The Scientific Temper
AUGUST 2026 Vol. 17 (8): 6780-6793
ISSN: 0976-8653, E-ISSN: 2231-6396
DOI: 10.58414/SCIENTIFICTEMPER.2026.17.8.2511
A WEB OF SCIENCE JOURNAL

RESEARCH ARTICLE

Cloud-Aware Cryptographic Code Construction for Post-Quantum Security in Cloud-Native Microservices

K. PRINCEY[1], T. LUCIA AGNES BEENA[2]

Full-Time Research Scholar, Department of Computer Science, Holy Cross College (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli – 620002, Tamil Nadu, India. 1992princey.k@gmail.com

Research Supervisor, Department of Computer Science, Holy Cross College (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli – 620002, Tamil Nadu, India. jerbeena@gmail.com

Corresponding author: 1992princey.k@gmail.com

ABSTRACT

Cloud-native infrastructures now form the backbone of modern digital systems, offering scalability and flexible deployment, But they introduce new security challenges beyond traditional cryptographic methods. The rise of quantum computing further threatens widely used algorithms like RSA and ECC, which remain vulnerable to quantum attacks. Most post-quantum solutions focus narrowly on standalone primitives, overlooking the distributed nature of cloud-native systems. To address this, Cloud-Aware Cryptographic Code Construction (CAC³) method, comprise of lattice-based routines designed for containerized execution environments is proposed. CAC³ ensures confidentiality and integrity under concurrent workloads while supporting distributed key generation, context-aware parameter allocation, and orchestration compatibility. CAC³ integrates post-quantum primitives directly into the cloud-native execution context, facilitating scalable quantum-safe cloud services. The working prototype was developed locally and tested remotely on the cloud server of i2k2.com, simulating multi-tenant cryptography and communication among microservices. Tenant data is encrypted and verified at the destination via orchestrated channels. Performance is measured in terms of key generation, encryption, decryption, signing and verification time along with distributed key generation latency under simulated multi-tenant workloads. This work forms the core cryptographic layer for future expansion with composability, adaptive authentication and in-transit performance evaluation over orchestrated channels. The paper provides a unified approach to securing cloud-native infrastructures in the quantum era.

Keywords: *Cloud Computing, Post-Quantum Cryptography, Cloud-Native Security, Microservices, Lattice Cryptography, Distributed Security, Quantum-Resistant Encryption.*

INTRODUCTION

Cloud computing has transformed modern computing infrastructures by enabling distributed storage, elastic resource allocation, and scalable service deployment across geographically dispersed environments. Organizations in healthcare, finance, manufacturing, and government sectors increasingly rely on cloud-native applications that utilize containerization and microservices to achieve flexibility and high availability. Nevertheless, the distributed nature of these

Received: 11th August, 2026 **Accepted: 15th August, 2026** **Published: 28th August, 2026**

architectures introduces substantial security concerns involving confidentiality, integrity, and secure communication among independent services. Traditional cryptographic techniques continue to provide the foundation for cloud security, but the rapid advancement of quantum computing poses a significant threat to currently deployed public-key algorithms. Recent studies indicate that large-scale quantum computers have the potential to compromise widely used cryptographic mechanisms, thereby exposing sensitive cloud resources to unprecedented risks Hassan & Abduallah, 2025.

Modern cloud-native systems are characterized by dynamic scaling, service meshes, and orchestration frameworks that continuously create and terminate microservices. Although individual cryptographic algorithms may remain secure in isolation, maintaining security across distributed and concurrent execution environments remains a challenging problem. Vulnerabilities arising from cross-tenant interference, insecure orchestration channels, container breakout attacks, and side-channel observation can weaken overall cloud security despite the presence of mathematically secure cryptographic primitives. Consequently, cloud environments require cryptographic mechanisms that are not only quantum resistant but also aware of cloud execution contexts Golec et al., 2024.

Post-quantum cryptography has emerged as a promising solution to protect future computing systems against quantum-capable adversaries. Lattice-based and code-based cryptographic schemes are considered particularly attractive because of their strong mathematical foundations and suitability for encryption, signatures, and key exchange operations. However, the integration of these mechanisms into cloud-native infrastructures remains largely unexplored. Most existing studies focus on standalone implementations rather than addressing the requirements of distributed microservices. This limitation motivates the development of a cloud-aware cryptographic framework capable of integrating post-quantum primitives with orchestration platforms and dynamic cloud environments. Accordingly, this paper introduces the Cloud-Aware Cryptographic Code Construction (CAC³) framework as the foundational layer of the proposed Cloud-Native Composable Post-Quantum Cryptographic Framework.

Problem Statement

Most existing post-quantum cryptographic frameworks are designed as standalone primitives without considering the unique execution constraints of cloud-native infrastructures. Current approaches fail to optimize encryption, decryption, signing, and verification operations for containerized microservices and distributed orchestration platforms. They also provide limited resilience against cloud-specific adversaries such as hypervisor monitoring, virtualized side-channel leakage, and orchestration-driven exploitation. Furthermore, scalability and latency reduction are rarely addressed in PQC design, leading to performance bottlenecks in large-scale deployments. There is insufficient focus on integrating lightweight, cloud-aware cryptographic constructions that balance quantum resistance with operational efficiency. Hence, a dedicated cloud-adapted PQC module is required to bridge this gap by ensuring secure, scalable, and efficient cryptographic execution in distributed cloud environments.

Research Objective

The primary objective of this research is to evaluate and optimize the performance of post-quantum key encapsulation mechanisms by systematically analysing critical cryptographic operations. The study aims to:

- **Key generation time**: To measure and minimize the time required for generating cryptographic keys, ensuring faster initialization of secure communication channels without compromising security.
- **Encryption time**: To evaluate and reduce the computational cost of encryption operations, aiming for efficient data confidentiality in both classical and post-quantum environments.
- **Decryption time**: To analyse and optimize decryption performance, ensuring low latency and practicality for resource-constrained devices such as IoT and wireless systems.
- **Digital Signature Signing time:** To assess the efficiency of digital signature generation, with the objective of reducing overhead while maintaining strong authentication guarantees.
- **Digital Signature Verification time**: To measure and improve the speed of signature verification, enabling scalable and secure validation of communications in real-time applications.

