An Exploratory Study of Attestation Mechanisms for Trusted Execution Environments

Jämes Ménétrey
james.menetrey@unine.ch
University of Neuchâtel
Switzerland

Christian Göttel
christian.goettel@unine.ch
University of Neuchâtel
Switzerland

Marcelo Pasin
marcelo.pasin@unine.ch
University of Neuchâtel
Switzerland

Pascal Felber
pascal.felber@unine.ch
University of Neuchâtel
Switzerland

Valerio Schiavoni
valerio.schiavoni@unine.ch
University of Neuchâtel
Switzerland

Abstract

Attestation is a fundamental building block to establish trust over software systems. When used in conjunction with trusted execution environments, it guarantees that genuine code is executed even when facing strong attackers, paving the way for adoption in several sensitive application domains. This paper reviews existing remote attestation principles and compares the functionalities of current trusted execution environments as Intel SGX, Arm TrustZone and AMD SEV, as well as emerging RISC-V solutions.

Keywords

TEEs, attestation, Intel SGX, Arm TrustZone, AMD SEV, RISC-V

ACM Reference Format:
Jämes Ménétrey, Christian Göttel, Marcelo Pasin, Pascal Felber, and Valerio Schiavoni. 2022. An Exploratory Study of Attestation Mechanisms for Trusted Execution Environments. In *Proceedings of the 5th Workshop on System Software for Trusted Execution (SysTEX '22 Workshop).* 7 pages.

1 Introduction

Confidentiality and integrity are essential building blocks for secure computer systems, especially if the underlying system cannot be trusted. For example, video broadcasting software can be tampered with by end-users who circumvent digital rights management. Also, virtual machines are candidly open to the indiscretion of their cloud-based untrusted hosts. The availability of Intel SGX, AMD SEV, RISC-V, Arm TrustZone-A/M *Trusted Execution Environments* (TEEs) into commodity processors significantly helps to build trusted applications. In a nutshell, TEEs execute software with stronger security guarantees, including privacy and integrity, without relying on a trustworthy operating system.

Remote attestation allows trusting a specific piece of software by verifying its authenticity and integrity. Through remote attestation, one ensures to be communicating with a specific, trusted (attested) program remotely. TEEs can support and strengthen the attestation process, ensuring the software being attested is shielded against powerful attacks and isolated from the outer system. However, TEEs are used for attestation using a variety of different techniques. This survey reviews the current practices regarding *remote attestation mechanisms* for TEEs [31], covering a selection of TEEs of the four major architectures, namely Intel SGX, Arm TrustZone-A/-M, AMD SEV and a few emerging implementations for RISC-V.

This paper is organised as follows. In §2, we describe the general principles of attestation and highlight the differences between local and remote attestation. In §3 we survey the existing support for attestation in the TEE implementations currently available in commodity hardware. We conclude in §4 discussing some future directions.

2 Attestation

Attestation is an operation through which one software environment proves that a specific program is running in specific hardware. Local attestation is used locally, between two software environments running in the same hardware, where one trusted environment proves its identity to another environment, hosted on the same system. Attestation is based on a local hardware-bound secret called *root of trust*, used to generate keys to sign the code being executed. One can assess whether an attestation is genuine by verifying that the signature (of a specific processor) matches the code supposed to be in execution. The result of an attestation can be used to establish new secrets, *i.e.*, to establish secure communication channels between both environments.

Remote attestation can establish trust between software environments running in *different* hardware. This document adopts the terminology from IETF [6]. A *relying party* wishes to establish a trusted relationship with an *attester*, leveraging a *verifier*. The attester provides the state of its system, indicating the hardware and the software stack that runs on its device by collecting a set of *claims*. An example of a claim is the device's application code measurement, typically a cryptographic hash. Claims are collected and cryptographically signed to form an *evidence*, later asserted or denied by the verifier. Once the attester is proven genuine, the relying party can safely interact with it and, for instance, transfer confidential data.

The problem of remotely attesting software has been extensively studied, and many implementations already exist based on software, hardware, or a combination of both. Software-based remote attestation [40, 11, 43] does not depend on any particular hardware, and it is adapted to low-cost devices. Hardware-based remote attestation can rely on tamper-resistant hardware as, for instance, a *Trusted Platform Module* (TPM) to ensure that the claims are trustworthy [46], or a *Physical Unclonable Function* (PUF) that prevents impersonations by using unique hardware marks produced at manufacture [24, 16]. Other approaches exist,

SysTEX '22 Workshop, @ASPLOS, Lausanne, Switzerland
2022.

