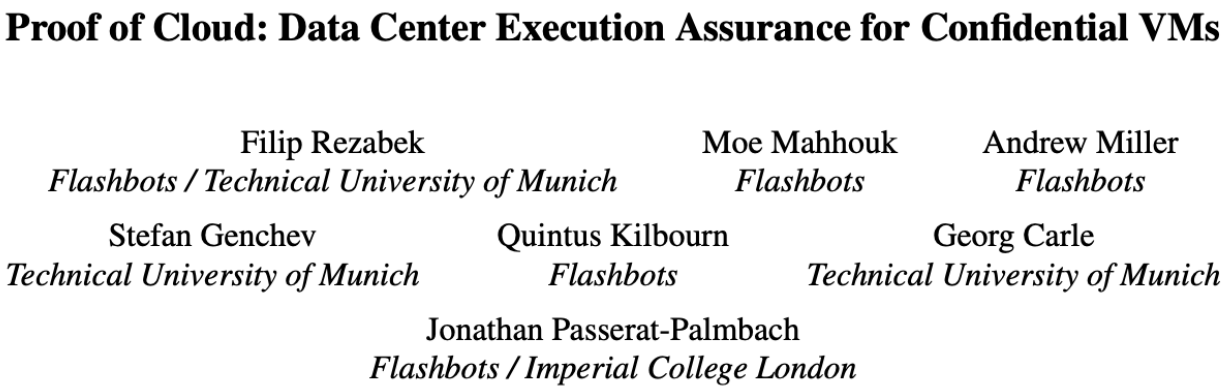
Proof of Cloud: Data Center Execution Assurance for Confidential VMs

Filip Rezabek

Moe Mahhouk
Andrew Miller
Quintus Kilbourn
Prof. Dr.-Ing. Georg Carle
Jonathan Passerat-Palmbach



Proof of Cloud: Data Center Execution Assurance for Confidential VMs

Filip Rezabek
Flashbots / Technical University of Munich

Moe Mahhouk
Flashbots

Andrew Miller
Flashbots

Stefan Genchev
Technical University of Munich

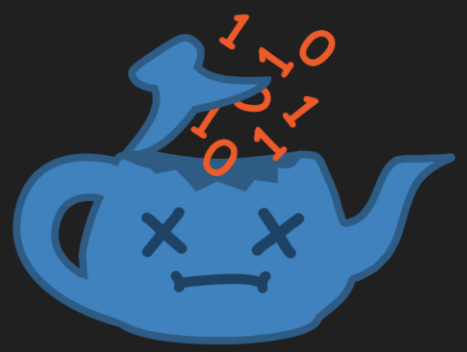
Quintus Kilbourn
Flashbots

Georg Carle
Technical University of Munich

Jonathan Passerat-Palmbach
Flashbots / Imperial College London

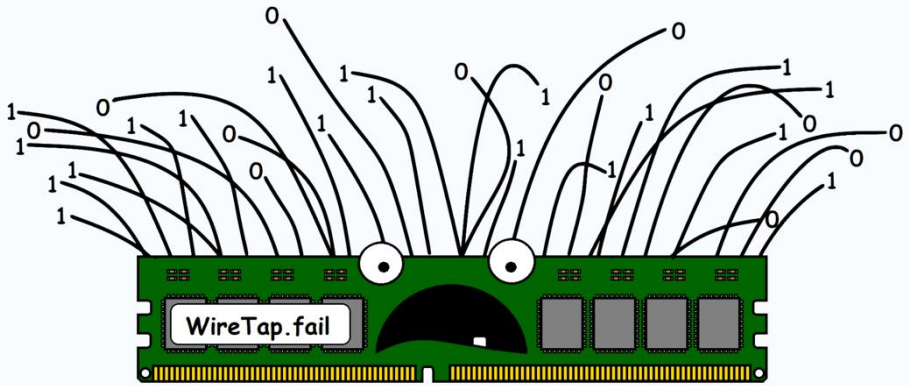
Flashbots

Motivation
Problem Definition



TEE.fail:

Breaking Trusted Execution Environments via DDR5 Memory Bus Interposition



WireTap:

Breaking Server SGX via DRAM Bus Interposition

Motivation

Problem Definition

Assume a sensitive workload e.g., **DeFi** or expensive **LLM** model

Trust assumptions are now on the host/operator being honest, to ensure physical security

However, **TEE attestation flows** mention *what* workload they run, but not *where* it runs.

→ Provide **assurance that CVM** runs in the **respective** infrastructure

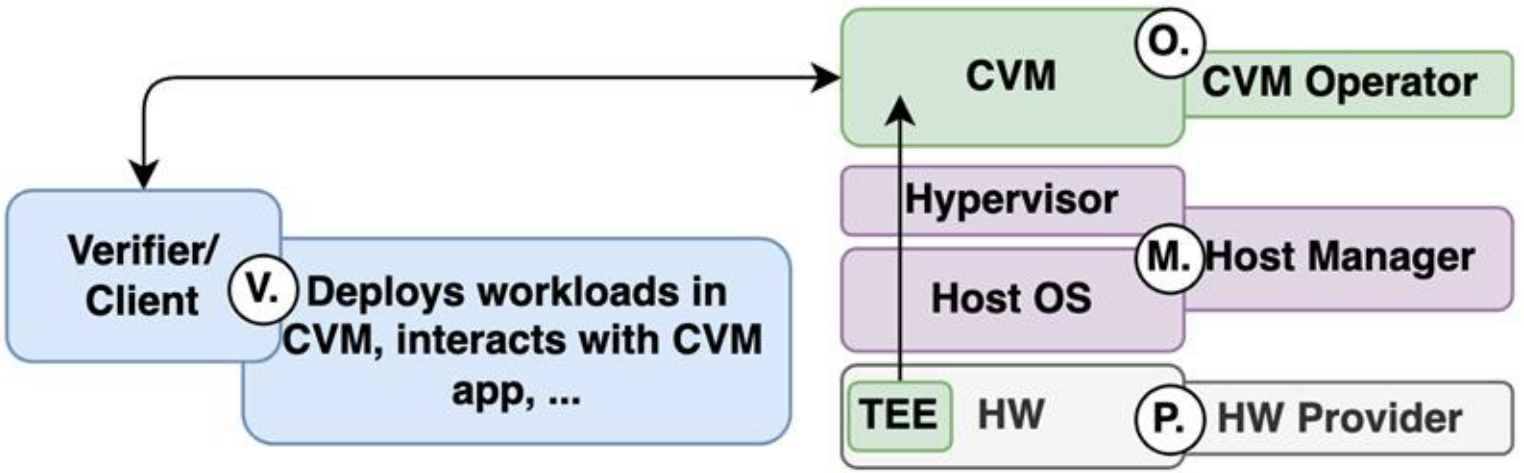
Narrowing the Gap between TEEs Threat Model and Deployment Strategies

Filip Rezabek[1,2], Jonathan Passerat-Palmbach[1,3], Moe Mahhouk[1], Frieder Erdmann[1], and Andrew Miller[1]

