Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Confidential Computing across Edge-to-Cloud for Machine Learning: A Survey Study

Sm Zobaed[1] | **Mohsen Amini Salehi**[2]

[1]Computer Science & Engineering Technology, University of Maryland Eastern Shore, Maryland, USA

[2]Computer Science & Engineering Department, University of North Texas (UNT), Texas, USA

Correspondence
Sm Zobaed
Email: szobaed@umes.edu

Present address
11868 College Backbone Rd, Princess Anne, MD 21853

Confidential computing has gained prominence due to the escalating volume of data-driven applications (*e.g.,* machine learning and big data) and the acute desire for secure processing of sensitive data, particularly, across distributed environments, such as edge-to-cloud continuum. Provided that the works accomplished in this emerging area are scattered across various research fields, this paper aims at surveying the fundamental concepts, and cutting-edge software and hardware solutions developed for confidential computing using trusted execution environments, homomorphic encryption, and secure enclaves. We underscore the significance of building trust in both hardware and software levels and delve into their applications particularly for regular and advanced machine learning (ML) (*e.g.,* large language models (LLMs), computer vision) applications. While substantial progress has been made, there are some barely-explored areas that need extra attention from the researchers and practitioners in the community to improve confidentiality aspects, develop more robust attestation mechanisms, and to address vulnerabilities of the existing trusted execution environments. Providing a comprehensive taxonomy of the confidential computing landscape, this survey enables researchers to advance this field to ultimately ensure the secure processing of users' sensitive data across a multitude of applications and computing tiers.

KEYWORDS
Confidential Computing, Edge-to-Cloud Continuum, Confidential ML, Trusted Execution Engine (TEE).

1 | INTRODUCTION

Every second, on average, more than 1.7 Megabytes of data is generated per person via different digital means including IoT sensors, organizational documentation, RSS feeds, transaction records, streaming, social media activities, and more to be processed by smart applications. The volume of data is growing faster than ever before, and it is expected that the worlds total data volume will exceed 149 zettabytes within the next three years[1]. In response, hyperscaler cloud providers (*e.g.,* AWS, Azure, and Google cloud) offer various services for large-scale data storage, processing, and analysis in a distributed manner—across edge-to-cloud continuum. Although the wide adoption of AI/ML-based applications operating across IoT and edge-cloud systems has accelerated advancements in various aspects of human's life, they have created a larger data exposure surface, and less user-control over his/her data and infrastructure. In particular, the emergence of off-premises ML-based and generative AI (*e.g.,* ChatGPT[2], Llama) applications in almost every domain—from personalized healthcare, search, and archives to finance, assistive technology, and social networks[3]—has further complicated the matter of confidentiality and has brought it to the forefront of users' priorities. Despite various security services offered by cloud providers (*e.g.,* AWS IAM[4], GuardDuty[5] and Security Hub[6]), organizations and individuals still have major concerns due to **lack of a natively-designed confidentiality solution** that encompasses the entirety of the hardware, systems (IoT-edge-cloud middleware), and application ecosystem.

Maintaining data confidentiality while data is *"at rest"* (*e.g.,* stored on the cloud storage) is a widely known problem with well-studied solutions[7]. Similarly, maintaining data confidentiality while it is *"in transit"*) state has established solutions[8], such as applying transport layer security (TLS) protocol[9]. Nevertheless, the main data security and confidentiality challenge brought up by pervasive computing across IoT-edge-cloud is while the data is *"in use"*, *i.e.,* while data is being processed[10].

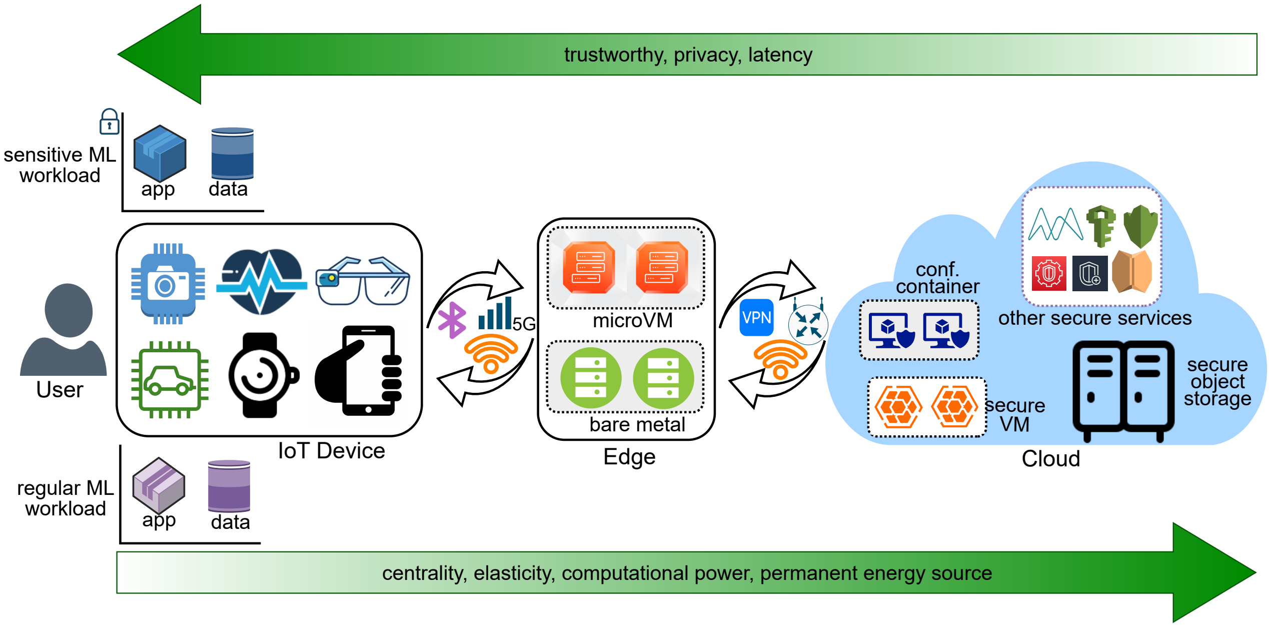


FIGURE 1 Overview of confidential computing across edge-to-cloud continuum. Sensitive and regular ML workloads are requested from various sensors and IoT devices. Depending on the workload type, requests travel through various LAN and WAN networks: a LAN between the device and edge tiers (*e.g.,* via Bluetooth, Wi-Fi, and 5G); and a WAN between edge and cloud tiers (*e.g.,* via VPN, Wi-Fi, 5G, or a hybrid network). As a result, the workload can be executed collaboratively across secure services hosted on the IoT, edge, and cloud tiers, such as secure VM, container, and object storage; plus other secure cloud services (*e.g.,* AWS Macie, IAM, KMS, GuardDuty, WAF, Shield).

As such, securing data-processing, particularly under distributed settings that has more exposure and are often controlled by multiple autonomous entities, has become the pressing desire. It is to address this very desire that the area of **confidential computing**, a.k.a. *secure computing*, is fast-emerging.

With the prevalence of cloud computing, confidential computing over privacy-preserving data has to occur on shared pay-per-use infrastructures. Specifically, cloud services, such as Infrastructure as a Service (IaaS), traditionally offer computing infrastructure within Virtual Machines (VMs). While isolation and virtualization techniques (*e.g.,* VMs and containers) have been instrumental, the proliferation of cloud services has exposed software systems to new security vulnerabilities and increased risks[3,11] that are mainly rooted in the off-premise nature of clouds. To mitigate the threats introduced by the cloud services, edge-cloud systems have emerged not only to reduce the latency of accessing compute service, but also to enable maintaining sensitive user data on-premise (*i.e.,* at the edge)[12,13,14]. However, as indicated in Figure 1, there are two inherent problems with the edge computing paradigm: (**a**) They are resource- and (sometimes) energy-limited, hence, they are often insufficient to process massive volume of sensitive data within real-time constraints; (**b**) Despite of being on-premise and more trustworthy, edge nodes have their own vulnerabilities, due to wide dispersion and physical exposure.

Although numerous studies have been undertaken to improve the real-timeness and scalability across the edge-to-cloud, comparatively less attention has been paid to the confidential computing across heterogeneous tiers of the continuum. Figure 1 represents the path for confidential computing across the user/IoT devices to the edge, and cloud. The sensitive and regular data, generated by the devices, are first pre-processed on the devices using trusted applications; Then, it travels through various forms of LAN and WAN networks (*e.g.,* Bluetooth, Wi-Fi, VPN, and 5G) to be collaboratively processed by the services hosted on the edge and cloud tiers.

Establishing confidential computing of smart (often ML-based) applications within the context of IoT-edge-cloud entails understanding and enabling trust across hardware and software levels. At the hardware level, trust should be achieved via implementing proper root-of-trust flows that ensure a system is booted from a trusted image (signed by a trusted provider) into a valid state. Hardware vendors, such as Intel, AMD, and ARM, have come forward with Trusted Execution Environment (TEE)[15] technology that provides a hardware-assisted fully isolated execution environment that is deemed as the core of confidential computing. TEE offers a tamper-resistant processing environment, running in a separate kernel providing an acceptable level of originality, integrity, and confidentiality for the data and code to be executed[15,16,12]. *The advantage of leveraging TEE-enabled cloud and edge infrastructures is that these systems remain secure even if their system software are compromised.* According to the Confidential Computing Consortium (CCC), TEE is the foundation for confidential computing, however, any TEE solution should also provide the ability for remote attestation, so that its trustworthiness can be verified. An application is considered as trusted, if it is executed within the TEE, its source code is available and inspectable, built into binaries with bit-exact reproducibility, and signed by a trusted authority[11]. Moreover, the data stored on and processed by a trusted application must be protected, and every interaction with any untrusted segments (*i.e.,* inter-application and remote service calls) is executed securely.

Provided the increasing prevalence and prominence of confidential computing across IoT-edge-cloud systems, and their large exposure surface, *the **goal** of this study is to pinpoint the scope and challenges of confidential computing within the context of edge-cloud systems*. For that purpose, this study includes the following contributions:

1. Describing the nuts and bolts of confidential computing across edge-cloud (Section 2).
2. Presenting a comprehensive taxonomy that shows the scope of confidential computing across edge-to-cloud (Section 3).
3. Demystifying the anatomy of the *trusted hardware technologies* as the main prerequisite to establish confidential computing across the continuum (Section 4).
4. Surveying the spectrum of middleware solutions for confidential computing that spans from the user-device level to the cloud data centers (Section 5).
5. Elaborating on the trusted application development frameworks to build confidential software solutions, particularly, for ML (Section 6).
6. The last contribution of this study is in Section 8 where we identify the areas of research that demand further exploration from the community to overcome major remaining challenges of confidential computing across edge-to-cloud.

2 | KEY CONCEPTS OF CONFIDENTIAL COMPUTING

2.1 | Trusted Execution Environment (TEE)

A trusted execution environment (TEE) is a tamper-resistant processing environment that executes on a separation kernel. In confidential computing, TEEs guarantee the protection of the code and data loaded inside an isolated area, which is also referred to as *enclave*. The design goal of TEE is to prevent the manipulation of various software adversaries (*e.g.,* malware and hacked OS) or even hardware adversaries who have physical access to the platform. TEE offers hardware-enforced security features that include isolated execution, guaranteeing the integrity and confidentiality of the enclave, and the ability of code authentication to execute inside the enclave through attestation. Figure 2 represents the building blocks and the core design aspect of TEE and we shed light on them in the rest of this section.

2.2 | Building Blocks of TEE

GlobalPlatform is a widely known international association that identifies, develops, and promotes technical standards to facilitate secure and interoperable deployment and management of embedded applications on secure chips. They have released the detailed hardware and software architecture of TEE highlighting full isolation between Rich OS Environment (ROE) and TEE[15]. Each environment separately allocates their own resources *e.g.,* RAM, ROM, CPU, and OS, while communications between the two environments are only performed via the TEE Client API.

Kernel Separation is a fundamental concept of the TEE architecture, as it underpins the essential property of isolated execution. This concept was first introduced in the context of a dual-execution environment model[17]. The primary objective of the separation kernel is to facilitate the cohabitation of diverse systems requiring different security levels on a single platform. The separation kernel divides the system into several partitions, ensuring robust isolation between them, except for the provision of a controlled interface for inter-partition communication. To assess the credibility of a kernel separation implementation, Sabt *et al.*developed a trust function based on its integrity measurement metrics that returns the level of trust of a given TEE[16].

The specific security requirements for separation kernels are outlined in the Separation Kernel Protection Profile (SKPP)[18]. SKPP describes the separation kernel as a combination of hardware, firmware, and/or software mechanisms with a core function

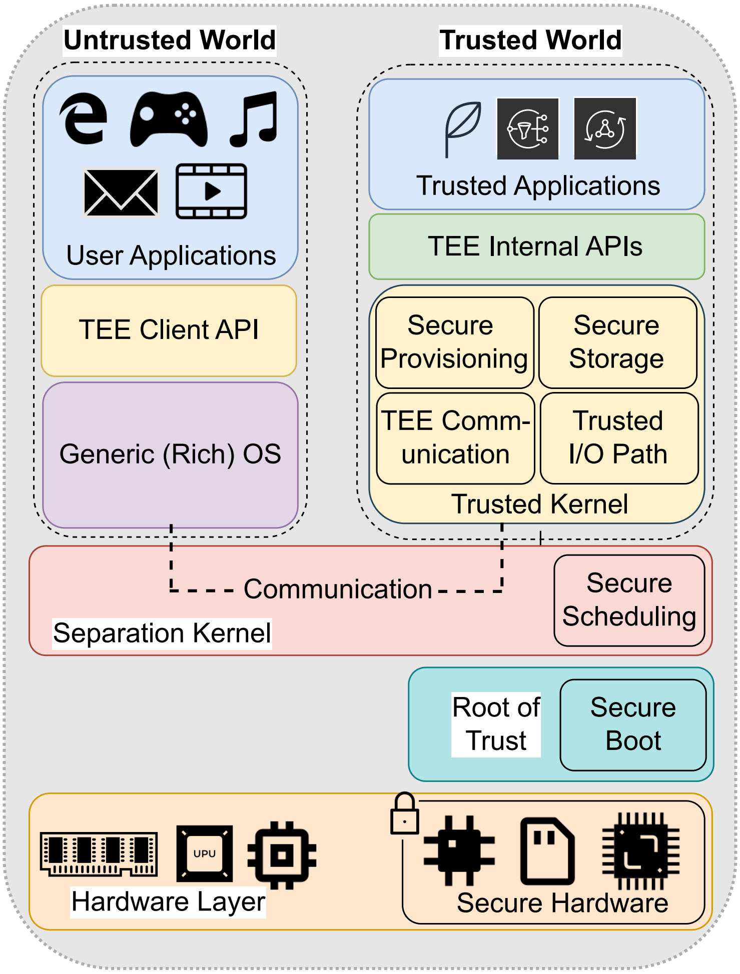


FIGURE 2 High-level architectural overview of TEE building blocks

to establish, isolate, and manage information flow between the subjects and the partitions. In contrast to conventional security kernels like operating systems, micro-kernels, and hypervisors, the separation kernel is more streamlined, offering both time and space partitioning, thus, contributing to system security and efficiency.

Secure Boot is a key component of a Trusted Execution Environment (TEE) that assures only authenticated software can be executed during the system boot-up phase. This mechanism acts as a critical line of defense against tampering attempts on boot loaders, key operating system files, and more. A device equipped with secure boot can interrupt the bootstrap process if modifications in the loaded code are detected, reinforcing the integrity of the system's operations.

Implemented at the firmware level, secure boot operates on the principle of trust chain establishment. As illustrated by Arbaugh *et al.*[19], secure boot verifies the integrity of each subsequent component according to a predefined reference value. This process unfolds across several stages, forming a chain of trust. The starting point of this chain is the Root of Trust (RoT), typically a hardware-based module, which is intrinsically trusted to form the foundation for verifying the trustworthiness of subsequent layers. Upon system start-up, the RoT initiates the process by verifying the bootloader's integrity. Once validated, the bootloader checks the integrity of the next software component, and this sequence continues. Each link in the chain is responsible to validate the subsequent component's authenticity and integrity before allowing its execution. This methodical and layered approach to validation ensures that the execution environment remains secure and trustworthy.

A critical part of secure boot in TEEs involves the use of cryptographic signatures. Each software component involved in the boot process is signed using a private key. During the boot process, the signature of each component is verified against a trusted public key, certifying the software's origin and maintaining its integrity.

Secure Scheduling assures a *balanced* and *efficient* coordination between the TEE and the rest of the system. Indeed, it should assure that the tasks running in the TEE do not affect the responsiveness of the main OS. Thus, the scheduler is often designed preemptive. Furthermore, the scheduler should take real-time constraints into consideration. Authors in[20] propose a secure scheduler that enhances the responsiveness of the main OS without compromising the real- time performance of the system.

Inter-Environment Communication is the mechanism that allows the TEE to interact with the rest of the system. While this provides numerous benefits, it also presents new threats, including message overload attacks, user and control data corruption attacks, memory faults caused by shared page removals, and unbound waits due to the non-cooperation of the system's untrusted part[16].

The chosen mechanism for inter-environment communication must ensure reliability, maintain minimal overhead, and protect communication structures. Several models of communication have been identified in the literature, including the GlobalPlatform TEE Client API, the secure Remote Procedure Call (RPC) of Trusted Language Runtime[21], and the real-time RPC of SafeG[22]. These models are designed to address the challenges associated with secure inter-environment communication and contribute to the secure and efficient operation of the TEE.