Features	**SGX**	**TrustZone**	**SEV**			**RISC-V**			
			Vanilla	SEV-ES	SEV-SNP	Keystone	Sanctum	TIMBER-V	LIRA-V
Integrity	●	○	○	○	●	●	○	○	○
Freshness	●	○	○	○	●	●	○	○	○
Encryption	●	○	●	●	●	●	○	○	○
Unlimited domains	●	○	◐	●	●	●	●	●	○
Open source	◐	◐	○	○	○	●	●	●	○
Local attestation	●	○	○	○	○	○	●	●	○
Remote attestation	●	◐	●	●	●	●	●	●	●
API for attestation	●	◐	○	○	●	●	●	●	●
Mutual attestation	○	◐	○	○	○	○	○	○	●
User-mode support	●	●	●	●	●	●	●	●	○
Industrial TEE	●	●	●	●	●	○	○	○	○
Isolation and attestation granularity	Intra-address space	Secure world	Virtual machine	Virtual machine	Virtual machine	Secure world	Intra-address space	Intra-address space	Intra-address space
System support for isolation	Microcode	Secure monitor	Firmware	Firmware	Firmware	Secure monitor + PMP	Secure monitor + PMP	Tagged memory + PMP	PMP

Table 1: Comparison of the state-of-the-art TEEs.

such as exposing a hardware secret fused in a die exclusively to a trusted environment. Hybrid solutions combine hardware and software [17, 9, 14, 33], in an attempt to leverage advantages from both sides. In §3 we describe how TEEs support remote attestation.

Trusted applications may need stronger trust assurances by ensuring both ends of a secure channel are attested. For example, when retrieving confidential data from a sensing IoT device (for sensitive data), the device must authenticate the remote party, while the latter must ensure the sensing device has not been spoofed or tampered with. *Mutual attestation* protocols have been designed to appraise the trustworthiness of both end devices involved in a communication. We report in §3 how mutual attestation has been studied in the context of TEE.

3 Issuing attestations using TEEs

Several solutions exist to implement hardware support for trusted computing, and TEEs are particularly promising. Typically, a TEE consists of isolating critical components of the system, *e.g.*, portions of the memory, denying access to more privileged but untrusted systems, such as kernel and machine modes. Depending on the implementation, it guarantees the confidentiality and the integrity of the code and data of trusted applications, thanks to the assistance of CPU security features. This work surveys modern and prevailing TEEs from processor designers and vendors with remote attestation capabilities for commodity or server-grade processors, namely Intel SGX [12], AMD SEV [15], and Arm TrustZone [34]. Besides, RISC-V, an open ISA with multiple open source core implementations, ratified the *Physical Memory Protection* (PMP) instructions, offering similar capabilities to memory protection offered by aforementioned technologies [36]. As such, we also included many emerging academic and proprietary frameworks that capitalise on standard RISC-V primitives, which are Keystone [27], Sanctum [13], TIMBER-V [45] and LIRA-V [42]. Finally, among the many other technologies in the literature, we omitted the TEEs lacking remote attestation mechanisms (*e.g.*, IBM PEF [20]) as well as the TEEs not supported on currently available CPUs (*e.g.*, Intel TDX [37], Realm [5] from Arm CCA [4]).

3.1 TEE cornerstone features

We propose a series of cornerstone features of TEEs and remote attestation capabilities and compare many emerging and well-established state-of-the-art solutions in Table 1. Each feature is detailed below and can either be missing (○), partially (◐) or fully (●) available. While we define these features below, we elaborate further about each TEE in the remainder of the section.

Integrity: an active mechanism preventing DRAM of TEE instances from being tampered with. *Freshness*: protecting DRAM of TEE instances against replay and rollback attacks. *Encryption*: TEE instances' DRAM is encrypted for providing some assurance that no unauthorised access or memory snooping of the enclave occurs. *Unlimited domains*: many TEE instances can run concurrently, while the TEE boundaries (*e.g.*, isolation, integrity) between these instances are guaranteed by hardware. Partial fulfilment means that the number of domains is capped. *Open source*: indicate whether the solution is either partially or fully publicly available. *Local attestation*: a TEE instance can attest to another instance running on the same system. *Remote attestation*: a TEE instance can be attested by remote parties. Partial fulfilment means no built-in support, but extended by the literature. *API for attestation*: an API is available by the trusted applications to interact with the process of remote attestation. Partial fulfilment means no built-in support, but extended by the literature. *Mutual attestation*: the identity of the attestation and the verifier are authenticated upon remote attestations. Partial fulfilment means no built-in support, but extended by the literature. *User mode support*: state whether the trusted applications are hosted in user mode, according to the processor architecture. *Industrial TEE*: contrast the TEEs used in production and made by the industry from the research prototypes designed by the academia. *Isolation and attestation granularity*: the level of granularity where the TEE operates for providing isolation and attestation of the trusted software. *System support for isolation*: the hardware mechanisms used to isolate trusted applications.

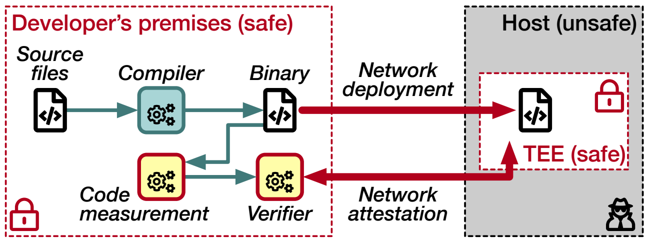


Figure 1: The workflow of deployment and attestation of TEEs.

3.2 TEEs and remote attestation

The attestation of software and hardware components require an environment to issue evidences securely. In practice, this role is usually assigned to some mechanism that cannot be tampered with. These environments rely on measuring the executed software (*e.g.*, by hashing its code) and combining that output with cryptographical values derived from the hardware, such as a root of trust fused in the die or a physical unclonable function. We analysed today's practices for the leading processor vendors for issuing cryptographically signed evidences.