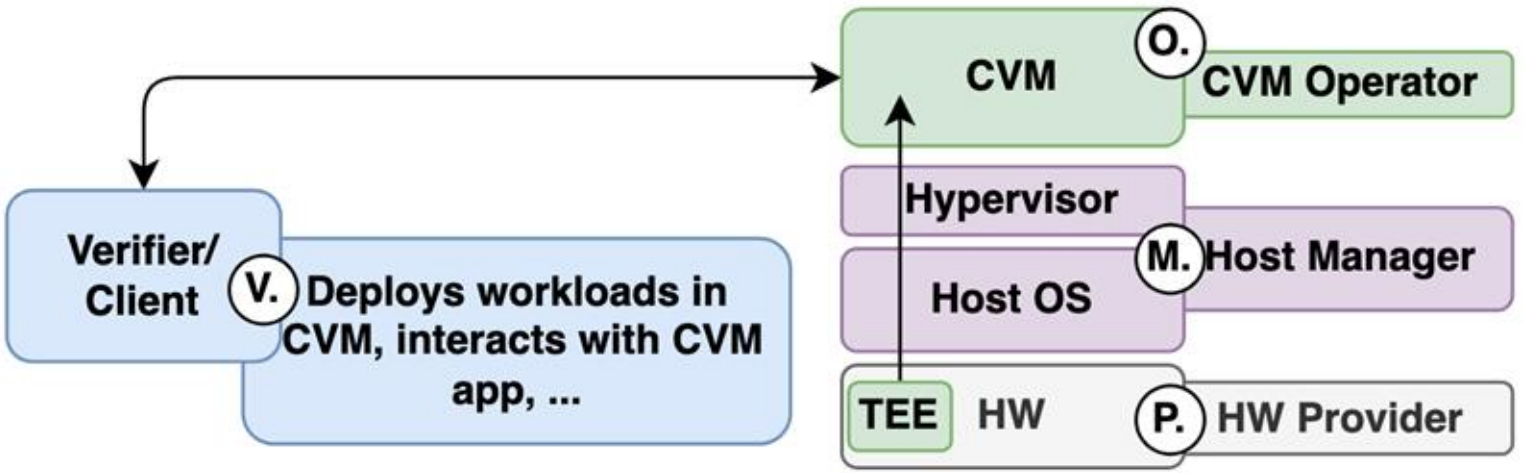
[1]Flashbots
[2]Department of Informatics, Technical University of Munich, Germany
[3]Imperial College London

https://arxiv.org/abs/2506.14964

Motivation
Setting



Motivation
Setting

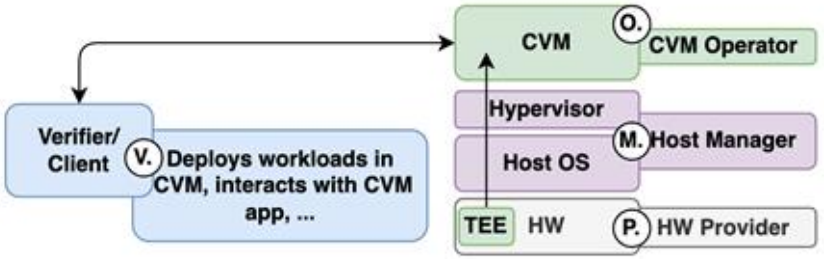


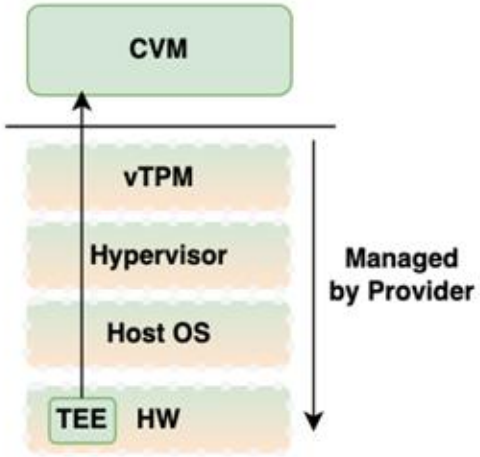
Verifier only sees the **remote attestation**

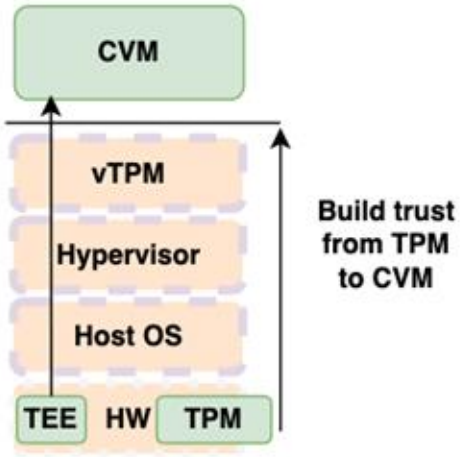
Motivation
Introduction

Two **scenarios**:

1. **Confidential Virtual Machine (CVM)** in cloud
2. **Bare** metal in cloud





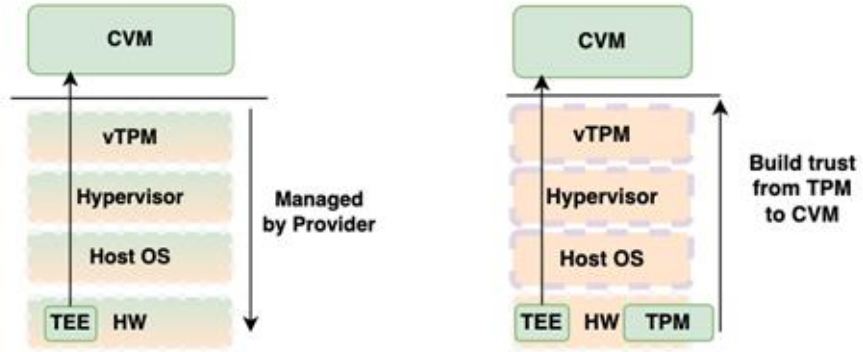


Motivation
Introduction

Two **scenarios**:

1. **Confidential Virtual Machine (CVM)** in cloud
2. **Bare** metal in cloud

Requirements – keep the current **Trust Domain** (**TD)** flow, but strengthen **Data Center Execution Assurance (DCEA)** aspects



Motivation
Introduction

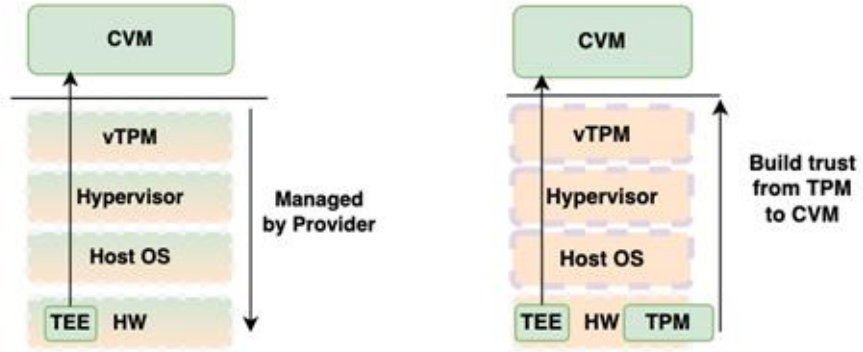
Two **scenarios**:

1. **Confidential Virtual Machine (CVM)** in cloud
2. **Bare** metal in cloud

Requirements – keep the current **Trust Domain** (**TD)** flow, but strengthen **Data Center Execution Assurance (DCEA)** aspects

- Orchestrate everything from **TD**
- Consider TPMs and **Dynamic Root of Trust Measurement (DRTM)** as an established technologies