Secure Storage is a fundamental component of a TEE, designed to ensure the confidentiality, integrity, and freshness of stored data, thereby preventing replay attacks and maintaining state continuity[23]. Access to this data is tightly controlled which ensures only authorized entities can access it[16]. A popular implementation approach is through sealed storage. This strategy is predicated on three core components: (1) an integrity-protected secret key exclusively accessible by the TEE; (2) cryptographic protocols, including authenticated encryption algorithms; and (3) a mechanism for data rollback protection, such as replay-protected memory blocks (RPMB)[24].

Trusted I/O Path is designed to safeguard the authenticity and, if necessary, the confidentiality of the communication channel between the TEE and peripherals such as keyboards or sensors. This design strategy ensures that input and output data are protected from eavesdropping or tampering by malevolent applications. Specifically, the Trusted I/O path provides protection against four major classes of attacks: screen-capture, keylogging, overlaying, and phishing. The establishment of a trusted path to user-interface (UI) enabled devices extends the functionality within the TEE, facilitating direct interaction between the human user and applications running within the TEE.

2.3 | Measuring the Trustworthiness of a System

Evaluating trust, especially in a comparative sense between different systems employing TEEs, requires a mechanism to quantify trust. In colloquial language, trust refers to the conviction of the integrity and reliability of a person or entity. This subjective property, while intrinsic to human relationships, is challenging to numerically capture. In computer systems, the notion of trust becomes even more nuanced. Here, an entity is typically trusted if it has consistently behaved as expected and will continue to do so in the future.

In the realm of computing, trust can be categorized as static or dynamic. Static trust is derived from an exhaustive evaluation against specific security benchmarks before the deployment of a system. A prominent example of a standard providing assurance measures for security evaluation is the Common Criteria[16]. This standard enumerates seven evaluation assurance levels (EAL1-EAL7), with each higher level encompassing the requirements of its predecessor. Thus, under static trust, the trustworthiness of a system is assessed once before its deployment. Conversely, dynamic trust is contingent upon the state of an active system and fluctuates accordingly. With the trust status of a system undergoing continuous changes, dynamic trust necessitates a regular assessment of the system's trustworthiness throughout its lifecycle.

TEE systems embody a hybrid form of trust encompassing both static and semi-dynamic aspects. Prior to deployment, a TEE is subjected to stringent certification procedures, including verification of its security level according to a protection profile, which outlines a predefined set of security requirements. For instance, the GlobalPlatform protection profile conforms to EAL2[16]. Moreover, each boot sequence includes a 'Root of Trust' (RoT) check, ensuring that the TEE in operation is the certified one provided by the platform vendor. This safeguards the integrity of the TEE code[25,26].

Once operational, the TEE code's integrity is protected by the underlying separation kernel. As such, the trust in TEEs is considered semi-dynamic the TEE is not expected to change its trust level while running due to the protection provided by the separation kernel. In this trust model, the trust measurements are integrity assessments, and the trust score is a binary indicator of the code's integrity state. The TEE is deemed trusted when its trust score is true and untrusted otherwise. The reliability of this trust score hinges on the integrity measurement definitions.

To assess the real trust value, Sabt *et al.*proposed a trust function f(TEE, protection profile, RoT, measurements)[16]. This function returns the trust level of a given TEE based on three parameters: the certifying protection profile, the reliability of RoT, and the integrity measurements. Such a function provides a quantitative basis to weigh trust in TEE systems.

2.4 | Trustworthy Code Execution

Confidential computing fundamentally ensures trustworthy code execution through the strategic isolation of enclaves, or Trusted Execution Environments (TEEs), from untrusted environments[25]. This secure partitioning is facilitated by the rigorous protection and management of enclave memory sections, orchestrated via the hardware and system software of the TEE. Distinct processors such as ARM, Intel, and AMD have each conceived their unique architectural approaches to facilitate this isolation.

Intel Software Guard Extensions (SGX)[27] employs a memory encryption engine to provide assured secrecy, integrity, and freshness of the CPU-DRAM traffic within the enclave memory ranges. In contrast, ARM TrustZone[28] utilizes distinct page tables, hardware privilege layers (*e.g.,* EL3 and Secure EL2/EL1/EL0), and a TrustZone address space controller (TZASC4) for its implementation.

2.5 | Remote Attestation in Confidential Computing

Remote attestation[25,29] enables the user (a.k.a. confidential workload owner) to ascertain the level of trust/integrity of a remote TEE prior to transmitting sensitive data/code. It enables the owner to authenticate the hardware, validate the trusted state of a distant TEE, and determine whether the intended program is operating securely within the TEE. A third party, besides the user and host, could perform the attestation[29,26]. Also, the service provider and enclave owner could execute the attestation directly[26]. An attestation server could be launched by the processor manufacturer, as with Intel's SGX attestation service[27], which is distinct from the cloud service provider.

2.6 | Confidential Computing vs Secure Computing

For instance, consider a financial institution that processes sensitive transactions. With confidential computing, portions of this process, the actual transaction processing, are isolated within a TEE, ensuring that even if the operating environment is breached, the transaction remains secure. Secure computing, however, goes beyond this level protection and also delves into areas such as the security of transaction data, as it enters and exits the system (data in transit) and while the data is at rest (stored).

3 | TAXONOMY OF CONFIDENTIAL COMPUTING ACROSS EDGE-TO-CLOUD

Figure 3 represents a comprehensive taxonomy of the scopes where confidential computing can be implemented, particularly, with respect to edge-to-cloud continuum and machine learning (ML) applications. The first level of the taxonomy broadly covers confidential computing at the hardware, system middleware, and application levels. These three are the main thrusts of this paper and subsequent Sections 4 to 6 essentially elaborate on each one of these thrusts.

Hardware is the cornerstone of the ongoing momentum towards confidential computing. At the *hardware* level, confidential computing is achieved in two main ways: (A) *Trusted Execution Environments (TEE)*[16] that deals with the secure execution of applications; and (B) *Secure Hardware Modules*[30] that are not in charge of application execution and deal with other aspects such as secure storage, secure key management, *etc.* . Prominent implementations of TEE, namely Intel Software Guard Extension (SGX)[31], AMD Secure Encrypted Virtualization (SEV)[32], and ARM TrustZone[28], are studied in Section 4.1. The other hardware category, *Secure Modules*, includes two main classes: Trusted Platform Module (TPM), and Secure Element (SE) that are elaborated in Section 4.2.

At the system software (a.k.a. *middleware*) level, we have considered the *computing paradigm* and *computing tier* aspects where the former encompasses serverless and serverful computing paradigms. The *computing tier* deals with the device, edge, fog, and cloud tiers and studies the confidential computing research works undertaken within and across these tiers.

At the *application level*, we discuss how developers and programming-level tools can reinforce or undermine confidential computing. In particular, we study the impact of software architecture (monolithic vs micro-service) used to develop an application on confidential computing. We unfold the structure of monolithic confidential computing applications (*e.g.,* S3C[33], S3BD[34], and ClusPr[35]), as well as microservice-based ones (*e.g.,* SAED[36]). What is more, we study how application partitioning (across edge-cloud) and application-data encryption can influence confidential computing.

Last but not least, at the application level, in Section 6.1.1, we pay specific attention to the interplay of machine learning (ML) and confidential computing that have given birth to a new research field, called *confidential ML*. In particular, we study federated learning, differential learning, and differential privacy techniques and their mutual impact with the confidential computing.

Confidential computing research is not a monolithic field, but rather it is a rich tapestry of interconnected research areas. Each of these areas contribute to our understanding and ability to implement confidential computing. To illuminate the breadth and depth of this field, we have compiled a comprehensive summary of prior studies in Table 1. This table categorizes each study according to its alignment with one or more areas of our taxonomy. By viewing these studies in aggregate, we can discern correlations, contributions, and gaps in the existing research. A more thorough view to the research works listed in the table provides a deeper understanding of the state-of-the-art in confidential computing. The table columns are in alignment with the categorization introduced in the taxonomy of Figure 3. In particular, for each study, we have highlighted the type of hardware being used and the computing paradigm being targeted by that study. The "Remarks" column focuses on the shortcoming of that study and highlights the potential future studies that can be accomplished.

TABLE 1: Prior studies undertaken on different aspects of edge-cloud confidential computing

Prior Studies	Hardware	Computing Paradigm	Service Model	Summary	Remarks
Sabt *et al.*[16] (2015)	TEE	User-end	IaaS	+ Present core components of TEE + Survey industrial & academic TEE	+ Study ARM TrustZone only - Discuss deprecated TEEs
Shepherd *et al.*[15] (2016)	TEE TPM, SE	User-end	Bare Metal	+ Survey on versatile trusted hardware components	- Limited discussion on hardwares' vulnerabilities - Little discussion on middle-wares and app framework
Ning *et al.*[12] (2018)	TEE	Serverful	IaaS	+ Evaluate edge-centric TEE architectures + Deploy TrustZone-based TEE on edge	- Do not discuss SGX-based TEEs - Do not discuss end-to-end confidential computing
Aublin *et al.*[37] (2018)	TEE	User-end	SaaS	+ Propose trusted logger service library for TA	- Focus on Intel SGX only - Do not guarantee end-to-end confidential computing
Nguyen *et al.*[38] (2018)	TEE	Serverful	SaaS	+ Propose storing mechanism for IoT logs on SGX-enhanced cloud nodes	- Focus on Intel SGX only - Do not ensure end-to-end data confidentiality
Valadares *et al.*[39] (2018)	TEE	User-end	SaaS	+ An attestation scheme to prevent physical attacks on IoT + Propose secure key management through key vault running in SGX	- Data integrity is not assured - Fall short to detect compromised source
Ayoade *et al.*[40] (2019)	TEE	User-end	SaaS	+ Secured data processing scheme with isolated trusted modules	- Focus on Intel SGX only - Do not ensure end-to-end confidential computing
Pinto *et al.*[41] (2019)	TEE	User-end	IaaS	+ Survey on TEEs and hardware-assisted virtualization + Discuss vulnerabilities of TC	- Only focus on ARM TrustZone
Van *et al.*[42] (2019)	TEE	Serverful	IaaS	+ Discover vulnerabilities in all open source SDKs used for enclave deployment + Study few TEE hardwares	- Only focus on trusted hardware

Brenner *et al.*[43] (2019)	TEE	Serverless	FaaS	+ Study library dependencies for securing FaaS clouds	- Do not consider TrustZone and SEV-enabled hardwares - Consider enclave is attack free
Ibrahim *et al.*[44] (2019)	TEE TPM	User-end	IaaS	+ Compare different TPMs + Discuss secure migration and cloning of VMs	- Briefly discuss TPM-driven application framework - Do not discuss using TPM across edge-cloud
Aslanpour *et al.*[45] (2021)	TEE	Serverless edge	Bare Metal	+ Study serverless edge + Discuss scopes and challenges of serverless edge computing	- Only focusing on middleware and applications
Valadares *et al.*[28] (2021)	TEE	User-end	IaaS	+ Survey on TEEs that are used in edge-cloud systems	- Do not cover cross-system application frameworks
Menetrey *et al.*[29] (2022)	TEE	Serverful	IaaS	+ Study attestation mechanisms for TEEs	- Do not discuss end-to-end confidential computing
Li *et al.*[46] (2022)	TEE	Serverless	FaaS	+ Survey on serverless apps deployed in SGX enclaves + Propose SGX extension for confidential serverless jobs	- Do not consider SEV and TrustZone
Wang *et al.*[47] (2022)	TEE	Serverless	FaaS	+ Use deep learning to optimize task allocation between local devices and edge servers + Propose SGX extension for confidential serverless jobs	- Specific hardware dependency may restrict app scalability - Limited consideration of security threats and broader system vulnerabilities
Zhao *et al.*[48] (2023)	TEE	serverless	FaaS	+ Introduce reusable enclave to solve the cold start problem in confidential serverless computing	- Overlooks security issue in states and enclave resets - Relies heavily on Intel SGX
Wu *et al.*[49] (2021)	N/A	Serverful	IaaS	+ Employ blockchain for secure task offloading between IoT and hybrid cloud-edge + Minimized energy consumption and response times without leveraging system dynamics	- Does not address confidential technologies such as SEV and Arm TrustZone
Qu *et al.*[50] (2021)	N/A	Serverful	IaaS	+ Integrate federated learning and blockchain to secure task offloading across IoT, edge, and cloud	- Does not indicate trusted hardware utilization
Li *et al.*[51] (2021)	N/A	Serverful	IaaS	+ Propose task offloading scheme for TrustZone-enabled edges	- Scalability challenges in multi-core setups

4 | CONFIDENTIAL COMPUTING HARDWARE

4.1 | TEE Hardware Technologies for Confidential Computing

Secure Space and *Memory Encryption* are crucial TEE hardware technologies that underpin the functioning of confidential computing systems. Understanding these technologies and how they are applied is key to grasping how hardware-based TEEs function.

Secure space, also referred to as isolated execution, is a hardware feature that allows the creation of private regions within the main memory. These regions, also known as enclaves or secure enclaves, function as separate execution environments where code and data can be securely processed and isolated from the rest of the system. This is to ensure that sensitive data is accessible only to authorized processes and is shielded from any potentially malicious software running on the same hardware platform. Technologies such as Intel SGX[52] and ARM TrustZone[28] utilize this concept, creating isolated secure spaces where sensitive code and data can operate in a controlled, confidential manner.

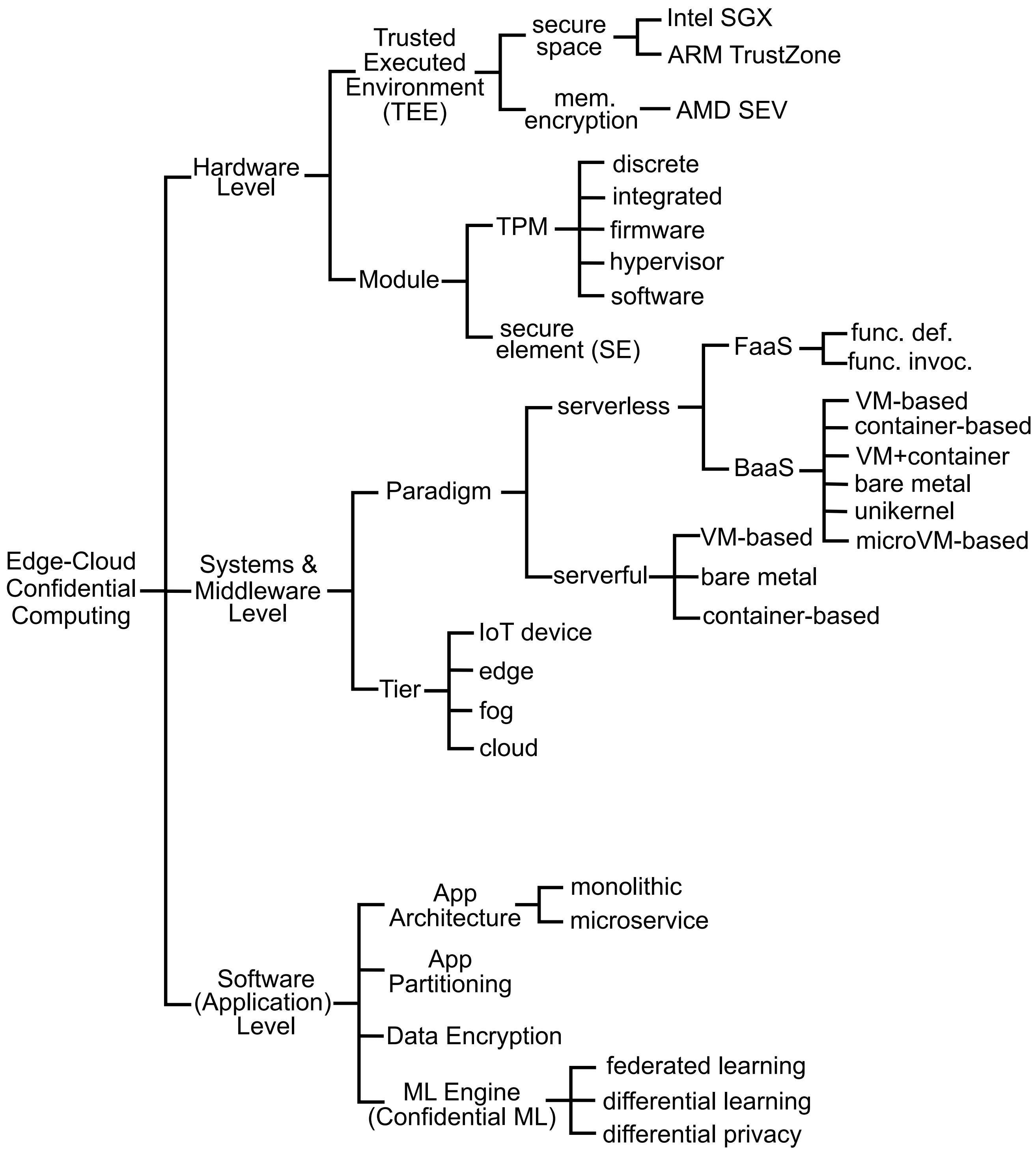


FIGURE 3 Taxonomy of confidential computing encompassing hardware level, systems and middleware, and software application levels

Memory encryption is a technology used to secure data at rest, in use, and in transit within and between the processor and memory. By encrypting the data within memory, this technology makes it incredibly difficult for an attacker to make sense of any data they might access, effectively preserving the confidentiality and integrity of the data. The encryption and decryption processes are carried out within the CPU, making the procedure transparent to applications, and do not require any modifications to the software. This technology is widely used in AMD's Secure Encrypted Virtualization (SEV)[32] where all the memory contents of a virtual machine are encrypted, ensuring the security of data from any possible external threats.