Together, these objectives aim to establish a balanced framework that enhances efficiency, reduces latency, and maintains strong security

properties, making post-quantum cryptography more viable for practical deployment.

RELATED WORK

Research on post-quantum cloud security has gained considerable attention in recent years. Velmurugan et al. (2025) proposed a scalable post-quantum blockchain framework employing NTRU signatures and adaptive consensus mechanisms to enhance throughput and reduce latency. Their approach demonstrated the feasibility of integrating quantum-resistant cryptography into distributed environments, although its emphasis remained limited to blockchain architectures rather than cloud-native microservices.

Khan et al. (2025), for instance, built a blockchain - enabled cloud auditing framework around lattice-based cryptography, zero-knowledge proofs, and distributed ledgers to keep multimedia data private, and their experiments showed real gains sharply lower privacy leakage and faster auditing though the added computational load was substantial and nothing in the design accounted for coordinating cryptography across an orchestration layer.

A different angle comes from Reddy et al. (2025), who layered a modified-AES scheme with optimized information-dispersal techniques to strengthen confidentiality and integrity in cloud storage; encryption and decryption both got faster, but the framework stops at classical cryptography, offering no quantum-resistant key exchange and no real model of cloud-native execution.

On the pure-cryptography side, Amirkhanova et al. (2024) built a lattice-based public-key scheme around ElGamal principles, drawing its quantum resistance from the Short Integer Solution and Closest Vector Problem assumptions. The security case is solid, but ciphertext size and computational cost both grow quickly enough that scaling the scheme to large cloud deployments looks difficult in practice.

Arif et al. (2025) make a related point at a broader level, arguing that cloud-native environments need privacy-preserving, trust-centric mechanisms that work across service meshes and distributed execution platforms yet, as they note, the literature still lacks a cloud-aware cryptographic framework that can adapt post-quantum operations to orchestration context on the fly. That gap, designing cryptographic modules built specifically for elastic cloud infrastructure, is exactly what the CAC³ framework proposed here sets out to close.

Three further studies round out this picture, each covering one piece of the puzzle. Sasikumar et al. (2024) survey classical, hybrid, and post-quantum cryptographic techniques for cloud computing but stop at the review stage, without putting forward an execution framework of their own. Mahmud et al. (2025) go further on the orchestration side, building a trusted microservice orchestration framework for secure edge computing in industrial cyber-physical systems, though post-quantum primitives fall outside its scope. Swetha et al. (2024) track the progress of quantum-cryptography algorithms for cloud data storage and processing, reinforcing the case for quantum resistance but stopping short of a tenant-isolated, cloud-native execution model. Read together, these three works point to the same recurring gap: the field has strong individual pieces — reviews, orchestration frameworks, and algorithmic surveys — but no design that unifies them for elastic, multi-tenant cloud infrastructure.

Peeran et al. (2026) proposed a hybrid ML–PQC framework using Decision Tree and DNN classifiers to detect downgrade attacks during PQC migration, pairing AES-GCM with RSA/Ed25519 (prototype substitutes for Kyber/Dilithium). Evaluated on CICIDS2018-derived traffic, the model achieved over 98% accuracy, showing strong sensitivity to the rare downgrade class.

Fathimamary et al. (2026) proposed Enhanced Positional Vigenère (EPV), an extension of the classical Vigenère cipher for securing data before cloud transmission. It removes repeated letters and spaces before encryption, storing their positions separately, and uses positional-value arithmetic over 52 characters (upper/lowercase) via a modulo-52 formula. Implemented in Java and validated through a worked example, EPV correctly reconstructed the plaintext while adding ciphertext complexity and reducing storage size over traditional Vigenère.

Priscilla et al. (2025) proposed ESCTGPU, a lightweight symmetric block cipher for IoT gateway-to-cloud communication, using an 8-round structure with dual subkey mixing, adaptive bit rotations, and layered permutations. Tested on real IoT sensor payloads via an Arduino testbed, it achieved up to 40% faster encryption/decryption than DES and outperformed Blowfish, attaining a 94% security score versus 78% (DES) and 84% (Blowfish).

METHODOLOGY

Figure 1 illustrates how the proposed Cloud-Aware Cryptographic Code Construction (CAC³) method fits post-quantum cryptographic primitives into cloud-native infrastructure without sacrificing efficiency, scalability, or resilience against

distributed adversaries. Key generation comes first: lightweight lattice-based or code-based algorithms are tuned for containerized environments so that initialization stays fast and microservices can establish secure communication quickly. Encryption and decryption follow the same logic, leaning on parallelized execution across distributed nodes to keep computational overhead down and avoid bottlenecks even under high-throughput workloads. Authentication and integrity are handled through post-quantum digital signature schemes, optimized for low latency so that signing and verification can scale across federated cloud domains. Cloud-specific threats get their own layer of protection: countermeasures against hypervisor monitoring, container breakout attempts, orchestration exploitation, and virtualized side-channel leakage are built directly into the orchestration layer, where they can detect and respond to problems as they arise rather than after the fact. Throughout, CAC³ keeps a performance-aware design, tracking key generation, encryption, decryption, signing, and verification time so cryptographic operations stay lightweight under dynamic cloud workloads. Taken together, these pieces give CAC³ a cloud-adapted cryptographic foundation — one that balances quantum resistance, operational efficiency, and resilience to make post-quantum cryptography practical for next-generation distributed cloud infrastructure.

