12th International Conference on Electrical, Electronics and Computer Engineering (IcETRAN), Čačak, 9-12 June 2025

New Approaches to Data Protection in the Cloud

Pavle Vuletić
University of Belgrade, School of Electrical Engineering
Belgrade Serbia
pavle.vuletic@etf.bg.ac.rs
ORCID: 0000-0001-5600-2652

Žarko Stanisavljević
University of Belgrade, School of Electrical Engineering
Belgrade Serbia
zarko@etf.bg.ac.rs
ORCID: 0000-0003-0272-5139

***Abstract*— Despite the flexibility and convenience of cloud services, data privacy during processing on untrusted resources, which cloud servers are, has hindered wider adoption. While technologies like network traffic and disk encryption ensure the protection of data-at-rest and data-in-transit, safeguarding data-in-use remained a challenge. Cryptographic methods like homomorphic encryption offer protection but are significantly slower and more resource-intensive than unprotected data processing. Recent advancements in processor capabilities allow for isolating and encrypting the entire memory of a virtual machine running on a server, leading to the emergence of trusted execution environments (TEEs) and confidential computing. These techniques ensure data protection during processing on untrusted resources. This paper provides an overview of current confidential computing concepts, mechanisms for ensuring trust in remote computations, and recently discovered threats.**

Keywords—confidential computing, cloud, trusted execution environments

I. Introduction

Over the past decade, the use and development of applications that rely on computing and processing various data sets have seen substantial growth, largely driven by advancements in artificial intelligence and machine learning. While the benefits and potential of these applications are clear, data processing remains a sensitive concern, as the data involved is often regarded as personally identifiable or confidential. Handling such data necessitates robust security measures, both to comply with strict legal regulations and to safeguard the business interests of the data owners.

Secure processing or performing secure computation means performing a computation on some data while that data remains secret to unauthorized actors. If the computation is done locally on a single computer or within the infrastructure of a single company without any data transfers outside, there is no need to additionally secure the data as long as the physical and network access to the computing infrastructure are secured and allowed only to the authorized actors. However, there are many cases in which the data owner does not have the adequate capacity to process the data, which led to the rise of *cloud computing* [1]. Also, there are situations when there is a need to do the computation over merged datasets of multiple data owners to achieve some common benefit. In both cases, data owners might want or have to preserve the privacy of their data. While the methods for protecting data at rest (on some kind of storage) and data in transit (during an exchange over an untrusted communication channel) are known for a long time using traditional encryption techniques and protocols, efficiently protecting data in use and providing performant secure computations remained an elusive goal for a long time. Despite the lack of full data processing lifecycle security, cloud services have had continuous market growth for the last two decades, and the forecast is that it will remain so at least until 2032, with a projected annual growth rate of 16.5% [2]. With the proper solution for the data-in-use protection, cloud computing would probably have an additional boost for growth with full confidence in data privacy.

There are two similar models of secure and verifiable computation [3]: *outsourced* and *secure multiparty computation* (SMC). With outsourced computation, a single data owner sends the data in the encrypted form to the other party which performs the computation on encrypted data and sends the encrypted result back to the data owner, without being able to have an insight into the raw data [4]. In the case of SMC, a group of data owners want to perform some computation over their joined data sets without revealing the raw data to any of the parties. The initial research of both models was in the area of cryptographic algorithms which are capable of securing data processing on untrusted hardware. The key enabler for the outsourced computation is *homomorphic encryption* which allows computation over the encrypted data. On the other hand, the secure multiparty computation can be achieved using mechanisms like garbled circuits with the oblivious transfer, secret-sharing, the extension of homomorphic encryption to the multi-user case, or functional encryption. The key issue of all of the previously mentioned mechanisms is data processing performance. Both outsourced and SMC using these mechanisms are by several orders of magnitude slower compared to non-protected data processing on regular hardware. Even the recently reported hardware and software accelerated secure multiparty computation systems based on garbled circuits [5] are slower 2-4 orders of magnitude than general purpose processors in performing some specific computations like dot product or gradient descent. Similar results are obtained for homomorphic encryption [5], which in addition has other implementation issues making secure computations difficult: noise growth, limits of the range of the numbers used in computation, and limited set of supported mathematical operations requiring the changes in the computation algorithms. This makes previously mentioned algorithms still an expensive and far from optimal solution for large-scale and big data secure data processing.

Another privacy-preserving computation approach tailored specifically for machine learning is federated learning. With federated learning many clients collaboratively and independently train a model under the orchestration of a central server. Parts of the model are trained locally by clients without any privacy protection because the data does not leave the client's devices. The clients exchange a minimal amount of information (e.g. intermediate results or model parameters)

ISBN 978-86-6200-031-6 ©2025 AM & ETRAN
https://doi.org/10.69994/12Ic25039

needed to fulfill the machine learning task, while raw data remains fully decentralized. However, one of the greatest challenges for federated learning remains reaching the accuracy of centralized machine learning performed over the whole dataset gathered from all the clients, in cases when the clients' data is not independent and identically distributed (IID). There is also a tension between data privacy and robustness (reliable results in cases of malicious clients who tend to poison the models) [6], which might be critical in the case of sensitive data analysis (e.g. medical data) which requires both strict privacy and very reliable results.

The other branch of secure computation and processing development is in the area of *confidential computing* which uses new processor capabilities that enable secure execution and data-in-use protection. Trusted Execution Environments (TEE) or secure enclaves isolate, and with some technologies encrypt, the data in memory during the processing process. TEEs have a minimal performance penalty on data processing (on the order of 10-15%, as we will show later in the paper), while being able to fully encrypt the content of RAM containing the data that is processed. Such an approach promises to provide a solution for privacy-preserving data processing. However, confidential computing approach does not come without its own set of issues. Ensuring that the processor whose owner is not trusted is in the correct state which preserves privacy, and that the data that is being sent to such a remote processor in a correct state is not easy. It is done through the process of *remote attestation* which has to be trustworthy to the users. Also, confidential computing techniques are continuously maturing, solving security issues found by various research groups. Despite that, the concept is adopted by major cloud providers (e.g. Google Cloud, Microsoft Azure), and the potential users have to understand the benefits of its use, but also the whole process needed to use such trusted enclaves in a way which allows full data security and privacy.

This paper, presents an overview of the current state-of-the-art in the field of confidential computing concepts, mechanisms that enable trust in remote computations, and lists recent threats that were discovered, in an attempt to provide a snapshot of the field in the first half of 2025. The rest of the paper is organized as follows. Section II provides an overview of the existing Trusted Execution Environment technologies. Section III gives the results of a performance evaluation of the most common TEE technologies, clearly indicating the main incentive for the development and use of this approach. Section IV brings the description of the remote attestation concept and describes in details remote attestation in two the most commonly used confidential computing technologies. Section V assesses the technology maturity through the enumeration of some of the vulnerabilities discovered in these technologies. Finally, Section VI concludes the paper.

II. Trusted Execution Environments

There are different approaches for trusted execution environments supported by different processor technologies. In this section we make an overview of the existing technologies, with more emphasis on those TEE solutions that are used and offered as services in cloud environments.

A. Arm TrustZone

Arm TrustZone is one of the first TEE technologies. It exists since the beginning of 2010s as two similar concepts: TrustZone-A and TrustZone-M, introduced in Arm-v6 architecture, made for Cortex-A and Cortex-M processors, used in various commodity devices like mobile phones or IoT devices. In both flavors, TrustZone divides the processor operation into two states: the *secure world* (TEE) and the *non-secure* or *normal* world (as described in the right part of Figure 1). The division of secure and normal worlds in newer TrustZone-M is based on memory map. Code running in secure world can access both secure and non-secure data, while non-secure programs can only access non-secure portions of memory [7]. In TrustZone-M secure memory space is further divided in two types: *secure*, where secure code and data are located, and *non-secure callable* which contains functions for non-secure programs to communicate with the secure code and access secure functions. This approach makes it impossible for the code in non-secure world to access or modify information in secure world. However, it is important to emphasize that Arm TrustZone only separates secure from non-secure applications while not providing any additional privacy protection for the data stored in RAM using cryptographic algorithms which is common today in other approaches. The content of the RAM memory in both worlds remains unencrypted.

