Challenging the Boundaries of Confidential Computing for AI

A Cross-Component Attestation Study on GCP

Cybersecurity Engineering Technical Report

Version 1.1

April 25, 2025

PUBLIC

CENSUS S.A., 128 Andrea Syngrou Ave, 11745, Athens, Greece. Tel: +30-210-220-8989/8990, Email: eu@census-labs.com
Web: https://www.census-labs.com

Document History

Version	Revision	Date	Description of changes	Pages
1	0	2025-04-16	First version released and shared with Google & Intel for comments.	All
1	1	2025-04-25	Fixed typo in Figures 2 & 6, and in attestation verification Method-2. Added note about ITTA composite attestation status on GCP.	9, 19, 20, 31

Table of Contents

Abstract

This report presents a technical evaluation of Confidential Computing capabilities in Google Cloud Platform (GCP), focusing on the integration of Intel Trust Domain Extensions (TDX) and NVIDIA H100 GPUs. The objective is to assess whether hardware-based attestation can be achieved consistently across both CPU and GPU components within a Confidential VM and to identify potential limitations in establishing end-to-end trust in such hybrid configurations.

The study was conducted on early-access GCP A3 instances equipped with Intel Sapphire Rapids CPUs and NVIDIA H100 Tensor Core GPUs. Confidential VMs were deployed and assessed using open-source tooling alongside vendor services such as Intel's Trust Authority and NVIDIA's Remote Attestation Service (NRAS). The evaluation covered quote generation, measurement reproducibility, cryptographic key hierarchies, and runtime verification workflows.

While core attestation flows performed as expected, the evaluation surfaced some security limitations. These include: opaque firmware trust anchors that cannot be independently verified, quote signing mechanics that concentrate trust in vendors, inconsistencies in secure storage models, and the unavailability of Multi-Instance GPU (MIG) mode in confidential configurations. These findings suggest the need for further improvements in the surrounding ecosystem before consistent, verifiable trust guarantees can be delivered at scale.

The report outlines the technical architecture, attestation processes, and tooling used in this evaluation, with a summary of findings that highlight the current strengths and constraints of Confidential GPU deployments in cloud setups.

1 Introduction and Scope

As AI/ML workloads become increasingly sensitive and distributed, the demand for verifiable compute integrity in untrusted environments has accelerated. Confidential Computing has emerged as a key strategy for securing data in use, especially when combined with GPU acceleration for model training and inference. This report evaluates the integration of Intel TDX-based Confidential VMs with NVIDIA H100 GPUs on Google Cloud Platform (GCP), with the goal of determining whether attestation and trust can be extended consistently across both components.

This evaluation explores the feasibility of establishing end-to-end attestation workflows, identifying the trust boundaries and operational limitations of GCP's early-access A3 instances. The analysis uses vendor-provided attestation services (Intel Trust Authority, NVIDIA NRAS) and open-source tooling to validate CPU and GPU components, examine key hierarchies, and verify runtime integrity.

The sections that follow document the technical setup, attestation process, and deployment considerations. A dedicated Security Findings section summarizes the key observations and implementation inefficiencies uncovered during the evaluation.

2 Reference Environment

The environment is hosted on Google Cloud Platform (GCP) using early-access A3 instances, which combine Intel Sapphire Rapids CPUs with NVIDIA H100 Tensor Core GPUs. These machines were accessed through GCP's private preview program, in which CENSUS is a participating member.

At the time of testing, it was not possible to launch Confidential VMs with H100 GPUs directly via the GCP Web UI or CLI. Instead, instance groups were used to allocate the necessary resources. Once provisioned, setup typically completed within minutes, unlike standard H100 VM requests, which could remain pending for up to two weeks.

The base image setup followed Google's documentation, while NVIDIA's more recent deployment guidance was used to configure both host and guest environments. A custom initialization script was developed to automate the installation of required drivers and libraries at first boot. Overall provisioning took approximately 20 minutes.

After each reboot, the guest VM was required to transition the GPU into a "ready" state before workloads could run. Although this state is not technically defined or reflected in the attestation report, its enforcement acts as a practical safeguard, blocking execution until the environment is correctly initialized.

All further attestation and validation operations documented in this report are carried out within this controlled deployment.

3 CPU Attestation Evaluation

The CPU attestation workflow establishes the foundation of trust in the Confidential VM. In this evaluation, Intel Trust Domain Extensions (TDX) were used to provide memory isolation and support for remote attestation in a cloud-hosted virtual environment.

To generate attestation quotes from within the guest VM, the google/go-tdx-guest library was used. This library interacts with the Linux kernel's `configfs-tsm` subsystem, abstracting the interface to various Confidential Computing backends. However, an Intel TDX-enabled VM cannot generate a directly verifiable quote. It produces a local report, which must be handed off to a Quoting Enclave (TDQE) operating outside the VM. The TDQE signs the final attestation quote with a platform-derived key, producing a verifiable response suitable for remote challengers.

During testing, this quote was verified using both Intel Trust Authority and the publicly available Intel Provisioning Certification Service (PCS). The validation process examined key measurements recorded in the quote and compared them to known baselines or expected values.

A set of fields in the attestation quote were identified as critical for verification:

- `MRSEAM`: Measurement of the Intel TDX module. This value reflects the state and integrity of the system-level TDX runtime.
- `MRTD`: Measurement of the initial contents of the guest's Trust Domain (TD), typically capturing the state of the guest firmware (TDVF).
- `RTMR[0-3]`: Runtime Measurement Registers. These record extensible measurements during the boot process and runtime operations. They serve a role similar to TPM's Platform Configuration Registers (PCRs).
- `REPORTDATA`: A field provided by the guest at quote generation time. It typically contains a cryptographic nonce, hash of session-specific material, or application-defined metadata.

These illustrate the raw values used during evaluation to confirm measurement reproducibility.

It is important to note that **Intel has the ability to resign attestation keys using the PCK private key**, since the certificate chain remains valid regardless of who submitted the quote. This makes it essential for **clients** to validate the measurements and **embedded fields themselves**, rather than relying solely on the verdict of the attestation service.

An overview of the quote generation flow by an Intel TDX Guest is illustrated in the following figure.

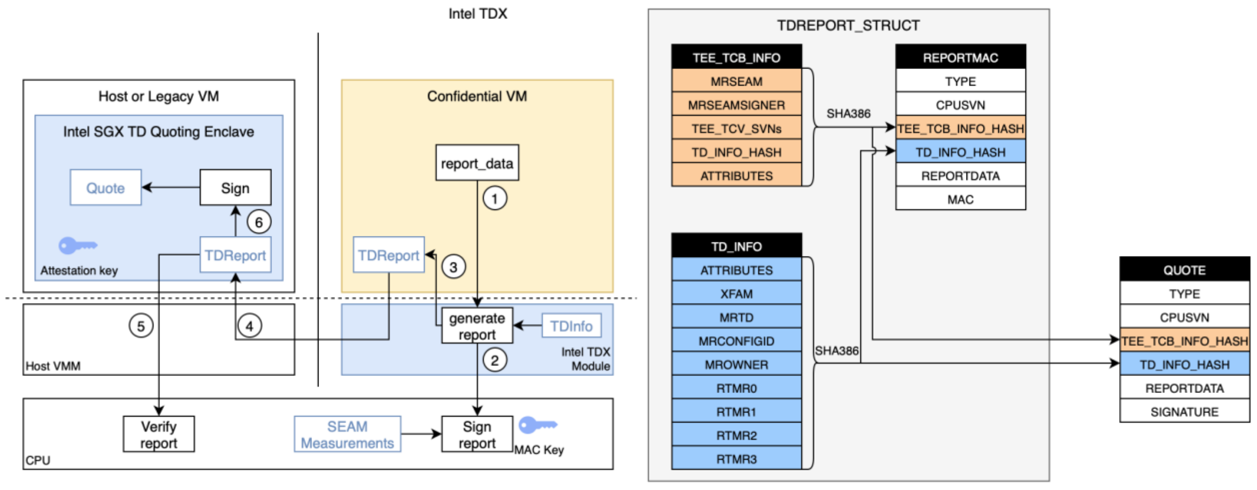

Figure 1 - Quote generation flow by an Intel TDX Guest

3.1 Cloud Image Validation

To establish trust in a Confidential VM, it is essential to verify that the VM was launched with a known and intended system configuration. This is achieved by validating a combination of launch-time and runtime measurements embedded in the attestation quote, specifically the `MRTD` and `RTMR[0-3]` registers.

A dedicated proof-of-concept implementation was developed to test the reproducibility of these measurement values. The process involved:

1. Fetching the latest Ubuntu 24.04 cloud image,
2. Generating a Unified Kernel Image (UKI) bundling the kernel, initrd, and command-line arguments,
3. Deploying the image to GCP using Confidential VM-compatible settings,
4. Parsing the vTPM event log to extract expected values and compare them against those reported in the attestation quote.

This reproducibility check demonstrated that several registers, particularly `RTMR1`, could be matched precisely. However, certain aspects remain nuanced:

- `RTMR2` and `RTMR2` were observed to remain in their default (zeroed) state. This is expected in configurations that directly boot a UKI and do not use intermediate bootloaders such as GRUB.
- `RTMR0` was not precomputed during the PoC and remains more complex to reproduce due to variability introduced by runtime extensions or kernel-level events.
- `MRTD`, which reflects the measurement of the Trust Domain Virtual Firmware (TDVF), was reproducible using Google's `gce-tcb-verifier` tool, though the underlying firmware source remains partially opaque.

