A Unified Framework for Attested Confidential VM Workloads in Public Clouds

PIC2 - Master in Computer Science and Engineering
Instituto Superior Técnico, Universidade de Lisboa

João Sereno — 99249*
joaohsereno@tecnico.ulisboa.pt

Advisors:
Professor Luís Eduardo Teixeira Rodrigues
Professor Nuno Miguel Carvalho dos Santos

Abstract

The growing adoption of cloud computing has introduced new challenges in securing sensitive workloads, particularly against threats originating from the Cloud Service Provider (CSP) itself. Confidential Virtual Machines (CVMs), enabled by hardware-backed Trusted Execution Environments such as AMD SEV-SNP, promise to protect data even while *in-use* by isolating execution from the underlying infrastructure. However, despite increasing support from major CSPs—namely Amazon Web Services (AWS), Azure, and Google Cloud Platform (GCP)—current CVM offerings often lack transparency, reproducibility, and consistent trust guarantees. This thesis investigates the practical security models underpinning CVMs in public clouds, offering a formal and empirical comparison of their roots of trust, attestation capabilities, and deployment caveats.

To address these limitations, this thesis proposes a unified framework for secure CVM lifecycle management. This framework abstracts over CSPs-specific APIs to automate image hardening, secure deployment, and remote attestation, enabling reproducible and auditable workflows across multiple providers. Additionally, it will support attested interaction scenarios, where CVMs execute workloads owned by third parties—such as confidential AI inference—under verifiable trust conditions.

The effectiveness of the framework will be evaluated through benchmarking of performance, resource overhead, and verifiability depth against existing solutions such as Keylime and Constellation. By exposing critical limitations in current CVM offerings, this work lays a practical foundation for the development of trustworthy confidential computing systems in real-world cloud environments.

Keywords — Confidential Computing, Confidential Virtual Machines, Remote Attestation, Cloud Security

This project has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101189689 and the Fundação para a Ciência e Tecnologia (FCT) under grant UIDB/50021/2020.

*I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa (https://nape.tecnico.ulisboa.pt/en/apoio-ao-estudante/documentos-importantes/regulamentos-da-universidade-de-lisboa/).

Contents

1 Introduction

Cloud computing has enabled businesses of all sizes to provide scalable, global services with minimal initial investment, no infrastructure maintenance, and access to a wide array of cost-effective solutions. However, these benefits come at the cost of trusting third-party Cloud Service Providers (CSPs) with potentially sensitive workloads without fully comprehending the privileged position played by the CSP. Despite the critical nature of such data, most cloud offerings do not provide meaningful protection against a malicious or compromised CSP.

Data security is typically discussed across three states: *at-rest* (when stored), *in-transit* (when transmitted), and *in-use* (when processed). While industry-standard encryption solutions adequately protect data *at-rest* and *in-transit*, safeguarding data *in-use* remains an open challenge [1]. For services that do not require processing plaintext data (such as messaging or file storage), end-to-end data encryption can be achieved without mechanisms to protect data *in-use*. However, any service requiring computation on plaintext data must rely on mechanisms that secure it during execution.

Approaches to protect data *in-use* fall into two broad categories: algorithmic and hardware-based. Algorithmic solutions such as Fully Homomorphic Encryption [2] and Secure Multi-Party Computation [3] allow computation over encrypted data without revealing inputs or outputs in plaintext. Despite their ideal security guarantees, these techniques suffer from high performance overheads and, therefore, limited practical applicability in complex scenarios like large-scale machine learning [4–6].

In contrast, hardware-based solutions offer a more practical trade-off between security and performance. These rely on Trusted Execution Environments (TEEs), which decrypt and process data within isolated hardware-backed environments that are designed to be protected even from privileged surrounding software [1]. To reduce the trust gap introduced by executing code on third-party infrastructure, TEEs support *remote attestation*—a mechanism that enables data owners to verify the state, contents and integrity of the TEE before provisioning with sensitive data.

The fundamental distinction between these approaches lies in their different positioning within a three-way trade-off among performance, utility, and Trusted Computing Base (TCB) size—where the TCB represents the collection of hardware and software components that must be trusted for the security of the entire system. Algorithmic solutions achieve a minimal TCB that includes only the cryptographic primitives and their correct implementation, making them theoretically robust against adversaries with extensive system access. However, hardware-based TEE solutions necessarily expand the TCB to include additional components: the TEE-enabling hardware itself, its firmware, and all the software components within the TEE. Critically, when TCB components are proprietary or supplied into the trusted system in a non-verifiable way, the entities in control are therefore included in the TCB, as they may introduce vulnerabilities or backdoors without reliable detection. While this expanded TCB introduces additional attack surfaces and trust assumptions, it enables the practical performance characteristics that make TEEs viable for real-world deployment scenarios where algorithmic approaches remain computationally prohibitive [7, 8].

Hardware-based TEEs had their first major implementation with Intel Software Guard Extensions (SGX) enclaves [9, 10]. Intel SGX enables the creation of isolated memory regions, called enclaves, within a user-space process. Code and data inside an enclave are protected from access or tampering by any other surrounding software, including the guest OS and hypervisor. The TCB is considered small for SGX enclaves, since it only contains the Intel SGX-enabled CPU, its firmware and the required code to perform the sensitive computation.

However, SGX enclaves come with significant limitations: they impose strict memory constraints and require developers to build or change applications to explicitly manage trusted and untrusted code portions. These constraints complicate development and limit applicability to certain use cases. Moreover, SGX has been subject to numerous physical and software-based side-channel attacks, highlighting challenges in maintaining strong isolation in the presence of a powerful adversary [9].

To address these practical limitations, Intel, with Trusted Domain Extensions (TDX) [11][1], and AMD, with Secure Encrypted Virtualization–Secure Nested Paging (SEV-SNP) [12], built CPUs capable of isolating and protecting an entire Virtual Machine (VM)'s memory space from the hypervisor—controlled by the Cloud Service Provider (CSP). VMs with hardware-backed, confidentiality and integrity protections are commonly called Confidential Virtual Machines (CVMs). The main goal of CVMs' technology is to prevent the CSP from leveraging

[1] For this thesis, Intel's implementation of CVMs will remain out of scope.

their advantageous position to gain unauthorized access to the VM's memory plaintext contents. This approach expands the SGX enclaves' TCB requirements to contain not only the CPU hardware and firmware, but also the complete guest OS—including all applications processing sensitive data. This architectural shift eliminates the need for application-specific modifications required by SGX while substantially reducing memory constraints, as the entire CVM memory space becomes available for confidential workloads [13].

CVMs enable use cases that were previously unattainable under the same trust assumptions. The fundamental capability that relying parties seek is the assurance that their sensitive data remains invisible to all entities and components outside the CVM, while simultaneously providing mechanisms to verify that only the expected, authorized components within the CVM have access to this data. One notable application powered by this capability is confidential AI inference and training for sensitive data, where data owners require certainty that both the AI model processing their data and all accompanying software components with data access are precisely the expected implementations, with no additional components gaining unauthorized access—effectively enabling complete privacy protection for AI workloads [14]. The realization of this capability depends on continued progress in tightening trust gaps, keeping performance overheads low, and making operational experience as seamless as possible.

Although AMD SEV-SNP technology enabled major CSPs to provide CVM solutions to their customers, there are notable setbacks when attempting to verify the deployment of advertised security features and guarantees while keeping the CSP outside of the TCB. Additionally, the absence of implementation standards for AMD SEV-SNP enabled CVMs has led to implementation divergence between CSPs, ultimately compromising customers' experience by forcing them to manage the implementation heterogeneity. To address this, this thesis aims to explore and empirically validate the confidential computing guarantees currently offered by major CSPs, while also contributing to making the correct and secure use of such services more accessible and transparent to end users.

The rest of the document is organized as follows. Section 2, outlines the objectives and expected outcomes of the work. Section 3 provides the necessary technical background—detailing how AMD enables CVMs, the role and value of Trusted Platform Modules (TPMs) used in conjunction with CVMs, and two key mechanisms that help ensure a CVM starts at a correct and secure state, and proves state integrity to a remote party. Section 4 presents a thorough comparison of CVM offerings from AWS, GCP, and Azure, followed by a review of on-premise approaches addressing limitations in current industry implementations, and an overview of current tooling that supports CVM deployment and attestation. Section 5 introduces the proposed framework to be developed as part of this thesis, and Section 6 defines the methodology for evaluating its key features. Finally, Section 7 outlines the schedule for the remaining work, and Section 8 summarizes the content present in this document.

2 Goals

This section outlines the goals of the thesis, which addresses the deployment and trust assessment of Confidential Virtual Machines (CVMs) in public cloud environments. Despite growing support for CVMs by major cloud providers, there remain significant gaps in the usability, transparency, and practical assurance of the security guarantees they offer. This work seeks to address these gaps by focusing on three core contributions:

1. **Characterization of Cloud CVM Trust Models:** The first contribution will be an analysis of the trust models associated with CVM offerings across leading cloud service providers (e.g., AWS, Azure, GCP). This includes identifying roots of trust, comparing advertised and obtained threat models, available attestation mechanisms, and operational guarantees provided by each platform. The analysis will combine static documentation review with empirical validation using controlled experiments to determine whether claims align with observed behavior. The result will be a framework where developers or system administrators can transparently compare trust guarantees across different CVM providers.

2. **Design and Implementation of Tooling for CVM Lifecycle Management:** The second contribution will be the development of a toolkit to support the deployment, provisioning, and remote attestation of CVMs. This toolkit will abstract provider-specific interfaces while exposing fine-grained visibility into the achieved trust properties. The goal is to reduce the operational overhead while solving the ambiguity

and lack of transparency currently faced by practitioners, enabling reproducible and auditable deployment workflows with verifiable trust assurances.

3. **Deployment and Evaluation of Confidential AI Inference Across Trust Models:** The third contribution is to demonstrate the practical utility of CVMs by deploying a representative confidential workload—specifically, an AI inference service—on multiple cloud platforms. This will include benchmarking performance, evaluating the operational friction of trust establishment, and mapping the trust guarantees attained under each provider's CVM model. The goal is to empirically determine the highest-assurance configuration achievable under real-world constraints, thereby illustrating both the feasibility and limitations of confidential computing for AI workloads.

3 Background

This section provides the necessary technical background to support the contributions of this thesis. It surveys the foundational technologies and concepts underpinning Confidential Virtual Machines (CVMs) and their security guarantees in cloud environments.

The section is organized as follows. Section 3.1 presents an overview of AMD Secure Encrypted Virtualization–Secure Nested Paging (SEV-SNP), the hardware technology enabling memory encryption and integrity protection for virtual machines. Section 3.2 discusses the role of the Trusted Platform Module (TPM) in platform attestation. Section 3.3 outlines the CVM boot process with emphasis on measured boot sequences. Finally, Section 3.4 introduces the concept of remote attestation, which enables verifiers to assess the trustworthiness of remote systems.

3.1 Overview on AMD SEV-SNP

AMD's Secure Encrypted Virtualization (SEV) technology, introduced with the EPYC processor family, enables the creation of CVMs by encrypting the memory of each guest VM. This encryption safeguards against unauthorized access by higher-privileged components, such as hypervisors. The evolution of SEV has seen several enhancements to improve the security guarantees provided to CVMs.

A critical component supporting SEV is the AMD–Secure Processor (AMD-SP)—a dedicated microcontroller coprocessor embedded in the processor die, previously referred to as the Platform Security Processor [15]. The AMD-SP is equipped with a small, private memory to store sensitive material such as memory encryption keys, attestation artifacts, and other security-critical data. To address potential scalability limitations, the AMD-SP can dynamically allocate and manage additional pages from the system's main memory, which are encrypted and access-controlled to ensure exclusivity to the AMD-SP. References to AMD-SP dedicated or reserved memory in this document include protected auxiliary memory regions.

The AMD-SP serves as the hardware root of trust that enables SEV-SNP's security model for CVMs. What follows is an examination of how these memory confidentiality and integrity protection mechanisms are implemented by AMD-SP. Understanding these foundational security primitives is essential for evaluating the trustworthiness of AMD's SEV-SNP implementation of CVMs.

SEV and SEV-ES Initially, the Secure Encrypted Virtualization (SEV) technology focused exclusively on ensuring VM memory confidentiality, by encrypting pages with a unique key per VM, securely generated at VM creation time [15]. However, this version lacked integrity protection mechanisms for the memory or the CPU register state. To address the latter, AMD introduced Secure Encrypted Virtualization–Encrypted State (SEV-ES), which extends SEV by encrypting the CPU register state upon VM exits. Specifically, before the CPU exits the VM context, the AMD-SP securely stores the register contents in its reserved memory, preventing the hypervisor from accessing sensitive register data [16].

While encryption keys are managed by AMD-SP firmware, actual memory encryption and decryption are offloaded to specialized hardware within the on-die memory controllers. These Advanced Encryption Standard (AES) engines are controlled by an additional metadata bit per memory access [15].

