A Unified Framework for Attested Confidential VM Workloads in Public Clouds

(Extended Abstract of the MSc Dissertation)

JOÃO SERENO, Instituto Superior Técnico, Portugal
SUPERVISOR: NUNO SANTOS, Instituto Superior Técnico, Portugal
SUPERVISOR: LUÍS RODRIGUES, Instituto Superior Técnico, Portugal

Confidential Virtual Machines (VMs) backed by AMD Secure Encrypted Virtualization–Secure Nested Paging (SEV-SNP) protect workloads from the infrastructure operator, and remote attestation makes this protection verifiable. In practice, however, the tools needed to independently predict and verify a VM's measurements are fragmented across providers with no unified workflow. This thesis analyzes the attestation properties of confidential VM offerings from Amazon Web Services, Microsoft Azure, and Google Cloud Platform, classifies the verification depth each enables, and introduces *Evident*, a framework that assembles a single pipeline for image construction, measurement derivation, deployment, and cross-provider remote attestation. The verifying party derives expected measurements directly from the VM image artifacts with no dependency on third-party reference values and no modifications to the workload. An evaluation using a confidential inference use case confirmed that the expected predictions matched, with the remote attestation process completing in under eight seconds.

Additional Key Words and Phrases: Confidential Computing, AMD SEV-SNP, Remote Attestation, Trusted Execution Environments, Cloud Security, Virtual TPM, Reproducible Measurements

1 Introduction

Confidential virtual machines (VMs), built on hardware such as AMD Secure Encrypted Virtualization–Secure Nested Paging (SEV-SNP) [1], allow workloads to execute on public cloud infrastructure while excluding the infrastructure operator from accessing the VM's memory or execution state. Remote attestation is the protocol that makes this protection verifiable: a verifier collects cryptographically signed evidence of a VM's configuration and compares it against expected values. Without attestation, a confidential VM provides hardware-level protection but no way for an external party to confirm that the protection is in place or that the intended software was loaded.

In practice, however, attestation for confidential VMs on public cloud infrastructure remains largely unsolved for general-purpose workloads. The three major providers that offer AMD SEV-SNP (Amazon Web Services, Microsoft Azure, and Google Cloud Platform) each expose attestation evidence differently, with varying degrees of transparency about what is included and how [10, 15]. The AMD Secure Processor measures the VM's initial memory contents at launch, covering the firmware, but does not measure subsequent boot stages. For the following measurements, a Trusted Platform Module (TPM) must be used, but, on all three major providers, its virtualized implementation remains proprietary, reintroducing a provider-managed component into the trusted computing base. Existing frameworks such as *Keylime* [14] and *Constellation* [9] focus on the remote attestation protocol itself but leave a key question unanswered: how can measurements be independently derived from the source artifacts of an arbitrary workload? No existing framework offers a unified workflow spanning the full confidential VM lifecycle. Verifiers must either trust a third party to supply reference measurements or accept first-observed values as a baseline. This thesis presents *Evident*, a framework that closes this gap by enabling the verifying party to independently predict and verify the measurements produced by a confidential VM, without requiring modifications to the deployed workload.

The thesis makes two primary contributions:

(1) a systematic comparison of AMD SEV-SNP confidential VM support across Elastic Compute Cloud, Azure Virtual Machines, and Google Compute Engine, accompanied by a software verification depth taxonomy.
(2) a lifecycle management pipeline design covering image construction, measurement derivation, upload, deployment, and cross-provider remote attestation, where the verifying party derives expected measurements directly from the VM image artifacts.

The *Evident* framework implements this pipeline for Elastic Compute Cloud and Google Compute Engine, with extensions including signed image distribution, a *Terraform* provider plug-in for deployment-integrated attestation, and a composable Confidential Certificate Authority for scalable attestation. Support for deriving Google Compute Engine launch digests was contributed to the open-source *sev-snp-measure* project.

2 Background

Cloud computing requires surrendering control over the infrastructure that processes data. An increasingly common answer to this tension is to anchor trust in hardware. A hardware root of trust concentrates the assumptions on the correctness of the silicon and its firmware, the integrity of the manufacturing process, and the manufacturer's key management. These assumptions are narrower and more stable than the ones they replace, and some can be tested through cryptographic attestation rather than accepted on faith.

2.1 Trusted Execution Environments

A Trusted Execution Environment (TEE) is a hardware-enforced isolated region of a processor in which code and data are protected against access or modification by any software running outside it. Its core capabilities are confidentiality and integrity: the TEE's

Authors' Contact Information: João Sereno, joaohsereno@tecnico.ulisboa.pt, Instituto Superior Técnico, Lisbon, Portugal; Supervisor: Nuno Santos, nuno.m.santos@tecnico.ulisboa.pt, Instituto Superior Técnico, Lisbon, Portugal; Supervisor: Luís Rodrigues, ler@tecnico.ulisboa.pt, Instituto Superior Técnico, Lisbon, Portugal.

memory is encrypted against external access, and unexpected modifications are detected in hardware. Critically for this thesis, a TEE also supports attestation, where the hardware generates a signed statement describing its configuration and the software loaded into it, allowing a remote party to verify the environment without trusting additional components. TEE implementations differ in scope: Intel *Software Guard Extensions* isolates at the process level [7], Arm TrustZone partitions into secure and normal worlds [13], and both Intel *Trusted Domain Extensions* and AMD SEV-SNP operate at the virtual machine level [1, 6, 11]. This thesis focuses on AMD SEV-SNP, designed for the cloud context with a threat model that explicitly includes a potentially compromised hypervisor.

