Confidential Computing in Public Cloud: Architectures for Privacy-Preserving Workloads

Ritesh Kumar

Independent Researcher
Pennsylvania, USA
ritesh2901@gmail.com

Abstract

Confidential computing is rapidly emerging as a critical technology for enabling privacy-preserving workloads in public cloud environments. By leveraging hardware-based trusted execution environments (TEEs), sensitive data can remain encrypted in memory during processing, significantly mitigating risks associated with cloud provider access, malicious insiders, and advanced persistent threats. This paper explores architectural patterns and design considerations for deploying privacy-sensitive applications utilizing confidential computing capabilities offered by major public cloud providers. We analyze various approaches for integrating TEEs into cloud-native frameworks, addressing key challenges related to data ingress/egress, attestation, key management, and the orchestration of confidential workloads. Furthermore, we discuss the trade-offs associated with performance, compatibility, and application lifecycle management. The proposed architectural blueprints aim to provide a practical foundation for technical architects designing systems that demand strong data confidentiality guarantees while harnessing the scalability and flexibility of the public cloud.

Keywords: Architectural Patterns, Attestation, Cloud Architecture, Confidential Computing, Data Confidentiality, Data Security, Key Management, Privacy Preservation, Public Cloud, Trusted Execution Environment (TEE)

I. INTRODUCTION

The pervasive adoption of public cloud computing has revolutionized IT infrastructure, offering unparalleled scalability, flexibility, and cost efficiency. Organizations across various sectors, including healthcare, finance, and government, are increasingly migrating sensitive workloads and processing confidential data in cloud environments [6]–[8]. This trend is driven by the need for advanced analytics, machine learning capabilities, and global accessibility that on-premises infrastructure often struggles to provide. However, processing sensitive information in a shared, multi-tenant public cloud inherently introduces significant privacy and security challenges [3].

Traditional cloud security models primarily focus on securing data at rest (encryption on storage) and in transit (encryption during transmission). While essential, these measures leave a critical vulnerability: data is typically decrypted in memory during processing by the CPU [1]. In this state, the data is potentially exposed to various threats, including malicious actors with access to the underlying infrastructure, compromised hypervisors, or even insiders within the cloud provider [1], [9]. For highly sensitive workloads subject to stringent regulatory requirements and privacy mandates (like GDPR, HIPAA, or CCPA), this period of exposure during computation is unacceptable and hinders broader cloud adoption for critical applications.

Confidential Computing emerges as a transformative technology designed to address this fundamental gap. At its core, Confidential Computing utilizes hardware-based Trusted Execution Environments (TEEs) to create secure, isolated enclaves within the CPU [2], [4], [5]. These enclaves protect data and code while they are in use, ensuring that the contents of the enclave are inaccessible to the rest of the system, including the operating system, hypervisor, and cloud provider infrastructure. The value proposition of Confidential Computing is the ability to process sensitive data in the public cloud with an unprecedented level of confidentiality, significantly reducing the trusted computing base and mitigating the risks associated with the processing phase [1], [9].

Without Confidential Computing, organizations face significant privacy risks when processing sensitive data in the cloud. Regulatory compliance becomes challenging, the potential for data breaches increases, and the ability to collaborate on sensitive datasets across organizational boundaries is severely limited due to trust concerns [9], [15]. Confidential Computing provides a technical foundation to overcome these barriers, enabling new use cases and unlocking the full potential of cloud computing for privacy-critical applications [9], [7].

This paper aims to explore the architectural patterns and design considerations for deploying privacy-preserving workloads leveraging confidential computing capabilities available in major public cloud environments [6], [7], [8]. We will delve into the technical challenges associated with integrating TEEs into cloud-native architectures, including aspects of data handling, attestation, key management, and workload orchestration [2], [10]. By examining practical architectural blueprints and discussing key implementation challenges, this paper seeks to provide technical architects with the necessary guidance to design and implement secure, confidential applications in the public cloud.

II. Background and Related Work

The evolution of cloud computing security has progressed through several phases, starting with encryption at rest and in transit, moving toward advanced identity and access management, and finally focusing on runtime security [1], [3]. However, protecting data during active computation remained a persistent challenge. Recognizing this critical gap, the concept of Confidential Computing emerged, fundamentally reshaping how sensitive workloads can be securely processed in public cloud environments [1], [9].