SEV-SNP Building upon SEV-ES, Secure Encrypted Virtualization–Secure Nested Paging (SEV-SNP) extends the SEV security model by introducing memory integrity protections [12]. SEV-SNP implements a new data structure called the Reverse Map Table (RMP), residing in AMD-SP dedicated memory, to track the ownership and access rights of each physical memory page. Each RMP entry maps a system physical address (sPA) to a guest physical address (gPA) and ownership metadata, ensuring that each physical page can only be owned by one entity and preventing the address remapping attacks that affected earlier SEV versions [12, 13]. Ownership may be assigned to a VM, the hypervisor, or the AMD-SP itself. A write memory access is blocked if it occurs from a context other than the page owner's, or if the translated gPA does not match the RMP's entry. Read accesses are always allowed because memory remains encrypted—AMD-SP guarantees that decryption only occurs when the page is read within the page owner's context [12].

SEV-SNP Additions SEV-SNP also introduces optional features to support different threat model requirements. These configurable security features can be reliably verified through inspection of the AMD-SP's remote attestation reports, ensuring transparency in the security posture of deployed CVMs.

One such feature is the Virtual Machine Privilege Levels (VMPLs), which allows the hypervisor to partition a CVM's address space into up to four privilege levels (VMPL0–VMPL3), each associated with a specific virtual CPU (vCPU) [12]. Access permissions are specific to each page and encoded in the RMP—enforced during RMP checks. The `RMPADJUST` instruction allows privilege level adjustments, with the restriction that a vCPU at a given VMPL can only modify permissions for less-privileged levels. Despite being initially controlled by the hypervisor, this feature is especially useful for implementing Secure VM Service Modules (SVSMs), detailed in Section 4.1.

SEV-SNP additionally supports CVM live migration, enabled by a special migration agent implemented as a CVM [12]. With live migration, Cloud Service Providers (CSPs) may manage their infrastructure resources more efficiently, and customers may not suffer from disruption of service in case physical maintenance is needed. This mechanism, however, remains out of scope for this thesis, as it is not yet supported in any major CSP.

Finally, SEV-SNP introduces configurable mitigations against complex attack vectors. For instance, to reduce the side-channel's attack surface, simultaneous multi-threading can be disabled, improving isolation at the cost of performance [12]. AMD chose to make such mitigations optional, allowing CSPs to align configurations with the specific threat models they want to provide.

3.2 Trusted Platform Module

Other than providing an isolated Trusted Execution Environment (TEE), CVMs need to provide reliable evidence that identifies the components within the trusted environment. The AMD–Secure Processor (AMD-SP) provides a single measurement representing the initial VM's memory contents. Hypervisors need to choose this initial memory content (as detailed in Sections 3.3 and 4.1), which typically is leveraged to issue the measurement of the CVM's firmware and no other boot component [17]. Such incomplete boot measurement leaves gaps for undetectable boot-level attacks. The Trusted Platform Module (TPM) is the foundational device of possible solutions that aim to cover this gap.

A Trusted Platform Module (TPM) is a dedicated cryptographic coprocessor implemented as a discrete hardware component that provides a hardware root of trust for general physical machines [18, 19]. The TPM operates independently from the main processor and system memory, establishing a hardware security root of trust that is designed not to be compromised through software attacks alone. This physical isolation is fundamental to the TPM's security model, as it ensures that critical security operations and sensitive cryptographic material remain protected even when the host system's software stack is compromised.

The TPM 2.0 specification [20] defines several core security functions that make it indispensable for secure computing architectures. These include secure key generation and storage, cryptographic operations using hardware-protected keys, platform integrity measurement and attestation, and secure boot verification. The TPM maintains these functions through a combination of volatile and non-volatile memory, cryptographic engines, and a dedicated execution environment that operates according to strictly defined interfaces and access controls.

This section covers the security primitives provided by TPMs that offer complementary capabilities to AMD-SP, essential for establishing comprehensive trust in public cloud-hosted CVMs' contents. While AMD-SP provides

hypervisor-issued launch measurements through its launch digest mechanism (covered in Section 3.3), TPMs enable guest-issued integrity measurement of the boot process through standardized interfaces. This distinction is crucial for CVM security architectures, as guest-issued measurements can provide more granular and transparent attestation of the CVM's internal state than hypervisor-issued measurements, which may be opaque or incomplete.

Platform Configuration Registers Platform Configuration Registers (PCRs) represent one of the most critical components of the TPM's integrity measurement architecture. PCRs are special-purpose registers within the TPM that store cryptographic hash values representing the measured state of various system components [19]. The TPM 2.0 specification defines at least 24 PCRs (numbered 0–23), though implementations may provide additional registers for vendor-specific purposes.

The interface to PCRs follows a specific cryptographic protocol that ensures the integrity of submitted measurements. A PCR value can only be modified through the "extend" operation, which takes the current PCR value, concatenates it with new measurement data, and stores the hash of this combination back into the PCR [20]. Mathematically, could be represented as: $\text{PCR}[i] = H(\text{PCR}[i] \parallel \text{new\_measurement})$, with H being the TPM's hash function and $\parallel$ denoting the concatenation operation. This approach creates a cumulative hash chain that captures the entire sequence of measurements in chronological order, making it computationally infeasible to forge a specific PCR state without knowing all the intermediate measurement values.

Each PCR serves a designated purpose within the platform's measurement architecture, although there is no target object enforcement at the time of PCR extension. PCRs 0–7 typically measure firmware and bootloader components, PCR 8–15 measure operating system components and configuration, while PCRs 16–23 are reserved for application-level measurements and debug purposes. These PCRs are categorized into Static Root of Trust for Measurement (SRTM) and Dynamic Root of Trust for Measurement (DRTM). SRTM measurements (PCRs 0–15) capture the boot-time state of firmware and OS components and remain static after system initialization, while DRTM measurements (PCRs 16–23) support runtime modifications for dynamic trust establishment. Other than extended, the PCRs may only be reset to their initial known values (defined in TPM 2.0 specification) [20]. System components are assigned locality levels (0–4) by the platform, which determine their permissions to reset specific PCRs. Proper locality assignment is security-critical, as components with inappropriate reset permissions could extend PCRs with false measurements, compromising the measurement chain's integrity [20].

TPM Remote Attestation TPM remote attestation is enabled by requesting a quote of the current state of PCR values to the TPM. This quote is digitally signed by a chip-specific private key, effectively anchoring the quote to the physical TPM. The quote also includes a nonce provided by the requesting party to ensure freshness, and additional metadata such as the PCR selection and signature algorithm used.

The security significance of TPM quotes lies in their ability to create a verifiable chain of trust from the hardware root. The quote is signed using an Attestation Key (AK), both derived from and endorsed by an Endorsement Key (EK) unique to each TPM. The TPM manufacturer should provide the necessary digital certificates to bind the quote (or the EK) to the TPM manufacturer. The specific key hierarchy and management are specific to the implementation.

Virtual TPM The virtualization of the TPM facilitated scaling the physical TPM's features to serve potentially any VM, even when co-located with other virtual Trusted Platform Module (vTPM) enabled guests. This virtualization of trust services is essential for maintaining security guarantees in cloud environments while preserving the familiar TPM interface that applications and operating systems expect.

Despite the promising idea, the implementation of vTPMs is crucial since trusting any of its functionalities ultimately relies on the hardware manufacturer's root of trust. With vTPMs, the entity in which the anchor is set widely depends on the implementations. Understanding these variations is key for evaluating the security guarantees that are lost or maintained when compared to a physical TPM.

- **Hardware-anchored vTPM implementations** maintain a direct connection to the physical TPM on the host system. In this model, each vTPM instance derives its cryptographic identity from the physical TPM, often through hierarchical key derivation or by using the physical TPM as a root certificate authority.

IBM's implementation described by Berger et al. in [21] exemplifies this approach, where vTPM instances are created as children of the physical TPM, inheriting their trust properties from the hardware root.

- **Hypervisor-anchored vTPM implementations** place trust in the hypervisor to provide TPM functionality. In this model, the hypervisor implements TPM operations in software while maintaining isolation between different vTPM instances through the same mechanisms used to isolate VMs. Qemu's vTPM emulator [22], VMware's vSphere [23], and others follow this approach, relying on the hypervisor's security architecture to protect vTPM state and operations. The security of this model depends critically on the hypervisor's integrity and the effectiveness of its isolation mechanisms, making it suitable for environments where the hypervisor is considered part of the Trusted Computing Base (TCB).

- **Software-only vTPM implementations** provide TPM functionality entirely through software emulation without relying on either physical TPMs or specialized hypervisor support. These implementations, such as the software TPM implementations used in some development and testing environments, offer maximum flexibility and portability but provide the weakest security guarantees. Since all cryptographic operations occur in software within the potentially compromised guest environment, these implementations are vulnerable to the same classes of attacks that affect other software-based security mechanisms. Even with the lack of reliability, it is worth noting that this approach does not rely on the hypervisor implementation, which gains some relative relevancy when deployed within a CVM.

3.3 Boot Process

The boot process of CVMs establishes the foundation upon which all subsequent security guarantees rely, making it a crucial operation in the overall security architecture. This foundational nature means that any compromise occurring at any stage of the boot sequence fundamentally undermines the security guarantees of all following stages, creating a cascading effect that can compromise the entire system's trustworthiness.

To address these inherent vulnerabilities, implementing a measured boot process that can be remotely attested and endorsed by trusted components provides relying parties with verifiable guarantees that the boot process of the corresponding CVM executed according to expected parameters. This approach requires relying parties to have these measurements independently verified to ensure that the CVM's boot sequence followed secure procedures and maintained integrity throughout the process. The measured boot process encompasses two distinct phases: the pre-boot phase, during which CVM resources are allocated and initial memory contents are loaded, and the actual boot phase, where the CPU begins executing the first instruction within the CVM's context and continues through complete system initialization while continuously performing measurements of boot components—both to be detailed in this section.

Section 3.4 covers how the collected evidence becomes trustworthy for remote parties, and how verifiability levels of such evidence impact the trust model of the deployed CVM.

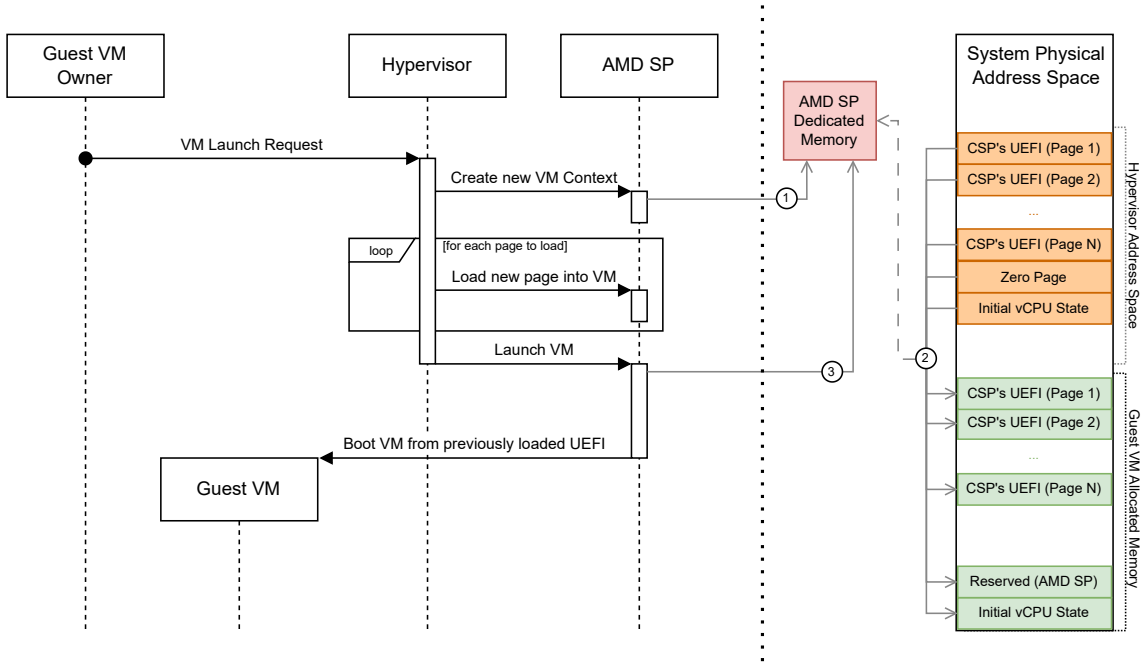
It should be noted that some procedures described are implementation-dependent on the specific hypervisor or boot components employed. When necessary for clarity and specificity, this analysis will reference implementations from QEMU's[2] x86 hypervisor, Open Virtual Machine Firmware (OVMF) [24] (which serves as the Unified Extensible Firmware Interface (UEFI) firmware implementation providing essential system initialization services that are not included within a VM's image), and the Linux kernel. While alternative implementations may be proprietary, fundamental behaviors and security principles remain common across them.

Pre-boot measurement The pre-boot measurement process constitutes a measured initial VM loading sequence that establishes a hopefully verifiable chain of trust for CVMs, with the measurement's integrity endorsed by AMD-SP. As illustrated in Figure 1, this process encompasses several distinct phases, each contributing to the construction of a tamper-evident Launch Digest (LD) that serves as the foundation for subsequent remote attestation.