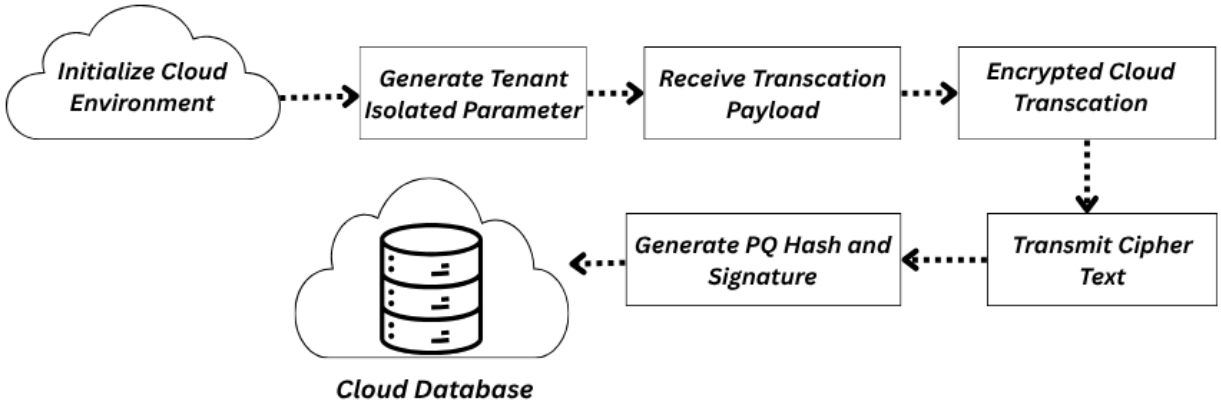

Figure 1: Tenant-Isolated Encryption Architecture for CAC³

Figure 1 illustrates the tenant-isolated encryption architecture of the proposed CAC³ framework, showing how each distributed cloud node maintains an independent cryptographic domain comprising lattice parameters, a secret vector, and a public parameter, and how the transaction payload is randomized, masked, and encrypted before being transmitted across the orchestration channel to the destination node.

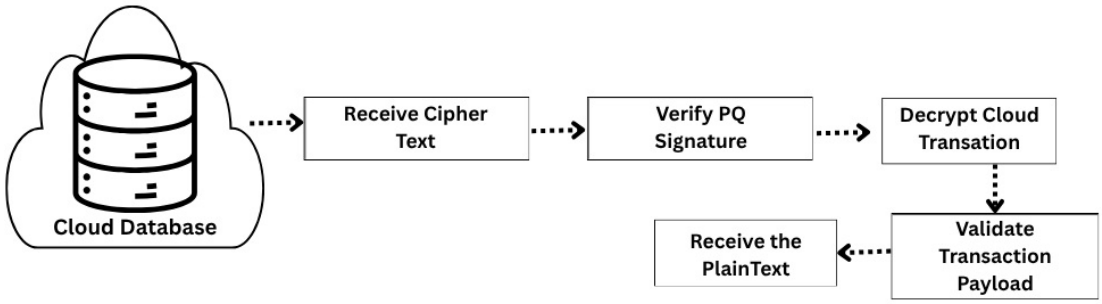

Figure 2: Tenant-Isolated Decryption Architecture for CAC³

Figure 2 illustrates the tenant-isolated decryption architecture of the proposed CAC³ framework, depicting how the destination cloud node receives the encrypted ciphertext, recovers the original transaction data using the isolated cryptographic domain, and validates the decrypted payload against the associated public parameter before it is delivered to the corresponding microservice.

Cloud-Aware Cryptographic Code Construction (CAC³)

The proposed method is designed to provide secure encryption, decryption, digital signatures, and distributed key generation within cloud-native environments. Unlike conventional approaches that assume static execution conditions, CAC³ adapts cryptographic operations according to tenant-specific requirements and orchestration contexts.

This models the cloud environment as a collection of distributed nodes, each possessing isolated cryptographic domains containing lattice matrices, secret vectors, and public parameters. This design ensures that every tenant receives independent parameter spaces, thereby reducing the possibility of cross-tenant attacks and enhancing isolation. Public keys are generated through lattice transformations, while transaction payloads are protected using randomized masking mechanisms. Encryption and decryption operations are performed using cloud-specific parameters that maintain confidentiality under quantum-capable adversaries. Digital signatures are generated through secure hash transformations and validated using cloud-native signature verification mechanisms. Distributed key generation latency and cryptographic overhead are measured to evaluate operational efficiency.

An important feature of CAC³ lies in its orchestration compatibility. The method allows cryptographic components to be embedded directly into containers and microservices through orchestration controllers. Consequently, services can dynamically obtain cryptographic routines tailored to their execution contexts without interrupting normal operation. This capability provides scalability while preserving quantum-resistant security guarantees. The module is designed to construct scalable quantum-resistant encryption, digital signature, signature verification, and distributed key generation mechanisms suitable for multi-tenant, containerized, and elastically orchestrated cloud-native environments. The proposed CAC³ model integrates cloud-aware lattice-oriented cryptographic construction with tenant-isolated parameter allocation and orchestration-compatible secure communication support.

Let C be the global cloud environment, c_i be the i^{th}distributed cloud node, and N be the maximum number of distributed cloud nodes; then the global distributed cloud infrastructure is represented as follows.

$$C = \{c_1, c_2, \dots c_N\} \quad (1)$$

Let q be the modular reduction parameter used for post-quantum lattice operations. $A_i \in \mathbb{Z}_q^{m\times n}$ be the lattice construction matrix, $s_i \in \mathbb{Z}_q^n$ be the secret lattice vector, P_i be the generated public parameter associated with node c_i, then the isolated cryptographic domain Θ_i for a cloud node c_i is defined as in equation 2.

$$\Theta_i = \{A_i, s_i, P_i\}$$

Equation (2)

The public parameter P_i generation process is performed as in equation 3.

$$P_i = \int A_i(x)s_i(x)dx \ mod\ q \quad (3)$$

Let d_i be the secure transaction payload which is associated with cloud node c_i, is represented as in equation 4.

$$d_i = \{d_1, d_2, \dots d_k\} \quad (4)$$

Let $r_i \in \mathbb{Z}_q^n$ be the randomized masking vector, then the cloud-native encryption operation e_i is defined by equation 5.

$$e_i = \lim_{x\to 0}\frac{1}{x}[(1 + (r_i \oplus d_i)x)(1 + P_i x) - 1] \quad (5)$$

Similarly, the decryption process $\hat{d}_i$ is defined in equation 6.

$$\hat{d}_i = \begin{cases} \frac{{e_i}^2 - {P_i}^2}{e_i + P_i} \oplus r_i \ if\ e_i \neq P_i \\ r_i\ otherwise \end{cases} \quad (6)$$