Confidential Computing relies on Trusted Execution Environments (TEEs)—isolated hardware-protected regions of memory designed to ensure the confidentiality and integrity of code and data during execution [1], [2]. TEEs prevent unauthorized access or tampering, even by privileged system software such as operating systems, hypervisors, or cloud administrators. Early research in trusted computing led to several notable TEE implementations, each with distinct architectural approaches and security guarantees [1].

One of the most widely recognized TEE technologies is Intel Software Guard Extensions (SGX), which introduced secure enclaves within Intel processors, enabling fine-grained protection of sensitive application code and data [2]. AMD Secure Encrypted Virtualization (SEV) extended the concept to virtual machines, offering memory encryption at the VM level while supporting broader application compatibility [4]. ARM TrustZone pioneered trusted execution concepts for mobile and embedded devices, partitioning the processor into secure and non-secure worlds to safeguard critical assets [5].

Building upon these hardware foundations, major cloud providers have introduced Confidential Computing capabilities into their service offerings. AWS Nitro Enclaves allow customers to create isolated compute environments without persistent storage or external networking, ideal for secure data processing and attestation use cases [6]. Azure Confidential VMs leverage AMD SEV-SNP to provide full VM memory encryption with minimal code changes, enabling broader workload compatibility [7]. Google Cloud

Confidential Computing offers Confidential VMs powered by AMD SEV and is expanding toward Confidential GKE Nodes for containerized workloads [8]. These advancements demonstrate growing momentum toward integrating confidential computing into mainstream cloud services [6]–[8].

While academic research has extensively explored TEE security properties, formal verification, and potential vulnerabilities [1], [11], much of the existing literature remains focused on theoretical models and isolated system designs. There is a notable lack of practical architectural guidance for integrating TEEs into real-world, cloud-native applications at scale. Topics such as secure workload orchestration, data ingress/egress management, attestation workflows, and multi-cloud interoperability remain underexplored from an architectural perspective [9].

This paper addresses this research gap by providing actionable architectural patterns, deployment considerations, and technical strategies to enable secure, privacy-preserving cloud workloads leveraging confidential computing technologies. By grounding our exploration in practical design challenges and cloud-native integration scenarios, we aim to bridge the divide between theoretical TEE research and operational cloud architecture [9].

III. Architecture for Privacy-Preserving Workloads

Designing privacy-preserving workloads leveraging confidential computing requires strict adherence to fundamental architectural principles that ensure security, scalability, and operational efficiency [1], [2], [9].

A. *Core Architectural Principles*

1) Minimal Trusted Computing Base (TCB): Reducing the size and complexity of the trusted code that executes within the TEE is paramount [1]. A smaller TCB minimizes the attack surface and simplifies security verification. Only the essential components of the application logic should reside inside the enclave, with non-sensitive operations delegated to the untrusted environment [9].

2) Secure Workload Isolation: Confidential workloads must be strictly isolated from both other workloads and the underlying host infrastructure. Isolation guarantees that even privileged cloud components, such as hypervisors and host operating systems, cannot access enclave-protected data [2], [4].

3) Explicit Trust Models: A clear definition of the trust boundary between the application components, cloud provider infrastructure, and external users is critical. Trust models should be formalized, incorporating cryptographic attestation to verify enclave integrity before any sensitive operations are authorized [1], [9].

4) Data Minimization within the TEE: Only sensitive data and critical code should be processed inside the TEE. Non-essential functions, such as logging, metrics collection, or non-sensitive business logic, should operate outside the enclave to optimize performance and simplify enclave management [1].

5) Secure Boot and Runtime Integrity: The integrity of the enclave at launch and during execution must be assured [2]. Secure boot processes combined with continuous attestation mechanisms help detect and mitigate attempts at runtime tampering or unauthorized modifications [1].

B. *Key Architectural Components*

Building upon the core principles, several critical components must be addressed when designing cloud-native confidential computing workloads [9].