Figure 1 illustrates the generic workflow for the deployment of trusted applications. Initially, the application is compiled and measured on the developers' premises. It is later transferred to an untrusted system, executed in the TEE. The trusted application then communicates with a verifier to establish a trusted channel. The TEE environment helps this transaction by exposing an evidence to the trusted application, which adds key material to it, preventing an attacker from eavesdropping on the communication. The verifier asserts the evidence comparing it to a list of reference values to identify genuine instances of trusted applications.

3.3 Intel SGX

Intel *Software Guard Extensions* (SGX) [12] introduced TEEs for mass-market processors in its Skylake architecture in 2015. SGX is a set of instructions to create encrypted regions of memory, called *enclaves*, protected in a special execution mode of the CPU. Figure 2a illustrates the high-level architecture of SGX. A memory region is reserved at boot time for storing code and data of encrypted enclaves. This memory area, called the *Enclave Page Cache* (EPC), is inaccessible to other programs running on the same machine, including the operating system and the hypervisor. The traffic between the CPU and the system memory remains confidential thanks to the *Memory Encryption Engine* (MEE). The EPC also stores verification codes to ensure that the RAM corresponding to the EPC was not modified by any software external to the enclave.

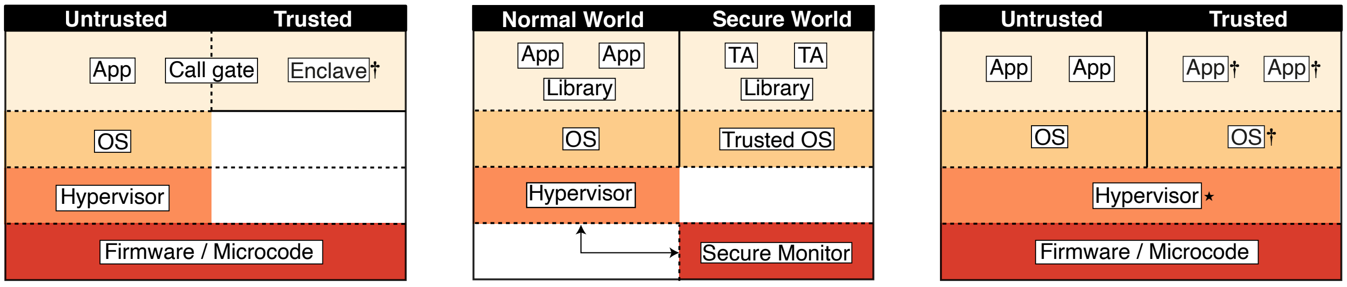
A trusted application executing in an enclave may establish a local attestation with another enclave running on the same hardware. Reports are structures that are created and signed by the `EREPORT` instruction. Reports contain identities, attributes (*i.e.*, modes and other properties), the trustworthiness of the *Trusted Computing Base* (TCB is the amount of hardware and software that needs to be trusted), additional information for the target enclave and a *Message Authentication Code* (MAC). This principle is further extended by Intel thanks to the usage of the built-in *quoting enclave*. A trusted application can hence receive a cryptographically signed evidence, *i.e.*, a *quote*, which may be enhanced by additional information, such as a public key (*e.g.*, used for establishing a communication channel). The quote binds a genuine Intel SGX processor with the measurement of the application when loaded into the enclave. This quote can then be forwarded to a relying party and be verified remotely using the Intel attestation service [3, 7] or a dedicated public key infrastructure [39]. Intel designed their remote attestation protocol based on the SIGMA protocol [25] and extended it to the *Enhanced Privacy ID* (EPID). While the quoting enclave is closed-source and the microcode [22] of Intel SGX are not disclosed, recent work analysed the TEE and its attestation mechanism formally [44, 38]. The other components of SGX (*i.e.*, kernel driver and SDK) are open source. MAGE [10] further extended the remote attestation scheme of Intel SGX by offering mutual attestation for a group of enclaves without trusted third parties.

Unlike local attestation, remote attestation requires an asymmetric-key scheme, which is made possible by the quoting enclave. The quoting enclave is a special enclave that has access to the device-specific private key through the `EGETKEY` instruction. First, enclaves do a local attestation with the quoting enclave. The quoting enclave replaces the MAC after verification by a signature created with the private device key. The EPID scheme does not identify unique entities, but rather a group of signers. Each signer belongs to a group, and the verifier checks the group's public key. Quotes are signed by the EPID key, which is bound to the firmware version of the processor [3]. The quoting enclave manages the EPID key and has exclusive access to it.

In a remote attestation scenario, a service (*i.e.*, verifier) submits a challenge to the untrusted application with a nonce (Fig.3-①). Together with the identity of the quoting enclave, the challenge is forwarded to the application enclave (Fig.3-②). The application enclave (*i.e.*, attester) prepares a response to the challenge by creating a manifest (*i.e.*, a set of claims) and a public key (Fig.3-③), that is used to send back confidential information to the application enclave. The manifest hash is used as auxiliary data in the report for the local attestation with the quoting enclave. After verifying the report (Fig.3-⑥), the quoting enclave replaces the MAC with the signature from the EPID key and returns the quote (*i.e.*, evidence) to the application (Fig.3-⑦) which sends it back to the service (Fig.3-⑧). The service verifies the signature of the quote (Fig.3-⑨) using either the EPID public key and revocation information or an attestation verification service [3]. Finally, the service ensures the integrity of the manifest by verifying the response to the challenge. *Data Center Attestation Primitives* (DCAP) [21] is an alternative solution to EPID that enables third-party attestation for SGX of server-grade processors.

