PEER-REVIEWED JOURNAL OF COMPUTER SCIENCE (PRJCS)

ISSN Online: 3139-5082 Open Access Scholarly Journal

Confidential Computing and Trusted Execution Environments: A Survey of Technologies

Dr. R. Pugazhenthi

Professor, Faculty of Mechanical Engineering, Vels Institute of Science, Technology & Advanced Studies, Tamil Nadu, India.

Article information

Received: 16th June 2026
Received in revised form: 29th June 2026
Accepted: 7th July 2026
Available online:13th July 2026

Volume: 1
Issue: 7
DOI: https://doi.org/10.5281/zenodo.21334866

Abstract

Conventional security controls protect data at rest through storage encryption and data in transit through transport encryption, yet data in use, meaning plaintext held in memory and registers during computation, has remained exposed to any sufficiently privileged component of the software stack. Confidential computing addresses this gap by performing sensitive computation inside a hardware-based, attested Trusted Execution Environment (TEE) that isolates code and data even from the operating system, hypervisor, and cloud operator. This survey reviews the motivation, threat model, and technology landscape of confidential computing. It examines process enclaves such as Intel SGX alongside confidential virtual machine designs including Intel TDX, AMD SEV and SEV-SNP, Arm TrustZone with the Confidential Compute Architecture, and the RISC-V Keystone framework, comparing them across isolation granularity, memory encryption, attestation, and vendor. It describes remote attestation and hardware roots of trust, surveys applications spanning confidential artificial intelligence, privacy-preserving analytics, blockchain, key management, and multi-party computation, and positions the paradigm as a complement to homomorphic encryption, secure multi-party computation, and differential privacy rather than a replacement. A dedicated security analysis reports side-channel and transient-execution attacks such as Foreshadow, Plundervolt, and SGAxe, together with representative performance overheads, and the review closes with standardization efforts and future directions including confidential graphics processing units.

Keywords: - Confidential Computing, Trusted Execution Environment, Intel SGX, AMD SEV-SNP, Remote Attestation, Side-Channel Attacks, Confidential AI, Memory Encryption.

I. INTRODUCTION

Modern information systems routinely protect data in two of its three classical states. Data at rest is guarded by full-disk and database encryption, and data in transit is guarded by transport protocols such as TLS. The third state, data in use, refers to plaintext values that must exist in memory, caches, and registers while a program actually computes on them. In traditional architectures this state is protected only by process isolation enforced by privileged software. Any component with higher privilege, including the operating system kernel, the hypervisor, firmware, or a cloud administrator, can in principle read or tamper with that plaintext. As organizations move regulated and proprietary workloads to shared, multi-tenant infrastructure, this residual exposure has become a central obstacle to trust.

Confidential computing responds to this problem. A Trusted Execution Environment, or TEE, is an isolated processing environment that provides confidentiality and integrity guarantees for the code and data placed inside it, backed by hardware rather than by the surrounding software [1]. The Confidential Computing Consortium defines

confidential computing itself as the protection of data in use by performing computation inside a hardware-based, attested TEE [2]. The distinguishing property is that the trust boundary is drawn around a small hardware-enforced enclave or protected virtual machine, allowing a workload to run correctly even when the platform software outside that boundary is untrusted or actively hostile [3].

Several forces drive adoption. Cloud and multi-tenant deployment concentrate sensitive workloads on infrastructure that customers do not physically control, so a technical means of reducing trust in the operator has direct commercial value. Regulatory regimes covering personal, financial, and health data create demand for demonstrable technical safeguards and for auditable evidence that a workload ran in an approved configuration. Beyond compliance, confidential computing enables collaborations that were previously impossible, such as two competitors jointly training a model on their combined data, or a hospital running an analysis supplied by an external vendor whose algorithm it cannot inspect, because each party can verify the protected environment before contributing anything sensitive.

