Computation Capabilities of Server-Side Trusted Execution Environments

A Comparison of TEEs to Privacy-Preserving Technologies

Vlad-Ștefan Popescu[1]

Supervisor and Responsible Professor: Lilika Markatou[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2025

Name of the student: Vlad-Ștefan Popescu
Final project course: CSE3000 Research Project
Thesis committee: Lilika Markatou, Tim Coopmans

Abstract

While securing data-in-use was assured by well-known encryption algorithms, the industry shifted towards trusting hardware manufacturers in exchange for efficiency speedups through Trusted Execution Environments. However, there are many technologies to choose from, each with its own design and trade-offs. Additionally, no work was conducted to systematically compare Trusted Execution Environments side-by-side with privacy-preserving techniques. Therefore, this literature review analyzes, in the first part, Intel's SGX, Keystone, Intel's TDX, and AMD's SEV from four angles that are strongly tied to data-in-use protection (functionality, efficiency, security, and usability). We observed that even though complex and inherently suffering to hardware-related attacks, TEEs offer a great option for confidential computing. Lastly, this research compares these state-of-the-art technologies to four other privacy-preserving techniques (Fully Homomorphic Encryption, Secure Multi-Party Computation, Oblivious RAM, and Structured Encryption) by drawing common properties and displaying them in equal use cases, showing that TEEs are a great choice for many use cases, but with stronger security issues.

1 Introduction

Cloud servers have become an essential part of today's infrastructure. As our dependence on them increased, so did their complexity, even though the software used by these devices tends to contain critical vulnerability bugs that can leak sensitive user data [1] or allow unauthorized remote access to the system [2]. Although cryptography standards can ensure data safety at rest and in transit, the attack surface is increased when computing time is taken into account, which can be a vulnerable step for a malicious user with leveraged privileges [3]. To address the issue of stealing data-in-use, multiple security models that promise confidentiality and integrity were generalized, such as **Oblivious RAM** [4], **Secure Multi-Party Computation** [5], **Structured Encryption** [6], or **Fully Homomorphic Encryption** [7].

One promising method to strengthen security in the cloud is by running software inside Trusted Execution Environments (TEEs), which are a technology able to leverage hardware extensions that can safely execute computations with respect to data integrity and confidentiality and code integrity [8]. They distinguish themselves from other methods by using built-in silicon-level cryptographic capabilities to encrypt and decrypt memory in separate isolated environments [9].

Currently, the implementations of this technology differ from manufacturer to manufacturer, each with varying security designs and trade-offs. For example, **ARM's TrustZone** abstracts itself from the rest of the hardware by introducing a "trusted world of execution" [10, p. 118], while **Intel's Software Guard Extension** (SGX) isolates computation through enclaves and ensures authenticity and integrity using remote attestation [11]. Thus, it might be difficult for an organization that handles sensitive data to assess the risks they are exposed to when choosing between these technologies.

To the best of our knowledge, no existing study provides a comprehensive analysis of TEEs, Fully Homomorphic Encryption (FHE), Secure Multi-Party Computation (MPC), Oblivious RAM (ORAM), and Structured Encryption (StE) within the same context. Existing literature tends to focus on individual technologies or pairwise comparisons, leaving a disparity in understanding their relative strengths, limitations, and appropriate use cases.

To combat the knowledge gap, the study is split into two parts: highlighting the existing properties of cloud-based Trusted Execution Environments and comparing them to those of four other confidential computing technologies (Oblivious RAM, Secure Multi-Party Computation, Structured Encryption, and Fully Homomorphic Encryption). Thus, the research questions can be formulated as follows:

1. What are the computational limitations and capabilities of server-side Trusted Execution Environments concerning **functionality**, **efficiency**, **security**, and **usability**?
2. How do Trusted Execution Environments compare to Fully Homomorphic Encryption (FHE), Oblivious RAM (ORAM), Structured Encryption (StE), and Secure Multi-Party Computation (MPC)?

To answer these questions, the work is organized as follows. Section 2 presents key terminology for understanding Trusted Execution Environments, previous work, and the methodology of this research. Section 3 will showcase an overview of related technologies, followed by relevant implementations of TEEs. The first question will be answered in detail in Section 4 by splitting the properties into four categories. The second question will be elaborated on in Section 5. In the last parts of the publication, Section 6 will underline possible risks that the research is creating, as well as reproducibility concerns, while Sections 7 and 8 will summarize the findings in the previous sections and come up with recommendations on possible future research that can be conducted to overcome the mentioned limitations.

As part of our contributions, we compiled selection criteria to restrict the detailed view to four state-of-the-art technologies: Intel SGX [11], Keystone [12], Intel TDX [13], and AMD SEV [14]. By the end of this research, we evaluated the TEEs, presenting their key takeaways under similar features. Overall, we noticed that confidential virtual machines offer a more user-friendly deployment with better performance than their enclave-based counterparts, but at the cost of a broader trusted computing base.

Lastly, we compared Trusted Execution Environments with the other four privacy-preserving methods. While all of them lacked in complexity when compared to TEEs and had better security assurances, Fully Homomorphic Encryption is still unfeasible due to efficiency issues, Oblivious RAM is meant to protect access patterns rather than direct data manipulation, and Structured Encryption is functionally limited to a set of querying operations. Secure Multi-Party Computation approaches the capabilities of TEEs in general use cases, at the cost of preprocessing overhead.

2 Preliminaries

2.1 Background

To introduce some relevant terminology for Trusted Execution Environments, three concepts highlighted in the security models of most TEEs are comprehensively described below. These concepts form the foundation for understanding how cloud-based TEEs achieve data-in-use confidentiality and integrity, and execution integrity. A broader perspective of security and architectural features has been included in the work by Maene et al. [15].

Attestation

Attestation has been defined in 2011 by Coker et al. as the process where an attester proves to an auditor that certain properties exist by presenting supporting cryptographic evidence [16]. In the current context, the server has to assure the client that the system is untampered. In cloud environments, it is of extensive use to be able to perform this pipeline from any client outside of the server. Thus, **remote** attestation using evidence generated by the hardware root of trust is usually a fundamental feature of server-side execution environments [17]. Additionally, some TEEs can perform **local** attestation to enable secure communication between channels on the same machine.

Isolation

Sabt et al. [18] abstracted, among other terms, the concept of isolation as the presence of a separation kernel mechanism which enables the division between multiple environments with controlled common partitions. They have also highlighted four policies that this feature meets: data separation, sanitization, control of information flow, and fault isolation. Generally, strong isolation can be assured in Trusted Execution Environments by nested virtualization [19] or by splitting the system into separate blocks with different privileges and capabilities [20].

Trusted Computing Base

The trusted computing base (TCB) of a TEE is the summation of hardware and software components vital for the security, which, once the rest of the system is compromised, protects the execution environment from threats [15]. While it is preferable to handle sensitive logic in hardware, Trusted Execution Environments with HW-only TCBs are not upgradable, thus making security changes impossible. The size of software TCB can range from thousands of lines of code [21, 12] to millions [19], a measurement which usually serves as a trade-off between the complexity of the functions of a TEE and its security.

2.2 Related Work

Previous research has been conducted in comparing different implementations of Trusted Execution Environments. Mofrad et al. [22] claimed they were the first to compare two widely used technologies in this field by means of use cases, security, functionality, and performance. An investigation on design trade-offs of popular cloud implementations was conducted by Li et al. [23], introducing the TEE Runtime Architectural Framework (TRAF) and highlighting four modes of TEE runtime management, focusing on security.

Distinctions between privacy-preserving computation algorithms and Trusted Execution Environments have been analyzed in the past. Xu et al. [24] compared Homomorphic Encryption, Secure Multi-Party Computation, and Trusted Execution Environments in the context of blockchain, referring to detailed research highlighting use cases of these technologies. However, it was done in parallel rather than analyzing common characteristics. Mulligan et al. [25, p. 137] stated that FHE and MPC "will inevitably become attractive" in the context of future performance improvements, mentioning the advantage of the lack of trusted hardware in the two cryptographic approaches. Oblivious RAM, Structured Encryption, and TEEs have not been compared side-by-side, but rather combined to complement each other. Kane et al. [26] introduced an oblivious graph querying scheme, where two ORAM techniques were used, and trusted execution diminished the communication complexity.

