

Attestation Mechanisms for Trusted Execution Environments Demystified

Jämes Ménétreay¹[0000–0003–2470–2827],
Christian Göttel¹[0000–0002–4465–6197], Anum Khurshid²[0000–0002–5788–069X],
Marcelo Pasin¹[0000–0002–3064–5315], Pascal Felber¹[0000–0003–1574–6721],
Valerio Schiavoni¹[0000–0003–1493–6603], and Shahid Raza²[0000–0001–8192–0893]

¹ University of Neuchâtel, Neuchâtel, Switzerland, first.last@unine.ch

² RISE Research Institutes of Sweden, Stockholm, Sweden, first.last@ri.se

Abstract. Attestation is a fundamental building block to establish trust over software systems. When used in conjunction with trusted execution environments, it guarantees the genuineness of the code executed against powerful attackers and threats, paving the way for adoption in several sensitive application domains. This paper reviews remote attestation principles and explains how the modern and industrially well-established trusted execution environments Intel SGX, Arm TrustZone and AMD SEV, as well as emerging RISC-V solutions, leverage these mechanisms.

Keywords: Trusted execution environments, Attestation, Intel SGX, Arm TrustZone, AMD SEV, RISC-V

1 Introduction

Confidentiality and integrity are essential features when building secure computer systems. This is particularly important when the underlying system cannot be fully trusted or controlled. For example, video broadcasting software can be tampered with by end-users to circumvent digital rights management, or virtual machines are candidly open to the indiscretion of their cloud-based untrusted hosts. The introduction of Trusted Execution Environments (TEEs), such as Intel SGX, AMD SEV, RISC-V and Arm TrustZone-A/-M, into commodity processors, significantly mitigates the attack surface against powerful attackers. In a nutshell, TEEs let a piece of software be executed with stronger security guarantees, including privacy and integrity properties, without relying on a trustworthy operating system. Each of these enabling technologies offers different degrees of guarantees that can be leveraged to increase the confidentiality and integrity of applications.

Remote attestation allows establishing a trusting relationship with a specific 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 that programs are shielded against many powerful attacks by isolating critical security software, assets and private information from the rest of the system. However, to the best of our knowledge, there is not a clear systematisation of attestation

mechanisms supported by modern and industrially well-established TEEs. Hence, the main contribution of this work is to describe the state-of-the-art best practices regarding remote attestation mechanisms of TEEs, covering a necessarily incomplete selection of TEEs, which includes the four major technologies available for commodity hardware, which are Intel SGX, Arm TrustZone-A/-M, AMD SEV and many emerging TEEs using the open ISA RISC-V. We complement previous work [36, 37] with an updated analysis of TEEs (e.g., introduction of Intel SGX and Arm TrustZone variations), a thorough analysis of remote attestation mechanisms and coverage of the upcoming TEEs of Intel and Arm.

2 Attestation

2.1 Local attestation

Local attestation enables a trusted environment to prove its identity to any other trusted environments hosted on the same system, respectively, on the same CPU if the secret provisioned for the attestation is bound to the processor. The target environment that receives the local attestation request can assess whether the issued proof is genuine by verifying its authentication, usually based on a symmetric-key scheme, using a *message authentication code* (MAC). This mechanism is required to establish secure communication channels between trusted environments, often used to delegate computing tasks securely. As an example, Intel SGX’s remote attestation (detailed in Section 3.3) leverages the local attestation to sign proofs in another trusted environment through a secure communication channel.

2.2 Remote attestation

Remote attestation allows to establish trust between different devices and provides cryptographic proofs that the executing software is genuine and untampered [18]. In the remainder, we adopt the terminology proposed by the IETF to describe remote attestation and related architectures [13]. Under these terms, a *relying party* wishes to establish a trusted relationship with an *attester*, thanks to the help of 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* of trustworthiness. A claim is a piece of asserted information collected by an attesting environment, e.g., a TEE. An example of claims is the code *measurement*, (i.e., a cryptographic hash of the application’s code) of an executing program within a TEE. TEEs also create additional claims that identify the *trusted computing base* (TCB is the amount of hardware and software that needs to be trusted), so verifiers are able to evaluate the genuineness of the platform. Claims are collected and cryptographically signed to form *evidence*, later observed and accepted (or denied) by the verifier. Once the attester is proven genuine, the relying party can safely interact with it and transfer confidential data or delegate computations.

The problem of remotely attesting software has been extensively studied in academia, and industrial implementations already exist. Three leading families of remote attestation methods exist: (i) software-based, (ii) hardware-based, and (iii) hybrid (software- and hardware-based). Software-based remote attestation [47] does not depend on any particular hardware. This method is particularly

adapted to low-cost use cases. Hardware-based remote attestation relies on a *root of trust*, which is one or many cryptographic values rooted in hardware to ensure that the claims are trustworthy. Typically, a root of trust can be implemented using tamper-resistant hardware, such as a *trusted platform module* (TPM) [55], a *physical unclonable function* (PUF) that prevents impersonations by using unique hardware marks produced at manufacture [30], or a hardware secret fused in a die (e.g., CPU) exposed exclusively to the trusted environment. Hybrid solutions combine hardware devices and software implementations [21], in an attempt to leverage advantages from both sides. Researchers used hardware/software co-design techniques to propose a hybrid design with a formal proof of correctness [40]. Finally, remote attestation mechanisms are popular among the TEEs due to their carefully controlled environments and their ability to generate code measurements. Section 3 delivers extensive analysis of the state of the art of the TEEs, including their support for remote attestation.