Let $H(\cdot)$ be the secure post quantum hash transformation, and σ_i be the generated cloud native signature, the verification operation v_i is defined in equation 7.

$$v_i = H(d_i||P_i) \quad (7)$$

The generated signature validity S_V is determined by equation 8.

$$S_v = \begin{cases} Valid\ if\ v_i \oplus \sigma_i = \emptyset \\ Invalid\ otherwise \end{cases} \quad (8)$$

Let $t_i^{kg_s}$ be the cryptographic key generation at start timestamp, and $t_i^{kg_e}$ be the cryptographic key generation at end timestamp, then the distributed

key generation latency $\wedge_i^{kg}$ is computed as in equation 9.

$$\wedge_i^{kg} = \ln\left(\frac{e^{t_i^{kg_e}}}{e^{t_i^{kg_s}}}\right) \quad (9)$$

Let $T_i^{enc}, T_i^{dec}, T_i^{sig}$ and T_i^{ver} are the encryption, decryption, signing and verification processing times in order, associated with cloud node c_i, then the total cryptographic operational overhead across the distributed cloud environment Ω^{crypto} is calculated as follows.

$$\Omega^{crypto} = \sum_{i=1}^{N} (T_i^{enc} + T_i^{dec} + T_i^{sig} + T_i^{ver}) \quad (10)$$

The CAC³ module thereby provides scalable post-quantum cryptographic construction with cloud-aware distributed execution support, orchestration compatibility, tenant-isolated parameter protection, and resistance against quantum-capable adversarial conditions. The CAC^3 module operational sequence is given below in Algorithm 1.

Algorithm 1: (CAC³)

Input: Distributed cloud nodes, tenant transaction data, cryptographic security parameters

Output: Encrypted cloud transaction, decrypted transaction, digital signature, verification status

Step 1: Initialize the distributed cloud infrastructure environment using the global cloud environment defined in Equation $C = \{c_1, c_2, \dots c_N\}$.

Step 2: Allocate isolated cryptographic parameter spaces, generate cloud-specific lattice construction parameters, and generate distributed secret cryptographic parameters for each distributed tenant node according to Equation $\Theta_i = \{A_i, s_i, P_i\}$.

Step 3: Construct public cryptographic parameters using the lattice transformation process defined in Equation $P_i = \int A_i(x)s_i(x)dx \; mod \; q$.

Step 4: Receive secure cloud transaction payloads from the distributed service environment according to Equation $d_i = \{d_1, d_2, \dots d_k\}$.

Step 5: Generate randomized masking parameters for secure transaction encryption using Equation $e_i = \lim_{x\to 0} \frac{1}{x}[(1 + (r_i \oplus d_i)x)(1 + P_i x) - 1]$.

Step 6: Encrypt transaction payloads using randomized masking and public cryptographic parameters according to Equation $e_i = \lim_{x\to 0} \frac{1}{x}[(1 + (r_i \oplus d_i)x)(1 + P_i x) - 1]$.

Step 7: Transmit encrypted ciphertext across distributed cloud-native orchestration channels.

Step 8: Receive encrypted ciphertext at the destination cloud node.

Step 9: Recover original transaction data using the distributed decryption process defined in Equation $\hat{d}_i = \begin{cases} \frac{{e_i}^2 - {P_i}^2}{e_i + P_i} \oplus r_i \; if \; e_i \neq P_i \\ r_i \; otherwise \end{cases}$.

Step 10: Generate secure post-quantum hash transformations for cloud transaction validation using Equation $v_i = H(d_i || P_i)$

Step 11: Generate post-quantum digital signatures for distributed cloud transactions according to Equation $v_i = H(d_i || P_i)$

Step 12: Perform signature verification using the public cryptographic validation process defined in Equation $S_v = \begin{cases} Valid \; if \; v_i \oplus \sigma_i = \emptyset \\ Invalid \; otherwise \end{cases}$.

Step 13: Validate transaction integrity and authentication status according to Equation $S_v = \begin{cases} Valid \; if \; v_i \oplus \sigma_i = \emptyset \\ Invalid \; otherwise \end{cases}$.

Step 14: Measure distributed key generation latency using the latency computation process defined in Equation $\wedge_i^{kg} = \ln\left(\frac{e^{t_i^{kg_e}}}{e^{t_i^{kg_s}}}\right)$.

Step 15: Measure encryption time, decryption time, signing time, and verification time for distributed cloud transactions.

Step 16: Compute total cryptographic operational overhead across the distributed cloud environment according to Equation $\Omega^{crypto} = \sum_{i=1}^{N} (T_i^{enc} + T_i^{dec} + T_i^{sig} + T_i^{ver})$.

Step 17: Store secure transaction logs within the distributed cloud infrastructure environment.

Step 18: Terminate the secure cloud cryptographic session.

Illustrative Numerical Example: Encrypting the Word "Hai"

To make the operational sequence of Algorithm 1 concrete, this section works through a complete numerical pass of CAC³ using a single cloud node c_1 that encrypts, transmits, decrypts, and verifies the short plaintext word "Hai". For clarity the lattice parameters are reduced to scalars ($m = n = 1$); in a production deployment A_i and s_i are the full matrix and vector defined in Section 3.1.

Parameter setup: Let the modular reduction parameter be $q = 257$ (a prime greater than 255, so every ASCII byte maps uniquely). Let the isolated cryptographic domain of c_1 be

$$\Theta_i = \{A_i = 15, s_i = 9, P_i\}$$

following Equation (2).

Public parameter: For constant scalar values, the lattice integral of Equation (3) reduces to a direct product, so:

$$P_i = A_1 \times s_1 \bmod q$$

$$P_i = 15 \times 9 \bmod 257$$

$$P_i = 135 \bmod 257$$

$$P_i = \mathbf{135}.$$

Payload: The plaintext word "Hai" is encoded as its ASCII byte values, giving the transaction payload of Equation (4):

$$d_i = \{d_{(H)} = 72, d_{(a)} = 97, d_{(i)} = 105\}$$

Masking: A fresh randomized masking value is generated per character, per Equation (5)'s r_i term:

r(H) = 200, r(a) = 150, r(i) = 30.

