Confidential Computing in Cloud Computing: Architectures, Technologies, and Challenges

Zixu Wang

Waterford

Abstract. Cloud computing has reshaped how organizations provision and consume computing resources, moving many sensitive workloads from trusted on-premises environments to shared, provider-managed infrastructure. Although encryption protects data at rest and in transit, information is often exposed while being processed, creating risk from insiders and compromised privileged components. Confidential Computing addresses this "data-in-use" gap by using hardware-based Trusted Execution Environments (TEEs) to isolate code and data during execution, reducing the need to trust the cloud operator.

This paper surveys Confidential Computing in cloud settings. It introduces key concepts and security properties, outlines common system architectures, and explains enabling hardware mechanisms and the role of remote attestation in verifying trusted execution before releasing secrets. It then discusses how confidential execution is integrated into real cloud operations, and evaluates representative use cases, benefits, limitations, and open challenges. The aim is to provide an accessible overview for readers with a general cloud background while highlighting practical trade-offs and directions for future research and adoption.

Keywords: Cloud Computing · Confidential Computing · Trusted Execution Environment · Cloud Security · Remote Attestation

1 Introduction

1.1 Background

Cloud computing has become the dominant paradigm for delivering computing infrastructure and services, offering scalability, elasticity, and cost efficiency that are difficult to achieve with traditional on-premises systems. By abstracting physical resources and enabling on-demand provisioning, cloud platforms allow organizations to rapidly deploy applications and scale workloads according to demand. This flexibility has accelerated digital transformation across industries and has made cloud computing an essential component of modern information systems.

As cloud adoption has matured, the nature of cloud workloads has evolved significantly. Early cloud deployments primarily hosted non-critical applications

and public-facing services. In contrast, contemporary cloud environments increasingly process sensitive and mission-critical data, including personally identifiable information, financial transactions, medical records, and proprietary algorithms. This shift has elevated security and privacy from secondary concerns to central design requirements.

Protecting confidentiality in cloud environments is therefore a critical challenge. Conventional mechanisms emphasize encryption of data at rest and secure transmission of data over networks. These techniques provide strong protection against many external attackers, but they do not address exposure during execution. Once encrypted data is loaded into memory for processing, it is typically accessible to privileged software components such as the operating system, hypervisor, firmware, and management tools. From a tenant perspective, the cloud platform may be operationally trusted but still not acceptable as a security principal that can read sensitive workloads.

1.2 Motivation

The traditional cloud trust model assumes that infrastructure providers operate their systems correctly and securely. However, academic research and operational experience show that privileged components can be compromised through vulnerabilities, misconfigurations, and supply-chain issues. Insider threats are also a concern: even if access is audited, the mere possibility that administrators can read plaintext data may violate internal governance or legal requirements.

These risks are particularly significant in domains such as healthcare, finance, and government services, where compliance obligations often require strict control over data access. They are also relevant in collaborative analytics among multiple organizations, where participants may be willing to share results but not raw inputs. In such environments, it is desirable to reduce reliance on trust in the cloud operator and instead depend on stronger, verifiable execution guarantees.

1.3 Problem Statement

The core problem addressed in this work is how to enable secure computation in cloud environments while minimizing reliance on the trustworthiness of the underlying infrastructure. Specifically, the challenge is to protect data confidentiality during execution in a way that remains scalable, compatible with existing cloud architectures, and practical for real-world workloads that depend on operating systems, libraries, networking, and storage.

This problem is not solved by encryption at rest or in transit alone. The key gap is “data in use”: plaintext values must exist in CPU registers and memory while code executes, and privileged infrastructure software may have the ability to observe or manipulate those states. Confidential Computing aims to narrow this gap with hardware-enforced isolation and verifiable boot and launch processes.