This validation process provides strong evidence that baseline measurements can be reliably precomputed and matched against live quotes, enabling a reproducible trust framework for Confidential VM deployments.

3.2 Key Hierarchy and Scope

The attestation mechanism for Intel TDX relies on a layered key hierarchy rooted in hardware-fused secrets. These keys form the foundation for generating secure and verifiable attestation quotes and enable the derivation of trust across the platform.

At the hardware level, two processor-unique keys are fused during manufacturing:

- **Root Provisioning Key (RPK)**: Maintained by Intel within a hardware security module (HSM). It acts as the root of trust for platform provisioning operations and is used to derive the Platform Provisioning ID (PPID).
- **Root Seal Key (RSK):** A unique per-chip key that is never stored or exported. It enables sealing of secrets specific to the platform instance.

Both keys are encrypted using the Fuse Encryption Key (FEK), also referred to as the Global Wrapping Key (GWK), a 128-bit AES key shared across processors of the same microarchitecture. This design allows Intel to maintain selective control over provisioning operations while ensuring that per-device keys remain secure.

From these roots, additional attestation-specific keys are derived:

1. **Derived Provisioning Key (DPK):** Derived from the RPK, used to generate the PPID. The PPID is encrypted using Intel's public **PPID Encryption Key (EPK)** before being sent to the Provisioning Certification Service (PCS). This encryption step helps prevent direct hardware identity exposure and mitigates linkability between requests, preserving platform anonymity while still allowing PCS to retrieve the associated platform's provisioning data.
2. **Attestation Key (AK):** Generated inside the TDQE using the CPU's EGETKEY instruction. This elliptic-curve key is used to sign the attestation quote and is certified by the PCS using the Platform Certification Key.
3. **Platform Certification Key (PCK):** A processor-specific key pair used to sign attestation keys. The private part is accessible only to Intel, and the public certificate chain is issued by the PCS.
4. Supporting Keys:
    a. **Seal Key**: Used for securely storing data outside the enclave between restarts.
    b. **Report Key**: Enables local attestation between enclaves on the same platform.
    c. **EINIT Token Key**: Used during enclave creation to validate that the requesting enclave is allowed to initialize.
    d. **Platform Provisioning ID**: Serves as a database lookup key for the PCS to retrieve the correct key material and TCB information.

The AK and its associated certificate chain are included in the final attestation quote. The chain of trust is rooted at Intel's Root CA, followed by the PCK CA, and finally the PCK itself. The PCK certifies the attestation key, enabling external parties to validate both the signature and platform integrity.

The complete flow of key derivation and certificate issuance was validated during this evaluation, and no deviations were observed from the expected key usage model. The structure of this hierarchy and the relationships between key derivations and certificate authorities are illustrated in the following figure.

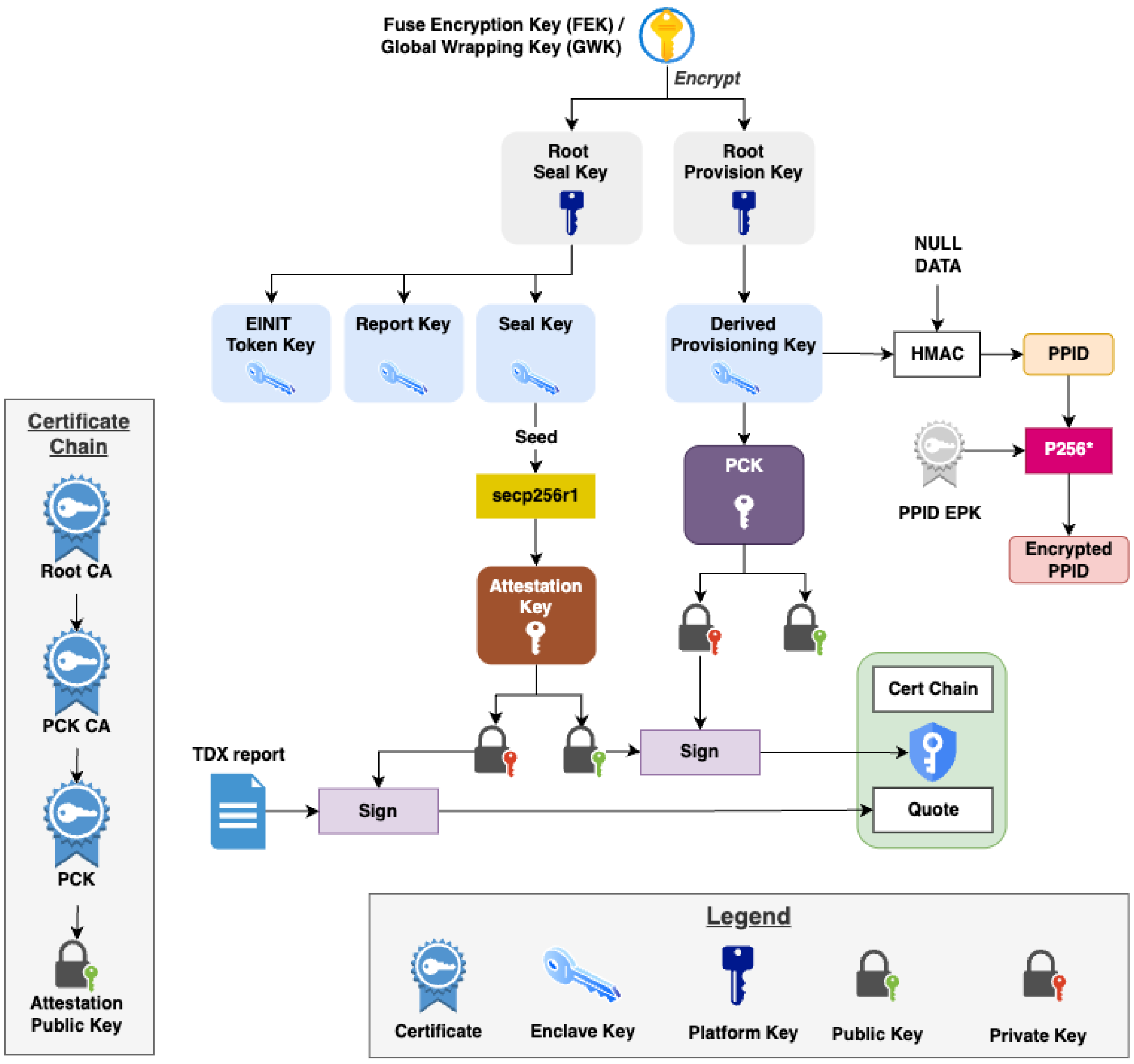

Figure 2 - Intel TDX Attestation Key Hierarchy

3.3 Attestation Sequence

The attestation flow in Intel TDX involves a multi-stage process that begins inside the guest VM and concludes to a signed quote that can be validated externally using Intel's certificate infrastructure. This flow includes report generation, key provisioning, quote signing, and remote verification by the client.

The sequence is as follows:

1. Local Report Generation (Guest VM)

The guest TD uses the `SEAMREPORT` instruction to generate a report containing key measurements, such as:

- `MRTD`: Launch measurement of the TD's memory image.
- `RTMR[0-3]`: Runtime extensible measurement registers.
- `REPORTDATA`: Arbitrary 64-byte payload provided by the guest (typically a nonce or hash).

This report is valid only within the platform and cannot be verified externally.

2. Attestation Key Generation (TDQE)

The TD Quoting Enclave (TDQE), running outside the guest, uses the `EGETKEY` instruction to deterministically derive an attestation key from the Root Seal Key. This key is used to sign the quote that will later be sent to the challenger.

3. Attestation Key Certification (PCE and PCS)

The TDQE constructs an attestation key report and submits it to the Provisioning Certification Enclave (PCE). The PCE verifies the report and requests a Platform Certification Key (PCK) from Intel's Provisioning Certification Service (PCS). The PCK is then used to sign the attestation key.

4. Quote Construction and Signing (TDQE)

The TDQE converts the guest report into an externally verifiable attestation quote and signs it using the attestation key. The response returned to the client includes:

- The signed quote.
- The signed attestation key.
- The full certificate chain (PCK, PCK CA, Root CA).

5. Quote Verification (Client)

The challenger performs the following checks:

- Validates the certificate chain starting from Intel's Root CA.
- Verifies the signature on the quote.
- Confirms measurement values such as `MRSEAM`, `MRTD`, and `RTMR[0-3]`.
- Matches the `REPORTDATA` field against the expected nonce or session hash.

In some flows, the client may optionally consult Intel Trust Authority for attestation token generation and verdict delivery (covered in 3.4), but this is not strictly required.

This sequence is visualized in Figure 3, which depicts the interactions between the guest, TDQE, PCE, PCS, and the remote verifier.