2.3 Methodology

To identify relevant literature on Trusted Execution Environments, the research was conducted using the Backward Snowballing method, starting from a System of Knowledge basis [23]. This led to a strong foundation of tens of materials from which four state-of-the-art general-purpose cloud Trusted Execution Environments and their properties were extracted: Intel's Software Guard eXtension (SGX) [21] and Trusted Domain eXtension (TDX) [13], AMD's Secure Encrypted Virtualization (SEV) [14], and Keystone [12]. They were filtered based on a choosing criteria, which is further detailed in the next section.

Furthermore, to enrich the literature with newer and less-cited work, a search query was compounded for Scopus[1], leading to 69 bodies of work (May 2025). It was meant to find correlations between the four mentioned Trusted Execution Environments, where functionality, efficiency, security, and usability properties are highlighted. The exact version of the query, including the applied search filters, can be found in Appendix A.

The comparison among the five techniques has been done with the assistance of the four colleagues who have individually followed a pre-established research process comparable to this reference. Similarly, they presented through a literature review existing protocols in FHE, MPC, ORAM, or StE, and showcased relevant properties for functionality, efficiency, security, and usability. For completeness and correctness reasons, other queries have been performed on Google Scholar [2] where needed, with terms found in Appendix A.

3 Existing Technologies

The following section will initially investigate the starting points for the current implementations of Trusted Execution Environments. Then, it will present the selection criteria, which will be applied to the existing literature to pick four state-of-the-art technologies. Lastly, we will present the defining characteristics of the selected TEEs.

[1] https://www.scopus.com/search/form.uri

[2] https://scholar.google.com/

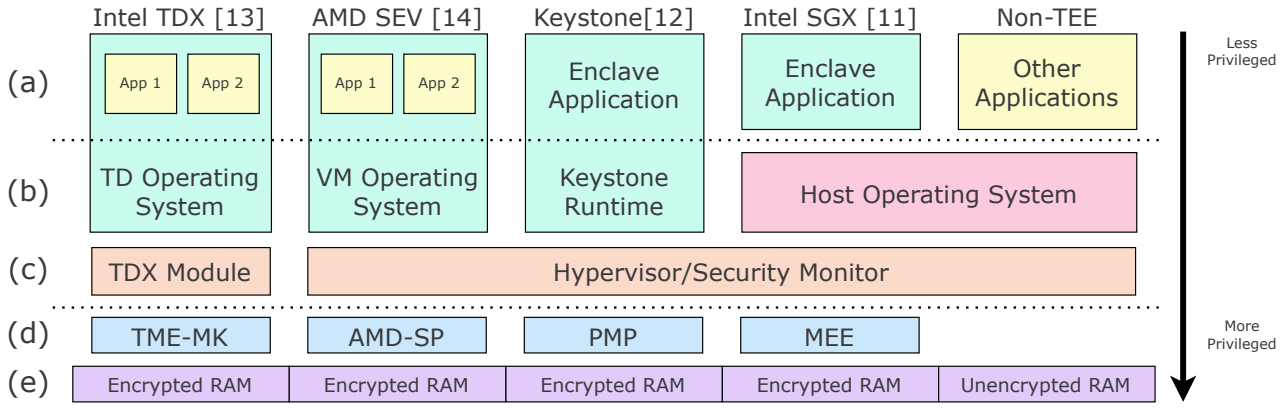

Figure 1: Execution stack from user-space to hardware of a high-level overview of selected Trusted Execution Environments. **(a)** User-level software; **(b)** Kernel-level software; **(c)** Highest privilege software; **(d)** Specialized security hardware; **(e)** System Random Access Memory;

3.1 Early Architectures

Before the existence of modern-day TEEs, the industry of privacy-preserving computations had to rely on more restricted hardware. **Smart cards** were the first technology to reliably execute unmodifiable code, controlled only by the manufacturer, on an isolated, trusted chip [27]. An impactful use case is the replacement of magnetic stripe cards by a French banking association, which reduced fraud by 75% [28]. In parallel with the financial sector, these cards are still widely used in identification, access, or the medical sector.

The field of isolated computation on specialized hardware saw a rapid growth in popularity and power, with technologies such as **Java cards** or **Subscriber Identity Modules** (SIMs) being added to the modern infrastructure. In the meantime, the hardware of personal computers was not inherently deployed with security in mind, which was needed with the increase in popularity [29]. For that, the Trusted Computing Group (TCG)[3] was formed by vendors to add hardware roots-of-trust (RoTs) to PCs. In 2003, they released the first **Trusted Platform Module** specification, TCPA 1.1b, which laid the basis of a new technology capable of storing and reporting integrity metrics, using the hardware root-of-trust [30]. This standard was the foundation of contemporary Trusted Execution Environments, many using the hardware RoT, attestation, and some of the measurements during secure boot.

Initially, TEEs were designed for mobile devices. One of the first Trusted Execution Environments to be widely deployed to commodity machines was ARM's TrustZone [31], with its release in the early 2000s. Even though it was not commonly used in the early stages, Apple started using it in the iPhone 5s' biometrics authentication system to protect the user's fingerprint [32]. Currently, this technology is deployed in billions of mobile phones worldwide.

With a growing industry of isolated execution on edge devices, the Open Mobile Terminal Platform organization was the first identity to define Trusted Execution Environments in 2009 formally [33]. Successively, in 2011, GlobalPlatform[4] released specifications for an interoperable TEE, called OP-TEE, aimed towards TrustZone-enabled devices [34].

Uses in cloud servers were later conceptualized with the release of an extension to the x86_64 architecture brought by Intel in their first TEE: Intel SGX [11]. The main contribution brought by this technology was its capability of protecting the computations and keys from the operating system. Among other cloud-based TEEs, we denote: Intel TDX [13], AMD SEV [14], IBM PEF [35], ARM CCA [36], or open-source options, such as Sanctum [37] or Keystone [12].

3.2 Selection Criteria

Two selection criteria were defined to identify the four technologies discussed in Section 4: the system has to be designed for diverse **cloud usages**, and the TEE has to be **popular** in literature, with applied use cases. Included in the comparison will be technologies that matched the criteria the best: Intel SGX [11], Keystone [12], AMD SEV [14], and Intel TDX [13]. Fig. 1 keeps a high-level overview of the designs, presenting the five main privilege levels.

Cloud Integration

Our initial goal was to omit entries unrelated to cloud computing. Technologies mainly designed for mobile architectures, such as ARM Trustzone [10] and OpenTEE [38], were excluded. Even though still classified as Trusted Execution Environments, technologies incapable of performing computations, such as Trusted Platform Modules (TPMs) [39], are out of scope for the fifth section, and thus were omitted. From the large pool of TEE implementations, we tried to focus on designs beneficial in the world currently.

Popularity

Both the number of search results on Google Scholar and the widespread usage in the industry served as important selection criteria, which became an important aspect in the comparison among the other four privacy-preserving techniques.

[3] https://trustedcomputinggroup.org/about/board-of-directors/

[4] https://globalplatform.org/

The template of the queries was ' "[technology name]" "execution environment" ', where *[technology name]* was replaced by the specific name of the technology being evaluated (e.g., Intel TDX). Our chosen TEEs yielded the following number of search results (as of June 2025): Intel SGX with 5260, followed by Keystone (1320), AMD SEV (954), and Intel TDX (377). Note that Intel TDX was released the latest. With known cloud providers, such as Microsoft, Google, and Alibaba, providing hardware-based confidential computing-ready environments [40, 41, 42], it served as extra validation to further include implementations that matched the first two criteria.

3.3 Enclave-based Technologies

Intel Software Guard Extension

Intel SGX [11] is the oldest of the chosen technologies for this comparison, with its first version being released in 2015 for Xeon processors. The isolation technique extends to the *x86_64* instruction set architecture by including new process-manipulating instructions that developers can indirectly use through the Intel SGX SDK. The enclaves are completely isolated from each other and run entirely in the user space [43]. Data and code are stored in the Enclave Page Cache (EPC), a section located in the CPU-protected Processor Reserved Memory (PRM, Fig. 1e) [21]. It is encrypted using the hardware memory encryption engine (MEE, Fig. 1d).

Initially, Intel SGX was not designed for unmodified high-performance computation, suffering from major slowdowns in the case of memory-consuming processes or when running multi-threaded programs [44]. Therefore, a second generation with a more permissive EPC allocation and a better CPU cache size was created, called SGXv2.