1) Data Ingress/Egress Design: Secure channels must be established to transfer data into and out of the enclave, ensuring confidentiality and integrity [2]. Data should be encrypted during transit and decrypted only within the secure boundaries of the TEE [9]. Access controls must govern ingress and egress points to prevent data leakage.

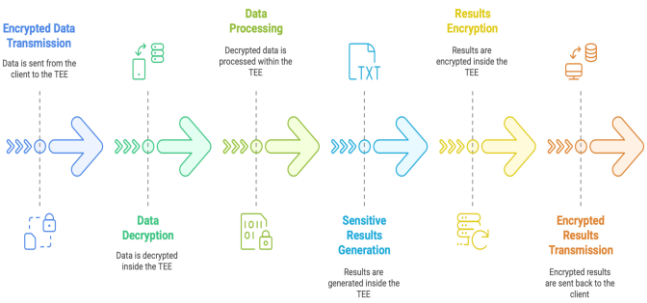

Fig 1: Data Ingress/Egress Flow with TEE

2) Enclave Lifecycle Management: Managing the enclave lifecycle—including provisioning, initialization, operation, update, and decommissioning—is essential for maintaining security guarantees [2], [9]. Workflows must include enclave attestation at startup, controlled updates using signed binaries, and secure teardown processes to eliminate residual data traces.

3) Secure Attestation Mechanisms: Attestation enables verification of the enclave's integrity before sensitive data or operations are permitted [1], [9]. Both local attestation (within a cloud provider's infrastructure) and remote attestation (by external verifiers) should be supported, incorporating evidence signing, verification protocols, and optional third-party attestation services [9].

4) Key Management Strategies: Sensitive cryptographic keys used within confidential workloads must be managed securely. Integration with cloud-native Key Management Systems (KMS) and Hardware Security Modules (HSMs) is recommended [6], [7]. Keys should only be provisioned to the enclave after successful attestation to prevent exposure to untrusted environments.

5) Confidential Workload Orchestration: Managing confidential workloads at scale requires secure orchestration frameworks. Kubernetes-based confidential computing extensions—such as confidential pods or confidential containers—allow workloads to be scheduled, updated, and monitored while preserving TEE integrity [12], [13]. Integration with service meshes and policy engines should ensure that enclave-specific security requirements are enforced during orchestration.

C. Sample Reference Architecture

The following reference architecture illustrates a typical deployment of a privacy-preserving workload in a public cloud using confidential computing capabilities [6], [7], [8].

- Client initiates a connection through a Secure Entry Point (e.g., an HTTPS-based API Gateway).
- The request is routed to a Confidential API Gateway capable of verifying the requestor's identity and enforcing access policies.
- Authorized requests are securely forwarded to a Confidential Application Running Inside a TEE (e.g., an enclave on AWS Nitro [6] or Azure Confidential VM [7]).
- The application processes the sensitive data entirely within the enclave [6], [7].
- Processed data is either returned to the client securely or stored in an Encrypted Storage Layer or Confidential Database, ensuring that persistent storage remains secure [8].

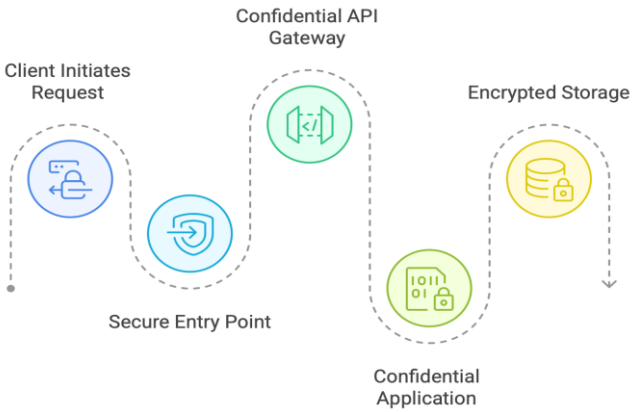

Fig 2: Sample Reference Architecture

IV. Implementation Considerations

While confidential computing offers strong privacy guarantees, integrating Trusted Execution Environments (TEEs) into cloud-native architectures introduces several practical challenges. These considerations impact application performance, development workflows, and system interoperability, and must be addressed carefully during design and deployment [1], [9].