The initialization phase commences when the guest owner issues a CVM instantiation request to the host system. The hypervisor responds by invoking the `SNP_GCTX_CREATE` instruction (operation 1 in figure 1), which establishes the guest context data structure within a reserved memory page owned by AMD-SP with exclusive

[2] https://www.qemu.org/

Figure 1: AMD SEV-SNP CVM pre-boot measurement and launch



control. This protected page encapsulates critical cryptographic material, including the memory encryption key that will secure all subsequent guest memory accesses [25]. Following successful context creation, the hypervisor calls the SNP_LAUNCH_START instruction, which initializes the guest context page contents and crucially sets the Launch Digest (LD) field to zero [25].

The memory provisioning phase represents the core of the measurement process, wherein guest memory is populated incrementally through repeated invocations of the SNP_LAUNCH_UPDATE instruction (operation 2 in figure 1). Each invocation performs a dual function: it transfers a memory page from the hypervisor's address space to the given gPA while simultaneously encrypting the page contents using the CVM-specific memory encryption key, and it extends the LD through a cryptographic hash chain operation [25]. The mathematical formulation of this digest extension is expressed as: LD = SHA384 (LD ‖ SHA384 (page_content) ‖ page_metadata). This construction ensures that any modification to page contents, metadata, or loading sequence will result in a evidently different final digest value.

The page metadata accompanying each SNP_LAUNCH_UPDATE call encodes several critical parameters that influence both the runtime behavior and measurement outcome. These parameters include the page type classification, content length, Virtual Machine Privilege Level (VMPL) access control policy, and the target gPA. Other than normal memory pages, there are Virtual Machine Save Area (VMSA) pages that describe a single vCPU's properties, zero pages with no contents, CPUID pages[3] that describe CPU function values, which AMD-SP validates before use, and secret pages[3] that AMD-SP injects generated keys to establish a secure communication channel between the guest and AMD-SP (the hypervisor does not have access to these keys) [25].

Although SNP_LAUNCH_UPDATE call order is reflected in the measurement, the hypervisor is free to load the CVM arbitrarily, since the metadata specifies the target gPA. The inclusion of the gPA in the measurement chain binds the initial memory layout to the LD, thereby ensuring a secure and attestable initial binary execution. Typically, the first binary instructions to run in the CVM context belong to the CVM's Unified Extensible Firmware Interface (UEFI)—the first major boot component responsible for identifying the memory layout, discovering

[3]Exceptionally, since the contents may vary between identical launches, the LD updates its value as if it was a zero page, otherwise, the LD could not be reproduced.

boot devices, and other initialization tasks—therefore, the UEFI, or some part of it, needs to be loaded initially.

The finalization phase is triggered by the `SNP_LAUNCH_FINISH` instruction (operation 3 in figure 1), which completes the LD computation, preventing future modifications for this CVM's lifecycle, and securely stores the final digest in AMD-SP's dedicated memory for inclusion in the subsequent attestation report. Upon successful completion of this instruction, the hypervisor may initiate guest execution through the standard x86 `VMRUN` instruction [25].

Boot measurement Following the AMD-SP's pre-boot measurement establishment, the CVM transitions to a vTPM-based boot measurement chain that extends measurements through the guest's software stack. The most prevalent scenario for CVMs involves utilizing a vTPM as the measurement endorsement mechanism. Any evidence endorsed by the vTPM inevitably depends on the vTPM's trust anchor, as established in Section 3.2. Since vTPM devices available to CVMs hosted in most Cloud Service Providers (CSPs) are not customizable and remain under the CSP's implementation control, the anchoring mechanism reliability becomes implementation-dependent. Section 4.1 explores each available anchoring option and examines their respective consequences on the overall trust model, while the remainder of this section proceeds under the assumption of a trusted vTPM trust anchor.

The boot measurement process involves multiple participating components, each requiring measurement *prior* to execution to maintain the integrity of the measurement chain [20, 26]. After a component is measured by the previous boot component, the vTPM's extend interface creates a tamper-proof record that preserves evidence of any unexpected components. If an unauthorized component is present, it will leave an irreversible mark in the measurement chain that can be subsequently identified through vTPM quote verification. When a component's measurement matches expected values, this indicates the component is as expected, and, since it is a correctly functioning boot component, it proceeds with loading and measuring the subsequent component before transferring execution control.

A critical vulnerability exists in this measurement chain: only one component executes before being measured, which is the Core Root of Trust for Measurement (CRTM). In physical machines, the CRTM is immutable and fused into the CPU's silicon, providing an inherent trust anchor. However, in VMs and CVMs, the CRTM is incorporated within the UEFI firmware, making it mutable and ultimately controlled by the hypervisor. While a correct CRTM would self-measure and extend the assigned PCR with a digest of its own code binary, a compromised CRTM could extend the same PCR with the expected measurement without actually performing the self-measurement procedure. This compromised CRTM could then load a compromised version of the next component while injecting expected measurements into the TPM, propagating this compromise through subsequent boot stages until achieving complete application-level VM compromise.

The pre-boot measurement phase plays a critical role in mitigating this vulnerability by reliably measuring the CRTM before its execution, thereby preventing this potential break in the measurement chain. As seen before in this section, the pre-boot phase typically measures the entire UEFI, which includes the CRTM, effectively preserving a trusted and gapless measurement chain from the very beginning of the boot process.

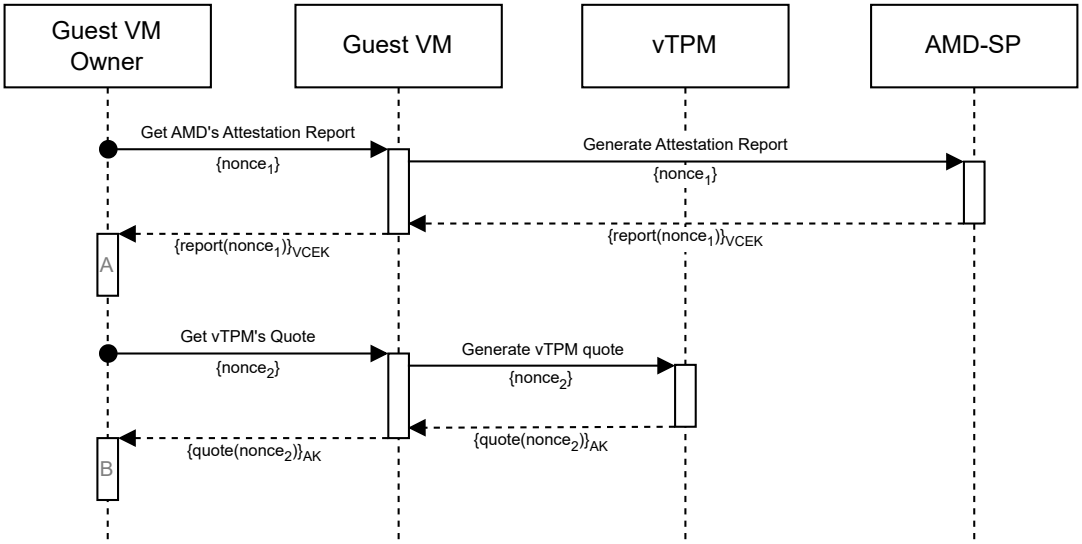
Since measurements from different components are often extended to the same PCR, and these components may execute in different chronological orders even across similar CVM launches, a measurement log accompanies the TPM quote to provide individual measurement details that comprise the final PCR values. This log's trustworthiness is validated by reproducing the measurement calculations and verifying that the reproduced PCRs match the TPM-endorsed values. When this verification succeeds, the measurement log itself becomes trustworthy, enabling individual measurements to be verified and interpreted independently of any chronological nondeterminism that may occur across different boot sequences.

3.4 Remote Attestation

Remote attestation is a protocol that enables a verifier to assess the trustworthiness and integrity of a remote computing platform. The Internet Engineering Task Force (IETF) Remote ATtestation procedureS (RATS) architecture [27], defined in Request for Comments (RFC) 9334, establishes the generalized conceptual framework for the attestation process across heterogeneous computing environments.

The RATS architecture delineates three primary entities: the *Attester* (the platform generating *evidence* of its state), the *Verifier* (the entity evaluating attestation *evidence*), and the *Relying Party* (the consumer of attestation

Figure 2: CVM remote attestation sequence



results). Reliable evidence constitutes a collection of measurements that accurately and completely depict a TCB component's state without possibility of generation or tampering by untrusted entities. As seen in Section 3.3, this reliability is achieved through a measurement chain where each component measures the next *before* transferring control, thereby transitioning trust from the root of trust. The root of trust component cryptographically signs the collected measurements, providing tamper-resistant and non-repudiable endorsement of the results. The verifier validates the trusted component's signature before verifying the measurements themselves, either by comparing them against reference trusted measurement values or by reproducing measurements from the trusted measurable components.

In figure 2, a RATS-compliant attestation sequence is illustrated for both AMD-SP and vTPM evidences. Currently, cloud environments are aiming to achieve these procedures, where AMD-SP evidences the correct and secure configuration of the CVM and initial memory contents, and the vTPM evidences a trusted boot process of the VM. Both pieces of evidence require verification to be deemed valid and meaningful (procedures `A` and `B` in figure 2). In procedure `A`, the Guest VM Owner (representing the Verifier), needs to check (1) if the provided nonce is included in the report and covered by the cryptographic signature; (2) if the Versioned Chip Endorsement Key (VCEK) signature has not been tampered with; (3) if a VCEK certificate endorsed by the AMD SEV Key (ASK) exists and is valid; and (4) if a ASK certificate endorsed by the AMD Root Key (ARK) exists and is valid. The public key certificates for ASK and ARK are available online[4]. The Launch Digest (LD) included in the attestation report is ideally reproducible by simulating the page loading process given the binary contents of the initial VM memory. This requires a transparent documentation on how the hypervisor executed the `SNP_LAUNCH_UPDATE` instruction for the running CVM. Similarly in procedure `B`, the Verifier needs to check (1) if the provided nonce is included in the vTPM quote and it is covered by the cryptographic signature of the Attestation Key (AK); (2) if the AK signature has not been tampered with; (3) if a AK certificate endorsed by the Endorsement Key (EK) exists and is valid; and (4) if the entity responsible for EK is trusted (might require another endorsement level, depending on implementation). Additionally, the validation of PCR values happens at this stage, where some PCRs can be directly compared with known-good values, while others require full PCR recomposition using the individual measurement log. In the latter, individual measurements are verifiable by themselves, but, typically due to non-deterministic sequencing of the measurements, the PCR is not directly verifiable.

[4] https://www.amd.com/en/developer/sev.html

Remote attestation in real-world scenarios faces significant challenges due to the complexity of establishing trusted measurement chains across multiple participating entities. The absence of universally accepted reference measurement values aggravates these difficulties, while achieving measurement reproducibility requires extensive transparency from all measurable components. Even when a trusted and reproducible measurement chain is established, selecting a reliable root of trust remains critical for securely maintaining and endorsing attestation measurements.

Trust relationships in secure computing environments exist on a spectrum rather than as binary choices, requiring relying parties to evaluate components based on the relative strength of their security guarantees. For instance, while both AMD-SP and hypervisor-anchored vTPM implementations may be deemed trustworthy within a deployment, their attestations carry different weights due to fundamental differences in their security foundations. AMD-SP provides hardware-rooted isolation and other security features that diminish the attack vectors, whereas hypervisor-anchored vTPM implementations rely on hypervisor isolation and software-based operations. This graduated trust model enables relying parties to accept attestations from multiple sources while weighting them according to their assurance levels and specific threat models.

Furthermore, for a better understanding and comparison, I propose a software verification depth taxonomy aiming to clarify the different depth levels of trust that may be established regarding a certain software component, each with distinct and incremental security guarantees, based on *attestation levels* proposed in [28].

L1: *Root of Trust Link Authenticity* At the foundational level, verifiers validate evidence freshness, cryptographic signatures, certificate chains anchored at a trusted certificate authority, and measurement coverage of the specific component. This evidence is endorsed by the root of trust that ensures the integrity of its contents. However, this level provides no semantic interpretation of the measurements themselves.

L2: *Reference-Based Measurement Matching* The second level involves comparing reported measurements against known reference values (often referred to as golden measurements). This verification requires access to reference databases or the ability to independently compute expected measurements from known measurable components. At this level, semantic interpretation is limited to what the trusted reference value source can provide. Trust is extended to the entities providing reference values.

L3: *Binary Measurement Reproducibility* At the third level, verifiers possess the actual binary artifacts whose measurements are attested. Verification involves recomputing hashes or other cryptographic measurements locally and comparing against reported values. If successful, the verifier has assurance that the binary artifacts in hand were the ones measured remotely and possibly executed (depending on the previously measured trusted component). This provides a larger auditable surface and greater accountability for the binary authors if these binary artifacts were cryptographically endorsed by them.

L4: *Source Code Measurement Reproducibility* The fourth level involves access to source code and build environments, offering the maximum auditable surface and enabling independent compilation and measurement computation. This approach requires comprehensive toolchain verification and reproducible build processes. Trust assumptions are minimized to the compiler, build environment, and measurements.

