data channel. The DSM may choose Birth Mode to report the old certificate and measurement collected at launch time. In this case, the DSM shall remove the SPDM MEAS_FRESH_CAP flag to warn the TVM that collects the evidence.

The DSM may also choose Refresh Mode with or without history to always report the fresh certificate and measurement with the SPDM MEAS_FRESH_CAP flag. If the update happens, the DSM or out of band (OOB) mechanism may notify the TVM on the measurement change, then the TVM can collect the evidence again and perform the verification process.

However, there is a race condition window. Before the TSM performs verification, the device may already transfer data to the TVM. This shall only be supported when the DSM follows NIST SP800-193 to perform a known good update - such as with a digital signature check, revocation list check, higher or equal SVN, etc.

Device Update with pre-authorization

If the device supports pre-authorization, then the DSM or OOB mechanism shall send a notification event to the TSM before the update happens. The TSM may collect the new image evidence information (such as measurement or SVN) and perform the verification based upon the update appraisal policy before the device updates the firmware.

If the final result is PASS, then the TSM can send the pre-authorization result “APPROVE” - otherwise the TSM should send “REJECT”. The DSM should follow the pre-authorization to perform the update or drop the update. If the VMM must update the device, the VMM can choose to detach the device from the TVM. If the device must update based upon its own policy, the device should disconnect with the TSM.

6 Summary

In this white paper, we discussed the device attestation in TEE confidential computing. In a generic device attestation model, the TSM collects the device identity (such as certificate chain) and evidence (such as measurement). After which a verifier in the TVM verifies the evidence based on the endorsement (such as root certificate), the reference value (such as reference measurement) and the policy.

The TVM may perform the local attestation as described previously. The remote party may perform the remote attestation for the TVM, TSM, and the device. As a special use case, the device may want to perform attestation to the TVM/TSM, which can be done via mutual attestation or two-phase attestation. Finally, the device may perform updates, and the TVM/TSM may need to perform attestation again after the update.

Appendix A: Mapping to Intel TDX

Intel Trust Domain Extensions (TDX) is a confidential computing architecture. Intel TDX architecture may support device I/O with software protection mechanisms. Intel TDX with TEE-IO supports device I/O with hardware protection mechanisms. The following section provides a high-level summary of Intel TDX mapping for the method described in this white paper. For more details on Intel TDX, please refer to the Intel TDX architecture specification. Table A-1 shows the generic terminology mapping for Intel TDX.

Table A-1. Terminology Mapping for Intel TDX

Term	Intel TDX 1.0/1.5	Intel TDX with TEE-IO	Description
TEE-IO	Device I/O with software protection	Device I/O with hardware protection	The definition for confidential computing with I/O device.
TEE Virtual Machine (TVM)	Tenant Trust Domain (TD)	Tenant Trust Domain (TD)	The trusted virtual machine.
TEE Security Manager (TSM)	Security Manager in Tenant TD (or) a dedicate Service TD	TDX Module + TEE-IO Provisioning Agent (TPA)	The trust policy enforcer on the host.

Device Attestation – Evidence Obtaining and Reporting

Device evidence obtaining and reporting is done by the TSM. In Intel TDX 1.0/1.5, the TSM is a special security manager in tenant TD or a dedicated service TD. In Intel TDX with TEE-IO, the TSM is the TEE-IO Provisioning Agent (TPA) which is an extension of TDX Module.

For example, the device service TD or TPA may issue SPDM GET_CERTIFICATE and GET_MEASURMENTS to the DSM, to collect the device certificate and measurement and verify their digital signature. Later, the TSM presents the data to a verifier in the tenant TD.

Device Attestation – Evidence Verification

First, the verifier in the tenant TD should refer to a set of trust anchors (a root CA certificate or an intermediate certificate) to determine whether the DICE certificate chain can be trusted. It should also refer to device certificate revocation lists (CRLs) to ensure that the device chain is not revoked.

Second, the verifier should refer to the device reference integrity manifest (RIM) to verify the device measurement record. For a DICE device, the verifier should use the measurement data in DICE TCB info.

The reference data could be provisioned in the local tenant TD, or it could be retrieved from a trusted entity such as cloud orchestrator from the network.

References

Standards

[ISO-SWID] ISO/IEC 19770-2:2015 Part 2: Software Identification Tag

[NIST-800-193] NIST, SP 800-193: Platform Firmware Resiliency Guidelines, https://csrc.nist.gov/publications/sp800

[NIST-800-155] NIST, SP 800-155: BIOS Integrity Measurement Guidelines (draft), https://csrc.nist.gov/publications/sp800

[NIST-SWID] NIST: Software-Identification-SWID, https://csrc.nist.gov/projects/Software-Identification-SWID

[NIST-8060] NISTIR 8060 Guidelines for the Creation of Interoperable SWID Tags, https://csrc.nist.gov/publications/detail/nistir/8060/final

[IETF-RATS] IETF, RATS: Remote Attestation Procedures Architecture, https://datatracker.ietf.org/doc/draft-ietf-rats-architecture/

[IETF-ATTS] IETF, RATS: Reference Interaction Models for Remote Attestation Procedures, https://datatracker.ietf.org/doc/draft-ietf-rats-reference-interaction-models/

[IETF-EAT] IETF, RATS: The Entity Attestation Token (EAT), https://datatracker.ietf.org/doc/draft-ietf-rats-eat/

[IETF-TPM] IETF, RATS: TPM-based Network Device Remote Integrity Verification, https://datatracker.ietf.org/doc/draft-ietf-rats-tpm-based-network-device-attest/

[IETF-COSWID-EXT] IETF, RATS: Reference Integrity Measurement Extension for Concise Software Identities, https://datatracker.ietf.org/doc/draft-birkholz-rats-coswid-rim/