A. *Performance Overheads*

1) Latency Impact of TEE Operations: Executing workloads inside a TEE typically introduces additional latency compared to traditional cloud deployments [2]. Factors contributing to latency include cryptographic operations, enclave entry/exit overhead (ECALLs/OCALLs in Intel SGX terminology), and attestation-related delays [2]. Applications that require frequent context switches between trusted and untrusted code paths are particularly susceptible to increased response times.

2) Resource Allocation and Management Inside Enclaves: TEEs often impose strict limits on available CPU, memory, and storage resources. For example, AWS Nitro Enclaves require careful VM sizing [6], while Intel SGX enclaves are constrained by enclave page cache (EPC) size [2]. Applications must be designed to efficiently manage enclave memory to avoid paging penalties, which can significantly degrade performance [2], [3].

3) Optimization Techniques: Performance impacts can be mitigated through several strategies [9]:

a) Minimal Trusted Code: Reducing the amount of code and data within the enclave minimizes execution overhead.

b) Batch Processing: Reducing frequent enclave transitions by batching operations can improve throughput.

c) Efficient Data Serialization: Optimizing serialization/deserialization between trusted and untrusted contexts reduces latency.

d) Asynchronous Workflows: Designing systems to offload non-sensitive processing outside the TEE asynchronously can preserve performance without sacrificing security.

B. *Development Lifecycle Challenges*

1) Developing Applications for TEEs: Traditional application development paradigms must be adapted for TEE environments [6]. Developers must partition application logic carefully between trusted and

untrusted components. Moreover, enclave-specific SDKs (such as Intel SGX SDK [2], AWS Nitro Enclaves SDK [6]) impose unique constraints on programming models, APIs, and libraries.

2) Debugging Encrypted Applications and Restricted Environments: Debugging inside a TEE is inherently difficult due to encrypted memory and restricted access [2]. Conventional debugging tools are often unusable. Instead, developers must rely on techniques such as:

- Secure logging mechanisms (where logs are cryptographically protected) [2].
- Remote attestation combined with integrity checks to validate execution correctness [9].
- Limited introspection tools provided by specific cloud providers or TEE frameworks [6].

3) CI/CD Pipeline Integration for Confidential Applications: Automating build, test, and deployment processes for confidential workloads requires specialized pipelines [10]:

- Secure Build Environments: Enclave binaries must be compiled in isolated, trusted environments to prevent supply chain attacks.
- Signing and Verification: Enclave code must be digitally signed, and signature verification should be integrated into the deployment process [9].
- Attestation Integration: Continuous delivery pipelines must validate enclave attestations to confirm that deployed workloads maintain integrity post-deployment [9].

C. *Compatibility and Integration*

1) Working with Existing or Legacy Cloud Services: Many legacy cloud services are not enclave-aware and may not support secure communication directly with TEEs. Architects must design secure bridges (such as proxy services or confidential gateways) to enable interaction while preserving confidentiality. Additionally, integrating confidential workloads with services such as serverless platforms (AWS Lambda, Azure Functions) may require additional layers of abstraction or secure hand-off mechanisms, as these services may not natively support TEEs [6], [7].

2) Cross-Cloud (Multi-Cloud) Confidential Workload Considerations and Portability Challenges: Building portable confidential applications across multiple cloud providers remains a significant challenge due to differences in TEE implementations, SDKs, and attestation mechanisms [4], [8]. For instance, applications written for Intel SGX-based platforms may not be directly compatible with AMD SEV-based infrastructures [2], [4]. Key portability considerations include:

- Abstracting enclave functionality behind standardized APIs [10].
- Utilizing open frameworks like Enarx or Open Enclave SDK where possible [10].
- Designing loosely coupled systems that minimize hardware-specific dependencies [9].

Overcoming these challenges is crucial for organizations aiming to maintain cloud-agnostic strategies without compromising workload confidentiality [9].

V. CHALLENGES AND TRADE-OFFS

While confidential computing provides significant advantages in securing data during processing, its adoption is accompanied by a set of challenges and architectural trade-offs. Technical architects must weigh these factors carefully to design effective, scalable, and maintainable privacy-preserving cloud applications [9].