1.4 Contributions

This paper makes three primary contributions. First, it provides a structured overview of Confidential Computing and situates it within the broader landscape of cloud security. Second, it analyzes architectural principles and hardware mechanisms that enable confidential execution, including TEEs and confidential virtual machines. Third, it discusses remote attestation, operational integration, application scenarios, and remaining challenges, offering guidance relevant to both researchers and practitioners.

2 Preliminaries and Security Properties

2.1 Data Protection States

Cloud security is often summarized using three protection states: data *at rest* (stored in disks or object storage), data *in transit* (moving over networks), and data *in use* (being processed). Encryption at rest and in transit is widely deployed and supported by mature tools such as TLS, disk encryption, and managed key services. The less-addressed state is data in use, where plaintext must appear in memory and CPU registers to be processed.

Confidential Computing focuses on protecting data in use. The main idea is that even if a malicious party controls privileged infrastructure software, they should not be able to directly read sensitive memory contents, extract keys, or observe intermediate computations in a straightforward way.

2.2 Trusted Computing Base

The **Trusted Computing Base (TCB)** is the set of components that must be trusted for a security claim to hold. In conventional cloud deployments, tenants effectively trust the hypervisor, platform firmware, and parts of the management plane for isolation and confidentiality. Confidential Computing tries to shrink and reshape this trust: the platform is expected to schedule and manage workloads, but not to read their protected memory. In practice, trust shifts toward the CPU package, microcode, and small security-critical services that produce and verify attestation evidence.

A smaller TCB is generally desirable, but it does not eliminate trust entirely. The practical question becomes whether the remaining trusted components are easier to validate, patch, and reason about than the broader software stack of a full cloud platform.

2.3 Security Goals and Scope

A typical Confidential Computing system targets the following goals:

- **Memory confidentiality**: preventing unauthorized reads of protected workload memory.

- **Isolation**: strengthening separation between tenants and between host and guest.
- **Verifiability**: enabling remote parties to verify that a workload is running under expected protections before releasing secrets.

Most systems explicitly exclude some threats, such as denial-of-service by a malicious operator, certain physical attacks, and some categories of side-channel leakage. Clear scoping matters because it determines what guarantees are realistic and what additional controls are required.

3 Literature Review

3.1 Traditional Cloud Security Models

Traditional cloud security relies on a combination of virtualization isolation, access control, monitoring, and the shared responsibility model. Hypervisors and container runtimes provide tenant separation; identity and access management limits access to resources; logging and monitoring detect abuse. These mechanisms establish a strong baseline for many workloads.

However, the baseline assumes privileged components are trustworthy. Vulnerabilities in hypervisors, firmware, and management services can undermine isolation and confidentiality. Even without exploitation, privileged administrators may have legitimate access to debugging interfaces and operational tooling that can expose data. As a result, traditional security models may not satisfy workloads that require protection from the infrastructure itself.

3.2 Trusted Execution Environments

Trusted Execution Environments (TEEs) provide isolated execution contexts that protect code and data from unauthorized access, even if the operating system is compromised. Intel SGX is a well-known example of enclave-based TEEs [1]. Research systems demonstrate how TEEs can support secure cloud analytics [2] and privacy-preserving machine learning services [3]. Practical deployment also motivated runtimes that support more general applications, such as secure container-like execution within enclaves [4].

TEEs introduce trade-offs. Strong isolation often comes with constraints on memory usage, system call interfaces, and debugging. These limitations motivate a broader approach where Confidential Computing is integrated into cloud architectures to balance security with deployability at scale.

3.3 Evolution of Confidential Computing

Confidential Computing extends TEE concepts to cloud-scale infrastructures. Instead of protecting only a small enclave within an application, modern designs also protect entire virtual machines using hardware-backed memory encryption. Industry adoption has accelerated support for remote attestation, integration with key management, and managed offerings that allow tenants to request confidential instances through standard cloud provisioning workflows.