2.3 Mutual 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 (where data is particularly sensitive), 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 also report how mutual attestation has also been studied in the context of TEEs [51], as we further detail in Section 3.

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 [19], AMD SEV [3], and Arm TrustZone [42]. 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. As such, we also included many emerging academic and proprietary frameworks that capitalise on standard RISC-V primitives, which are Keystone [33], Sanctum [20], TIMBER-V [54] and LIRA-V [49]. Finally, among the many other technologies in the literature, we omitted the TEEs lacking remote attestation mechanisms (e.g., IBM PEF [26]) as well as the TEEs not supported on currently available CPUs (e.g., Intel TDX [27], Realm [11] from Arm CCA [8]).

Features	SGX		TrustZone		SEV			RISC-V			
	Client SGX	Scalable SGX	TrustZone-A	TrustZone-M	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		VM			Secure world	Intra-address space		
System support for isolation	μ code + XuCode		SMC	MPU	Firmware			SMC + PMP	Tag + MPU	PMP	

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

Feature	Description
Integrity	An active mechanism preventing DRAM of TEE instances from being tampered with. Partial fulfilment means no protection against physical attacks.
Freshness	Protecting DRAM of TEE instances against replay and rollback attacks. Partial fulfilment means no protection against physical attacks.
Encryption	DRAM of TEE instances is encrypted to assure 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 attests running on the same system to another instance.
Remote attestation	A TEE instance attests genuineness to remote parties. Partial fulfilment means no built-in support but is 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 is extended by the literature.
Mutual attestation	The identity of the attester and the verifier are authenticated upon remote attestations. Partial fulfilment means no built-in support but is 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.

Table 2: Features of the state-of-the-art TEEs.

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 in Table 2 and can either be missing (○), partially (◐) or fully (●) available. Besides, we elaborate further on each TEE in the remainder of the section.

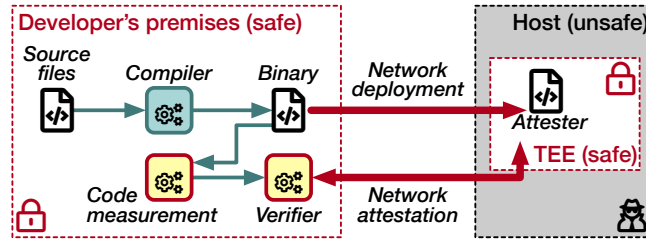


Fig. 1: The workflow of deployment and attestation of TEEs.

3.2 Trusted environments and remote attestation

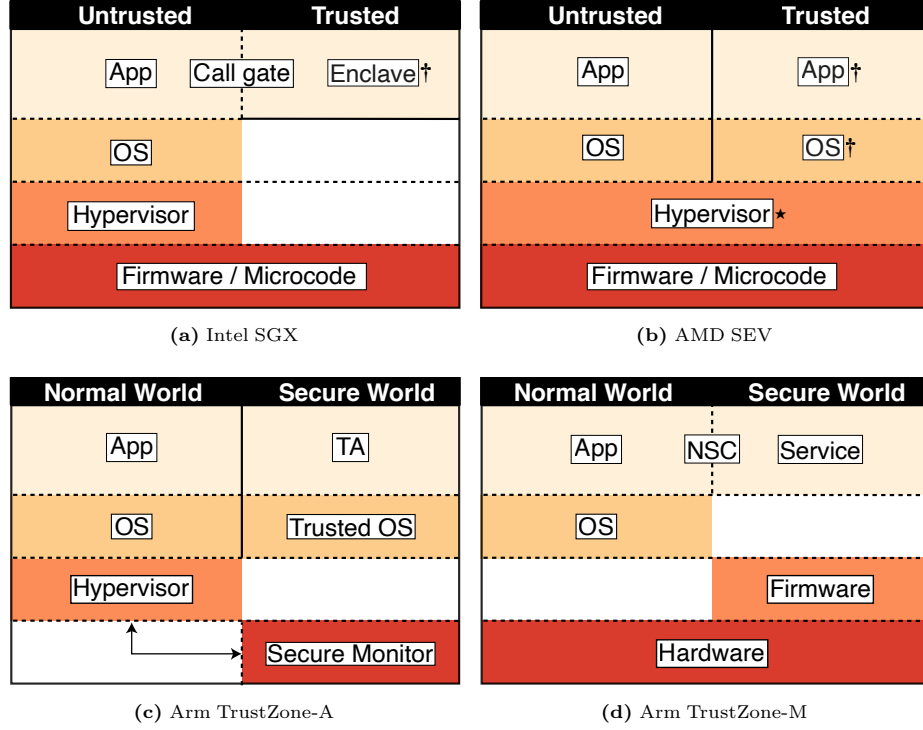
The attestation of software and hardware components requires an environment to issue evidence securely. This role is usually assigned to some software or hardware mechanism that cannot be tampered with. These environments rely on the code measurement of the executed software and combine that claim with cryptographic values derived from the root of trust. We analysed today's practices for the leading processor vendors for issuing cryptographically signed evidence.