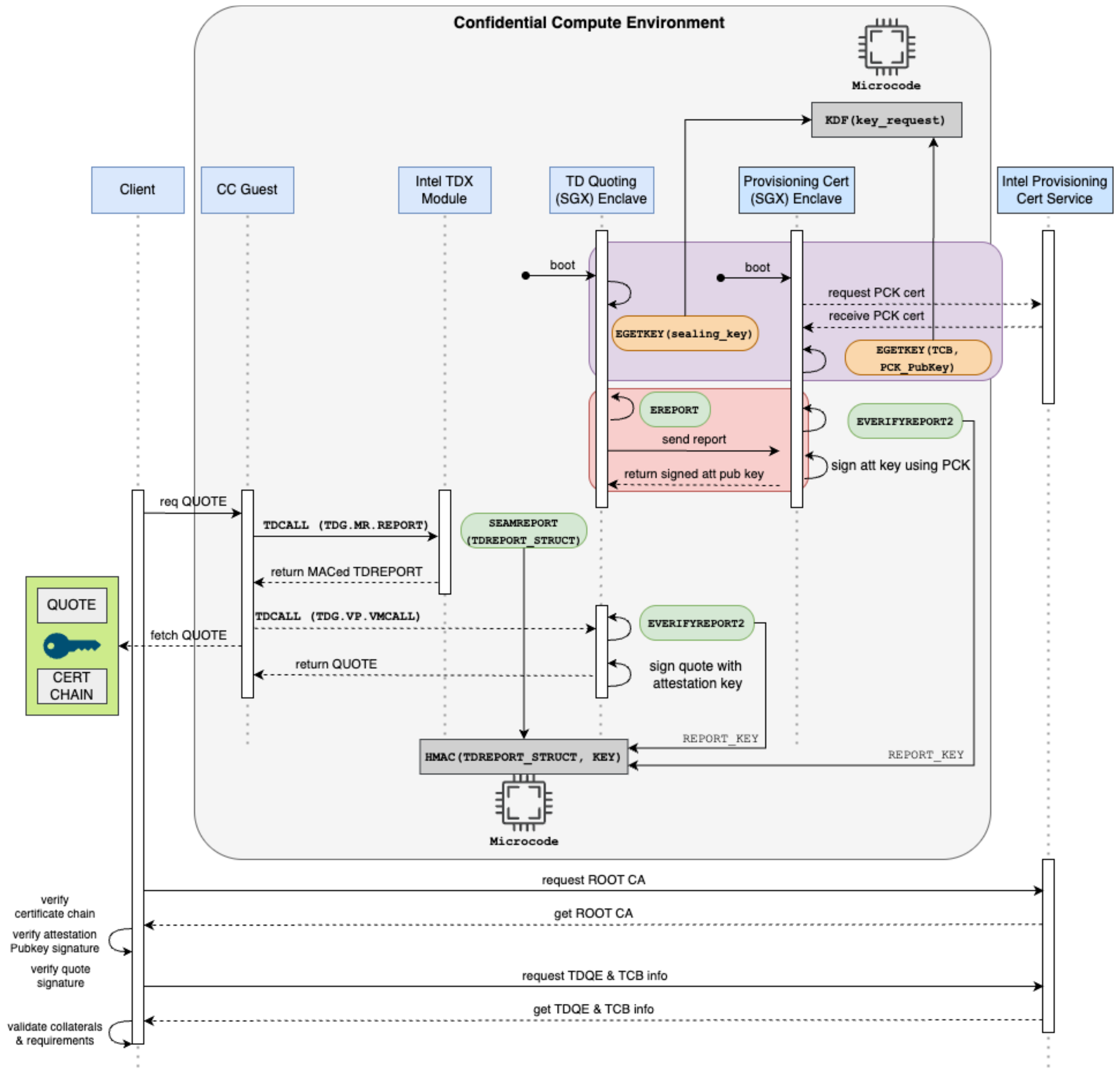

Figure 3 - CPU Attestation Sequence with Intel TDX

The robustness of this attestation chain depends on the integrity of the TDX module and Intel's certification services. While the overall chain is verifiable using public materials, any failure to validate measurement registers or misalignment with policy expectations can result in an unverifiable or untrusted attestation state.

3.4 Intel Tiber Trust Authority Flow

Intel Trust Authority (ITTA) provides an API-based appraisal service that allows clients to validate attestation quotes produced by Confidential VMs running under Intel TDX. Rather than relying on manual quote validation and certificate path resolution, the client submits the quote to ITTA and receives a signed attestation token containing an assessment of the attester's state and measurements.

The flow implemented during this evaluation consisted of the following steps:

1. Quote Submission

The attestation quote generated by the TDQE is submitted to the ITTA appraisal endpoint via a POST request. The payload includes the base64-encoded quote string.

```
curl --request POST \
  --url https://api.eu.trustauthority.intel.com/appraisal/v1/attest \
  --header 'Accept: application/json' \
  --header 'Content-Type: application/json' \
  --header 'x-api-key: $API_KEY' \
  --data '{ "quote": "BAAC...AAAAAAA=" }'
```

2. Token Response

ITTA returns a signed attestation token formatted as a JWT. This token embeds key quote fields and metadata used to evaluate the trustworthiness of the platform.

For example:

```
{
  "token": "eyJhbGciOiJQU...43lqj4z"
}
```

3. Embedded Quote Fields

The decoded token contains fields derived from the original quote, including:

- Static Configuration: `tdx_mrconfigid`, `tdx_mrowner`, `tdx_mrownerconfig`
- SEAM & TDX Runtime: `tdx_mrseam`, `tdx_mrsignerseam`, `tdx_seam_attributes`
- Guest Measurements: `tdx_mrtd`, `tdx_report_data`, `tdx_rtmr0 - tdx_rtmr3`
- Security Posture: `tdx_is_debuggable, dbgstat`

- Policy Evaluation: `policy_ids_matched, policy_ids_unmatched`
- TCB Status: `attester_tcb_status` (derived from Intel PCS)

These values are used by relying parties to assess whether the attesting VM complies with expected platform integrity and policy criteria.

4. Client-Side Responsibilities

Although ITTA performs initial validation of the quote and returns a signed result, the client remains responsible for:

- Verifying the JWT signature on the returned token,
- Validating that embedded fields match challenge-specific values like nonces,
- Interpreting claims such as `attester_tcb_status` and policy match results in accordance with their own trust model.

This separation of responsibilities streamlines attestation processing but also introduces an **additional trust anchor, the attestation service itself**.

As illustrated in the following figure, all the quote validation is performed by the Trust Authority service, and the client, beyond trusting the verdict, should still manually verify the signature of the attestation token and cross-check the embedded values against those included in the submitted quote. The validation of the `attester_tcb_status` field in the response depends on Intel PCS. Additional claims, such as `tdx_is_debuggable`, `dbgstat`, `policy_ids_matched,` and `policy_ids_unmatched`, require interpretation against manually defined policies.

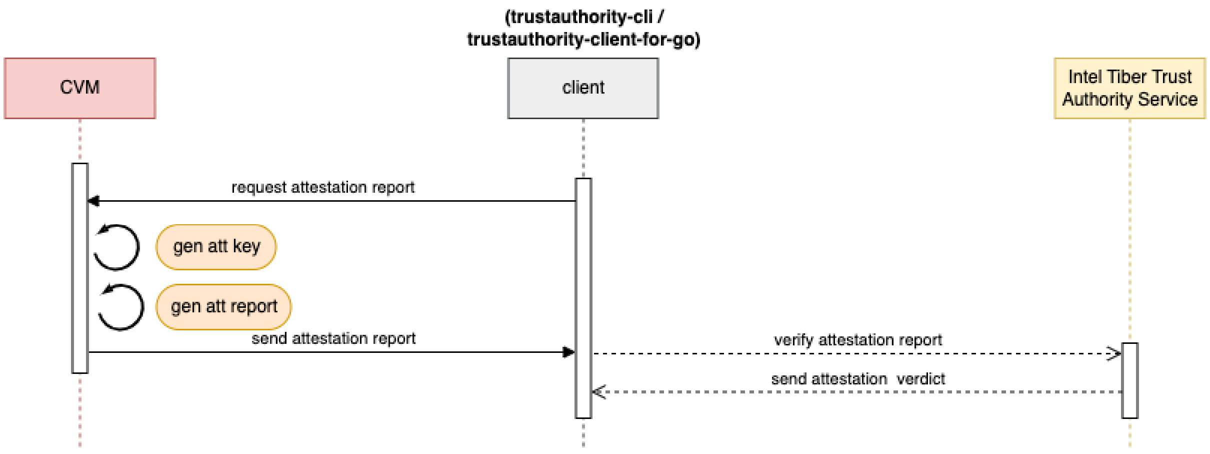
(trustauthority-cli / trustauthority-client-for-go)
CVM
client
Intel Tiber Trust Authority Service
request attestation report
gen att key
gen att report
send attestation report
verify attestation report
send attestation verdict

Figure 4 - TDX Quote Validation Workflow via Intel Trust Authority

This sequence summarizes the interactions between the attesting VM, ITTA, and the client verifier.

3.5 ICS Attestation Flow

As an alternative to Intel Trust Authority, attestation can be performed manually by interacting directly with Intel's Provisioning Certification Service (PCS). This approach, commonly referred to as the ICS attestation flow, enables clients to independently validate attestation quotes without relying on external appraisal services.

This method is based entirely on publicly documented Intel APIs and provides greater transparency and control over the validation process. The flow consists of the following steps:

1. Quote Parsing (Client)

The attestation quote is generated by the TD Quoting Enclave (TDQE) and sent to the client. Using libraries such as `google/go-tdx-guest`, the client parses the quote and extracts key fields:

- Platform measurements: `MRSEAM, MRTD, RTMR[0–3]`
- Session data: `REPORTDATA`
- Configuration and debug attributes

2. Collateral Retrieval (Client → PCS)

The client queries Intel PCS to retrieve the necessary collateral for verifying the quote:

- Platform Certificate Chain (PCK, PCK CA, Root CA)
- TCB Info for the target platform
- Certificate Revocation Lists (CRLs)

These assets are made available via Intel's publicly documented REST APIs.

3. Attestation Key & Certificate Chain Validation

The client verifies that:

- The quote is correctly signed using the attestation key,
- The attestation key is certified by the PCK,
- The full chain terminates at Intel's Root CA,
- All certificates are valid and not revoked.