AMD SEV-SNP delegates memory security to a dedicated security coprocessor (the AMD *Secure Processor*) embedded in the CPU die [1]. Each confidential VM is assigned a unique encryption key unknown to the hypervisor, ensuring memory pages are encrypted at rest and decrypted only within the correct CPU context. SEV-SNP also introduces virtual machine privilege levels (VMPLs), allowing the memory of a single confidential VM to be partitioned into up to four hierarchical tiers. The *Secure Processor* can generate a signed attestation report committing to the VM's configuration and a measurement of the software loaded at launch, signed by a key whose authenticity traces back to AMD through a certificate chain. The *Secure Processor* may use the Versioned Chip Endorsement Key (VCEK) or Versioned Loaded Endorsement Key (VLEK), depending on the set configuration.

2.2 Trusted Platform Module

The *Secure Processor*'s attestation report covers the initial memory contents at launch, typically the VM firmware. Components loaded after the firmware (bootloader, kernel, filesystem) fall outside the *Secure Processor*'s measurement. The Trusted Platform Module (TPM) addresses this gap through Platform Configuration Registers (PCRs), hardware-protected registers updated only through a one-way extend operation: the current value is concatenated with a new measurement and replaced with the hash of the combination [4, 16]. The resulting value is a cumulative digest of every measurement in chronological order and cannot be forged without knowing the exact sequence. In cloud environments, physical TPMs are replaced by virtual TPMs emulating the interface in software, with trust anchors that vary by implementation.

2.3 Boot process

Trust in a boot sequence rests on measuring a component before executing it. The confidential VM boot unfolds in two phases. In the pre-boot phase, the *Secure Processor* measures every memory page loaded into the VM before any instruction executes inside it, producing a launch digest that accumulates a cryptographic record of each page's contents and metadata [2]. In the boot phase, once the VM is running, successive components measure the next and extend those measurements into the virtual TPM. The two phases form a contiguous chain: the *Secure Processor* covers what no software component can reliably cover, and the TPM is intended to carry the chain forward.

2.4 Remote attestation and verification depth

Remote attestation follows the *Remote Attestation Procedures* architecture published by the *Internet Engineering Task Force* [5], with three entities: the *Attester* that generates evidence, the *Verifier* that evaluates the evidence, and the *Relying Party* that consumes the results. Verification requires checking the certificate chain of the *Secure Processor*'s report and virtual TPM's quote, validating freshness with a nonce, and comparing measurements against expected values.

This thesis proposes a software verification depth taxonomy with four levels, inspired by G. Scopelliti *et al.* [15]. L1 (Root of Trust link authenticity) validates cryptographic signatures and certificate chains not yet requiring semantic interpretation of measurements. L2 (Reference-based measurement matching) compares against known reference values or past measurement collections. L3 (Binary measurement reproducibility) involves the verifier possessing the actual binary artifacts and deriving their measurements locally. L4 (Source code measurement reproducibility), the verifier gets the source code and build instructions that allow them to independently construct the binary artifacts from which the measurements are derived.

3 Related Work

3.1 Public cloud provider comparison

All three providers offer AMD SEV-SNP-based confidential VMs, but diverge on firmware measurement transparency, as illustrated in Figure 1. Elastic Compute Cloud publishes both the firmware source code and the measurement procedure, making the launch measurement reproducible (L4). Google Compute Engine provides the endorsed firmware binary through public interfaces (L3) but does not disclose the source. Azure Virtual Machines includes a proprietary Host Compatibility Layer (HCL) component in the launch measurement; neither the HCL binary nor the measurement methodology is available, and the guest can only fetch a single attestation report generated at launch without verifier-controlled freshness. On all three providers, the virtual TPM is closed-source and cannot be substituted by the customer.

When defining the threat model, the relying party needs to decide if they accept the AMD *Secure Processor* as a trustworthy implementation of the AMD SEV-SNP behavior specification and if they accept the virtualized TPM implementation of each cloud provider as a trustworthy implementation of the TPM specification. Trusting the AMD *Secure Processor* gives the relying party assurance on the confidential VM's fundamental confidentiality and integrity protection mechanisms, as well as integrity assurance regarding the VM's firmware. Measurements from later boot components are only useful if the relying party trusts the virtual TPM implementation. The verification depth of any component is bounded by its predecessor's: for instance, a compromised firmware could measure a correct bootloader but load a different one.

The providers also differ in attestation evidence exposure. Elastic Compute Cloud uses VLEK (rather than chip-specific VCEK) for report signing, hiding the physical chip identity but with no security implications since both are rooted in AMD Root Key (ARK). Azure blocks direct access to the *Secure Processor* device interface and the