We focus on **Intel TDX** → Requirements on attestation not met for **AMD**



TPM
Refresher

Trusted Platform Module (TPM) is a HW element on a **motherboard**

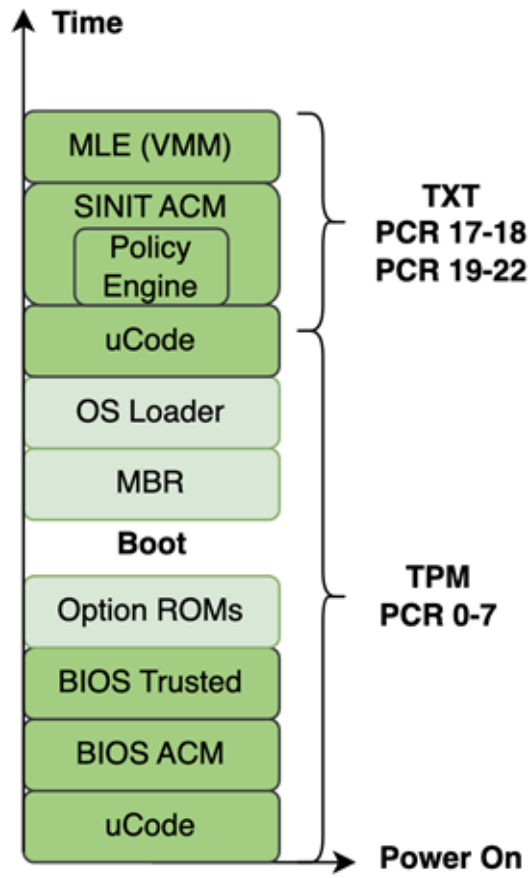
Commonly use for features e.g., **secure boot, key storage,** and **attestation**

Relies on own set of registers called **Platform Configuration registers (PCRs)**

- **Only extended, not overwritten**
- Reflects platform boot **measurements**

What is a difference of **vTPM** and **TPM**?

- HW vs SW-based solution, depends on hypervisor integrity
- TPM better for **bare-metal** trust anchor
- **vTPM** provides secure virtualization



Intel TXT
Refresher

Generally - Dynamic Root of Trust Measurement (DRTM)

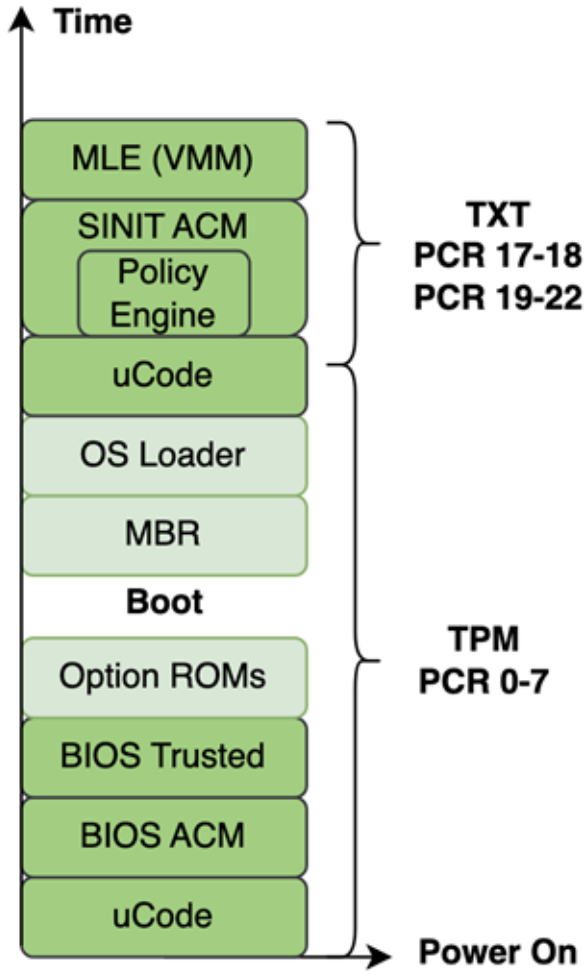
Intel Trusted Execution Technology (TXT)

- **HW based** security feature
- Verifies launch of hypervisors or **OS**
- Works with **TPM** to ensure platform state

PCRs interactions

- **TXT** uses **TPM** to store measurements in **PCRs**
- Validates BIOS, bootloader, and kernel hashes before boot

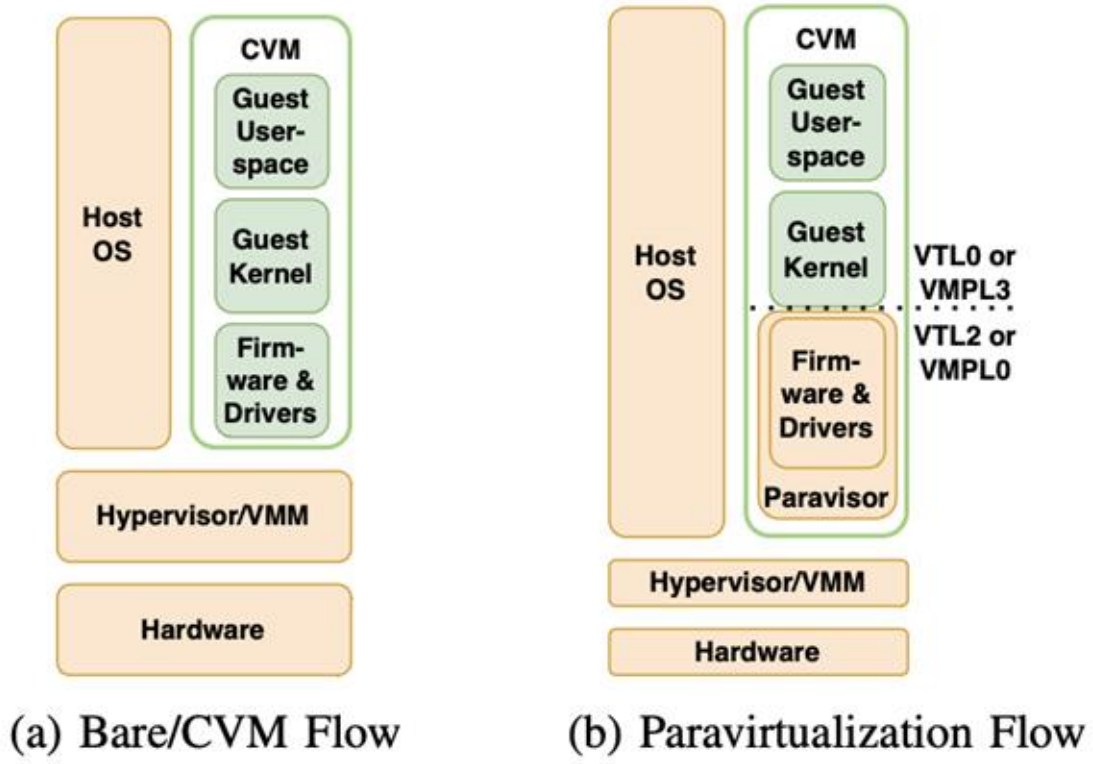
→ TPM + TXT = verifiable host boot chain.