One of the drawbacks of running entirely in the least privileged level is the need to rely on the host operating system to handle system calls, and to call instructions such as *ECREATE* or *EDESTROY* to create and to end existing enclaves, respectively. This design flaw led to attacks where the OS-controlling malicious actor could perform a ROP-chain attack [45] or where the execution integrity was dismantled by crafted exception [46].

We considered this technology in the comparison mainly due to its popularity in existing literature, as some current technologies are built on the security model of SGX.

Keystone

Keystone [12] is the only open-source[5] technology of the four, having transparency and customizability as the main design principles. Notably, developers can exclude unwanted trusted software when the computation does not depend on it. Additionally, Keystone is only limited to the RISC-V architecture, and vendors are the ones who can support this technology by implementing the required cryptographic primitives, such as the Physical Memory Protection (PMP, Fig. 1d).

Similarly to Intel SGX, developers can create applications that run in the user space called *eapps*. In contrast to the other discussed enclave-based technology, Keystone's trusted stack offers a higher privilege component, the Keystone Runtime, which communicates directly with the RISC-V security monitor. While providing an SDK, one can also execute binaries that are entirely or partially managed by the runtime, thus further reducing the surface of the trusted computing base [12].

While not being known for its industrial usage, we chose Keystone as it is the most popular academically oriented TEE, followed by Sanctum [37], with 618 search results as of June 2025. Moreover, we aimed to introduce at least one open-source project.

[5] https://github.com/keystone-enclave/keystone

3.4 Confidential Virtual Machines

Intel Trusted Domain Extension

The newest of all, released in 2021 as a continuation of Intel SGX, Intel TDX was introduced with the 5th generation of the Xeon processors [19]. Its design leverages an extension called *TDX Module* (Fig. 1c), which replaces the responsibilities of the hypervisor and controls all virtual machines, called here *Trusted Domains*. This new component communicates with the hypervisor using *SEAM* instructions [13], and manages the Secure Extended Page Table, encrypted by the Total Memory Encryption Multi-Key (TME-MK, Fig. 1d).

Some components are reused between the two Intel Trusted Execution Environments. For example, attestation is done similarly to its predecessor by invoking an Intel SGX enclave called the *TD-quoting enclave* through a special new instruction called *SEAMREPORT* [13]. Since both technologies can use the same process, environments running each TEE can attest to each other [47].

While not the most popular among the queried technologies, TDX falls within the scope of our study thanks to its fast adoption in cloud servers. Among use cases, we remark its deployment in data analytics [48] and AI workloads [49].

AMD Secure Encrypted Virtualization

Launched as an integration of AMD hardware memory encryption in the AMD-V virtualization architecture of EPYC processors, AMD SEV leverages an AES encryption engine by using a 32-bit microcontroller integrated in the hardware called AMD Secure Processor (AMD-SP, Fig. 1d) [14]. Contrary to the design of Intel TDX, AMD SEV extends its TCB by trusting the hypervisor when managing the virtual machines' resources by enhancing the memory layout, adding an extra bit to allow for partial memory encryption [22].

Security extensions have been launched years after the initial release to improve the flawed design, namely SEV-ES (Encrypted State) [50] and SEV-SNP (Secure Nested Paging) [51]. The first addition blocks the hypervisor from reading unencrypted register values once VMs stop running [50]. AMD SEV-SNP, built on top of ES, solved integrity issues using a Reverse Map Table data structure and introduced an optional extra division of the privilege levels [51].

AMD SEV is the most popular confidential virtual machine, being deployed in cloud AMD-based processors [41]. Thus, it was included in the survey as the state-of-the-art model for CVMs. Moreover, the lack of integrity protection in the first iteration of this technology led to a better understanding of the security requirements in TEEs [23].

4 Properties of TEEs

Tables covering all details discussed below can be found in Appendix B, split by subsections. They summarize key findings, including data extracted from existing literature.

4.1 Functionality

Parties Involved in Computation

The main parties participating in the secure computation process are the server, the client, and the attesting service. Firstly, the purpose of the server is to provide TEE-enabled hardware, as well as physical resources. It is not considered to be part of the trusted side. Secondly, the client verifies the environment, initiates the computing process, and receives the results. They sit between the server and the attesting service. Lastly, the attesting service verifies the quotes received from the client to validate their genuineness and integrity. It is part of the trusted set.

Remote Attestation

Intel SGX and Intel TDX use the Quoting Enclave (QE) component, which holds all signing functionality, as the process is "too complex to be implemented in hardware" [21, p. 83]. In SGX, the QE leverages Intel Enhanced Privacy IDs (EPIDs) to sign without revealing information about the platform [11]. In TDX, the TDX module generates a report, which is verified by the Quoting Enclave (QE) and then signed using an attestation key that is certified by Intel's Provisioning Certification Key [19]. Keystone generates the attestation key through a secure boot that is signed by the root-of-trust [17]. One useful feature of attestation in this technology is the capability of adding arbitrary data. Similarly to Keystone, AMD SEV uses the root-of-trust that is strongly tied to the hardware, while the SNP extension is capable of performing remote attestation at any time [17].

Data Sealing

A feature present in enclave-based TEEs, sealing is the process of storing encrypted data on permanent storage to be used later, even after a system reset. Since permanent storage is not part of the TCB, it is of great necessity to hide data from possible adversaries accessing the disk. In Keystone, *get_sealing_key* can be called in the Supervisor Binary Interface (SBI), and is derived from the enclave hash and the security monitor's private key [52]. Intel SGX data sealing keys can have two modes: sealing to the Enclave Identity and sealing to the Sealing Identity. The latter is useful especially when willing to share sealing keys among multiple enclaves [11]. Concluding, enclave-based TEEs offer additional functionality, with Intel's sealing being more flexible.

4.2 Efficiency

Task Performance Overhead

Generally, confidential virtual machines and Keystone offer a near-native performance, but take extensive time to boot [12, 19]. In the case of early versions of vanilla Intel SGX, the small size of the EPC of just 128 MB and the poor multi-thread scaling led to a massive degradation in performance for multi-threaded high computation tasks (up to 126×) [44]. However, this limitation has been addressed in SGXv2 by permitting dynamic EPC allocation [53].

One of the largest bottlenecks is the usage of I/O operations due to the limitations on accessing shared memory. Coppolino et al. [54] benchmarked Intel SGX with Gramine and Occlum extensions, Intel TDX, and AMD SEV on I/O intensive processes, NGINX, and Node.js. It was shown that the smallest overhead was in TDX, with an average of 28.6%, followed by SEV and SGX by a large margin. Overall, confidential virtual machines offer the best performance, with Keystone being close in results, and SGX improving greatly with the release of SGXv2.

Memory Protection Overhead

On top of execution latency, TEEs also add extra integrity protection overhead through the usage of memory metadata or authentication codes. In CVMs, additional bytes are used to assure memory integrity, which represent MAC values in TDX [13] or Reverse Map Table entries in AMD SEV-SNP [51]. Contrary to the specifications of the Confidential Computing Consortium, AMD SEV does not use any memory integrity protection mechanisms. Similarly to TDX, SGX uses MACs for integrity, on top of EPC page metadata kept in the PAGEINFO structure [21]. Keystone adds extra overhead to protect memory integrity by either attaching software to the Runtime component or leveraging a hardware memory encryption engine [12].

4.3 Security

Trusted Computing Base Size

Enclave-based TEEs are lighter in trusted components, yet it is difficult to measure the security based on size in terms of lines of code. Intel SGX's design was made to minimize the TCB. Thus, only CPU microcode and some privileged containers, such as the Quoting Enclave, are part of the trusted set [21]. As Keystone offers an open-source implementation of its primitives, it is trivial to measure the TCB. As mentioned by the authors, an *eapp* running on the default implementations of the security monitor and the runtime reaches a total of 15 thousand lines of code [12]. Under similar configurations, the trusted computing bases of Intel TDX and AMD SEV are of related sizes when comparing blocks of equal responsibility [19]. However, the measurement does not include the Quoting Enclave and the TDX Module Loader sizes. With the number of lines of code in CVMs sitting in the order of millions, including the guest operating system, their TCBs suffer from complexity issues. Therefore, enclave-based TEEs are to be preferred in a security-first policy from this aspect.