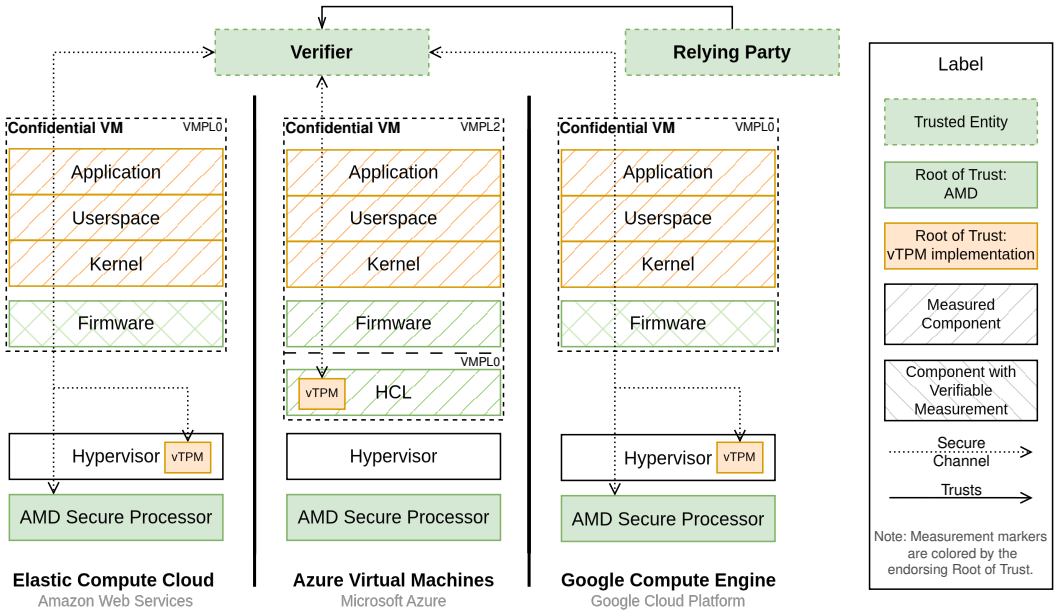

Fig. 1. Component measurement chain at public cloud providers

single attestation report is stored in the HCL, weakening freshness because the verifier cannot request a new report with a verifier-chosen nonce. This makes the verifying party unable to confirm that the report originates from the target VM and makes Azure the only provider where the verifier cannot conclude that memory confidentiality and integrity protections are in place.

3.2 On-premises solutions

Several proposals extend the trust chain without requiring the relying party to trust a provider-managed virtual TPM. SNPGuard [17] enlarges the set of components measured by the *Secure Processor*: the hypervisor hardcodes kernel and initrd measurements into the firmware binary, which the firmware verifies before transferring control, covering later boot components in a single hardware root of trust. Decentriq used micro confidential VMs [8] (based on Firecracker [3]), where the *Secure Processor* loads and measures a complete package containing the firmware, the bootloader, the kernel, and kernel command-line arguments, making the launch digest the only required measurement to verify. AMD's Secure VM Service Module (SVSM) specification standardizes the execution of privileged services, including a virtual TPM, inside the highest Virtual Machine Privilege Level (VMPL), isolated from both the hypervisor and the guest operating system. V. Narayanan *et al.* [12] built a complete ephemeral virtual TPM implementation following this specification, effectively linking the TPM's trust anchor to the *Secure Processor*. All of these require firmware or hypervisor configuration that public cloud customers cannot replicate.

3.3 Existing frameworks

Keylime [14] is an open-source framework that automates remote attestation using TPM devices. It provides a production-ready pipeline but does not recognize AMD *Secure Processor* attestation reports, so any assurance rests entirely on the virtual TPM. Reference values are an external input whose provenance is the operator's responsibility. *Constellation* [9] developed by Edgeless Systems, deployed *Kubernetes* clusters inside confidential VMs and integrated attestation into the Transport Layer Security (TLS) handshake. It did not independently reproduce *Secure Processor* launch measurements; its expected measurements were embedded in release artifacts, requiring trust in the platform vendor. *Trustee*[1], from the *Confidential Containers* project, also separates attestation verification from reference value provisioning and pushes the problem of deriving those values entirely to the operator.

Contrast[2] (*Constellation*'s successor) and *Confer*[3] represent strong approaches, as they leverage bare-metal instances in public cloud providers that allow them to deploy and configure their own hypervisor of choice, with the possibility to overcome the limitations observed in Subsection 3.1. *Contrast* targets *Kubernetes* clusters deployments, and uses an approach similar to the V. Narayanan *et al.* [12] proposal. *Confer* applies confidential VMs to private artificial intelligence inference on Intel *Trusted Domain Extensions*, achieving L4 for core system components, but operates as a single-application service rather than a general-purpose framework. Neither addresses the generic deployment and verification problem in public cloud infrastructure without managing its own bare-metal instance.

[1] https://github.com/confidential-containers/trustee
[2] https://www.edgeless.systems/products/contrast
[3] https://github.com/conferlabs/confer-image