The secure space and memory encryption technologies are not mutually exclusive and can often be combined in various ways to provide multiple layers of security, enhancing the overall security posture of the system.

Table 2 compares the attributes of these three TEE hardware implementations from various aspects. In the rest of this section, we elaborate on these attributes for each hardware and compares them against each other.

4.1.1 | Intel Software Guard eXtension (Intel SGX)

Intel SGX, an extension to the x86-64 instruction set, provides a facility for executing select portions of an application within a hardware-assisted TEE, specifically within a secure enclave, as noted in Figure 4. This enclave delineates a trusted world where code and data are isolated, maintaining their integrity and confidentiality from the rest of the system. Within this trusted environment, both the application code and data are preserved in an encrypted state and are decrypted solely within the CPU, which guarantees confidentiality. Furthermore, the enclave ensures robust security against external interference from both other applications and system software, irrespective of their privilege levels, whether they operate in the user mode (ring 3) or the kernel mode (ring 0).

SGX creates a limited size of the encrypted memory region, referred to as *Enclave Page Cache (EPC)*, where all the enclaves are created. Based on the hardware access control mechanism, any unauthorized access to the enclave memory is deemed as a page-fault. SGX allows the code inside the enclave to directly access the memory outside EPC, however, such memory accesses are controlled by the operating system (OS) memory management system[31]. Enclaves are also unable to access other enclaves' contents. SGX supports multi-threading within enclaves to accelerate the parallel execution of trusted applications.

Intel SGX provides *remote attestation* policy that evaluates the enclave identity, integrity of the code inside of it, and guarantees the authenticity of the Intel processor. Remote Attestation serves as a verification protocol for the service provider to evaluate the health of enclave(s) created at a remote location[53,54,27]. SGX also offers *enclave sealing* mechanism that encrypts the enclave to be safely stored in an untrusted storage medium, such as a hard drive, for later use. It also helps the enclave while retrieving data and secrets from the sealed file without performing a new remote attestation[54,52].

While Intel SGX fulfills most of the objectives set forth by the GlobalPlatform TEE, such as secure storage and isolated execution, it does not provide a native trusted User Interface (UI) or network communication directly from the enclaves. Specifically, the Intel SGX technology focuses on the processor and memory communication aspects. However, SGX fails to provide any feature to facilitate secure communication with the I/O devices. Hence, it is necessary to integrate SGX with other solutions to enable secure communications, for instance, hypervisor-based trusted path architectures with respect to I/O devices[31,55].

	Intel SGX	**AMD SEV**	**ARM TrustZone**
Processor Architecture	x86-64	x86-64	ARM
Secure Storage	Yes	No	No
Remote Attestation	Yes	Yes	No
Memory Isolation	Yes	Yes	Yes
Memory Size Limit	Up to 128 MB EPC	Up to available RAM	3–5 MB
Trusted I/O	No	Yes	Yes
Operation Level	Ring 3	Ring 0	Ring -2
Compatibility	Windows	Linux-based VMs and hypervisors	Android, Linux
SDK	Provided	Not required	Provided
Memory Integrity Protection	Yes	No	No
Multithreading	Yes	Yes	No
Applications	Simple and security-sensitive	Complex and legacy	Lightweight

TABLE 2 Comparing properties of the TEE hardware technologies: Intel SGX, AMD SEV, and ARM TrustZone

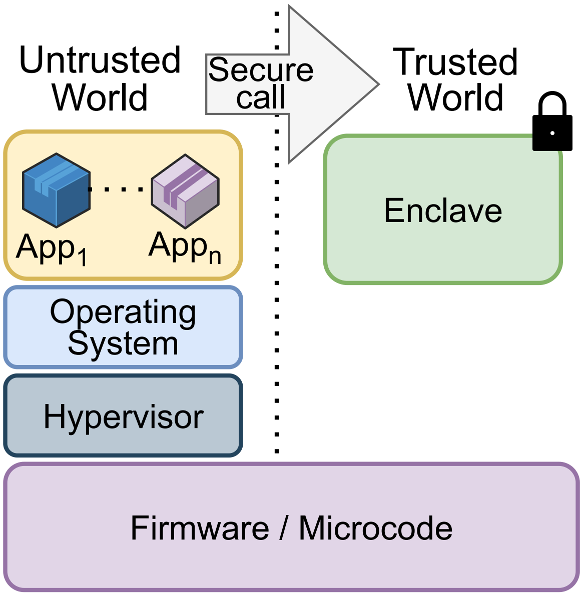


FIGURE 4 Illustration of secure isolation provided by Intel SGX. SGX consists of enclaves that offer a trusted world where sensitive code and data are isolated from the rest of the system to maintain integrity and confidentiality.

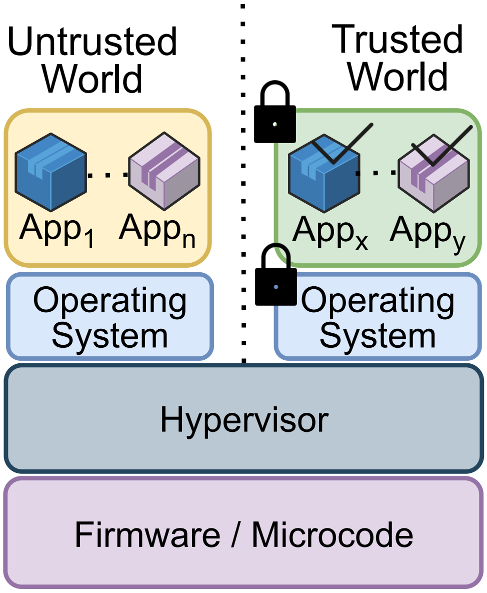


FIGURE 5 High-level architecture of ARM TrustZone. TrustZone provides secure world that facilitates isolated execution. It has access to all hardware resources and can interact with Untrusted/Normal world without tempering privacy.

4.1.2 | ARM TrustZone

ARM TrustZone is another secure space-based TEE hardware architecture that extends the security aspect to the entire system design, allowing any part of the system to be protected. TrustZone technology provides an infrastructure that allows chip designers to choose a range of components that can assist with specific functions inside of a secure environment.

ARM TrustZone architecture can be isolated in two logical states: a secure/trusted world and a normal (*i.e.*, insecure) world 5. The mechanism that controls information flow between the two states is called *monitor*. Communication with the secure world occurs from the insecure world through the Secure Monitor Call (SMC) instruction. When an SMC instruction is invoked from the normal world, the CPU core performs a context switch to the secure world and freezes its normal-world execution. All other CPU cores of a multi-core system can remain in normal-world mode uninterruptedly. To facilitate secure world processing, TrustZone can separate physical memory into two partitions, with one partition being exclusively accessible by the secure world. This isolation is enforced by the memory controller providing access control for memory regions based on the current state. While the normal world cannot access memory allocated to the secure world, the secure world can access normal-world memory. Therefore, when an application executes in a secure world, TrustZone can isolate parts of the memory for its use via preventing accessing these locations by the applications executing in the real world[28].

ARM technology is predominantly used in single-purpose systems, such as IoT and edge computing, where the chip is specific to the target market (*e.g.,* in smartphones, smartglasses, *etc.*). Hence, an ARM-based device only has one TrustZone with the processor. This is unlike Intel SGX that take advantage of multiple enclaves.

4.1.3 | Memory Encryption-based TEE: AMD SEV

AMD Secure Encrypted Virtualization (SEV) technology is considered to be the most recent hardware assisted TEE that encrypts and protects system memory. The technology has brought *AES-128* bit encryption engine inside the System on Chip (SoC) that encrypts and decrypts the data upon leaving or entering the SoC.

As shown in Figure 6, SEV isolates virtualized environments (*e.g.,* containers and VMs) from the underlying platforms (*e.g.,* hypervisor) through memory encryption. Although hypervisors are commonly used as trusted components in the virtualization security model, they cannot guarantee the security of confidential workloads. For instance, to preserve data/workload confidentiality in the cloud, users need to secure their VM-based workloads from the cloud provider (administrator). This leads to the necessity of hardware level VM isolation which is what SEV can fulfill. SEV allows a single VM to be assigned a unique AES encryption key to encrypt the data they use. Consequently, even if the hypervisor tries to read the memory inside the guest OS, it can only fetch the encrypted bytes. AES encryption provides increased confidentiality protection of memory. An attacker without proper knowledge of the encryption key cannot decipher VM data. Note that SEVs memory encryption keys are generated from a hardware random number generator and is stored in dedicated hardware registers that cannot be directly read by systems. In addition, the hardware is designed in such a way that the same plain-text is encrypted differently in different memory locations[32].

AMD SEV does not require any modifications to the user application software and the memory encryption is transparent to the application executing in the SEV-protected VM. SEV uses the AMD Memory Encryption Engine which is capable of working with different encryption keys for encrypting and decrypting different VM memory spaces on the same platform.

In SEV, a unique encryption key is associated with each guest VM. Upon code and data arrival to the SoC, SEV tags all of the code and data associated with the guest VM in the cache and limits the access only to the tags owner VM. Upon data leaving the SoC, the VM encryption key is identified by the tag value and data is encrypted with the VM key. Moreover, initializing a SEV-protected VM requires direct interaction with the AMD secure processor. The AMD secure processor provides a set of APIs for provisioning and managing the platform in the cloud. The hypervisors SEV driver can invoke these APIs.

In the AMD SEV architecture, the guest owner manages the guest secrets and generates the policies for VM migration or debugging. The Diffie-Hellman key exchange protocol is used between the guest owner and the AMD secure processor to open a secure channel between the guest owner and the AMD secure processor. The guest owner is enabled to authenticate the secure processor and exchange information to set up the protected VM. Also, the SEV architecture defines the shared page (unencrypted) and the private page (encrypted) that can be set for each protected guest VM. The C-bit is set to identify the private pages by the guest OS.

4.1.4 | Other TEE Implementations

While Intel SGX, AMD SEV, and ARM TrustZone have gained significant attention in the confidential computing realm, several other TEE implementations also hold considerable promise to realize confidential computing. Here are some notable mentions:

RISC-V: RISC-V[56] is an open standard instruction set architecture that is growing in popularity for its flexibility and extensibility. Several initiatives (*e.g.,* [57,58,59]) are underway to develop TEE implementations based on the RISC-V architecture, aiming to provide hardware-level security guarantees while preserving the open and customizable nature of RISC-V.

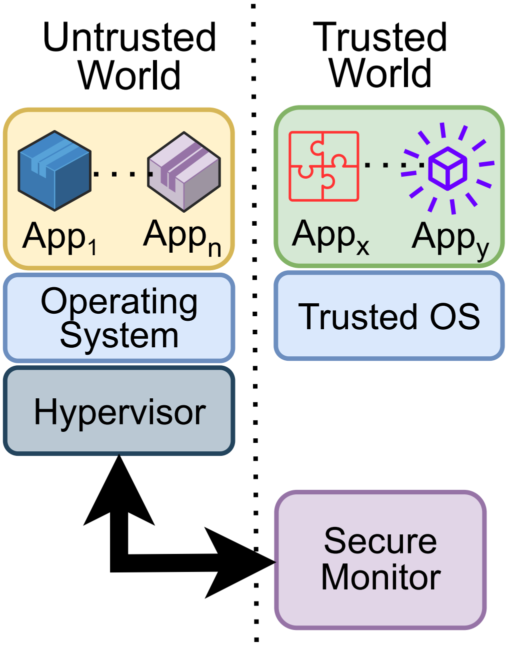


FIGURE 6 High-level architecture of AMD SEV. SEV leverages memory encryption to isolate virtualized environments (*e.g.,* containers and VMs) for confidential processing from the underlying platforms.

Open Enclave SDK: Microsoft's Open Enclave SDK[60] is a library that allows developers to create applications that leverage TEEs across different hardware platforms. It abstracts the specifics of the underlying TEE technology, making it easier for developers to create confidential computing solutions without the need to understand the complexities of each hardware solution.

Keystone: Keystone[61] is an open-source project that aims at building a secure and customizable enclave based on RISC-V the architecture. It strives to provide an open, flexible, and extensible TEE that is verifiable by the community.

4.1.5 | Advancements in the TEE Adoption

The integration and adoption of Trusted Execution Environments (TEEs) have made significant strides over recent years, spanning various applications and addressing multiple challenges. The research works undertaken by Zhang *et al.*[62] and Ning *et al.*[63] survey existing TEEs, highlighting their utility and potential roadblocks. Furthermore, the evolution of TEEs has been driven by the need for robust security measures in modern defense systems. As exemplified by MemSentry[64], TEEs are being used to leverage hardware features to improve overall system security. The implementation of TEEs in this regard has revolutionized defense system security, making TEEs an integral part of contemporary defense technologies.

Adoption of TEEs in data analytics and debugging processes is also gaining momentum. This is illustrated by the work of Schuster *et al.*[65], where Intel SGX was utilized to secure data analytics. Similarly, the work of Ning *et al.*[66] leverages the ARM TrustZone technology to enhance transparency in debugging and tracing processes.

In light of these advancements, the practical adoption of TEEs has extended beyond theoretical applications, becoming a crucial tool in mobile, wearable systems, data analytics, defense technologies, and debugging processes. Continued work in this area is expected to drive further advancements, cementing the role of TEEs in ensuring secure and efficient solutions in distributed systems and, particularly, within the context of IoT and edge-to-cloud continuum.

4.2 | Hardware Modules for Confidential Computing

4.2.1 | Trusted Platform Module (TPM)

TPM is a dedicated secure co-processor chip that is designed to carry out cryptographic operations (*i.e.,* digital rights managements) via ensuring safer computing across multiple environments. TPMs are placed on the motherboards to provide trusted computing capabilities to the system. The chip includes a set of security mechanisms to construct it as temper-resistant, so that malicious programs unable to tamper the security codes of the TPM. Each TPM contains an RSA key pair, called the the Endorsement Key (EK) that is generated during the manufacturing process of the TPM chip. Each chip owns a unique and identifiable EK that is managed within the chip and is inaccessible by any software. When a system user/administrator takes ownership of the system, The Storage Root Key (SRK) is created based on EK and the owner-specified password.

Besides EK, TPMs generate another key pair, named Attestation Identity Keys (AIK), that protect the device against unauthorized firmware or software manipulation via hashing critical sections of them before execution. When the system attempts to connect to the network, the hashes are sent to an authentication server for verification purpose. If any hashed components is compromised since the last execution, the network access authentication fails.

The key advantages of adopting TPM technology for confidential computing are twofold:

1. Generating, storing, and controlling the use of cryptographic keys.
2. Creating an unforgeable hash key summary of the hardware and software configuration, digital right management, and software licensing. Via examining the hash key, a third party can verify the integrity of the software.

The initial version of TPM 1.2 was released in 2005 and it has been updated to TPM 2.0 in 2019 with a wider range of security features. The advancements of TPM 2.0 over the previous version are as follows:

- *Making use of newer algorithms*: TPM 1.2 leverages SHA-1 hashing algorithm that raises security concerns. Hence, SHA-256 algorithm was adopted and TPM 2.0 now supports a variety of newer algorithms that improve the performance of drive signing (*i.e.,* signing device drivers) and key generation.
- *Supporting more data types*: TPM 1.2 only supports unstructured data in NVRAM, whereas, TPM 2.0 supports unstructured data, Counter, Bitmap, Extend, and PIN pass/fail.
- *Supporting more hierarchical structures*: TPM 1.2 only has a storage hierarchy, whereas, TPM 2.0 supports three hierarchical structures that are: (A) Platform Hierarchy (PH)[67] that represents the root of trust for the platform and is typically controlled by the platform manufacturer. The PH is responsible for managing platform-specific operations, such as initializing the TPM and controlling its critical functions. (B) Storage Hierarchy (SH) that is in charge of managing keys and authorizations related to the storage and retrieval of sensitive data within the TPM. (C) Endorsement Hierarchy (EH) which is mainly used to establish the identity and authenticity of the TPM via creating certificates, signing them using the EK, and participate in the attestation protocols to prove the platform's trustworthiness.

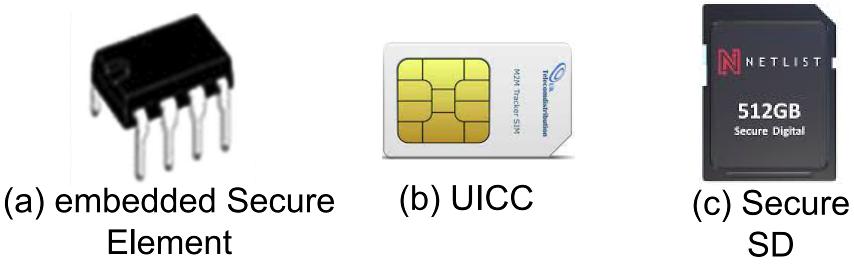
4.2.2 | Various Implementations of TPM

- *A discrete TPM* is implemented as a separate function or feature chip, with the required computing resources that are contained within the discrete physical chip package. A discrete TPM has full control of dedicated internal resources such as volatile and nonvolatile memory, cryptographic logic, *etc.* and it can only access and use these resources. Hence, they are considered the most secure type of TPM. Intel Trusted eXecution Technology (TXT)[68] and Platform Trust Technology (PTT)[69] are some implementations of discrete TPM 2.0.
- *Integrated TPMs* are implemented as a dedicated hardware that are integrated into embedded into the hardware of a computing device (*e.g.,* laptop or server motherboard). However, they are logically separate from other components of the system.
- *A firmware-based TPM (fTPM)*[70] is a software implementation of a TPM that resides in the firmware of a computing device. It emulates the behavior and features of a hardware-based TPM using firmware code. It can also operate using the resources of a multi-feature compute device such as SoC CPU in the context of a TEE. An fTPM does not have its own dedicated storage, thus, it relies on the operating system and relevant platform services to get the storage access right.