Threat Model

The threat model of Trusted Execution Environments usually excludes data in states other than in use. However, considering the functionalities described above, processes such as attestation and sealing that generate data in transit and at rest, respectively, should be included in the discussion [8]. The strong protection mechanisms imply a powerful adversary, which can be categorized under the following scenarios:

- **Software-controlling attacker**, capable of controlling high privilege level processes, such as the operating sys-

tem or the hypervisor. This includes a potentially malicious cloud service provider or a remote attacker who compromises the host system.

- **Physical access attacker**, capable of monitoring hardware and performing basic attacks. This scenario is usually omitted in threat models of individual TEEs.

On the trusted side, we denote the client requesting the computation results, the TEE components part of the trusted computing base, and the attesting service. Attacks such as breaking cryptographic primitives, performing difficult hardware intrusions, or denial of service are considered outside of the scope of the threat model.

Architectural Design Flaws

Work classifying the specific attacks concerning Trusted Execution Environments was done by Muñoz et al. [55]. Even though the publication covers mostly mobile technologies, the following taxonomy can also be applied to cloud-based implementations:

- **Software attacks**, regarding host software such as the operating system and the hypervisor, or programs running inside the TEE.
- **Architectural attacks**, exploiting hardware flaws or logic, including micro-architectural components such as the cache or the encryption engine.
- **Side-channel attacks**, where information can be leaked through indirect signals, such as power usage or timing.

Software code-reuse attacks have been theorized in Intel SGX [56]. In this work, an attacker was shown to be capable of exploiting a possible memory corruption vulnerability to control the entire enclave by being able to place arbitrary data in memory. To mitigate this issue, the author proposed adding an extra layer of encryption by XORing arbitrary data writes with a secret key. To generalize, the SDKs of Keystone and Intel SGX are primarily written in C/C++, so developers have to consider the memory issues when developing enclave applications.

Design-logic attacks on confidential virtual machines at the hypervisor level interaction were shown feasible by injecting interrupts. For example, *Heckler* [57] rips the confidentiality and integrity checks in Intel TDX and AMD SEV-SNP, allowing the hypervisor to bypass authentication mechanisms in services such as *sudo* or *OpenSSH*. Mitigation techniques were proposed by the authors in collaboration with the vendors through code patches, but only Intel fully stopped this attack. Therefore, attack vectors on TEEs differ among designs, and both CVMs and enclave-based Trusted Execution Environments have architecture security flaws.

A subclass of the side-channel category, **Speculative attacks** such as Spectre [58] or Meltdown [59] were shown to be viable in Trusted Execution Environments. The SGXpectre attack [60] managed to retrieve sealing keys through cache timing. Similarly, malicious access to the hypervisor in SEV led to the leakage of AES encryption keys [61], while an extensive report by Google [62] on TDX revealed the required prerequisites for an attacker to perform Spectre. Lastly, Keystone was tested for Spectre exploitation, but was found secure thanks to its cache protection [63].

Known Public Attacks

By querying the *CVE.org*[6] database in June 2025, AMD SEV and Intel SGX, including their extensions, display 49 and 48 known public vulnerabilities of any score, respectively. Ten CVEs are assigned to Intel TDX, but with higher scores, the minimum being 5.6. Keystone does not have any entries, but some GitHub issues regarding security problems can be found in the open-source repositories [64, 65].

4.4 Usability

Ease of deployment

A very useful criterion to measure how convenient it is for developers to move their applications to a trusted environment is the ease of deployment. Steps not related to computation, such as attestation, are omitted. Thus, this subsection answers the question "*What modifications must be made to integrate TEEs into an existing cloud computing stack?*". Below, we show that Intel SGX requires the most modifications of all subjects, CVMs being the easiest to use of all TEEs.

Intel's TDX permits unmodified user-level apps to run inside the protected VM, only with some modifications to the operating system, called in this case the *TDX-Enlightened OS* [66]. The AMD SEV design allows for no modification of the ecosystem. With confidential virtual machines being straightforward in porting existing or legacy systems, enclave-based Trusted Execution Environments have some prerequisites. While developers can use unmodified RISC-V binaries in Keystone, the host requires creating minimal software that launches the enclave runtime and application [67]. Lastly, Intel SGX requires code modification by leveraging the SGX SDK. In an interview conducted by Geppert et al. [68], changes in this direction are seen as expensive to implement for current systems, and adoption is expected to be done only once regulated. This challenge can be partially overcome through open-source extensions such as Occlum [69] or Gramine [70].

Debugging

Developers are permitted under specific conditions to inspect decrypted memory and registers, step through their programs, or monitor execution states using the provided platform tools. Generally, TEEs enable debugging only if it is allowed prior to launching the process. Thus, the debugging method in AMD SEV is through an API and can be enabled by setting the guest policy *NODBG* bit [71]. Using the two commands, the developers can decrypt and encrypt memory through the firmware. In Intel TDX, trusted domains can be debugged by both *On-TD* software, allowing for all architectural supported features, or by *Off-TD* software, at the VMM level, to modify guest state [66]. Intel SGX enclaves opt for a debugging bit in the enclave metadata [21]. The processes are meant not to be stopped between instructions when deployed in *Release* mode. Even if it is expected for a general-purpose debugger to see the execution as one instruction, an attack [72] managed to single-step the flow of an enclave. Keystone does not permit explicit debugging for enclave applications, but the SM can be debugged through standard ways, as described on the GitHub repository [73].

[6] https://www.cve.org

5 Comparison with Other Techniques

Tables summarizing the topics reached below can be found in Appendix C. Additional contributions brought by the graphics are the applicability and the use cases of each technology, as shown in columns three and four of Table 5.

5.1 Fully Homomorphic Encryption

Homomorphic Encryption is a privacy-preserving technique that allows users to perform operations directly on encrypted data, without revealing the plaintext in the untrusted computation environment [7]. It is based on mathematical NP-hardness properties and is composed of four probabilistic polynomial-time algorithms: *KeyGen*, *Enc*, *Dec*, and *Eval*. Of great interest is the **Fully** Homomorphic Encryption technique, with the first generation being proposed in 2009 by Gentry [74]. It is the variant capable of performing arbitrary computations on encrypted data.

Functionally, Fully Homomorphic Encryption can perform the same operations as Trusted Execution Environments. While TEEs' operations are seen as the decryption of memory on the hardware level, FHE transforms operations into functionally complete logic gate sets, such as the NAND gate [75]. An overhead in Trusted Execution Environments that is not found in Fully Homomorphic Encryption is the need for attestation.

From an **efficiency** perspective, TEEs outperform FHE by a margin. While confidential virtual machines and Keystone were shown to reach near-native speeds for most computations, it is not yet feasible to use FHE schemes to perform any kind of complex operations [76]. The analyzed algorithms showed a clear bottleneck in bootstrapping or in the *KeyGen* primitive, some taking hours to perform AES encryption.

The **security** model of Fully Homomorphic Encryption has a different paradigm than that of Trusted Execution Environments. While previously we analyzed documented attack vectors and settings in which they can be performed, the properties of FHE have been abstracted through indistinguishability. By default, FHE is semantically secure, making it IND-CPA secure, and extensions to existing constructions led to IND-CCA1 assurances, but not to IND-CCA2 [77]. Side-channel attacks, a drawback of the TEE security, were viable in Fully Homomorphic Encryption only for client-side adversaries [78].

5.2 Secure Multi-Party Computation

Secure Multi-Party Computation is a cryptographic paradigm that enables a group of users to privately compute without ever disclosing their data to the others [5]. At the basis of this technique, there stand two useful protocols: *Oblivious Transfer* [79], used to securely deliver inputs from one party to another without revealing which input was chosen, and *Shamir Secret Sharing* [80], which enables the distribution of private data across multiple parties.

As in the case of Fully Homomorphic Encryption, Secure Multi-Party Computation can **functionally** perform the same operations as TEEs by evaluating logic circuits. The first protocols that enabled arbitrary circuits in MPC were *Yao's Garbled Circuits* [81] and *GMW* [82]. Similarly to TEEs by having to share inputs among parties, a cloud server and a client using either of the technologies have to interact before the computation is performed. As another important aspect, MPC allows for workload distribution across multiple systems, a property that is not considered in the design of Trusted Execution Environments. Efforts to extend this feature to TEEs were previously theorized with proof-of-concepts [83].