3.4 Research Gap

Many studies focus on a single mechanism (e.g., a particular enclave design) or a narrow threat model. For practical adoption, users need an end-to-end view: how hardware protections interact with virtualization, how attestation gates secrets, how operational tooling changes, and what residual risks remain. A system-level perspective is therefore valuable for decision-making.

4 System Architecture

4.1 High-Level Architecture

A Confidential Computing system spans multiple layers of the cloud stack. At the application layer, workloads run normally but are placed inside a protected boundary (an enclave or a confidential VM). The guest operating system and runtime still perform common functions such as scheduling, memory allocation, and network stack operations. The key difference is that the host and hypervisor are restricted from inspecting protected memory, and the platform provides mechanisms for tenants to verify protections.

At a high level, confidential execution introduces an additional isolation boundary between tenant workloads and privileged infrastructure components. This boundary is enforced by hardware, with supporting software that provisions confidential instances, establishes keys, and provides attestation evidence.

4.2 Hardware Foundations

Modern processors incorporate hardware mechanisms that enable confidential execution, including transparent memory encryption, isolated execution regions, and secure key storage. Memory encryption is central: it aims to keep DRAM contents unintelligible outside the CPU package, so that reading raw memory pages does not reveal plaintext data.

Some designs also provide integrity features to detect tampering, while others primarily emphasize confidentiality against reads. Regardless of the exact mechanism, key material is typically generated and managed by hardware so that the host cannot directly obtain the keys used to protect guest memory.

4.3 Cloud Integration

Cloud providers integrate Confidential Computing by offering specialized VM types or execution environments that enable hardware protections. These offerings are often paired with remote attestation mechanisms so tenants can validate that the expected protections are active. Integration is designed to fit into existing cloud patterns: users select an instance type, deploy images, and then use attestation-based secret provisioning for sensitive configuration.

From an operational perspective, this integration must be compatible with orchestration, autoscaling, logging, identity systems, and storage services. The challenge is to improve confidentiality without breaking the usability benefits that made cloud adoption attractive in the first place.

4.4 Threat Model

The threat model typically assumes that the cloud operator and host software stack (including the hypervisor) may be untrusted or compromised. The main goal is to prevent this privileged environment from reading tenant secrets during execution. Physical attacks and denial-of-service are often out of scope. Side channels may be partially addressed, but they remain a significant concern (discussed later).

5 Hardware Technologies and Design Choices

5.1 Enclaves vs. Confidential Virtual Machines

Two common approaches are:

- **Enclave-based TEEs**: protect an application component inside an enclave. This can reduce the amount of trusted code but may require adapting applications and handling I/O carefully. Intel SGX is a representative example [1].
- **Confidential VMs**: protect an entire VM so that the host cannot read guest memory. This often supports unmodified workloads more easily, at the cost of trusting a larger software stack inside the guest. AMD SEV and related mechanisms are widely discussed in this category [5], as are Intel TDX-style confidential VM approaches [6].

5.2 Practical Implications

In practice, the choice depends on workload characteristics:

- For applications where only a small part handles secrets (e.g., key handling, sensitive parsing), enclave approaches can isolate that component with a smaller boundary.
- For legacy systems or full-stack services that are difficult to refactor, confidential VMs can provide protection with fewer code changes.

Both approaches rely on correct hardware behavior and require strong attestation to ensure the environment is configured as intended. Both also need careful engineering to avoid leaking secrets through logs, crash dumps, or application-layer data flows.

6 Remote Attestation and Key Management

6.1 Role of Remote Attestation

Remote attestation enables a tenant (or a tenant-controlled service) to verify the security properties of a confidential environment before releasing secrets. Without attestation, a user cannot reliably distinguish a truly protected instance from a normal instance that could be inspected by the host.

A typical goal is: *secrets should only be provisioned if the workload proves it is running in an approved confidential configuration.* This turns confidentiality into an enforceable operational policy rather than a vague promise.

