Proving Attributes about Confidential Compute Services with Validation and Endorsement Services

Anjo Vahldiek-Oberwagner
Intel Labs
Berlin, Germany
anjovahldiek@gmail.com

Marcela S. Melara
Intel Labs
Hillsboro, OR, USA
marcela.melara@intel.com

***Abstract*—We propose validation and endorsement services (VES), an abstract architecture for establishing trust in confidential compute services (CCS), such as confidential AI applications, that address two main challenges in current remote attestation frameworks. First, we form a hierarchy of VESes, enabling *in-band* validation of CCS workload attributes, which delegates trust to the VESes. Upon validation, VESes efficiently store endorsements for future lookups about a CCS. Second, VESes facilitate the dynamic discovery of verified CCSes that meet specified attributes. Our abstraction ultimately aims to enhance the flexibility and trustworthiness of confidential compute deployments.**

***Index Terms*—Confidential Computing, Trusted Execution Environments, Attestation**

1. Motivation and Problem Statement

Hardware-based trusted execution environments (TEEs) are commonly used in confidential cloud computing deployments like confidential AI [12]. A core feature of TEEs is remote attestation, which authenticates the initial state of the TEE. The attestation is typically over one or more secure hashes [5], [6] of the memory footprint, and digitally signed with a device-specific cryptographic key. In the case of confidential VMs (cVMs), the attestation process involves initializing the Open Virtual Machine Firmware (OVMF) and performing trusted boot with runtime measurement registers (RTMRs) [6] or a virtual trusted platform module (TPM). These mechanisms combined allow remote parties to ascertain that the desired TEE was created and expected code was successfully loaded and started [2]. Unfortunately, cVM properties like memory size and other sometimes proprietary information (like the OVMF image hash) may overwrite the attestation hashes and make them unusable for third parties. Cloud providers have sought to address this issue by publishing so-called golden values (i.e., known good hash values) as a baseline for validating cVM attestations [1], [4].

However, the main challenge with hash-based attestation in TEEs today is that the meaning and intended guarantees that these secure hashes represent must be established separately. For example, new standards like the healthcare confidential compute specification [3] include strong demands for demonstrable properties, such as the link between source code and attested executable hash values at runtime, to validate the origin of code running. Other desirable workload attributes include vetted golden values of a TEE or other system components, the geographical location of a TEE, secure binary compilation options (e.g., compiled with CET), controlled data sources or network connections, or build pipeline restrictions. But the semantics of the hash values representing these attributes are often lost at runtime, especially during software updates or changes to the runtime environment.

The current state of the art identifies workload attributes by tracking runtime information [13], [15], allowing a runtime attestation value to be traced back, for example, to the initial source code, AI model, or training data. Yet, many attributes still need to be established at build time or cannot be easily determined dynamically at execution time. This is a significant drawback for the dynamic nature of cloud deployments because it hinders application vendors' ability to demonstrate specific attributes prior to deployment.

An alternative approach for validating the integrity of a TEE relies on reproducible builds to exactly recreate a TEE [7]. Reproducibility, though, requires access to source code and build configuration, which may be proprietary, needs a separate authenticity mechanism, and scales poorly when interested parties need to establish trust in the TEE dynamically at runtime [9].

Because of the challenges with using hash-based TEE attestation or reproducible builds in dynamic cloud deployments, users of confidential compute services may be unable to ascertain workload attributes, even when they are present. The goal of this paper is to overcome these limitations with validation and endorsement services (VES) that validate TEE attributes. We additionally address the gap of discovering trustworthy VESes and validated confidential computing services.

2. Design Overview

Figure 1 illustrates VES components and their interactions. As a pre-requisite, CCS ABC registers its TEE attestation, which serves as a unique identity, and service capabilities with a CCS registry. This registry is the central hub for CCS users to discover CCSes and their *endorsed* attributes. To create an endorsement, a CCS first locates VESes with specific capabilities in the VES registry. A VES can then validate the attributes of the CCS by consulting an endorsement cache or by directly interacting with the CCS. If the validation passes, a new endorsement is created and stored in the endorsement cache.