This survey takes an honest position throughout. Careful reviewers have shown that vendor and consortium definitions of the field are not always precise or complete, and that security comparisons with neighboring technologies can be uneven [4]. The paper therefore presents the guarantees these systems offer together with the attacks and overheads that qualify them. It surveys the main TEE technologies and their trade-offs, explains remote attestation, catalogues applications, positions confidential computing among other privacy-enhancing technologies, and analyzes side channels and performance. Representative performance numbers are labelled as such, because reported figures vary widely with hardware generation, workload, and measurement methodology, and no single figure characterizes the field.

II. BACKGROUND AND THREAT MODEL

The threat model of confidential computing is unusually strong on the software side. The adversary is assumed to hold privileged control of the platform. It may run a malicious or compromised operating system and hypervisor, control system firmware, manipulate paging and interrupts, and in some deployments observe or interfere with the physical memory bus. What the workload owner wants is a guarantee that, despite this control, the confidentiality and integrity of the protected computation are preserved, or that any violation is detectable [5]. This inversion of the usual trust relationship, in which the platform owner is not automatically trusted by the workload owner, is the defining characteristic of the paradigm and separates it from conventional access control, which assumes a trusted kernel.

Three hardware mechanisms make such guarantees plausible. The first is memory encryption. Protected memory is encrypted, and in stronger designs integrity-protected, by a hardware engine sitting between the processor and the memory controller, so that data leaving the processor package appears as ciphertext to anyone inspecting DRAM or capturing a cold-boot image. Keys are generated within the hardware and are never exposed to software, and stronger designs bind ciphertext to its physical address to defeat relocation and replay. The second mechanism is access control enforced in silicon. Special processor states or address-space tags prevent even privileged ring-zero software from reading enclave memory in plaintext, and attempts to do so return ciphertext or raise a fault. Intel SGX, for example, introduced instructions that build an enclave in protected memory, measure its initial contents, and mediate entry and exit, keeping the enclave sealed from the surrounding process and the kernel [6].

The third mechanism is a hardware root of trust combined with attestation. Because the workload owner does not trust the platform, the platform must prove what it is running. A root of trust embedded in the hardware measures the code and configuration loaded into the TEE and signs that measurement with a device key that chains to the silicon vendor. A remote party can then verify the measurement before releasing secrets to the environment. A companion facility, sealing, lets an enclave encrypt data to its own identity so that persisted secrets can be recovered only by the same measured code on the same platform. This survey treats attestation in a dedicated section because it is what converts local isolation into remotely verifiable trust.

It is important to state the limits of the model at the outset. Hardware isolation defends against direct reads of protected memory, but the shared microarchitecture beneath the isolation boundary can still leak information. A privileged operating system can observe an enclave indirectly, for instance by manipulating page tables and watching the resulting fault pattern to infer secret-dependent control flow, an approach known as a controlled-channel attack [7]. Such channels are not blocked by memory encryption, because they exploit metadata and timing rather than the encrypted contents, and they motivate much of the security analysis presented later in this paper.

III. TEE TECHNOLOGIES

TEE designs differ principally in the granularity of the protected unit. Process enclaves protect a region within a single application address space, while confidential virtual machine designs protect an entire guest, its kernel included, from the hypervisor. This choice cascades into the size of the trusted computing base, the effort needed to port software, and the attack surface. Table 1 compares the main platforms across granularity, memory protection, attestation, and vendor.

A. Process Enclaves: Intel SGX

Intel Software Guard Extensions, or SGX, pioneered the process-enclave model on commodity server and client processors [6]. A developer partitions an application into an untrusted host and a trusted enclave; the enclave lives in an encrypted region of physical memory, and code enters and leaves it through controlled call gates. The trusted computing

base is small, limited to the enclave code and the processor, which is attractive for security. The cost is a demanding programming model, because the enclave cannot make system calls directly and every crossing of the boundary carries overhead. Early hardware limited the protected memory to a small enclave page cache, so larger footprints paid heavy paging costs, while later generations changed the cache size and the encryption approach, which in turn changed the performance profile. A comprehensive technical account of the architecture and its early security analysis remains a standard reference [5].