3.4 Arm TrustZone architectures

Arm TrustZone [34] provides the hardware elements to establish a single TEE per system. Figure 2b illustrates the high-level architecture of TrustZone. Broadly adopted by commodity devices (including mobile devices, IoT edge nodes, *etc.*), TrustZone splits the processor into two states: the secure world (TEE) and the normal world (untrusted environment). A secure monitor instruction (*i.e.*, the SMC) is switching between worlds, and each world operates with their own user and kernel spaces. The trusted world uses a trusted operating system (such as OP-TEE [29]) and runs *Trusted Applications* (TAs) as isolated processes. The normal world uses a traditional operating system such as Linux.



(a) Intel SGX

(b) Arm TrustZone

(c) AMD SEV

Figure 2: High-level description of major TEE architectures († indicates attested components, ⋆ is untrusted for SEV-SNP).

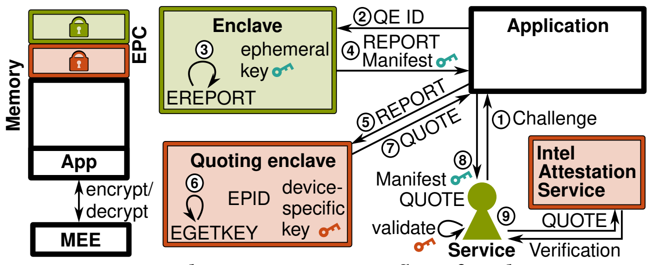


Figure 3: The remote attestation flow of Intel SGX.

Despite the commercial success of TrustZone, it lacks attestation mechanisms, preventing relying parties from validating and trusting the state of a TrustZone TEE remotely. Many protocols have been proposed for Arm TrustZone one-way remote attestation [47, 28], as well as for mutual remote attestation [1, 41], extending the capabilities of built-in hardware. These protocols require the availability of extra hardware with a root of trust in the secure world, a secure source of randomness for cryptographic operations, and a secure boot mechanism. Indeed, devices lacking built-in attestation mechanisms may rely on a secret fused in the die as a root of trust to derive private cryptographic materials (*e.g.*, a private key for evidence issuance). Secure boot can measure the integrity of individual boot stages on devices and prevent tampered systems from being booted. As a result, remote parties can verify issued evidences in the TEE and ensure the trustworthiness of the attesters.

In the following, we describe the remote attestation mechanism of Shepherd et al. [41] as a study case. This solution establishes mutually trusted channels for bi-directional attestation, based on a *Trusted Measurer* (TM), which is a software component located in the trusted world and authenticated by the TEE's secure boot, to generate evidences based on the OS and TA states (*i.e.*, a set of claims). A private key is provisioned and sealed in the TEE's secure storage and used by the TM to sign evidences, similarly to a firmware TPM [35].

Using a dedicated protocol for remote attestation, the bi-directional attestation is accomplished in three rounds. First, the attester sends a handshake request to the verifier containing the identity of both parties and the cryptographic materials to initiate a key establishment. Second, the verifier answers to the handshake by including similar information (*i.e.*, both identifies and cryptographic materials), as well as a signed evidence of the verifier's TEE, based on the computed common secret (*i.e.*, using Diffie-Hellman). Finally, the attester sends back a signed evidence

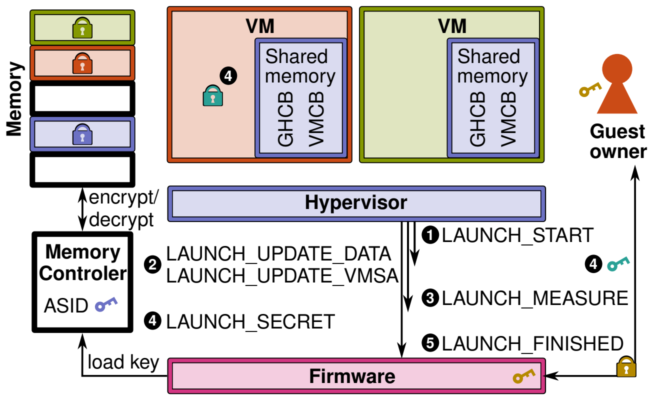


Figure 4: The remote attestation flow of AMD SEV.

of the attester's TEE, based on the same common secret. Once both parties assert that the evidences are genuine, they can derive common secrets to establish a trusted channel of communication.

3.5 AMD SEV

AMD *Secure Encrypted Virtualization* (SEV) [15] allows isolating virtualised environments (*e.g.*, containers and virtual machines) from trusted hypervisors. Figure 2c illustrates the high-level architecture of SEV. SEV uses an embedded hardware AES engine, which seamlessly relies on multiple keys to encrypt memory. It exploits a closed Arm Cortex-v5 processor as a secure co-processor, used to generate cryptographic materials kept in the CPU. Each virtual machine and hypervisor is assigned a particular key and tagged with an *Address Space Identifier* (ASID), preventing cross-TEE attacks. The tag restricts the code and data usage to the owner with the same ASID and protects from any unauthorised usage inside the processor. Code and data are protected by AES encryption with a 128-bit key based on the tag outside the processor package. SEV-ES (*SEV Encrypted State*) [23] is a successor to SEV where register states are encrypted, and the guest operating system needs to grant the hypervisor access to specific guest registers, fixing an error in SEV that could leak sensitive information during interrupts from guests to the hypervisor through registers [19].