With Arm-v9 architecture, in 2021, came an enhanced TrustZone with the addition of secure virtualization and new Dynamic TrustZone technology. Secure virtualization works through the creation of protected execution environments called *realms* (Figure 1). Realm Management Extension (RME), an extension to the TrustZone architecture is the hardware component of the Arm Confidential Compute Architecture (CCA). RME dynamically transfers resources and memory to a new protected address space that higher privileged software or TrustZone firmware cannot access. Realms allow a lower-privileged software, like an application or a Virtual Machine (VM), to protect its content. Realms also prevent execution from attacks using software that runs at higher privilege levels, like an OS or a hypervisor. The instances of virtual machines in the *secure world*, are isolated from each other using stage 2 memory translations and protections. Dynamic TrustZone technology enables also pages of memory to be dynamically transitioned from the Non-secure world to the Secure world and back again. Finally, CCA enables encryption of all data in Secure assigned DRAM through the Memory Protection Engine [8].

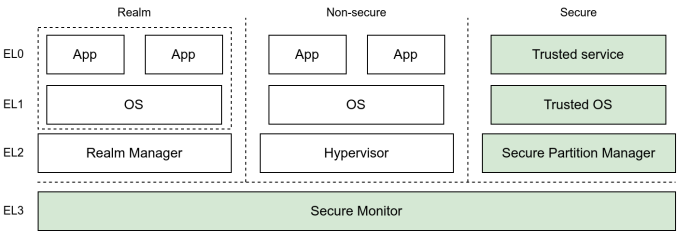

Fig. 1. ARMv9 Confidential Compute Architecture

Apple processors, used in company's consumer electronic products are based on Arm architecture. However, they have a proprietary Secure Enclave architecture (Fig 2.). Secure Enclave is a secure subsystem integrated into Apple system on a chip. It is isolated from the main processor and able to keep sensitive user data secure even when the Application Processor kernel becomes compromised. Secure Enclave Processor (SEP) is used to process the data in the Secure Enclave. Between the SEP and DRAM, there is a Memory Protection Engine (MPE) which when the SEP writes data to its dedicated memory region, encrypts the block of memory using AES and calculates a Message Authentication Code tag for the memory using Xor–encrypt–xor (XEX) mode [9]. Also, both secure and non-secure areas have separate AES engines that generate cryptographic keys which never exit the engine. SEP can transfer the wrapping key to the AES engine in a *non-secure* area order to receive wrapped key for file encryption, generated in AES engine in a non-secure part of the processor, so that Secure Enclave can access files encrypted in the non-secure area if the data-at-rest protection is used.

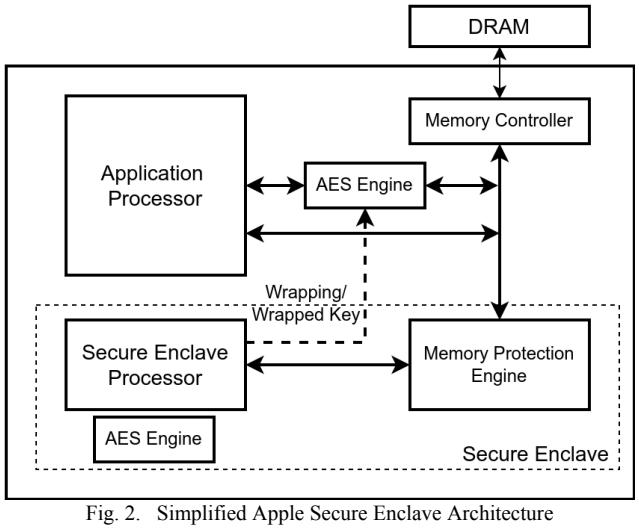

Fig. 2. Simplified Apple Secure Enclave Architecture

B. Intel SGX

The previous section outlined the evolution of Arm's secure execution environment technologies, starting with memory isolation, progressing to full virtual machine isolation, and ultimately incorporating encryption. However, the concept of encrypting secure application components in memory was actually introduced earlier, in 2015, by Intel's Software Guard Extensions (SGX) with the Skylake architecture. This technology appeared in both server processors (e.g., Xeon) and client processors (e.g., Core processors up to the 11th generation). SGX uses symmetric cryptographic algorithms for on-the-fly encryption and data decryption (before writing to and upon reading from system memory) with a hardware-based encryption engine integrated into the CPU. The cryptographic keys used cannot be exported from the processor.

SGX introduced a new set of processor instructions that allow the creation of encrypted memory regions, known as enclaves, within user-space processes. When data is written to enclave memory, the processor encrypts it using a symmetric encryption algorithm (AES), and a cryptographic key used for this is created within the processor and never leaves it. Upon reading data or instructions from the enclave, the processor intercepts the data and decrypts it with the same key. This method provides cryptographic protection for the enclave's data but requires applications to be split into trusted and untrusted parts, making it costly for existing applications to adopt this approach, as they often need to be redesigned or rewritten. Despite this, SGX's approach marked the introduction of on-the-fly memory encryption and the beginning of confidential computing.

Another limitation is the restricted capacity of an SGX enclave, determined by the size of the Enclave Page Cache (EPC). The EPC size varies depending on the processor, with previous generations of Intel processors (before Ice Lake) offering EPC memory sizes ranging from 32 MB to 256 MB, which was insufficient for many large data applications. However, starting with Intel's Ice Lake processors in 2021, the EPC size has been significantly increased, with a maximum size of up to 512GB per processor and up to 1TB on multi-socket systems (512GB per processor). The enclave space can be shared, but adding more enclaves reduces the performance of each one. As of the 11th generation, Intel has stopped supporting SGX on desktop processors but continues to support it on Intel Xeon processors.

SGX's approach minimizes the Trusted Compute Base (TCB) by limiting it to just the CPU and the enclave (trusted part of the application), thus keeping the underlying operating system (OS) and hypervisor outside the TCB. In contrast, AMD's SEV approach, discussed in the next section, involves a larger TCB that includes the virtual machine OS but excludes the hypervisor. The downside of a larger TCB is that it creates more potential vulnerabilities, requiring users to examine more lines of code within the TCB to ensure its trustworthiness. On the other hand, encrypting the entire virtual machine's RAM allows for the seamless reuse of existing software without requiring modifications for secure enclave use.

C. AMD SEV

As previously noted, an alternative approach to SGX enclaves involves isolating and encrypting the memory of the entire virtual machine. This method, known as Secure Encrypted Virtualization (SEV) [11], was introduced by AMD in 2016 and has since been enhanced twice: with SEV-ES (Encrypted State) in 2017 [12], and SEV-SNP (Secure Nested Paging) in 2020 [13].

Figure 3 provides a high-level overview of the memory encryption architecture for an AMD-SEV virtual machine. AMD EPYC processors feature a dedicated ARM-based Platform Secure Processor (PSP), which is responsible for creating and storing symmetric cryptographic keys. When data is written to or read from the memory of a Secure Virtual Machine (SVM), the PSP intercepts the data and performs encryption or decryption accordingly. Figure 3 illustrates which parts of the computing resources on a remote server can be trusted by the AMD SEV virtual machine owner (shown in green) and which cannot (shown in red). Ensuring secure processing in such a potentially untrusted environment requires careful installation of the SVM, along with a verification

process (attestation) to confirm the integrity of the installation, both of which are detailed in this paper.

The performance overhead of on-the-fly encryption and decryption during memory writes and reads in the TEE is typically only a few percent compared to processing without encryption enabled [14]. This development led to a new area of research focused on protecting data in use, giving rise to confidential computing—the use of TEEs to safeguard data processed on untrusted hardware. Confidential computing is particularly relevant for outsourced computing in cloud environments, where the hardware is controlled by cloud providers rather than the users themselves. For a long time, data privacy concerns were a major barrier to the broader adoption of cloud services. Without additional safeguards, malicious cloud administrators or providers could access the user's resources and compromise sensitive data. Although users can secure their data stored on cloud disks by encrypting it with their keys, the data must be decrypted for processing. This is the point at which a malicious cloud provider could gain access to both the encryption key and the data by performing a memory dump. By using a hardware-based TEE, such risks can be mitigated, as it encrypts all sensitive user data within the TEE. The data in the server's RAM is encrypted using cryptographic keys stored and protected by the CPU. As a result, if a server administrator attempts to perform a memory dump, they will not be able to access the user's data.