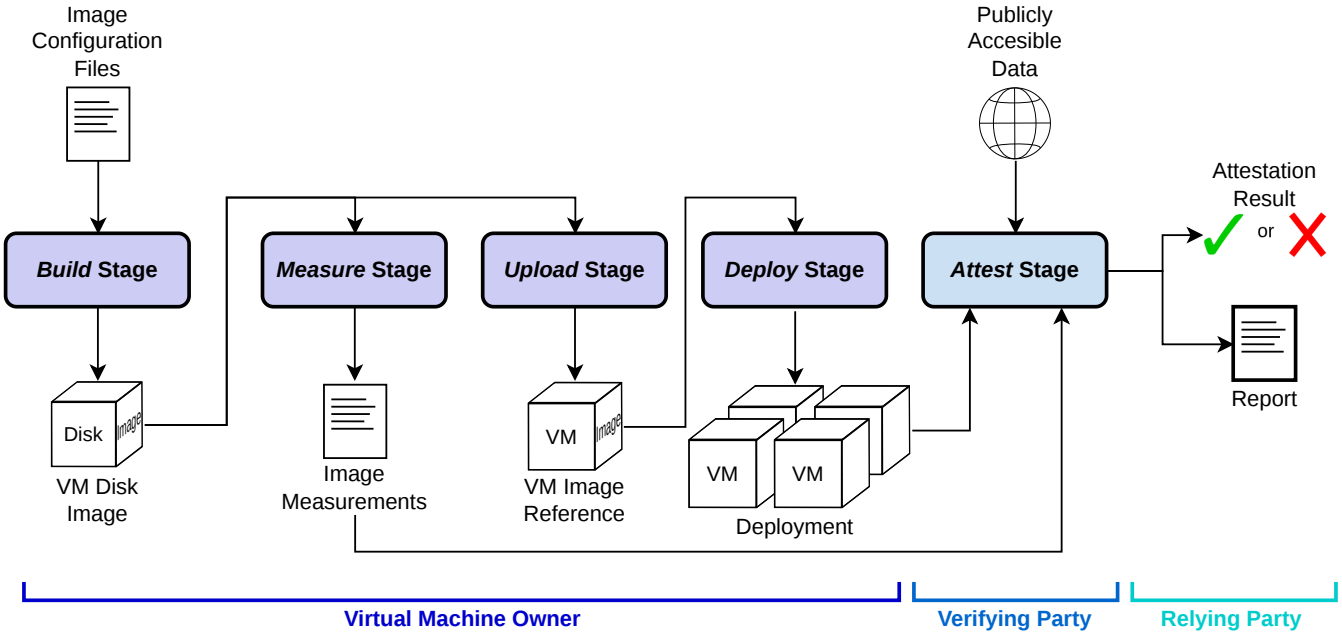

Fig. 2. *Evident* confidential VM lifecycle management stages

4 *Evident* Framework

Evident automates the trust-enabling operations surrounding confidential VM lifecycle management. Its purpose is to make confidential VMs a verifiable generic platform for any given workload, without requiring modifications to the workload itself. The design approach is fundamentally different from the studied alternatives. In *Evident*, the verifying party derives its own expected measurements from the relevant artifacts. In contrast, all surveyed systems design the verifier as a consumer of third-party-derived expected measurements. The framework is workload-agnostic and cloud-agnostic at the architecture level, using custom-built VM images as its primitive. The implementation scope includes AMD SEV-SNP as the hardware module and provider modules for Elastic Compute Cloud and Google Compute Engine.

4.1 Threat model

The hypervisor may be adversarial. Co-located VMs may attempt side-channel or direct memory access. The cloud provider is not assumed benign. The network is assumed susceptible to attacks on both confidentiality and integrity. The threat model accepts AMD SEV-SNP as a trustworthy implementation of their advertised specification and accepts the provider's virtual TPM as a correct realization of the TPM 2.0 specification, resistant to integrity and confidentiality attacks by any actor, including the provider. This is a bounded assumption about a specific component, rather than trusting the cloud provider as a whole. If the *Secure Processor* assumption is removed, no attestation report is meaningful. If only the virtual TPM assumption is removed, conclusions are limited to firmware integrity. The framework collects all available evidence unconditionally; how that evidence is interpreted is the decision of the relying party.

4.2 Challenges

Reproducing the *Secure Processor*'s launch digest requires knowing both the measured contents and the exact page-loading sequence, which varies across providers. Reproducing virtual TPM PCR values requires knowing which components were measured, in what order, and into which register. The framework addresses this by constraining the build environment to produce deterministic images whose measurements can be predicted. Filesystem immutability facilitates verification of filesystems and applications. If the root filesystem were writeable, a malicious provider could inject files into the image before launch, and such modifications might not be captured by the virtual TPM's boot measurements.

4.3 Architecture

The framework addresses five lifecycle stages, illustrated in Figure 2: *build* packages the workload inside a VM image; *measure* analyzes the image and derives the expected measurements; *upload* prepares the image for instantiation on a supported cloud provider; *deploy* instantiates confidential VMs across one or more providers; and *attest* verifies the actual measurements against the expected ones.

The framework has two main components, as shown in Figure 3. The *Evident* Server runs inside the confidential VM, where it collects evidence from the AMD *Secure Processor* and the virtual TPM on behalf of the verifier. The *Evident* Client runs in the interested party's environment and orchestrates all lifecycle stages. As pictured in 2, the VM Owner uses the Client to build, measure, upload, and deploy the confidential VMs. Once launched, the Server listens

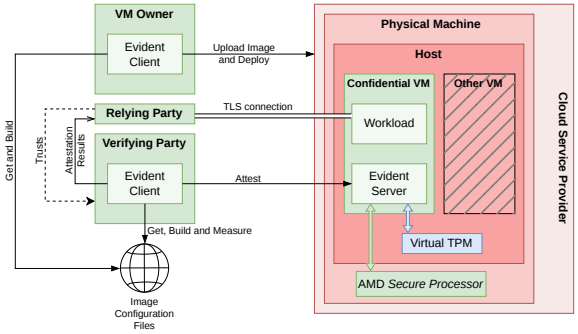

Fig. 3. *Evident* Framework context view

for remote attestation requetss and, upon receiving one, gathers evidence from both trusted components. The Verifier uses the Client to first build and measure the expected VM image independently, then performs remote attestation against the target confidential VM and reports the observed state. The Relying Party relies on the Verifier's conclusions to decide whether to interact with the deployed workload based on its own trust model.

Each stage has well-defined inputs and outputs that are meaningful even outside the framework's context, making each stage's implementation replaceable, provided it supplies the expected inputs to dependent stages. The Client and Server communicate via the *gRPC* protocol, defined independently of either component's implementation. This protocol is the contract both sides must conform to; a user who does not trust the provided implementation can reimplement either side in accordance with the protocol specification without loss of interoperability. The Server prioritizes vulnerability protection and minimal resource usage, since it runs alongside potentially sensitive workloads. The Client is responsible for the critical attestation logic, where source code readability and auditability were prioritized over all other concerns, because the Client's correctness must be assessable by the relying party.

The framework operates exclusively on the VM image and its configuration; it does not instrument, intercept, or depend on the behavior of the included workload. Users take the VM image configuration as a minimal base onto which additions, specifically the workload, can be merged. Whenever possible, this thesis implementation reuses functionality from other established tools and libraries, reducing the amount of additional software that must be trusted.

4.4 Extensions

Several optional features were designed on top of the core pipeline without altering it.

An *image distribution* mechanism allows a distributor to sign a manifest containing build instructions, expected measurements, and the resulting image digest, enabling trusting consumers to skip the build stage or reproduce the build independently for verification. If a trust model does not accept the distributor, the manifest contains enough information to replicate the build from source and compare the results.