Figure 1 illustrates the generic workflow TEE developers usually follow 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 facility. Once the trusted application is loaded and required to receive sensitive data, it communicates with a verifier to establish a trusted channel. The TEE environment must facilitate this transaction by exposing evidence to the trusted application, which adds key material to bootstrap a secure channel from the TEE, thus preventing an attacker from eavesdropping on the communication. The verifier examines the evidence, maintaining a list of reference values to identify genuine instances of trusted applications. If recognised as trustworthy, the verifier can proceed to data exchanges.

3.3 Intel SGX

Intel Software Guard Extensions (SGX) [19] introduced TEEs for mass-market processors in 2015. Figure 2a illustrates the high-level architecture of SGX. Specifically, Intel's Skylake architecture introduced a new set of processor instructions to create encrypted regions of memory, called *enclaves*, living within the processes of the user space. Intel SGX exist in two flavours: *client* SGX and *scalable* SGX [12]. The former is the technology released in 2015, designed and implemented into consumer-grade processors, while the latter was released in 2021, focusing on server-grade processors. The key differences between the two variants are: (i) the volatile memory available to enclaves, 128 MB and 512 GB, respectively, (ii) the multi-socket support and (iii) the lack of integrity and replay protections against hardware attacks for the latter. Researchers conduct work to bring integrity protection for scalable SGX [12].

These instructions are their own ISA that is implemented in XuCode [28] and together with *model specific registers* provide the requirements to form the implementation of SGX. XuCode is a technology that Intel developed and integrated into selected processor families to deliver new features more quickly and,

**Fig. 2:** Overview of the industrial TEE architectures.

(† denotes the attested elements)

(★ means trusted for SEV/SEV-ES, untrusted for SEV-SNP)

particularly for SGX, reduce the impact a (complex) hardware implementation would have had on the features. It operates from protected system memory in a special execution mode of the CPU, which are both set up by system firmware. SGX is, to date, the only technology that is making use of XuCode.

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 DRAM 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. Toward this end, Intel SGX issues a set of claims, called *report*, that contains identities, attributes (i.e., modes and other properties), the trustworthiness of the TCB, additional information for the target enclave and a MAC. Unlike local attestation, remote attestation uses an asymmetric-key scheme, which is made possible by a special enclave, called *quoting enclave*, that has access to the device-specific private key. Intel designed their remote attestation protocol based on the SIGMA protocol [31]

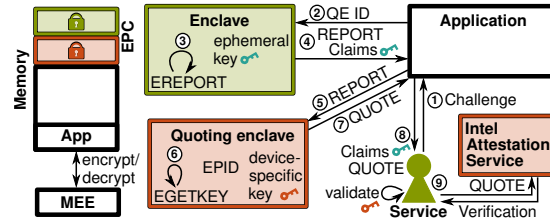


Fig. 3: The remote attestation flow of Intel SGX.

and extended it to the *enhanced privacy ID* (EPID). The EPID scheme does not identify unique entities, but rather a group of attesters. Each attester belongs to a group, and the verifier checks the group’s public key. Evidence is signed by the EPID key, which guarantees the trustworthiness of the hardware and is bound to the firmware version of the processor.

In a remote attestation scenario, a verifier submits a challenge to the attester (i.e., application enclave) with a nonce (Fig.3-①). The attester prepares a response to the challenge by creating a set of claims, a public key (Fig.3-③), and performs a local attestation with the quoting enclave. After verifying the set of claims (i.e., report), the quoting enclave signs the report to form evidence with the EPID key obtained using the EGETKEY instruction (Fig.3-⑥) and returns the evidence to the attester (Fig.3-⑦), which sends it back to the verifier (Fig.3-⑧). The public key contained in the evidence enables the creation of a confidential communication channel. Finally, the verifier examines the signature of the evidence (Fig.3-⑨) using the Intel attestation service (IAS) [5, 14]. If deemed trustworthy, the verifier may provision sensitive data to the attester using the secure channel.

More recently, Intel introduced the Data Center Attestation Primitives (DCAP) [46], an alternative solution to EPID, enabling third-party attestation. Thanks to DCAP, the verifiers have their own attestation infrastructure and prevent depending on external dependencies (e.g., IAS) during the attestation procedure. DCAP introduces an additional step, where the quote (Fig.3-⑦) is signed using *elliptic curve digital signature algorithm* (ECDSA) by the attestation collateral of the attestation infrastructure. Instead of contacting the IAS (Fig.3-⑨), the service retrieves the attestation collateral associated with the received evidence from the attestation infrastructure in order to validate the signature.