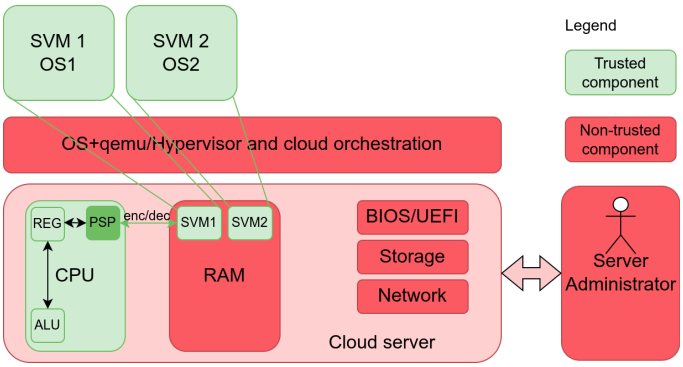

Fig. 3. High-level architecture of the AMD SEV VM protection

In contrast to SGX, the SEV approach has no memory limitations beyond the amount of available RAM for the virtual machine, allowing it to run any existing application within the secure VM. Even early comparison studies [15] found that the SGX approach is best suited for scenarios where security is paramount but performance is not a major concern, while the SEV approach is better suited for performance-intensive applications. Subsequent experiments have confirmed these findings. For example, Akram et al. compared SGX and SEV approaches using high-performance computing (HPC) benchmarks and concluded that SGX is not suitable for HPC. This is due to its limited secure memory size and complex programming model, which results in significant performance degradation compared to unencrypted execution [14].

D. Intel TDX

In 2023, Intel released Trust Domain Extensions (TDX) in their 4th generation of Xeon processors. TDX, similarly to the AMD SEV, allows deploying hardware-isolated virtual machines (VMs) called trust domains (TDs). TD VMs are isolated from the virtual machine manager (VMM), hypervisor, and other software on the host platform [16]. Intel TDX combines several technologies, including virtual machine extensions (VMX), instruction-set-architecture (ISA) extensions, Intel Total Memory Encryption Multi-key (Intel TME-MK), and CPU-attested software modules.

Figure 4. shows a high-level architecture of an Intel TDX virtual machine memory encryption. The initial version (TDX 1.0) provides memory confidentiality and integrity, address-translation integrity, CPU-state confidentiality and integrity, secure interrupt and exception delivery, and remote attestation. Figure 4. shows which parts of computing resources on a remote server can be trusted by the Intel TDX virtual machine owner (depicted in green) and which are not trusted (red). The main component is the Intel TDX module, which is provided and digitally signed by Intel. To host this module securely, but also to enforce security policies for TDs, a new SEcure Arbitration Mode – SEAM mode of the CPU is introduced. The SEAM Range Register (SEAMRR) is used to identify a memory space that is reserved for the Intel TDX module. Access to the SEAM memory range is allowed only to software that is executed inside the SEAM memory range. This memory range can be protected using AES in XTS mode with an ephemeral 128-bit memory encryption key. Intel TDX module is loaded into the SEAM memory range using a special Authenticated Code Module (ACM) called SEAM loader (SEAMLDR). This module can verify the digital signature on the Intel TDX module and load it into the SEAM memory range. Once loaded, the Intel TDX module provides an interface to VMM for creating, deleting, and executing TDs. When TD is created, the VMM provides memory pages for TD's code, data, and metadata.

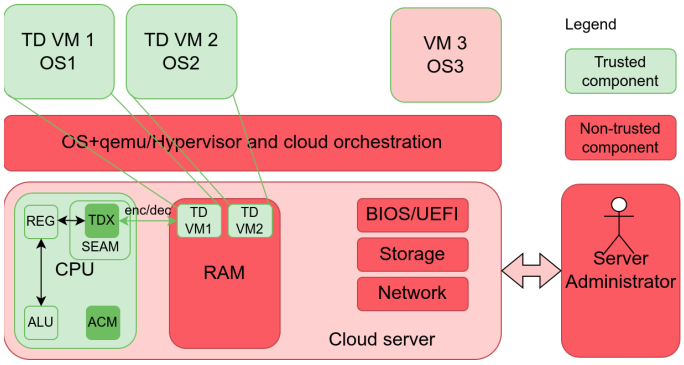

Fig. 4. High level architecture of the Intel TDX VM protection

The TME-MK engine is used to enable memory encryption and integrity for TD. The AES-XTS 128-bit memory encryption is used, and can be improved with SHA-3 based MAC for each cache line to protect cache integrity. Intel TDX module has the ability to create a CPU-generated unique and ephemeral AES-XTS 128-bit key for the TME-MK engine. The keys in the TME-MK are inaccessible by software or by using external interfaces to SoC.

Since TDs can access shared memory in order to communicate with untrusted entities, support for address translation is provided by creating two Extended-Page Tables (EPTs) for each active TD, one secure and one shared. The

Intel TDX module provides secure EPT management functions to the VMM. The CPU Translation Lookaside Buffer (TLB) has a tag that can identify the TD that created the translation. CPU-state confidentiality and integrity are provided by the Intel TDX module by encrypting memory pages provided by the VMM, which are used for Virtual-Machine-Control Structures (VMCS), state-save area, secure EPT, etc. This encryption is done using TD's private key.

VMX Advanced Programmable Interrupt Controller (APIC) virtualization and virtual interrupt architecture are used to enable interrupt and exception delivery for TDs. This mechanism is designed to avoid the need for modifications in the operating system in the TD. A similar concept to the previously described CPU-state confidentiality and integrity is used to secure the interrupt mechanism. Intel TDX module uses TD's private key to protect the virtual APIC page.

E. RISC-V enclaves

RISC-V architecture is able to limit the physical addresses accessible by software running on a hardware thread. An optional Physical Memory Protection (PMP) register unit provides machine-mode control registers to allow physical memory access privileges (read, write, execute) to be specified for each physical memory region. These capabilities are a basic primitive used by Keystone [18], an open-source framework for TEEs, or ZAYA Secure OS, which creates secure and isolated regions in the RISC-V environment. The architecture of the Keystone framework is very similar to the first two Arm TrustZone versions. However, the open-source nature of RISC-V architecture enables more flexibility in the organization of the secure enclaves. Previously mentioned platforms were used mainly among researchers to provide specific security functions [19].

The Keystone framework does not restrict the use of memory encryption. However, due to the lack of a hardware encryption engine, its authors used software-based memory encryption, which has a penalty on performance [18]. There are similar attempts from the academia and design contests to create TEE hardware extensions on RISC-V [20]. Since August 2022, the RISC-V community has established the Application Platform - Trusted Execution Environment task group [21][22] to define the reference architecture for confidential computing on RISC-V platforms. This task group is working on Confidential VM Extension I/O (CoVE-IO) [23], which will bring to the RISC-V virtual machine-based TEE, similar to the other previously mentioned.

F. GPU enclaves

Confidential computing concepts also appeared in the Graphics Processing Unit (GPU) market. In 2022, NVIDIA created the first GPUs based on Hopper architecture (H100), which supported, similarly to AMD SEV and Intel TDX, the creation of confidential and encrypted virtual machines. In this case the use is tailored for massive calculations that are common in deep learning or large language model training. This allows the protection of the user data and AI model weights against the infrastructure and the GPU provider.

G. IETF standardization

In 2017, IETF established a Trusted Execution Environment Provisioning (teep) work group, which aims to develop an application layer protocol providing TEEs with the lifecycle management of trusted applications and security domain management. The group has so far published one informational RFC that describes the architecture for TEE provisioning [24]. This document discusses the motivation for designing and standardizing a protocol for managing the lifecycle of Trusted Applications running inside such a TEE.

III. ENCRYPTED TEE PERFORMANCE

In this section, we present some performance experiments conducted at the Laboratory for Information Security of our faculty, aimed at showcasing the potential benefits of using a confidential computing paradigm for complex data processing tasks. The purpose of these measurements is not to provide a comprehensive analysis of the underlying platforms and technologies but to highlight the differences when compared to traditional approaches.