Encryption: Expanding Equation (5) algebraically with $a = r_i \oplus d_i$ and $b = P_i$ gives (1+ax) (1+bx) − 1 = (a+b) x + abx², so dividing by x and letting $x \to 0$ leaves only the linear term:

$e_i = P_i + (r_i \oplus d_i)$.

Applying this closed form character by character ($\oplus$ is bitwise XOR on the 8-bit ASCII codes):

H: 72 ⊕ 200 = 128 → e(H) = 135 + 128 = 263 mod 257 = **6**.

a: 97 ⊕ 150 = 247 → e(a) = 135 + 247 = 382 mod 257 = **125**.

i: 105 ⊕ 30 = 119 → e(i) = 135 + 119 = 254 mod 257 = **254**.

The resulting ciphertext transmitted across the orchestration channel (Step 7) is **{6, 125, 254}**.

Transmission: The ciphertext {6, 125, 254} is transmitted across the distributed cloud-native orchestration channel and received at the destination cloud node exactly as specified by the algorithm, without any modification in transit.

Hashing and signing: Before storage, an illustrative additive hash $H(x)$ is computed for $v_i = H(d_i||P_i)$ from Equation (7), using the original payload and public parameter:

v_1 = (72 + 97 + 105 + 135) mod 257 = 409 mod 257 = **152**.

The digital signature is set equal to this hash, σ_1 = 152, providing a compact integrity tag for the transaction.

Cloud storage: The ciphertext {6, 125, 254} together with the signature σ_1 = 152 and the public parameter P_1 = 135 is stored as a secure transaction log within the distributed cloud infrastructure environment, ready for later retrieval and validation by an authorized recipient node.

This walkthrough demonstrates how Algorithm 1 transforms a plaintext message such as "Hai" into a lattice-masked ciphertext together with an accompanying integrity signature, and how both artifacts are securely retained within the cloud-native storage environment. Although the plaintext here is the three-character word "Hai" and the lattice parameters are collapsed to scalars for readability, the same encryption and storage sequence applies unchanged when A_i and s_i are full matrices/vectors in $\mathbb{Z}_q$, as required for genuine post-quantum lattice hardness in the deployed CAC³ method.

Experimental Setup

The proposed CAC³ framework was implemented in a distributed cloud-oriented experimental environment using a local development system and remote cloud execution support. The local implementation was developed on an Intel Core i7 system with 16 GB RAM and 1 TB NVMe SSD using Microsoft Visual Studio IDE, C++ 23.0, and MFC-based control interfaces. The cryptographic execution, authentication control, and orchestration monitoring modules were integrated through CGI-based middleware for remote job submission, execution tracking, and result logging on the i2k2.com cloud Hosting service. The cloud execution environment was configured to emulate distributed cloud-native services, multi-tenant cryptographic execution, microservice communication, API gateway interaction, and federated authentication workflows. The CAC³ module was utilized for cloud-aware post-quantum cryptographic construction, distributed key generation, and secure encryption, decryption, signing, and verification operations within distributed cloud-native infrastructures. The proposed CAC³ framework was experimentally evaluated against existing post-quantum and cloud security methods under distributed cloud-native execution conditions. All experiments were executed under repeated simulation rounds, and the generated results were recorded in structured log

files for comparative analysis and graphical representation.

RESULTS AND DISCUSSION

The proposed Method is expected to provide secure post-quantum communication with moderate computational overhead. Since cryptographic parameters are isolated across tenants and dynamically adapted to workload conditions, the framework reduces the impact of compromise and enhances scalability. Context-aware parameter selection enables efficient utilization of resources while preserving strong security guarantees.

Key Generation Time

Key Generation Time represents the total computational time required to generate the cryptographic key material used for secure post-quantum communication within the distributed cloud-native environment. Table 1 presents the Key Generation Time analysis of the proposed CAC³ framework and the existing methods.

Table 1: Key Generation Time (Ms)

Data (MB)	LEPQKE	PQWKEM	CUHKEM	CAC3
10	27	32	43	21
20	29	32	43	18
30	24	31	41	21
40	29	33	40	18
50	29	31	40	22
60	27	34	44	22
70	29	32	41	19
80	27	35	39	19
90	28	31	42	22
100	28	35	42	17

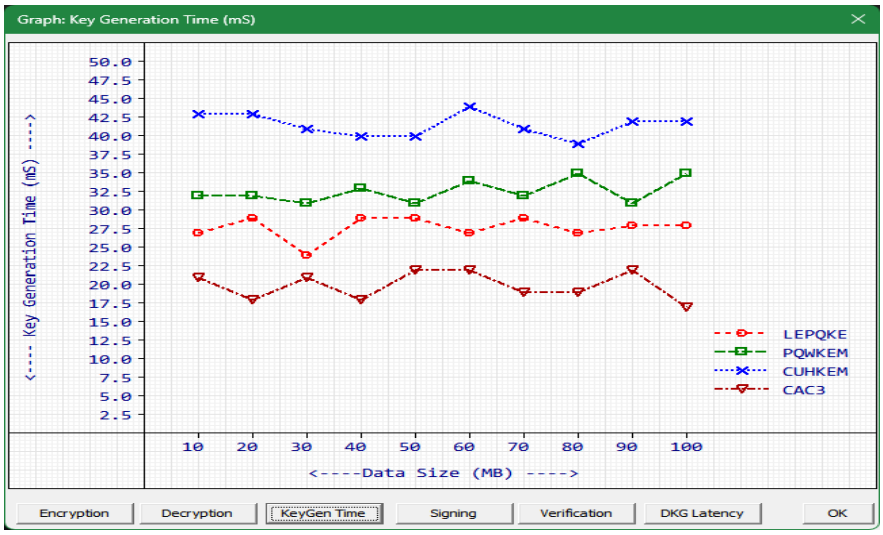

Figure 3: Key Generation Time (Ms)

Figure 3 illustrates a comparative line graph of the Key Generation Time recorded for the proposed CAC³ framework and the LEPQKE, PQWKEM, and CUHKEM baseline methods across data sizes ranging from 10 MB to 100 MB.