From a **performance** point of view, Secure Multi-Party Computation suffers from slow preprocessing (offline) phases, similarly to how confidential virtual machines encounter slow startups. In a benchmarking suite of a widely used MPC protocol called *SPDZ* [84], performing basic arithmetic approximations was done in milliseconds, which is orders of magnitude slower than the near-native capabilities of TEEs. The offline phases were, as expected, the bottlenecks, with seconds of preprocessing at most. Even though the overhead is considerable, it is less than in Fully Homomorphic Encryption, and extensive work is done in reducing the computing time through compilers [85].

The distributed model of Secure Multi-Party Computation shifts some of the trust among parties. The misbehaving parties are denoted *corrupted* and can be of three categories: semi-honest, malicious, or covert [5], with the latter two reflecting real settings. While many of the MPC schemes were designed to resist stronger adversaries, the ones with weaker assurances, such as Yao's GC, can be extended for performance costs [86]. As breaking cryptographic protocols is outside of the threat model of TEEs, MPC offers in this way a stronger **security**.

5.3 Oblivious RAM

Oblivious RAM is a privacy-preserving technique that hides a client's access patterns in an untrusted environment. It was conceptualized by Goldreich as a method of securing software against adversaries observing the frequency of memory read and write operations [4]. The key point in ORAM is to make all steps oblivious; therefore, the schemes use specialized sorting, hashing, random permuting, or storing [87].

Functionally, Oblivious RAM and Trusted Execution Environments perform different types of computation. While TEEs can run any processing on encrypted data, ORAM is only used in the context of querying stored data on an untrusted server, with all calculations being done client-side. A recurring theme in these two techniques is the implementation in hardware, which was done for *Path ORAM* [88] in a secure processor [89].

As in the case of previous protocols, Oblivious RAM has some computation overhead in each step. Of great interest are the communication, query, and storage (both client and server-side) complexities. There is no oblivious RAM scheme that has the best **efficiency** in balance of performance and memory overhead [87]. The hardware-based *Path ORAM* implementation on the *Ascend* processor [89] revealed an average performance slowdown of 12-13.5×, significantly higher than the near-native execution speeds typically achieved by Trusted Execution Environments.

Both Trusted Execution Environments and Oblivious RAM are designed with the assumption that the cloud service provider (CSP) cannot be trusted. However, the generally considered adversary in ORAM is *semi-honest* [87], com-

pared to malicious in TEEs. Another shared property of these technologies is the exposure to side-channel attacks, as access timing reveals patterns correlating to memory behavior [90]. Fortunately, these **security** issues of Trusted Execution Environments can be diminished by Oblivious RAM solutions. For example, Alam and Chen [91] developed an easy-to-use framework for oblivious computation using Intel SGX that aims at hiding access patterns, one of the problems in Intel SGX side-channel attacks [92].

5.4 Structured Encryption

A well-known technique for enabling secure and efficient data querying inside untrusted environments is Structured Encryption. Originally proposed by Chase and Kamara [6], it allows efficient search operations while limiting information leakage beyond the query or the data. This method was created to be **usable** with the current systems, and is capable of running SQL queries [93], being implemented by MongoDB for queryable databases [94].

While not capable of performing any possible computation like FHE, MPC, or TEEs, Structured Encryption is a powerful tool for querying data. Thus, a user of StE can run boolean or range searches on encrypted data structures, such as multimaps, trees [95], or graphs [6]. Even if restricted only to querying, Structured Encryption has enough **functionality** to enable Private Set Intersection [95], a useful computation that allows two or more parties to calculate the intersection of their items without revealing other items to the others.

The **efficiency** of Structured Encryption is one of the most remarkable advantages, as it is enhanced by parallel computing to achieve sublinear performance [96]. By being efficient in I/O access, an area where Trusted Execution Environments struggle, StE can achieve minimal performance overhead. However, fast I/O is possible if the storage space grows linearly or if one performs a constant number of operations [97]. As such, we consider Structured Encryption a better option when having to perform I/O heavy operations, compared to TEEs.

Structured Encryption is known to leak information about the way the data is structured, the size of the response, or the accessing patterns [98]. This statement applies to both passive and active adversaries, and mitigations for any of them compromise performance. As discussed in ORAM, the **security** of Trusted Execution Environments also suffers from memory-accessing patterns in the case of side-channel attacks, volume leakage not being a problem as it is in StE.

6 Responsible Research

No **conflict of interest** exists between the author and any of the vendors providing the technologies in this work, nor any of the authors whose work was mentioned. The selection criteria in Section 3 were chosen before assessing the implementations, keeping a neutral perspective on the findings. However, we acknowledge that both the selection criteria and the extracted materials in the literature review may reflect inherent **bias**. To combat this issue, we attempted, where possible, to verify the information from multiple sources and maintain transparency in the research.

Artificial intelligence was not used to generate text for any sections of this scientific material, but to correct spelling and grammar mistakes through Grammarly and Writefull. These tools are freely available for commercial use, but with limited features. All materials have been individually collected and reported by the authors. This decision was made to ensure critical engagement through an own understanding of the studied technologies while adhering to academic integrity policies.

From an **ethical viewpoint**, this study does not cover human interaction, sensitive data, or critical tests that could affect the functionality of live systems. However, we advise caution when interpreting the results presented in this work, as they are drawn from secondary resources, rather than primary research. The readers are encouraged to reevaluate the stated claims.

Under no circumstances do the authors of this work encourage the **malicious use**, distribution, or fabrication of the vulnerabilities mentioned previously. The presentation of security issues, CVEs, and threat models is showcased solely for educational, academic research, to strengthen the understanding of these issues in the context of privacy-preserving technologies.

Regarding **reproducibility**, all materials are extracted from publicly available forums or websites (e.g., Google Scholar, Scopus, and IEEE eXplore). We recognize that some indicators in this research are time-dependent and will change if queries are to be further conducted in the future. Here, we denote the number of materials found on Scopus and Google Scholar, the "Popularity" queries of Section 3.2, as well as the number of CVEs in Section 4.3. We provided all the used queries in Appendix A, including the platform and the used filters. These should be interpreted relative to the research period, April-June 2025.

7 Future Work and Limitations

The study was conducted under format constraints and limited time, providing a broader overall material than expected. Some details could have been added to offer a clearer comparison concerning efficiency, security, and usability. For example, to understand the performance issues of each discussed technique, additional benchmarking tests could have been conducted to collect data of significant use, both between TEEs and the five privacy-preserving methods. Among other properties of great interest in discussion, we recommend future work comparing live migration, secure boot verification, trusted I/O, and mitigations against cache and physical attacks. Additionally, we leave the possibility of combining two or more of the discussed techniques for future research. We would expect an increase in security, as seen in the ORAM-TEE merge, with a decrease in performance.

8 Conclusions

This work highlighted in Sections 3 and 4 the main properties of Intel SGX, Keystone, Intel TDX, and AMD SEV concerning functionality, efficiency, security, and usability by reviewing literature regarding Trusted Execution Environments. We displayed in Tables 1 to 4 the trade-offs of each technology

and showed that each design differs greatly from the others. Moreover, we remarked that there is no "silver bullet" when choosing to work with TEEs, and even though they offer generally insignificant performance overheads, they suffer from damaging attacks caused by strong malicious actors.

Intel SGX, the most popular technology among existing TEEs, was not initially designed for high-performance computations, considering its small EPC size. However, SGXv2 removed such limitations and improved the efficiency of parallelized programs. Being the first breakthrough in Trusted cloud Execution Environments, it inspired many other vendors to implement such isolation techniques in their processors.

Keystone showed that TEEs do not have to be fully tied to specific hardware models, but leverage primitives that enable security at the physical level. While not used at a large scale in the industry, it is an actively improved open-source implementation that supports and facilitates scientific research.

Intel TDX is a powerful continuation of Intel SGX, inheriting attestation mechanisms. Among the four discussed technologies, it is the only one to include high-privilege software in its TCB, called the TDX Module, whose purpose is to manage the trusted domains.

AMD SEV was the first widely adopted confidential virtual machine. Even though it did not initially implement many of the security mechanisms that are necessary to protect against integrity attacks, extensions to it were added to diminish these issues. As of June 2025, popular cloud vendors enable AMD SEV-SNP for services running on AMD EPYC processors.

In the fifth section, we surveyed other promising techniques to perform computations on encrypted data. Tables 5 and 6 display a summary of their features. While in theory, the algorithmic approaches appeared promising with the ongoing growth in computing power, in practice, they still have a considerable overhead (FHE, MPC) or are limited to a subset of operations (ORAM, StE). However, they counter stronger adversaries when compared to TEEs, which shift the security responsibility to the hardware vendors.