We conducted two sets of experiments, both using the SciML-bench tool [25], which enables a comparison of the performance across all stages of the machine learning data processing pipeline and offers a more versatile set of features compared to other scientific machine learning benchmarking methods [26]. The secure virtual machine used for these tests was installed on a server equipped with an AMD EPYC 7313P 16-Core processor, 32GB of RAM, and a 2TB SSD, located in our data center. This processor is part of the 3rd generation of AMD EPYC processors and supports SEV, SEV-ES, and SEV-SNP. The secure virtual machines were allocated 16 cores and 16 GB of RAM. We tested two different SciML-bench use cases with real-world datasets: improving the signal-to-noise ratio of electron microscope images (*em_denoise* - Test 1) and searching for patterns in X-ray images (*dms_structure* - Test 2). The dataset sizes for these two tests were 5GB and 8.6GB, respectively. Table 1 presents the performance slowdown observed during the training phase of the machine learning pipeline due to the use of different SEV technologies. The full results of these experiments can be found in our previous study [27].

TABLE I. TRAINING TIME PERFORMANCE EVALUATION ON ETF SERVER [27]

	SEV			SEV-SNP		
Test	Slow down [%]	CPU load [%]	RAM usage [%]	Slow down	CPU load [%]	RAM usage [%]
1	3.328	87.372	32.901	8.371	86.003	33.486
2	7.954	90.744	58.155	14.435	88.233	59.864

As it can be seen from the experimental results, the slowdown is not the same for the tested algorithms and probably depends on the memory access patterns of the training algorithm, which should be explored further. However, in all cases, the performance penalty was less than 15%, which is negligible compared to the other solutions for privacy-preserving data processing using pure cryptographic methods, mentioned in the introduction.

The second set of experiments was executed on the Google Cloud Platform (GCP) infrastructure. Table 2 summarizes the

results of the same *em_denoise* model training task done on a virtual machine without confidential computing and on a virtual machine with AMD SEV-SNP and Intel TDX switched on. It shows a relative slowdown of the model training when confidential computing techniques are used. AMD SEV-SNP was tested on the *n2d-standard-4* machine, with 16GB of RAM and four vCPU. Intel TDX was tested on the *c3-standard-4* machine, with 16GB of RAM and four vCPUs.

TABLE II. TRAINING TIME PERFORMANCE EVALUATION IN GOOGLE CLOUD

TEE	Slowdown [%]	RAM Usage [%]
SEV-SNP	3.82	5.30
TDX	14.36	4.80

Again, the results show up to 15% slowdown and similar performance penalty due to the data encryption. These results should not be used to conclude that AMD SEV-SNP is more performant than Intel TDX, as for such a conclusion, more experiments are needed with different datasets and training models, as well as with a better insight into the computing capabilities of the processor cores. However, the results further confirm that the performance penalty of using confidential computing is significantly lower than when traditional homomorphic encryption or similar algorithms are used. This is the main driver for the research in this field and the fast adoption of the technologies in commercial clouds.

IV. REMOTE ATTESTATION

The previous section clearly shows the main incentive for using encrypted TEEs – minimal performance loss while enabling privacy-preserving computations. However, as described in sections II.C. and II.D., servers with processors capable of encrypting memory content on-the-fly are located in a completely hostile environment. These processors are installed in servers whose other hardware (e.g., BIOS, disk, network cards, etc.) is generally not trusted, with hypervisor software which is not trusted (e.g., can be modified by the server owner) and maintained by the company and administrators who are not trusted. How can one in these circumstances still be convinced that the data that is sent to such a remote server will be processed by a processor with the required capabilities, that these capabilities are switched on on the processor, and that the configuration is such that it ensures data privacy? The key process for ensuring trust in the data processing in case of confidential computing is the remote attestation.

Remote attestation is not exclusively used in confidential computing. It is a general process through which one system can know and prove that another system can be trusted. IETF summarized different remote attestation approaches in [28], a result of the Remote ATtestation ProcedureS (rats) work group. Besides the use to prove that the remote system is capable of confidential computing and able to protect the data or code privacy, other use cases for remote attestation are:

- Network endpoint assessment, where one network device (e.g. router) might want to check the identity and version information about the hardware and software on the machines attached to its network. It might do the admission to the network based on the received and verified data and let only those devices that meet some criteria (e.g. antivirus present and updated and so on),
- Critical Infrastructure Control, where one organization managing such an infrastructure wants to ensure that only authorized code and users can control some critical process and that the process is protected from unauthorized manipulation or other threats,
- Hardware Watchdog, where one might implement a forced hardware reset if the device is locked by a malware and does not pass the attestation, or
- Fast IDentity Online (FIDO) Biometric Authentication, where some website or mobile application might verify that the biometric authentication comes from a trusted device.

The document further establishes and defines key roles and defines different architectures and communication patterns of the remote attestation. The two most essential roles are:

- *Attester* – a device which must provide the Evidence which enables inference of the extent to which the Attester is considered trustworthy. In the case of confidential computing, this is the processor with TEE capability.
- *Verifier* – an entity that appraises the validity of Evidence about an Attester and produces Attestation Results.

There might be other entities in the attestation process as well, depending on the attestation pattern, as described in [28]. For confidential computing use-cases and ensuring data-in-use protection and data privacy, specific cryptography-based attestation protocols are established as described in the subsequent sections. In the remainder of this section we will focus on the remote attestation for AMD SEV and Intel TDX technologies, as two technologies which are used in today's confidential computing offers by major cloud providers.

A. AMD SEV Attestation

Remote attestation is a process that proves to the SVM user that SEV protection is in place and that the virtual machine was not a subject to manipulation. Before sending secrets to the SVM, the SVM user must verify the attestation information. While this process is the same for the SEV and the SEV-ES, it changed with the SEV-SNP. In the following subsections, the SEV and the SEV-SNP attestations are described, and their differences are highlighted.

1) SEV and SEV ES Attestation

SEV and SEV ES Attestation processes involve three main actors: 1) AMD which proves that processors have the SEV capability and that it is switched on for each particular virtual machine, 2) the *Platform Owner* which owns a server and verifies its authenticity and 3) the secure virtual machine user - *Guest Owner* who verifies the installation (*Verifier*). The first

operation in the process of the remote attestation is the creation of a digital certificate chain that is used to verify the platform on which the SVM will be executed. The keys that are parts of the certificate chain are:

- AMD Root Key (ARK) - an RSA 2048 asymmetric key pair,
- AMD SEV Signing Key (ASK) - an RSA 2048 asymmetric key pair,
- Chip Endorsement Key (CEK) - an Elliptic Curve Digital Signature Algorithm (DSA) P-384 asymmetric key pair,
- Platform Endorsement Key (PEK) - an Elliptic Curve DSA curve P-384 asymmetric key pair,
- Owner Certificate Authority (OCA) - an Elliptic Curve DSA curve P-384 asymmetric key pair, and
- Platform Diffie-Hellman Key (PDH) - an Elliptic Curve Diffie-Hellman (DH) curve P-384 asymmetric key pair.

At the root of the certificate chain is ARK. The ARK private key is used to sign ASK. ARK and ASK public key certificates can be downloaded from AMD Key Distribution Server (KDS). The ASK private key is then used to sign the CEK. CEK is unique for each AMD SEV processor, so by signing CEK with ASK, AMD enables users to verify that the platform is run on an authentic AMD SEV processor produced by AMD. PEK is then signed with the private key of CEK and with the private key of OCA. The OCA belongs to the Platform Owner. By signing the PEK with OCA the Platform Owner enables Guest Owner to verify the authenticity of the owner of the platform. PEK private key is used to sign PDH. PDH is available to the user of the secure VM (Guest Owner) at the beginning of the remote attestation process, and used for further key exchange.

The Platform Owner should export the PDH certificate and the whole certificate chain (Figure 5) to the Guest Owner. The Guest Owner needs to do the following in order to be sure that the virtual machine is brought up in the enclave and that his/her code/data will run safely inside an SVM. These steps are:

- Fetching Platform Owner's certificate chain and the code of a virtual machine firmware (in the case of AMD SEV, OVFM (Open Virtual Machine Firmware) is the firmware that is supported and recommended), and
- Verification of the certificate chain. Verification is done by using the AMD's and the Platform Owner's certificates and respective Certificate Revocation Lists.