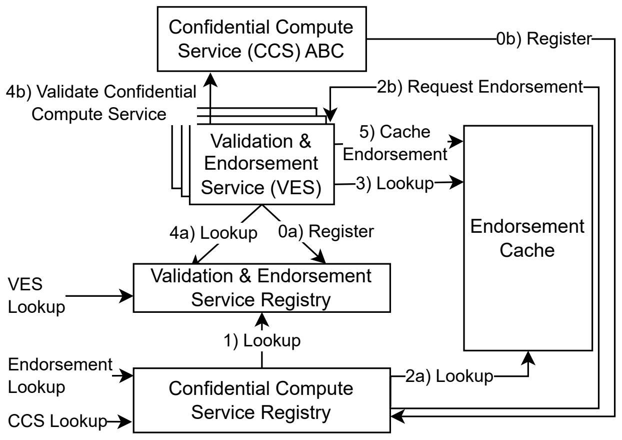
Confidential Compute Service (CCS) ABC
0b) Register
4b) Validate Confidential Compute Service
2b) Request Endorsement
Validation & Endorsement Service (VES)
5) Cache Endorsement
3) Lookup
Endorsement Cache
4a) Lookup
0a) Register
VES Lookup
Validation & Endorsement Service Registry
1) Lookup
Endorsement Lookup
CCS Lookup
Confidential Compute Service Registry
2a) Lookup

Figure 1. Overview of Endorsement Services Architecture

An essential VES in our architecture is the *root VES* from which we bootstrap trust by endorsing attributes about other VESes (much how certificate authorities are trusted today). That is, our architecture assumes VES themselves run in TEEs and allows VESes to treat other VESes as CCSes. Thus, the root VES is responsible for performing the remote attestation with any CCS given different TEE-specific protocols. To gain trust in any root VES, the entity who operates it must submit its executable golden values and a certificate generated during the root VES' build process into an endorsement cache. This separation allows us to cryptographically bind the root VES' attestation hash to its source code, and is required to avoid infinite endorsement chains. This approach also improves upon reproducible builds by addressing the non-deterministic nature of frequently-updated TEE attestation values and scalability challenges.

2.1. Validation & Endorsement Services (VES)

VESes provide endorsements that are cryptographically signed with the VES' identity. The attributes a VES endorses are varied and depend on the VES' own endorsed capabilities. We consider several VESes in practice:

TEE Golden Values: Given the complexity of TEE attestation contents, this VES determines at runtime the expected attestation value depending on deployment specific parameters (e.g., VM size) or the deployed TEE host system, which determines ordering of hash computations. This VES makes it easier to work with TEE attestations, since CCS users may not know golden values a priori.

Supply Chain: Given a CCS' runtime hash, the VES provides authenticated supply chain metadata such a Software Bills of Materials (SBOM) [10] or build metadata for the CCS [14]. This VES could rely on a transparency log [11] for tamper-evident endorsement caching.

Secure Compilation: Using the Supply Chain VES, this VES examines finer grained compile-time attributes [8], for instance, to determine if the compilation used security related flags like control flow integrity or address sanitization.

Geographical Location: Legislation may limit data movement across legislative boundaries (e.g., EU borders in GDPR). This VES performs several ping experiments with the CCS in question to determine its location and offer a coarse endorsement in case it is located within the requested legislative boundary. This process requires careful consideration of data privacy policies, network latency and routing paths to ensure accuracy.

Input Limitation: CCSes may restrict their storage and networking capabilities to improve their security stance. This VES checks whether runtime parameters reflect the restriction.

To discover any of these VESes with the expected security stance CCS users may query the VES registry. The query requires both the VES' capabilities and desired attributes. Ideally, the VES registry is decentralized to avoid a single point of failure and a single controlling entity to avoid availability issues.

2.2. Discovering CCSes with Specified Attributes

The final step in our proposed architecture is the discovery of CCes that meet specified attributes. This process allows users to find services that have been endorsed by trusted parties for the attributes they require. The discovery mechanism should be user-friendly, efficient, and capable of handling a large number of services.