While the quoting enclave, the microcode and XuCode are closed-source, recent work analysed the TEE and its attestation mechanism formally [50, 45]. The other components of SGX (i.e., kernel driver and SDK) are open source. MAGE [16] further extended the remote attestation scheme of Intel SGX by offering mutual attestation for a group of enclaves without trusted third parties. Similarly, OPERA [17] proposes a decentralised attestation scheme, unchaining the attesters from the IAS while conducting attestation.

Intel SGX has many advantages but suffers from a few limitations as well. First, most of the SGX implementation limits the EPC size to 93.5 MB [52]. While smaller programs offer smaller attack surfaces, exceeding this threshold increases the memory access latency because of its pagination mechanism. Newer Intel Xeon processors extend that limit to 512 GB, but drop integrity protection

and freshness against physical attacks. Besides, the enclave model prevents performing system calls and direct hardware access since the threat model distrusts the outer world, leading to the development of partitioned applications.

3.4 Arm TrustZone architectures

Depending on the architecture of Arm’s processors, TrustZone comes in two flavours: TrustZone-A (for Cortex-A) and TrustZone-M (for Cortex-M). While they share many design aspects, we detail how different they are in the remainder.

Arm TrustZone-A provides the hardware elements to establish a single TEE per system [42]. Figure 2c illustrates the high-level architecture of TrustZone-A. Broadly adopted by commodity devices, TrustZone splits the processor into two states: the secure world (TEE) and the normal world (untrusted environment). A *secure monitor* (SMC) is switching between worlds, and each world operates with its own user and kernel spaces. The trusted world uses a trusted operating system (e.g., OP-TEE) and runs *trusted applications* (TAs) as isolated processes. The normal world uses a traditional operating system (e.g., Linux).

Despite the commercial success of TrustZone-A, it lacks attestation mechanisms, preventing relying parties from validating and trusting the state of TrustZone-A remotely. Nevertheless, researchers proposed several variants of one-way remote attestation protocols for Arm TrustZone [56, 34], as well as mutual remote attestation [2, 48], thus extending the built-in capabilities of the architecture for attestation. All of these propositions require the availability of hardware primitives on the *system-on-chip* (SoC): (i) a root of trust in the secure world, (ii) a secure source of randomness for cryptographic operations, and (iii) a secure boot mechanism, ensuring the sane state of a system upon boot. Indeed, devices lacking built-in attestation mechanisms may rely on a root of trust to derive private cryptographic materials (e.g., a private key for evidence issuance). Secure boot measures the integrity of individual boot stages on devices and prevents tampered systems from being booted. As a result, remote parties can verify issued evidence in the TEE and ensure the trustworthiness of the attesters.

We describe the remote attestation mechanism of Shepherd et al. [48] 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 claims and issue evidence based on the OS and TA states. A private key is provisioned and sealed in the TEE’s secure storage and used by the TM to sign evidence, similarly to a firmware TPM [43]. Using a dedicated protocol for remote attestation, the bi-directional attestation is accomplished in three rounds:

1. The attester sends a handshake request to the verifier containing the identity of both parties and the cryptographic materials to initiate keys establishment.
2. The verifier answers to the handshake by including similar information (i.e., both identifies and cryptographic materials), as well as evidence of the verifier’s TEE, based on the computed common secret (i.e., using Diffie-Hellman).
3. Finally, the attester sends back signed evidence of the attester’s TEE, based on the same common secret.

Once the two parties validated the genuineness of the evidence, they can derive further shared secrets to establish a trusted communication channel.

Arm TrustZone-A also presents some advantages and drawbacks. Hardware is independently accessible by both worlds, which is helpful for TEE applications utilising peripherals. On the other hand, the reference and open-source trusted OS, i.e., OP-TEE, limits the memory available to TAs by a few MB [24]. Due to this constraint, software needs to be partitioned to leverage TrustZone. Besides, the system must be installed in a particular way: a trusted OS is required, instead of creating TEE instances directly in the regular OS, bringing more complexity. Finally, OP-TEE is small and does not implement a POSIX API, making developing TAs difficult, notably when porting legacy code. While most components of TrustZone have open-source alternatives (e.g., the firmware and the trusted OS), many vendors do not disclose the implementation of the secure monitor.

Arm TrustZone-M (TZ-M) much like its predecessor TrustZone-A, provides an efficient mechanism to isolate the system into two distinct states/processing environments [7]. The TZ-M extension brings trusted execution into resource-constrained IoT devices (e.g., Cortex-M23/M33/M35P/M55). When a TZ-M enabled device boots up, it always starts in the secure world, where the memory is initialised before transferring the control to the normal world. Despite the similarity regarding the high-level concept, TrustZone-M differs from TrustZone-A in low-level implementation of some features. The switch between the secure and the normal world is embedded in hardware and is much faster than the secure monitor [6]. This makes the context switching efficient and suitable for constrained devices. The normal world applications directly call the secure world functions using the *non-secure callable* (NSC) region (Figure 2d). TrustZone-M lacks complex memory management operations like the *memory management unit* (MMU) and only supports the *memory protection unit* (MPU) to enforce even finer levels of access control and memory protection [9]. In TZ-M enabled IoT devices, the secure world runs a concise trusted firmware which provides secure processing in the form of secure services (e.g., TrustedFirmware-M), which is a reference implementation of Platform Security Architecture (PSA) [10]) and the normal world supports real-time operating systems (e.g., Zephyr OS, Arm MBED OS, FreeRTOS).