The next step for the Guest Owner is to exchange the master secret with the Platform Owner. The Guest Owner generates a new Elliptic Curve Diffie-Hellman key pair and sends the public key, along with a *nonce* value (N) to the Platform Owner. Now both the Platform Owner and the Guest Owner can calculate the same *secret value* (Z) from PDH and this new key using the ECDH algorithm. For this algorithm the Guest Owner is using the generated ECDH private key and the PDH public key and the Platform Owner is using the received ECDH public key and the PDH private key. Then the *master secret* (M) can be created using the counter mode, as a key derivation function based on Z and N in a HMAC-SHA-256 pseudorandom function [29]. Finally, the Z value is deleted and the value M can be used for further communication.

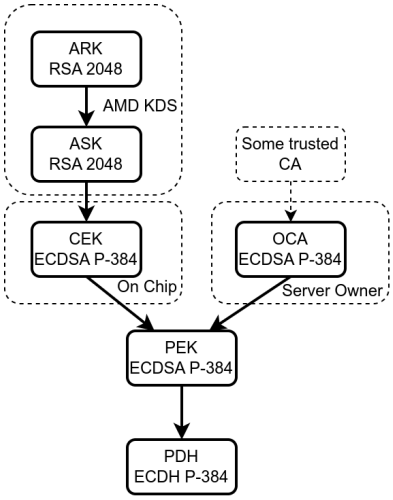

Fig. 5. AMD SEV certificate chain

The *master secret* M is used to derive other keys that are needed to secure confidentiality and integrity between the *client* (Guest Owner) and the *server* (Platform Owner). Guest Owner first creates the following keys:

- Transport Integrity Key (TIK), which is used for integrity protection using HMAC-SHA-256 and
- Transport Encryption Key (TEK), which is used for confidentiality protection using AES-128.

Both keys are generated randomly from the entropy source. Another pair of keys that are also used in establishing the secure communication between the client and the server, but are calculated on both end from master secret using HMAC-based Key Derivation Function, are:

- Key Integrity Key (KIK) - which is used for integrity protection using HMAC-SHA-256 and
- Key Encryption Key (KEK) - which is used for confidentiality protection using AES-128.

Both Guest and Platform Owner can create the same KEK and KIK from the master secret. Then, the client (Guest Owner) encrypts TEK and TIK keys using KEK and protects their integrity using KIK and sends them to the Platform Owner. TEK and TIK keys are be used to protect further secure communication between the Platform and Guest Owner.

After all previous steps, the client now can perform remote attestation by obtaining the *measurement* from the server and making sure that the measurement matches the expected value. The *measurement* is calculated as an HMAC over seven concatenated fields, with the TIK key used for calculation and TEK key. Full description of these fields can be found in [27]. We want to emphasize here the following fields: GTCX.POLICY, GTCX.LD - SHA256 output digest over OVMF UEFI used during VM launch, and *mnonce* - a random value generated by the PSP firmware at the server CPU.

GTCX is the Guest Context and the GTCX.POLICY is the value used by the client that defines certain virtual machine

configuration options like enabling SEV-ES or debugging mode. UEFI used during the SVM launch must be built as a stateless firmware file that does not use a non-volatile random-access memory store. AMD supports OVMF at the moment. The server has to send the UEFI image and the *mnonce* to the client so that he/she can calculate the expected value of the measurement. If the calculated value of the measurement matches the value received measurement value from the server, then the remote attestation is successful, and Guest Owner can be sure that: 1) the virtual machine is brought up using the desired context, 2) that this specific UEFI image was used to bring up the SVM, 3) that the virtual machine is running on the CPU with AMD SEV capabilities which are switched on. However, with this process Guest Owner cannot be sure about the integrity of the operating system on the secure virtual machine.

Hypervisor incorporates the attestation process into the boot process of a secure virtual machine. In case of the incorrect measurement, the boot process will not complete. If the client-server secure communication is successfully established then the machine will be booted up and the entire VM content will be encrypted using the VM Encryption Key (VEK), which is an AES-128 symmetric key randomly generated and available only to the processor for encryption/decryption.

In addition to the calculation of the UEFI digest during the attestation process, as described above, QEMU software [30] extended the input to the HMAC function to include additional elements and supports the following measurement: *hash(firmware_blob || kernel_hashes_blob || vmsas_blob)*, where *firmware_blob* is the content of the entire firmware flash file (for example, OVMF.fd), *kernel_hashes_blob* which includes the hashes of the guest operating system kernel, *initrd*, and *cmdline* that are passed to the guest, and *vmsas_blob* is the concatenation of all Virtual Machine State Areas (VMSA) of the guest vCPUs. VMSA is a crucial data structure used for managing the state of a virtual machine. This means that with SEV and SEV-ES attestation it is possible to include some parts, but whole guest operating system into the measurement.

B. SEV-SNP Attestation

The process of attestation in the SEV-SNP involves the same actors as in previous SEV versions. However, the SEV-SNP introduces two new keys for the attestation report signing: the Versioned Chip Endorsement Key (VCEK) and the Versioned Loaded Endorsement Key (VLEK). VCEK is a private Elliptic Curve DSA key constructed using the CEK key hashed together with the version numbers of all TCB components. VCEK is unique to each AMD chip with the specific TCB. The VCEK is signed with ASK and ARK keys to authenticate the attestation report. TCB components are:

- Microcode of the CPU,
- SNP firmware,
- PSP operating system, and
- PSP bootloader.

VLEK is an alternative to VCEK and can replace it in the attestation report signing. VLEK is also an Elliptic Curve DSA P-384 signing key signed by AMD. Unlike VCEK which gets the unique seed from the chip, each Platform Owner needs to enroll with AMD and download a unique VLEK seed from the AMD Key Derivation Service (KDS). To use the VLEK, the Platform Owner needs to send to the KDS the current TCB version and the chip ID of the processor that the Platform Owner is using. The AMD KDS calculates so-called VLEK *hashstick* from the delivered TCB and the Platform Owners VLEK seed and wraps it with a AES-256-GCM key derived from the chip ID information. The Platform Owner then provisions the platform with the wrapped VLEK hashstick. Once the VLEK hashstick is loaded, the firmware can use the VLEK as a replacement for the VCEK. VLEK functionality is introduced in the latest available version of the SEV SNP Firmware ABI Specification, and its use is not yet available as a service from AMD [31].

The Guest Owner can provide the Platform Owner with an identity (ID) block, which is submitted to the hypervisor when launching the Secure Virtual Machine (SVM). The ID block is a data structure that includes the expected measurement launch digest, the Guest Owner's policy, and the cryptographic signature of the ID block itself. This block is signed with a private ECDSA ID key. If the measured value of the ID block differs from the calculated measurement at launch, the SVM launch will fail. The Guest Owner sends the ID block and the public ID key required for signature verification to the hypervisor, which in turn passes the public ID key to the SNP firmware.

The SEV-SNP guest policy functions similarly to the guest policy in the previous SEV versions. It consists of binary flags that define how the SVM should be booted. This policy includes details like the debug flag and the minimum major and minor versions of the SNP firmware.

A notable change in SEV-SNP is the modified process for fetching and sending the attestation report to the Guest Owner. The attestation report is obtained from the SVM using a kernel driver (sev-guest driver) that communicates with the SNP firmware to retrieve the report. The sev-guest driver sends encrypted messages, using the VM Platform Communication Keys (VMPCKs), via the hypervisor to the SNP firmware. During the SVM launch, a special secret memory page containing the VMPCKs is inserted into the SVM. The firmware decrypts the messages, and if the message is an attestation report request, it sends the report back to the hypervisor, which then forwards it to the guest.