A. *Hardware Dependency and Availability Across Regions/Providers*

Confidential computing capabilities are tightly coupled to specific hardware features such as Intel SGX [2], AMD SEV [4], or proprietary technologies like AWS Nitro Enclaves [6]. Not all cloud regions or providers uniformly support these capabilities [6]–[8]. For instance, certain instance types or TEE-enabled VMs may only be available in select regions, limiting deployment flexibility and complicating disaster recovery and global scaling strategies. This hardware dependency also imposes risks related to hardware lifecycle changes, supply chain issues, and evolving security vulnerabilities specific to a given TEE implementation [2], [4], [6].

Architects must design solutions with awareness of regional availability maps, service coverage gaps, and fallback strategies in case TEE resources are unavailable [6]–[8].

B. *Limited Memory and Resources Inside Some TEE Types*

Many TEEs impose significant resource constraints, particularly on available memory [2], [6]. For example, Intel SGX enclaves have limited Enclave Page Cache (EPC) memory, and exceeding these limits can lead to expensive paging operations that drastically impact performance [2]. Similarly, AWS Nitro Enclaves require careful CPU and memory partitioning from the parent instance [6].

Applications must be explicitly designed to operate efficiently within these resource boundaries, minimizing memory usage inside the enclave and externalizing non-sensitive workloads wherever feasible [3], [9].

C. *Operational Complexity*

Deploying and managing confidential workloads introduces additional operational challenges compared to standard cloud deployments.

1) Monitoring and Logging: Monitoring the health and performance of workloads running inside TEEs is inherently more difficult. Standard monitoring agents cannot access enclave-internal operations without compromising security. Confidential-aware observability patterns—such as secure telemetry pipelines or enclave-generated signed logs—must be established to maintain operational visibility [9].

2) Handling Attestation Failures and Enclave Crashes: Attestation failures (due to hardware issues, software misconfigurations, or cloud provider service disruptions) can prevent workloads from starting or cause service interruptions [2], [6]. Similarly, enclave crashes due to bugs, memory exhaustion, or runtime faults must be detected and remediated securely. Designing robust error handling, alerting, and automated recovery mechanisms is critical to maintaining service reliability.

D. *Balancing Security Guarantees with Performance Requirements*

Confidential computing architectures must carefully balance strong security assurances against acceptable application performance [1], [9]. Overzealous enclave usage—for example, encrypting or isolating all application components unnecessarily—can introduce excessive latency, resource overhead, and increased complexity [3].

Effective designs prioritize protecting only sensitive data and critical logic inside TEEs while leaving less sensitive operations to run outside in standard cloud environments [9]. Performance optimization techniques such as batching, minimal trusted codebases, and asynchronous processing become essential to maintaining user experience and service-level objectives (SLOs) [1].

E. Vendor Lock-in Considerations

Different cloud providers offer confidential computing capabilities through proprietary implementations and APIs, increasing the risk of vendor lock-in [6]–[8]. Workloads designed specifically for AWS Nitro Enclaves, for example, may not port seamlessly to Azure Confidential VMs or Google Confidential Computing instances without substantial re-engineering [6], [7], [8].

To mitigate this risk, architects should consider:

- Using open-source SDKs and frameworks that abstract hardware dependencies (e.g., Open Enclave SDK [10]).
- Designing modular systems where enclave-specific components are loosely coupled from broader application logic [9].
- Monitoring developments in cross-cloud confidential computing standards and emerging interoperability initiatives by organizations such as the Confidential Computing Consortium [9].

Vendor lock-in remains an important strategic consideration, particularly for organizations pursuing multi-cloud or hybrid cloud architectures.

VI. FUTURE TRENDS

As confidential computing matures, new technological advancements, integration models, and industry use cases are emerging. These trends are expected to shape the future of privacy-preserving workloads in public cloud environments and beyond.