B. Confidential Virtual Machines: AMD SEV-SNP and Intel TDX

Confidential virtual machines protect a whole guest rather than a slice of one process, which greatly reduces the burden of rewriting applications. AMD Secure Encrypted Virtualization encrypts guest memory with a per-guest key managed by a dedicated secure processor, and its SEV-SNP extension adds integrity protection and a reverse-map table that prevents a malicious hypervisor from remapping, replaying, or aliasing guest pages [8]. Intel Trust Domain Extensions, or TDX, provide a comparable abstraction called a trust domain, isolating the guest from the virtual machine monitor through a combination of memory encryption, integrity metadata, and a firmware module that mediates transitions between the untrusted host and the protected domain [9]. Both approaches let an unmodified guest operating system and its applications run inside the protected boundary, shifting the trust boundary from the individual enclave to the whole virtual machine and easing lift-and-shift migration of legacy workloads.

C. Arm TrustZone and the Confidential Compute Architecture

Arm TrustZone partitions a system into a normal world and a secure world, providing a coarse hardware-enforced split that has been widely deployed in mobile and embedded devices for trusted services such as payment, biometric handling, and key storage [10]. The classic two-world model is powerful but limited: it offers one privileged secure world rather than many mutually isolated environments, and its trusted software base has historically been large, which enlarges the attack surface. The Confidential Compute Architecture, or CCA, extends the Armv9 platform with the Realm Management Extension to create dynamically isolated protected virtual machines whose confidentiality is enforced by hardware and a small monitor, and whose design has been subjected to formal verification [11]. This brings the confidential virtual machine model to the Arm ecosystem alongside the established TrustZone facility, and the small verified monitor is an explicit response to the trusted-base concern.

D. RISC-V Keystone

Keystone is an open framework for building TEEs on unmodified RISC-V hardware [12]. Rather than fixing one enclave abstraction in silicon, it uses standard RISC-V physical memory protection primitives together with a small trusted security monitor running in machine mode to construct customizable enclaves. Because the monitor and its policies are open and modular, Keystone lets researchers and integrators tailor the trusted computing base, the isolation model, and the memory protections to a target platform, which is valuable for transparency and for constrained devices where a vendor black box is undesirable. Its openness also makes it a common vehicle for security research, since the entire trusted path can be inspected and modified.

The choice between process enclaves and confidential virtual machines is a genuine trade-off rather than a strict ranking. Process enclaves minimize the trusted computing base but demand application partitioning and impose costly boundary crossings, so they suit small, well-defined secrets such as keys and narrow computations. Confidential virtual machines preserve compatibility by protecting an entire guest, which eases migration of existing workloads but enlarges the trusted computing base to include a full guest operating system, so a vulnerability in that guest is inside the boundary. Deployment context, legacy code, memory footprint, and the acceptable trust boundary usually decide the outcome, and hybrid patterns that run an enclave inside a confidential virtual machine are emerging to combine the strengths of both.

Table 1. Representative comparison of major TEE technologies. Details vary by hardware generation and firmware.

Technology	Granularity	Memory protection	Attestation	Vendor
Intel SGX	Process enclave	Encryption; integrity in early generations	Remote (EPID / DCAP)	Intel
AMD SEV-SNP	Confidential VM	Encryption and integrity, anti-remap	Remote attestation report	AMD
Intel TDX	Confidential VM (trust domain)	Encryption and integrity	Remote quote via TDX module	Intel
Arm TrustZone	Secure world (two worlds)	Isolation; encryption platform dependent	Vendor and platform dependent	Arm
Arm CCA (Realms)	Confidential VM (isolated Realm)	Encryption and integrity	Realm attestation token	Arm
RISC-V Keystone	Customizable enclave	PMP isolation; encryption optional	Monitor-based attestation	Open source

IV. REMOTE ATTESTATION AND ROOT OF TRUST

Isolation alone is not sufficient for a relying party that never touches the machine. That party must be convinced that a genuine TEE, running the expected code in an approved configuration, is at the other end of the connection before it entrusts any secret. Remote attestation provides this assurance. A hardware root of trust measures the initial contents and configuration of the protected environment, and the platform produces signed evidence, often called a quote or attestation report, that binds this measurement to a key rooted in the silicon vendor. The relying party checks that the measurement matches an approved value and that the signature chains to a trusted manufacturer before it proceeds.