Key fields in the SEV-SNP attestation report include the measurement of the launched SVM, the policy provided by the Guest Owner, TCB information, the digest of the ID key that signed the ID block, and the report data provided by the guest. The report data is a 512-bit data block provided to the firmware at the time of the attestation report request. The firmware does not interpret the report data but instead includes it in the attestation report as is. This allows the report data to carry relevant information to the Guest Owner. For instance, the report data could contain the public key of a generated key pair, which could then be used to establish a trusted communication channel with the Guest Owner. Upon receiving the attestation report, the Guest Owner verifies it by checking the signature and ensuring the information is valid.

C. TDX Attestation

The TDX remote attestation process aims to convince the user of the TD virtual machine (called challenger in this context but representing the relying party according to the RFC9334) that the software is running inside a TD on an Intel TDX system at a required security level [16][17]. The process is partially based on the existing Intel infrastructure developed for the attestation process of the Intel SGX-capable processors.

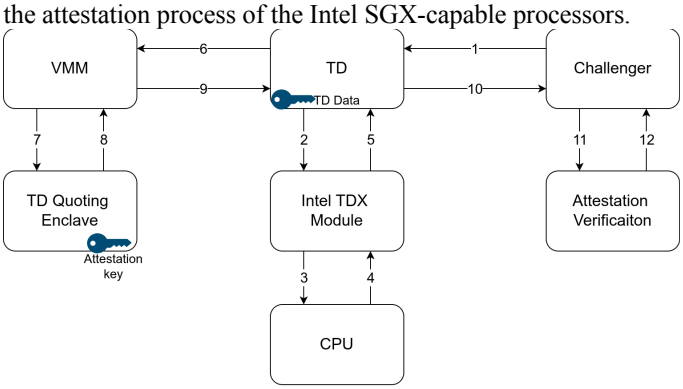

Fig. 6. Remote attestation process for Intel TDX

Figure 6. depicts the remote attestation process for Intel TDX. The process starts with the challenger sending an attestation request to the TD (step 1). The TD requests the Intel TDX module to provide the TD report that includes attestation information (step 2). The Intel TDX module then invokes the SEAMREPORT instruction that asks the CPU to generate a report (step 3). CPU prepares the report and sends it to the TDX module (step 4). The final report that the TDX module returns to the TD (step 5) contains: TD-provided data, measurements of the TD, and security version numbers (SVNs) of all elements in TDX TCB. Once TD receives this report it sends it to VMM (step 6). The VMM sends the report to the TD quoting enclave (step 7). TD quoting enclave verifies the MAC on the report and, if verified, converts the report into a Quote by signing it with the TD's asymmetric attestation key (step 8). This Quote is returned to TD (step 9), and TD sends it to the challenger (step 10). The challenger can perform quote verification. To support the attestation process, the entire Intel infrastructure used for SGX attestation is adapted in order to be used for TDX, as well.

TD-provided data is generated by the software in the TD, and it can include a public key for secure communication with the relying party. Measurements of the TD are kept in measurement registers: TDMR (TD measurement register) and RTMR (runtime extendable measurement register). These two registers are initialized by the Intel TDX module in the process of TD creation. The TDMR register contains measurements of the initial pages added to the TD and metadata associated with these pages. The RTMR registers contain measurements of additional code and data at runtime.

Figure 6 shows which components are part of the Intel TDX TCB. For each component, SVN is assigned. The SEAMREPORT instruction that can be invoked by the Intel TDX module can generate a report structure that includes SVNs of the TDX TCB components. This structure's integrity is protected using MAC, and another instruction, EVERIFYREPORT2 is provided to be used by the enclaves to verify the MAC. TCB is considered to be up to date if all components have SVNs greater than the threshold.

TD quoting enclave is an Intel SGX enclave used for remote attestation. It is used to generate a Quote based on the report it receives and verifies. For signing the generated Quote, it uses a TD's asymmetric attestation key. This key is an Elliptic-Curve-DSA-based key representing the TDX TCB version. This is part of Intel TDX design that enables the updates in components that are part of the TCB. This process is called TCB Recovery. Once the update is completed, a new TD asymmetric attestation key will be generated and used for attestation.

Intel SGX Data Center Attestation Primitives (DCAP) are used for the certification of TD quoting enclaves. It provides an additional enclave named Provisioning Certification Enclave (PCE) that acts as a Certificate Authority (CA) for TD quoting enclaves that are running on the same platform. TD quoting enclaves generate attestation keys and sends them to PCE. Once the PCE authenticates the received request, it creates a certificate-like structure that associates TD quoting enclave and its attestation key. This structure is then signed by the Provisioning Certification Key (PCK) that is TCB specific. Intel publishes certificates and certificate revocation lists (CRLs) for PCKs.

D. Attestation through the TLS

The first attestation processes were implemented to be executed during the boot process, blocking it if the settings do not pass the measurement. Also QEMU software enables obtaining the attestation report through the QMP management protocol. This is sufficient for the secure virtual machine user who uses the virtual machine in shell mode, as the user can have full control over the virtual machine.

However, for cloud-based web applications, it would be very useful if the web application users could verify that the application is run on a secure virtual machine, meaning that the data processed in it could be privacy protected. There is an ongoing branch of work in IETF on the standardization of the Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) [32]. In both cases, the remote attestation process is done during the TLS establishment phase through the protocol extensions in the TLS 1.3 handshake. If the web application is running on a secure virtual machine with confidential computing capabilities turned on, and the TLS connection enhanced with the remote attestation extensions is established, this can mean that the data sent to the application might be private. Of course, there are other aspects of ensuring that the web application processes data privately that depend on how the application works. The application should not store the data it processes unencrypted on disks in the cloud at any moment, and all the data processed in it must be kept in memory at all times. Some guidelines for programming such applications are given in our previous work [33].

V. TEE Solution Security

The main limitations of using confidential computing in the cloud stem from the technology's readiness level and some security concerns related to existing trusted execution environments (TEEs). Similar conclusions were drawn in a paper by Chen [34]. Research papers and vulnerability reports

detailing potential attacks on TEE-enabled processors indicate that the technology is still evolving. For instance, Wilke et al. [35] presented a known plaintext attack on AMD SEV or SEV-ES. In this attack, an attacker who controls the hypervisor and has knowledge of parts of the Secure Virtual Machine (SVM) kernel can successfully determine the tweak values used in AES XE mode. This allows the attacker to insert random 16-byte blocks into the encrypted memory, ultimately leading to the execution of arbitrary code and compromising confidentiality. This attack takes advantage of the lack of memory integrity in the early versions of AMD SEV. Two additional attacks exploit the absence of memory integrity and the unprotected content of CPU registers in the original AMD SEV, as described in [36] and [37]. Both attacks manipulate the Virtual Machine Control Block (VMCB). In the first, the memory content is transferred to the instruction pointer register (RIP) in the VMCB, and an encryption/decryption oracle is created, causing memory leakage. The second attack involves an attacker observing the VMCB content to reconstruct the executed code and using instruction-based sampling (IBS) to identify the applications running inside the VM. Other types of attacks have also been identified. For example, Mengyuan et al. [38] exploited the improper use of the address space identifier (ASID), allowing the attacker to link the victim's ASID to their VM, leading to a loss of confidentiality. This attack requires control over the hypervisor and is possible for SEV and SEV-ES. Even more targeted attacks, such as [39-41], focus on address-translation vulnerabilities, using page faults to detect memory locations or the hypervisor's control over the nested page table (NTP) to detect and modify the guest physical address (GPA), all resulting in a loss of confidentiality. Finally, hardware-based attacks, like the one in [42], demonstrated how a voltage glitching attack on any SEV version could allow an attacker to load custom firmware and execute it on the platform's secure processor (PSP).

As this overview shows, SEV technologies have evolved rapidly, with three versions released in just four years. These changes were primarily driven by the discovery of vulnerabilities. In the original SEV approach, memory encryption kept the hypervisor outside the Trusted Compute Base (TCB), but some secret information could still be accessed through CPU registers when the VM stopped running. This issue was addressed by AMD in the SEV-ES release, which encrypted CPU registers when the VM was paused. Both SEV and SEV-ES suffered from a significant design flaw: the lack of memory integrity. While researchers have proposed custom solutions, such as the one in [43], AMD addressed this issue in the SEV-SNP release. However, certain attacks, such as those in [42], remain effective across all generations of SEV. Given this ongoing evolution, it is expected that future updates will continue to improve the technology, eventually leading to a mature and fully secure solution for secure virtual machines, with all known vulnerabilities eliminated. Until that time, it is important to recognize the limitations and current capabilities of existing technologies.