4 Related Work

This section reviews the current state of CVMs usage and attestation. Section 4.1 analyzes and compares configurations offered by major CSPs, highlighting their limitations and associated threat models. Section 4.2 explores available tools that help users make the most of the security guarantees currently offered by CSPs.

4.1 Confidential Computing Offering to the General Public

Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) have all introduced support for AMD SEV-SNP-based CVMs in their services, respectively, in Elastic Compute 2 (EC2), Azure Virtual Machines instances, and Google Compute Engine (GCE). While these services promise memory confidentiality and integrity protection against privileged adversaries, their configurations reveal differences in security transparency and attestation capabilities, as shown in Table 1.

The security mechanisms (see Section 3) collaboratively aim to offer the relying party a single, yet complex, security guarantee: “Only trusted software components have access to plaintext sensitive data”. The following

Table 1: Summary of available CVM configurations in AWS EC2, Azure VMs, and GCP GCE

	Supported Instance Types	AMD EPYC Generation*	Locations	Min. Price per hour per vCPU (USD)	AMD SEV-SNP enablement fee per hour per vCPU
EC2	c6a m6a, r6a	Milan (3rd)	West Europe East US	0.0530	+10%
Azure	DCas_v5, DCas_v6† ECas_v5, ECas_v6†	Milan (3rd) Genoa (4th)†	East & Southeast Asia Central India Middle East North, Central & West Europe East & West US	0.0520	0%
GCE	C2D, C3D C4D‡, N2D	Milan (3rd) Genoa (4th) Turin (5th)‡	Southeast Asia West Europe Central US	0.0549	+5%

Information gathered on May 17th, 2025.
* Naples (1st) and Rome (2nd) generations offerings were suppressed since they do not support AMD SEV-SNP.
† Although documented, they were not available to be instantiated.
‡ No pricing was found as it is still in technical preview.

paragraphs progressively compare CSP's different implementation aspects, their shortcomings, and qualities.

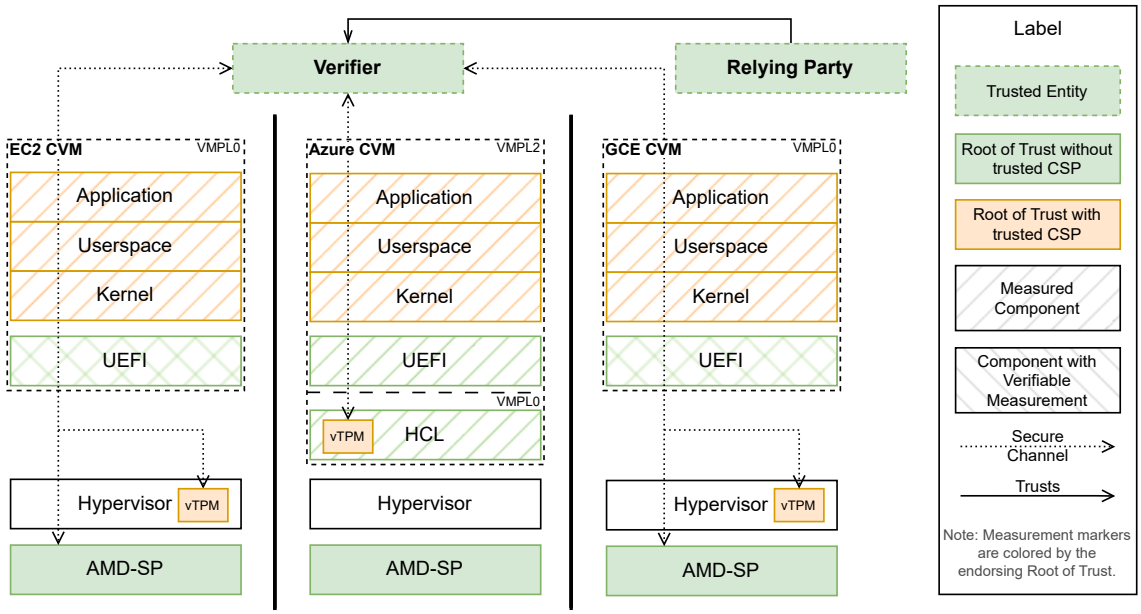
All three platforms support deployment through either provider-supplied base images or customer-uploaded VM images [29–32]. However, the use of provider-supplied images fundamentally contradicts CVM security objectives by requiring trust in the CSP, as any auditing mechanism ultimately depends on the CSP to provide the actual deployed image. Consequently, uploading custom VM images built in trusted environments represents the optimal security approach across all platforms, establishing verifiable initial trust states despite remaining vulnerable to pre-instantiation tampering attacks. Note that no VM image includes the VM's Unified Extensible Firmware Interface (UEFI); therefore, the customer cannot customize it. Figure 3 contains an overview of the hardware and software components present in each CSP-hosted CVM.

Firmware Transparency and Measurement Challenges The treatment of UEFI and AMD-SP measurements reveals fundamental differences in security architectures among providers. AWS distinguishes itself by publishing specific modifications to the Open Virtual Machine Firmware (OVMF), an open-source UEFI implementation from the Tianocore EDK II project [24,33]. AWS deploys this patched OVMF version across their AMD SEV-SNP CVM instances, with the AMD-SP measurement covering the UEFI component. This well-defined measurement scope, combined with open-source firmware and measurement sequence transparency, enables complete reproducibility and independent verification through tools like Virtee's[5] command-line utilities for calculating expected measurements [34].

Azure adopts a fundamentally different approach that undermines the security value of AMD-SP measurements entirely. The platform's proprietary UEFI, closed-source Hardware Compatibility Layer (HCL), and opaque measurement methodology render the AMD-SP measurement effectively meaningless for security verification [17]. Without documentation of what the measurement encompasses or how it is derived, combined with Azure's closed-source UEFI, prevents independent reproduction of AMD-SP measurements, forcing practitioners to adopt insecure "trust-on-first-use" methodologies where initial observations serve as unverifiable golden references. This approach violates established security principles and creates operational inconveniences and vulnerabilities as the detection of a different attested measurement could indicate both a CVM boot-level attack or a harmless UEFI update, without distinction.

[5] https://virtee.io/

Figure 3: Entities and components present in Remote ATtestation procedureS (RATS) attestation at AWS, Azure and GCP



GCE occupies an intermediate position between AWS's transparency and Azure's opacity. Like AWS, GCE's AMD-SP measurement covers only the UEFI and special memory pages, providing a well-defined measurement scope. While GCE's endorsed UEFI binary can be retrieved at runtime from public-facing Google buckets [35], its source code remains private, limiting the verifiability depth of this critical component. Google has recently made progress by open-sourcing the measurement derivation process, though the implementation requires adaptation for practical customer use, as it remains embedded within internal tooling contexts [36]. This approach provides more transparency than Azure while currently falling short of AWS's accessible and fully reproducible measurements.

Virtual TPM Implementations and Trust Anchoring All three platforms provide vTPM functionality, but with fundamentally different trust models and security guarantees. AWS offers NitroTPM as an optional component operating at the hypervisor level [37]. However, NitroTPM's trust foundation depends on AWS-managed hypervisor components rather than independently managed hardware-rooted security primitives, effectively requiring AWS inclusion in the customer's TCB for meaningful security guarantees [17].

Azure implements vTPM functionality through its Hardware Compatibility Layer (HCL), a proprietary paravisor utilizing Virtual Machine Privilege Levels (VMPLs) for hardware-enforced isolation from guest components, while also relying on AMD SEV-SNP for hardware-backed isolation from the hypervisor [38]. Despite providing the best architecture (among the three) for effective CSP exclusion of the TCB, the HCL's effectiveness is compromised by opaque measurement procedures and proprietary implementation. Without cryptographic measurement verification, the vTPM implementation cannot be linked to a hardware root of trust, undermining the provenance of meaningful security guarantees unless Azure is included in the customer's TCB, which allows the reliance on Azure-endorsed vTPM quotes [17].

GCE, similarly to AWS, implements hypervisor-level vTPM devices; therefore, these measurements cannot be trusted as they remain under hypervisor control and are vulnerable to undetectable malicious modification [39]. To trust the vTPM's measurements, Google needs to be included in the customer's TCB [17].

Attestation Architecture and Boot Chain Security The AMD-SP serves as the sole independent hardware root of trust across all implementations, but critical security gaps emerge beyond initial UEFI measurement. None of the platforms extend AMD-SP-rooted measurements to subsequent boot components; hence, no software

component on top of the firmware gets its verifiability root at AMD-SP. Additionally, all platform vTPMs are either hypervisor-anchored or lack a verifiable anchor. Without extending the TCB to include the CSP, customers cannot reliably attest to bootloader, operating system, or application integrity, leaving CVMs vulnerable to boot-level attacks without detection [17].

It is also worth noting that each software stack layer's reliable verifiability depends on the layer below. For instance, an unverified malicious kernel could override the measurement process of the userspace layer, providing expected applications' measurements when, in reality, those could be compromised. For that reason, when classifying the verifiability depth level against the taxonomy defined in the subsection 3.4, no software component with the same measuring root of trust can have a deeper level of verifiability than the previous layer.

Azure introduces additional complexity through its HCL caching mechanism for attestation reports. While cached reports remain verifiable through AMD-SP signatures, disabling direct access to AMD-SP attestation reports eliminates freshness guarantees essential to assure the evidence's authenticity [17, 40]. The inability to request fresh attestation reports with customer-defined nonces enables replay attacks using legitimate but outdated reports, potentially from different CVM instances.

AWS's Nitro system, despite proprietary implementation, provides publicly available design documentation [37] and has undergone an independent security assessment by NCC Group [41]. However, this review focused on design-level analysis rather than implementation verification, still requiring trust in AWS's security claims.

Security Trade-offs and Practical Implications The architectural limitations create two distinct security scenarios across all platforms. When excluding the CSP from the customer's TCB, vTPM attestation mechanisms cannot be trusted, leaving CVMs vulnerable to boot-level attacks from both CSPs and external adversaries. Malicious bootloaders, compromised operating systems, or malicious applications could execute undetected in this configuration.

Conversely, including the CSP in the TCB enables vTPM attestation, significantly reducing the attack surface by detecting boot-level compromises from external adversaries. Under this model, only the CSP retains exploitation capability, contradicting fundamental CVM objectives but providing demonstrably stronger practical security guarantees within current architectural constraints. However, this extension of trust also contributes to a larger attack surface than ideal, appending CSP vulnerabilities to their own. For example, an external entity might be able to exploit a CSP-managed vTPM's vulnerability and, therefore, compromise the customer's system.

Platform-Specific Considerations Azure's forthcoming OpenHCL [42] represents a potential improvement over current proprietary implementations, though production environments continue to use the closed-source variant [42]. OpenHCL switch combined with transparent and reliable measurement mechanisms rooted in AMD-SP, could potentially provide the best in the market CSP's TCB exclusion capabilities without undermining CVM security guarantees.

Although Azure and Google provide attestation verification services, no platforms offer a solution capable of fulfilling the RATS's Verifier role without including the CSP in the customer's TCB. While this absence reflects the inherent challenge of establishing trust without CSP involvement, it underscores the fundamental tension between CVM security objectives and practical deployment constraints in current cloud environments.

Future Prospects Several academic efforts have explored mechanisms to exclude the CSP from the customer's TCB. All of the following require the hypervisor to execute additional behaviors to enable proposed mechanisms or different CVM architectures, which customers have no control over on their cloud deployments. Although these proposals may not be implemented just yet, they serve as implementation guidelines that CSPs could follow to offer better security guarantees for their customers, as it ultimately depends on their efforts to do so.

One notable proposal, SNPGuard [43], not only binds the UEFI to the AMD-SP measurement (as CSPs do), but also the Linux kernel, kernel command line arguments, and the initramfs—enough boot components that enable a completely verifiable minimal OS with limited capabilities. It requires the hypervisor to hardcode the measurements of these components in UEFI's binary, which are then loaded into the CVM and, therefore, measured by AMD-SP. The UEFI then performs a secure Linux Direct Boot [44], by loading each component, measuring it, and comparing the measurement with the hypervisor-provided one. This solution is feasible since

the guest owner has access to the correct UEFI and the extra measured components, so they can reproduce the expected measurement. If the attestation report has the expected measurement, the guest owner knows that a correct UEFI was loaded and the correct extra components' measurements were inserted. Because the UEFI is correct, it will compare the extra components' measurements before transferring execution to them; otherwise, UEFI would abort the boot process. Upon a successful boot process, and the inclusion of the expected measurement in the AMD-SP attestation report, the guest owner has attested the UEFI, kernel, kernel command line arguments, and the initramfs, with a single hardware root of trust— the AMD-SP. This solution does not use a TPM or vTPM device.