Register states are stored with SEV-ES for each virtual machine in a *Virtual Machine Control Block* (VMCB) that is divided into an unencrypted control area and an encrypted *Virtual Machine Save Area* (VMSA). The hypervisor manages the control area to indicate event and interrupt handling, while VMSA contains register states.

Integrity protection ensures that encrypted register values in the VMSA cannot be modified without being noticed and that virtual machines resume with the same state. Requesting services from the hypervisor due to interrupts in virtual machines are communicated over the *Guest Hypervisor Communication Block* (GHCB) that is accessible through shared memory. Hypervisors do not need to be trusted with SEV-ES because they no longer have access to guest registers. *SEV Secure Nested Paging* (SNP) [2] were proposed to prevent rollback attacks [8] allowing a malicious cloud provider with physical access to SEV machines to easily install malicious firmware and be able to read in clear the (otherwise protected) system.

At its core, SEV leverages a *Chip Endorsement Key* (CEK), a secret fused in the die of the processor and issued by AMD for its attestation mechanism. The three editions of SEV may start the virtual machines from an unencrypted state, similarly to SGX enclaves. In such cases, the secrets and confidential data must then be provisioned using remote attestations queries. The AMD secure processor cryptographically measures the content of the virtual machine into a launch digest (*i.e.*, claim). In addition, AMD-SNP measures the metadata associated with memory pages, ensuring the digest also considers the layout of the initial guest memory. While SEV and SEV-ES only support remote attestation during the launch of the guest operating system, SEV-SNP supports a more flexible model. That latter bootstraps private communication keys, enabling the guest virtual machine to request attestation reports (*i.e.*, evidence) at any time and obtain cryptographic materials for data sealing, *i.e.*, storing data securely at rest.

SEV uses six launch commands for hypervisors to prepare encrypted memory before enabling SEV for virtual machines. The `LAUNCH_START` command (Fig.4-❶) creates a guest context in the firmware with the public key of the guest owner provided by the hypervisor. As the hypervisor is loading the virtual machine into memory, `LAUNCH_UPDATE_DATA` commands (Fig.4-❷) are called to encrypt the memory and calculate measurements. The hypervisor initialises the VMSA inside the VMCB with the `LAUNCH_UPDATE_VMSA` command, which is only available if SEV-ES is enabled. When the virtual machine is loaded, and the VMSA is initialised, the hypervisor calls the `LAUNCH_MEASURE` command (Fig.4-❸), which produces a measurement of the encrypted virtual machine. The SEV firmware provides guest owners with the measurement containing a signature of the state of their virtual machine to prove that it is in the expected state. The guest owner verifies that the virtual machine launched correctly and has not been interfered with before provisioning any sensitive data to the virtual machine. Sensitive data, such as image decryption keys, is provisioned through the `LAUNCH_SECRET` command (Fig.4-❹) after which the hypervisor calls the `LAUNCH_FINISHED` command (Fig.4-❺) to indicate that the virtual machine can be executed.

Software development is eased, as AMD SEV protects the whole virtual machine, which comprises the operating system, compared to the SGX paradigm where the applications must be split in an untrusted and trusted part. Nonetheless, this approach increases the attack surface of the secure environment since the TCB is enlarged. The guest operating system must also support SEV, cannot access host devices (PCI passthrough), and the first edition of SEV (called *vanilla* in Table 1) is limited to 16 virtual machines.

3.6 RISC-V architectures

Several TEE designs were proposed for RISC-V based on its physical memory protection (PMP) instructions, some even including support for remote attestation. In the following, we describe the designs we deemed more important for the scope of our study.

Keystone [27] is a modular framework that provides the building blocks to create trusted execution environments. Keystone implements a secure monitor at machine mode (M-mode) and relies on the RISC-V PMP instructions, without requiring any hardware change. Users can select their own set of security primitives, *e.g.*, memory encryption, dynamic memory management and cache partitioning. Each trusted application executes in user mode (U-mode) and embeds a runtime that executes in supervisor mode (S-mode). The runtime decouples the infrastructure aspect of the TEE (*e.g.*, memory management, scheduling) from the security aspect handled by the secure monitor. Keystone utilises a secure boot mechanism that measures the secure monitor image, generates an attestation key and sign them using a hardware-visible secret (*i.e.*, root of trust). The secure monitor exposes a *Supervisor System Interface* (SBI) for the enclaves to communicate. A subset of the SBI is dedicated to issue evidences signed by provisioned keys (*i.e.*, endorsed by the verifier), based on the measurement of the secure monitor, the runtime and the enclave's application. Arbitrary data can be attached to the evidence, enabling an attester to create a secure communication channel with a verifier using standard protocols. When a remote attestation request occurs, the remote party (*i.e.*, verifier) sends a challenge to the trusted application. The response contains the evidence with the public session key of the attester. Finally, the evidence is verified based on the public signature and the measurements of components (*i.e.*, claims). While Keystone does not describe in-depth the protocol, the authors provide a case study of remote attestation.