Intel TDX
Refresher

Intel Trusted Domain Extensions (TDX)

- **HW based** security feature
- **VM**-based TEE
- Bare/CVM flow or paravirtualization



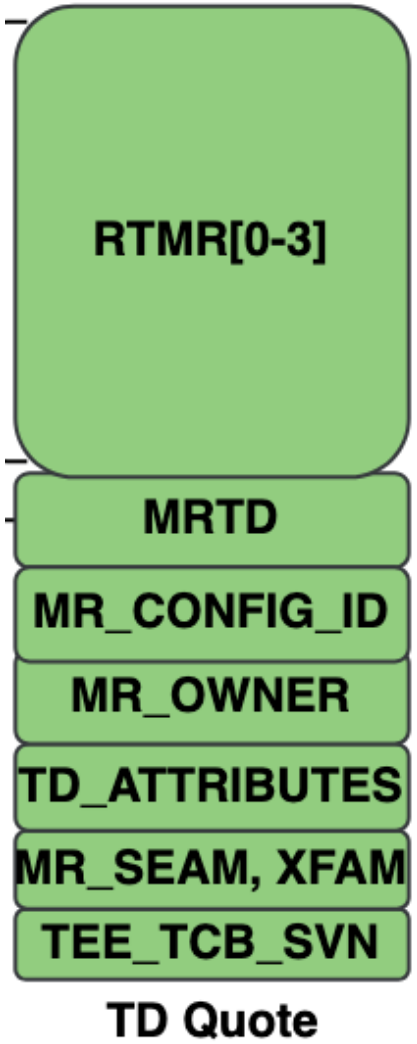
Intel TDX
Refresher

Intel Trusted Domain Extensions (TDX)

- **HW based** security feature
- **VM**-based TEE

Attestation

- Root of trust from Intel - **setup** flow, using **Intel SGX**
- During **runtime,** attests several information
  - Firmware provided by Intel
  - Measurement of the Trust Domain (MRTD)
  - Information about the VM
  - **Runtime Measurement Registers (RTMR)**



Key limitation:

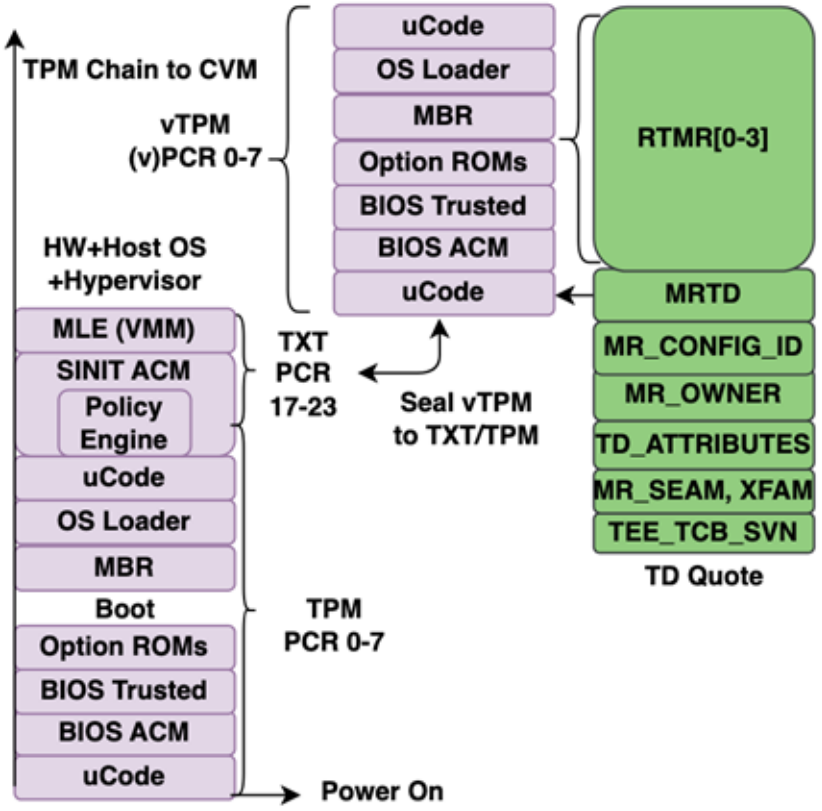
- TDX quote proves **the VM state**, but not **where the CPU is installed**.

General Direction

Combining two Roots of Trust

TPM – TXT – vTPM: follows one certificate chain provided and managed by the **cloud provider**

TD flow: a root of trust from **Intel**



General Direction

Combining two Different Roots of Trust

The **way** to compare the fields inside of **TD**:

- Side note: no equivalent to **RTMRs** provided by AMD, **ARM CCA** seems promising with Realm Extensible Measurement (REM)

PCRs	Covered Parts	Intel TDX	AMD SEV-SNP	ARM CCA
0	Virtual firmware (image)	MRTD	LD*	RIM
1, 7	Virtual firmware data & config	RTMR[0]	—	REM[0]
2–5	OS kernel, initrd, boot params	RTMR[1]	—	REM[1]
8–15	OS apps / user-space integrity	RTMR[2]	—	REM[2]
—	Reserved	RTMR[3]	—	REM[3]

* LD - LAUNCH_DIGEST, Launch-time only.

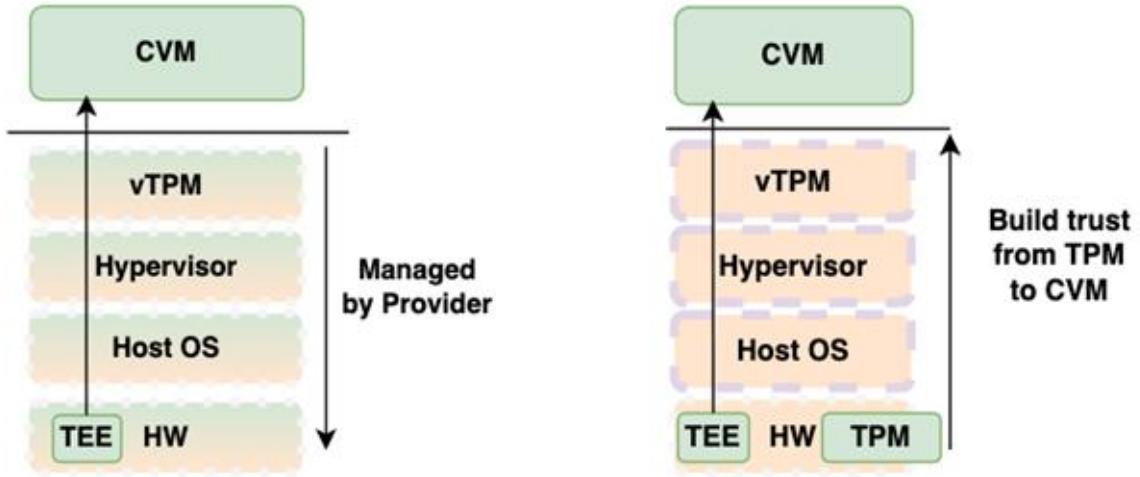
Cheng et al.. 2024. Intel TDX Demystified: A Top-Down Approach. ACM Comput. Surv. 56, 9, Article 238 (September 2024), 33 pages. https://doi.org/10.1145/3652597