Attestation certificates bind an instance's cryptographic identity to its measured state. The VM generates an asymmetric key pair (instance keys) upon launch and includes the public key in the user data field of both the *Secure Processor*'s attestation report request and the TPM's quote request, so that it becomes covered by the root-of-trust signing key's signature. A VM that presents both the expected measurements and the identification of that key pair lets the verifying party conclude that the instance keys belong to the attested VM and that their confidentiality and integrity are protected by the confidential VM mechanisms. Upon successful attestation, the Verifier signs a certificate for the instance, enabling a secure communication channel anchored in the attestation result. A relying party that trusts the Verifier can then verify that subsequent connections reach the attested VM by checking the certificate chain.

A *Confidential Certificate Authority* (Confidential CA) design addresses the scalability limitation of requiring the Verifier to individually attest every instance. A Confidential CA is itself a confidential VM whose workload is the *Evident* Client running in a dedicated serving mode. It listens for incoming attestation requests from other confidential VMs, performs the full remote attestation protocol on each, and issues certificates upon success. The Confidential CA is loaded at build time with a set of trusted package submitter identities, each identified by a public key, and accepts new attestable image measurements dynamically through signed package submissions validated against those keys. As illustrated in Figure 4, the design is composable, *i.e.*, a Confidential CA can itself obtain a certificate from a higher-level Confidential CA, forming a hierarchy where each certificate in the chain was issued by an entity that was itself attested before it began issuing. Users may attest the root Confidential CA using the standard *Evident* Client workflow. A browser or other client that trusts the root Confidential CA's certificate can then interact with any leaf confidential VM that presents a certificate rooted to it, without performing the full attestation protocol independently. All key material and certificates exist only in volatile memory; a restart destroys them and forces re-attestation.

A *Terraform* provider plug-in integrates attestation into the deployment workflow, treating the attestation result as a Terraform resource with dependency declarations on VM resources, polling instances until they are ready, and performing attestation as part of a single deployment operation.

4.5 Discussion

The framework cannot resolve the closed-source nature of all three providers' virtual TPM implementations. On Elastic Compute Cloud, the absence of a standard TPM endorsement key certificate limits third-party attestation. On Azure Virtual Machines, direct access to the *Secure Processor* device interface is blocked, and the single pre-generated attestation report lacks freshness guarantees. Disk encryption and runtime integrity verification are deferred to future work. A relying party that does not fully accept the threat model may still find the collected evidence useful, since a deployment in which all TPM-reported measurements align with expected values is at minimum consistent with a correctly booted system, and a mismatch signals a problem regardless of whether the TPM is fully trusted. The evidence may also serve third parties with different threat models, or become meaningful if a provider later publishes independently verifiable proof of its virtual TPM operations.

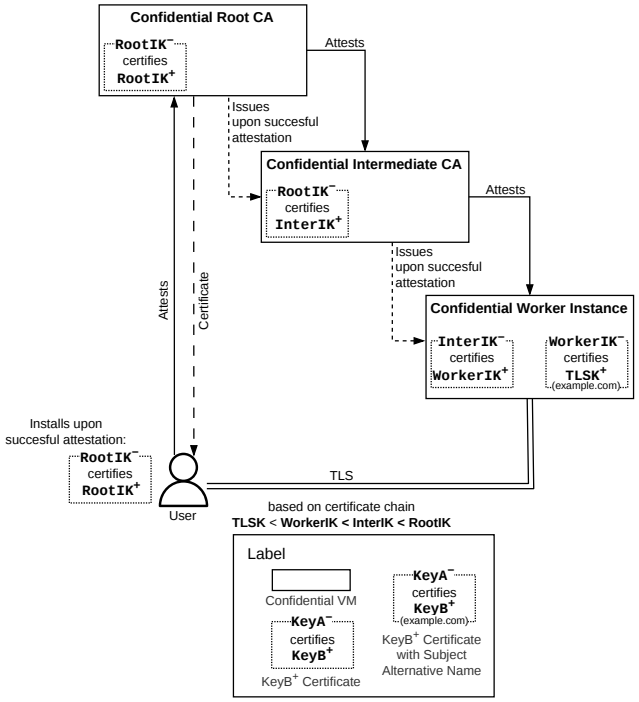

Fig. 4. Hierarchical certificate issuance with Confidential Certificate Authorities

5 Implementation

This section documents a concrete implementation of the design presented in the previous section, covering the engineering constraints that shaped technology choices and the practical obstacles resolved before the framework could produce meaningful attestation results on real infrastructure. The implementation supports AMD SEV-SNP and targets Elastic Compute Cloud and Google Compute Engine.

5.1 Technology stack

The Client is implemented in *Go*, chosen for its readability and auditability: explicit error handling, a minimal language surface, and static compilation that produces a single binary with no runtime dependencies. The Server is implemented in *Rust*, chosen for memory safety without garbage collection and minimal runtime overhead. Both communicate over *gRPC*, with Protocol Buffers providing the language-neutral service definition that serves as the formal contract between them. The transport layer uses TLS with a self-signed certificate generated by the Server at launch; this certificate is ephemeral and provides only transport encryption, with identity authentication deferred to the attestation process itself.

5.2 VM image construction

The implementation uses *NixOS*, a Linux distribution where the entire operating system configuration is specified declaratively. *Nix Flakes* pins every input (package repositories, external dependencies, toolchains) to exact revisions, making reproducibility achievable, *i.e.*, given the same flake and the same inputs, the build process should produce an identical output. Every package and its dependencies are addressed by a cryptographic hash of their inputs, making it straightforward to trace exactly what is included in the final image. The framework provides a *NixOS* configuration as a template that users adapt by adding their workload, enabling services, and adjusting system parameters. The image configuration declares a Unified Kernel Image (UKI), a single portable executable binary that bundles the Linux kernel, initrd (initialization process), and the kernel command-line arguments into one file. With direct kernel boot, the firmware loads and measures this single binary, eliminating the ambiguity caused by separately substitutable components.