A verifier appraises the evidence against a policy and, if satisfied, allows the relying party to continue, typically by releasing a decryption key or by establishing a secure channel terminated inside the enclave, so that plaintext appears only within the protected boundary. Intel deployments have used an enhanced privacy identifier scheme that preserves platform anonymity and, later, a data center attestation model that lets an operator run its own verification service without contacting the vendor for every request. AMD SEV-SNP and Arm CCA issue their own signed reports and tokens for the protected guest. To harmonize these vendor-specific flows, the Internet Engineering Task Force published an architecture for remote attestation procedures that names the common roles of attester, verifier, and relying party and defines how evidence and appraisal results move between them [13]. This standardized vocabulary matters because real deployments increasingly mix hardware from several vendors, and interoperable attestation is what lets a single policy engine reason about them uniformly.

Attestation is also a point of fragility, and an honest treatment must say so. If an attacker can extract the platform attestation key, it can forge quotes for environments that are not genuine, undermining every guarantee that depends on attestation, because a relying party has no other way to distinguish a real TEE from an emulated one. This risk is not hypothetical, as later side-channel results demonstrate, and it is one reason the security analysis in this survey treats key extraction as a first-class concern and why vendors provision fresh keys and revoke compromised ones after disclosure.

V. APPLICATIONS

Confidential computing has moved from research prototypes to production offerings across several domains. In each case the value proposition is the same: sensitive data can be processed on infrastructure the data owner does not fully trust, with hardware isolation and attestation standing in for physical control, and often with a written record of exactly which code touched the data.

Confidential artificial intelligence is a prominent driver. Machine learning training and inference often combine a proprietary model with sensitive input data, and TEEs allow both to be processed without exposing either to the host or to the other party. Early work demonstrated data-oblivious training and inference for common algorithms inside enclaves, taking care to eliminate secret-dependent memory access patterns that side channels could otherwise observe, since naive code inside an enclave can still leak through its access trace [14]. A later systematization of knowledge surveys the design space of machine learning with confidential computing, cataloguing where protection is placed, which parts of the pipeline are covered, and what residual risks remain when only part of a workload is enclosed [15]. Because model sizes and memory footprints frequently exceed the practical limits of process enclaves, confidential virtual machines and confidential accelerators have become the preferred substrate for this workload.

Privacy-preserving analytics is a second established application. Systems have shown that distributed data-processing frameworks can run with their computation isolated inside enclaves, keeping input records, intermediate state, and results confidential from the cloud platform while still producing verifiable outputs, and while keeping the untrusted framework, operating system, and hypervisor outside the trusted computing base [16]. Related deployments cover encrypted databases and query processing, joint analytics across organizations that cannot share raw data, and regulated reporting pipelines where an auditor needs assurance that the correct computation ran. Key management and secret handling form a third domain: hardware security module functionality, certificate authorities, and wallet or signing services benefit directly from an environment where private keys never appear in plaintext outside the TEE, even to the machine administrator.

Blockchain and distributed ledger systems use TEEs to build trusted oracles that feed authenticated external data to smart contracts, to run confidential smart-contract execution where transaction contents stay hidden, and to accelerate consensus by trusting a hardware-attested component instead of an expensive protocol. TEEs are also used to accelerate or complement multi-party computation. A hardware-isolated environment can shoulder work that would otherwise require costly cryptographic exchanges among the parties, for example by having each participant submit encrypted inputs that are decrypted and combined only inside an attested enclave, trading a purely cryptographic trust assumption for a hardware one where the performance budget demands it.

The extension of confidential computing to accelerators is the most consequential recent development for these applications. The NVIDIA H100 introduced hardware-enforced confidential computing on a graphics processing unit, encrypting data transferred between the central processor TEE and the accelerator over the bus and providing measured, attested boot of the device so that a confidential virtual machine can extend its trust boundary onto the accelerator [17]. Independent benchmarking of confidential computing on Hopper-class GPUs reports that the overhead, while workload dependent, becomes small for large models where compute dominates data movement [18]. This matters because it

removes the practical ceiling that kept large-scale confidential machine learning tied to central processors, and it makes confidential training and inference of large models economically realistic.