4. Measurement Verification

The extracted fields from the quote are evaluated against reference values:

- `MRTD` and `RTMR[0-3]` compared against known baselines,
- `REPORTDATA` validated against the expected nonce or challenge,
- Attributes like `tdx_is_debuggable` and `tdx_seam_attributes` are checked for consistency with policy.

While this approach requires more effort than using Intel Trust Authority, it **avoids expanding the Trusted Computing Base (TCB) to include a third-party validation service**. This can be advantageous in scenarios where tighter control over trust decisions is required or where regulatory constraints exist.

Figure 5 illustrates the interaction between the client, the Confidential VM, and Intel PCS.

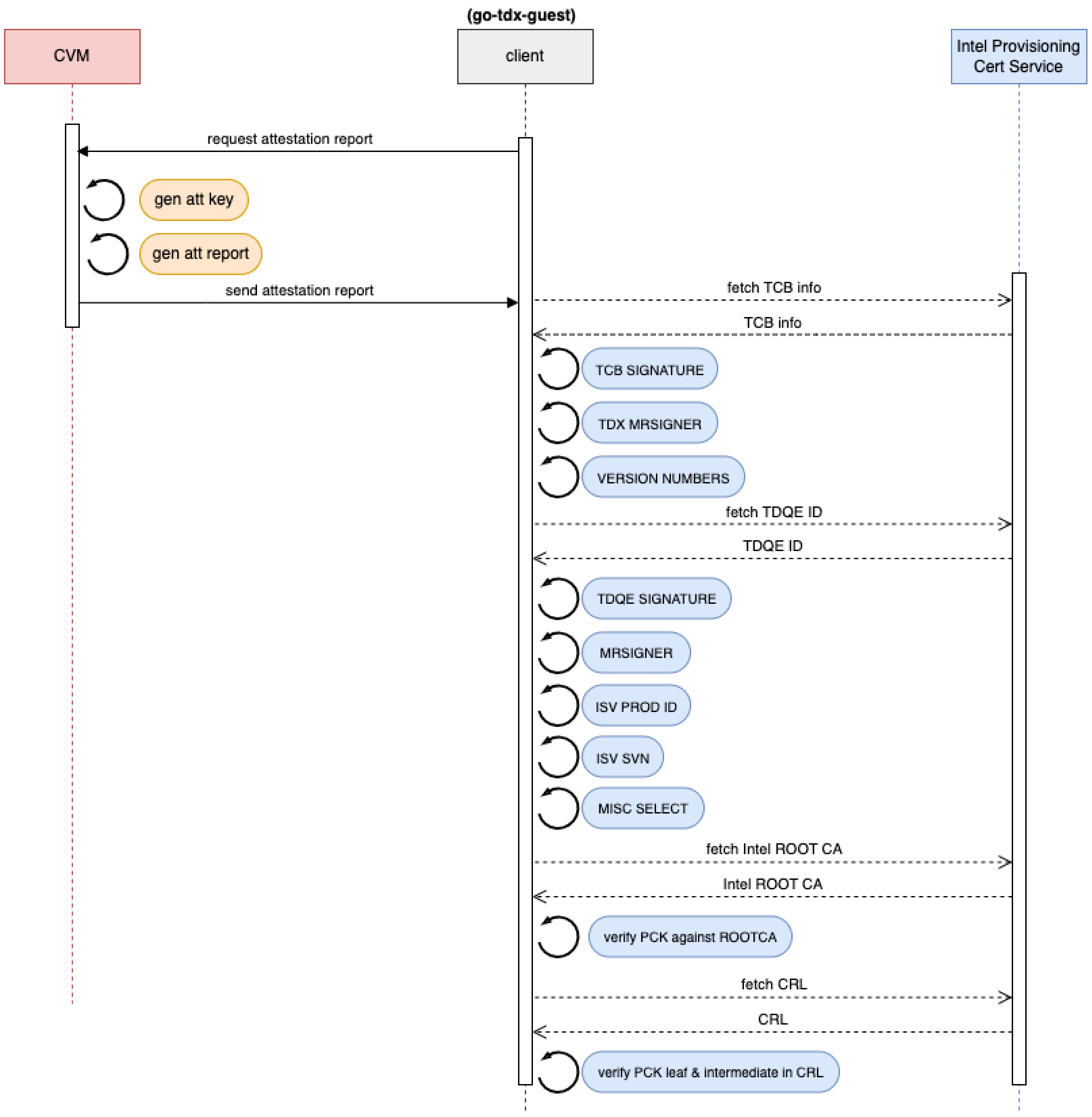

Figure 5 - ICS Attestation Flow – Client Validation via Intel PCS

3.6 TCB Build Reproducibility

Reproducibility of the Trusted Computing Base (TCB) is an important consideration for attestation workflows, as it allows the relying party to confirm that platform software components match expected builds. During this evaluation, build reproducibility was assessed for both the Intel SGX stack and the Intel TDX module.

3.6.1 Intel SGX

Intel provides a reproducibility framework for SGX enclaves through a dedicated GitHub repository (`intel/linux-sgx`). The repository includes a `sgx_reproducible` branch with deterministic build instructions and squashed commits aligned with official SDK releases. The build process relies on a controlled environment using:

- Ubuntu 20.04 as the base system,
- The Nix package manager to enforce consistent toolchain versions,
- Docker to encapsulate the build environment.

This setup enables end-to-end reproducibility of enclave binaries, including:

- Intel SGX SDK and IPP cryptographic libraries,
- Enclave source code and build configuration,
- The TD Quoting Enclave (TDQE), which is critical for Intel TDX quote generation.

As part of the evaluation, the team reproduced the enclave corresponding to `MRENCLAVE` value `e5a3a7b5d830c2953b98534c6c59a3a34fdc34e933f7f5898f0a85cf08846bca`, which maps to the DCAP version 1.19 (shipped in SGX releases 2.22–2.24). Using the `sgx_2.22_reproducible` tag, the binary was successfully rebuilt and matched the known enclave hash. The verification was completed using the `reproducibility_verifier.sh` utility. This script allows users to compare an Intel-signed enclave with a locally built unsigned version, checking that the `MRENCLAVE` values match. An example usage is shown below:

```
# openssl genrsa -out dummy_key.pem -3 3072
# ./reproducibility_verifier.sh $SIGNED_BIN $UNSIGNED_BIN dummy_key.pem
$ENCLAVE_CONFIG

* intel signed AE    : /root/linux-
sgx/linux/reproducibility/code_dir/sgx/external/dcap_source/QuoteGeneration/psw/ae
/data/prebuilt/libsgx_tdqe.signed.so
* user unsigned AE   : /root/linux-
sgx/linux/reproducibility/code_dir/out/ae/tdqe.so
* user private key   : dummy_key.pem
* intel config.xml   : /root/linux-
sgx/external/dcap_source/QuoteGeneration/quote_wrapper/tdx_quote/enclave/linux/con
fig.xml

Reproducibility Verification PASSED!
```

This approach is effective for validating the integrity of enclaves used in the attestation process and provides a reproducible baseline for SGX-enabled components in the Intel TDX stack.

3.6.2 Intel TDX Module

At the time of this evaluation, no equivalent reproducibility framework was available for the Intel TDX module. Although measurements such as `MRSEAM` are included in the attestation quote, there is currently no official documentation or tooling that allows third parties to independently reproduce the TDX module and validate its measurement hash.

As a result, `MRSEAM` must be **treated as a black-box value** unless the TDX module implementation is made publicly verifiable. This limits the ability of external challengers to confirm the integrity of the `SEAM` runtime and increases reliance on the platform provider to supply a known-good value.

This asymmetry between SGX and TDX reproducibility highlights a gap in current transparency and verifiability tooling for emerging Confidential Computing platforms.

3.7 Deployment Considerations

The deployment model for attestation in this environment is shaped by fundamental architectural differences between Intel SGX and TDX, particularly with respect to where enclaves are executed and what can be cryptographically verified in attestation reports.

3.7.1 Execution Model Differences

Intel SGX enclaves cannot run inside TDX-based Confidential VMs. SGX enclaves are user-mode applications that operate in standard host environments with SGX support. In contrast, Intel TDX relies on system-level components, specifically, the TDX Module and SEAM, to provide memory isolation and attestation capabilities within virtualized environments.

In the context of cloud infrastructure, enclaves used for attestation (e.g., TD Quoting Enclaves) are deployed by the cloud provider and are signed by Intel. These enclaves may run either directly on the host OS or within a separate SGX-enabled VM, depending on how the platform is configured. From the perspective of a TDX guest, this distinction is not observable, and SGX's threat model provides protection in both scenarios. **This means that while TDX guests can trigger attestation, they do not run their own enclaves**.

3.7.2 Verification of Enclave Signers

In SGX-based attestation, the identity of the enclave developer can be cryptographically verified using the `MRSIGNER` field. This field represents the SHA-256 hash of the public key used to sign the enclave and allows relying parties to enforce signer-based policies.

For TDX, the corresponding field (`MRSIGNERSEAM`) is intended to serve a similar purpose by representing the signer of the TDX Module (SEAM). However, during this evaluation, and based on

Intel-provided guidance, the `tdx_mrsignerseam` field was observed to be `null`. Intel has indicated that this behavior is expected when Intel itself is the signer.

Although both `MRSIGNER` and `MRSIGNERSEAM` are defined fields and are supposed to contain valid hash digests, the null value in `tdx_mrsignerseam` suggests that this field may be unused, rather than undefined. This interpretation is supported by Intel documentation and by behavior observed in the decoded attestation tokens.