The root filesystem uses *erofs*, a read-only compressed filesystem, with a *dm-verity* hash tree computed at build time over the entire partition. The root hash of this tree is embedded in the kernel command-line, which is itself part of the UKI. When the kernel mounts the root filesystem, *dm-verity* verifies each block read against this hash tree. Any modification to the filesystem contents, whether made before or after deployment, produces a hash mismatch and causes the read to fail. The integrity of the entire filesystem is thus reduced to the integrity of the root hash, and the root hash is measured as part of the UKI. Directories requiring write access (`/tmp`, `/var`, and others) are mounted as volatile *tmpfs* whose contents exist only in RAM, encrypted by AMD SEV-SNP. The `/nix/store` directory containing all binaries and libraries is part of the read-only partition; it is not possible to install system-wide applications that were not packaged with the VM image at build time.

5.3 Measurement derivation

For the AMD *Secure Processor* launch digest, the framework uses the *sev-snp-measure-go* library, which simulates the hypervisor's page-loading sequence. For Elastic Compute Cloud, Amazon publishes the firmware source code and measurement procedures. For Google Compute Engine, support was added to the same library as part of this thesis work, making both providers addressable through a common interface. Google publishes the firmware binary through public interfaces, but not the source code. The endorsed firmware binary is obtained through a multi-step process: a launch endorsement blob is fetched from a public Google Cloud Storage bucket indexed by its launch digest value, its signature is verified against Google's cloud integrity root certificate, and the binary itself is then downloaded from a separate location. The launch digest also depends on the number of virtual cores assigned to the confidential VM, since each virtual core needs an initial state configuration page measured by the *Secure Processor*. The current implementation derives the launch digest at attest time, when the instance configuration is known.

For virtual TPM PCR prediction, the framework predicts three registers. PCR 4 records the measurement of the UKI loaded by firmware. PCR 11 records individual sections of the UKI (kernel, initrd, kernel command-line containing the *dm-verity* root hash) as measured by the *systemd-stub*, which is the first component to take control within the UKI. PCR 12 records UKI extensions; since *Evident* images do not use extensions, it is expected to be zero, and the framework hardcodes this expectation as a verification that no unexpected extensions were applied. The UKI is extracted from the raw VM disk image using the *godiskfs* library, avoiding

root permissions, and *systemd-pcrlock* derives the expected PCR values by parsing the UKI's portable executable binary structure and computing the measurement that the firmware would extend into the corresponding PCR for each section.

5.4 Remote attestation workflow

The *attest* stage follows a three-phase pipeline uniform across providers, with provider-specific behavior confined to well-defined points within each phase.

In the evidence collection phase, the Client generates a 64-byte cryptographically random nonce and sends it to the *Evident* Server. The Server uses the nonce to request attestation evidence from both the AMD *Secure Processor* and the virtual TPM, then returns a bundle containing the attestation report, the TPM quote, and the instance key. The AMD processor model is inferred from the collected evidence, which determines the certificate chain used in subsequent verification.

In the hardware evidence verification phase, the AMD trust anchor certificates (ARK, ASK, ASVK) are fetched from AMD's key distribution service. The attestation report's signature is certified against the evidence-included VCEK or VLEK, and the full certificate chain is verified. The Client then recomputes a digest over the nonce, the serialized instance public key, and provider-specific additional material, comparing it against the data field in the report to check both freshness and key binding. A mismatch indicates that the report was generated at a different time or for a different instance key. The firmware binary is obtained through provider-specific means, and the expected launch digest is computed by simulating the page-loading sequence; a mismatch means the firmware loaded into the confidential VM at boot differs from the expected binary.

In the software evidence verification phase, the TPM quote signature is verified through a provider-specific attestation key (AK) trust mechanism. On Google Compute Engine, the AK certificate is retrieved from Google's key distribution service and verified via a chain to a public root certificate authority. On Elastic Compute Cloud, no AK certificate exists; instead, the TPM make-credential/activate-credential protocol establishes that the AK and the Endorsement Key (EK) reside in the same TPM. If the instance identifier is supplied, a provider-specific access-controlled interface returns a provider-endorsed EK for comparison. This interface is not publicly available, which limits independent third-party attestation on Elastic Compute Cloud. After AK trust is established, the quote's freshness binding is checked, and the PCR digest embedded in the quote is compared against the expected value derived from the *measure* stage. A mismatch means the image contents differ from the previously measured ones.

On Google Compute Engine, the AK certificate is retrieved from Google's key distribution service and verified via a chain of trust to a Google-managed public root certificate authority. The endorsed firmware binary is obtained through a multi-step process: a launch endorsement blob is fetched from a public storage bucket indexed by the launch digest from the *Secure Processor* report, its signature is verified against Google's cloud integrity root certificate,

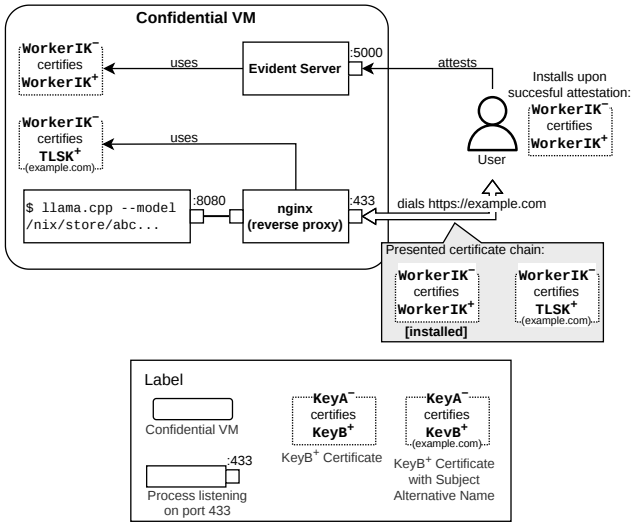

Fig. 5. Confidential inference components

and the firmware binary is then downloaded from a separate location. On Elastic Compute Cloud, the TPM make-credential/activate-credential protocol establishes that the AK and Endorsement Key (EK) reside in the same TPM. If the instance identifier is supplied, a provider-specific interface returns a provider-endorsed EK for comparison against the one retrieved directly from the instance. The interface is access-controlled, which limits independent third-party attestation. The endorsed firmware binary is retrieved directly from Amazon's GitHub repository releases.