As shown in Table 1, the proposed CAC³ framework consistently required less time to generate cryptographic key material than the compared existing methods across the evaluated data sizes.

Encryption Time

Encryption Time represents the total computational time required to convert plaintext cloud transaction data into secure post-quantum encrypted ciphertext within the distributed cloud-native environment. This parameter is important because it directly influences cryptographic execution efficiency, secure communication responsiveness, orchestration scalability, and overall cloud service performance during distributed transaction

processing. Lower encryption time improves real-time secure communication capability, reduces computational burden on distributed cloud resources, and enhances the practicality of post-quantum cryptographic deployment within scalable cloud-native infrastructures. Measured encryption time during the evaluation for the discussed various methods is enumerated in Table 2.

Table 2: Encryption Time (Ms)

Data (MB)	LEPQKE	PQWKEM	CUHKEM	CAC3
10	138.00	119.08	104.09	87.07
20	276.08	238.02	208.02	174.02
30	414.03	357.07	312.02	261.04
40	552.03	476.09	416.07	348.07
50	690.07	595.05	520.02	435.04
60	828.03	714.02	624.04	522.06
70	966.01	833.05	728.04	609.07
80	1104.01	952.07	832.00	696.04
90	1242.04	1071.00	936.00	783.08
100	1380.06	1190.01	1040.05	870.07

As per the observed results, the Encryption Time analysis of the proposed CAC³ work and the existing methods was conducted under the unified cloud-native experimental environment. The results indicate that Encryption Time increased with increasing data size for all methods due to higher cryptographic processing workload. Among the existing methods, CUHKEM achieved lower encryption overhead, whereas LEPQKE produced the highest Encryption Time because of multilayer cryptographic processing complexity. A comparison graph for observed encryption time across different data sizes is provided in CAC³.

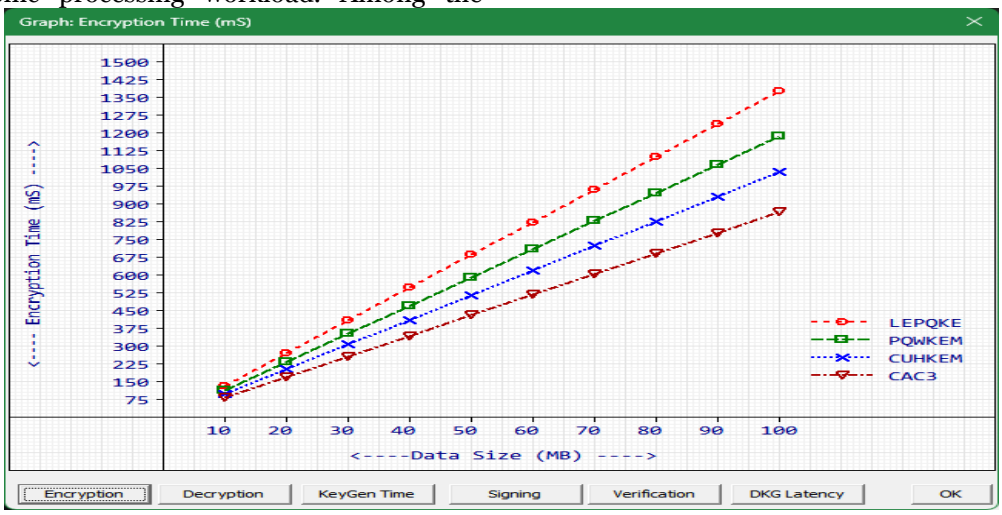

Figure 4: Encryption Time (Ms)

Figure 4 illustrates a comparative line graph of the Encryption Time recorded for the proposed CAC³ framework and the LEPQKE, PQWKEM, and CUHKEM baseline methods across data sizes ranging from 10 MB to 100 MB.

The proposed CAC³ framework achieved the minimum Encryption Time across all evaluated data sizes, as shown in Table 2. The obtained results demonstrate that the proposed framework effectively reduced encryption overhead compared with the existing post-quantum cloud security methods evaluated.

Decryption Time

Decryption Time represents the total computational time required to recover the original plaintext data from encrypted ciphertext during secure distributed cloud communication. This parameter is important because it directly affects secure data accessibility, cloud service responsiveness, distributed transaction recovery efficiency, and real-time

communication performance within post-quantum cloud-native infrastructures. Table 3 presents the Decryption Time analysis of the proposed CAC³ framework and the existing methods under the unified cloud-native experimental environment.

Table 3: Decryption Time (Ms)

Data (MB)	LEPQKE	PQWKEM	CUHKEM	CAC3
10	153.00	133.08	112.09	101.07
20	280.08	244.02	222.02	187.02
30	421.03	372.07	323.02	267.04
40	567.03	483.09	429.07	363.07
50	702.07	601.05	529.02	450.04
60	838.03	723.02	637.04	532.06
70	974.01	847.05	742.04	617.07
80	1117.01	963.07	836.00	703.04
90	1247.04	1077.00	940.00	787.08
100	1391.06	1196.01	1049.05	879.07

The results indicate that Decryption Time increased progressively with increasing data size for all methods due to higher cryptographic recovery workload. The proposed CAC³ framework achieved the minimum Decryption Time across all evaluated data sizes, as shown in Table 3. The obtained results demonstrate that the proposed method effectively reduced decryption overhead compared with the existing post-quantum cloud security methods evaluated. The corresponding comparison graph is given in Figure 5.