A. Confidential Containers

One of the most significant evolutions is the rise of Confidential Containers—the fusion of containerization and confidential computing [12], [13]. Projects such as Kata Containers [12], Confidential GKE Nodes, and Azure Confidential Containers aim to simplify the deployment of containerized workloads within TEEs without requiring extensive application changes. These solutions extend enclave protections to entire containerized runtimes, enabling scalable orchestration of confidential workloads using familiar Kubernetes and container tooling [12], [13]. Advancements in confidential container runtimes will further democratize confidential computing by lowering development and operational barriers.

B. Confidential AI/ML

Protecting intellectual property and sensitive training data is a major concern for organizations adopting AI/ML workloads [14], [16]. Confidential AI/ML leverages TEEs to enable privacy-preserving model training, fine-tuning, and inference [14], [16]. Techniques such as secure enclaved model training, federated learning with confidential aggregation, and TEE-protected inferencing are gaining traction [16]. Future architectures will likely combine confidential computing with AI frameworks (e.g., TensorFlow, PyTorch) to enable trusted machine learning workflows without exposing models or training data to untrusted infrastructure.

C. Multi-Party Confidential Computing and Secure Multiparty Computation (MPC)

The integration of Confidential Computing with Secure Multiparty Computation (MPC) opens new possibilities for collaborative analytics across organizations without revealing private data [15]. Multi-party confidential computing models allow multiple enclaves, operated by different stakeholders, to jointly compute on encrypted datasets while preserving each party's confidentiality [15]. Applications include joint fraud detection across banks, collaborative medical research, and privacy-preserving advertising. Future advancements will focus on standardizing secure enclave-to-enclave communication protocols and scalable MPC orchestration [15].

D. *Standardization Efforts*

Industry-wide standardization is critical to ensuring interoperability, portability, and trust in confidential computing systems [9]. The Confidential Computing Consortium (CCC), under the Linux Foundation, is driving initiatives to define open standards, common attestation formats, and unified APIs for enclave-based applications [9]. Efforts such as the Open Enclave SDK [10], Keystone Enclave framework, and standards for confidential container runtimes aim to reduce vendor fragmentation and enable cross-cloud confidential workload portability [9].

Continued standardization will accelerate adoption and foster an ecosystem of interoperable confidential computing solutions [9].

E. *Broader Adoption Across Regulated Industries*

Highly regulated industries such as Finance, Healthcare, and Government sectors are leading early adoption of confidential computing technologies [9]. Financial institutions are exploring secure multiparty analytics for anti-money laundering (AML) compliance [15], healthcare providers are enabling privacy-preserving clinical research, and government agencies are leveraging enclaves for classified workload processing [9]. As regulatory requirements around data privacy intensify globally (e.g., GDPR, HIPAA, CCPA), confidential computing architectures are expected to become a standard component of compliance-driven cloud strategies [9].

F. *Confidential Edge Computing*

The rise of edge computing introduces new privacy and trust challenges as sensitive data is processed closer to the source. Confidential Edge Computing brings TEE protections to edge nodes, enabling secure data collection, real-time processing, and decentralized analytics without exposing data to intermediate or compromised edge infrastructure. Frameworks that combine confidential computing with edge orchestration platforms (e.g., Kubernetes at the Edge, AWS Wavelength) are emerging to enable secure and scalable edge architectures for industries such as manufacturing, autonomous vehicles, and smart cities.

VII. CONCLUSION

As organizations continue to migrate sensitive workloads to public cloud environments, ensuring the confidentiality and integrity of data during processing has become a critical architectural challenge [1], [9]. Traditional security models that focus on data at rest and in transit are insufficient to fully protect against modern threats, leaving a significant exposure gap during computation [1], [9]. Confidential computing, through the use of Trusted Execution Environments (TEEs), addresses this fundamental gap by enabling privacy-preserving workload execution even in untrusted infrastructure [1], [2].

In this paper, we presented architectural patterns, key design principles, and practical implementation considerations for deploying privacy-preserving workloads using confidential computing capabilities in public cloud environments [6]–[8]. We discussed essential architectural components such as secure data ingress/egress, enclave lifecycle management, attestation mechanisms, key management strategies, and confidential workload orchestration [9].

A reference architecture was proposed to illustrate how these components can be integrated to achieve scalable, secure, and operationally feasible confidential computing solutions [6]–[8].