In April 2023, Google published a security review of Intel TDX [44], which was performed to assure cloud users and providers that this technology is secure and to determine the threat model for the technology. The result of this review was that all 10 identified security issues were fixed by Intel prior to production release, 9 of them were fixed in TDX code.

In 2024, two papers from ETH Zurich revealed that the use of malicious interrupts could be leveraged to compromise the operation of the confidential virtual machine, managing to gain full control over the machine. Heckler is an attack in which the malicious hypervisor injects non-timer interrupts to break the confidentiality and integrity of the confidential virtual machines [45]. When the interrupt handlers that have global effects are used, it is possible to manipulate a confidential virtual machine register states to change the data and control flow. With both AMD SEV-SNP and Intel TDX, the authors demonstrated that they managed to bypass authentication with OpenSSH and sudo operations. This vulnerability was reported to Intel and AMD as CVE-2024-25744 for int 0x80. It was mitigated with a kernel patch for SEV-SNP and TDX, but at the moment of writing the paper, the attack remained unmitigated for other interrupts. In 2024 there were 8 other registered TDX vulnerabilities registered in the CVE database, all resolved in the Linux kernel.

Similarly to the previous, the same authors found another vulnerability that exploits interrupts. AMD introduced a new exception, #VC, to facilitate the communication between the VM and the untrusted hypervisor. With the WeSee attack [46], the hypervisor can inject the malicious #VC into a victim's secure virtual machine CPU to compromise the security guarantees of AMD SEV-SNP. Specifically, WeSee injects interrupt number 29, which delivers a #VC exception to the VM which then executes the corresponding handler that performs data and register copies between the VM and the hypervisor. WeSee shows that using well-crafted #VC injections, the attacker can induce arbitrary behavior in the VM which can include arbitrary reads, writes, and code injection. The authors demonstrated three use cases in which they managed to leak kTLS keys for NGINX, bypass the firewall, and obtain a root shell. This vulnerability was reported to AMD and assigned vulnerability identification CVE-2024-25742. In its response AMD-SB-3008, AMD confirmed the vulnerability but stated that they believe that this vulnerability lies in the Linux kernel implementation of SEV-SNP and mitigations addressing some of the vulnerability should be addressed there and also stated that AMD supports additional hardware security features that are designed to protect against the reported attack that is not currently supported in Linux. After that, changes were made to the Linux kernel, hardening the #VC instruction emulation. In 2024, there were 3 other registered SEV-SNP-related CVE vulnerabilities. For example, for the TDX, two vulnerabilities were resolved by patching the Linux kernel, and one (CVE-2024-56161) required the update of the system BIOS image. Two interesting attacks on Intel TDX are reported in [47]. The reason why these attacks are interesting is that they belong to a group of side-channel attacks for which the Intel TDX has built-in defense mechanisms, which the authors of the paper successfully circumvented.

Table 3. shows the number of vulnerabilities recorded in the CVE database per year and per technology. The row for AMD SEV shows vulnerabilities in all SEV versions. For all the technologies, it can be observed that upon the introduction of a new technology (e.g., SEV SNP in 2020), there is an increase in the number of discovered vulnerabilities. The key reason is the fact that the whole system is complex and requires changes in many places in the hardware and software stack (BIOS, operating systems, hypervisor), and these changes have to be made and verified in all these systems. The previous examples

show that this whole field of confidential computing is still a work in progress. As a field of active development, there are sometimes gaps between the available hardware features and software implementations in hypervisors and operating systems supporting confidential computing. Also, the technologies are still maturing, with vulnerabilities being removed with every new release, which was seen in the AMD SEV evolution towards SEV-SNP.

TABLE III. NUMBER OF VULNERABILITIES RECORDED PER YEAR

	2017	2018	2019	2020	2021	2022	2023	2024
SEV			1	2	7	2	2	5
TDX							5	10
SGX	1	3	7	7	9	8		

Finally, we need to stress that all TEE technologies used today rely on remote attestation based on cryptographic algorithms that are not quantum computer resistant. Given the typical operational lifetime of servers procured today, there is a possibility that during that time, the key exchange and signature mechanisms used in their processors will become obsolete. However, we believe that swapping to some post-quantum mechanisms is not going to present a significant burden for processor manufacturers.

VI. CONCLUSION

This paper summarizes recent developments in the field of confidential computing, ignited by the emergence of new processor features that isolate and protect portions of the computer memory. The overview of the processor feature development shows that trusted execution environments converge towards protected and encrypted virtual machines, as a method that provides simple existing software integration into the confidential computing. Performance shown in computing over complex datasets is a clear motivation fo further research in the field and the use in production.

On the other side, operating secure enclaves in the untrusted environment requires the use of complex remote attestation mechanisms to ensure the users that their data and code are going to be operated in a truly private manner. It requires from the administrators a high level of understanding of many aspects of operating virtual machines, including the booting process, operating system, network and the confidential computing principles. Lack of understanding in any of these might lead to non-secure deployments and a false sense of security. Further, application developers have to be aware of the fact that the application is run in the cloud environment, and special care (protection at rest) is needed if the data is to be stored on cloud disks at any moment.

The field of confidential computing is still a moving target, with new mechanisms appearing daily. This has an impact on the development of related software (e.g., operating systems, hypervisors, UEFI code) which does not follow quickly enough new hardware capability development. This has led to the discovery of vulnerabilities in existing technologies. Typically, all these vulnerabilities are quickly resolved. Although there are vulnerabilities, confidential computing is relatively young and is expected to stay as a concept. It is already used and offered as a service in the largest existing cloud platforms and will be used. Also, it is expected that there will be newer versions in which some concepts will be further refined and strengthened.

ACKNOWLEDGMENT

This work was financially supported by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia under contract number: 451-03-136/2025-03/200103.

REFERENCES

[1] Rashid, F. Y. (2020). The rise of confidential computing: Big tech companies are adopting a new security model to protect data while it's in use - [News]. IEEE Spectrum, 57(6), 8–9. https://doi.org/10.1109/MSPEC.2020.9099920

[2] Cloud Computing Market Size, Share & Industry Analysis, By Type (Public Cloud, Private Cloud, and Hybrid Cloud), By Service (Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS)), By Enterprise Type (SMEs and Large Enterprises), By Industry (BFSI, IT and Telecommunications, Government, Consumer Goods and Retail, Healthcare, Manufacturing, and Others), and Regional Forecast, 2024-2032, https://www.fortunebusinessinsights.com/cloud-computing-market-102697, accessed on March 17th 2025.

[3] Evans D., Kolesnikov V., Rosulek M., A Pragmatic Introduction to Secure Multi-Party Computation, Now Foundations and Trends, 2018. https://ieeexplore.ieee.org/document/8584398

[4] Yang, Y., Huang, X., Liu, X., Cheng, H., Weng, J., Luo, X. and Chang, V., 2019. A comprehensive survey on secure outsourced computation and its applications. IEEE Access, 7, pp.159426-159465.

[5] Mo J., Gopinath J., Reagen B.. 2023. HAAC: A Hardware-Software Co-Design to Accelerate Garbled Circuits. In Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA '23). Association for Computing Machinery, New York, NY, USA, Article 10, 1–13. https://doi.org/10.1145/3579371.3589045

[6] Kairouz et al. (2021), "Advances and Open Problems in Federated Learning", Foundations and Trends® in Machine Learning: Vol. 14: No. 1–2, pp 1-210. http://dx.doi.org/10.1561/2200000083

[7] Yiu J., ARMv8-M Architecture Technical Overview, White Paper, 10-Nov-2015

[8] Parker J. Arm's Dynamic TrustZone technology, June 2021, https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/introducing-arms-dynamic-trustzone-technologyIntroducing (Accessed on March 30th 2025)

[9] Apple Secure Enclave, December 2024, https://support.apple.com/guide/security/secure-enclave-sec59b0b31ff/web (Accessed on March 30th 2025)

[10] Johnson S., Makaram R., Santoni A., Scarlata V., Supporting Intel SGX on Multi-Socket Platforms, Intel, https://www.intel.com/content/www/us/en/content-details/843058/supporting-intel-sgx-on-multisocket-platforms.html

[11] AMD Secure Encrypted Virtualization (SEV), https://www.amd.com/en/developer/sev.html, Accessed on March 19th 2025.