Since TZ-M is a relatively new addition, recently available for the IoT infrastructure, existing work on attestation mechanisms for the hardware/software is scarce. Nonetheless, TZ-M fulfils some basic requirements for attestation like (i) secure storage, (ii) secure boot, (iii) secure inter-world communication and (iv) isolation of software. Thus, schemes like [1] have leveraged TZ-M to develop attestation and use TZ-M’s TEE capabilities to establish a chain of trust. TrustedFirmware-M, following the guidelines of PSA, also supports initial attestation of device-specific data in the form of a secure service. We provide further details of the remote attestation mechanism introduced in DIAT [1]. It aims at providing run-time attestation of on-device data integrity in autonomous embedded systems in the absence of a central verifier. They provide attestation of the data integrity by identifying the software components (or modules), i.e., the claims, that process the data of concern, verifying that the modules are not

modified, ensuring that all modules of software that influence data are benign. Data integrity is provided by attestation, ensuring correct processing of the sensitive data. The main steps of the protocol are described below:

- The verifier sends a request for data to the attester along with a nonce. The data can represent collected environmental (e.g., a sensing edge device) or compute-bound (e.g., machine learning) information.
- The attester generates the requested data and issues evidence, called the *attestation results*, which are the list of all the software modules that affect the data, and the control flow of each module is derived using the control flow graph. The attester signs the data and the evidence with its secret key and sends the authenticated data to the verifier.
- The verifier assesses the authenticity and integrity of the data by tracing the software modules from the evidence. Since the evidence is comprised of software modules that process the data and the frequency of execution of a module, unauthorised data modifications and code reuse attacks are detected and prevented.

TrustZone-M provides several advantages as a TEE to support remote attestation but also has a few drawbacks. It provides efficient isolation of the software modules and a faster context switch between the secure and normal world. This is advantageous as it is critical to have minimum attestation latency in the real-time operations of embedded systems like autonomous vehicles, industrial control systems, unmanned aerial vehicles, etc. The availability of hardware-unique keys in TZ-M enabled devices further ensures that the evidence generated by the TCB cannot be forged. Besides, the software stack may be fully open source, thanks to the absence of a secure monitor. On the other hand, since the components involved in measuring, attesting, and verifying the data/system need to be protected as part of the TCB, it increases the TCB size on the attested devices, raising the attack surface.

3.5 AMD SEV

AMD Secure Encrypted Virtualization (SEV) [3] allows isolating virtualised environments (e.g., containers and virtual machines) from trusted hypervisors. Figure 2b illustrates the high-level architecture of SEV. SEV uses an embedded hardware AES engine, which relies on multiple keys to encrypt memory seamlessly. 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 (VM) 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 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.

The original version of SEV could leak sensitive information during interrupts from guests to the hypervisor through registers [25]. This issue was addressed with SEV Encrypted State (SEV-ES) [29], where register states are encrypted, and the guest operating system needs to grant the hypervisor access to specific guest

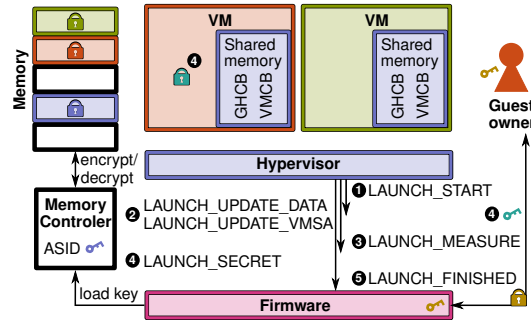


Fig. 4: The remote attestation flow of AMD SEV.

registers. Register states are stored with SEV-ES for each VM in a *virtual machine control block* (VMCB) that is divided into an unencrypted control area and an encrypted *virtual machine save area*. 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 VMs resume with the same state. Requesting services from the hypervisor due to interrupts in VMs 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. However, the remote attestation protocol was recently proven insecure [15], exposing the system to rollback attacks and 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. Future iterations of this technology, i.e., SEV Secure Nested Paging (SEV-SNP) [4], plan to overcome these limitations, typically by means of in-silico redesigns.

At its core, SEV leverages a root of trust, called *chip endorsement key*, 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 VMs from an unencrypted state, similarly to Intel SGX enclaves. In such cases, the secrets and confidential data must then be provisioned using remote attestation. The AMD secure processor creates a claim based on the measurement of the content of the VM. In addition, SEV-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 VM to request evidence at any time and obtain cryptographic materials for data sealing, i.e., storing data securely at rest.