Sanctum [13] has been the first proposition with support for attesting trusted applications. It offers similar promises to Intel's SGX by providing provable and robust software isolation, running in enclaves. The authors replaced Intel's opaque microcode with two open-source components: the *measurement root* (`mroot`) and a secure monitor as a means to provide verifiable protection. A remote attestation protocol is proposed with a design for deriving trust from a root of trust. Upon booting the system, `mroot` generates the necessary keys and hands off to the secure monitor. Similarly to SGX, Sanctum owns a dedicated signing enclave, that receives a derived private key from the secure monitor to generate evidences. The remote attestation protocol requires the attester to establish a session key with the verifier. Afterwards, a regular enclave can request an evidence to the signing enclave based on multiple claims, such as the hash of the code of the requesting enclave and some information coming from the key exchange messages. This evidence is then forwarded to the verifier by the secure channel previously established for examination. This work has been further extended to establish a secure boot mechanism and an alternative method for remote attestation by deriving a cryptographic identity from manufacturing variation using a PUF, which is useful when a root of trust is not present [26].

TIMBER-V [45] achieved the isolation of execution on small embedded processors thanks to hardware-assisted memory tagging. Tagged memory transparently associates blocks of memory with additional metadata. Unlike Sanctum, they aim to bring enclaves to smaller RISC-V with limited physical memory. Similarly to TrustZone, user and supervisor modes are split into secure and normal worlds. The secure supervisor mode runs a trust manager, called *TagRoot*, which manages the tagging of the memory. The secure user mode improves the model of TrustZone, as it can handle multiple isolated enclaves. They combine tagged memory with an MPU to support an arbitrary number of processes. The trust manager exposes an API for the enclaves to retrieve an evidence, based on a given enclave identity, a secret platform key (*i.e.*, root of trust), and an arbitrary identifier provided by the trusted application. The remote attestation protocol is twofold: the remote party (*i.e.*, verifier) sends a challenge to the trusted application (*i.e.*, attester). Next, the challenge is forwarded to the trust manager as an identifier to issue an evidence, which is authenticated using a MAC. The usage of symmetric cryptography is unusual in remote attestation because the verifier requires to own the secret key to verify the evidence. The authors added that TIMBER-V could be extended to leverage public key cryptography for remote attestation.

LIRA-V [42] drafted a mutual remote attestation for constrained edge devices. While this solution does not enable arbitrary code execution in a TEE, it introduces a comprehensive remote attestation mechanism. The proposed protocol relies exclusively on machine mode (M-mode) or machine and user mode (M-mode and U-mode). Claims are computed on parts of the device physical memory regions by a program stored in the ROM. LIRA-V's mutual attestation is similar to the protocol illustrated in TrustZone-A, and requires provisioned keys as a root of trust. The first device (*i.e.*, verifier) sends a challenge with a public session key. Next, the second device (*i.e.*, attester) answers with a challenge and public session key, as well as an evidence bound to that device and encrypted using the established shared session key. Finally, if the first device asserts the evidence, it becomes the attester and issues an evidence to be sent to the second device, which becomes the verifier.

We omitted some other emerging TEEs leveraging RISC-V as they lack a remote attestation mechanism. For instance, SiFive, a provider of commercial RISC-V processors, proposes Hex-Five MultiZone [18], a zero-trust computing architecture enabling the isolation of software, called *zones*. The multi zones kernel ensures the sane state of the system using secure boot and PMP and runs unmodified applications by trapping and emulating functionality for privileged instructions. HECTOR-V [32] is a design for developing hardened TEEs with a reduced TCB. Thanks to a tight coupling of the TEE and the SoC, the authors provide runtime and peripherals services directly from the hardware and leverage a dedicated processor and a hardware-based security monitor, which ensure the isolation and the control-flow integrity of the trusted applications, called *trustlets*. Finally, Lindemer et al. [30] enable simultaneous thread isolation and TEE separation on devices with a flat address space (*i.e.*, without an MMU), thanks to a minor change in the PMP specification.

4 Conclusion

This work compares state-of-the-art remote attestation schemes, which leverage hardware-assisted TEEs, helpful for deploying and running trusted applications from commodity devices to cloud providers. TEE-based remote attestation has not yet been extensively studied and seems to remain an industrial challenge.

Our survey highlights four architectural extensions: Intel SGX, Arm TrustZone, AMD SEV, and upcoming RISC-V TEEs. While SGX competes with SEV, the two pursue significantly different approaches. The former provides a complete built-in remote attestation protocol for multiple, independent, trusted applications. The latter is designed for virtualized environments, shielding VMs from untrusted hypervisors, and provides instructions to help the attestation of independent VMs. Arm TrustZone and native RISC-V do not provide means for attesting software running in the trusted environment, relying on the community to develop software-based alternatives. However, TrustZone-M supports a root of trust, helping to develop an adequately trusted implementation in software. RISC-V extensions differ a lot, offering different combinations of software and hardware extensions, some of which support a root of trust and multiple trusted applications.