VI. SECURITY ANALYSIS AND SIDE-CHANNEL ATTACKS

An honest survey must weigh the guarantees against a substantial body of attacks. Hardware isolation blocks direct reads of protected memory, but the microarchitecture shared beneath the isolation boundary leaks information through timing, caches, speculation, and even power delivery. These channels do not read ciphertext; they infer secrets from the side effects of computation, which encryption does not hide. Table 2 summarizes several notable attacks together with the mitigations that followed.

Transient-execution attacks have been the most damaging class. Foreshadow, also tracked as L1 Terminal Fault, showed that speculatively executed instructions could read enclave data resident in the L1 cache and exfiltrate it through a covert channel, and the same flaw could be used to extract SGX attestation keys, undermining the very trust that attestation is supposed to establish [19]. A systematic evaluation later organized the broader family of Meltdown-type and Spectre-type transient attacks and their defenses, showing that the attack surface is structural rather than a single isolated defect and that many variants share a common exploitation pattern [20]. SGAxe extended this line by combining a transient-execution data leak with the attestation problem to recover a machine's attestation keys, allowing an attacker to sign fraudulent quotes that the vendor service would accept as genuine [21]. Because attestation is the linchpin of remote trust, key extraction is an especially severe outcome, since it lets an adversary impersonate any number of legitimate platforms.

Fault-injection attacks form a second class. Plundervolt demonstrated that a privileged attacker could abuse an undocumented voltage-scaling interface to induce computation faults inside an SGX enclave, corrupting the integrity of results and, in some cases, exposing cryptographic keys, all through software and with no physical access to the machine [22]. Alongside these, the controlled-channel technique noted earlier remains a reliable way for a malicious operating system to infer secret-dependent behavior by manipulating paging and observing the resulting faults [7], and classical cache and branch-prediction side channels continue to threaten code whose control flow or memory access depends on secrets.

The honest conclusion is that TEEs raise the bar substantially against direct memory disclosure but do not by themselves defeat microarchitectural leakage. Mitigations have arrived through microcode updates, hardware changes such as flushing buffers on enclave exit, disabling simultaneous multithreading for sensitive workloads, redesigned attestation key provisioning, and defensive coding that removes secret-dependent memory and timing behavior. A defense-in-depth posture that combines these with oblivious algorithms is therefore prudent rather than optional. Many mitigations carry a performance cost, and several attacks were addressed only after public disclosure, so a careful deployment keeps microcode current, constrains its threat model to a specific hardware generation, and does not assume that any single side channel is the last one to be found.

Table 2. Notable side-channel and fault attacks against TEEs and representative mitigations.

Attack	Target and mechanism	Representative mitigation
Controlled channel	Untrusted OS observes page-fault patterns	Oblivious code and data access, self-paging
Foreshadow (L1TF)	Transient read of enclave data from L1 cache	Microcode, L1 flush on exit, disable SMT
SGAxe	Attestation-key recovery via transient leak	Attestation key re-provisioning, microcode
Plundervolt	Software voltage-scaling fault injection in SGX	Disable voltage interface, firmware fix
Transient-execution family	Speculation-based cross-domain leakage	Speculation barriers, buffer flushing, isolation

VII. PERFORMANCE OVERHEADS

Performance is the other axis on which confidential computing is judged, and the picture is nuanced. Overhead depends on the platform, the hardware generation, and above all the workload. Compute-bound tasks that fit within the protected memory and cross the TEE boundary rarely tend to see modest overhead, while memory-bound and input-output-intensive tasks, and workloads that make frequent boundary transitions, pay considerably more. Process enclaves that exceed the protected memory capacity suffer from paging costs as encrypted pages are evicted and reloaded, and confidential virtual machines add a per-transition cost when the guest interacts with untrusted devices or issues calls that must exit the protected domain.