The remote attestation process takes place when SEV is starting the VMs. First, the attester, called *hypervisor*, executes the **LAUNCH_START** command (Fig.4-1) which creates a guest context in the firmware with the public key of the verifier, called *guest owner*. As the attester is loading the VM into memory, the **LAUNCH_UPDATE_DATA/LAUNCH_UPDATE_VMSA** commands (Fig.4-2) are called to encrypt the memory and calculate the claims. When the VM is loaded, the attester calls the **LAUNCH_MEASURE** command (Fig.4-3), which produces evidence

of the encrypted VM. The SEV firmware provides the verifier with evidence of the state of the VM to prove that it is in the expected state. The verifier examines the evidence to determine whether the VM has not been interfered with. Finally, sensitive data, such as image decryption keys, is provisioned through the `LAUNCH.SECRET` command (Fig.4-4) after which the attester calls the `LAUNCH.FINISHED` command (Fig.4-5) to indicate that the VM can be executed.

Software development is eased, as AMD SEV protects the whole VM, which comprises the operating system, unlike Intel SGX, where the applications are split into untrusted and trusted parts. 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 VMs.

3.6 RISC-V architectures

There exist several proposals for TEEs designs for RISC-V based on PMP instructions. These proposals include support for remote attestation, such as those previously described. We survey the most important ones in the following.

Keystone [33] is a modular framework that provides the building blocks to create trusted execution environments, rather than providing an all-in-one solution that is inflexible and is another fixed design point. Instead, they advocate that hardware should provide security primitives instead of point-wise solutions. Keystone implements a secure monitor at machine mode (M-mode) and relies on the RISC-V PMP instructions to provide isolated execution and, therefore, does not require any hardware change. Since Keystone leverages features composition, the framework 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. As such, Keystone programmers can roll their custom runtime to fine-grained control of the computer resources without managing the TEE’s security. Keystone utilises a secure boot mechanism that measures the secure monitor image, generates an attestation key and signs them using a 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 evidence 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 evidence, enabling an attester to create a secure communication channel with a verifier using key establishment protocols (e.g., Diffie-Hellman). When a remote attestation request takes place, the verifier sends a challenge to the trusted application. The response contains evidence with the public session key of the attester. Finally, the verifier examines the evidence based on the public signature and the claims (i.e., measurements of components), leading to establishing a secure communication channel. While Keystone does not describe in-depth the protocol, the authors provide a case study of remote attestation.

Sanctum [20] 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/XuCode with two open-source components: the *measurement root* (**mroot**) and a secure monitor to provide verifiable protection. A remote attestation protocol is proposed, as well as a comprehensive design for deriving trust from a root of trust. Upon booting the system, **mroot** generates the cryptographic materials for signing if started for the first time and hands off to the secure monitor. Similarly to SGX, Sanctum utilises a *signing enclave*, that receives a derived private key from the secure monitor for evidence generation. The remote attestation protocol requires the attester, called *enclave*, to establish a session key with a verifier, called *remote party*. Afterwards, an enclave can request evidence from 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. The evidence is then forwarded to the verifier through the secure channel 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 hardware secret is not present [32].

TIMBER-V [54] 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 featuring only limited physical memory. Similarly to TrustZone, the user mode (U-mode) and the supervisor mode (S-mode) are split into a secure and normal world. 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 concurrent enclaves, which are isolated from each other. They combine tagged memory with an MPU to support an arbitrary number of processes while avoiding the overhead of large tags. The trust manager exposes an API for the enclaves to retrieve evidence, based on a given enclave identity, a root of trust, called the *secret platform key*, and an arbitrary identifier provided by the enclave. The remote attestation protocol is twofold: the verifier (i.e., remote party) sends a challenge to the attester (i.e., enclave). Next, the challenge is forwarded to the trust manager as an identifier to issue 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 [49] drafted a mutual remote attestation for constrained edge devices. While this solution does not enable the execution of arbitrary code in a TEE, it introduces a comprehensive remote attestation mechanism that leverages PMP for code protection of the attesting environment and the availability of a root of trust to issue evidence. The proposed protocol relies exclusively on machine mode (M-mode) or machine and user mode (M-mode and U-mode). The claim, which is the code measurement, is computed on parts of the physical memory regions by a pro-

gram stored in the ROM. LIRA-V’s mutual attestation protocol works similarly to the protocol illustrated in TrustZone-A, in three rounds 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 evidence bound to that device and encrypted using the established shared session key. Finally, if the first device validates the evidence, it becomes the attester and issues evidence for the second device, which becomes the verifier. This protocol has been formally verified and enables the creation of a trusted communication channel upon the validation of evidence.

Lastly, we omitted some other emerging TEEs leveraging RISC-V as they lack remote attestation mechanisms. These technologies are yet to be researched for bringing such capabilities. We briefly introduce them here for completeness. SiFive, the provider of commercial RISC-V processor IP, proposes Hex-Five MultiZone [23], 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 [39] 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. [35] 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 Future work

TEEs and remote attestation are fast-moving research areas, where we expect many technological and paradigm enhancements in the next decades. This section introduces the next trusted environments announced by Intel and Arm. Besides, we also describe a shift to VM-based TEEs and conclude on attestation uniformity.