Whether provided by manufacturers or developed by third parties, remote attestation remains an essential part of the design of trusted computing solutions. They are the foundation of trust for remote computing where the target environments are not fully trusted. Current solutions widely differ in terms of maturity and security. Whereas some TEEs are developed by leading processor companies and provide built-in attestation mechanisms, others still lack proper hardware attestation support and require software solutions instead. Our study sheds some light on the limitations of state-of-the-art TEEs and identifies promising directions for future work.

Acknowledgments

This publication incorporates results from the VEDLIoT project, which received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957197.

References

[1] Jaehwan Ahn, Il-Gu Lee, and Myungchul Kim. 2020. Design and implementation of hardware-based remote attestation for a secure internet of things. *Wireless personal communications.*
[2] AMD. 2020. Strengthening VM isolation with integrity protection and more. *White paper.*
[3] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. 2013. Innovative technology for CPU based attestation and sealing. In *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy* (HASP '13). Citeseer.
[4] Arm. 2021. Confidential compute-architecture (CCA). https://bit.ly/3v1fhGQ.
[5] Arm. 2021. The realm management extension (RME), for Armv9-A. Technical report. (June 23, 2021).
[6] Henk Birkholz, Dave Thaler, Michael Richardson, Ned Smith, and Wei Pan. 2021. Remote attestation procedures architecture. Technical report. Internet Engineering Task Force, (April 2021).
[7] Ernie Brickell and Jiangtao Li. 2007. Enhanced privacy ID: a direct anonymous attestation scheme with enhanced revocation capabilities. In *Proceedings of the 2007 ACM workshop on privacy in electronic society* (WPES '07).
[8] Robert Buhren, Christian Werling, and Jean-Pierre Seifert. 2019. Insecure until proven updated: analyzing AMD SEV's remote attestation. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security* (CCS '19).
[9] Xavier Carpent, Norrathep Rattanavipanon, and Gene Tsudik. 2018. Remote attestation of IoT devices via SMARM: shuffled measurements against roving malware. In *2018 IEEE international symposium on hardware oriented security and trust* (HOST '18). IEEE.
[10] Guoxing Chen and Yinqian Zhang. 2020. Mage: mutual attestation for a group of enclaves without trusted third parties. *Arxiv preprint arxiv:2008.09501.*
[11] Young-Geun Choi, Jeonil Kang, and DaeHun Nyang. 2007. Proactive code verification protocol in wireless sensor network. In *Computational science and its*