[12] Kaplan, D., 2017. Protecting VM register state with SEV-ES. White paper.

[13] SEV-SNP, AMD, 2020. Strengthening VM isolation with integrity protection and more. White Paper, January, p.8.

[14] Akram, A., Giannakou, A., Akella, V., Lowe-Power, J. and Peisert, S., 2021, May. Performance analysis of scientific computing workloads on general purpose TEEs. In 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS) (pp. 1066-1076). IEEE.

[15] Mofrad S, Zhang F, Lu S, Shi W. A comparison study of intel SGX and AMD memory encryption technology. In: Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy. 2018

[16] Intel, Intel Trust Domain Extensions (Intel TDX), Whitepaper, February 2023, available at: https://cdrdv2.intel.com/v1/dl/getContent/690419, Accessed March 2025

[17] Sardar, M.U., Fossati, T., Frost, S. and Xiong, S., 2023. Formal specification and verification of architecturally-defined attestation mechanisms in arm cca and intel tdx. IEEE Access, 12, pp.361-381.

[18] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, D. Song. 2020. Keystone: an open framework for architecting trusted execution environments. In Proceedings of the Fifteenth European Conference on Computer Systems (EuroSys '20). Association for Computing Machinery, New York, NY, USA, Article 38, 1–16. https://doi.org/10.1145/3342195.3387532

[19] V. Ushakov et al., "Trusted Hart for Mobile RISC-V Security," 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Wuhan, China, 2022, pp. 1587-1596, doi: 10.1109/TrustCom56396.2022.00228.

[20] J. Ma et al., "Construction of RISC-V Lightweight Trusted Execution Environment Based on Hardware Extension," 2021 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS), Shenyang, China, 2021, pp. 237-242, doi: 10.1109/ICPICS52425.2021.9524261.

[21] Confidential Computing on RISC-V platforms - Application-Processor Trusted Execution Environment (AP-TEE) TG, https://github.com/riscv-non-isa/riscv-ap-tee/blob/main/charter.adoc

[22] RISC-V AP-TEE TG - https://github.com/riscv-non-isa/riscv-ap-tee

[23] Confidential VM Extension I/O (CoVE-IO) for Confidential Computing on RISC-V platforms Editor - Samuel Ortiz, Jiewen Yao, RISC-V AP-TEE-IO Task Group, Version 0.1, 2023/04: https://lists.riscv.org/g/tech-ap-tee-io/attachment/47/0/riscv-cove-io-0.1.0_comments_QCOM.pdf

[24] Pei, M., Tschofenig, H., Thaler, D., and D. Wheeler, "Trusted Execution Environment Provisioning (TEEP) Architecture", RFC 9397, DOI 10.17487/RFC9397, July 2023, <https://www.rfc-editor.org/info/rfc9397>.

[25] Thiyagalingam J., Papay J., Leng K, Jackson S, Shankar M., Fox G., Hey T., {SciML-Bench: A Benchmarking Suite for AI for Science, 2021, https://github.com/stfc-sciml/sciml-bench

[26] Thiyagalingam, J., Shankar, M., Fox, G. et al. Scientific machine learning benchmarks. Nat Rev Phys 4, 413–420 (2022). https://doi.org/10.1038/s42254-022-00441-7

[27] Miladinović, D.; Milaković, A.; Vukasović, M.; Stanisavljević, Ž.; Vuletić, P. Secure Multiparty Computation Using Secure Virtual Machines. Electronics 2024, 13, 991. https://doi.org/10.3390/electronics13050991

[28] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/info/rfc9334>.

[29] Chen, L. Recommendation for Key Derivation Using Pseudorandom Functions; US Department of Commerce, National Institute of Standards and Technology: Gaithersburg, MD, USA, 2008.

[30] QEMU, A generic and open source machine emulator and virtualizer https://www.qemu.org/

[31] SEV Secure Nested Paging Firmware ABI Specification. Available online: https://www.amd.com/system/files/TechDocs/56860.pdf (accessed on 30 March 2025).

[32] H. Tschofenig, Y. Sheffer, P. Howard, I. Mihalcea, Y. Deshpande, A. Niemi, T. Fossati, Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS), Internet draft, 2024-10-21, https://datatracker.ietf.org/doc/draft-fossati-tls-attestation/

[33] M. Vukasović, D. Miladinović, A. Milaković, P. Vuletić, Ž. Stanisavljević, Programming Applications Suitable for Secure Multiparty Computation Based on Trusted Execution Environments, TELFOR 2022, pp. 1 - 4, IEEE, Belgrade, Nov, 2022

[34] Chen, K. (2023). Confidential High-Performance Computing in the Public Cloud. IEEE Internet Computing, 27(1), 24-32.

[35] Wilke, L., Wichelmann, J., Morbitzer, M. and Eisenbarth, T., 2020, May. Sevurity: No security without integrity: Breaking integrity-free memory encryption with minimal assumptions. In 2020 IEEE Symposium on Security and Privacy (SP) (pp. 1483-1496). IEEE.

[36] F. Hetzelt and R. Buhren, “Security Analysis of Encrypted Virtual Machines,” in Proceedings of the 13th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, ser. VEE ’17. New York, NY, USA: ACM, 2017, pp. 129–142. [Online].

[37] J. Werner, J. Mason, M. Antonakakis, M. Polychronakis, and F. Monrose, “The SEVerESt Of Them All: Inference Attacks Against Secure Virtual Enclaves,” in Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, ser. Asia CCS ’19. ACM, 2019, pp. 73–85.

[38] Li, M., Zhang, Y. and Lin, Z., 2021, November. Crossline: Breaking" security-by-crash" based memory isolation in amd sev. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (pp. 2937-2950).

[39] M. Li, Y. Zhang, and Z. Lin, “Exploiting Unprotected I/O Operations in AMD’s Secure Encrypted Virtualization,” in 28th USENIX Security Symposium (USENIX Security 19). USENIX Association, 2019.

[40] M. Morbitzer, M. Huber, J. Horsch, and S. Wessel, “SEVered: Subverting AMD’s Virtual Machine Encryption,” in Proceedings of the 11th European Workshop on Systems Security, ser. EuroSec’18. New York, NY, USA: ACM, 2018, pp. 1:1–1:6. [Online].

[41] M. Morbitzer, M. Huber, and J. Horsch, “Extracting Secrets from Encrypted Virtual Machines,” in Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy, ser. CODASPY ’19. ACM, 2019, pp. 221–230.

[42] R. Buhren, H.-N. Jacob, T. Krachenfels, J.-P. Seifert. 2021. One Glitch to Rule Them All: Fault Injection Attacks Against AMD's Secure Encrypted Virtualization. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21)., New York, NY, USA, 2875–2889. https://doi.org/10.1145/3460120.3484779

[43] Gu, J., Wu, X., Zhu, B., Xia, Y., Zang, B., Guan, H. and Chen, H., 2020. Enclavisor: A hardware-software co-design for enclaves on untrusted cloud. IEEE Transactions on Computers, 70(10), pp.1598-1611.

[44] Aktas, E., Cohen, C., Eads, J., Forshaw, J. and Wilhelm, F., 2023. Intel trust domain extensions (TDX) security review. Google security review.

[45] Schlüter B, Sridhara S, Kuhne M, Bertschi A, Shinde S. Heckler: Breaking Confidential VMs with Malicious Interrupts. In: 33rd USENIX Security Symposium (USENIX Security), August 14-16, 2024

[46] Schlüter B, Sridhara S, Bertschi A, Shinde S. WeSee: Using Malicious #VC Interrupts to Break AMD SEV-SNP. In: 45th IEEE Symposium on Security and Privacy (IEEE S&P), May 20-23, 2024.

[47] Wilke, L., Sieck, F. and Eisenbarth, T., 2024, December. TDXdown: Single-Stepping and Instruction Counting Attacks against Intel TDX. In Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security (pp. 79-93).