- *Hypervisor TPMs*[71] are virtual TPMs (vTPMs) provided by hypervisors, hence, they are dependent on the hypervisors. The vTPMs run in an isolated execution environment that is hidden from the other software applications executing inside VMs. The aim of such special execution scheme is to secure their code from the software in the VMs. The vTPMs offer a security level comparable to fTPMs. Google Cloud Platform has implemented and utilizes vTPMs in its offerings[72].

4.2.3 | Secure Element (SE): Hardware Module for Confidential Computing

A Secure Element is a tamper-resistant platform that is capable of hosting programs (*e.g.,* code and script) and confidential data, such as cryptographic keys, securely according to the security processes set forth by its owner. The concept of a secure element came to light in the mid of 1970s in the form of a smart card which was based on a one-chip. A secure micro-controller running a secure ultra-light operating system, which was restricted to execute only one application for a long time[15].



(a) embedded Secure Element

(b) UICC

(c) Secure SD

FIGURE 7 Modern forms of Secure Elements (SEs). Part (a), Embedded Secure ELement (eSE), is used in the smartphone hardware. Part (b), UICC, is for secure identification and authentication of users in cellular (mobile phone) networks. Part (c), Secure SD card, provides additional security layer via enabling the secure data storage and processing within the card.

In contrast to TPMs, SEs are able to execute secure code and are not restricted to perform only cryptographic operations. Smart cards were the only type of SE used with different types of connectivity, such as USB dongles and contactless smart cards over a long time period. Recent invention of Near Field Communication (NFC)[73] introduces three new forms of SE that are shown in Figure 7. These SEs are as follows:

- Embedded SE (eSE)[74] is a smart card micro-controller integrated into the NFC chip or directly used in the hardware of the mobile phone. It provides a secure environment for the management, storage, and processing of confidential and cryptographic data. It is typically integrated into the Near Field Communication (NFC) chip of mobile devices or directly into their hardware, forming an essential component of confidential computing. The eSE protects sensitive data such as encryption keys, digital certificates, and user credentials. In applications such as mobile payments, access control, or authorization, the eSE performs secure transactions by using the stored data to authenticate interactions, thereby offering a secure and confidential computing environment.
- Universal Integrated Circuit Card (UICC)[75], commonly referred to as a SIM card, contains mobile subscriber identity data to facilitate secure user identification and authentication on mobile devices. It establishes security groundwork for smartphones by protecting user identity information, network authorization metadata, and security keys. The UICC enables secure network connection and confidential communication, effectively preserving data integrity and privacy.
- Secure SD card[76] is another type of smart card micro-controller that enables features such as secure data storage, hardware encryption, and tamper-resistant processing. These cards operate by utilizing hardware encryption algorithms to secure data stored on them. This ensures the confidentiality of the data, rendering it unreadable without the corresponding encryption key, even when the card is relocated to another device. As a result, Secure SD cards offer a robust solution for secure and portable storage of sensitive data in the realm of confidential computing.

5 | CONFIDENTIAL COMPUTING MIDDLEWARE ACROSS EDGE-TO-CLOUD

Middleware in any form of distributed system, including edge-to-cloud, is an abstraction layer that lies between the OS of each machine and the applications executing across the distributed system. The aim of the middleware is to hide system complexity and enable seamless sharing of resources between users and applications. A middleware-based distributed system is susceptible to attack, as anyone with the root access to the worker machines is able to inspect, modify, kill, or modify code or data executing on that machine. Hence, to establish confidential computing across edge-cloud, enabling trusted computing on the middleware is a necessity. Confidential Computing addresses this sort of attacks via maintaining the confidentiality during the execution of a software irrespective to the privilege levels of any individual who has access to the system. In this Section, we will discuss the scopes in the middleware where confidential computing is performed.

5.1 | Confidential Computing: Edge-to-Cloud Perspective

The cloud computing paradigm offers large scale managed sharing and interpolation among the dispersedly controlled resources. However, data security and privacy concerns are the crucial factors that have made many organizations reluctant to use cloud[34]. As example, the cloud infrastructures are often vulnerable to insider threats, such as former employees accesses, manipulating, or destroying a copy of confidential data (including on-site backups). Numerous recent data privacy violations[77] in the cloud environments have raised serious data privacy concerns. In one incident[78], more than 14 million Verizon customer accounts information were leaked from their cloud repository. In another incident[79], confidential information of over three billion Yahoo users were leaked. Consequently, the cloud beneficiaries (*i.e.,* individuals, organizations) with sensitive data are hesitant to fully embrace the cloud paradigm due to the privacy concerns. Users expect that all their processing and the communications in the cloud are trusted. In practice, a cloud provider explains the compliance and service level agreement (SLA) to beneficiaries, however, that could not "guarantee" technical enforcement or transparency such that the beneficiaries can safely run their sensitive workload on the cloud.

A trusted cloud infrastructure is expected to provide increased reliability, technical enforcement, and security assurances. Confidential cloud computing (CCC) meets these expectations via allowing the beneficiaries to specifically define the required hardware and software that have access to their workloads (*i.e.,* data and applications). Adopting confidential computing provider users with the full control over their workloads, software, and hardware systems. It prevents cloud-hosting infrastructures (*e.g.,* hypervisors[80]) access to their sensitive data. Establishing CCC depends on the cloud computing paradigm, namely *serverful* (elaborated in the next section) vs *serverless*[81], and the middleware used to offer the services in each paradigm. Moreover, it depends to the type of isolation (virtualization) that is offered in each paradigm[82].

Edge computing has emerged to fill the gap between client machines and remote cloud datacenters. Being deployed near to the user, edge systems have are able to handle data-driven and/or compute-intensive applications, such as augmented reality, video analytics, and ML. As we move from the cloud to user premises, communication latency decreases, but trustworthiness increases[7]. As the edge nodes are close to the end users premises, they are often deemed trustworthy and data privacy is ensured to match with the computing requirements. A large body of research (*e.g.,* [83,84,85,86,87,88]) have been conducted in the usage scenarios and performance of edge computing. However, the privacy and security of the edge computing are not much dealt with.

5.2 | Confidential Computing in the Serverful Cloud Paradigm

The serverful computing paradigm refers to the conventional form of resource provisioning in the cloud that is based on Bare Metal (BM) servers, VMs, or containers listening for requests on port 80.

5.2.1 | Bare Metal (BM)

BM resource provisioning describes an environment where physical dedicated servers are provisioned to customers. The important point is that BM servers do not use any form of virtualization and hypervisor[89]. Consequently, BM users have full control over the allocated servers, including its processing, storage, and networking subsystems[90]. This is not the case for virtualized (multi-tenant) servers running on a shared hardware. As such, BM users have the freedom of configuring any trusted OS environments and applications as well as installing hypervisors to create their own VMs to satisfy their requirements.

Deploying a trusted application on the BM server requires attention to the security of the underlying platform, including using a trusted hardware, patched OS, and controlling access permissions. At the hardware level, making use of a trusted execution environment (TEE) is a must and, for that purpose, cloud providers often rely on Intel SGX and AMD SEV to offer confidential computing in the execution of trusted applications[91]. Popular cloud service providers such as IBM, Alibaba, and Platform9 clouds have configured Intel SGX and offer it in their BM deployments. In[92], IBM provides complete documentation of provisioning a fully SGX-based BM server.

Figure 8 shows a high-level diagram representing the Intel SGX application setup in a BM server of the IBM cloud. According to this figure, a trusted application composed of generic and sensitive code/data, respectively, can make use of the generic and SGX cores of the Intel CPU. Such implementation facilitates confidential computing to the extent that even an attacker with root privilege to the BM instance cannot access or tamper the code, data, or the returned outputs. In addition, even the cloud provider cannot access or tamper with the code/data, despite their direct access to the hardware as well as root access to the host OS[91].

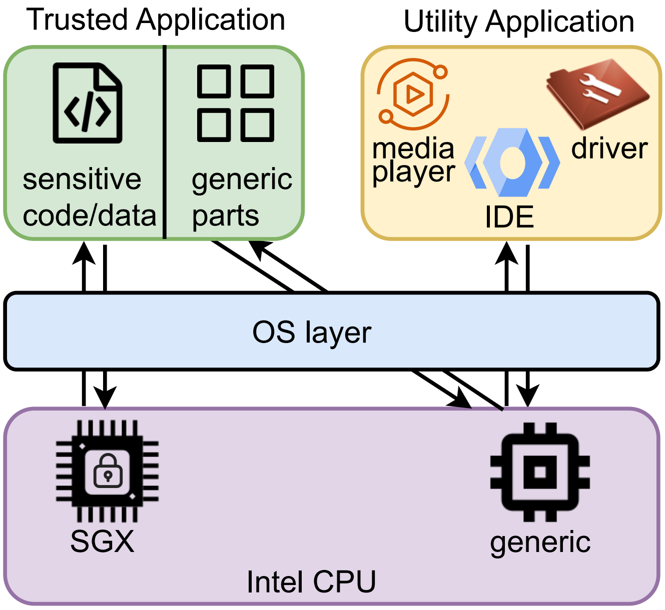


FIGURE 8 Trusted Application setup on top of the Intel SGX-based bare metal (BM) server

5.2.2 | Confidential Virtual Machines (VMs)

A VM provides an isolated and exclusive execution environment by running its own OS and functions separately from the other VMs, even when they run on the same physical host machine. VMs serve many use cases as they can be deployed across on-premises and cloud environments. Particularly, public cloud providers are using VMs to provide cost-efficient and flexible (with root access) virtual resources to multiple users at the same time (a.k.a. multi-tenancy). In this case, VMs are hosted on remote servers that are not under the control of the VMs' owners (users). Hence, the trustworthiness becomes a major challenge for cloud VMs.

For the purpose of confidential computing, *confidential VMs* have been proposed to run alongside of other standard VMs atop hypervisor. By definition, VMs inherently maintain a high degree of isolation from each other. In addition to this inherent isolation, a confidential VM is protected by hardware-based encryption keys that prevent a malicious VM manager to breach its confidentiality. Confidential VMs track the system record in the background for attestation purposes and to verify the system security. Although it is essential to deploy confidential VMs on top of TEEs for fast and easy adoption, it raises some challenges. For example, the VM administrator has full read/write control over the it, which is too coarse in many cases. Another concern is that the VM TCB is large. This because the VM image is far more than just a kernel and an application; it includes a large number of system services. In the worst case scenario, this configuration still proves to be more secure than running the software on-premises or on the existing cloud infrastructure.

Confidential VM via AMD SEV. AMD SEV is one of the most common TEE technologies that is used as the hardware to deploy confidential VM. To provide runtime protection, AMD's memory encryption engine encrypts the memory contents of the SEV-enabled confidential VMs using AES-128 encryption algorithm. An integrity-protected firmware deployed on top of a dedicated co-processor called Platform Security Processor (PSP) is responsible to generate the required encryption keys. The co-processor has its own memory and nonvolatile storage while having access to the system memory of the main processor. Moreover, the integrity-protected firmware provides APIs that can be used by the host hypervisor for encryption key managements on behalf of all SEV-enabled confidential VMs running on that system. The APIs also handle secure data transfer between the host hypervisor and the virtual memory of a guest confidential VM. It is noteworthy that AMD-SEV does not exploit any software, therefore, developers do not require any AMD APIs or libraries to make their applications compatible.

Nevertheless, because cloud are deemed as untrusted providers who can modify a VM while it is being deployed, the runtime protection does not "guarantee" the confidentiality of workloads executed on an SEV-enabled VM. As such, remote attestation feature can be used to establish trustworthiness. This feature can be used to verify an authentic SEV-enabled AMD platform configuration for the VMs on the cloud. At present, AMD-SEV is only available in AMD's EPYC series processors that are intended only for servers.

Confidential VM via Intel SGX. Confidential VMs are also configured on a special implementation of Intel SGX, called *vSGX*, that enables VMs to use Intel SGX technology if it is available on the hardware. To use vSGX, the bare metal hypervisor host (*e.g.,* ESXi) needs to be installed on an SGX-enabled CPU and vSGX functionality is enabled in the BIOS of the host machine. Despite AMD SEV, it is observed that Intel SGX-enabled VMs require considerable amount of effort to make the code compatible. Unlike AMD SEV, however, Intel SGX requires the programmers to use the Intel-SDK to specify which parts of the application will be executed on the trusted or untrusted subsystems.

SGX's isolated memory regions are ideal for small-TCB (Trusted Computing Base) services, even though using them to run confidential VMs is challenging. This is mainly because of two reasons: (i) VMs need large TCBs, because a VM image is significantly larger than just a kernel and an application. (ii) Lack of support for multiple address spaces and privileged and unprivileged mode separation. Provided these facts, *we can conclude that AMD SEV is more applicable for confidential VMs than SGX*.

5.2.3 | Confidential Containers

Container is a software package that contains all of the required components (such as file systems, libraries, environment variables, *etc.*) to execute applications in any environment without having side-effects on other applications on the same machine. As containers can deploy and execute applications in isolation that access a shared OS kernel, they are comparable to VMs. Therefore, containers are used as a replacement (or complement) of VMs where the allocation of hardware resources is carried out through containers.

Containers are lightweight and orchestrated to virtualize single applications. They create isolation boundaries at the application level, not at the server level. This isolation means that if anything goes wrong (*i.e.,* a process consuming too many resources, unexpected exceptions) in a particular container, it will only affect that container, not the entire VM or server. It also eliminates compatibility issues between containerized applications on the same operating system.

The containers that execute confidential workloads need to be isolated, secured (encrypted), and inaccessible so that it can be protected from misuse. That is why, the idea of *confidential container* has emerged to conceptualize running a container within a hardware TEE platform, thereby, ensuring protection against vulnerabilities at the guest OS, hypervisor, and host OS levels. In this case, because of the TEE involvement, a confidential container provides a set of features alongside of an existing container deployment to make it secured, encrypted, and more isolated, thereby, achieving a higher data security. Confidential containers can mitigate the limitation of confidential VMs via offering finer degree of control and ensuring faster and secure execution of containerized applications. It also allows running standard container images with no modification or recompilation in code within a TEE setup.

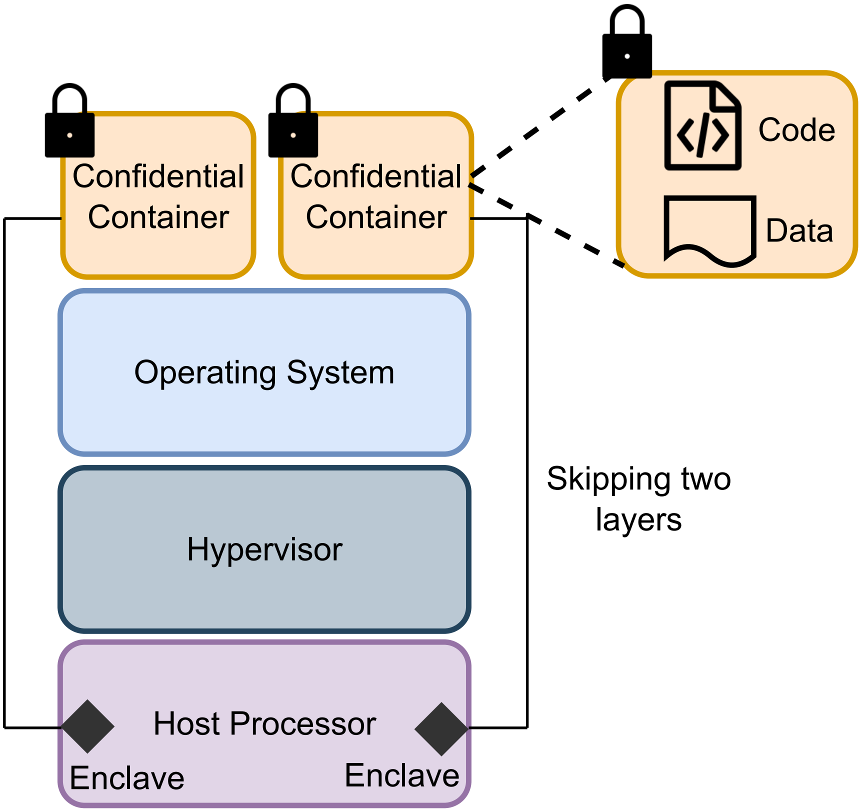


FIGURE 9 Layered view of the confidential container deployment in Microsoft Azure. Azure leverages SGX to offer direct execution to the host processor via bypassing the guest OS, host OS, or hypervisor from the trust boundary.

Kubernetes[93], a widely adopted portable and open-source platform, used for managing (a.k.a. orchestrating) the containerized workloads and services, thereby, mitigating the burden to rely on and trust the system administrators to securely launch containers. More specifically, upon launching a confidential container on a Kubernetes node that is capable of performing confidential computing, it creates a process-isolated and *sandboxed enclave*. By definition, *Enclaves are secured portions of the TEE's processor and memory*. The memory allocated by the enclave is process-specific, allowing for isolation across containers and protection for each container. Through container encryption, signing, and attestation, enclaves ensure code integrity and prevent malicious attacks that might attempt to tamper with the code inside the container.