Intel unveiled Trust Domain Extensions (TDX) [27] in 2020 as its upcoming TEE, introducing the deployment of hardware-isolated virtual machines, called *trust domains*. Similarly to AMD SEV, Intel TDX is designed to isolate legacy applications running on regular operating systems, unlike Intel SGX, which requires tailored software working on a split architecture (i.e., untrusted and trusted parts). TDX leverages Intel Virtual Machine Extensions and Intel Multi-Key Total Memory Encryption, as well as proposes an attestation process to guarantee the trustworthiness of the trust domains for relying parties. In particular, it extends the remote attestation mechanisms of Intel SGX to issue claims and evidence, which has been formally verified by researchers [44].

Arm announced Confidential Compute Architecture (CCA) [8] as part of their future Armv9 processor architecture, consolidating TrustZone to isolate secure virtual machines. With this aim in mind, Arm CCA leverages Arm Realm Management Extension [11] to create a trusted third world called *realm*, next to the existing normal and secure worlds. Arm designed CCA to provide remote attestation mechanisms, assuring that relying parties can trust data and transactions.

These two recent initiatives highlight a convergence into the VM-based isolation paradigm. Initially started by AMD, that architecture of TEEs has many advantages. In particular, it reduces the developers’ friction in writing applications, since the underlying operating system and API are standard and no different compared to the outside of the TEE. Furthermore, a convergence of the paradigm may ease the development of unified and hardware-agnostic solutions for trusted software deployment, such as Open Enclave SDK [41] or the recent initiatives promoting WebAssembly as an abstract portable executable code running in TEEs [22, 53, 38]. Remote attestation may also benefit from these unified solutions by abstracting the attestation process behind standard interfaces.

5 Conclusion

This work compares state-of-the-art remote attestation schemes, which leverage hardware-assisted TEEs, which help deploy and run trusted applications from commodity devices to cloud providers. TEE-based remote attestation has not yet been extensively studied and remains 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 virtualised 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 alternatives. However, TrustZone-M supports a root of trust, helping to develop an adequately trusted implementation. 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 academia, remote attestation remains an essential part 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, and from the Swedish Foundation for Strategic Research (SSF) aSSIsT.

References

1. Abera, T., Bahmani, R., Brasser, F., *et al.*: DIAT: data integrity attestation for resilient collaboration of autonomous systems. In: NDSS '19
2. Ahn, J., Lee, I.-G., and Kim, M.: Design and implementation of hardware-based remote attestation for a secure internet of things. *Wireless Personal Communications* 114(1) (2020)
3. AMD: Secure Encrypted Virtualization API: technical preview. Tech. rep. 55766, (2019)
4. AMD: Strengthening VM isolation with integrity protection and more. White Paper (2020)
5. Anati, I., Gueron, S., Johnson, S., *et al.*: Innovative technology for CPU based attestation and sealing. In: HASP '13
6. Arm: Arm trustzone technology for the Armv8-M architecture. Tech. rep. 100690, (2016)
7. Arm: *Armv8-m architecture reference manual*. DDI0553. 2016.
8. Arm: Introducing Arm confidential compute architecture. Tech. rep. DEN0125, (2021)
9. Arm: Memory protection unit. Tech. rep. 100699, version 2.1 (2018)
10. Arm: Platform security architecture application guide. Tech. rep., version 2 (2019)
11. Arm: *The realm management extension (RME), for Armv9-A*. DDI0615. 2021.
12. Aublin, P.L., Mahhouk, M., and Kapitza, R.: Towards TEEs with large secure memory and integrity protection against HW attacks. In: SysTEX '22
13. Birkholz, H., Thaler, D., Richardson, M., *et al.*: Remote attestation procedures architecture. Tech. rep. draft-ietf-rats-architecture-12, Internet Engineering Task Force (2021)
14. Brickell, E., and Li, J.: Enhanced privacy ID: a direct anonymous attestation scheme with enhanced revocation capabilities. In: WPES '07
15. Buhren, R., Werling, C., and Seifert, J.-P.: Insecure until proven updated: analyzing AMD SEV's remote attestation. In: CCS '19. ACM
16. Chen, G., and Zhang, Y.: Mage: mutual attestation for a group of enclaves without trusted third parties. arXiv preprint [arXiv:2008.09501](https://arxiv.org/abs/2008.09501) (2020)
17. Chen, G., Zhang, Y., and Lai, T.-H.: Opera: open remote attestation for Intel's secure enclaves. In: CCS '19. ACM
18. Coker, G., Guttman, J., Loscocco, P., *et al.*: Principles of remote attestation. *Int. J. Inf. Secur.* 10(2) (2011)
19. Costan, V., and Devadas, S.: "Intel SGX Explained". Cryptology ePrint Archive.
20. Costan, V., Lebedev, I., and Devadas, S.: Sanctum: minimal hardware extensions for strong software isolation. In: USENIX Security 16
21. De Oliveira Nunes, I., Jakkamsetti, S., Rattanaivanon, N., *et al.*: On the TOCTOU problem in remote attestation. In: CCS '21. ACM
22. Enarx, <https://enarx.dev>
23. Garlati, C., and Pinto, S.: A clean slate approach to Linux security RISC-V enclaves. In: EW '20
24. Göttel, C., Felber, P., and Schiavoni, V.: Developing secure services for IoT with OP-TEE: a first look at performance and usability. In: IFIP DAIS '19. Springer
25. Hetzelt, F., and Buhren, R.: Security analysis of encrypted virtual machines. In: VEE '17. ACM
26. Hunt, G.D.H., Pai, R., Le, M.V., *et al.*: Confidential computing for OpenPOWER. In: EuroSys '21. ACM
27. Intel: Trust Domain Extensions, (2020). <https://intel.ly/3L901wS>
28. Intel: XuCode, (2021). <https://intel.ly/3rYAhMI>
29. Kaplan, D.: Protecting VM register state with SEV-ES. Tech. rep., (2017)
30. Kong, J., Koushanfar, F., Pendyala, P.K., *et al.*: PUFatt: embedded platform attestation based on novel processor-based PUFs. In: DAC '14. IEEE
31. Krawczyk, H.: SIGMA: the 'SIGn-and-MAC' approach to authenticated Diffie-Hellman