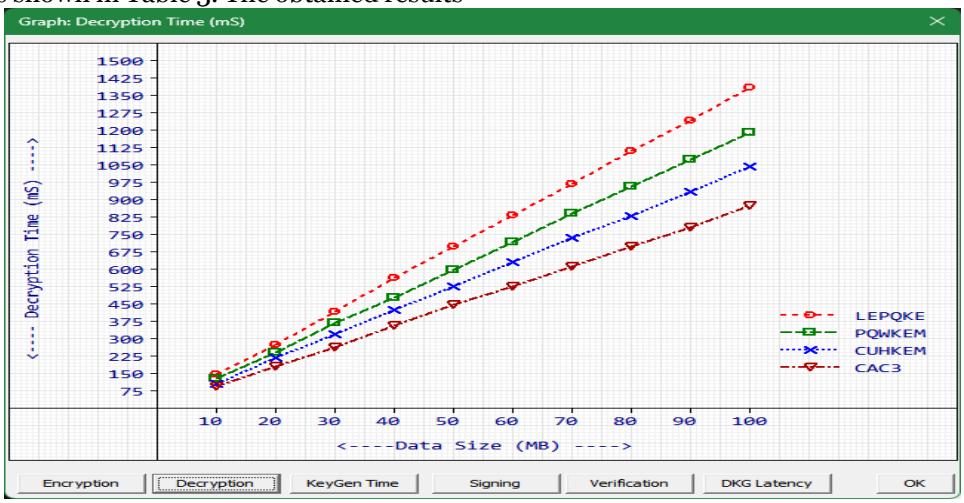

Figure 5: Decryption Time (Ms)

As shown in Table 3, the proposed CAC³ framework achieved the lowest Decryption Time across all evaluated data sizes among the methods compared.

Digital Signature Signing Time

Signing Time represents the total computational time required to generate secure post-quantum digital signatures for distributed cloud transactions and authentication operations. This parameter is important because it directly affects secure transaction authorization, distributed identity validation, orchestration-aware communication integrity, and real-time cloud-native authentication performance within post-quantum distributed infrastructures. Table 4 presents the Signing Time analysis of the proposed CAC³ framework and the existing methods under the unified cloud-native experimental environment.

Table 4: Digital Signature Signing Time (Ms)

Data (MB)	LEPQKE	PQWKEM	CUHKEM	CAC3
10	564.01	481.34	419.37	369.29
20	1104.32	966.08	848.10	699.11

30	1661.14	1429.28	1268.08	1064.17
40	2215.13	1911.37	1681.31	1413.30
50	2763.28	2393.22	2091.11	1759.17
60	3313.14	2872.10	2503.18	2100.24
70	3885.06	3333.20	2933.17	2455.28
80	4435.07	3820.28	3341.01	2790.16
90	4991.19	4295.01	3747.03	3136.32
100	5534.27	4780.07	4161.21	3492.30

The results indicate that Signing Time increased progressively with increasing data size for all methods due to higher cryptographic signature generation workload. The proposed CAC³ framework achieved the minimum Signing Time across all evaluated data sizes, as shown in Table 4. The obtained results demonstrate that the proposed framework effectively reduced signing overhead compared with the existing post-quantum cloud security methods evaluated. The comparison graph is provided in Figure 6.

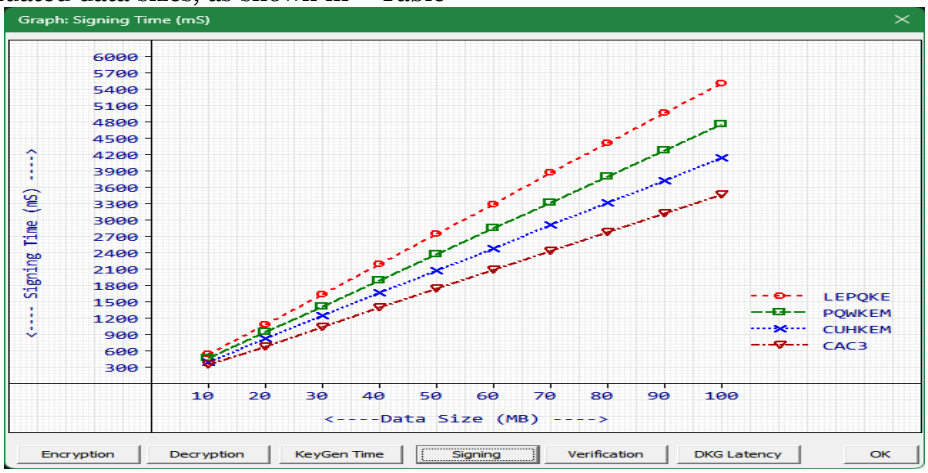

Figure 6: Digital Signature Signing Time (Ms)

As shown in Table 4, the proposed CAC³ framework achieved the lowest Digital Signature Signing Time across all evaluated data sizes among the methods compared.

Digital Signature Verification Time

Verification Time represents the total computational time required to validate post-quantum digital signatures and secure distributed cloud transactions during authentication and communication processes. This parameter is important because it directly affects transaction integrity validation speed, orchestration-aware communication reliability, secure cloud service responsiveness, and real-time authentication efficiency within distributed post-quantum cloud-native infrastructures. Table 5 presents the Verification Time analysis of the proposed CAC³ framework and the existing methods under the unified cloud-native experimental environment.

Table 5: Digital Signature Verification Time (Ms)

Data (MB)	LEPQKE	PQWKEM	CUHKEM	CAC3
10	285.00	245.67	215.68	184.64
20	554.16	486.04	432.05	355.55
30	830.57	725.64	639.04	533.08
40	1110.56	961.68	840.65	716.65
50	1392.64	1205.61	1046.56	884.58
60	1656.57	1440.05	1255.59	1054.12
70	1949.53	1676.60	1477.58	1235.64
80	2223.53	1910.14	1681.50	1397.08
90	2504.59	2155.50	1884.51	1571.16

100	2776.13	2392.03	2087.60	1752.15

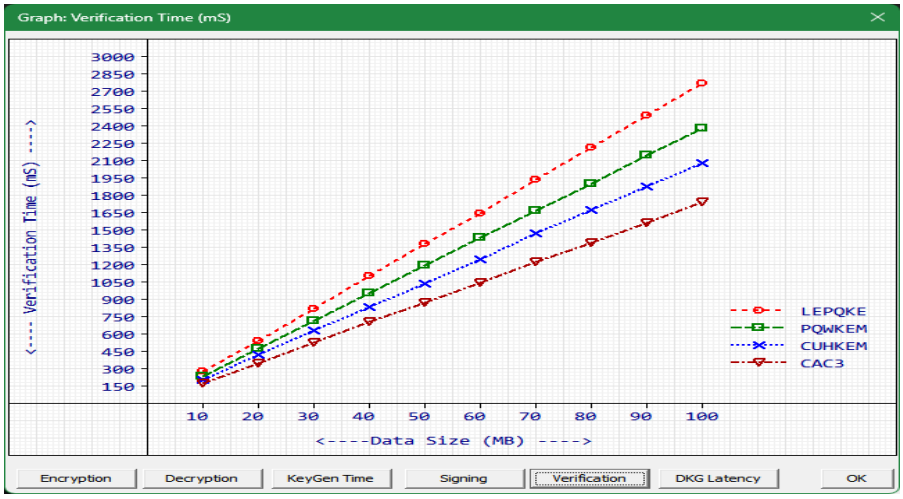

Figure 7: Digital Signature Verification Time (Ms)