To achieve this, we propose a CCS registry. The registry not only accepts CCS registrations, but it is also responsible for requesting CCS endorsements from VESes if a user looks up a CCS, but no registered CCS has sufficient endorsements. In this case, the CCS registry searches for a suitable VES and requests that a CCS that might fit the user's request be validated. Upon completion and storing endorsements in the cache, the registry may return CCSes that suit the user's request.

3. Conclusion

In this paper, we present an emerging approach to enhancing the security and trustworthiness of confidential compute services (CCS) through the use of validation and endorsement services (VES). Our proposed architecture includes mechanisms for discovering and establishing trust in VESes, using VESes to validate CCS attributes, and discovering CCSes with specified attributes.

Future work will focus on implementing and evaluating the proposed architecture in real-world cloud environments. We plan to explore additional use cases, attributes and further refine the validation, endorsement and discovery mechanisms to improve their efficiency and security.

Overall, our approach aims to bridge the gap between the need for dynamic TEE workload attributes and the current limitations of hash-based attestations or reproducible builds, providing a more flexible and trustworthy solution for confidential compute deployments.

Acknowledgements

Parts of this paper were generated or writing refined using Github Copilot.

References

[1] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan. RFC 9334: Remote ATtestation procedureS (RATS) Architecture. IETF Data Tracker, Jan 2023.

[2] Antoine Delignat-Lavaud, Cédric Fournet, Kapil Vaswani, Sylvan Clebsch, Maik Riechert, Manuel Costa, and Mark Russinovich. Why Should I Trust Your Code? *Commun. ACM*, 67(1), 2023.

[3] Gematic. Healthcare Confidential Compute Specification. https://gemspec.gematik.de/prereleases/Dev_HCC/gemSpec_HCC_V0.9.0_20241118/, 2025. Accessed: 2025-02-20.

[4] Google. Verify a Confidential VM instance's firmware. https://cloud.google.com/confidential-computing/confidential-vm/docs/verify-firmware#retrieve, 2025.

[5] Intel. Intel® Software Guard Extensions Programming Reference. https://www.intel.com/content/dam/develop/external/us/en/documents/329298-002-629101.pdf, 2014. Accessed: 2025-02-20.

[6] Intel. Intel® Trust Domain CPU Architectural Extensions. https://cdrdv2.intel.com/v1/dl/getContent/733582, 2021. Accessed: 2025-02-20.

[7] Chris Lamb and Stefano Zacchiroli. Reproducible builds: Increasing the integrity of software supply chains. 39(2), 2022.

[8] Marcela S. Melara. Software Supply Chain Attribute Integrity (SCAI). *arXiv, 2210.05813*, 2023.

[9] Marcela S. Melara and Chad Kimes. Auditing the CI/CD Platform: Reproducible Builds vs. Hardware-Attested Build Environments, Which is Right for You? In *Proceedings of the ACM Workshop on Supply Chain Offensive Research and Ecosystem Defenses (SCORED)*, 2024.

[10] National Telecommunications and Information Administration. SOFTWARE BILL OF MATERIALS. https://www.ntia.gov/page/software-bill-materials, 2025.

[11] Zachary Newman, John Speed Meyers, and Santiago Torres-Arias. Sigstore: Software signing for everybody. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2022.

[12] Mark Russinovich. Azure AI Confidential Inferencing: Technical Deep-Dive. https://techcommunity.microsoft.com/blog/azureconfidentialcomputingblog/azure-ai-confidential-inferencing-technical-deep-dive/4253150, 2024. Accessed: 2025-02-20.

[13] M. Spoczynski, M. S. Melara, and S. Szyller. Atlas: A Framework for ML Lifecycle Provenance & Transparency. In *Proceedings of the Workshop on System Software for Trusted Execution (SysTEX)*, 2025.

[14] The Linux Foundation. Safeguarding artifact integrity across any software supply chain. https://slsa.dev, 2025.

[15] S. Torres-Arias, H. Afzali, T. K. Kuppusamy, R. Curtmola, and J. Cappos. in-toto: Providing farm-to-table guarantees for bits and bytes. In *Proceedings of the USENIX Security Symposium*, pages 1393–1410, 2019.