Security Analysis

Possible Attack Strategies

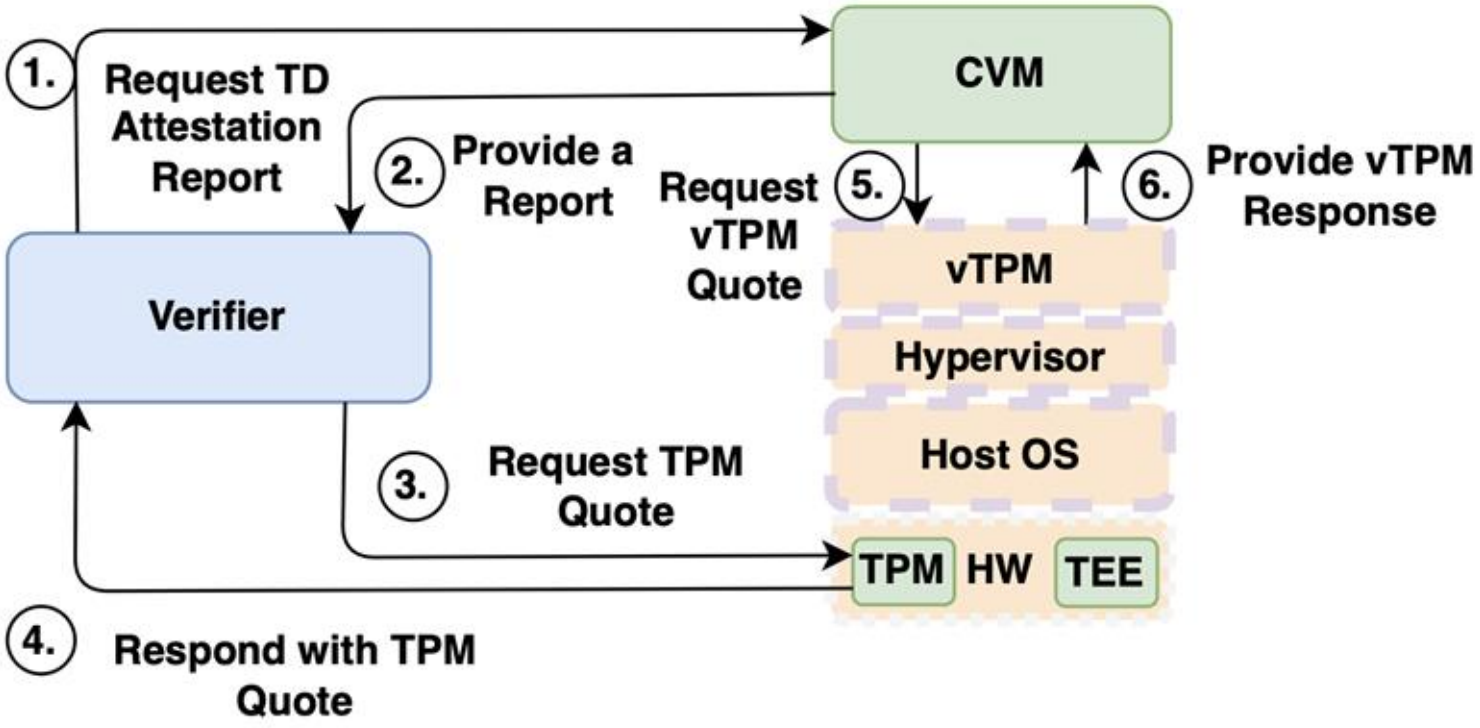
Identified **six relevant** attacks

1. Relay & Proxy – Proxy quote to another machine
2. Quote Forgery – Forge or simulate vTPM quote
3. Measurement Inconsistency – vTPM is not possible to match with RTMR
4. Channel Interception – Man-in-the-Middle between vTPM and TD flow
5. Identity Substitution – replace AK
6. Component Compromise – run malicious vTPM



Security Analysis

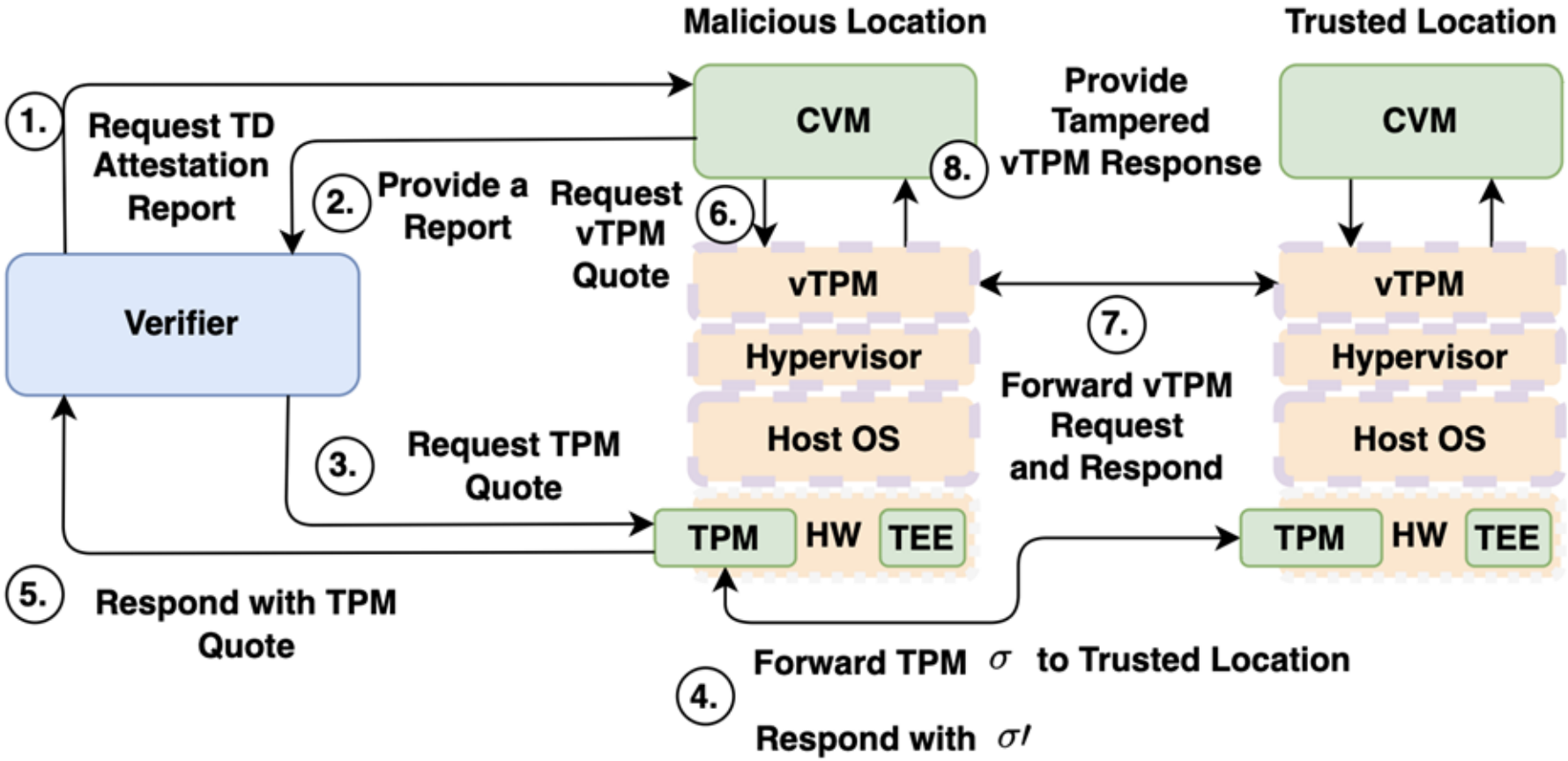
Relay & Proxy – naïve attestation flow



Security Analysis

Relay & Proxy deployment

The main challenge in **bare metal** deployments – malicious host creates a ***Frankenstein attestation*** by **proxying** traffic of **CVM** through from home lab through e.g., **GCP**



Security Analysis
Possible Attack Strategies

A#	Category name	Representative actions	Scenario I	Scenario II
A1	Relay/proxy ("mix-and-match")	Proxy quote to another machine; proxy; Synchronized replay of nonce/PCRs	✗	✓
A2	Quote/measurement forgery	Forge or simulate vTPM quote; Inject falsified PCR/RTMR values	✓	✓
A3	Measurement inconsistency	PCRs ≠ TD-RTMRs; `report_data` mismatch	✓	✓
A4	Channel interception/tampering	MITM on TD to vTPM path; MITM on host to pTPM bus	✗	✓
A5	Identity & key substitution	Spoof TPM-EK certificate; Replace expected AK	✗	✓
A6	Privilege-level component compromise	Run modified or malicious vTPM binary	✗	✓