6.2 Attestation Workflow

A simplified workflow includes:

1. **Evidence generation**: the platform measures relevant state (e.g., firmware configuration, launch measurements, TEE mode) and produces signed evidence.
2. **Verification**: the tenant verifies evidence using trusted verification logic and checks a policy (expected measurements, required protections enabled).
3. **Secret release**: secrets are delivered only after verification succeeds, typically over a secure channel to the protected workload.

This pattern is essential for confidential deployments that involve keys for encrypted storage, API credentials, or proprietary model parameters.

6.3 Integration with KMS and Secret Managers

Key management integration is often the practical "core" of Confidential Computing. A confidential workload may need disk keys, database credentials, or service tokens at startup. If those secrets are injected through standard mechanisms without attestation, the trust model collapses. Therefore, confidential deployments commonly combine:

- attestation-based access control to secrets,
- encryption at rest with customer-managed keys,
- identity-based authorization to ensure only approved workloads can request secrets.

This also supports auditability: policies can be logged and reviewed to show that secrets are released only to verified confidential environments.

7 Use Cases and Application Scenarios

7.1 Privacy-Sensitive Data Processing

Confidential Computing enables secure processing of sensitive data in regulated domains such as healthcare, finance, and government services. Organizations can move analytics workloads to the cloud while reducing the risk that privileged infrastructure components can read plaintext data during processing.

Common patterns include confidential ETL pipelines, secure aggregation, and encrypted data lakes where decryption occurs only within protected execution. This can simplify compliance arguments, particularly when organizations must demonstrate controls against insider access.

7.2 Machine Learning Workloads

Machine learning introduces confidentiality concerns for both data and models. Training data may contain personal records, while model weights and architectures may be proprietary. Confidential execution can protect:

- **Inference inputs**: user queries and features that may be sensitive.
- **Model parameters**: weights that represent intellectual property.

For inference services, confidential endpoints can provide stronger privacy guarantees, especially when combined with attestation so clients can verify they are sending data to a protected environment.

7.3 Multi-Party Collaboration

In collaborative scenarios involving multiple organizations, Confidential Computing can support joint computation while preserving data confidentiality. Parties can contribute data to a shared computation without revealing raw inputs to the operator. Compared to cryptographic approaches such as MPC or fully homomorphic encryption, Confidential Computing can be easier to deploy and often more efficient, but it relies on hardware trust assumptions and remains exposed to certain side channels.

7.4 Comparison with Traditional Cloud Deployments

Compared to traditional VMs and containers, confidential workloads reduce trust in privileged infrastructure components. However, they introduce:

- additional setup (attestation policies, secret provisioning),
- potential performance overhead from memory encryption and restricted operations,
- reduced visibility for debugging and forensics.

These trade-offs suggest that confidential execution is most valuable when the workload sensitivity justifies the added complexity.

8 Discussion

8.1 Benefits

Confidential Computing enhances trust in cloud platforms by reducing reliance on infrastructure-level trust assumptions. It enables new application scenarios, supports compliance with strict regulatory requirements, and strengthens defenses against insider threats. It also encourages better secret handling practices because secret provisioning becomes explicitly tied to verified execution properties.

8.2 Operational Trade-offs

Confidential Computing changes day-to-day operations. Debugging becomes harder because memory inspection and snapshots may be restricted. Monitoring may shift toward in-guest telemetry rather than hypervisor-level introspection. Incident response and forensics may require new procedures that balance confidentiality with investigatory needs. For many organizations, successful adoption depends on building robust CI/CD and policy workflows rather than treating confidential instances as a simple VM toggle.

9 Limitations and Challenges

9.1 Performance Overhead

Memory encryption and additional protection checks can impose overhead. The magnitude depends on workload characteristics. Compute-bound workloads may see modest impact, while memory-bound workloads can experience higher overhead due to encrypted memory traffic and protection metadata. Overhead can sometimes be reduced by isolating only the most sensitive components or using confidential VMs selectively for specific tiers (e.g., key services or inference gateways).