- and its use in the IKE protocols. In: Boneh, D. (ed.) CRYPTO '03. Springer
32. Lebedev, I., Hogan, K., and Devadas, S.: Invited paper: secure boot and remote attestation in the Sanctum processor. In: CSF '18. IEEE
 33. Lee, D., Kohlbrenner, D., Shinde, S., *et al.*: Keystone: an open framework for architecting trusted execution environments. In: EuroSys '20. ACM
 34. Li, W., Li, H., Chen, H., *et al.*: Adattester: secure online mobile advertisement attestation using trustzone. In: MobiSys '15. ACM
 35. Lindemer, S., Midéus, G., and Raza, S.: Real-time thread isolation and trusted execution on embedded RISC-V. In: SECRISC-V '20
 36. Maene, P., Götzfried, J., Clercq, R. de, *et al.*: Hardware-based trusted computing architectures for isolation and attestation. IEEE Transactions on Computers 67(3) (2018)
 37. Ménétrey, J., Pasin, M., Felber, P., *et al.*: An exploratory study of attestation mechanisms for trusted execution environments. In: SysTEX '22
 38. Ménétrey, J., Pasin, M., Felber, P., *et al.*: TWINE: an embedded trusted runtime for WebAssembly. In: ICDE'21. IEEE
 39. Nasahl, P., Schilling, R., Werner, M., *et al.*: HECTOR-V: a heterogeneous CPU architecture for a secure RISC-V execution environment. In: ASIA CCS '21. ACM
 40. Nunes, I.D.O., Eldefrawy, K., Rattanaivanon, N., *et al.*: VRASED: a verified hardware/software co-design for remote attestation. In: USENIX Security 19
 41. Open Enclave SDK, <https://openenclave.io>
 42. Pinto, S., and Santos, N.: Demystifying Arm TrustZone: a comprehensive survey. ACM Computing Surveys 51(6) (2019)
 43. Raj, H., Saroiu, S., Wolman, A., *et al.*: fTPM: a Software-Only implementation of a TPM chip. In: USENIX Security 16
 44. Sardar, M.U., Musaev, S., and Fetzer, C.: Demystifying attestation in Intel Trust Domain Extensions via formal verification. IEEE Access 9 (2021)
 45. Sardar, M.U., Quoc, D.L., and Fetzer, C.: Towards formalization of enhanced privacy ID (EPID)-based remote attestation in Intel SGX. In: DSD '20. IEEE
 46. Scarlata, V., Johnson, S., Beaney, J., *et al.*: Supporting third party attestation for Intel SGX with Intel data center attestation primitives. White paper (2018)
 47. Seshadri, A., Luk, M., Shi, E., *et al.*: Pioneer: verifying integrity and guaranteeing execution of code on legacy platforms. In: SOSP '05. ACM
 48. Shepherd, C., Akram, R.N., and Markantonakis, K.: Establishing mutually trusted channels for remote sensing devices with trusted execution environments. In: ARES '17. ACM
 49. Shepherd, C., Markantonakis, K., and Jaloyan, G.-A.: LIRA-V: lightweight remote attestation for constrained RISC-V devices. In: SPW '21. IEEE
 50. Subramanyan, P., Sinha, R., Lebedev, I., *et al.*: A formal foundation for secure remote execution of enclaves. In: CCS '17. ACM
 51. Turan, F., and Verbaudhede, I.: Propagating trusted execution through mutual attestation. In: SysTEX '19. ACM
 52. Vaucher, S., Pires, R., Felber, P., *et al.*: SGX-aware container orchestration for heterogeneous clusters. In: ICDCS '18. IEEE
 53. Veracruz, <https://veracruz-project.com>
 54. Weiser, S., Werner, M., Brasser, F., *et al.*: TIMBER-V: tag-isolated memory bringing fine-grained enclaves to RISC-V. In: NDSS '19
 55. Xu, W., Zhang, X., Hu, H., *et al.*: Remote attestation with domain-based integrity model and policy analysis. IEEE TDSC 9(3) (2012)
 56. Zhao, S., Zhang, Q., Qin, Y., *et al.*: SecTEE: a software-based approach to secure enclave architecture using TEE. In: CCS '19. ACM