Specifically, the validity of `tdx_mrsignerseam` appears to depend on a *field-validity bitmap* encoded in the structure, where the first member determines whether certain fields are to be interpreted as active. When the corresponding bit is unset, the field may be ignored by downstream verifiers.

As a result, relying parties currently cannot cryptographically verify the signer of the TDX Module in cloud deployments, even though they can verify the TDX measurements (e.g., `MRSEAM`) and other structural fields.

4 GPU Attestation Evaluation

In addition to CPU-level attestation using Intel TDX, this evaluation explored GPU-level attestation for NVIDIA H100 devices within the Confidential VM. The aim was to assess whether a complete chain of trust could be established that includes the GPU runtime, and to validate the correctness and verifiability of its attestation reports.

NVIDIA provides two main components to support GPU attestation:

1. **NVIDIA Remote Attestation Service (NRAS):** a cloud-hosted API that receives GPU measurements and returns a signed attestation token.
2. **Confidential Computing Attestation SDK (CCA-SDK):** a user-space library that facilitates local attestation report generation within the guest environment and prepares quotes for submission to NRAS.

The attestation process is centered on extracting the GPU's internal measurements and submitting them to NRAS for validation. The flow includes the following high-level steps:

1. Initializing the GPU device via CUDA/Xid interface,
2. Generating an attestation report locally using the SDK,
3. Sending the report to NRAS for remote appraisal,
4. Receiving a signed attestation token (JWT or similar format) containing claims about the GPU's configuration and trustworthiness.

It should also be noted that, at the time of writing, the ITTA composite attestation method (CVM + GPU) was not functional on GCP due to compatibility issues. As a result, a manual chaining approach was implemented, combining the attestation sequences of the two components.

This section outlines how the tools were integrated into the evaluation setup, the structure and content of the attestation report, and the constraints encountered during full-stack validation. The specific attestation flow, key hierarchies, and runtime conditions are detailed in the following subsections.

4.1 Implementation Details

The implementation of the attestation process is divided into two parts: a server and a client. The server exposes two endpoints, one over plaintext for attesting the CPU, and another over HTTPS for attesting the GPU. The hash of the server's certificate is embedded in the CPU attestation quote, ensuring that the client is communicating with the same Intel TDX server that generated the quote. The attestation process proceeds as follows:

1. The client initiates the process by sending a request to `/attest` with a locally generated nonce:
    a. The server generates an Ed25519 key pair and self-signs a certificate embedding the nonce.

   b. The server generates an attestation quote, including the nonce and a hash of the certificate in the `report_data` field.
   c. The server returns both the certificate and the attestation quote to the client.
2. The client verifies the quote signature using the Intel SGX Root CA (`intel-sgx-root-ca.pem`):
   a. The client checks the `MRSEAM`, `MRTD`, and `RTMR` register values against known good values.
3. The client verifies that the nonce embedded in the quote matches the original nonce generated in Step 1.
4. The client verifies that the hash of the certificate embedded in the quote matches the certificate returned in Step 1(c).
5. The client connects to the GPU attestation server via HTTPS at the `/attest_gpu` endpoint:
   a. The client verifies the TLS certificate of the server matches the one retrieved in Step 1(c).
   b. The server generates a new nonce.
   c. The server checks whether NVIDIA Confidential Computing is enabled, verifies the GPU is not in development mode, and confirms the GPU is in a ready state.
   d. The server performs GPU attestation.
   e. The server returns an Entity Attestation Token (EAT) to the client.
6. The client validates the EAT token against the NVIDIA Remote Attestation Service (NRAS) and evaluates the claims using a defined policy (`NVGPURemotePolicyToken.json`).
7. The client uploads the workload through the same server used in Step 5.

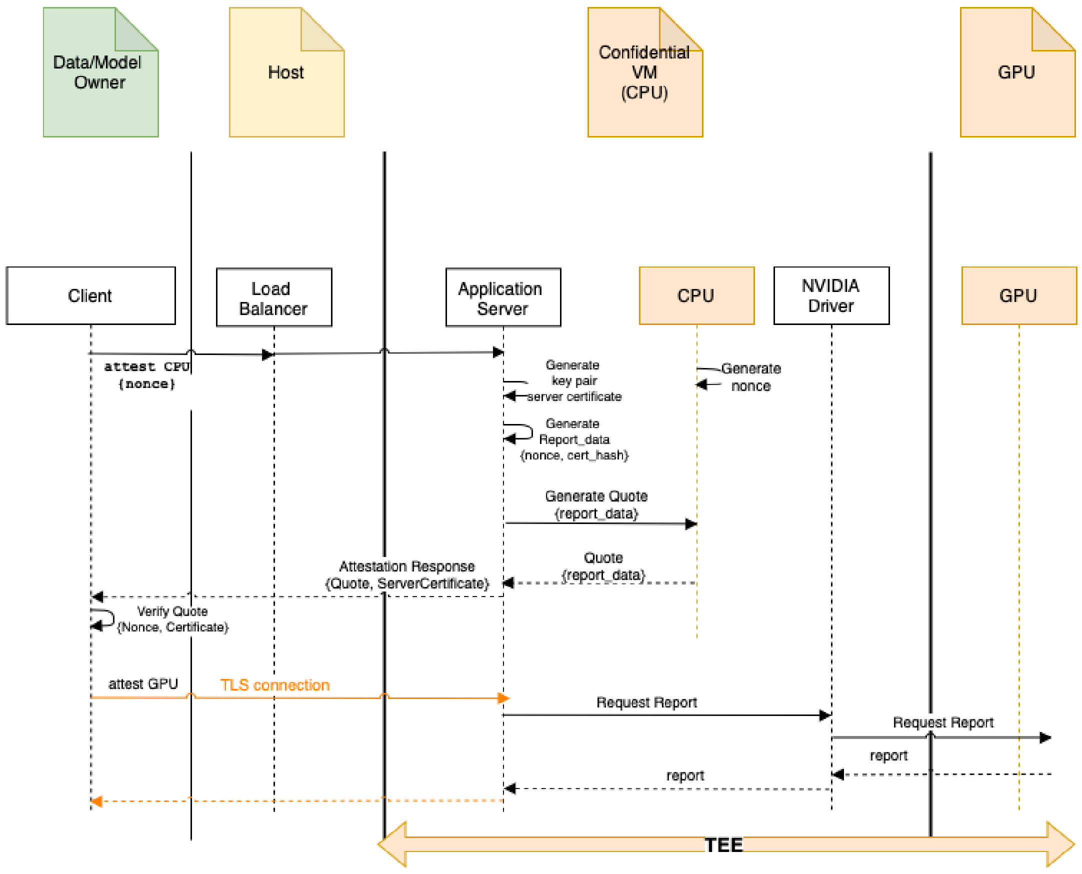

Figure 6 – Attestation Flow

The entire NVIDIA H100 GPU attestation flow is illustrated below:

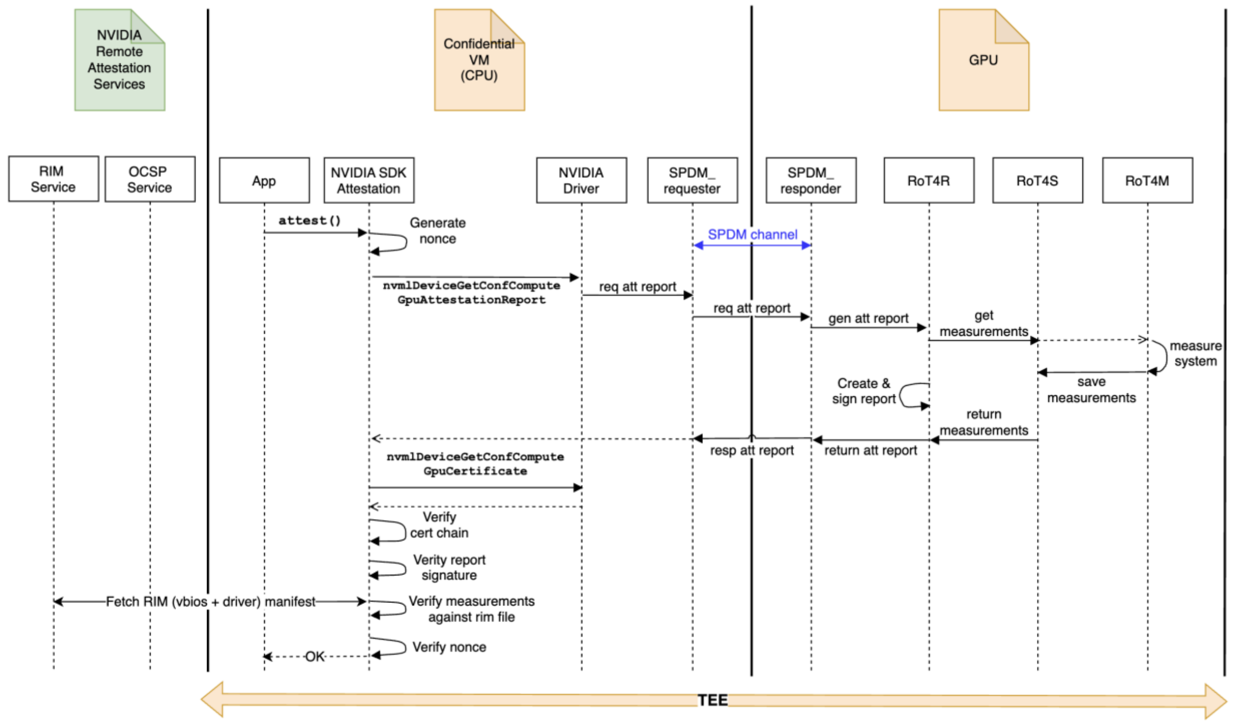

Figure 7 - NVIDIA H100 GPU Attestation Flow

4.2 Data Channels and Key Hierarchy

Data movement between the CPU (host) and GPU (device) is handled by copy engines, which are divided into Host-to-Device (H2D) and Device-to-Host (D2H) channels. Each Logical Copy Engine (LCE) maps to a Physical Copy Engine (PCE) and is configured for a specific transfer direction. Every channel is associated with dedicated cryptographic keys, which are derived from a secure key hierarchy rooted in the SPDM protocol.