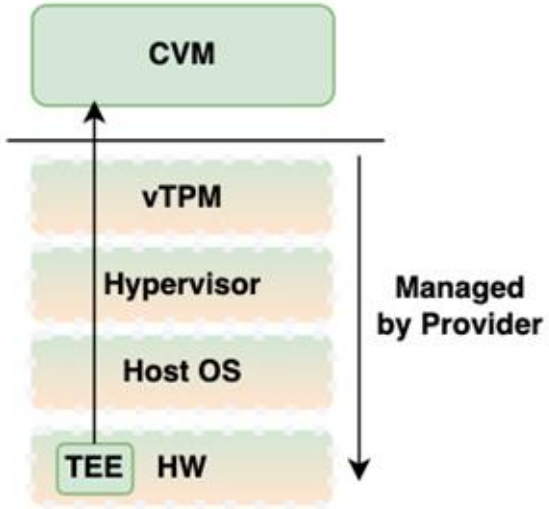
Scenario 1
Solution - CVM + vTPM

Extend classical **TD** flow

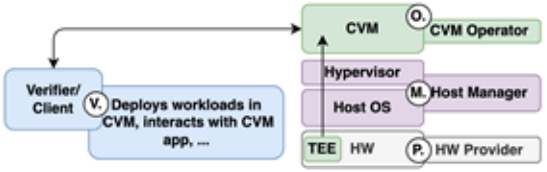
CVM interacts with **vTPM**

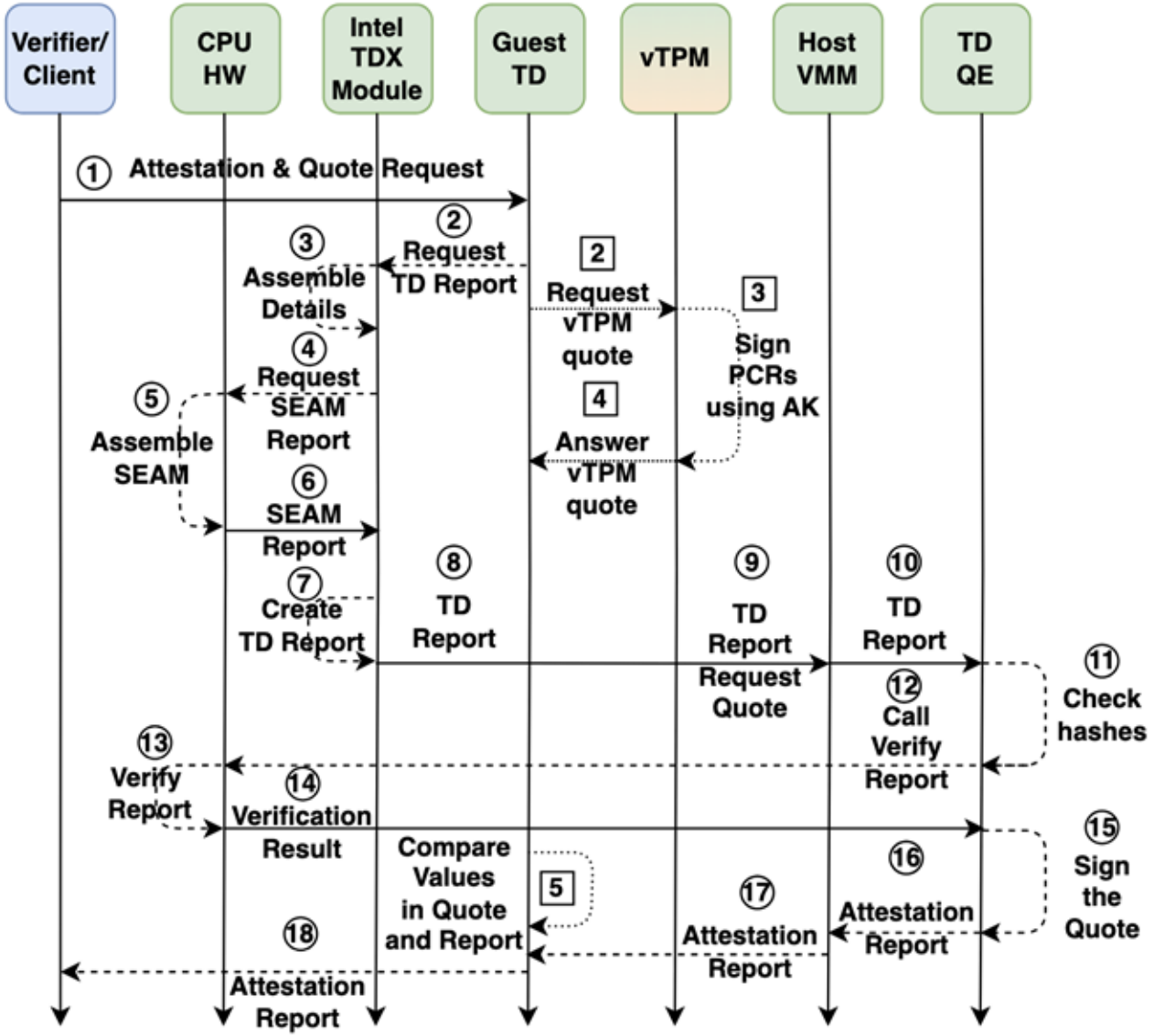
- vTPM provide **PCR** information matching secure boot
- TD contains these details in **Runtime Measurement Registers (RTMR)**