Decentriq[6] documented another way to completely mitigate all the hypervisor's unmeasured inputs on the boot process [45]. Instead of deploying full, traditional VM-based CVMs, they figured that microVMs (popularized by the Firecracker project[7]) had one interesting implementation consequence—hypervisor isolation; therefore, the hypervisor inputs were minimal compared to traditional VMs. According to [13], hypervisor boot input types on traditional VM-based CVMs are summarized as firmware code, UEFI variables, Advanced Configuration and Power Interface (ACPI) tables, PCI option ROM, OS loader, kernel code, and kernel command line parameters, which require AMD-SP or a vTPM to reliably measure them. With micro confidential VMs (microCVMs), AMD-SP initially loads and measures the complete package, containing the UEFI, initramfs, kernel, kernel command line arguments, CPU layout, and the memory layout. Given that all of the used components are open-source, the measurement becomes fully reproducible. Although microCVMs make hypervisor inputs easier to measure, they lack OS flexibility, support for PCI-only devices (*e.g.* GPUs), and live-migration capabilities [46]. MicroVMs were designed and implemented to host Function-as-a-Service workloads—ephemeral and short-lived guests, and, therefore, they currently might not be fit to sustain many use cases out of the box when turned into microCVMs.

As seen in Azure's CVM implementation, Virtual Machine Privilege Levels (VMPLs) may be used to offer an environment for privileged code modules to execute isolated from both the hypervisor and the guest OS. To overcome possible disparity of implementations, AMD has published a specification for standard communications between the guest OS and the collection of privileged services—the Secure VM Service Module (SVSM) [47]. The SVSM allows, among other things, the secure and isolated execution of a vTPM available to the whole system. For its measurements to maintain relevancy, it needs to be measured by AMD-SP on CVM launch—which is the recommended behavior in [47] as they indicate to measure the entirety of the SVSM, including the vTPM. With the SVSM, additional services may be developed independently, as the interface is already specified. Since the guest OS only needs to use the SVSM interface, remote attestation procedures may be further standardized and broadly used without requiring many changes in both the CSP-managed components or user-managed components. A popular ongoing project, CoCoNut-SVSM [48], is actively moving forward into building services and standardizing remote attestation across different platforms.

In [49], the authors achieved a complete vTPM-focused implementation of the SVSM, where they execute an ephemeral state vTPM (e-vTPM) inside VMPL0, isolated from the guest OS running in VMPL1, with no other additional services. As recommended by the SVSM specification, they effectively link the vTPM anchor of trust to AMD-SP by measuring all the VMPL0 components on boot. Unfortunately, this design is not reproducible in current CSPs environments since VMPL feature activation and configuration are controlled by the hypervisor. A customer is not able to define VMPL layouts, use customized firmware, or modify AMD-SP measurement sequences.

Achievable Trust Models Comparison Table 2 contains a summarization of the different trust models that can be achieved given the current offering of CVMs in CSPs and the state-of-the-art proposals for on-premises-based solutions. Firstly, the only solutions that do not enable freshness verification of AMD-SP attestation reports are the ones hosted in Azure—regardless of trusting the HCL or not [17, 28]. Therefore, no software component in Azure CVMs can be verifiable at any level with trust anchored at AMD-SP. Secondly, all solutions except SNPGuard and Decentriq's microCVMs use vTPMs. Since vTPMs in CSPs are closed-sourced and/or running in untrusted environments, their security features are only verifiable through CSP endorsement, which, although every CSP makes it available, is only credited if the CSP is in the customer's TCB. Furthermore, when dealing

[6] https://www.decentriq.com/
[7] https://firecracker-microvm.github.io/

Table 2: Summary of possible trust models given the verifiability levels on CVM deployments within CSPs and on-premises

	Untrusted CSP			Trusted CSP			Other Proposals		
	AWS	**Azure**	**GCP**	**AWS**	**Azure**	**GCP**	**SNPGuard [43]**	**microCVM [45]**	**e-vTPM [49]**
AMD-SP	✓	✗	✓	✓	✗	✓	✓	✓	✓
vTPM	✗	✗	✗	✓	✓	✓	–	–	✓
Firmware	●	○	◕	●	◑	◕	●	●	●
Other boot components	○	○	○	●	◑	◕	●	●	●
Kernel	○	○	○	●	◑	◕	●	●	●
Userspace and applications	○	○	○	●	◑	◕	○	○	●

✓: Verifiable authenticity ✗: Non-verifiable authenticity –: Not used
Black coloring represents trust anchoring in AMD-SP
Gray coloring represents trust anchoring in the vTPM (with no AMD-SP link)
○ : L0—Shallow or no verifiability
◔ : L1—Root of Trust Link Authenticity
◑ : L2—Reference-Based Measurement Matching
◕ : L3—Binary Measurement Reproducibility
● : L4—Source Code Measurement Reproducibility

with Linux VM images, all software components except the firmware are open-source, so, potentially, achieving L4 verifiability depth. Yet, it ultimately relies on the verifiability depth of the firmware. Azure's CVMs do not have their firmware open-sourced; therefore, no software component can achieve a verifiability level deeper than the firmware's. The same occurs with GCP's CVMs. In e-vTPM, the vTPM implementation's trust is anchored at AMD-SP, making any vTPM measurement also anchored at AMD-SP.

4.2 Trusting Cloud Service Providers

Despite the counter-intuitive inclusion of the CSP in the customer's TCBs, best-effort approaches need to leverage vTPMs, even if implemented by the CSP, as there is no better choice. To this end, two notable open-source frameworks are available.

Keylime Keylime [50] is an open-source framework developed at MIT Lincoln Laboratory that automates remote attestation using a TPM 2.0 specification device implementation. The system addresses the fundamental challenge of verifying system integrity across distributed computing environments by providing automated, cryptographically-backed attestation of both boot-time and runtime system state. Although it was first designed to use hardware TPMs, its use can be adapted to systems with vTPM devices, as long as they implement the TPM 2.0 specification.

The framework operates through three primary components, implementing distinct but interconnected functions. The *Keylime Agent* resides on each attested machine, interfacing directly with the TPM 2.0 device to collect measurements and maintain runtime integrity monitoring. Agents operate with minimal privileges and implement secure communication protocols for reporting measurements to remote verifiers.

The *Keylime Verifier* serves as the central attestation authority, evaluating agent trustworthiness based on collected measurements and predefined policies. Verifiers maintain databases of expected measurements, policy definitions, and historical attestation data while implementing algorithms that compare received measurements against known-good baselines, yet still accommodate legitimate system variations, such as system updates.

The *Keylime Registrar* provides secure enrollment and key management services, validating TPM credentials and managing cryptographic keys for secure inter-component communication. The registrar maintains the authoritative registry of participating systems and facilitates the establishment of the initial trust relationship.

Although Keylime offers a production-ready framework for TPM-based remote attestation, when used in cloud environments, it does not recognize or support AMD-SP attestation reports. Therefore, any security property interpreted from Keylime's monitoring relies inherently on the vTPM trust anchor—the CSP. Furthermore, potential issues may arise in the gathering of golden measurements, as their source might be malicious—Keylime does not provide a measurement reproducing mechanism from the source measurable component. Moreover, Keylime does not offer any deployment tools as it assumes the instances are already running and provisioned with the agent (although agent installation tools are available).

Constellation Edgeless Systems[8] develops Constellation [51], a solution to deploy a Kubernetes[9] cluster inside CVMs using AWS, Azure, GCP or STACKIT[10] cloud infrastructure by leveraging CVMs backed by AMD SEV-SNP and Intel TDX. The platform aims to keep all data encrypted and prevent access from the infrastructure layer, effectively removing the CSP from the TCB.

However, achieving complete CSP exclusion from the TCB remains challenging. While Constellation aims to provide a chain of measurements that customers can trace from their expected software to the underlying trusted hardware [51], this level of verifiable attestation that fully excludes CSPs is impossible to achieve with existing cloud infrastructure limitations in AWS, Azure, and GCP. The platform still requires trust in CSPs, particularly regarding firmware components and platform configuration measurements that are not fully verifiable across all cloud providers. Constellation only discloses the measurement unreproducibility in a documentation page [52], without explicitly stating the conclusions of such shortcomings. Through manual analysis of the system's repository [53], Constellation verifiability depth levels were classified as they are presented in Table 3, with significant gaps to what is considered possible in Table 2.

Constellation leverages a custom OS based on Fedora[11]. It includes a bootloader, a unified kernel image, and two crucial partitions. The root filesystem partition operates as read-only with `dm-verity` [54], a Linux kernel module that ensures the integrity of a specified read-only partition, containing base system stateless components, including a bootstrapper agent. The encrypted state partition houses Kubernetes components and the Constellation agent, initialized by the bootstrapper during startup [55].

The platform implements attested TLS (aTLS), integrating attestation statements directly into Transport Layer Security (TLS) handshakes rather than relying solely on traditional certificate authorities. This enables secure interaction with Kubernetes join services but raises questions about verification overhead, as real-time measurement verification may introduce performance implications, not particularly referred to in the tool's documentation [52].

Constellation provides two partition encryption modes addressing different trust models. The constellation-managed mode derives encryption keys from a master secret that can be internally generated or externally sourced. The user-managed mode integrates with external key management systems supporting CSP services and third-party solutions implementing Key Management Interoperability Protocol (KMIP). However, the constellation-managed approach raises questions about master secret generation and storage security [56].

Several implementation aspects challenge Constellation's security claims. AMD SEV-SNP measurement verification lacks transparency regarding reproducibility, making independent validation difficult. The platform's claims of providing fully attestable environments may be overstated given current CSP infrastructure limitations and identified verification gaps. These limitations suggest that while Constellation provides significant security improvements over traditional Kubernetes deployments, achieving complete infrastructure independence remains aspirational rather than fully realized under current cloud computing constraints.

[8] https://www.edgeless.systems/
[9] https://kubernetes.io/
[10] https://www.stackit.de/en/
[11] https://www.fedoraproject.org/

Proprietary Solutions The confidential computing landscape includes several proprietary solutions that make similar security claims to Constellation while operating under identical CSP infrastructure constraints. Anjuna[12] provides CVM deployment across AWS, GCP, and Azure with claims of cryptographic attestation assurance for memory encryption. BeeKeeperAI[13] offers an AI algorithm development acceleration environment using AWS, GCP, and Azure's CVMs, while Decentriq[14] operates a SaaS platform for confidential data collaboration both on-premise and in Azure. Opaque[15] provides a confidential AI platform within AWS, GCP, and AWS's infrastructure with secure attestation mechanisms. All these solutions advertise robust security guarantees while leveraging the same underlying CSP infrastructure that presents the attestation and verification challenges documented throughout this analysis, underscoring the difference between the advertised and effective security guarantees.

The proprietary nature of these solutions presents a fundamental TCB redistribution rather than TCB reduction. Even if these companies have developed techniques to address the inherent limitations of CSP-based confidential computing, their closed-source implementations prevent independent verification of these claims. While memory encryption powered by AMD SEV-SNP hardware may function as advertised, the proving mechanisms remain subject to the same attestation gaps identified in open-source solutions. Memory encryption alone, though reliably attestable in AWS and Google, cannot protect against boot-level attacks that may insert undetected malicious software into CVMs. Such compromises can undermine memory encryption benefits since privileged processes can access other processes' memory in plaintext within the same CVM.

4.3 Open Problems and Limitations

This section overviews the current frontiers of confidential computing research. It begins by acknowledging several other fundamental challenges that, while critical to the field's advancement, lie beyond this thesis's scope. It then contextualizes this thesis's contributions within the existing solution landscape, highlighting the specific challenges being addressed.

Unaddressed Open Problems Several fundamental challenges remain unresolved as CVMs matures toward production deployment:

Availability While confidential computing progressively excludes the CSP from the customer's TCB, no current mechanism prevents hypervisor-initiated denial of service through resource starvation or scheduling manipulation. Although there is no enforcement, Service Level Agreements (SLAs) provides contractual assurance through verifiable availability metrics [57–59]. Availability attacks from both hypervisor and external sources (*e.g.*, Distributed Denial of Service (DDoS)) are considered out-of-scope.

Runtime Integrity Memory encryption alone cannot prevent malicious code execution within CVMs. While Linux's Integrity Monitoring Architecture (IMA) [60] combined with TPM-based attestation, provide runtime software integrity monitoring capabilities, they suffer from Time-of-Check-Time-of-Use (TOCTOU) vulnerabilities [61]. By design, IMA detects only integrity breaches reflected on monitored files, relies on Linux's kernel integrity protections, and requires correct application states to be both predictable and completely measurable. This thesis assumes runtime integrity of trusted components post-boot.

Specialized Workload Performance Workloads requiring accelerator devices, such as GPUs for large machine learning model inference and training, present added challenges. When sensitive data exits the Trusted Execution Environment (TEE) for external processing in such devices, these require dedicated memory protection and secure communication channels with CVMs. NVIDIA addressed this with the H100 architecture's attestable, hardware-backed memory encryption [62]. Performance benchmarks by Zhu et al. [63] demonstrate less than 7% overhead for large language model inference, confirming that GPU memory encryption does not constitute a performance bottleneck. However, confidential accelerator integration with CVMs remains outside this thesis's scope as its availability is limited in public CSPs.

Trustworthy Software The maximum verifiability depth level defined in Section 3.4 ensures only that a specific open-source software component is executing remotely, without addressing software quality or correctness. While open-source software typically enables reproducible builds—essential for remote attestation

[12] https://www.anjuna.io/
[13] https://www.beekeeperai.com/
[14] https://www.decentriq.com/
[15] https://opaque.co/

verification—additional trust-enhancing properties exist beyond this thesis's scope. These include formal verification [64], traceable authorship, comprehensive documentation, independent security audits, and organizational endorsements [65].