There are two types of channels in this architecture:

1) CE-backed channels, used for:
    a. CPU-to-GPU and GPU-to-CPU data transfers
    b. In-GPU memory movement
    c. Page table entry (PTE) copies
    d. GPU-to-GPU memory transfers
2) Non-CE channels, used for:
    a. CPU-to-SEC2 encrypted data movement
    b. The Work Launch Channel (WLC) when Confidential Computing is enabled
    c. The Launch Confirmation Indicator Channel (LCIC), which complements WLC

The GPU driver maintains a key store consisting of 18 key-spaces: 1 for the GSP (GPU Security Processor), 1 for the SEC2 microcontroller, and 16 for the individual LCEs.

Each key-space is uniquely identified and mapped to a kernel RM (Resource Manager) channel. Keys are unidirectional, with separate entries for: Encryption (AES-based), and Authentication (HMAC-based) operations.

Keys are identified by a key ID, formed by combining the key-space ID with fixed per-key parameters. Keys are derived using the HKDF (HMAC-based Key Derivation Function). While HKDF allows for optional pre-shared keys (PSKs), the NVIDIA GPU driver does not use PSKs in this implementation.
As shown in Figure 8, the store provides 112 key slots, broken down as follows:

- 10 slots for **GSP**-related communication,
- 6 slots for SEC2 operations,
- 96 slots reserved for **LCE (CE-backed)** channels.

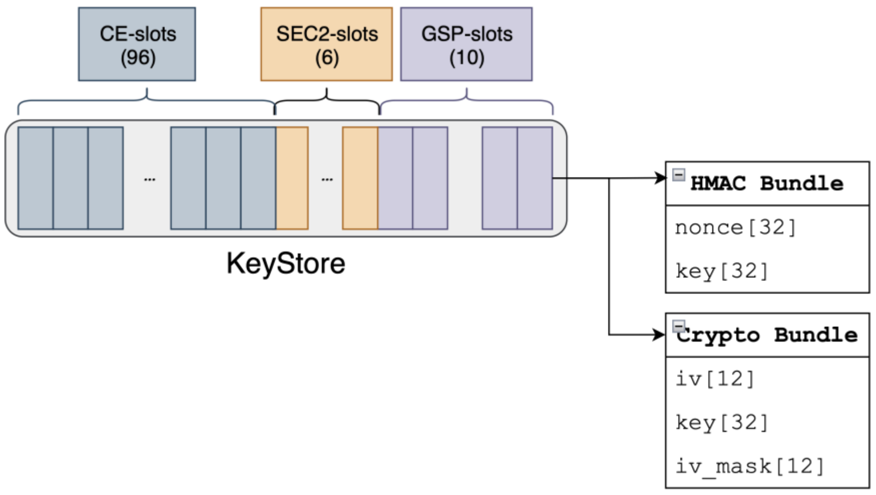

Figure 8 - GPU Key Store Layout

Each slot may hold either an encryption key bundle or an HMAC bundle, which includes a cryptographic key and a nonce or initialization vector (IV). Slot behavior differs slightly across domains:

- SEC2 and CE slots include monotonic counters in their IVs, incremented on each key update,
- GSP slots do not use counters.

Out of 112 total slots, 44 are actively used in the current configuration:

1. CPU–GSP channels:

- Two D2H/H2D keys for DMA-based transactions,
- Two D2H/H2D keys for locked GSP RPCs,
- Two D2H keys for fault buffer reads (H2D faults are not supported).

2. CPU–SEC2 channels:

- Two H2D keys for user-level data encryption and signing,
- Two H2D keys for kernel-level data encryption and signing,
- Two H2D keys for memory scrubber transactions.

3. CPU–CE channels (slots 2 to 9) :

- Each CE channel uses 4 keys (2 for user data, 2 for kernel data),
- Keys are unidirectional (H2D and D2H),
- Each CE channel is assigned a distinct namespace, isolating its key derivation context (see Figure 9).

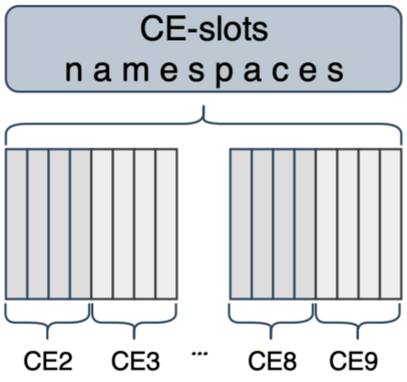

Figure 9 - CE Slot Namespaces

Key derivation across these slots follows a consistent hierarchy, starting from a hardware-embedded root and expanding into multiple branches, including:

- Encryption keys,
- HMAC signing keys,
- Key-wrapping keys used during provisioning and updates.

The full derivation path is shown in Figure 10. Each derived key is tied to a slot index, usage purpose, and domain-specific counters, ensuring isolation across workloads and device functions.

SEC2 channels play a critical role in launching workloads securely. They handle encrypted data transfers from untrusted host memory to protected GPU video memory (vidmem), coordinated by the SEC2 microcontroller.

Untrusted memory shared between CPU and GPU is populated with encrypted bundles. These bundles are moved across SEC2 or CE channels using authenticated encryption (AE) schemes to ensure confidentiality and integrity during transfer.

While CE channels are architecturally designed to support GPU-to-GPU peer transfers, this feature is not yet enabled in the current driver and firmware release.

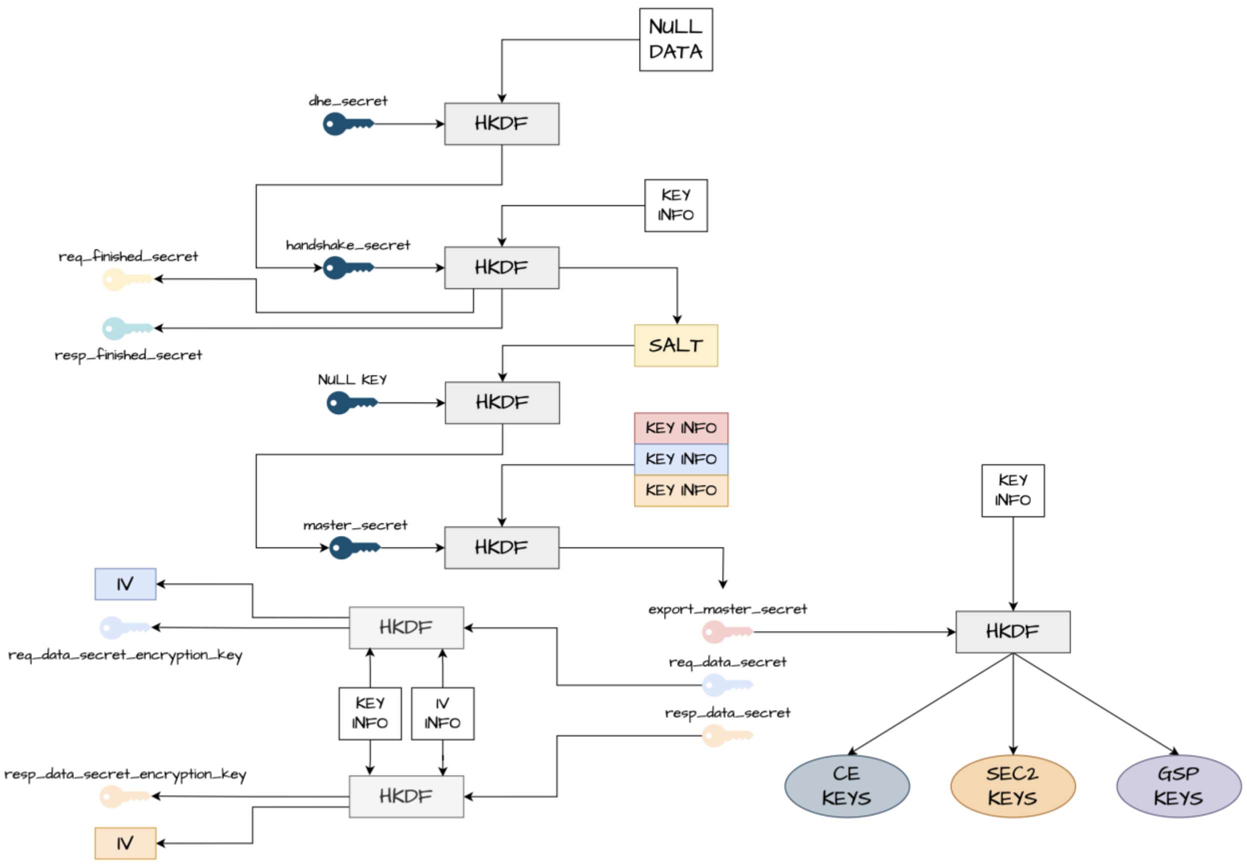

Figure 10 - Key Derivation Hierarchy

4.3 Deployment Considerations

NVIDIA H100 supports Multi-Instance GPU (MIG) operation, allowing a single GPU to be partitioned into multiple isolated compute units. This feature is particularly important for enabling multi-tenant deployment scenarios, where distinct tenants or workloads share the physical GPU while maintaining strong isolation.