Comparison between values in **registers** in **CVM**

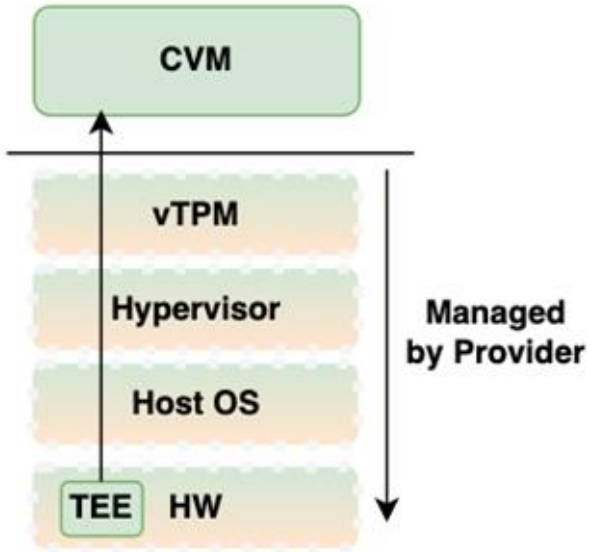


Scenario 1
Solution - CVM + vTPM

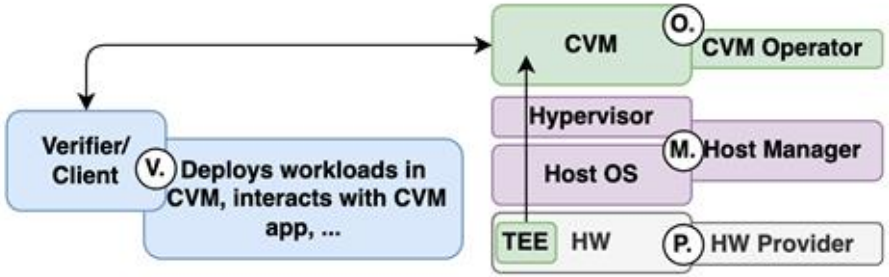


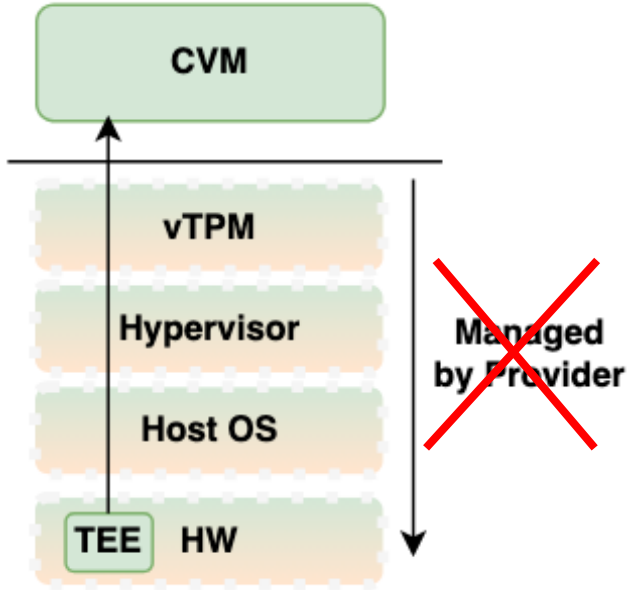


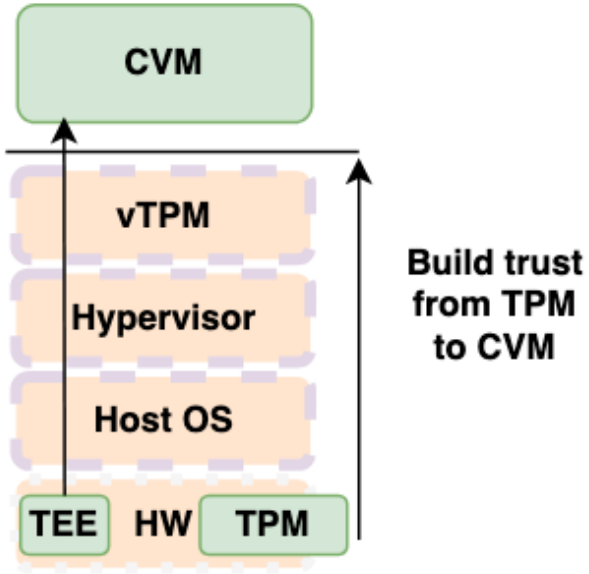
Scenario 2
Bare Metal Deployment



Scenario 2
Bare Metal Deployment







Scenario 2

Bare Metal Deployment

The main challenge in **bare metal** deployments – malicious host creates a ***mix-and-match attestation*** by **proxying** traffic of **CVM** through from home lab through e.g., **GCP**

Two challenges:

1. Binding host to location e.g., **GCP**
2. Binding CVM to host on a ***malicious*** **host**

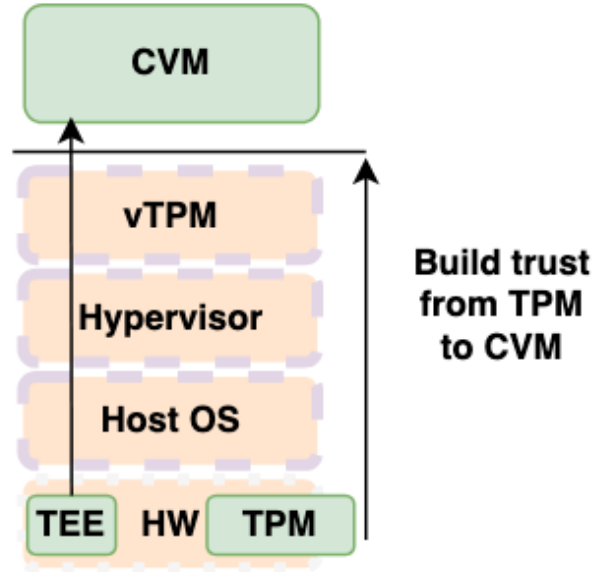
We must find **a minimal root of trust on host OS**, which **overlaps** with **TD attestation view**

Scenario 2
Bare Metal Deployment

Solution to **Challenge 1**:

Issue a cert to **TPM endorsement key (EK)** of the platform

- Assume only **GCP** would issue such a cert to own HW
- Similar like in case of **vTPM,** this contains relevant information



Scenario 2
Bare Metal Deployment

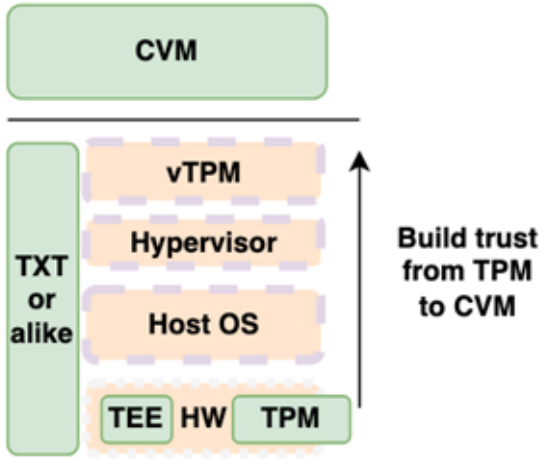
Solution to **Challenge 1**:

Issue a cert to **TPM' EK** on the platform

- Assume only **GCP** would issue such a cert to own HW
- Similar like in case of **vTPM,** this contains relevant information

Challenge 2: Build a **chain of trust** from **TPM** to **CVM**

CVM: Obtains a TD and *TXT/vTPM* quote (or similar) to compare the details and if they match



Solutions to Challenge 2
Bare Metal Deployment

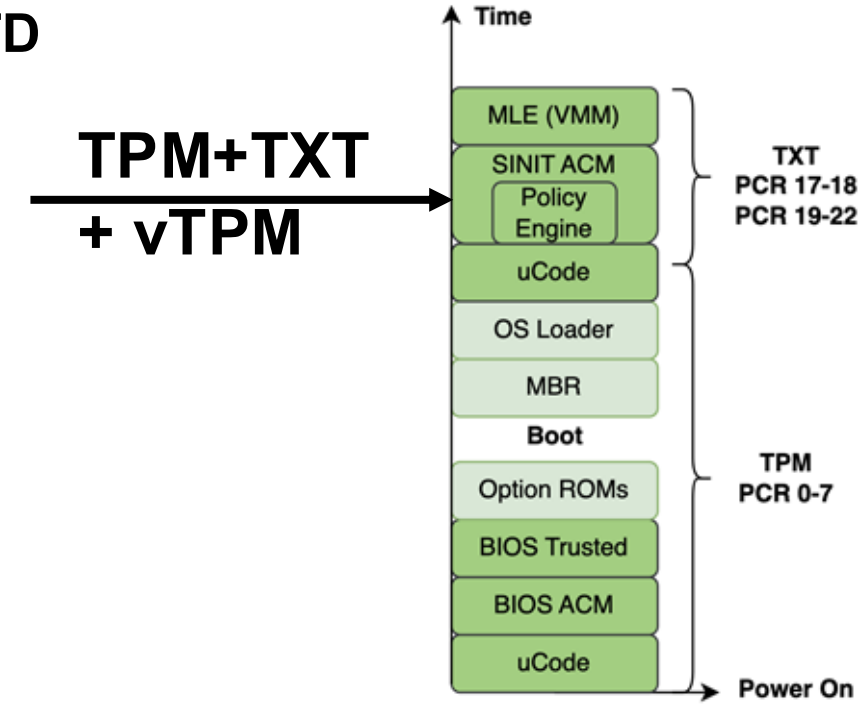
Challenge 2: Build a **chain of trust** from **TPM** to CVM