Addressed Limitations Beyond these challenges, CVM implementations by CSPs exhibit critical shortcomings in remote attestability, which fundamentally undermine the benefits that would result from resolving any of the out-of-scope problems. Despite attestable completeness seen with e-vTPMs being technically achievable, its realization depends entirely on individual CSP implementation choices. Consequently, a transparency-oriented framework is essential to enable customers to achieve the maximum theoretically possible verifiability depth for software components in practice, thereby allowing them to reliably leverage CVM security benefits.

This thesis addresses the need for a transparency-oriented, usable, and verifier assurance complete framework for customers to securely and knowingly use and benefit from the security benefits CVMs provide.

Table 3: Summary of practical trust models given the verifiability levels on CVM deployments while having the CSP within the TCB

		Trusted CSP								
		Keylime [50]			Constellation [51]			Thesis Target		
		AWS	**Azure**	**GCP**	**AWS**	**Azure**	**GCP**	**AWS**	**Azure**	**GCP**
AMD-SP		–	–	–	✓	✗	✓	✓	✗	✓
vTPM		✓	✓	✓	✓	✓	✓	✓	✓	✓
Firmware	**L1**	●	●	●	●	●	●	●	●	●
	L2	●	●	●	●	●	●	●	●	●
	L3	○	○	○	○	○	○	●	○	●
	L4	○	○	○	○	○	○	●	○	○

✓: Verifiable authenticity ✗: Non-verifiable authenticity –: Not used
○ : Not verifiable
● : Verifiable with vTPM root of trust
● : Verifiable with AMD-SP root of trust

Table 3 presents a contextual comparison of this thesis' proposed framework against currently available open-source solutions. As shown, existing options fail to utilize currently achievable verifiability depth levels with hardware roots of trust, instead relying unnecessarily on CSPs. Presently, beyond firmware, no additional software components can be reliably linked to a hardware root of trust within CSPs infrastructures (Table 2; therefore, this framework does not aim to make novel contributions regarding verifiability depth of additional software components.

The severe lack of transparency in CVM solutions—whether proprietary or open-source—deployed on CSP-owned infrastructure creates a critical vulnerability: customers handling sensitive data may unknowingly trust potentially compromised systems that lack reliable detection mechanisms for security breaches, possibly misguided by the CSP, the intermediary tooling, or the lack of deeper knowledge on AMD SEV-SNP technology. This framework proposal addresses this fundamental problem by elevating CVM adoption through transparent tooling that explicitly communicates both security guarantees and their inherent limitations. By doing so, it enables organizations to make informed decisions about appropriate use cases for CVMs while providing accessible pathways to achieve maximum possible verifiability depth across all trusted system components. From a commercial perspective, this transparency-driven approach hopefully creates a positive feedback loop wherein increased product adoption drives further investment and solution improvements, as heightened public attention to CVM technology use cases naturally leads to greater scrutiny of CSPs shortcomings and corresponding improvements.

5 Architecture

As discussed in earlier sections, while AMD SEV-SNP technology offers strong hardware-enforced security guarantees for CVMs, realizing its full potential remains difficult in current public cloud environments. The major CSPs—AWS, Azure, and GCP—support CVMs to varying extents, but each exposes different trust models, attestation mechanisms, and configuration limitations. These inconsistencies present a significant barrier to adoption, especially for developers and system administrators who require reproducible, verifiable, and user-friendly workflows.

To encourage broader and more effective use of CVMs, this thesis proposes the design and implementation of a framework that simplifies secure CVM usage across public cloud infrastructure. The proposed system aims to abstract over provider-specific interfaces and limitations, providing a unified and extensible interface to deploy, provision, attest, and interact with CVMs in a secure, transparent, and reproducible manner.

5.1 Design Goals

The design of the framework is guided by the following goals:

- **Security-Awareness**: Provide the best-effort trust guarantees available on each CSP by leveraging supported features, while documenting any limitations or gaps transparently.
- **Modularity and Extensibility**: Support the addition of new CSPs, roots of trust, attestation procedures, and workload types without significant reengineering.
- **Automation**: Reduce operational overhead by automating image preparation, deployment, and attestation steps.
- **Flexibility**: Enable deployment scenarios where the guest VM owner is distinct from the workload owner, and mutually distrusting relying parties.
- **Reproducibility**: Ensure that deployments and their associated trust evidence can be reproduced and audited using declarative and version-controlled inputs.

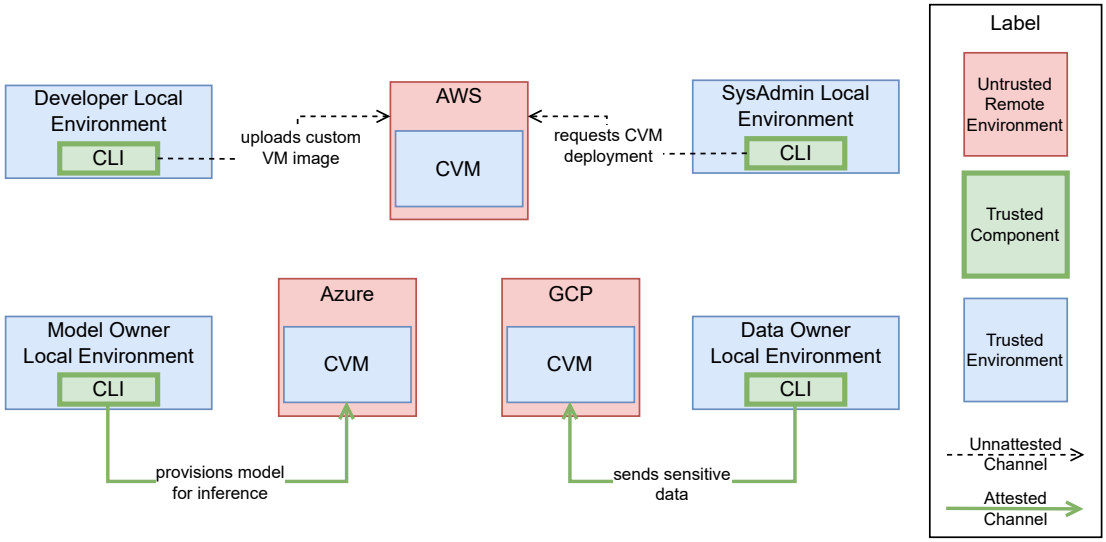
5.2 Framework Overview

The goal of the proposed framework is to empower users to securely deploy and interact with CVMs without needing to manually manage CSP-specific caveats. At its core, the framework abstracts over provider-specific interfaces and limitations, offering a unified and extensible interface to deploy, provision, attest, and interact with CVMs in a secure, transparent, and reproducible manner. Figure 4 provides an overview of the framework context. Although the framework allows simpler scenarios, this one is complex enough to cover all of its expected functionalities. In this scenario, two distinct, mutually *trusting*, sensitive information owners want to collaborate in confidential AI inference. For this, the Data Owner does not want to share confidential data with any party but the Model Owner (and trusted components). As for the Model Owner, they don't want to share their proprietary machine learning model with any party, except for trusted components. Both need to ensure that the trusted components they expect are the actual components that are running inside the CVM they will interact with. For this, they will leverage remote attestation procedures. The specific channel instantiation depicted in Figure 4 is exemplary; it does not represent CSP-specific protocols or limitations.

Comparing with the Remote ATtestation procedureS (RATS) roles described in Section 3.4, a simplified view of it could be that both the Model Owner and the Data Owner are simultaneously Relying Parties and Verifiers (for their own), while the CVMs are the Attesters.

However, leveraging Cloud Service Providers infrastructure to run Confidential Virtual Machines requires some setup: as intended, the Developer will use the framework's Command Line Interface (CLI) to assist in uploading a custom VM image to be launched as a CVM. The System Administrator (SysAdmin) will use the CLI to request a specified CVM deployment in any CSP, with the sole focus on each one's achievable trust models—changing the deployment target between CSPs should be as easy as modifying an environment variable. Note that both the Developer and the SysAdmin do not interact directly with the CVM, but rather with the CSP

Figure 4: Framework Context Diagram



through their defined APIs. In this scenario, for simplicity, the SysAdmin assumes the CVM launches correctly and does not need to perform a sanitary remote attestation.

Once a CVM is running, it should be immediately available for an attested channel establishment, which is part of the expected behavior by the framework's custom VMs. If successful, the Model Owner may safely provide its proprietary model, and then the Data Owner inputs their sensitive data and gets the inference result back. During this use case, no confidential information was leaked as they were transferred upon establishment of an attested channel—a channel which depends on a successful evidence verification for cryptographic relevance and its included measurements.

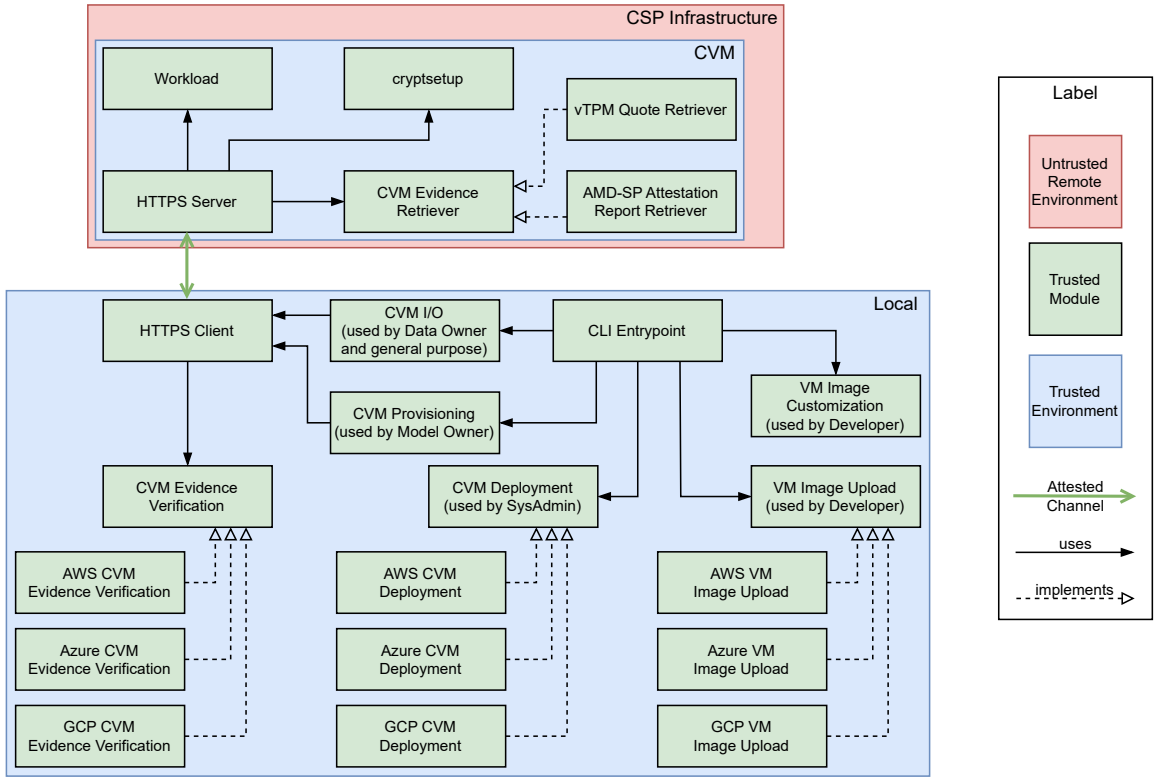
Figure 5 offers a module view of the trusted components composing parts. The CLI may accommodate a variety of use cases as it contains all the necessary functionalities for a user to use CVMs in any desired way. As per my design goals, CSP specific requirements are abstracted, and the architecture is resilient to the addition of new CSPs. To support new roots of trust, a new module needs to be developed to retrieve the new pieces of evidence, and the implementations of 'CVM Evidence Verification' need to be modified to support a meaningful new evidence verification.

The 'HTTPS Client' is responsible for retrieving the evidence from the 'HTTPS Server', verifying it, and then proceeding with the TLS handshake. Only then, a secure channel to an attested CVM is ensured. The workload module in Figure 5 is marked as a trusted module. This is only true when the workload is present at the CVM boot time. If the workload is provisioned after the fact, then another measurement techniques need to take place in order to make the workload a trusted module. This is what happens when the Model Owner provisions their model—it was only allowed in the exemplification scenario, since the Data Owner trusted the Model Owner beforehand.

To guarantee reproducibility and reduce human error, configurable options prioritize the definition of one or more configuration files, letting users specify all deployment and security parameters in a structured, predictable way. This approach ensures that every step—from image hardening to attestation checks—can be replayed and audited using version-controlled inputs.

Finally, the framework will provide use-case templates that demonstrate common deployment scenarios, such as the one depicted in Figure 5, or multi-node Kubernetes clusters inside CVMs. These templates serve both as practical examples and as starting points for users to adapt this flexible framework to their own requirements.