Figure 9 illustrates deployment architecture of confidential containers in Microsoft Azure Kubernetes service. Azure leverages the Intel SGX processor, which allows user-level code from containers to allocate private memory regions to execute the code. The discrete execution model per container per node allows the hardware to run applications directly on the host processor and encode a dedicated block of memory within a single container. To launch existing docker containers, applications on confidential compute nodes require Intel SGX wrapper software to help containers execute with the special CPU instruction set. SGX creates a direct execution to the host processor via bypassing the guest OS, host OS, or hypervisor from the trust boundary. This step reduces the overall attack surface and vulnerabilities, while enabling process-level isolation within a single node.

5.3 | Confidential Computing in the Serverless Cloud Paradigm

Serverless computing is an execution model in which a cloud provider dynamically allocates—and charges the user for—only the compute resources and storage needed to execute a particular piece of code (a.k.a. function). Serverless computing paradigm enables developers to concentrate on the business logic by writing fine-grained and standalone functions with minimal overhead of the deployment, management, and scalability. Naturally, there are servers involved in the back-end, however, their provisioning and maintenance are entirely handled by the provider and is transparent from the user's perspective. The paradigm aims at mitigating the job of cloud programmers[94] and is considered as the next generation of cloud computing system. Consequently, hyperscaler clouds such as Amazon, Microsoft, and Google have introduced AWS Lambda[95], Azure Functions[96], and Google Cloud Functions,[97], respectively. According to the recent research[98,99], around 54% serverless applications contain only one function whereas 50% of them have less than one second of executing latency. Serverless paradigm can be defined as the aggregation of Function-as-a Service (FaaS) and Backend-as-a Service (BaaS)[100]. While FaaS focuses on the front-end development of functions, BaaS focuses on the transparent and isolated execution of the functions[94].

Alongside the regular executions, serverless computing is used to execute privacy-preserving workloads such as authentication, personalized chatbot, biometrics (*e.g.,* face/fingerprint recognition) processing. To protect the user privacy in the serverless paradigm from different security threats, including malicious cloud software and suspicious cloud insider, it is essential to enable confidential computing within this paradigm. Secure enclaves, offered by Intel SGX, are widely used as a trusted hardware component to offer a fully isolated execution environment for privacy-preserving serverless applications. However, the existing CC architectures are not well-suited for confidential processing, as they cause performance degradation up to 422.6%[99]. The majority of this overhead is causes by the enclave initialization: hardware enclave creation and attestation measurement generation during the startups and secret data transfer among the functions during their executions. Li *et al.*show that secret data transfer between functions can take up to 34.7% of the execution time[99]. Although recent studies[101,102,103] propose different software-based optimization techniques, the end-to-end latency to invoke a function is nowhere close to the satisfactory range. Li *et al.*conclude that the design technology of the current SGX is the reason for the high latency overhead. According to their paper, the existing SGX hardware architecture is not featured with memory sharing option across the enclave instances[99].

Although SGX enclaves ensure security, such excessive start-up latency (cold-start) is ill-suited for today's confidential serverless computing. To improvise enclave abstraction, Li *et al.*extend the existing SGX implementation with *region-wise plugin enclaves* that can be immutably mapped into different existing isolated enclaves to reuse attested common states across the functions[99]. They instrument the dynamic resizing policy of SGX to implement a hardware-enforced copy-on-write mechanism for maintaining consistency between the content and measurement of a plugin enclave. They show that the plugin enclave-based serverless can scale up to 10× enclave instances than existing SGX hardware while reducing $\approx$ 96% overhead of function startup latency.

In[104], Feng *et al.*enable a fast enclave creation without compromising the security guarantees. They propose the notion of Guarded Page Table (GPT) to enable memory isolation with page-level granularity and Mountable Merkle Tree (MMT). Together, these concepts can achieve on-demand abstraction and integrity protection. Upon applying these two concepts, they propose the notion of *shadow enclave* that supports fast enclave creation without compromising security guarantees. Their experimental evaluations indicate that the shadow enclave system is capable to scale to thousands of concurrent secure enclaves with proper resource utilization and reduce the start-up latency by three orders of magnitude[104].

5.4 | Confidential Computing on the IoT (Device) Tier

Unarguably, establishing confidential computing across edge-to-cloud entails enabling it on each tier contributing to the continuum. Despite differences in the scale of deployed resources on the edge, fog, and cloud, the confidential computing solutions we discussed in the previous two parts (*i.e.,* serverless and serverful paradigms) overlap across these tiers. This is because: (a) all these tiers predominantly operate based on some standard middleware solutions, such as Kubernetes[93] and OpenStack[105]; (b) similar isolation techniques (*e.g.,* VMs, containers, *etc.*) are employed across these tiers; and (c) servers of these tiers are configured with widely-used operating systems, which is often some flavor of Linux.

The IoT device tier, however, does not share the aforementioned three similarities with the other tiers. In practice, IoT devices are categorized as embedded systems[40] that often have their own custom firmware. Moreover, they are highly distributed and accessible that makes them more vulnerable. These qualities imply different type of threats and confidentiality solutions for the IoT tier. Accordingly, in this section, we concentrate on the traits of IoT devices and challenges and solutions of enabling confidential computing on them.

Ultimately, the comprehensive *IoT standardization* will be pivotal in fully realizing the potential of confidential computing at the IoT tier[106]. Currently, there are some fragmented efforts for IoT standardization (*e.g.,* ISO/IEC 30141[107] and TS 103645[108]) that prescribe issues like using a common vocabulary and reusable designs for IoT devices. However, there is yet to be a consensus on these standards and this discrepancy is the Achilles' heel for the confidential computing across the entire continuum. That is why, organizations are actively seeking for alternative security mitigation strategies.

First and foremost, organizations should consider zero-trust access (*ZTA*) to verify users' and devices before every application session. This assures that the users and devices meet the organization's policy to access that application, therefore, dramatically mitigates IoT-level risks.

Micro-segmentation[109] is deemed as another strategy with a significant potential to mitigate vulnerabilities caused by the distributed nature of the IoT tier. The concept of micro-segmentation strategically fragments the expansive IoT network into isolated 'micro-segments'. Each of these micro-segments serves as a secured endpoint and is strictly maintained by well-defined security policies. This structure allows for grouping of the IoTs based on their distinct roles within the network and the sensitivity level of the data they manipulate. This is achieved via localizing the potential threats to individual segments, thereby, mitigating the risk of widespread network compromise and enhancement of the confidential computing across the continuum.

The dynamic nature of IoT networks, with devices of varied security capabilities continuously joining and leaving, presents another level of complexity. However, with security policy identification for each micro-segment, localized threat detection and response mechanisms, and efficient application partitioning across heterogeneous TEEs, this complexity can be handled.

It is noteworthy that, in addition to these mitigation strategies, basic solutions (such as connection encryption) can still help and are necessary to mitigate the risks of the IoT tier.

5.5 | Accuracy vs Performance Trade-offs in Confidential Computing Solutions

The selection of confidential computing-based solutions and their corresponding platforms plays a crucial role in determining the overall system performance and accuracy of the computational tasks. This trade-off is particularly evident as more rigorous security measures often require additional computational overhead, which can adversely affect the performance and sometimes the accuracy of the system.

- **Impact on Performance:** Confidential computing environments, especially those utilizing TEEs (*e.g.,* Intel SGX, ARM TrustZone), often experience performance degradation. This is primarily due to the additional steps involved in encrypting and decrypting data, managing secure enclaves, and the potential for increased page faults due to smaller memory spaces in secure enclaves. For instance, the overhead associated with context switching and memory copying in and out of TEEs can significantly slow down application response times.
- **Impact on Accuracy:** In machine learning applications, particularly those deployed in edge computing scenarios, the partitioning of data and computation tasks to enhance confidentiality can lead to reduced model accuracy. This reduction occurs because critical features may be segmented and processed on different nodes, potentially leading to incomplete learning or insufficient feature representation. Moreover, latency introduced during data transfer between partitions can delay real-time analytics, affecting the timeliness and relevance of the model outputs.
- **Balancing the Trade-offs:** Optimizing the trade-off between accuracy and performance involves selecting the right level of confidentiality and the appropriate computational strategies. Techniques such as model quantization and pruning can be employed to reduce the computational burden on TEEs but often, at the expense of slight decreases in model accuracy[110]. In addition, advanced data encoding methods can minimize the computational overhead associated with encryption while maintaining the integrity and confidentiality of the data.
- **Recent Advances and Comparisons:** Recent advancements in lightweight cryptographic protocols and more efficient secure enclave management techniques have begun to mitigate these trade-offs. Studies comparing the performance of applications running in traditional vs. enclave-based environments provide insights into the practical impacts of these technologies. For example, enhancements in enclave paging systems and the introduction of more granular control over data access and processing have shown to lessen the performance penalties while safeguarding data confidentiality.

Prior Studies	Hardware	Computing Paradigm	SDK	DL Library	Training	Protection Objective			Contribution
						Data Privacy	Model Confidentiality	Training Integrity	
Ohrimenko *et al.* [112] (2016)	SGX	Cloud	SGX SDK	Fast CNN	Y	Y	N	N	- Data-oblivious ML for classification and clustering
Hunt *et al.* [113] (2018)	SGX	Cloud	SGX SDK	Theano	Y	Y	Y	N	- ML-as-a-service implementation on multi enclaves
Lee *et al.* [114] (2019)	SGX	Edge-Cloud	SGX SDK	Caffe	N	Y	Y	N/A	- On demand loading - Channel partitioning
Mo *et al.* [115] (2020)	TrustZone	Edge	OP TEE	Darknet	Y	N	Y	N	- Privacy measurement - Layer-wise partitioning
Liu *et al.* [116] (2021)	TrustZone	Cloud	OP-TEE	-	N	N	Y	N/A	- Weights & Feature-map partition
Zhang *et al.* [117] (2021)	SGX	Cloud	SCONE	Tensorflow	Y	Y	Y	N	- Multi enclaves for training
Mo *et al.* [118] (2021)	SGX + TrustZone	Edge-Cloud	OP-TEE	Darknet	Y	Y	Y	Y	- Layer-wise training - Privacy measure
Sander *et al.* [119] (2023)	SGX	Cloud	SGX SDK	ONNX	N	Y	Y	N/A	- Distributed inference using garbled circuits

TABLE 3 Prior studies that leverage Confidential Computing to facilitate confidential machine learning (ConfML)

6 | TRUSTED APPLICATION DEVELOPMENT ACROSS EDGE-TO-CLOUD

6.1 | Confidential Machine Learning (ConfML)

Machine Learning (ML) can be broadly defined as methods to automate pattern learning from massive datasets to the extent that data analysts no longer need to manually identify the hidden characteristics and correlations of the data. Although ML-based applications are getting wide adoption and are envisaged to revolutionize our world, their algorithms do not ensure confidentiality of the sensitive data/workloads.

To address the data exposure risk and lack of confidentiality, enabling confidential computingfor the ML workloads is crucial. Recent advancements in TEEs in both high-end and low-end mobile devices make them a prime contender for achieving confidentiality and integrity in ML. Towards that, Confidential Machine Learning, a.k.a. *ConfML* protocol, has been proposed where the data owners must adhere to when from when they share training data with an ML service [111]. This protocol protects the privacy of training data during the training process.

Conventionally, encryption can safeguard the confidentiality of both data at rest and data in transit. Using encryption for ML, however, requires decrypting the data before the start of training which implies data vulnerability until the end of training. ConfML eliminates this vulnerability via protecting the privacy of training data during the training process [111]. The ConfML protocol consists of two phases that bookend the training process: *(i)* The data owner encrypts the training data files with a secret key before submitting them to the ML service. The secret-key is not accessible to the ML service. *(ii)* After acquiring the network-trained-on-scrambled-data from the ML service, the data owner modifies the network to behave as if it were trained on the original (unscrambled) data using the secret-key from step (i). These two procedures ensure that the ML service never has access to the original data, while the data owners obtain the necessary networks.

6.1.1 | Existing Solutions for ConfML

Prior literature has focused on attaining ML using TEEs, with some aiming to overcome the aforementioned obstacles. In Table 3, we summarize and compare these past studies from various aspects.

The simplest approach is to install the whole ML training/inference process within TEEs. In such a scenario, the ML task's maximal capacity is severely constrained by the TEE's space and compute limits. Then, the aim is to optimize the efficiency of each "bit" of TEEs' secure memory for ML computation, while striking a trade-off between the number of layers and the number of neurons in the neural network model. We outline some applicable strategies below:

1. Employing inference rather than training. Training process is resource intensive, to carry out the backward propagation (*e.g.,* memory used to save model gradients and intermediate activations) [110].

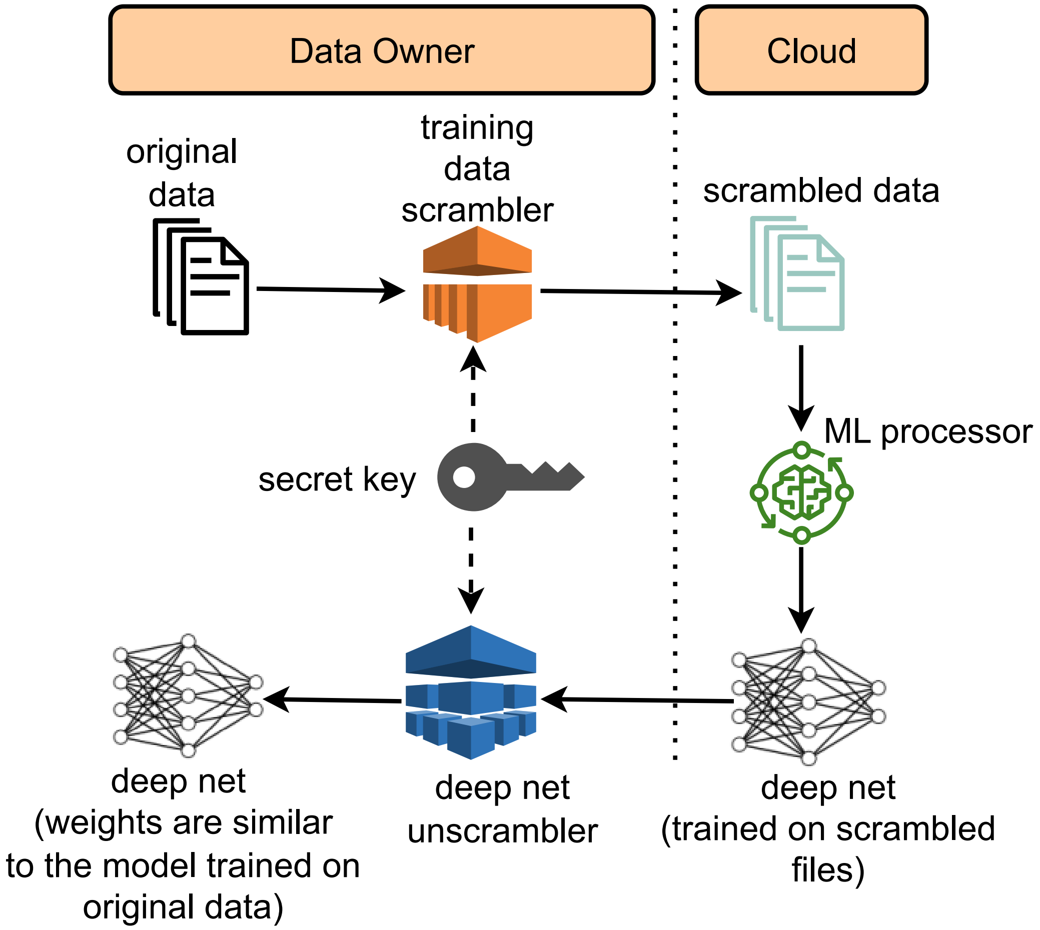


FIGURE 10 Workflow of confidential Machine Learning (ConfML). The workflow consists of two phases of the training process: Data owner and Cloud site. The data owner encrypts the training data with a private key and sends it to the cloud-resided ML service. The trained model is then converted by the owner using this key to function as though trained on original data.

2. Selecting a modest batch size. A high batch size results in a substantial memory footprint, as each sample in the batch generates its own activations for all the neural network model layers.
3. Striking a balance between the feature extractor (such as convolutional layers) and the classifier (*e.g.*, fully connected layers)[120]. In fact, a correctly constructed feature extractor can reduce the feature dimension while still capturing critical information, enabling a compact classifier to perform well.

6.2 | Secure Multi-party Computations

Secure multi-party computation (SMPC) has emerged as a powerful tool for privacy-preserving machine learning (ML) that enables multiple parties to collaboratively compute a function over their inputs, while keeping the inputs private. The foundation of SMPC is: by distributing the computation across different parties, one can limit the risk of privacy breaches. Each party holds a piece of the overall data, and computations are performed in such a way that no single party can reconstruct the entire dataset. This is particularly crucial in ML, where data may contain sensitive personal information. In recent years, a variety of protocols and frameworks have been developed to facilitate SMPC in ML applications. Some notable examples and strategies include:

Homomorphic Encryption (HE)-based Methods: HE allows computations to be performed on encrypted data, ensuring that the raw data remains secure throughout the process. These methods are suitable for applications where data privacy is paramount, and they have been successfully applied in ML models like linear regression and decision trees.[121,122]

Secret Sharing Schemes: By distributing secret shares of data among parties, ML models can be trained without any single entity ever having access to the complete data. This approach is often combined with techniques like differential privacy to further enhance security. Garbled Circuits: Originally used for boolean circuit evaluation, garbled circuits have been adapted for ML tasks. They allow parties to jointly evaluate neural networks or other complex models without revealing individual inputs.