[IETF-CORIM] IETF, RATS: Concise Reference Integrity Manifest (CoRIM), https://datatracker.ietf.org/doc/draft-birkholz-rats-corim/

[IETF-COSWID] IETF, SACM, Concise Software Identification Tags (CoSWID), https://datatracker.ietf.org/doc/draft-ietf-sacm-coswid/

[IETF-TLS] IETF, RFC8446: The Transport Layer Security (TLS) Protocol Version 1.3, https://tools.ietf.org/html/rfc8446

[TCG-TPM2-KEY] TCG, TPM 2.0 Keys for Device Identity and Attestation, https://trustedcomputinggroup.org/resource/tpm-2-0-keys-for-device-identity-and-attestation/

[TCG-RIMIM] TCG, Reference Integrity Manifest (RIM) information Model, https://trustedcomputinggroup.org/resource/tcg-reference-integrity-manifest-rim-information-model/

[TCG-PCRIM] TCG, PC Client Reference Integrity Manifest (RIM) Specification, https://trustedcomputinggroup.org/resource/tcg-pc-client-reference-integrity-manifest-specification/

[TCG-FIM] TCG, PC Client Platform Firmware Integrity Measurement (FIM), https://trustedcomputinggroup.org/resource/tcg-pc-client-platform-firmware-integrity-measurement/

[TCG-PLAT-CERT] TCG, PC Client Platform Certificate Profile, https://trustedcomputinggroup.org/resource/tcg-platform-certificate-profile/

[TCG-DICE-LAYERING] TCG, DICE Layering Architecture, https://trustedcomputinggroup.org/resource/dice-layering-architecture/

[TCG-DICE-ATTS] TCG, DICE Attestation Architecture, https://trustedcomputinggroup.org/resource/dice-attestation-architecture/

[TCG-DICE-CERT] TCG, DICE Certificate Profiles, https://trustedcomputinggroup.org/resource/dice-certificate-profiles/

[TCG-DICE-SYM] TCG, DICE Symmetric Identity Based Device Attestation, https://trustedcomputinggroup.org/resource/symmetric-identity-based-device-attestation/

[TCG-DICE-] TCG, DICE Symmetric Identity Based Device Attestation, https://trustedcomputinggroup.org/resource/symmetric-identity-based-device-attestation/

[TCG-TPA-IM] TCG, DICE Endorsement Architecture for Devices, https://trustedcomputinggroup.org/wp-content/uploads/TCG-Endorsement-Architecture-for-Devices-r38_5May22.pdf

[TCG-TAP-USE] TCG, Trusted Attestation Protocol (TAP) Use Cases for TPM 1.2 and 2.0 and DICE, https://trustedcomputinggroup.org/resource/tcg-trusted-attestation-protocol-tap-use-cases-for-tpm-families-1-2-and-2-0-and-dice/

[TCG-EVENT-LOG] TCG, Canonical Event Log Format, https://trustedcomputinggroup.org/wp-content/uploads/TCG_IWG_CEL_v1_r0p30_13feb2021.pdf

[DMTF-0274] DMTF, DSP0274: Security Protocol and Data Model (SPDM) Specification, https://www.dmtf.org/dsp/DSP0274

[DMTF-2058] DMTF, DSP2058: Security Protocol and Data Model (SPDM) Architecture Whitepaper, https://www.dmtf.org/dsp/DSP2058

[PCI-PCIE] PCI, PCI Express Base 6.0 Specification, https://members.pcisig.com/wg/PCI-SIG/document/16609

[PCI-TDISP] PCI, Trusted Device Interface Security Protocol (TDISP) ECN, https://members.pcisig.com/wg/PCI-SIG/document/16849

[CXL-CXL] CXL, CXL 3.0 Specification, https://www.computeexpresslink.org/

[OCP-SIOV] OCP, Scalable I/O virtualization, https://www.opencompute.org/documents/ocp-scalable-io-virtualization-technical-specification-revision-1-v1-2-pdf

[AMD-SEV] AMD, AMD Secure Encrypted Virtualization (SEV), https://developer.amd.com/sev/

[ARM-RME] ARM, ARM Realm Management Extension (RME), https://developer.arm.com/documentation/den0129

[INTEL-TDX] Intel, Intel Trust Domain Extensions (TDX), https://software.intel.com/content/www/us/en/develop/articles/intel-trust-domain-extensions.html

[RISCV-APTEE] RISCV, RISCV AP TEE, https://github.com/riscv-non-isa/riscv-ap-tee/blob/main/specification/attestation.adoc

Web Resources

[CCC-HWTEE] Confidential Computing Consortium, Hardware-Based Trusted Execution for Applications and Data, https://confidentialcomputing.io/whitepaper-01-latest/

[CCC-CCDD] Confidential Computing Consortium, Confidential Computing Deep Dive v1.0, https://confidentialcomputing.io/whitepaper-02-latest/

[TPM-ATTS] TPM Remote Attestation Best Known Methods, https://tpm2-software.github.io/tpm2-tss/getting-started/2019/12/18/Remote-Attestation.html

[ANDROID-ATTS] Android Key Stone, https://source.android.com/security/keystore/attestation

[RATLS] Integrating Remote Attestation with Transport Layer Security (RA-TLS), https://github.com/cloud-security-research/sgx-ra-tls

[OPENENCLAVE] open enclave, https://github.com/openenclave/openenclave

[OPENENCLAVE-RATLS] attested TLS, https://github.com/openenclave/openenclave/blob/master/samples/attested_tls/AttestedTLSREADME.md

[OPA] open policy agent (OPA), https://www.openpolicyagent.org/

[OPA-REGO] open policy language (Rego), https://www.openpolicyagent.org/docs/latest/policy-language/

[INTOTO] in-toto, https://in-toto.io/