Figure 5: Framework Module Diagram



5.3 Anticipated Benefits

By abstracting away the complexity of dealing with inconsistent CSP support and tooling, this framework aims to lower the barrier to entry for confidential computing. It also allows users to make informed decisions by explicitly exposing the security guarantees, along with their limitations, associated with each deployment. Furthermore, by serving as an open and auditable reference implementation, the framework contributes to a more trustworthy and transparent CVM ecosystem.

The anticipated outcomes include increased visibility into real-world CVM usage patterns, empirical assessment of current trust models, and stronger incentives for CSPs to improve their confidential computing security guarantees. Ultimately, this work will facilitate the secure and reproducible use of CVMs in the cloud, aligning with both academic and industry goals for privacy-preserving computing.

6 Evaluation

The evaluation of the proposed framework will focus on validating its effectiveness across three dimensions: performance, cost and resource overhead, and comparative advantages over existing tools and plain scripts. These evaluation criteria are aligned with the design goals presented in Section 5.1, namely automation, reproducibility, security-awareness, and transparency.

Since the framework is still under development, this section defines the evaluation methodology to be used upon completion, identifying the metrics and benchmarks that will be collected to assess the quality and practicality of the solution.

6.1 Performance Evaluation and Baseline Comparison

A central objective is to ensure that the framework's automation does not introduce unacceptable delays. To measure this, the elapsed time for each automated workflow in a typical CVM deployment will be recorded. Specifically, VM image customization will be timed from the moment a user submits a configuration file until the hardened VM image is fully generated and uploaded to each CSP. This is expected to take a longer amount of time, but this shall only be repeated upon custom VM image updates. The deployment latency will be measured from the API call to instantiate a CVM until the instance becomes reachable over the network. Next, startup time will be measured as the duration required for initial bootstrapping tasks—such as agent installation—to complete. Lastly, the remote attestation duration will be measured from the request for evidence to the final measurement verification. All these times will be compared to shell scripts that reflect the recommended way to achieve the same goals, according to each CSP official documentation.

6.2 Cost and Resource Overhead

To understand the trade-offs introduced by the framework, its impact will be quantified on both the TCB and the runtime performance of CVMs. First, the number of Lines of Code (LOC) added to the TCB by the framework—both in its core logic and in CSP-specific adapters—will be counted. Since minimizing the TCB is a key security objective, this metric will indicate how much new trusted logic is introduced.

Second, the resource cost of attestation and secure interaction mechanisms will be evaluated from two perspectives. The duration for which a VM remains active but idle during provisioning or boot attestation will be measured; this "idle window" represents resource waste and can translate directly into additional CSP charges. Also, CPU and memory utilization will be profiled on a running CVM to quantify the overhead imposed by the CVM resident utilities and by secure communication protocols. These measurements will be taken on each supported CSP to highlight any provider-specific performance bottlenecks.

6.3 Comparison with Existing Solutions

To place the framework's benefits in context, a comparative evaluation against two representative tools, Constellation and Keylime, will be performed. Constellation will be used to benchmark deployment duration, security transparency, and architectural flexibility. First, a baseline Constellation workflow to deploy and attest a multi-node Kubernetes cluster inside CVMs will be reproduced. From this experiment, metrics such as end-to-end deployment time and the number of manual configuration steps required will be extracted. The user-facing remote attestation feedback will be assessed, noting where Constellation documents or omits details related to firmware measurement and PCR verification.

Next, these results will be compared to the same workflow executed through the proposed framework. In particular, the differences in the time spent on reproducible measurement checks, the clarity of trust assumptions, and the ease of adapting to different use cases, such as third-party AI model owners or custom image hardening policies, will be highlighted.

Keylime serves as a second point of comparison, focusing on TPM-based attestation. The comparison will focus on the remote attestation performance, resource overhead, the verifiability level reached, and practical limitations in the deployment of multi-cloud environments.

7 Scheduling of Future Work

Future work is scheduled according to Table 4. First, cloud-specific attestation mechanisms will be implemented, accompanied and verified by empirical research. The first version of the framework is considered completed once deployment and provisioning tooling are implemented. As the research evolves, discoveries and technical content will be documented in the dissertation. The framework is expected to be continuously enhanced as the evaluation reveals critical improvement points.

8 Conclusions

This thesis addresses the gap between AMD SEV-SNP's security promises and the reality of CVM deployments in public clouds. While the technology enables hardware-backed memory encryption and integrity protection, AWS, Azure, and GCP implementations suffer from opaque attestation mechanisms, inconsistent trust models, and limited transparency. Existing tools like Keylime [50] and Constellation [51] fail to fully leverage available security features, often not completely exploring AMD-SP rooted security guarantees. This work aims to enable secure processing of sensitive workloads, such as confidential AI inference, with verifiable trust guarantees.

The proposed framework provides a unified solution through three contributions: comprehensive analysis of CSP trust models, automated CVM lifecycle management tooling, and practical confidential AI deployments. By abstracting CSP's complexities and maximizing verifiability depth, the framework enables informed decision-making while achieving the best possible security guarantees within current infrastructure constraints. This transparency-driven approach facilitates immediate CVM adoption and, consequently, pressures CSPs to improve their offerings, advancing the ecosystem toward more trustworthy confidential computing.

Acknowledgments

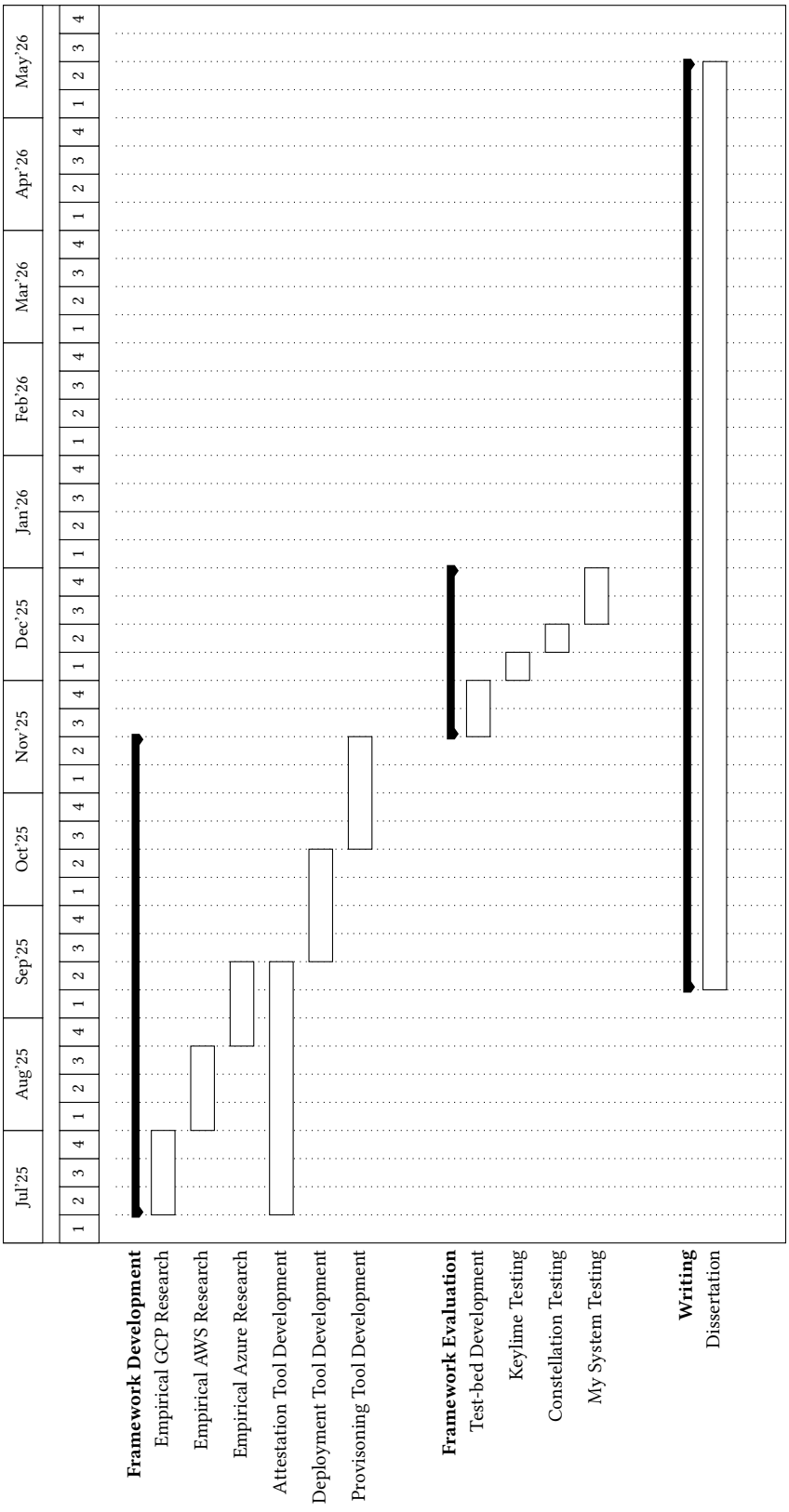
I thank my advisors, Professor Luís Rodrigues and Professor Nuno Santos, for their insightful discussions, guidance, and support. I also thank Daniel Castro for his exceptional leadership by example and invaluable assistance. This project has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101189689 and the Fundação para a Ciência e Tecnologia (FCT) under grant UIDB/50021/2020.

Acronyms

ACPI Advanced Configuration and Power Interface
AES Advanced Encryption Standard
AK Attestation Key
AMD-SP AMD–Secure Processor
ARK AMD Root Key
ASK AMD SEV Key
AWS Amazon Web Services
CLI Command Line Interface
CRTM Core Root of Trust for Measurement
CSP Cloud Service Provider
CVM Confidential Virtual Machine
DDoS Distributed Denial of Service
DRTM Dynamic Root of Trust for Measurement
EC2 Elastic Compute 2
EK Endorsement Key
GCE Google Compute Engine
GCP Google Cloud Platform
gPA guest physical address
HCL Hardware Compatibility Layer
HTTPS Hyper Text Transfer Protocol Secure
IETF Internet Engineering Task Force
IMA Integrity Monitoring Architecture
KMIP Key Management Interoperability Protocol
LD Launch Digest
LOC Lines of Code

OVMF Open Virtual Machine Firmware
PCR Platform Configuration Register
RATS Remote ATtestation procedureS
RFC Request for Comments
RMP Reverse Map Table
SEV Secure Encrypted Virtualization
SEV-ES Secure Encrypted Virtualization–Encrypted State
SEV-SNP Secure Encrypted Virtualization–Secure Nested Paging
SGX Software Guard Extensions
SLA Service Level Agreement
sPA system physical address
SRTM Static Root of Trust for Measurement
SVSM Secure VM Service Module
TCB Trusted Computing Base
TDX Trusted Domain Extensions
TEE Trusted Execution Environment
TLS Transport Layer Security
TOCTOU Time-of-Check-Time-of-Use
TPM Trusted Platform Module
UEFI Unified Extensible Firmware Interface
VCEK Versioned Chip Endorsement Key
vCPU virtual CPU
VM Virtual Machine
VMPL Virtual Machine Privilege Level
VMSA Virtual Machine Save Area
vTPM virtual Trusted Platform Module

Table 4: Work Schedule



Bibliography

[1] Confidential Computing Consortium, "Confidential computing: Hardware-based trusted execution for applications and data," November 2022, retrieved on May 2nd, 2025. [Online]. Available: https://confidentialcomputing.io/wp-content/uploads/sites/10/2023/03/CCC_outreach_whitepaper_updated_November_2022.pdf

[2] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, ser. STOC '09. New York, NY, USA: Association for Computing Machinery, 2009, pp. 169–178. [Online]. Available: https://doi.org/10.1145/1536414.1536440

[3] A. C. Yao, "Protocols for secure computations," in *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, 1982, pp. 160–164.

[4] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, ser. CCSW '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 113–124. [Online]. Available: https://doi.org/10.1145/2046660.2046682

[5] C. Moore, M. O'Neill, E. O'Sullivan, Y. Doröz, and B. Sunar, "Practical homomorphic encryption: A survey," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 2792–2795.

[6] R. Podschwadt, D. Takabi, P. Hu, M. H. Rafiei, and Z. Cai, "A Survey of Deep Learning Architectures for Privacy-Preserving Machine Learning With Fully Homomorphic Encryption," *IEEE Access*, vol. 10, pp. 117 477–117 500, 2022.

[7] A. Akram, A. Giannakou, V. Akella, J. Lowe-Power, and S. Peisert, "Performance Analysis of Scientific Computing Workloads on General Purpose TEEs," in *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2021, pp. 1066–1076.