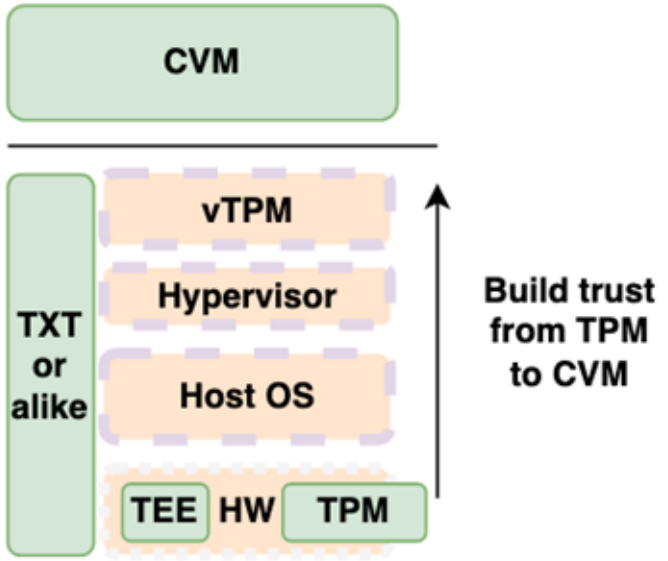
Solution 2 (inspired by CVM + vTPM):

Run **vTPM** that builds on top of **TXT,** and is bound to **TPM**

→ Sealing of the **vTPM attestation key** to PCRs 17+18

Make it accessible to **TD**





Security Guarantees
Data Center Execution Assurance

Attack Scope (ID)	System-Level Mitigation (Concrete)	Formal Model Mechanism (Abstract)
Relay & Proxy (A1: Proxy)	**Freshness & Channel:** Nonces are injected into both TD Report and TPM Quote.	**Challenge-Response:** The ideal $\mathcal{G}_{CloudTDX}$ requires nonce n_V for location proofs, which the simulator $\mathcal{S}$ enforces via the signature transformation.
Forgery & Compromise (A2, A4, A6)	**TXT Sealing:** The AK is sealed to PCR 17–18. The hardware enforces that the key is unusable if the host stack (A6) or vTPM binary differs from the measured launch state.	$\mathcal{G}_{TPM}$ **Access Control:** The global functionality enforces specific `Policy` checks over the simulated `PCRs`. If the adversary state differs, `TPM2_Quote` returns $\perp$.
Inconsistency & Substitution (A3, A5)	**Crypto Binding & Identity:** The TD embeds $hash(AK_{pub})$ in its report. The verifier validates the EK certificate chain against the manufacturer or provider CA.	W^{DCEA} **Binding Logic:** The wrapper explicitly computes binding B and aborts if it mismatches the vTPM output. $\mathcal{G}_{TPM}$'s registry $\mathcal{R}_{Root}$ is initialized only with valid EKs.

Security Guarantees
Data Center Execution Assurance

Proof Sketch:
Our proof relies on the reduction properties of the **AGATE** framework

Step 1: **Reduction** via **AGATE** composition.

Step 2: **Simulator** construction.

Step 3: **Handling** the proxy attack.

Step 4: **Indistinguishability**.

Shell 1: $sh_{CloudTEE}[\mathsf{prog}]$

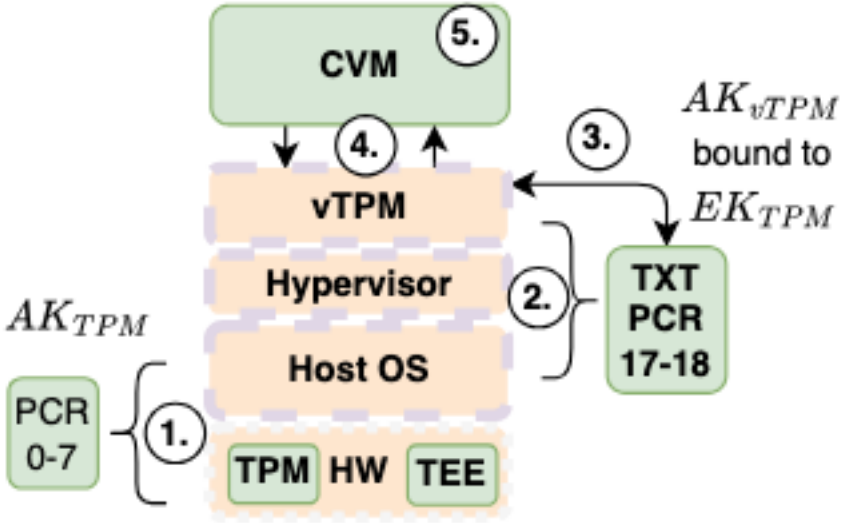
Parameters. Program prog.
Interfaces. $\mathbb{O} = \mathbb{O}^{std} \cup \{\mathsf{Quote}, \mathsf{GetLocationProof}\}$, $\mathbb{A} = \emptyset$.
Extended Identity. $(sh_{CloudTEE}[\mathsf{prog}], (eid||pid, \mathsf{att}||idx))$.
Init
Initialize empty set of virtual Turing Machine Instances (ITIs).
On input (INSTALL) from $\mathcal{G}_{att}^{mod}$:
If virtual ITI $(\mathsf{prog}, (eid, idx))$ does not exist:
Create virtual ITI $(\mathsf{prog}, (eid, idx))$.
If ideal functionality $(\mathcal{G}_{TEE}, (idx, \perp))$ does not exist:
Create ideal functionality $(\mathcal{G}_{TEE}, (idx, \perp))$.
(The shell ensures the existence of the helper functionality it relies on.)
On input (inp) from $\mathcal{G}_{att}^{mod}$:
Execute program $(\mathsf{prog}, (eid, idx))$ step by step on input inp.
For each instruction i:
If $i \in \mathbb{O}^{std}$:
Allow standard execution of i.
Else If $i = \mathsf{GetLocationProof}(\mathsf{nonce})$:
Send (LOC_PROOF, nonce) to $\mathcal{G}_{TEE}$.
Receive response λ.
Write λ to the subroutine output of $(\mathsf{prog}, (eid, idx))$.
Else If $i = (\textbf{return } v)$:
Output v with source $(sh_{CloudTEE}[\mathsf{prog}], (eid||pid, \mathsf{att}||idx))$.

Implementation Flow
Deployments in the Wild

	S1		S2	
Platform	**CVM**	**vTPM**	**Bare metal**	**HW TPM**
AWS	✗	✓	✗	✗
GCP	✓	✓	✓	✓
Azure	✓	✓	✗	✗
IBM Cloud	✓	○	✗	✗
OVH	✓	✗	✓	○

Implementation Flow
Bare Metal Deployment

1.&2. Measured Launch Anchor
3. AK provisioning and sealing
4. In guest binding
5. Compositive Verification



Summary

Introduced solution for **Data Center Execution Assurance (DCEA)** for **CVM** and **bare metal**

Combines two **individual** roots of trust from **TPM** to **TD**

Requirements on the **TPM** → certified by trusted party, e.g., GCP, or others for home deployments

Strengthens the **threat model** of current **TDs**

Generalizable, as long **RTMR** like **features** are present e.g., ARM CCA

Future work:

- Privacy of the **vTPM** quote certificate chain
- Possible use other **capabilities** of (v)TPM and platforms
- Complete **demos**

Discussion

- Multi-tenancy – side channels attacks from other VMs?
- Privacy – disclose less sensitive info
- Usability vs security
  - Hardening of CVM