Functional Encryption (FE): FE goes a step further by enabling the computation of specific functions on encrypted data and returning only the result. This restricts the information exposure even more tightly than HE.[123]

Decentralized ML via Blockchain: Leveraging blockchain technology, decentralized ML models can be trained and shared across a secure, distributed network, protecting the integrity of the data and the model.

6.3 | Application Architecture and Partitioning

Current software systems either follow the monolithic or the micro-service-based architecture. In the former, the entire application is one tightly-coupled entity, whereas, in the latter, one application is composed of multiple loosely-coupled micro-services forming a workflow (Directed Acyclic Graph–DAG) together. While there is an extensive debate on the pros and cons of each architectural approach in terms of performance (*e.g.,* [124]), there has been relatively less discussion on the impact of such architectures on confidential computing. While monolithic applications are inflexible and often do not provide much room to boost confidential computing, the loosely-coupled and independently-developed properties of micro-service-based applications are instrumental in achieving confidential computing.

Several research works have been undertaken towards this direction. Wu *et al.*propose a dynamic partitioning approach that optimizes the distribution of tasks between mobile devices and cloud or edge servers under variable network conditions and hardware capabilities[125]. This methodology underscores the importance of adaptability in partitioning strategies, especially in environments where bandwidth and computational resources are inconsistent. Xu *et al.*propose a probabilistic method to manage service migration for maintaining service continuity and quality in mobile scenarios[126]. This method is particularly relevant when services need to be dynamically relocated to the edge servers closer to the user's current location, thereby, reducing latency and response times while considering user mobility. Likewise, Zhou *et al.*introduce CNN-leveraged application partitioning to accelerate IoT task executions[127]. Several recent survey studies[128,129,130,131] highlight a critical aspect of application partitioning and migration: the dynamic and probabilistic nature of decision-making in response to real-time environmental changes.

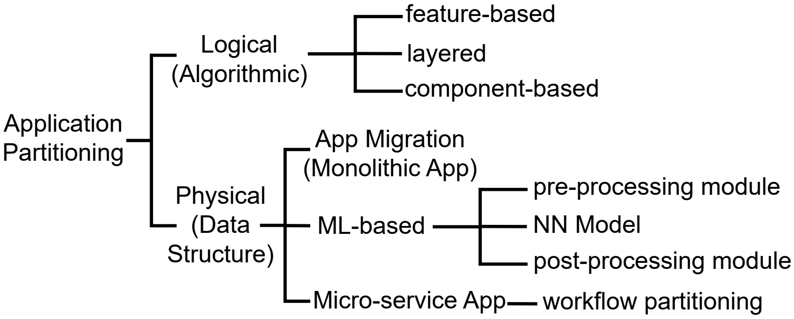


FIGURE 11 The taxonomy of application partitioning with respect to different software architectures (microservice vs monolithic) and at various granularity levels

The key to confidential computing at the software level is *the ability to partition the sensitive and non-sensitive parts of the application and allocate them separately*. Nevertheless, as shown in Figure 11, the application partitioning itself can occur at different granularity levels.

According to the taxonomy, the application logic (*i.e.,* algorithm) can be written such that the sensitive parts are executed on the trusted machine/TEE and the non-sensitive parts are executed on the normal (non-trusted) machines. For instance, SAED[36], a tool for secure semantic search of encrypted cloud data, realizes confidential computing via partitioning the search logic/algorithm. The intelligent part that captures semantics of the user's search query is performed on-premise (on a trusted edge server). After that, the augmented search query set is encrypted and outsourced to (untrusted) cloud to perform massive "pattern matching" on the encrypted cloud-stored dataset. The logical partitioning strategies can be further divided into three subcategories: feature-based, layered, and component-based partitioning.

Feature-based Partitioning involves fragmenting the application based on the features (services) to the user. For sensitive operations (*e.g.,* payment processing, personal data management), the application logic is designed to execute within the TEE. Non-sensitive features (*e.g.,* geographical data processing) can run outside the TEE. This selective execution ensures that only the parts of the application that handle confidential information are subject to the security overhead of TEEs.

Many applications are structured based on the *layered* design pattern; Each layer can be examined for its sensitivity and security requirements. For instance, the data access layer dealing with database operations could be considered sensitive, thus, operate within the TEE. In contrast, the presentation layer, which handles user interface components, could remain outside the TEE. SAED[36], a smart and secure semantic search solution across edge-cloud, is another example where the layered partitioning approach is instrumental to have the best of both worlds—intelligence and privacy. In SAED, the intelligence layer is performed on on-premises edge, whereas, the large-scale encrypted pattern matching occurs in another layer on the cloud. The layered partitioning not only simplifies the development process, but also allows for better resource allocation by only using TEEs where it is absolutely vital.

In addition to the layered approach, there are other software engineering design patterns where the application logic is decomposed into independent *components/modules* that each one has a specific responsibility. Model-View-Controller (MVC)[132], Pipe-and-Filter[133], and Repository[134] are popular representatives of such design patters. In all these cases, sensitive components that handle encryption, authentication, or other security-critical functions can be assigned to operate within the TEE. Other components, such as logging or view, can function outside the TEE to maintain the system efficiency. Component-based partitioning can be adjusted to operate at a high granularity level, thereby, contributing to an application's overall maintainability and scalability.

While the logical partitioning of the application entails developers involvement in confidential computing, there are physical partitioning approaches (see Figure 11) that leverage the *application's data structures* and/or *architecture* to realize confidential computing in a more transparent fashion from the developer's perspective. In this category, confidential computing can happen in three levels, namely no-partitioning; workflow-level partitioning; and ML-level partitioning. In the rest of this section, we elaborate on these levels.

6.3.1 | No-partitioning for Monolithic Applications

This is particularly applicable to legacy monolithic applications or those that have to run uninterruptedly, such as for monitoring industrial operations. For instance, "measurement while drilling"[135] is an application that has to continuously run to avoid missing any event. Realizing confidential computing of such applications can be performed via *migration* of the application to the trusted machine. The migration itself can occur in an online (live)[136,137] or offline ways.

6.3.2 | Workflow-Level (Coarse-Grained) Partitioning

This can realize confidential computing for applications whose architecture is based on a workflow of micro-services. For instance, fire extinguishing[138] that includes micro-services for video pre-processing, feature extraction, fire detection, and alert generation micro-services. In this case, the challenge is *efficient partitioning* of micro-services (*i.e.,* tasks) of such workflows across the underlying system (*e.g.,* edge-to-cloud) such that the sensitive micro-services are executed on the trusted machines/tiers and the rest can be done on normal machines/tiers. The other challenge in this type of partitioning is how to partition the micro-service DAG so that the application can still meet its quality of service (QoS) constraints (*e.g.,* deadline)[139]. Such partitioning can be accomplished by the underlying middleware, however, sometimes the developer or solution architects must be aware of the workflow topology.

6.3.3 | ML-Level (Fine-Grained) Partitioning

This approach particularly applies to ML applications that function based on neural network models that are prohibitively large for migration and/or process sensitive data. In fact, many ML-based applications are considered location-dependent, because

(i) they are tightly coupled to the sensor input data (*e.g.,* captured images from a camera); (ii) there are privacy concerns in outsourcing the sensor data; and (iii) they perform inference based on large neural network (NN) models whose migration imposes a prohibitively large overhead.

For such applications, approaches based on distributed inference[140] can be achieved that keep the source data (*e.g.,* captured images) and perform the pre-processing step on the source (trusted) machine, and then the neural network processing can be performed on a normal (potentially untrusted) machine[80]. It is even possible to vertically partition the NN model and perform some layers of it on the trusted machine and process the rest of it on the untrusted one[141]. In this manner, as shown in Figure 12, the layers of an NN model are vertically partitioned and form sub-models that can be assigned to untrusted machines. The vertical model splitting is known to impose a lower data transfer overhead than an alternative method, known as horizontal splitting[142]. More importantly, in vertical splitting, only the intermediate feature vector has to be transferred across fogs which has two benefits: (A) reducing the amount of data-transfer overhead; and (B) preserving the sensitive data at the trusted machine.

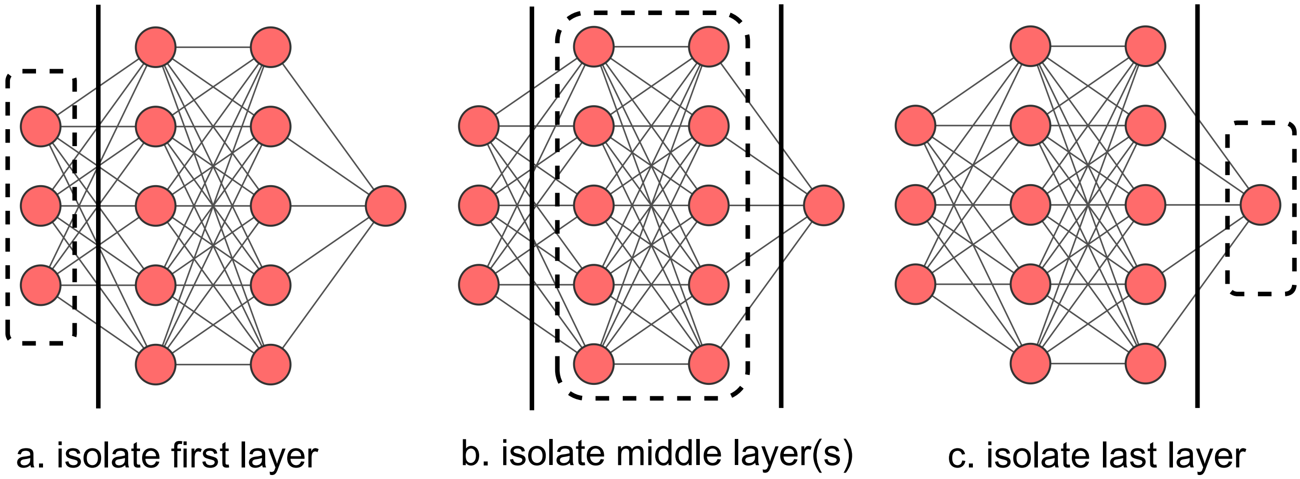


FIGURE 12 Partitioning neural network model of the ML applications and executing it with TEE.

6.4 | Application Data Encryption

Application Data Encryption is a vital security mechanism in trusted computing. As applications often handle sensitive user data, it is of paramount importance to secure this data not only when it is being processed, but also while it is at-rest or is in-transit. At-rest data protection involves encoding or transforming the data before it is written to the storage, thereby, making it unreadable to unauthorized users. Even if attackers manage to breach the storage system, the encrypted data remains unintelligible without the correct decryption key, such as those provided by AES and RSA methods. In-transit application data encryption safeguards sensitive data from being intercepted and deciphered during the transmission (*e.g.,* via SSL, TLS, and HTTPS protocols).

Similarly, for in-process protection, application data encryption is critical. Upon loading the data into the secure enclave for processing, it is decrypted, and upon completion, the results are encrypted before exiting the enclave. The enclave's isolated execution and the secure storage ensure that encryption keys are safe from exposure, thereby, providing a secure environment for application data processing. Furthermore, hardware-assisted security features embedded in TEEs, such as SGX or TrustZone, provide additional layers of protection. These features, coupled with robust encryption strategies, create a secure envelope around the application data, significantly enhancing its safety and the overall security posture of the system.

7 | REMAINING CHALLENGES OF CONFIDENTIAL COMPUTING

7.1 | Strengthening the Privacy Aspect of Confidential Computing

As one of the key protective objectives of TEE, privacy must be precisely and preferably theoretically specified using information theory principles. While majority of the privacy measures are evaluated empirically through execution of the attacks, establishing privacy theoretically is challenging owing to the prevalence of assaults.

Among contenders, *differential privacy* (DP)[143] is a widely recognized approach to achieve privacy. The DP approach ensures that the sensitive (private) information of an individual in a dataset is protected and their sensitive information is effectively anonymized, even when subjected to arbitrary side information. Importantly, the use of DP for ML-based applications has been extensively explored (*e.g.,* [144,145,146,147]). However, DP has generalization challenges, which is defined as the capacity of a privacy-preserving algorithm to maintain its efficacy when applied to data beyond the initial training dataset. As an example, one may use DP-training to demonstrate the effectiveness of protection against data replication attacks via recovering the original training data from gradients generated on it. Nevertheless, it is important to note that traditional DP training is primarily designed to protect the privacy of individual records in the final model, not the intermediate gradients. Therefore, protecting against such attacks using DP might require adjusting the granularity of DP from the sample-level (*i.e.,* sensitivity of individual records in the dataset) to the feature-level (*i.e.,* sensitivity of individual features within each record). While such adjustments can theoretically enhance privacy protection against gradient-based attacks, it is also likely to significantly impact the model's performance. In fact, there is a trade-off between privacy and utility in such protections.

Potential Solution: To overcome the limitations of differential privacy in ML applications, researchers are to further explore advanced cryptographic techniques, such as homomorphic encryption (HE), that enables computation on encrypted data, maintaining data privacy while allowing for useful operations. However, existing HE implementations suffer from the burden of extensive computational and memory overhead. Recent research studies[148] have been undertaken to propose architectural optimizations and algorithmic enhancements that address these challenges. These advancements include quantization, data packing, and bit-slicing techniques to streamline HE operations for confidential app deployment in edge devices.

In addition, the development of more sophisticated differential privacy mechanisms that offer a better balance between privacy and utility could significantly mitigate the generalization challenges. Innovations in privacy models that extend beyond the traditional frameworks of differential privacy, potentially incorporating concepts from quantum computing or advanced statistical methods, have the potential to offer new pathways to protect privacy more effectively and efficiently.

7.2 | Performance and Latency Issues of HE

HE allows computations to be carried out on ciphertext and produce an encrypted result that, upon decryption, matches the result of operations performed on the plaintext. This capability is benevolent for privacy-preserving nature of cloud-based ML task execution, as it ensures that sensitive data remains encrypted throughout the processing. However, translating complex ML algorithms into a domain where they can be executed homomorphically introduces substantial computational overhead. The main performance bottleneck is the significant increase in latency that can be orders of magnitude greater than computations performed on unencrypted data. Likewise, the latency in homomorphic operations is primarily due to the complex mathematical operations involved in addition and multiplication of ciphertexts. For ML models, especially, deep learning networks that require a large number of multiplicative depths for activation functions, the latency becomes unusually high. This issue is compounded upon scaling up to models with millions of parameters, such as those commonly used in image recognition and natural language processing.

Potential Solution: In spite of these challenges, recent advances in HE libraries and frameworks have started to address these performance issues. These improvements are largely driven by new libraries and frameworks designed to optimize HE operations; some notable ones are mentioned below:

- **Concrete-ML** library[149] optimizes traditional HE schemes for ML tasks by reducing computational complexity and latency.
- **TenSEAL** [121] provides tools for performing tensor operations on encrypted data, facilitating the integration of HE into existing tensor-based ML models (*e.g.,* TensorFlow, PyTorch).
- **PySyft**[150] is a flexible, open-source framework for encrypted, privacy-preserving machine learning. PySyft extends PyTorch and TensorFlow to enable multi-party computations (MPC) and HE.

- **HElib**[151] is an open-source library that implements Brakerski-Gentry-Vaikuntanathan HE scheme along with several optimizations to enhance performance in Smart-Vercauteren ciphertext packing techniques, efficient recryption methods, *etc.* .

7.3 | Addressing Vulnerabilities of TEE: SGX, TrustZone, and SEV

Developing strong and effective confidential computing solutions must take into account the constraints posed by the usage of TEE. The problem is that side-channel and reverse-engineering attacks are not included in the threat model of the TEEs, such as Intel SGX. Moreover, there are a number of security alerts reported concerning TrustZone, including kernel and driver issues, as well as hardware-related vulnerabilities that affect various hardware components of the platform[41].

Software developers may think that TEE is absolutely safe, but this is not the case. They must take into account defects and vulnerabilities in hardware and software components of TEEs. It is important for future research to address vulnerabilities of the current TEE technologies such as Intel SGX, ARM TrustZone, and AMD SEV. While these technologies offer strong data security measures, they are not immune to attacks. For example, the threat model of Intel SGX does not consider side-channel and reverse-engineering attacks, which are potential hardware vulnerabilities. Additionally, there have been security alerts concerning ARM TrustZone, such as kernel and driver issues, as well as hardware-related vulnerabilities that affect various hardware components of the platform.

Potential Solution: Recent research studies[152,153] have proposed several methods, such as address sanitizing and address masking to overcome several vulnerabilities of TEEs. Address sanitizing leverages the Memory Protection Unit (MPU) in non-secure states to verify privilege levels during state transitions, thereby, preventing unauthorized access. Address masking involves placing non-secure user-space and kernel-space programs within predefined memory regions, providing an additional layer of security via limiting the exploitable address spaces. Incorporating these techniques within TEEs can mitigate specific attack vectors, such as return-to-non-secure attacks, hence, enhancing the resilience of the secure environments. Intel SGX proposes dynamic memory isolation to limit the potential damage from side-channel attacks via isolating memory regions during the execution phases. This approach reduces exposure to timing-based attacks and improves security of the confidential processing[27]. Another study[154] introduces control-flow integrity (CFI) to prevent control-flow hijacking in TEEs. By tracking valid control flows and interrupting unauthorized branches, CFI can mitigate risks of unauthorized code execution within the secure environment.