Each MIG partition includes a dedicated portion of GPU compute resources, memory, and bandwidth. Critically, each partition operates with its own driver context, which implies an independent key store and cryptographic isolation. This architecture aligns with the zero-trust model used in Confidential Computing and ensures that derived keys, memory regions, and execution contexts are not shared between tenants.

This architectural model is illustrated in Figure 11 which shows both single-tenant (full-GPU) and multi-tenant (MIG-based) configurations.

In a multi-tenant MIG setup:

- Each tenant VM runs an isolated instance of the NVIDIA driver.
- Each instance maintains a separate key derivation scope.
- No cryptographic material is shared across MIG boundaries.

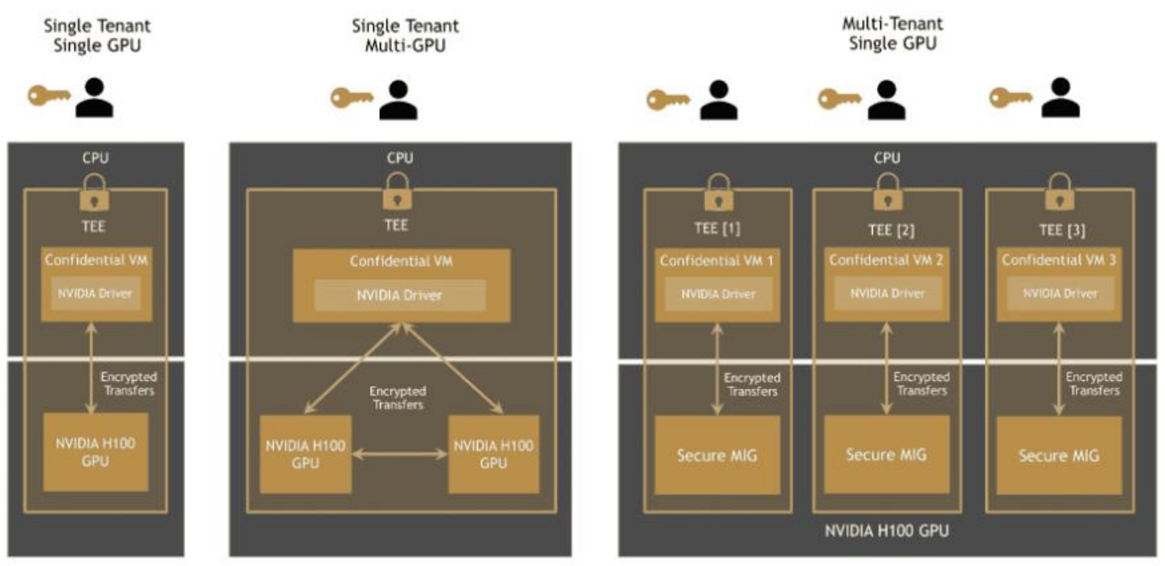

Figure 11 - H100 Deployment Topologies Using MIG[1]

4.3.1 Observed Limitations

During testing, MIG support could not be enabled on the H100 instance used in this evaluation. The following command failed to enable MIG mode:

```
$ sudo nvidia-smi -i 0 -mig 1
Unable to enable MIG Mode for GPU 00000000:04:00.0: Not Supported
Treating as warning and moving on.
All done.
```

Subsequent checks confirmed that MIG was not active:

```
$ nvidia-smi -i 0 --query-gpu=pci.bus_id,mig.mode.current --format=csv
pci.bus_id, mig.mode.current
00000000:04:00.0, [N/A]
```

The cause of this limitation was not determined during the experiment. It may stem from:

[1] Source: *NVIDIA H100 Tensor Core GPU Architecture*

- Cloud provider instance type restrictions,
- Hardware-specific configuration,
- Driver or firmware support limitations within Confidential VMs.

4.3.2 Driver Attestation Security Inefficiency

When initializing the GSP firmware resource manager (RM), the NVIDIA driver extracts the VBIOS image from ROM and parses it for FWSEC microcode commands. GPU microcontrollers (MCUs) rely on Falcon microprocessors, which load their firmware from flash (VBIOS) via the kernel driver. This firmware is stored in an ELF binary, typically located at:

```
/lib/firmware/nvidia/<NV_VERSION_STRING>/gsp_ga10x.bin
```

This ELF includes custom sections such as `.fwimage`, which encapsulates the firmware image intended for the GPU MCUs.

The firmware is baked into the VM image and therefore becomes part of its overall TCB measurement. During the attestation process, the GPU signs measurement data that includes

- VBIOS contents,
- Fuses,
- Driver-loaded firmware (including the GSP ELF).

As a result, the GPU attestation report includes cryptographic digests of these components. However, this process does not provide any guarantees about the integrity of the driver itself. The report confirms that the firmware the GPU runs was measured, but it does not attest the origin or trustworthiness of the kernel driver used to load that firmware.

To ensure full trust in the driver, its integrity must be verified through CPU-side attestation mechanisms, such as IMA (Integrity Measurement Architecture), dm-verity or other root filesystem validation systems.

Without such validation, there remains a risk that a malicious or compromised driver could tamper with SPDM session setup or cryptographic operations, undermining the security guarantees of the confidential GPU environment.

In essence, while the GPU attestation confirms that it is executing known firmware, the responsibility for validating the kernel-level infrastructure that loads this firmware lies elsewhere, in the CPU attestation domain.

4.3.3 Secure Non-Volatile Storage

Secure persistent storage is a critical concern in attested Confidential VM deployments. This section explores two strategies for managing disk encryption in a way that preserves confidentiality, both during execution and after the VM lifecycle ends. These models are illustrated in Figure 12 and Figure 13.

4.3.3.1 Approach-1: External Key Broker

In this model, the decryption key is stored and managed by a trusted external Key Broker Service (KBS). During VM initialization, the guest presents a valid attestation report to the KBS, which verifies the platform's integrity before releasing the key. The guest then uses this key to mount or decrypt its persistent volume.

The main advantage of this approach is remote revocability: access to storage can be denied at any time by revoking key release. This is ideal for compliance-sensitive environments, multi-party workloads, or scenarios requiring fine-grained control over data access.

However, this model introduces a dependency on network connectivity to the KBS. If the broker becomes unreachable, the guest cannot retrieve its key, which may block workload execution. It also introduces a larger threat surface if the KBS is exposed over public or semi-trusted networks.

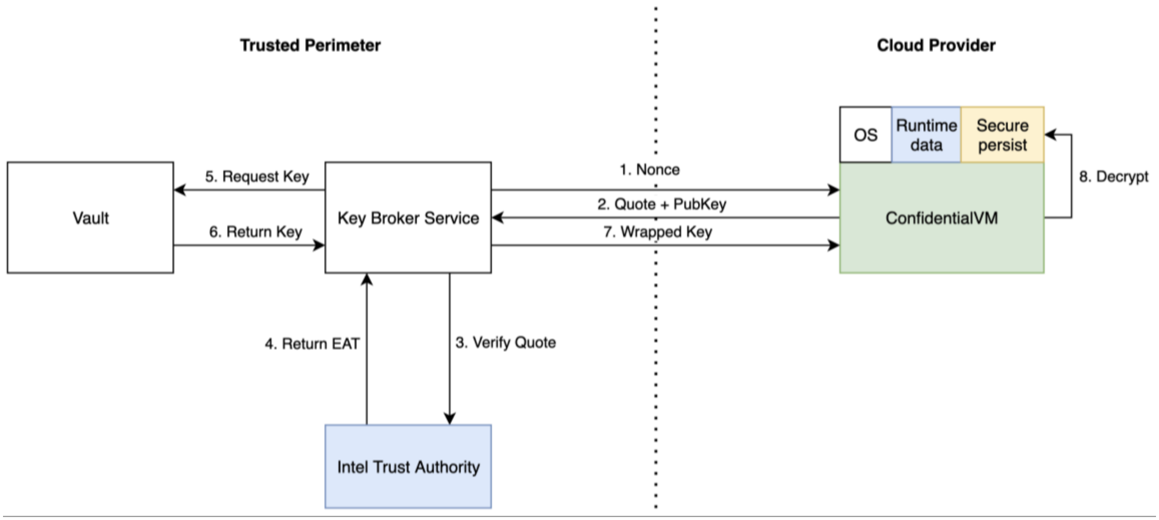

Figure 12 - Approach 1 - External Key Broker

4.3.3.2 Approach-2: In-VM Key Generation with vTPM

An alternative model involves generating the encryption key entirely within the guest, using a vTPM during provisioning. The key is derived from attested measurements (e.g., PCRs) and sealed to the guest's TCB, such that it can only be recovered by the same VM in a verified state.

This model removes the need for external communication entirely: once provisioned, the guest can decrypt its disk autonomously, assuming the environment remains consistent.

The downside is a lack of external revocation capability. Once a key is generated and sealed inside the guest, it cannot be revoked externally, even in cases of guest compromise or policy violation. This approach favors simplicity and minimal trust surfaces but sacrifices operational control.