By focusing on real-world challenges—such as performance overheads, development lifecycle complexities, compatibility hurdles, and vendor lock-in risks—this paper aims to provide technical architects and cloud practitioners with actionable guidance for designing and deploying confidential applications at

scale [9]. The practical architectural strategies outlined here help bridge the gap between theoretical advancements in confidential computing and operational cloud deployments [9].

Looking ahead, future research opportunities include advancing confidential container orchestration frameworks [12], enabling privacy-preserving AI/ML workflows inside TEEs [14], [16], improving multi-party computation protocols for cloud environments [15], and driving broader standardization efforts through industry consortia [9], [10]. Furthermore, the expansion of confidential computing to edge environments promises new opportunities and challenges as privacy-preserving computation moves closer to data sources [9].

Confidential computing stands as a pivotal technology for the next generation of cloud security architectures. As ecosystem maturity, hardware capabilities, and software frameworks evolve, confidential computing will play an increasingly central role in enabling trust, privacy, and regulatory compliance across public, hybrid, and multi-cloud landscapes [9].

REFERENCES

[1] F. McKeen, I. Alexandrovich, A. Berenzon, C. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative Instructions and Software Model for Isolated Execution," in *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy (HASP)*, 2013, pp. 10–10. doi: 10.1145/2487726.2488368.

[2] Intel Corporation, "Intel® Software Guard Extensions (Intel® SGX) Developer Guide," 2021. [Online]. Available: https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sgx-developer-guide.html

[3] K. Chen, "Confidential High-Performance Computing in the Public Cloud," in *IEEE Internet Computing*, vol. 27, no. 1, pp. 24-32, 1 Jan.-Feb. 2023, doi: 10.1109/MIC.2022.3226757.

[4] AMD Corporation, "AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More," 2021. [Online]. Available: https://www.amd.com/en/technologies/secure-encrypted-virtualization

[5] ARM Limited, "ARM TrustZone Technology," [Online]. Available: https://developer.arm.com/ip-products/security-ip/trustzone

[6] Amazon Web Services, "AWS Nitro Enclaves," 2024. [Online]. Available: https://docs.aws.amazon.com/enclaves/latest/user/nitro-enclave.html

[7] Microsoft Azure, "Azure Confidential Computing – Protect Data In Use," 2024. [Online]. Available: https://azure.microsoft.com/en-us/solutions/confidential-compute

[8] Google Cloud, "Confidential Computing Overview," 2024. [Online]. Available: https://cloud.google.com/confidential-computing/docs/confidential-computing-overview

[9] Confidential Computing Consortium, "Confidential Computing: Hardware-Based Trusted Execution for Applications," Whitepaper, 2022. [Online]. Available: https://confidentialcomputing.io/wp-content/uploads/sites/10/2023/03/CCC-A-Technical-Analysis-of-Confidential-Computing-v1.3_unlocked.pdf

[10] Open Enclave SDK, "Open Enclave SDK Documentation," 2024. [Online]. Available: https://openenclave.io/sdk/

[11] M. Costa, L. Esswood, and M. Peinado, "Secure Computation on Encrypted Data," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 523–538.

[12] The Linux Foundation, "Kata Containers: Secure Lightweight Virtualization for Cloud-Native Applications," 2023. [Online]. Available: https://katacontainers.io/

[13] Red Hat, "Introducing Confidential Containers on Bare Metal," Red Hat Blog, Jan. 20, 2025. [Online]. Available: https://www.redhat.com/en/blog/introducing-confidential-containers-bare-metal

[14] Z. Chen, S. Huang, X. Li, et al., "Confidential Computing on NVIDIA H100 GPU: A Performance Benchmark Study," *arXiv preprint*, arXiv:2409.03992, 2024. [Online]. Available: https://arxiv.org/abs/2409.03992

[15] Duality Technologies, "Secure Multiparty Computation | MPC Cryptography," [Online]. Available: https://dualitytech.com/glossary/multiparty-computation/

[16] Microsoft, "Confidential AI Inference Architecture," GitHub Repository, 2025. [Online]. Available: https://github.com/microsoft/confidential-ai/blob/main/inference/docs/arch.md