It is essential for future research to focus on finding solutions for these vulnerabilities and ensure the maximum achievable TEE securities. This can be achieved via identifying the specific weaknesses of the TEE technologies and developing new and more effective solutions to address them. This includes the development of TEEs with enhanced hardware defenses and the establishment of a security lifecycle management system for continuous vulnerability assessment and patching. Furthermore, the next generation of TEE architectures should inherently resist known attack vectors, incorporating design principles that anticipate and mitigate potential threats.

7.4 | Integration of Distributed Trust and Confidential Computing

Distributed trust[155] is defined as trust models and solutions where the trust is established as a result of consensus across multiple nodes/entities. Blockchain[156] and distributed ledger[157] are popular examples of distributed trust. Solutions based on distributed trust have the potential to be integrated with and enhance the security and scalability of confidential computing. For instance, the decentralized nature of blockchain can provide an immutable and tamper-proof record of all data and computations, while confidential computing can protect the data and computations themselves from unauthorized access and tampering. Additionally, distributed trust can provide secure multi-party computation, enabling multiple parties to securely and privately collaborate on sensitive data and computations, while preserving the privacy and security of the underlying data.

Potential Solution: The convergence of blockchain technology with confidential computing could usher in a new era of secure and transparent data processing, where the integrity of computations is verifiable without compromising data confidentiality. Developing secure frameworks for multi-party computation, underpinned by TEEs, can facilitate privacy-preserving data analysis and collaboration. New consensus mechanisms tailored for confidential computing contexts should also be explored, focusing on scalability, efficiency, and security in distributed environments.

While traditional trust models often rely on centralized trust authorities, which can become bottlenecks and introduce vulnerabilities, distributing trust across multiple entities can mitigate the risk of single points of failure and more resilient protection of sensitive data. To further advance the field of confidential computing, however, the following aspects of distributed trust have to be further explored:

- *Trust establishment and management:* Developing novel mechanisms to establish and maintain trust across heterogeneous and dynamic environments, such as edge-to-cloud continuum. Techniques for trust negotiation, delegation, and revocation can also be investigated to ensure secure and efficient collaboration across various participating entities.
- *Zero Trust Access in IoTs:* Zero Trust model is a security concept centered on the belief that organizations should not automatically trust anything inside or outside their perimeters rather must verify everything trying to connect to their systems before granting access. This concept is particularly relevant to IoT, where devices are often highly diverse and can present significant security vulnerabilities. By adopting zero trust protocols, organizations can ensure that each IoT device is authenticated and its behavior continuously monitored, regardless of its location or network. Zero trust can be particularly effective when combined with other security technologies such as micro-segmentation and network access control (NAC), ensuring that IoT devices have only the minimum necessary access rights, and that any suspicious behavior can be rapidly detected and addressed. Implementing Zero trust in IoT environments can therefore greatly enhance their resilience against potential security threats.
- *Consensus algorithms:* Designing new consensus algorithms that can ensure trust and integrity in distributed confidential computing systems, while maintaining efficiency and scalability. These algorithms should be robust to malicious activities and adaptable to changing network conditions.
- *Secure data sharing and collaboration:* Implementing secure data sharing mechanisms that allow multiple parties to collaboratively process sensitive data while preserving data privacy and confidentiality. This can involve leveraging techniques such as multi-party computation[158], federated learning[145,144], and blockchain-based solutions[159].
- *Monitoring and auditing:* Developing monitoring and auditing tools to ensure the security and compliance of distributed trust deployments. These tools should be capable of detecting and mitigating security threats, as well as providing transparency and accountability to all participating entities.
- *Performance optimization:* Investigating methods to optimize the performance of distributed trust deployments while maintaining security and privacy guarantees. This can involve exploring trade-offs between security and performance, as well as developing adaptive algorithms that can dynamically adjust to different workloads and network conditions.

7.5 | Dedicated TEE Designs for General ML and LLM Workloads

Efficient execution of both General Machine Learning (ML) tasks and Large Language Models (LLMs) requires modern computer architectures equipped with advanced parallelization features, such as multi-threading on CPUs and accelerators (*e.g.,* GPU, TPU, *etc.*). Such hardware components are critical for performing computationally intensive operations such as matrix multiplications during forward and backward passes which are heavily performed in both regular ML and advanced ML (*e.g.,* LLM, computer vision) workloads. However, most current Trusted Execution Environments (TEEs) lack such capabilities, as they are not typically integrated with application-specific hardware (ASICs) optimized for ML workloads. Integrating these features within TEEs could enhance their performance for LLM-based applications, but it also increases the Trusted Computing Base (TCB) size, thereby expanding the potential attack surface. Moreover, synchronization bugs can cause severe vulnerability, as demonstrated by prior work on SGX[160] and similar risks may emerge in GPU-TEEs.

Avoiding such a dilemma depends on how to limit the trust boundary and reduce the TCB when applying parallel processing. One approach is similar to the way TPM operates: one ML accelerator (GPU/TPU) is physically isolated from the rest of the motherboard and the processing system. In this model, the accelerators would be accessed exclusively through secure buses linked to TEE-enabled CPUs which ensures sensitive data remains within a trusted boundary. Although this approach constrains the trust boundary and no one can physically breach the GPU, there are also other possible approaches such as ML as a Service (MLaaS)[26] with proper remote attestation and verification for securely handling a wide range of workloads.

Potential Solution: The advent of TEE architectures specifically designed for different ML workloads can strike a balance between computational efficiency and security. These designs can integrate specialized AI/ML accelerators to facilitate high-performance processing while reducing the TCB through innovative isolation techniques. For instance, securely isolating accelerators or using secure communication channels within TEEs can protect sensitive computations without compromising computational speed. Moreover, introducing secure multi-tenancy models within TEEs allows for efficient resource sharing while providing strong isolation for concurrent ML and LLM tasks. Additionally, secure multi-tenancy models within TEEs could enable a new paradigm for Machine Learning as a Service (MLaaS) by providing strong isolation guarantees alongside efficient resource utilization.

8 | SUMMARY

The overarching goal of confidential computing is to establish end-to-end data security and privacy that includes data at-rest, data in-transit, and data in-use across various computing systems. While other types of security has been extensively explored, confidential computing has mainly emerged to deal with the security of data in-use (while being processed). Confidential computing has become a vital research area due to: **(a)** the exponentially increasing volume of data generated by various sources, ranging from IoT-based sensors to social media activities; **(b)** data processing commonly occurs off-premises and on third-party servers—across edge-to-cloud continuum—where the need for secure processing of sensitive data is of paramount importance; and **(c)** prevalence of data-driven ML applications that process and identify confidential information about businesses and individuals.

Accordingly, this study aims at providing an overarching understanding of the fundamental concepts and of confidential computing at the hardware, middleware, and application levels. Moreover, this study surveys the existing solutions and state-of-the-art techniques available for confidential computing. More specifically, we delved into the core components of confidential computing, such as Trusted Execution Environments (TEEs), secure enclaves, and examined their applications in diverse domains, including cloud computing, IoT, edge computing, and particularly with respect to ML applications. In addition, we discussed the importance of establishing trust in both hardware, middleware, and software levels, along with the role of remote attestation procedures and trusted application development frameworks in achieving confidential computing.

Despite the considerable progress made in the field of confidential computing, several challenges and research opportunities have remained unexplored. Future research endeavors should focus on enhancing privacy aspects within confidential computing solutions, developing more secure and anonymous attestation mechanisms, and addressing vulnerabilities in current trusted execution environments. Furthermore, it is crucial to explore scalable and efficient confidential computing approaches that can effectively handle the massive volume of sensitive data being processed via ML-based solutions.

In conclusion, this survey offers a thorough overview of the confidential computing landscape, highlighting its importance in the ever-evolving digital age. By examining existing solutions, challenges, and future research directions, we hope to inspire researchers and practitioners alike to continue advancing this field, ultimately ensuring true secure processing of sensitive data across various applications and domains.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers of the paper. This research is supported by the National Science Foundation (NSF) under awards# CNS-2419588, CNS-2418188, and OISE-2417064.

REFERENCES

1. Bello-Orgaz G, Jung JJ, Camacho D. Social big data: Recent achievements and new challenges. *Journal of Information Fusion.* 2016;28:45–59.
2. Sanderson K. GPT-4 is here: what scientists think. *Jounal of Nature.* 2023;615(7954):773.
3. The 15 biggest data breaches of the 21st century. www.csoonline.com/article/2130877/data-breach/the-biggest-data-breaches-of-the-21st-century.html; Accessed November 21, 2019.
4. AWS IAM. https://aws.amazon.com/iam/; Accessed February '23.
5. What is Amazon GuardDuty?. https://docs.aws.amazon.com/guardduty/latest/ug/what-is-guardduty.html; Accessed February '23.
6. What is AWS Security Hub?. https://docs.aws.amazon.com/securityhub/latest/userguide/what-is-securityhub.html; Accessed March '23.
7. Zobaed S. AI-Driven Confidential Computing across Edge-to-Cloud Continuum. *arXiv preprint arXiv:2301.00928.* 2023.
8. Koutsopoulos HN, Noursalehi P, Zhu Y, Wilson NH. Automated data in transit: Recent developments and applications. In: 2017:604–609.
9. What is TLS (Transport Layer Security)?. https://www.cloudflare.com/learning/ssl/transport-layer-security-tls; Accessed Feb '21.
10. Protecting the three states of data. https://www.sealpath.com/blog/protecting-the-three-states-of-data; Accessed March 10, 2023.
11. Mulligan DP, Petri G, Spinale N, Stockwell G, Vincent HJ. Confidential Computinga brave new world. In: IEEE. 2021:132–138.
12. Ning Z, Liao J, Zhang F, Shi W. Preliminary study of trusted execution environments on heterogeneous edge platforms. In: 2018:421–426.
13. Yu W, Liang F, He X, et al. A survey on the edge computing for the Internet of Things. *Journal of IEEE access.* 2017;6:6900–6919.

14. Gong C, Lin F, Gong X, Lu Y. Intelligent Cooperative Edge Computing in Internet of Things. *Journal of Internet of Things.* 2020;7(10):9372–9382.
15. Shepherd C, Arfaoui G, Gurulian I, et al. Secure and trusted execution: Past, present, and future-a critical review in the context of the internet of things and cyber-physical systems. In: 2016:168–177.
16. Sabt M, Achemlal M, Bouabdallah A. Trusted execution environment: what it is, and what it is not. In: . 1. 2015:57–64.
17. Sabt M, Achemlal M, Bouabdallah A. The dual-execution-environment approach: Analysis and comparative evaluation. In: 2015:557–570.
18. Cinque M, Cotroneo D, De Simone L, Rosiello S. Virtualizing mixed-criticality systems: A survey on industrial trends and issues. *Future Generation Computer Systems.* 2022;129:315–330.
19. Arbaugh WA, Farber DJ, Smith JM. A secure and reliable bootstrap architecture. In: 1997:65–71.
20. Sangorrin D, Honda S, Takada H. Integrated scheduling for a reliable dual-os monitor. *Information and Media Technologies.* 2012;7(2):627–638.
21. Santos N, Raj H, Saroiu S, Wolman A. Using ARM TrustZone to build a trusted language runtime for mobile applications. In: 2014:67–80.
22. Sangorrín D, Honda S, Takada H. Reliable and efficient dual-os communications for real-time embedded virtualization. *Information and Media Technologies.* 2013;8(1):1–17.
23. Jangid MK, Chen G, Zhang Y, Lin Z. Towards Formal Verification of State Continuity for Enclave Programs.. In: 2021:573–590.
24. Li W, Xia Y, Chen H. Research on arm trustzone. *Journal of GetMobile: Mobile Computing and Communications.* 2019;22(3):17–22.
25. Fei S, Yan Z, Ding W, Xie H. Security vulnerabilities of SGX and countermeasures: A survey. *ACM Computing Surveys (CSUR).* 2021;54(6):1–36.
26. Mo F, Tarkhani Z, Haddadi H. Sok: Machine learning with confidential computing. *arXiv preprint arXiv:2208.10134.* 2022.
27. Chen G, Chen S, Xiao Y, Zhang Y, Lin Z, Lai TH. Sgxpectre: Stealing intel secrets from sgx enclaves via speculative execution. In: 2019:142–157.
28. Valadares DCG, Will NC, Spohn MA, Souza Santos dDF, Perkusich A, Gorgonio KC. Trusted Execution Environments for Cloud/Fog-based Internet of Things Applications.. In: 2021:111–121.
29. Ménétrey J, Göttel C, Pasin M, Felber P, Schiavoni V. An Exploratory Study of Attestation Mechanisms for Trusted Execution Environments. *arXiv preprint arXiv:2204.06790.* 2022.
30. What is a Hardware Security Module (HSM)?. https://www.entrust.com/resources/hsm/faq/what-are-hardware-security-modules; Accessed June '23.
31. Weiser S, Werner M. Sgxio: Generic trusted i/o path for intel sgx. In: 2017:261–268.
32. AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More. https://www.amd.com/en/processors/amd-secure-encrypted-virtualization; Accessed July 12, 2022.
33. Woodworth J, Salehi MA, Raghavan V. S3C: An architecture for space-efficient semantic search over encrypted data in the cloud. In: IEEE. 2016:3722–3731.
34. Woodworth JW, Amini Salehi M. S3BD: Secure semantic search over encrypted big data in the cloud. *Concurrency and Computation: Practice and Experience.* 2019;31(11):e5050.
35. Zobaed S, Amini Salehi M. Privacy-preserving clustering of unstructured big data for cloud-based enterprise search solutions. *Journal of Concurrency and Computation: Practice and Experience.* 2022;34(22):e7160.
36. Zobaed SM, Amini Salehi M, Buyya R. SAED: Edge-based intelligence for privacy-preserving enterprise search on the cloud. In: 2021:366–375.
37. Aublin PL, Kelbert F, O'Keeffe D, et al. Libseal: Revealing service integrity violations using trusted execution. In: 2018:1–15.
38. Nguyen H, Ivanov R, Phan LT, Sokolsky O, Weimer J, Lee I. LogSafe: Secure and scalable data logger for IoT devices. In: 2018:141–152.
39. Valadares DCG, Silva dMSL, Brito AEM, Salvador EM. Achieving data dissemination with security using FIWARE and Intel software guard extensions (SGX). In: 2018:1–7.
40. Ayoade G, El-Ghamry A, Karande V, Khan L, Alrahmawy M, Rashad MZ. Secure data processing for IoT middleware systems. *Journal of Supercomputing.* 2019;75(8):4684–4709.
41. Pinto S, Santos N. Demystifying arm trustzone: A comprehensive survey. *ACM computing surveys (CSUR).* 2019;51(6):1–36.
42. Van Bulck J, Oswald D, Marin E, Aldoseri A, Garcia FD, Piessens F. A tale of two worlds: Assessing the vulnerability of enclave shielding runtimes. In: 2019:1741–1758.
43. Brenner S, Kapitza R. Trust more, serverless. In: 2019:33–43.
44. Ibrahim FA, Hemayed EE. Trusted cloud computing architectures for infrastructure as a service: Survey and systematic literature review. *Computers & Security.* 2019;82:196–226.
45. Aslanpour MS, Toosi AN, Cicconetti C, et al. Serverless edge computing: vision and challenges. In: 2021:1–10.
46. Li X, Leng X, Chen Y. Securing Serverless Computing: Challenges, Solutions, and Opportunities. *arXiv preprint arXiv:2105.12581.* 2021.
47. Wang H, Cai L, Hao X, Ren J, Ma Y. ETS-TEE: An energy-efficient task scheduling strategy in a mobile trusted computing environment. *Tsinghua Science and Technology.* 2022;28(1):105–116.
48. Zhao S, Xu P, Chen G, Zhang M, Zhang Y, Lin Z. Reusable enclaves for confidential serverless computing. In: 2023:4015–4032.
49. Wu H, Wolter K, Jiao P, Deng Y, Zhao Y, Xu M. EEDTO: An energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing. *Journal of Internet of Things.* 2021;8(4):2163–2176.
50. Qu G, Cui N, Wu H, Li R, Ding Y. ChainFL: A simulation platform for joint federated learning and blockchain in edge/cloud computing environments. *IEEE Transactions on Industrial Informatics.* 2021;18(5):3572–3581.
51. Li Y, Zeng D, Gu L, Zhu A, Chen Q, Yu S. PASTO: enabling secure and efficient task offloading in trustZone-enabled edge clouds. *IEEE Transactions on Vehicular Technology.* 2023.
52. Costan V, Devadas S. Intel sgx explained.. *IACR Cryptol. ePrint Arch..* 2016;2016(86):1–118.
53. Mofrad S, Zhang F, Lu S, Shi W. A comparison study of intel SGX and AMD memory encryption technology. In: 2018:1–8.
54. Van Bulck J, Minkin M, Weisse O, et al. Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution. In: 2018:991–1008.
55. Liang H, Li M, Chen Y, Jiang L, Xie Z, Yang T. Establishing trusted i/o paths for sgx client systems with aurora. *IEEE Transactions on Information Forensics and Security.* 2019;15:1589–1600.
56. TEE SoC Based on RISC-V. https://riscv.org/blog/2023/02/tee-soc-based-on-risc-v; Accessed May, 2023.