[8] L. Jiang and L. Ju, "FHEBench: Benchmarking Fully Homomorphic Encryption Schemes," 2022. [Online]. Available: https://arxiv.org/abs/2203.00728

[9] V. Costan and S. Devadas, "Intel SGX Explained," Cryptology ePrint Archive, Paper 2016/086, 2016. [Online]. Available: https://eprint.iacr.org/2016/086

[10] S. Chakrabarti, R. Leslie-Hurd, M. Vij, F. McKeen, C. Rozas, D. Caspi, I. Alexandrovich, and I. Anati, "Intel® Software Guard Extensions (Intel® SGX) Architecture for Oversubscription of Secure Memory in a Virtualized Environment," in *Proceedings of the Hardware and Architectural Support for Security and Privacy*, ser. HASP '17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3092627.3092634

[11] P.-C. Cheng, W. Ozga, E. Valdez, S. Ahmed, Z. Gu, H. Jamjoom, H. Franke, and J. Bottomley, "Intel TDX Demystified: A Top-Down Approach," 2023. [Online]. Available: https://arxiv.org/abs/2303.15540

[12] Advanced Micro Devices, Inc., "AMD SEV-SNP: Strengthening VM isolation with integrity protection and more," Advanced Micro Devices, Inc., Tech. Rep., 2020, retrieved on March 24th, 2025. [Online]. Available: 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

[13] M. Misono, D. Stavrakakis, N. Santos, and P. Bhatotia, "Confidential VMs Explained: An Empirical Analysis of AMD SEV-SNP and Intel TDX," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 8, no. 3, Dec. 2024. [Online]. Available: https://doi.org/10.1145/3700418

[14] F. Mo, Z. Tarkhani, and H. Haddadi, "Machine learning with confidential computing: A systematization of knowledge," *ACM Comput. Surv.*, vol. 56, no. 11, Jun. 2024. [Online]. Available: https://doi.org/10.1145/3670007

[15] Advanced Micro Devices, Inc., "AMD Memory Encryption," Advanced Micro Devices, Inc., Tech. Rep., 2016, retrieved on April 11th, 2025. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/memory-encryption-white-paper.pdf

[16] ——, "Protecting VM register state with SEV-ES," Advanced Micro Devices, Inc., Tech. Rep., 2017, retrieved on April 23rd, 2025. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/Protecting-VM-Register-State-with-SEV-ES.pdf

[17] J. Eisoldt, A. Galanou, A. Ruzhanskiy, N. Küchenmeister, Y. Baburkin, T. Dai, I. Gudymenko, S. Köpsell, and R. Kapitza, "SoK: A cloudy view on trust relationships of CVMs – How Confidential Virtual Machines are falling short in Public Cloud," 2025. [Online]. Available: https://arxiv.org/abs/2503.08256

[18] Trusted Computing Group, "Trusted Platform Module (TPM) Summary," Trusted Computing Group, April 2008, retrieved on May 29th, 2025. [Online]. Available: https://trustedcomputinggroup.org/resource/trusted-platform-module-tpm-summary/

[19] W. Arthur, D. Challener, and K. Goldman, *A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security*, 1st ed. USA: Apress, 2015. [Online]. Available: https://doi.org/10.1007/978-1-4302-6584-9

[20] *TCG PC Client Specific TPM Interface Specification*, Trusted Computing Group, March 2013, official specification. Version 1.3. Retrieved on May 29th, 2025. [Online]. Available: https://trustedcomputinggroup.org/resource/pc-client-work-group-pc-client-specific-tpm-interface-specification-tis/

[21] S. Berger, R. Caceres, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn, "vTPM: Virtualizing the Trusted Platform Module," in *15th USENIX Security Symposium (USENIX Security 06)*. Vancouver, B.C. Canada: USENIX Association, Jul. 2006. [Online]. Available: https://www.usenix.org/conference/15th-usenix-security-symposium/vtpm-virtualizing-trusted-platform-module

[22] *QEMU vTPM Specification*, QEMU Project, official documentation. Accessed on May 29th, 2025. [Online]. Available: https://qemu-project.gitlab.io/qemu/specs/tpm.html

[23] VMware, "vSphere Virtual TPM (vTPM) Questions and Answers," VMware, retrieved on May 29th, 2025. [Online]. Available: https://www.vmware.com/docs/vsphere-virtual-tpm-vtpm-questions-answers

[24] TianoCore Community, "OVMF," TianoCore Project, accessed on March 27th, 2025. [Online]. Available: https://github.com/tianocore/tianocore.github.io/wiki/OVMF

[25] Advanced Micro Devices, Inc., *SEV Secure Nested Paging Firmware ABI specification*, Advanced Micro Devices, Inc., May 2025, official specification. Revision 1.58. Retrieved on March 24th, 2025. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/epyc-technical-docs/specifications/56860.pdf

[26] TianoCore Project, "TCG trusted boot chain in EDKII," accessed on May 25th, 2025. [Online]. Available: https://tianocore-docs.github.io/edk2-TrustedBootChain/release-1.00/3_TCG_Trusted_Boot_Chain_in_EDKII.html

[27] H. Birkholz, N. Smith, D. Thaler, L. Lundblade, and M. Sethi, "Remote attestation procedures architecture," 2023, rFC 9334. Accessed on May 10th, 2025. [Online]. Available: https://datatracker.ietf.org/doc/rfc9334/

[28] G. Scopelliti, C. Baumann, and J. T. Mühlberg, "Understanding Trust Relationships in Cloud-Based Confidential Computing," in *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2024, pp. 169–176.

[29] *Amazon Machine Images - Amazon EC2*, Amazon Web Services, official documentation. Accessed on March 26th, 2025. [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html

[30] *Import your VM as an image - VM Import/Export*, Amazon Web Services, official documentation. Accessed on March 27th, 2025. [Online]. Available: https://docs.aws.amazon.com/vm-import/latest/userguide/import-vm-image.html

[31] *Create an Azure virtual machine offer on Azure Marketplace using your own image*, Microsoft Corportation, accessed on March 31st, 2025. [Online]. Available: https://learn.microsoft.com/en-us/partner-center/marketplace-offers/azure-vm-use-own-image

[32] *Manually import boot disks - Compute Engine Documentation*, Google Cloud, accessed on March 31st, 2025. [Online]. Available: https://cloud.google.com/compute/docs/import/import-existing-image

[33] Amazon Web Services, "AWS UEFI firmware for AMD SEV-SNP instances," accessed on March 26th, 2025. [Online]. Available: https://github.com/aws/uefi

[34] Virtee, "sev-snp-measure: Calculate AMD SEV/SEV-ES/SEV-SNP measurement for confidential computing," accessed on March 25th, 2025. [Online]. Available: https://github.com/virtee/sev-snp-measure

[35] Google Cloud, "Verify a Confidential VM instance's firmware," accessed on March 31st, 2025. [Online]. Available: https://cloud.google.com/confidential-computing/confidential-vm/docs/verify-firmware

[36] Google, "gce-tcb-verifier: Reference code for creating and verifying a GCE firmware signed reference value message," accessed on March 31st, 2025. [Online]. Available: https://github.com/google/gce-tcb-verifier

[37] Amazon Web Services, "The Security Design of the AWS Nitro System," Amazon Web Services, Inc., Tech. Rep., February 2024, retrieved on May 25th, 2025. [Online]. Available: https://docs.aws.amazon.com/pdfs/whitepapers/latest/security-design-of-aws-nitro-system/security-design-of-aws-nitro-system.pdf

[38] Microsoft Corporation, "Virtual TPMs in Azure confidential VMs," accessed on March 28th, 2025. [Online]. Available: https://learn.microsoft.com/en-us/azure/confidential-computing/virtual-tpms-in-azure-confidential-vm

[39] Google Cloud, "Confidential Space security overview," accessed on June 15th, 2025. [Online]. Available: https://cloud.google.com/docs/security/confidential-space

[40] Microsoft Azure, "Confidential VM Platform Guest Attestation Sample Apps – CVM Guest Attestation," accessed on March 28th, 2025. [Online]. Available: https://github.com/Azure/confidential-computing-cvm-guest-attestation/blob/main/cvm-guest-attestation.md

[41] NCC Group, "Public Report – AWS Nitro System API & Security Claims," NCC Group, Tech. Rep., April 2023, architecture review of AWS Nitro System design conducted in Q4 2022. Retrieved on May 25th, 2025. [Online]. Available: https://www.nccgroup.com/us/research-blog/public-report-aws-nitro-system-api-security-claims/

[42] H. Pulapaka and Marysia, "OpenHCL: Evolving Azure's virtualization model," September 2024, microsoft Tech Community, Updated Sep 18, 2024. Accessed on May 26th, 2025. [Online]. Available: https://techcommunity.microsoft.com/blog/windowsosplatform/openhcl-evolving-azure%E2%80%99s-virtualization-model/4248345

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

[44] QEMU Project, "Direct Linux Boot," qEMU Documentation. Accessed on June 3rd, 2025. [Online]. Available: https://qemu-project.gitlab.io/qemu/system/linuxboot.html

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

[46] QEMU Project, "microvm virtual platform," accessed on June 3rd, 2025. [Online]. Available: https://www.qemu.org/docs/master/system/i386/microvm.html

[47] Advanced Micro Devices, "AMD Secure VM Service Module (SVSM) Specification," AMD, Tech. Rep., July 2023, official specification. Revision 1.00. Retrieved on March 27th, 2025. [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/epyc-technical-docs/specifications/58019.pdf

[48] Coconut SVSM Contributors, "Coconut Secure Virtual Machine Service Module (SVSM)," accessed on March 27th, 2025. [Online]. Available: https://github.com/coconut-svsm/svsm

[49] V. Narayanan, C. Carvalho, A. Ruocco, G. Almasi, J. Bottomley, M. Ye, T. Feldman-Fitzthum, D. Buono, H. Franke, and A. Burtsev, "Remote attestation of confidential VMs using ephemeral vTPMs," in *Proceedings of the 39th Annual Computer Security Applications Conference*, ser. ACSAC '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 732–743. [Online]. Available: https://doi.org/10.1145/3627106.3627112

[50] N. Schear, P. T. Cable, T. M. Moyer, B. Richard, and R. Rudd, "Bootstrapping and maintaining trust in the cloud," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, ser. ACSAC '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 65–77. [Online]. Available: https://doi.org/10.1145/2991079.2991104

[51] Edgeless Systems, "Constellation - The Kubernetes Built for Confidential Computing," accessed on May 9th, 2025. [Online]. Available: https://www.edgeless.systems/products/constellation

[52] ——, "Constellation's attestation process," accessed on March 30th, 2025. [Online]. Available: https://docs.edgeless.systems/constellation/architecture/attestation

[53] ——, "Constellation: The first Confidential Kubernetes," accessed on April 30th, 2025. [Online]. Available: https://github.com/edgelesssys/constellation

[54] Linux Kernel Documentation, "dm-verity," accessed on June 15th, 2025. [Online]. Available: https://docs.kernel.org/admin-guide/device-mapper/verity.html

[55] Edgeless Systems, "Constellation's VM images," accessed on March 30th, 2025. [Online]. Available: https://docs.edgeless.systems/constellation/architecture/images

[56] ——, "Constellation's key management and cryptographic primitives," accessed on June 3rd, 2025. [Online]. Available: https://docs.edgeless.systems/constellation/architecture/keys

[57] Amazon Web Services, "Service Level Agreements," accessed on June 15th, 2025. [Online]. Available: https://aws.amazon.com/legal/service-level-agreements

[58] Google Cloud, "Google Compute Engine Service Level [agreement]," accessed on June 15th, 2025. [Online]. Available: https://cloud.google.com/compute/sla

[59] Microsoft Corporation, "Service Level Agreements (SLA) for Online Services," accessed on June 15th, 2025. [Online]. Available: https://www.microsoft.com/licensing/docs/view/Service-Level-Agreements-SLA-for-Online-Services

[60] R. Sailer, X. Zhang, T. Jaeger, and L. v. Doorn, "Design and Implementation of a TCG-based Integrity Measurement Architecture," in *USENIX Security Symposium*, vol. 13, no. 2004, 2004, pp. 223–238.

[61] S. Bratus, N. D'Cunha, E. Sparks, and S. W. Smith, "TOCTOU, traps, and trusted computing," in *Trusted computing - challenges and applications*, P. Lipp, A.-R. Sadeghi, and K.-M. Koch, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 14–32.

[62] G. Dhanuskodi, S. Guha, V. Krishnan, A. Manjunatha, M. O'Connor, R. Nertney, and P. Rogers, "Creating the First Confidential GPUs: The team at NVIDIA brings confidentiality and integrity to user code and data for accelerated computing." *Queue*, vol. 21, no. 4, pp. 68–93, Sep. 2023. [Online]. Available: https://doi.org/10.1145/3623393.3623391

[63] J. Zhu, H. Yin, P. Deng, A. Almeida, and S. Zhou, "Confidential Computing on NVIDIA Hopper GPUs: A Performance Benchmark Study," 2024. [Online]. Available: https://arxiv.org/abs/2409.03992

[64] P. Paradžik, A. Derek, and M. Horvat, "Formal Security Analysis of the AMD SEV-SNP Software Interface," 2025. [Online]. Available: https://arxiv.org/abs/2403.10296

[65] E. Brito, F. Castillo, P. Pullonen-Raudvere, and S. Werner, "TrustOps: Continuously building trustworthy software," in *Enterprise design, operations, and computing. EDOC 2024 workshops*, M. Kaczmarek-Heß, K. Rosenthal, M. Suchánek, M. M. Da Silva, H. A. Proper, and M. Schnellmann, Eds. Cham: Springer Nature Switzerland, 2025, pp. 53–67.