References

[1] Zakir Durumeric, Frank Li, James Kasten, Johanna Amann, Jethro Beekman, Mathias Payer, Nicolas Weaver, David Adrian, Vern Paxson, Michael Bailey, et al. The Matter of Heartbleed. In *Proceedings of the 2014 conference on internet measurement conference*, pages 475–488, 2014.

[2] Douglas Everson, Long Cheng, and Zhenkai Zhang. Log4shell: Redefining the Web Attack Surface. In *Proc. Workshop Meas., Attacks, Defenses Web (MADWeb)*, pages 1–8, 2022.

[3] Ellison Anne Williams. Data Protection: Data in Use Is the Point of Least Resistance. https://www.securityweek.com/data-use-point-least-resistance/, April 2024.

[4] Oded Goldreich. Towards a Theory of Software Protection and Simulation by Oblivious RAMs. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 182–194, 1987.

[5] Yehuda Lindell. Secure Multiparty Computation. *Communications of the ACM*, 64(1):86–96, 2020.

[6] Melissa Chase and Seny Kamara. Structured Encryption and Controlled Disclosure. In *International conference on the theory and application of cryptology and information security*, pages 577–594. Springer, 2010.

[7] Chiara Marcolla, Victor Sucasas, Marc Manzano, Riccardo Bassoli, Frank HP Fitzek, and Najwa Aaraj. Survey on Fully Homomorphic Encryption, Theory, and Applications. *Proceedings of the IEEE*, 110(10):1572–1609, 2022.

[8] Confidential Computing Consortium. A Technical Analysis of Confidential Computing. *Confidential Computing Consortium–Linux Foundation, Technical Report v1*, 3, 2022.

[9] Oualid Demigha and Ramzi Larguet. Hardware-Based Solutions for Trusted Cloud Computing. *Computers Security*, 103:102117, 2021.

[10] Clive Shepherd and Konstantinos Markantonakis. *Trusted Execution Environments*, chapter 6.3. Springer International Publishing AG, Cham, 2024.

[11] Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. Innovative Technology for CPU Based Attestation and Sealing. In *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, volume 13. ACM New York, NY, USA, 2013.

[12] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanovic, and Dawn Song. Keystone: An Open Framework for Architecting Trusted Execution Environments. In *Proceedings of the Fifteenth European Conference on Computer Systems*, EuroSys'20, 2020.

[13] Intel. Intel Trust Domain Extensions Whitepaper. Technical report, Intel Corporation, 2020. Accessed: 2025-05-22.

[14] David Kaplan, Jeremy Powell, and Tom Woller. AMD Memory Encryption. *White paper*, 13:12, 2016.

[15] Pieter Maene, Johannes Götzfried, Ruan De Clercq, Tilo Müller, Felix Freiling, and Ingrid Verbauwhede. Hardware-Based Trusted Computing Architectures for Isolation and Attestation. *IEEE Transactions on Computers*, 67(3):361–374, 2017.

[16] George Coker, Joshua Guttman, Peter Loscocco, Amy Herzog, Jonathan Millen, Brian O'Hanlon, John Ramsdell, Ariel Segall, Justin Sheehy, and Brian Sniffen. Principles of Remote Attestation. *International journal of information security*, 10:63–81, 2011.

[17] Jämes Ménétrey, Christian Göttel, Anum Khurshid, Marcelo Pasin, Pascal Felber, Valerio Schiavoni, and Shahid Raza. Attestation Mechanisms for Trusted Execution Environments Demystified. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 95–113. Springer, 2022.

[18] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. Trusted Execution Environment:

What It is, and What It is Not. In *2015 IEEE Trustcom/BigDataSE/Ispa*, volume 1, pages 57–64. IEEE, 2015.

[19] Masanori Misono, Dimitrios Stavrakakis, Nuno Santos, and Pramod Bhatotia. Confidential VMs Explained: An Empirical Analysis of AMD SEV-SNP and Intel TDX. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 8(3):1–42, 2024.

[20] Sandro Pinto and Cesare Garlati. Multi Zone Security for ARM Cortex-M Devices. In *Embedded World Conference*, volume 2020, page 99, 2020.

[21] Victor Costan and Srinivas Devadas. Intel SGX Explained. Cryptology ePrint Archive, Paper 2016/086, 2016.

[22] Saeid Mofrad, Fengwei Zhang, Shiyong Lu, and Weidong Shi. 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*, pages 1–8, 2018.

[23] Mengyuan Li, Yuheng Yang, Guoxing Chen, Mengjia Yan, and Yinqian Zhang. Sok: Understanding Design Choices and Pitfalls of Trusted Execution Environments. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, pages 1600–1616, 2024.

[24] Yuqing Xu, Guangxia Xu, Yong Liu, Yuan Liu, and Ming Shen. A Survey of the Fusion of Traditional Data Security Technology and Blockchain. *Expert Systems with Applications*, page 124151, 2024.

[25] Dominic P Mulligan, Gustavo Petri, Nick Spinale, Gareth Stockwell, and Hugo JM Vincent. Confidential Computing—a Brave New World. In *2021 international symposium on secure and private execution environment design (SEED)*, pages 132–138. IEEE, 2021.

[26] Seyni Kane and Anis Bkakria. A Privacy-Preserving Graph Encryption Scheme Based on Oblivious RAM. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 101–108. Springer, 2024.

[27] David Naccache and David M'Raihi. Cryptographic Smart Cards. *IEEE micro*, 16(3):14–24, 1996.

[28] Katherine M. Shelfer and J. Drew Procaccino. Smart Card Evolution. *Commun. ACM*, 45(7):83–88, July 2002.

[29] Will Arthur, David Challener, Kenneth Goldman, Will Arthur, David Challener, and Kenneth Goldman. History of the TPM. *A Practical Guide to TPM 2.0: Using the New Trusted Platform Module in the New Age of Security*, pages 1–5, 2015.

[30] Trusted Computing Group. TCG Architecture Overview Specification, Version 1.1b. https://trustedcomputinggroup.org/wp-content/uploads/TCPA_Main_TCG_Architecture_v1_1b.pdf, 2003. Accessed: 2025-06-09.

[31] Sandro Pinto and Nuno Santos. Demystifying ARM Trustzone: A Comprehensive Survey. *ACM computing surveys (CSUR)*, 51(6):1–36, 2019.

[32] Wenhao Li, Yubin Xia, and Haibo Chen. Research on ARM TrustZone. *GetMobile: Mobile Computing and Communications*, 22(3):17–22, 2019.

[33] Open Mobile Terminal Platform. Advanced Trusted Environment: OMTP TR1. Technical report, Technical Report (v1. 1), 2009.

[34] Heedong Yang and Manhee Lee. Demystifying ARM TrustZone TEE Client API Using OP-TEE. In *The 9th International Conference on Smart Media and Applications*, pages 325–328, 2020.

[35] Guerney DH Hunt, Ramachandra Pai, Michael V Le, Hani Jamjoom, Sukadev Bhattiprolu, Rick Boivie, Laurent Dufour, Brad Frey, Mohit Kapur, Kenneth A Goldman, et al. Confidential Computing for OpenPOWER. In *Proceedings of the Sixteenth European Conference on Computer Systems*, pages 294–310, 2021.

[36] ARM. Arm Confidential Compute Architecture. https://www.arm.com/architecture/security-features/arm-confidential-compute-architecture, 2021.

[37] Victor Costan, Ilia Lebedev, and Srinivas Devadas. Sanctum: Minimal Hardware Extensions for Strong Software Isolation. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 857–874, 2016.

[38] Brian McGillion, Tanel Dettenborn, Thomas Nyman, and N Asokan. Open-TEE–an Open Virtual Trusted Execution Environment. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 400–407. IEEE, 2015.

[39] Dongxi Liu, Jack Lee, Julian Jang, Surya Nepal, and John Zic. A Cloud Architecture of Virtual Trusted Platform Modules. In *2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, pages 804–811. IEEE, 2010.

[40] Microsoft Azure. Confidential Computing Documentation. https://learn.microsoft.com/en-us/azure/confidential-computing/, 2025. Accessed: 2025-05-26.

[41] Google Cloud. Confidential VM Overview. https://cloud.google.com/confidential-computing/confidential-vm/docs/confidential-vm-overview, 2024. Accessed: 2025-05-26.