57. Nashimoto S, Suzuki D, Ueno R, Homma N. Bypassing Isolated Execution on RISC-V using Side-Channel-Assisted Fault-Injection and Its Countermeasure. *IACR Transactions on Cryptographic Hardware and Embedded Systems.* 2022:28–68.
58. Nashimoto S, others . PoC TEE: RISC-V Proof-of-Concept Trusted Execution Environment. 2022.
59. The First TEE For RISC-V. https://hex-five.com/multizone-security-tee-riscv; Accessed May, 2023.
60. Open Enclave SDK. https://openenclave.io/sdk; Accessed May 2023.
61. Keystone. https://keystone-enclave.org; Accessed June, 2023.
62. Zhang F, Zhang H. SoK: A study of using hardware-assisted isolated execution environments for security. In: , , , 2016:1–8.
63. Ning Z, Zhang F, Shi W, Shi W. Position paper: Challenges towards securing hardware-assisted execution environments. In: , , , 2017:1–8.
64. Koning K, Chen X, Bos H, Giuffrida C, Athanasopoulos E. No need to hide: Protecting safe regions on commodity hardware. In: 2017:437–452.
65. Schuster F, Costa M, Fournet C, et al. VC3: Trustworthy data analytics in the cloud using SGX. In: 2015:38–54.
66. Ning Z, Zhang F. Ninja: Towards Transparent Tracing and Debugging on ARM. In: 2017:33–49.
67. Platform Hierarchy. https://ebrary.net/24759/computer_science/platform_hierarchy; Accessed June '23.
68. Intel Trusted Execution Technology (TXT). https://www.intel.com/content/www/us/en/developer/articles/tool/intel-trusted-execution-technology.html; Accessed June '23.
69. Intel Platform Trust Technology (PTT): TPM For The Masses. https://www.onlogic.com/company/io-hub/intel-platform-trust-technology-ptt-tpm-for-the-masses; Accessed June '23.
70. Raj H, Saroiu S, Wolman A, et al. fTPM: A Software-Only Implementation of a TPM Chip. In: 2016:841–856.
71. Virtual Trusted Platform Module Overview. https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.security.doc/GUID-6F811A7A-D58B-47B4-84B4-73391D55C268.html; Accessed June '23.
72. Everything you need to know about TPM to be able to install Windows 11. https://rb.gy/pv2ky; Accessed December, 2021.
73. near-field communication (NFC). https://www.techtarget.com/searchmobilecomputing/definition/Near-Field-Communication; Accessed June, 2023.
74. Secure Element. https://https://encyclopedia.kaspersky.com/glossary/secure-element; Accessed June, 2023.
75. Smartcard, UICC and secure element testing. https://korea.fime.com/services/smartcard-and-sim-testing/smartcard-uicc-and-secure-element-testing; Accessed June, 2023.
76. AES SECURITY MEMORY CARD. https://www.flexxon.com/aes-security-sd-microsd-card; Accessed June, 2023.
77. Zobaed S, Amini Salehi M. Big Data in the Cloud. In: Schintler LA, McNeely CL., eds. *Encyclopedia of Big Data*, , Springer, 2018.
78. Cloud Leak: How A Verizon Partner Exposed Millions of Customer Accounts. https://www.upguard.com/breaches/verizon-cloud-leak; Accessed April '22.
79. Every Single Yahoo Account Was Hacked - 3 Billion in All. https://www.money.cnn.com/2017/10/03/technology/business/yahoo-breach-3-billion-accounts/index.html; Accessed February '21.
80. Samani DG, Amini Salehi M. Exploring the Impact of Virtualization on the Usability of the Deep Learning Applications. In: CCGrid '22. 2022.
81. Denninnart C, Amini Salehi M. Harnessing the Potential of Function-Reuse in Multimedia Cloud Systems. *IEEE Transactions on Parallel and Distributed Systems.* 2021;33(3):617–629.
82. Ghatreh Samani D, Denninnart C, Bacik J, Amini Salehi M. The Art of CPU-Pinning: Evaluating and Improving the Performance of Virtualization and Containerization Platforms. In: ICPP '20. 2020.
83. Grassi G, Jamieson K, Bahl P, Pau G. Parkmaster: An in-vehicle, edge-based video analytics service for detecting open parking spaces in urban environments. In: 2017:1–14.
84. Wu X, Dunne R, Zhang Q, Shi W. Edge computing enabled smart firefighting: opportunities and challenges. In: 2017:1–6.
85. Yi S, Hao Z, Zhang Q, Zhang Q, Shi W, Li Q. Lavea: Latency-aware video analytics on edge computing platform. In: 2017:1–13.
86. Chen Y, Feng Q, Shi W. An industrial robot system based on edge computing: An early experience. In: 2018.
87. Zhang Q, Yu Z, Shi W, Zhong H. Demo abstract: Evaps: Edge video analysis for public safety. In: 2016:121–122.
88. Qi B, Kang L, Banerjee S. A vehicle-based edge computing platform for transit and human mobility analytics. In: 2017:1–14.
89. Zhang X, Zheng X, Wang Z, Yang H, Shen Y, Long X. High-density Multi-tenant Bare-metal Cloud. In: 2020:483–495.
90. Bare Metal Cloud vs IaaS: What are the Differences?. https://phoenixnap.com/blog/bare-metal-cloud-vs-iaas; Accessed February 1, 2022.
91. Securing Applications On Bare-metal Instances. https://www.anjuna.io/blog/securing-applications-on-bare-metal-instances; Accessed February 1, 2022.
92. Securing Applications On Bare-metal Instances. https://cloud.ibm.com/docs/bare-metal?topic=bare-metal-bm-server-provision-sgx; Accessed February 1, 2022.
93. Production-Grade Container Orchestration. https://kubernetes.io/; Accessed June '23.
94. Denninnart C, Chanikaphon T, Amini Salehi M. Efficiency in the Serverless Cloud Paradigm: A Survey on the Reusing and Approximation Aspects. *Journal of Software-Practice and Experience (SPE).* 2023;In press.
95. Serverless Examples. https://github.com/aws-samples; Accessed December, 2021.
96. Azure Functions overview. https://learn.microsoft.com/en-us/azure/azure-functions/functions-overview?pivots=programming-language-csharp; Accessed June '23.
97. Cloud Functions for Firebase Sample Library. https://github.com/firebase/functions-samples; Accessed December, 2021.
98. Shahrad M, Fonseca R, Goiri Í, et al. Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. In: 2020:205–218.
99. Li M, Xia Y, Chen H. Confidential serverless made efficient with plug-in enclaves. In: 2021:306–318.
100. Jindal A, Chadha M, Benedict S, Gerndt M. Estimating the capacities of function-as-a-service functions. In: 2021:1–8.
101. Trach B, Oleksenko O, Gregor F, Bhatotia P, Fetzer C. Clemmys: Towards secure remote execution in faas. In: 2019:44–54.
102. Weisse O, Bertacco V, Austin T. Regaining lost cycles with HotCalls: A fast interface for SGX secure enclaves. *ACM SIGARCH Computer Architecture News.* 2017;45(2):81–93.
103. Orenbach M, Lifshits P, Minkin M, Silberstein M. Eleos: ExitLess OS services for SGX enclaves. In: 2017:238–253.
104. Feng E, Lu X, Du D, et al. Scalable Memory Protection in the PENGLAI Enclave. In: 2021:275–294.
105. The Most Widely Deployed Open Source Cloud Software in the World. https://www.openstack.org; Accessed June '23.
106. Kugler L. Standards to Secure the Sensors That Power IoT. *Communications of the ACM.* 2023;66(6):14–16.

107. Di Martino B, Rak M, Ficco M, Esposito A, Maisto SA, Nacchia S. Internet of things reference architectures, security and interoperability: A survey. *Journal of Internet of Things.* 2018;1:99–112.
108. Cirne A, Sousa PR, Resende JS, Antunes L. IoT security certifications: Challenges and potential approaches. *Journal of Computers & Security.* 2022;116:102669.
109. Wasicek A. The future of 5G smart home network security is micro-segmentation. *Journal of Network security.* 2020;2020(11):11–13.
110. Zobaed S, Mokhtari A, Champati JP, Kourouma M, Amini Salehi M. Edge-MultiAI: Multi-Tenancy of Latency-Sensitive Deep Learning Applications on Edge. *arXiv preprint arXiv:2211.07130.* 2022.
111. Graepel T, Lauter K, Naehrig M. ML confidential: Machine learning on encrypted data. In: Springer. 2012:1–21.
112. Ohrimenko O, Schuster F, Fournet C, et al. Oblivious multi-party machine learning on trusted processors.. In: . 16. 2016:10–12.
113. Hunt T, Song C, Shokri R, Shmatikov V, Witchel E. Chiron: Privacy-preserving machine learning as a service. *arXiv preprint arXiv:1803.05961.* 2018.
114. Lee T, Lin Z, Pushp S, et al. Occlumency: Privacy-preserving remote deep-learning inference using sgx. In: 2019:1–17.
115. Mo F, Shamsabadi AS, Katevas K, et al. Darknetz: towards model privacy at the edge using trusted execution environments. In: 2020:161–174.
116. Liu Z, Lu Y, Xie X, Fang Y, Jian Z, Li T. Trusted-dnn: A trustzone-based adaptive isolation strategy for deep neural networks. In: 2021:67–71.
117. Zhang C, Xia J, Yang B, et al. Citadel: Protecting data privacy and model confidentiality for collaborative learning. In: 2021:546–561.
118. Mo F, Haddadi H, Katevas K, Marin E, Perino D, Kourtellis N. Ppfl: Enhancing privacy in federated learning with confidential computing. *GetMobile: Mobile Computing and Communications.* 2022;25(4):35–38.
119. Sander J, Berndt S, Bruhns I, Eisenbarth T. DASH: Accelerating Distributed Private Machine Learning Inference with Arithmetic Garbled Circuits. *arXiv preprint arXiv:2302.06361.* 2023.
120. Lawhern VJ, Solon AJ, Waytowich NR, Gordon SM, Hung CP, Lance BJ. EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces. *Journal of neural engineering.* 2018;15(5):056013.
121. Benaissa A, Retiat B, Cebere B, Belfedhal AE. Tenseal: A library for encrypted tensor operations using homomorphic encryption. *arXiv preprint arXiv:2104.03152.* 2021.
122. Louk M, Lim H. Homomorphic encryption in mobile multi cloud computing. In: 2015:493–497.
123. Chotard J, Dufour-Sans E, Gay R, Phan DH, Pointcheval D. Dynamic decentralized functional encryption. In: Springer. 2020:747–775.
124. So many bad takes—What is there to learn from the Prime Video “monolith” story?. thestack.technology/prime-video-monolith-architecture-debate-bad-takes-adrian; Accessed May ’23.
125. Wu H, Knottenbelt WJ, Wolter K. An efficient application partitioning algorithm in mobile environments. *IEEE Transactions on Parallel and Distributed Systems.* 2019;30(7):1464–1480.
126. Xu M, Zhou Q, Wu H, Lin W, Ye K, Xu C. PDMA: Probabilistic service migration approach for delay-aware and mobility-aware mobile edge computing. *Software: Practice and Experience.* 2022;52(2):394–414.
127. Zhou L, Wen H, Teodorescu R, Du DH. Distributing deep neural networks with containerized partitions at the edge. In: 2019.
128. Candal-Ventureira D, González-Castaño FJ, Gil-Castiñeira F, Fondo-Ferreiro P. Is the edge really necessary for drone computing offloading? An experimental assessment in carrier-grade 5G operator networks. *Software: Practice and Experience.* 2023;53(3):579–599.
129. Liu B, Luo Z, Chen H, Li C. A survey of state-of-the-art on edge computing: Theoretical models, technologies, directions, and development paths. *IEEE Access.* 2022;10:54038–54063.
130. Islam A, Debnath A, Ghose M, Chakraborty S. A survey on task offloading in multi-access edge computing. *Journal of Systems Architecture.* 2021;118:102225.
131. Luo Q, Hu S, Li C, Li G, Shi W. Resource scheduling in edge computing: A survey. *IEEE Communications Surveys & Tutorials.* 2021;23(4):2131–2165.
132. Aniche M, Bavota G, Treude C, Gerosa MA, Van Deursen A. Code smells for model-view-controller architectures. *Empirical Software Engineering.* 2018;23:2121–2157.
133. Wulf C, Hasselbring W, Ohlemacher J. Parallel and generic pipe-and-filter architectures with TeeTime. In: IEEE. 2017:290–293.
134. Ampatzoglou A, Michou O, Stamelos I. Building and mining a repository of design pattern instances: Practical and research benefits. *Entertainment Computing.* 2013;4(2):131–142.
135. Tang S, Liang Z, Zhu Y. Numerical investigation on heat transfer characteristics in electronic cavity of downhole measurement-while-drilling system. *Journal of Thermal Science and Engineering Applications.* 2021;13(1).
136. Chanikaphon T, Salehi MA. Ums: Live migration of containerized services across autonomous computing systems. In: IEEE. 2023:467–472.
137. Manatura S, Chanikaphon T, Chantrapornchai C, Amini Salehi M. FastMig: Leveraging FastFreeze to Establish Robust Service Liquidity in Cloud 2.0. In: IEEE. 2024:81–90.
138. Dunnings AJ, Breckon TP. Experimentally defined convolutional neural network architecture variants for non-temporal real-time fire detection. In: 2018:1558–1562.
139. Hussain RF, Amini Salehi M. Resource Allocation of Industry 4.0 Micro-Service Applications across Serverless Fog Federation. *submitted to the Future Generation Computing Systems (FGCS).* 2023.
140. Hosseinalipour S, Azam SS, Brinton CG, et al. Multi-Stage Hybrid Federated Learning over Large-Scale D2D-Enabled Fog Networks. *arXiv preprint arXiv:2007.09511.* 2020.
141. Ko JH, Na T, Amir MF, Mukhopadhyay S. Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained internet-of-things platforms. In: 2018:1–6.
142. Chinchali SP, Cidon E, Pergament E, Chu T, Katti S. Neural networks meet physical networks: Distributed inference betweenedge devices and the cloud. In: 2018:50–56.
143. Dwork C. Differential privacy: A survey of results. In: Springer. 2008:1–19.
144. Adnan M, Kalra S, Cresswell JC, Taylor GW, Tizhoosh HR. Federated learning and differential privacy for medical image analysis. *Scientific reports.* 2022;12(1):1953.
145. Zhao K, Hu J, Shao H, Hu J. Federated multi-source domain adversarial adaptation framework for machinery fault diagnosis with data privacy. *Reliability Engineering & System Safety.* 2023;236:109246.
146. Ponomareva N, Hazimeh H, Kurakin A, et al. How to dp-fy ml: A practical guide to machine learning with differential privacy. *arXiv preprint arXiv:2303.00654.* 2023.

147. Zhang X, Yang F, Guo Y, Yu H, Wang Z, Zhang Q. Adaptive differential privacy mechanism based on entropy theory for preserving deep neural networks. *Mathematics.* 2023;11(2):330.
148. Sinha S, Saha S, Alam M, et al. Exploring Bitslicing Architectures for Enabling FHE-Assisted Machine Learning. *Transactions on Computer-Aided Design of Integrated Circuits and Systems.* 2022;41(11):4004–4015.
149. Zama . Concrete ML: a Privacy-Preserving Machine Learning Library using Fully Homomorphic Encryption for Data Scientists. 2022. https://github.com/zama-ai/concrete-ml.
150. Ziller A, Trask A, Lopardo A, et al. Pysyft: A library for easy federated learning. *Federated learning systems: Towards next-generation AI.* 2021:111–139.
151. Halevi S, Shoup V. Design and implementation of HElib: a homomorphic encryption library. *Cryptology ePrint Archive.* 2020.
152. Ma Z, Tan X, Ziarek L, Zhang N, Hu H, Zhao Z. Return-to-Non-Secure Vulnerabilities on ARM Cortex-M TrustZone: Attack and Defense. In: 2023:1–6.
153. Kumari KA, Sharma A, Chakraborty C, Ananyaa M. Preserving health care data security and privacy using Carmichael's theorem-based homomorphic encryption and modified enhanced homomorphic encryption schemes in edge computing systems. *Journal of Big Data.* 2022;10(1):1–17.
154. Yeo G, Kim Y, Song S, Kwon D. Efficient CFI Enforcement for Embedded Systems Using ARM TrustZone-M. *Journal of IEEE Access.* 2022;10:132675–132684.
155. Wei L, Yang Y, Wu J, Long C, Li B. Trust management for internet of things: A comprehensive study. *Journal of Internet of Things.* 2022;9(10):7664–7679.
156. Krichen M, Ammi M, Mihoub A, Almutiq M. Blockchain for modern applications: A survey. *Sensors.* 2022;22(14):5274.
157. Burkhardt D, Werling M, Lasi H. Distributed ledger. In: IEEE. 2018:1–9.
158. Riazi MS, Weinert C, Tkachenko O, Songhori EM, Schneider T, Koushanfar F. Chameleon: A hybrid secure computation framework for machine learning applications. In: 2018:707–721.
159. Wan J, Li J, Imran M, Li D, others . A blockchain-based solution for enhancing security and privacy in smart factory. *IEEE Transactions on Industrial Informatics.* 2019;15(6):3652–3660.
160. Cloosters T, Rodler M, Davi L. TeeRex: discovery and exploitation of memory corruption vulnerabilities in SGX enclaves. In: 2020:841–858.