9.2 Side-Channel Attacks

Confidential Computing improves isolation but does not fully prevent side-channel attacks. Cache timing, branch prediction, and other microarchitectural behaviors can leak information in some threat models. Mitigations include constant-time implementations, limiting co-tenancy, careful scheduling, and adopting hardened libraries, but side-channel resistance remains an active and challenging area.

9.3 Usability and Tooling

Deploying confidential applications may require specialized expertise:

- writing and maintaining attestation policies,
- managing trusted image measurements and secure boot configurations,
- ensuring logs, metrics, and crash dumps do not leak secrets,
- handling new failure modes related to attestation and key release.

These factors can slow adoption, particularly in smaller organizations without dedicated security engineering teams.

9.4 Portability and Standardization

The ecosystem is fragmented across hardware vendors and cloud providers. Differences in attestation formats, policy APIs, and supported features can complicate portability. Standardization efforts that define common evidence formats and verification flows would reduce integration cost and enable broader tooling reuse across platforms.

10 Adoption Guidelines

10.1 Selecting Candidate Workloads

A practical adoption strategy is to start with workloads that have clear confidentiality requirements and relatively bounded operational complexity. Typical candidates include key management-related services, regulated analytics pipelines, and proprietary ML inference endpoints. Workloads that require extensive hypervisor-level observability or extremely low latency may be better suited for later phases or for hybrid approaches.

10.2 Policy-Driven Secret Provisioning

The most important operational principle is to tie secret release to attestation verification. Organizations should define clear policies describing what must be true for a workload to receive secrets (e.g., confidential mode enabled, approved image hash, secure boot required). Implementing this in an automated deployment pipeline reduces human error and improves auditability.

10.3 Defense in Depth

Confidential Computing should complement, not replace, existing cloud security controls. Encryption at rest and in transit, least-privilege IAM, network segmentation, and secure software development remain essential. Confidential execution primarily reduces one class of risk: privileged infrastructure access to workload plaintext during computation.

11 Conclusion and Future Work

11.1 Conclusion

Confidential Computing represents a significant advancement in cloud security by extending protection to data during computation. By leveraging hardware-based isolation, it addresses a fundamental limitation of traditional cloud security models that focus mainly on data at rest and in transit. This paper reviewed key concepts, architectures, and hardware foundations; explained the central role of remote attestation and key management; and analyzed application scenarios and trade-offs. Overall, Confidential Computing is most valuable when organizations require stronger guarantees against insider threats or infrastructure compromise and are prepared to manage the operational changes introduced by attestation-based workflows.

11.2 Future Work

Future research and engineering efforts may focus on reducing overhead for memory-intensive workloads, strengthening defenses against microarchitectural side channels, improving developer tooling and debugging support, and increasing portability through standardization. Another promising direction is integrating Confidential Computing into broader security paradigms such as zero trust architectures, where continuous verification and policy enforcement become default assumptions for cloud operations.

References

1. Costan, V., Devadas, S.: Intel SGX Explained. *IACR Cryptology ePrint Archive*, 2016.
2. Schuster, F., et al.: VC3: Trustworthy Data Analytics in the Cloud Using SGX. *IEEE Symposium on Security and Privacy*, 2015.
3. Hunt, T., et al.: Chiron: Privacy-Preserving Machine Learning as a Service. *USENIX Security*, 2018.
4. Arnautov, S., et al.: SCONE: Secure Linux Containers with Intel SGX. *USENIX OSDI*, 2016.
5. AMD: Secure Encrypted Virtualization (SEV). Whitepaper, 2020.
6. Intel: Trust Domain Extensions (TDX). Technical Documentation, 2022.
7. Microsoft Azure: Confidential Computing Documentation, 2023.
8. Google Cloud: Confidential Computing Overview, 2023.
9. Amazon Web Services: Nitro Enclaves, 2023.