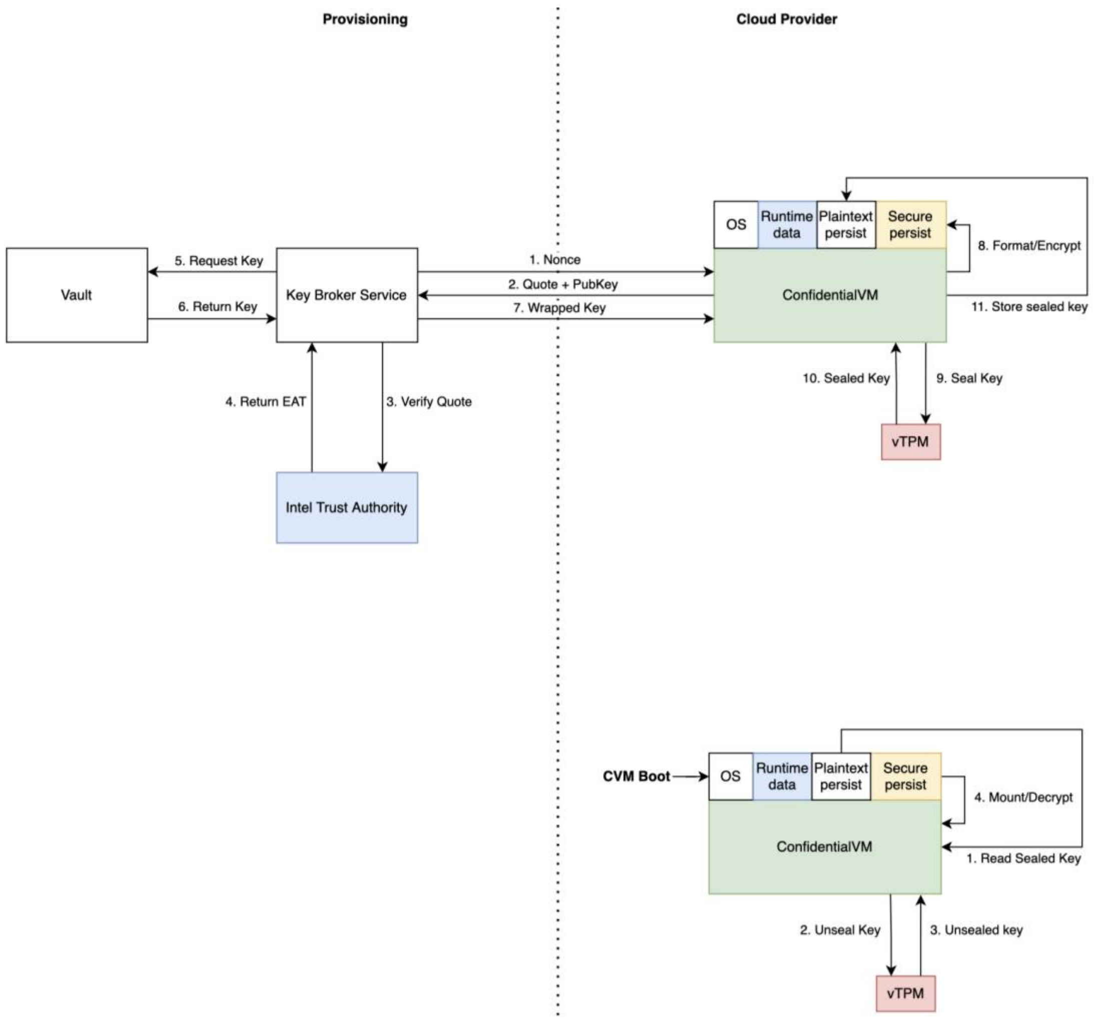

Figure 13 - Approach 2 - In-VM Key Generation with vTPM

These two models reflect a trade-off between external control and guest autonomy:

- Use KBS-based provisioning when revocation, auditing, and oversight are top priorities.
- Use vTPM-based provisioning when minimizing external dependencies is more important than runtime revocability.

In both designs, the effectiveness of storage protection hinges on strong platform attestation, ensuring keys are only available to guests in a trusted state.

5 Security Findings

This section summarises the key security observations and implementation inefficiencies identified during the evaluation. These findings reflect practical issues encountered across attestation flow, deployment setup, TCB composition, and operational tooling.

5.1 Environment and Runtime Behavior

Installing newer driver versions than those specified in the deployment guide (e.g., NVIDIA 565 vs. 550) causes instability. In this evaluation, version 565 triggered kernel panics in the guest VM, reinforcing the importance of strict driver version pinning for Confidential GPU deployments.

The `build_and_launch_docker.sh` script used to generate attestation-related firmware components reuses the `sgx.build.env` Docker image if already present. This can result in non-deterministic builds when the underlying `.nix` environment differs from earlier runs, potentially compromising build reproducibility.

5.2 CPU Attestation and TCB Transparency

Confidential VMs under Intel TDX use a Trust Domain Virtual Firmware (TDVF) as the first-stage bootloader. The MRTD register captures a measurement of this firmware. However, while Intel provides a design guide for TDVF, implementations may differ between cloud providers.

Google Cloud offers a method for verifying the TDVF binary via a signed UEFI executable, but **does not provide source code.** Tools like `gce-tcb-verifier` can attempt to reproduce the measurement, but without source transparency, trust in the firmware ultimately relies on blind acceptance or significant reverse engineering.

5.3 Attestation Integrity and Trust Model Inefficiencies

Intel's Provisioning Certification Service (PCS) can derive the PCK private key, since quote submission to PCS is not based on a traditional certificate signing request (CSR). This translates into Intel being able to re-sign a tampered attestation quote and produce a valid token without disrupting the certificate chain. While such an event would be malicious and unlikely, it illustrates the asymmetry of signing authority and its implications on third-party trust anchoring.

Furthermore, in the Intel Tiber Trust Authority (ITTA) model, the token returned by the attestation API must be manually validated against the original quote. Failing to do so shifts the root-of-trust from hardware to the attestation API service. This risk is not just theoretical: a non-malicious bug discovered during testing and reported to Intel demonstrates how operational inconsistencies can undermine attestation guarantees if not independently validated.

5.4 GPU Deployment and MIG Support

For any deployment involving the H100 GPU in a Confidential VM, it is critical to verify whether MIG (Multi-Instance GPU) support is available and enabled in the target cloud environment. In this evaluation, MIG could not be activated on the GCP test instance.

In environments where MIG is supported, each VM is expected to have its own instance of the NVIDIA driver and, by extension, a **separate key store**. This architecture reinforces per-tenant isolation at the cryptographic level and supports secure multi-tenant GPU sharing.

6 Concluding Remarks

This evaluation demonstrates the feasibility and architectural complexity of extending attestation across both CPU and GPU boundaries within Confidential VM environments. CPU attestation was performed using Intel TDX through manual quote parsing and certificate chain validation (Method-1), as awell as integration with Intel's Trusted Authority service (Method-2). GPU attestation was conducted via NVIDIA's Remote Attestation Service (NRAS), verifying device integrity and runtime state. The evaluation was carried out under realistic deployment constraints on Google Cloud Platform, highlighting practical considerations for multi-component trust in cloud-based confidential workloads.

The evaluation confirms that attestation and isolation guarantees generally operate as intended, but also highlights **underlying security inefficiencies and limitations** that surface the need for continued refinement and ecosystem alignment. These include: opaque firmware supply chains (TDVF measurement without source transparency), non-revocable secure storage strategies, inconsistent MIG enablement across cloud offerings, and the **implicit expansion of the trust boundary to vendor-managed APIs**. While these issues do not negate the security posture of the underlying technologies, they reveal areas where design assumptions and operational realities are misaligned.

As confidential computing continues to evolve, robust evaluation of this kind is essential. The observations here are intended to support platform security architects and infrastructure providers in closing critical gaps, strengthening isolation models, and advancing toward a more verifiable, zero-trust foundation for AI infrastructure.

Acronyms and Abbreviations

Acronym	Description
AE	Authenticated Encryption
CE	Copy Engine
CCA SDK	Confidential Computing Attestation Software Development Kit (NVIDIA)
CGI	Compute GPU Instance
CRL	Certificate Revocation List
CVM	Confidential Virtual Machine
D2H	Device-to-Host
DCAP	Data Center Attestation Primitives
DMAR	Direct Memory Access Remapping
DRM	Digital Rights Management (in NVIDIA's kernel driver context)
ELF	Executable and Linkable Format
FWSEC	Firmware Security Command
GCP	Google Cloud Platform
GSP	GPU System Processor (NVIDIA firmware environment)
H2D	Host-to-Device
HKDF	HMAC-based Key Derivation Function
HMAC	Hash-based Message Authentication Code
ICS	Intel Certificate Service
IMA	Integrity Measurement Architecture
KBS	Key Broker Service
LCIC	Launch Confirmation Indicator Channel
LCE	Logical Copy Engine
MIG	Multi-Instance GPU
MRSEAM	Measurement of the SEAM Module (Intel TDX)
MRTD	Measurement of the TD's Initial State (Intel TDX)
NRAS	NVIDIA Remote Attestation Service
PCR	Platform Configuration Register (TPM)
PCS	Provisioning Certification Service (Intel)
PCE	Physical Copy Engine
PSK	Pre-Shared Key
RM	Resource Manager (NVIDIA Kernel Driver Component)
RTMR	Runtime Measurement Register
SEC2	Security Microcontroller on NVIDIA GPUs
SGX	Software Guard Extensions (Intel)
SPDM	Security Protocol and Data Model
TCB	Trusted Computing Base
TDX	Trust Domain Extensions (Intel)
TDQE	TD Quoting Enclave
TPM	Trusted Platform Module
vTPM	Virtual Trusted Platform Module
VBIOS	Video BIOS (firmware interface for NVIDIA GPUs)
JWT	JSON Web Token