5.5 Extensions

Image distribution produces a GPG-signed package containing the VM image, a *Nix* derivation file that captures a complete view of all build inputs, expected measurements, and a manifest with metadata and digests. The *reproduce* execution mode interprets the included manifest, rebuilds the image from source, and compares the results against the endorsed package. If both images match, the package is re-endorsed by the reproducing party; a package may carry multiple endorsements from different parties.

The Confidential Certificate Authority runs the *Evident* Client in *serve-certify* mode. When a confidential VM requests a certificate, the Confidential CA performs the full remote attestation procedure against the requesting instance and compares the observed measurements against all previously accepted measurement sets from signed package submissions. A match constitutes a successful attestation, and the requesting instance receives a certificate. The set of attestable images grows dynamically without redeploying the Confidential CA, while the trust boundary remains well-defined: only packages endorsed by a submitter whose key was trusted at build time can influence which images the authority will accept.

6 Evaluation

6.1 Confidential inference use case

The framework was validated end-to-end through a confidential inference service, illustrated in Figure 5: *llama.cpp* serving Meta's Llama 3.1 8B model inside a confidential VM. The model weights are included in the image at build time, become part of the read-only *erofs* partition, and are covered by the *dm-verity* hash tree, making the model's integrity transitively captured by the UKI measurement and verifiable through the TPM's PCR. The use case was completed on both Elastic Compute Cloud (`m6a.8xlarge` in `eu-west-1`) and Google Compute Engine (`n2d-standard-16` in `europe-west-3`). Upon successful attestation, the client retrieves and installs the instance's self-signed certificate, whose public key was bound to the attested measurements, and subsequent HTTPS connections use TLS anchored in the instance's certificate; therefore, the service is accessible without browser warnings only after installing the instance's self-signed certificate. The same use case was also validated with the hierarchical Confidential Certificate Authority configuration, where the user only needed to attest the root Confidential CA and install its certificate; all subsequent certificate issuance to leaf instances was automated.

6.2 Measurement correctness

All predicted measurements matched the corresponding actual values on both providers, except Elastic Compute Cloud's launch digest, due to a recently reported apparent firmware versioning issue. The retrieved PCR digest (that reflects PCRs 4, 11, and 12) matches the expected one, confirming the UKI and filesystem integrity. The framework claims that, given correct inputs, the derivation is deterministic and accurate; the results for the tested configurations support this claim.

6.3 Remote attestation report

The *Evident* Client produces a structured report documenting every verification step, including what was verified, observed and expected values, and the security guarantee obtained if it passes. A successful attestation means all checks passed; a failed one identifies exactly which verification did not match, displays both values, and explains what assurance is lost. The remaining passing checks are still reported and may be meaningful under different trust models.

6.4 Latency breakdown

Non-network-dependent tasks (certificate chain verification, digest comparison, measurement derivation) have near-instant execution times. The dominant cost is network communication. The end-to-end remote attestation time is approximately 1.8 seconds on Elastic Compute Cloud and 4.7 seconds on Google Compute Engine (medians over 100 runs, including network delays). The difference is dominated by AMD *Secure Processor* report generation latency: retrieving the report is approximately 30–50 times slower on Google Compute Engine than on Elastic Compute Cloud. The cause could not be identified. In contrast, retrieving the virtual TPM quote is approximately twice as fast on Google Compute Engine.

6.5 Resource consumption

To better observe *Evident* Server resource consumption, the resource usage was monitored under sustained concurrent load (deliberately unrealistic conditions). The *Evident* Server uses approximately 15 MB of RAM regardless of concurrency level. On Elastic Compute Cloud, CPU usage rises approximately 40% of a single core under high concurrency, with throughput reaching approximately 20 requests per second. On Google Compute Engine, CPU usage remains near 0% regardless of concurrency (the server spends nearly all time blocked on the *Secure Processor* device interface), with throughput plateauing at approximately 0.3 requests per second. In realistic deployments where attestation requests are infrequent, the server's resource consumption is negligible.

7 Conclusions

This thesis analyzed AMD SEV-SNP confidential VM offerings across three major cloud providers and confirmed substantial differences in attestation evidence transparency, with Google Compute Engine currently achieving the highest practical verification depth. The *Evident* framework was designed and implemented to close the resulting gap: the verifying party derives expected measurements directly from the VM image artifacts rather than consuming reference values from a third party. The pipeline spans the full confidential VM lifecycle, requires no workload modifications, and supports cross-provider attestation on Elastic Compute Cloud and Google Compute Engine. The evaluation confirmed measurement correctness, demonstrated end-to-end functionality through a confidential inference use case, and showed that the framework introduces negligible overhead (approximately 15 MB of RAM, end-to-end attestation under 8 seconds).

Several directions remain open: adding disk encryption for workloads that require persistent, confidential storage, broadening provider coverage to include Azure Virtual Machines, monitoring runtime integrity to detect post-boot vulnerabilities, and strengthening packaging tooling to enable automated reproduction of image builds from the included manifest.