[42] Alibaba Cloud. TEE-Based Confidential Computing. https://www.alibabacloud.com/help/en/ack/ack-managed-and-ack-dedicated/user-guide/tee-based-confidential-computing/, 2025. Accessed: 2025-05-26.

[43] Juan Wang, Zhi Hong, Yuhan Zhang, and Yier Jin. Enabling Security-Enhanced Attestation with Intel SGX for Remote Terminal and IoT. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(1):88–96, 2017.

[44] Ayaz Akram, Anna Giannakou, Venkatesh Akella, Jason Lowe-Power, and Sean Peisert. Performance Analysis of Scientific Computing Workloads on General Purpose TEEs. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1066–1076. IEEE, 2021.

[45] Jinhua Cui, Jason Zhijingcheng Yu, Shweta Shinde, Prateek Saxena, and Zhiping Cai. Smashex: Smashing SGX Enclaves Using Exceptions. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, pages 779–793, 2021.

[46] Supraja Sridhara, Andrin Bertschi, Benedict Schlüter, and Shweta Shinde. SIGY: Breaking Intel SGX Enclaves with Malicious Exceptions & Signals. *arXiv preprint arXiv:2404.13998*, 2024.

[47] Pau-Chen Cheng, Wojciech Ozga, Enriquillo Valdez, Salman Ahmed, Zhongshu Gu, Hani Jamjoom, Hubertus Franke, and James Bottomley. Intel Tdx Demystified: A Top-Down Approach. *ACM Computing Surveys*, 56(9):1–33, 2024.

[48] Intel Corporation. Ant Group Develops Confidential Computing for SaaS. https://www.intel.com/content/www/us/en/customer-spotlight/stories/ant-group-customer-story.html, April 2024. Customer success story.

[49] Sam Lugani and Jai Haridas. How Confidential Computing Lays the Foundation for Trusted AI. https://cloud.google.com/blog/products/identity-security/how-confidential-computing-lays-the-foundation-for-trusted-ai/, May 2025. Google Cloud Blog, Identity Security.

[50] David Kaplan. Protecting VM Register State with SEV-ES. *White paper*, 46:158, 2017.

[51] AMD. Strengthening VM Isolation with Integrity Protection and More. *White Paper, January*, 53(2020):1450–1465, 2020.

[52] Markus Switalla. A Review of the Keystone Trusted Execution Framework. 2023.

[53] Muhammad El-Hindi, Tobias Ziegler, Matthias Heinrich, Adrian Lutsch, Zheguang Zhao, and Carsten Binnig. Benchmarking the Second Generation of Intel SGX Hardware. In *Proceedings of the 18th International Workshop on Data Management on New Hardware*, pages 1–8, 2022.

[54] Luigi Coppolino, Salvatore D'Antonio, Giovanni Mazzeo, and Luigi Romano. An Experimental Evaluation of TEE Technology: Benchmarking Transparent Approaches Based on SGX, SEV, and TDX. *Computers & Security*, 154:104457, 2025.

[55] Antonio Muñoz, Ruben Ríos, Rodrigo Román, and Javier López. A Survey on the (in)security of Trusted Execution Environments. *Computers Security*, 129:103180, 2023.

[56] Andrea Biondo, Mauro Conti, Lucas Davi, Tommaso Frassetto, and Ahmad-Reza Sadeghi. The Guard's Dilemma: Efficient {Code-Reuse} Attacks Against Intel {SGX}. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1213–1227, 2018.

[57] Benedict Schlüter, Supraja Sridhara, Mark Kuhne, Andrin Bertschi, and Shweta Shinde. {HECKLER}: Breaking Confidential {VMs} with Malicious Interrupts. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 3459–3476, 2024.

[58] Paul Kocher, Jann Horn, Anders Fogh, , Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre Attacks: Exploiting Speculative Execution. In *40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019.

[59] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown: Reading Kernel Memory from User Space. In *27th USENIX Security Symposium (USENIX Security 18)*, 2018.

[60] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H Lai. Sgxpectre: Stealing Intel Secrets from SGX Enclaves via Speculative Execution. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 142–157. IEEE, 2019.

[61] Li-Chung Chiang and Shih-Wei Li. Reload+ Reload: Exploiting Cache and Memory Contention Side Channel on AMD SEV. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 1014–1027, 2025.

[62] Erdem Aktas, Cfir Cohen, Josh Eads, James Forshaw, and Felix Wilhelm. Intel Trust Domain Extensions (TDX) Security Review. *Google security review*, 2023.

[63] Anh-Tien Le, Trong-Thuc Hoang, Ba-Anh Dao, Akira Tsukamoto, Kuniyasu Suzaki, and Cong-Kha Pham. A Cross-Process Spectre Attack via Cache on RISC-V Processor with Trusted Execution Environment. *Computers and Electrical Engineering*, 105:108546, 2023.

[64] Keystone Enclave. Support for Per-Thread Stack. https://github.com/keystone-enclave/keystone-runtime/issues/21, 2020. Accessed: 2025-06-01.

[65] Keystone Enclave. VA Double-Mapping Vulnerability. https://github.com/keystone-enclave/keystone/issues/83, 2019. Accessed: 2025-06-01.

[66] Intel Corporation. Intel® TDX Module 1.0 Public Specification. https://cdrdv2-public.intel.com/733568/tdx-module-1.0-public-spec-344425005.pdf, 2023. Accessed: 2025-06-08.

[67] Keystone Enclave Project. Keystone Enclave Project - Tutorials. https://docs.keystone-enclave.org/en/latest/Getting-Started/Tutorials/index.html, 2025. Accessed: 2025-06-08.

[68] Tim Geppert, Jan Anderegg, Leoncio Frei, Simon Moeller, Stefan Deml, David Sturzenegger, and Nico Ebert. Overcoming Cloud Concerns with Trusted Execution Environments?: Exploring the Organizational Perception of a Novel Security Technology in Regulated Swiss Companies. In *55th Hawaii International Conference on System Sciences (HICSS), virtual, 3-7 January 2022*, pages 6822–6829. University of Hawai'i at Manoa, 2022.

[69] Youren Shen, Hongliang Tian, Yu Chen, Kang Chen, Runji Wang, Yi Xu, Yubin Xia, and Shoumeng Yan. Occlum: Secure and Efficient Multitasking Inside a Single Enclave of Intel SGX. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 955–970, 2020.

[70] Chia-Che Tsai, Donald E Porter, and Mona Vij. {Graphene-SGX}: A Practical Library {OS} for Unmodified Applications on {SGX}. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, pages 645–658, 2017.

[71] Advanced Micro Devices, Inc. AMD SEV Key Management API Specification. https://www.amd.com/content/dam/amd/en/documents/epyc-technical-docs/programmer-references/55766_SEV-KM_API_Specification.pdf, 2020. Accessed: 2025-06-09.

[72] Jo Van Bulck, Frank Piessens, and Raoul Strackx. SGX-Step: A Practical Attack Framework for Precise Enclave Execution Control. In *Proceedings of the 2nd Workshop on System Software for Trusted Execution*, pages 1–6, 2017.

[73] Keystone Enclave Project. How to Debug - Keystone Documentation. https://github.com/keystone-enclave/keystone/blob/master/docs/source/Getting-Started/How-to-Debug.rst, 2025. Accessed: 2025-06-09.

[74] Craig Gentry. *A Fully Homomorphic Encryption Scheme*. Stanford university, 2009.

[75] Léo Ducas and Daniele Micciancio. Fhew: bootstrapping homomorphic encryption in less than a second. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 617–640. Springer, 2015.

[76] Paulo Martins, Leonel Sousa, and Artur Mariano. A Survey on Fully Homomorphic Encryption: An Engineering Perspective. *ACM Computing Surveys (CSUR)*, 50(6):1–33, 2017.

[77] Ran Canetti, Srinivasan Raghuraman, Silas Richelson, and Vinod Vaikuntanathan. Chosen-Ciphertext Secure Fully Homomorphic Encryption. In *IACR International Workshop on Public Key Cryptography*, pages 213–240. Springer, 2017.

[78] Furkan Aydin and Aydin Aysu. Leaking Secrets in Homomorphic Encryption with Side-Channel Attacks. *Journal of Cryptographic Engineering*, 14(2):241–251, 2024.