applications (ICCSA '07). Osvaldo Gervasi and Marina L. Gavrilova, editors.
[12] Victor Costan and Srinivas Devadas. 2016. Intel SGX explained. Cryptology ePrint Archive. (2016).
[13] Victor Costan, Ilia Lebedev, and Srinivas Devadas. 2016. Sanctum: minimal hardware extensions for strong software isolation. In *25th USENIX security symposium* (USENIX Security 16). (August 2016).
[14] Ivan De Oliveira Nunes, Sashidhar Jakkamsetti, Norrathep Rattanavipanon, and Gene Tsudik. 2021. On the TOCTOU problem in remote attestation. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security* (CCS '21).
[15] Advanced Micro Devices. 2019. Secure Encrypted Virtualization API: technical preview. Technical report. Advanced Micro Devices, (July 2019).
[16] Wei Feng, Yu Qin, Shijun Zhao, and Dengguo Feng. 2018. Aaot: lightweight attestation and authentication of low-resource things in IoT and CPS. *Computer networks*.
[17] Aurélien Francillon, Quan Nguyen, Kasper B. Rasmussen, and Gene Tsudik. 2014. A minimalist approach to remote attestation. In *2014 design, automation test in europe conference exhibition* (DATE '14).
[18] Cesare Garlati and Sandro Pinto. 2020. A clean slate approach to Linux security RISC-V enclaves. In *Proceedings of the embedded world conference, nuremberg, germany* (EW '20).
[19] Felicitas Hetzelt and Robert Buhren. 2017. Security Analysis of Encrypted Virtual Machines. In *Proceedings of the 13th ACM SIGPLAN/SIGOPS international conference on virtual execution environments* (VEE '17).
[20] Guerney D. H. Hunt, Ramachandra Pai, Michael V. Le, Hani Jamjoom, Sukadev Bhattiprolu, Rick Boivie, Laurent Dufour, Brad Frey, Mohit Kapur, Kenneth A. Goldman, Ryan Grimm, Janani Janakirman, John M. Ludden, Paul Mackerras, Cathy May, Elaine R. Palmer, Bharata Bhasker Rao, Lawrence Roy, William A. Starke, Jeff Stuecheli, Enriquillo Valdez, and Wendel Voigt. 2021. Confidential computing for OpenPOWER. In *Proceedings of the 16th european conference on computer systems* (EuroSys '21).
[21] Intel. 2018. Intel SGX data center attestation primitives (DCAP). Product brief. (2018).
[22] Intel. 2021. XuCode. https://intel.ly/3rYAhMI.
[23] David Kaplan. 2017. Protecting VM register state with SEV-ES. Technical report. (February 2017).
[24] Joonho Kong, Farinaz Koushanfar, Praveen K. Pendyala, Ahmad-Reza Sadeghi, and Christian Wachsmann. 2014. PUFatt: embedded platform attestation based on novel processor-based PUFs. In *2014 51st ACM/EDAC/IEEE design automation conference* (DAC '14).
[25] Hugo Krawczyk. 2003. SIGMA: the 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols. In *Advances in cryptology* (CRYPTO '03). Dan Boneh, editor.
[26] Ilia Lebedev, Kyle Hogan, and Srinivas Devadas. 2018. Invited paper: secure boot and remote attestation in the sanctum processor. In *2018 IEEE 31st computer security foundations symposium* (CSF '18).
[27] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. 2020. Keystone: an open framework for architecting trusted execution environments. In *Proceedings of the 15th european conference on computer systems* (EuroSys '20).
[28] Wenhao Li, Haibo Li, Haibo Chen, and Yubin Xia. 2015. Adattester: secure online mobile advertisement attestation using trustzone. In *Proceedings of the 13th annual international conference on mobile systems, applications, and services* (MobiSys '15).
[29] Linaro. 2021. OP-TEE. https://www.op-tee.org.
[30] Samuel Lindemer, Gustav Midéus, and Shahid Raza. 2020. Real-time thread isolation and trusted execution on embedded RISC-V. In *First international workshop on secure RISC-V architecture design exploration* (SECRISC-V '20).
[31] Pieter Maene, Johannes Götzfried, Ruan de Clercq, Tilo Müller, Felix Freiling, and Ingrid Verbauwhede. 2018. Hardware-Based Trusted Computing Architectures for Isolation and Attestation. *IEEE transactions on computers*.
[32] Pascal Nasahl, Robert Schilling, Mario Werner, and Stefan Mangard. 2021. HECTOR-V: a heterogeneous CPU architecture for a secure RISC-V execution environment. In *Proceedings of the 2021 ACM asia conference on computer and communications security* (ASIA CCS '21).
[33] Ivan De Oliveira Nunes, Karim Eldefrawy, Norrathep Rattanavipanon, Michael Steiner, and Gene Tsudik. 2019. VRASED: a verified hardware/software co-design for remote attestation. In *28th USENIX security symposium* (USENIX Security 19). (August 2019).
[34] Sandro Pinto and Nuno Santos. 2019. Demystifying Arm TrustZone: a comprehensive survey. *ACM computing surveys (CSUR)*.
[35] Himanshu Raj, Stefan Saroiu, Alec Wolman, Ronald Aigner, Jeremiah Cox, Paul England, Chris Fenner, Kinshuman Kinshumann, Jork Loeser, Dennis Mattoon, Magnus Nystrom, David Robinson, Rob Spiger, Stefan Thom, and David Wooten. 2016. fTPM: a Software-Only implementation of a TPM chip. In *25th USENIX security symposium* (USENIX Security 16). (August 2016).
[36] 2019. RISC-V ISA and privileged architecture specs. https://bit.ly/3LN9Ili.
[37] Muhammad Usama Sardar, Saidgani Musaev, and Christof Fetzer. 2021. Demystifying attestation in Intel trust domain extensions via formal verification. *IEEE access*.
[38] Muhammad Usama Sardar, Do Le Quoc, and Christof Fetzer. 2020. Towards formalization of enhanced privacy id (EPID)-based remote attestation in Intel SGX. In *2020 23rd euromicro conference on digital system design* (DSD '20).
[39] Vinnie Scarlata, Simon Johnson, James Beaney, and Piotr Zmijewski. 2018. Supporting third party attestation for Intel SGX with Intel data center attestation primitives. *White paper*.
[40] Arvind Seshadri, Mark Luk, Elaine Shi, Adrian Perrig, Leendert Van Doorn, and Pradeep Khosla. 2005. Pioneer: verifying integrity and guaranteeing execution of code on legacy platforms. In *Proceedings of acm symposium on operating systems principles* (SOSP '05).
[41] Carlton Shepherd, Raja Naeem Akram, and Konstantinos Markantonakis. 2017. Establishing mutually trusted channels for remote sensing devices with trusted execution environments. In *Proceedings of the 12th international conference on availability, reliability and security* (ARES '17).
[42] Carlton Shepherd, Konstantinos Markantonakis, and Georges-Axel Jaloyan. 2021. LIRA-V: lightweight remote attestation for constrained RISC-V devices. In *2021 IEEE security and privacy workshops* (SPW '21).
[43] Rodrigo Vieira Steiner and Emil Lupu. 2019. Towards more practical software-based attestation. *Computer networks*.
[44] Pramod Subramanyan, Rohit Sinha, Ilia Lebedev, Srinivas Devadas, and Sanjit A. Seshia. 2017. A formal foundation for secure remote execution of enclaves. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security* (CCS '17).
[45] Samuel Weiser, Mario Werner, Ferdinand Brasser, Maja Malenko, Stefan Mangard, and Ahmad-Reza Sadeghi. 2019. TIMBER-V: tag-isolated memory bringing fine-grained enclaves to RISC-V. In *NDSS* (NDSS '19).
[46] Wenjuan Xu, Xinwen Zhang, Hongxin Hu, Gail-Joon Ahn, and Jean-Pierre Seifert. 2012. Remote attestation with domain-based integrity model and policy analysis. *IEEE TDSC*.
[47] Shijun Zhao, Qianying Zhang, Yu Qin, Wei Feng, and Dengguo Feng. 2019. SecTEE: a software-based approach to secure enclave architecture using TEE. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security* (CCS '19).