Acknowledgments

This work was developed within the scope of: proj. no.62–"Responsible AI", financed by European Funds, namely "Recovery and Resilience Plan"–Component 5: "Agendas Mobilizadoras para a Inovação Empresarial", included in the NextGenerationEU funding program; the EU's Horizon Europe research and innovation programme under Grant Agreement No 101189689; FCT under grants UID/50021/2025 and UID/PRR/50021/2025; and, IAPMEI under grant C6632206063-00466847 (PT Smart Retail). I would also like to thank Daniel Castro for his invaluable help and guidance.

References

[1] Advanced Micro Devices, Inc. 2020. *AMD SEV-SNP: Strengthening VM isolation with integrity protection and more.* Technical Report. Advanced Micro Devices, Inc. https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf Retrieved on March 24th, 2025.

[2] Advanced Micro Devices, Inc. 2025. *SEV Secure Nested Paging Firmware ABI specification.* Advanced Micro Devices, Inc. https://www.amd.com/content/

dam/amd/en/documents/epyc-technical-docs/specifications/56860.pdf Official specification. Revision 1.58. Retrieved on March 24th, 2025.

[3] Alexandru Agache, Marc Brooker, Alexandra Iordache, Anthony Liguori, Rolf Neugebauer, Phil Piwonka, and Diana-Maria Popa. 2020. Firecracker: Lightweight Virtualization for Serverless Applications. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA, 419–434. https://www.usenix.org/conference/nsdi20/presentation/agache

[4] Will Arthur, David Challener, and Kenneth Goldman. 2015. *A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security* (1 ed.). Apress, USA. https://doi.org/10.1007/978-1-4302-6584-9

[5] Henk Birkholz, Ned Smith, Dave Thaler, Laurence Lundblade, and Mohit Sethi. 2023. Remote attestation procedures architecture. https://datatracker.ietf.org/doc/rfc9334/ RFC 9334. Accessed on May 10th, 2025.

[6] Pau-Chen Cheng, Wojciech Ozga, Enriquillo Valdez, Salman Ahmed, Zhongshu Gu, Hani Jamjoom, Hubertus Franke, and James Bottomley. 2023. Intel TDX Demystified: A Top-Down Approach. arXiv:2303.15540 [cs.CR] https://arxiv.org/abs/2303.15540

[7] Victor Costan and Srinivas Devadas. 2016. Intel SGX Explained. Cryptology ePrint Archive, Paper 2016/086. https://eprint.iacr.org/2016/086

[8] Decentriq. 2022. Swiss cheese to cheddar: securing AMD SEV-SNP early boot. https://www.decentriq.com/article/swiss-cheese-to-cheddar-securing-amd-sev-snp-early-boot Accessed on March 27th, 2025.

[9] Edgeless Systems. [n. d.]. Constellation - The Kubernetes Built for Confidential Computing. https://www.edgeless.systems/products/constellation Accessed on May 9th, 2025.

[10] Jana Eisoldt, Anna Galanou, Andrey Ruzhanskiy, Nils Küchenmeister, Yewgenij Baburkin, Tianxiang Dai, Ivan Gudymenko, Stefan Köpsell, and Rüdiger Kapitza. 2025. SoK: A cloudy view on trust relationships of CVMs – How Confidential Virtual Machines are falling short in Public Cloud. arXiv:2503.08256 [cs.CR] https://arxiv.org/abs/2503.08256

[11] Masanori Misono, Dimitrios Stavrakakis, Nuno Santos, and Pramod Bhatotia. 2024. Confidential VMs Explained: An Empirical Analysis of AMD SEV-SNP and Intel TDX. *Proc. ACM Meas. Anal. Comput. Syst.* 8, 3 (Dec. 2024). doi:10.1145/3700418

[12] Vikram Narayanan, Claudio Carvalho, Angelo Ruocco, Gheorghe Almasi, James Bottomley, Mengmei Ye, Tobin Feldman-Fitzthum, Daniele Buono, Hubertus Franke, and Anton Burtsev. 2023. Remote attestation of confidential VMs using ephemeral vTPMs. In *Proceedings of the 39th Annual Computer Security Applications Conference* (Austin, TX, USA) *(ACSAC '23)*. Association for Computing Machinery, New York, NY, USA, 732–743. doi:10.1145/3627106.3627112

[13] Sandro Pinto and Nuno Santos. 2019. Demystifying Arm TrustZone: A Comprehensive Survey. *ACM Comput. Surv.* 51, 6, Article 130 (Jan. 2019), 36 pages. doi:10.1145/3291047

[14] Nabil Schear, Patrick T. Cable, Thomas M. Moyer, Bryan Richard, and Robert Rudd. 2016. Bootstrapping and maintaining trust in the cloud. In *Proceedings of the 32nd Annual Conference on Computer Security Applications* (Los Angeles, California, USA) *(ACSAC '16)*. Association for Computing Machinery, New York, NY, USA, 65–77. doi:10.1145/2991079.2991104

[15] Gianluca Scopelliti, Christoph Baumann, and Jan Tobias Mühlberg. 2024. Understanding Trust Relationships in Cloud-Based Confidential Computing. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 169–176. doi:10.1109/EuroSPW61312.2024.00023

[16] Trusted Computing Group. 2008. Trusted Platform Module (TPM) Summary. https://trustedcomputinggroup.org/resource/trusted-platform-module-tpm-summary/ Retrieved on May 29th, 2025.

[17] L. Wilke and G. Scopelliti. 2024. SNPGuard: Remote Attestation of SEV-SNP VMs Using Open Source Tools. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE Computer Society, Los Alamitos, CA, USA, 193–198. doi:10.1109/EuroSPW61312.2024.00026