Figure 1 shows representative reported overhead across a spread of workload types, drawn from the qualitative pattern seen across published measurements rather than from any single system. Figure 2 contrasts two confidential virtual machine platforms across workload categories, again as representative reported values rather than a controlled head-to-head benchmark. The consistent message is that no single overhead number characterizes the field; a memory-bound analytics job and a compute-bound inference job on the same hardware can differ by an order of magnitude, and the same

code can move between those regimes as its working set grows. Confidential graphics processing units follow the same logic, with reported overhead shrinking as model size grows and computation comes to dominate the cost of encrypted data transfer across the bus [18]. For capacity planning, the only sound approach is to measure the specific workload on the specific platform generation and to avoid extrapolating from a headline figure.

Fig 1: Representative reported performance overhead across workload types. Values are illustrative of published ranges, not a single measurement.

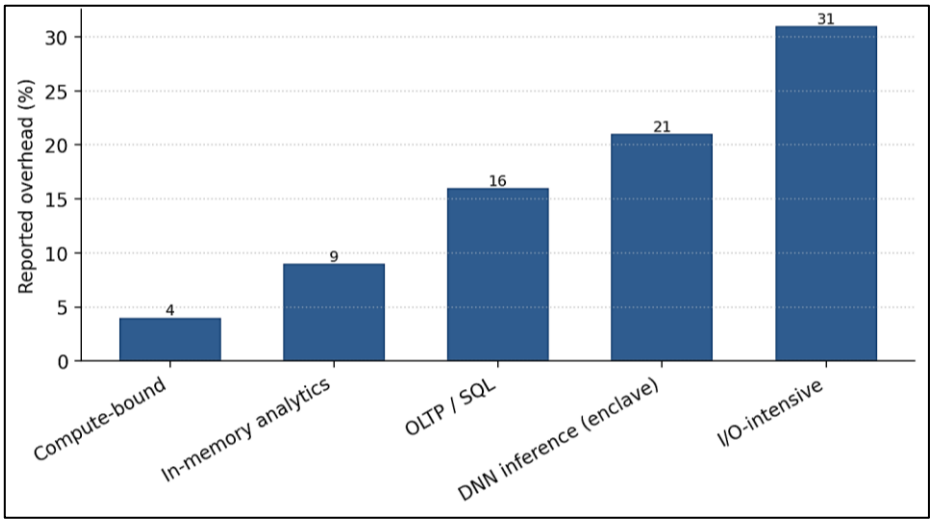
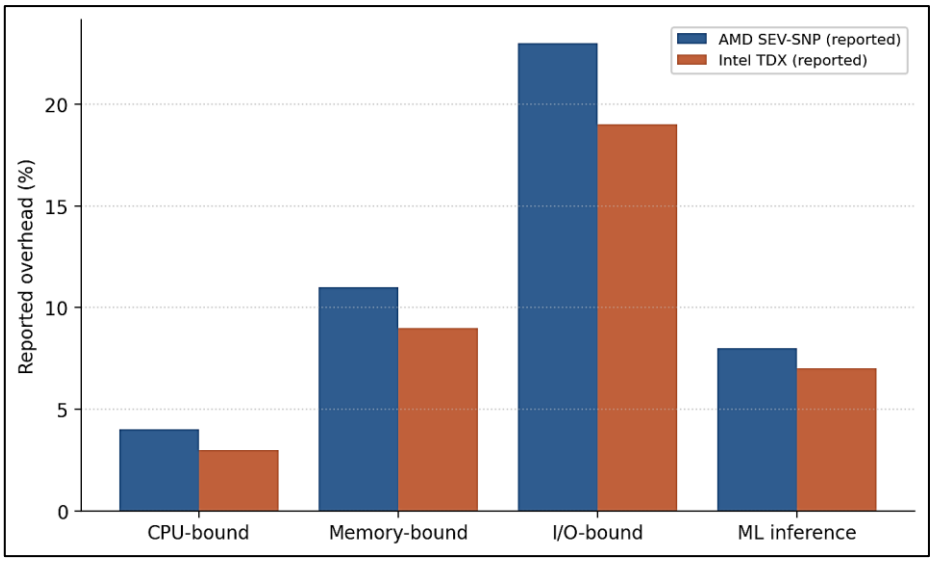


Fig 2: Representative reported overhead for two confidential virtual machine platforms by workload category. Illustrative values, not a controlled benchmark.