[79] Michael O Rabin. How to Exchange Secrets with Oblivious Transfer. *Cryptology ePrint Archive*, 2005.

[80] Adi Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979.

[81] Andrew Chi-Chih Yao. How to Generate and Exchange Secrets. In *27th annual symposium on foundations of computer science (Sfcs 1986)*, pages 162–167. IEEE, 1986.

[82] Silvio Micali, Oded Goldreich, and Avi Wigderson. How to Play ANY Mental Game. In *Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC*, pages 218–229. ACM New York, 1987.

[83] Simon Ott, Benjamin Orthen, Alexander Weidinger, Julian Horsch, Vijayanand Nayani, and Jan-Erik Ekberg. MultiTEE: Distributing Trusted Execution Environments. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, pages 1617–1629, 2024.

[84] Abdelrahaman Aly and Nigel P Smart. Benchmarking Privacy Preserving Scientific Operations. In *International Conference on Applied Cryptography and Network Security*, pages 509–529. Springer, 2019.

[85] Marcella Hastings, Brett Hemenway, Daniel Noble, and Steve Zdancewic. Sok: General Purpose Compilers for Secure Multi-Party Computation. In *2019 IEEE symposium on security and privacy (SP)*, pages 1220–1237. IEEE, 2019.

[86] Chuan Zhao, Shengnan Zhao, Minghao Zhao, Zhenxiang Chen, Chong-Zhi Gao, Hongwei Li, and Yu-an Tan. Secure Multi-Party Computation: Theory, Practice and Applications. *Information Sciences*, 476:357–372, 2019.

[87] Zhao Chang, Dong Xie, and Feifei Li. Oblivious RAM: A Dissection and Experimental Evaluation. *Proceedings of the VLDB Endowment*, 9(12):1113–1124, 2016.

[88] Emil Stefanov, Marten van Dijk, Elaine Shi, T-H Hubert Chan, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: An Extremely Simple Oblivious RAM Protocol. *Journal of the ACM (JACM)*, 65(4):1–26, 2018.

[89] Christopher W Fletcher, Marten van Dijk, and Srinivas Devadas. A Secure Processor Architecture for Encrypted Computation on Untrusted Programs. In *Proceedings of the seventh ACM workshop on Scalable trusted computing*, pages 3–8, 2012.

[90] Christopher W Fletchery, Ling Ren, Xiangyao Yu, Marten Van Dijk, Omer Khan, and Srinivas Devadas. Suppressing the Oblivious RAM Timing Channel While Making Information Leakage and Program Efficiency Trade-Offs. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, pages 213–224. IEEE, 2014.

[91] AKM Mubashwir Alam and Keke Chen. TEE-MR: Developer-Friendly Data Oblivious Programming for

Trusted Execution Environments. *Computers & Security*, 148:104119, 2025.

[92] Shufan Fei, Zheng Yan, Wenxiu Ding, and Haomeng Xie. Security Vulnerabilities of SGX and Countermeasures: A Survey. *ACM Computing Surveys (CSUR)*, 54(6):1–36, 2021.

[93] Seny Kamara and Tarik Moataz. SQL on Structurally-Encrypted Databases. In *Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part I 24*, pages 149–180. Springer, 2018.

[94] Alankrit Chaturvedi. Security Challenges and Solutions in MongoDB. *International Journal of Science and Research (IJSR)*, 9(1), 2020.

[95] Archita Agarwal, David Cash, Marilyn George, Seny Kamara, Tarik Moataz, and Jaspal Singh. Updatable Private Set Intersection from Structured Encryption. *Cryptology ePrint Archive*, 2024.

[96] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Cătălin Roşu, and Michael Steiner. Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation. *Cryptology ePrint Archive*, 2014.

[97] David Cash and Stefano Tessaro. The Locality of Searchable Symmetric Encryption. In *Advances in Cryptology–EUROCRYPT 2014: 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings 33*, pages 351–368. Springer, 2014.

[98] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'neill. Generic Attacks on Secure Outsourced Databases. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1329–1340, 2016.

A Queries for Reproducibility

The listed query was used in Scopus to retrieve relevant literature. It was meant to find features regarding efficiency, security, functionality, and usability as presented in surveys for the four chosen technologies.

TITLE-ABS-KEY((”trusted execution environment*” OR ”trusted environment*”) AND (”cloud” OR ”server-side”) AND (efficien* OR secur* OR function* OR usabil*) AND (compar* OR survey OR evaluation OR review) AND (”Intel SGX” OR ”AMD SEV” OR ”Intel TDX” OR ”Keystone”))

Filters:

- Only include articles and conference papers.
- Publication stage should be final.
- Exclude non-English items.

”[technology-name]” ”[keyword]”

Entries for *[technology-name]*: ”fully homomorphic encryption”, ”oblivious ram”, ”structured encryption”, ”secure multi-party computation”, ”secure multiparty computation”.

Entries for *[keyword]*: ”benchmark”, ”performance”, ”use case”, ”efficiency”, ”security”, ”attacks”, ”threat model”, ”usability”, ”functionality”, ”trusted execution environment”.

B Properties of TEEs

Table 1: Comparison of TEE implementations on functionality.

Technology	Attestation	Data sealing
Intel SGX [11]	QE with EPIDs	Enclave/Sealing Identity modes
Keystone [12]	Hardware with secure boot-signed key	SBI-derived key
Intel TDX [13]	QE with Intel PCK	N/A (VM-based)
AMD SEV [14]	Hardware root-of-trust only	N/A (VM-based)

Table 2: Comparison of TEE implementations on efficiency.

Technology	Performance overhead	Integrity overhead
Intel SGX [11]	High (up to 126× I/O in early versions)	EPC metadata + MACs
Keystone [12]	Near-native (higher than CVMs)	Runtime software or MEE hardware
Intel TDX [13]	Near-native (28.6% avg I/O overhead)	MACs for integrity
AMD SEV [14]	Near-native (higher than TDX)	SEV: None; SNP: Reverse Map Table

Table 3: Comparison of TEE implementations on security.

Technology	TCB size	Known Vulnerabilities (June 2025)
Intel SGX [11]	Minimal (CPU microcode + QE)	48 CVEs
Keystone [12]	Smallest (can be variable)	No CVEs (GitHub issues)
Intel TDX [13]	Largest (incl. TDX Module Loader + QE)	10 CVEs (higher min score: 5.6)
AMD SEV [14]	Large (more hardware)	49 CVEs

Table 4: Comparison of TEE implementations on usability.

Technology	Deployment	Debugging
Intel SGX [11]	Code modifications (SDK/Gramine/Occlum)	Debug bit in metadata (restricted in Release mode)
Keystone [12]	RISC-V binaries or SDK	SM debuggable; enclave apps restricted
Intel TDX [13]	Minimal changes (TDX-Enlightened OS)	On-TD/Off-TD debugging
AMD SEV [14]	No modifications needed	API-based (NODBG bit)

C Comparison with Privacy-Preserving Techniques

Table 5: Functionality and Usability Comparison of Privacy-Enhancing Techniques

Technique	Computation Type	Parties Communication	Applicability	Use Cases
FHE	Any computation	Non-interactive Client-server	Available in open-source libraries	Medical data analysis Recommender systems Confidential ML
MPC	General computation (excluding specialized protocols)	Multiple clients or distributed parties	Used in practice but with limitations	Secure auctions DNA comparison Collaborative research
ORAM	Data access	Non-interactive Client-server	Used in secure processors and oblivious DBs	SGX integration ObliDB, Signal protocol
StE	Specific data access on encrypted structures	Non-interactive Client-server	Practical protocols for specific structures	Encrypted DBMS (e.g. MongoDB)
TEE	Any computation	Interactive Client-server with attestation service	Optional in real world cloud deployment	Data analytics Trusted AI workloads Medical Federated Learning

Table 6: Security and Performance Comparison of Privacy-Enhancing Techniques

Technique	Threat Model	Information Leakage	Performance Overhead
FHE	IND-CCA2 Adaptive attack	None by itself	High: Key Generation & Polynomial Operations
MPC	Semi-honest or malicious	Nothing beyond function output	Constant or Linear
ORAM	Semi-honest or malicious	Leakage through side-channel attacks	Logarithmic
StE	Semi-honest	Access pattern sometimes response volume	Sublinear
TEE	Malicious actor controlling server	Access patterns, plaintext in CPU	Generally near-native, bottleneck in I/O heavy