VIII. RELATION TO OTHER PETs, STANDARDIZATION, AND FUTURE DIRECTIONS

Confidential computing sits within a broader family of privacy-enhancing technologies, and it is best understood as a complement to them rather than a competitor. Homomorphic encryption allows computation directly on ciphertext and offers strong cryptographic guarantees with no hardware trust assumption, but it remains costly for general workloads and is practical mainly for restricted operations. Secure multi-party computation lets several parties jointly compute a function without revealing their inputs, again with a cryptographic rather than a hardware trust basis, at the price of communication rounds and coordination overhead that grow with the parties and the function. Differential privacy protects the outputs of an analysis by bounding what any individual record can contribute, which is a different and orthogonal guarantee about disclosure in results rather than about isolation during computation.

These techniques combine naturally with TEEs, and the strongest systems mix them. A hardware-isolated environment can accelerate the expensive stages of a multi-party computation, host the aggregation step of a distributed protocol, or enforce a differential-privacy mechanism on results before they leave the boundary, so that a compromise of the surrounding platform reveals neither the inputs nor an unbounded view of the outputs. In federated learning, for example, a TEE at the aggregator can protect model updates while a differential-privacy mechanism bounds leakage from the released model, a pattern that is particularly valuable in the healthcare setting where patient data cannot be centralized and both the raw records and the shared model carry disclosure risk [23]. The design choice among these tools depends

on the required trust assumption, the acceptable overhead, and whether the concern is confidentiality during computation, correctness of a joint result, or bounded disclosure in an output.

Standardization is maturing in parallel. The Confidential Computing Consortium works to align terminology and to compare offerings on a common basis [2], while the remote-attestation architecture from the Internet Engineering Task Force gives interoperable roles and evidence flows across vendors [13]. Critical scholarship continues to press for precise, verifiable definitions and fair comparisons, which strengthens the field rather than weakening it [4]. A further standardization concern is cryptographic longevity: the device keys and signatures that anchor attestation and memory protection rely on public-key primitives that must migrate to post-quantum algorithms over the coming decade, and the standardization, algorithm selection, and deployment path for that migration is itself an active area of study whose outcome will shape the roots of trust used by every TEE [24].

Looking ahead, several directions stand out. Confidential accelerators, led by confidential graphics processing units, extend the trust boundary to the hardware where large-scale artificial intelligence actually runs [17], [18]. Confidential containers and managed platform services aim to lower the operational barrier so that developers gain protection without partitioning applications by hand. Verified security monitors and formally analyzed architectures, as pursued for Arm CCA [11] and open frameworks such as Keystone [12], promise smaller and more trustworthy foundations. Composable, cross-vendor attestation, resistance to the next transient-execution attack rather than the last one, and clear, honest communication of what the guarantees do and do not cover remain the hardest and most important open problems.

IX. CONCLUSION

Confidential computing closes a long-standing gap by protecting data while it is in use, using hardware-based, attested Trusted Execution Environments so that a workload can run correctly even on infrastructure its owner does not control. The technology landscape now spans process enclaves such as Intel SGX, confidential virtual machines including Intel TDX, AMD SEV-SNP, and Arm CCA, the widely deployed Arm TrustZone, and open designs such as RISC-V Keystone, unified by remote attestation rooted in silicon and increasingly harmonized by cross-vendor standards.

The honest assessment is one of strong but bounded guarantees. Hardware isolation defeats direct disclosure of protected memory, yet side-channel and transient-execution attacks such as Foreshadow, Plundervolt, and SGAxe show that the shared microarchitecture beneath the boundary continues to leak, and that attestation keys are a high-value target whose loss is catastrophic. Overheads are real and highly workload dependent, so deployments should measure rather than assume. Positioned as a complement to homomorphic encryption, secure multi-party computation, and differential privacy, extended onto confidential accelerators, and supported by maturing standards for attestation and a migration path toward post-quantum cryptography, confidential computing is a durable and increasingly central building block for trustworthy computation on untrusted infrastructure.

REFERENCES

[1] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not," in Proc. IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, 2015, pp. 57–64.

[2] Confidential Computing Consortium, "A Technical Analysis of Confidential Computing," Whitepaper, version 1.3, 2023.

[3] D. P. Mulligan, G. Petri, N. Spinale, G. Stockwell, and H. J. M. Vincent, "Confidential computing: A brave new world," in Proc. IEEE Int. Symp. Secure and Private Execution Environment Design (SEED), 2021, pp. 132–138.

[4] M. U. Sardar and C. Fetzer, "Confidential computing and related technologies: A critical review," *Cybersecurity*, vol. 6, no. 10, 2023.

[5] V. Costan and S. Devadas, "Intel SGX Explained," Cryptology ePrint Archive, Rep. 2016/086, 2016.

[6] F. McKeen *et al*., "Innovative instructions and software model for isolated execution," in Proc. 2nd Int. Workshop Hardware and Architectural Support for Security and Privacy (HASP), 2013.

[7] Y. Xu, W. Cui, and M. Peinado, "Controlled-channel attacks: Deterministic side channels for untrusted operating systems," in Proc. IEEE Symp. Security and Privacy (S&P), 2015, pp. 640–656.

[8] AMD, "AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More," White Paper, 2020.

[9] P.-C. Cheng *et al*., "Intel TDX Demystified: A Top-Down Approach," *ACM Computing Surveys*, vol. 56, no. 9, pp. 1–33, 2024.

[10] S. Pinto and N. Santos, "Demystifying Arm TrustZone: A comprehensive survey," *ACM Computing Surveys*, vol. 51, no. 6, Art. no. 130, 2019.

[11] X. Li *et al*., "Design and Verification of the Arm Confidential Compute Architecture," in Proc. 16th USENIX Symp. Operating Systems Design and Implementation (OSDI), 2022, pp. 465–484.

[12] D. Lee, D. Kohlbrenner, S. Shinde, K. Asanovic, and D. Song, "Keystone: An open framework for architecting trusted execution environments," in Proc. 15th European Conf. Computer Systems (EuroSys), 2020.

[13] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan, "Remote ATtestation procedureS (RATS) Architecture," RFC 9334, IETF, Jan. 2023.

[14] O. Ohrimenko *et al*., "Oblivious multi-party machine learning on trusted processors," in Proc. 25th USENIX Security Symp., 2016, pp. 619–636.

[15] F. Mo, Z. Tarkhani, and H. Haddadi, "Machine learning with confidential computing: A systematization of knowledge," *ACM Computing Surveys*, vol. 56, no. 11, 2024.

[16] F. Schuster *et al*., "VC3: Trustworthy data analytics in the cloud using SGX," in Proc. IEEE Symp. Security and Privacy (S&P), 2015, pp. 38–54.

[17] NVIDIA, "Confidential Compute on NVIDIA Hopper H100," Technical White Paper WP-11459-001, 2023.

[18] J. Mohan *et al*., "Confidential Computing on NVIDIA Hopper GPUs: A Performance Benchmark Study," arXiv:2409.03992, 2024.

[19] J. Van Bulck *et al*., "Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution," in Proc. 27th USENIX Security Symp., 2018, pp. 991–1008.

[20] C. Canella *et al*., "A systematic evaluation of transient execution attacks and defenses," in Proc. 28th USENIX Security Symp., 2019, pp. 249–266.

[21] S. van Schaik, A. Kwong, D. Genkin, and Y. Yarom, "SGAxe: How SGX Fails in Practice," Technical Report, 2020.

[22] K. Murdock, D. Oswald, F. D. Garcia, J. Van Bulck, D. Gruss, and F. Piessens, "Plundervolt: Software-based fault injection attacks against Intel SGX," in Proc. IEEE Symp. Security and Privacy (S&P), 2020, pp. 1466–1482.

[23] "Federated Learning for Privacy-Preserving Healthcare Data Analytics," *Peer-Reviewed Journal of Computer Science (PRJCS)*, E-ISSN 3139-5082, 2025.

[24] "Post-Quantum Cryptography: Standardisation, Algorithms, and Deployment Migration," *Peer-Reviewed Journal of Computer Science (PRJCS)*, E-ISSN 3139-5082, 2025.