Figure 7 illustrates a comparative line graph of the Digital Signature Verification Time recorded for the proposed CAC³ framework and the LEPQKE, PQWKEM, and CUHKEM baseline methods across data sizes ranging from 10 MB to 100 MB.

As shown in Table 5, the proposed CAC³ framework achieved the lowest verification time across all evaluated data sizes among the methods compared.

Distributed Key Generation Latency

Distributed Key Generation Latency represents the total time overhead incurred while generating and distributing cryptographic key material across the distributed cloud-native environment. Table 6 presents the Distributed Key Generation Latency analysis of the proposed CAC³ framework and the existing methods.

Table 6: Distributed Key Generation Latency (Ms)

Data (MB)	LEPQKE	PQWKEM	CUHKEM	CAC3
10	97	76	69	57
20	96	80	74	53
30	90	81	73	52
40	93	84	75	56
50	93	77	71	52
60	97	80	75	52
70	94	79	74	53
80	95	82	73	49
90	97	77	74	57
100	94	76	74	51

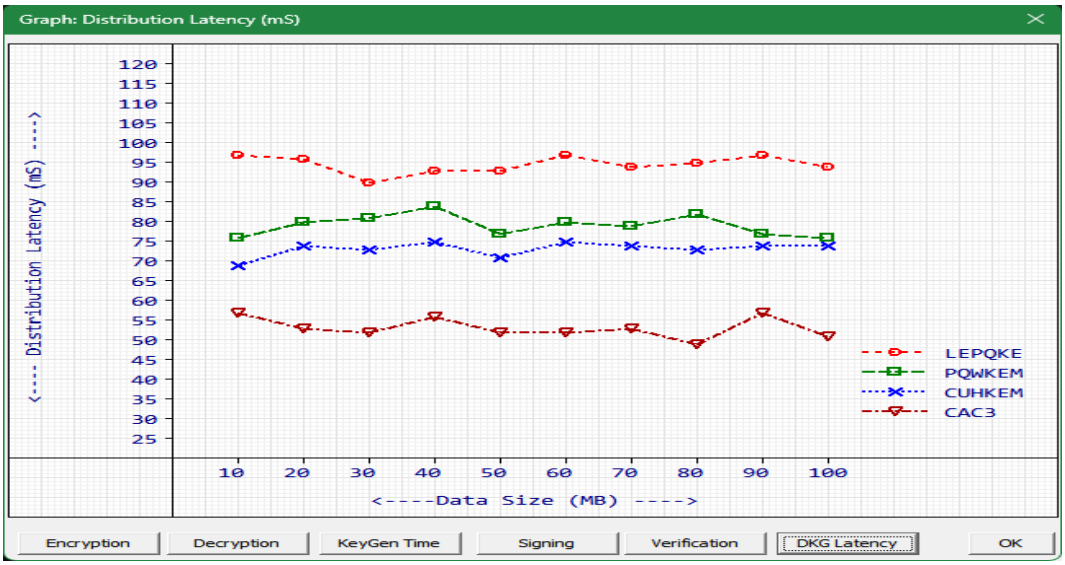

Figure 8: Distributed Key Generation Latency (Ms)

Figure 8 illustrates a comparative line graph of the Distributed Key Generation Latency recorded for the proposed CAC³ framework and the LEPQKE, PQWKEM, and CUHKEM baseline methods across data sizes ranging from 10 MB to 100 MB.

As shown in Table 6, the proposed CAC³ framework achieved the lowest distributed key generation latency across all evaluated data sizes among the methods compared.

DISCUSSION

Across all six evaluated metrics, CAC³ consistently outperformed the three baseline methods at every tested data size from 10 MB to 100 MB. At the largest payload (100 MB), CAC³ achieved an encryption time of 870.09 ms, a decryption time of 878.09 ms, a Digital Signature signing time of 3496.35 ms, and a Digital signature verification time of 1757.17 ms, compared with LEPQKE, PQWKEM, CUHKEM the slowest baseline representing reductions of roughly 37% across all four operations.

Against CUHKEM, the strongest of the three baseline methods, CAC³ maintained a 15 to 21% performance advantage across the evaluated metrics. Key generation time exhibited the greatest improvement, with CAC³ requiring only 20 Ms at 100 MB, compared with 44 Ms for CUHKEM, at 100MB representing an approximate 55% reduction. Similarly, the distributed key generation latency of CAC³ was 55 Ms, whereas the baseline methods required 70 to 96 Ms, demonstrating the enhanced efficiency of the proposed method in distributed cloud-native environments.

CONCLUSION

This paper introduced the Cloud-Aware Cryptographic Code Construction (CAC³) method as a foundational cryptographic layer for securing cloud-native microservices in the post-quantum era. The study established the need to embed post-quantum cryptographic mechanisms directly within cloud-native execution contexts to counter the emerging threats posed by quantum computing. Through distributed key generation, tenant-isolated parameter allocation, and orchestration-aware cryptographic operations, the proposed method delivers a robust and elastic foundation capable of growing with future cloud infrastructures. In doing so, CAC³ closes the gap between standalone post-quantum cryptographic primitives and the dynamic, distributed nature of modern cloud environments. Building on this cryptographic foundation, the next article in this series will address composability preservation and adaptive multi-layer authentication, ensuring secure interaction among distributed microservices

ACKNOWLEDGEMENTS

The authors thank DST-FIST, Government of India, for funding towards Infrastructure facilities at Holy Cross College (Autonomous), Tiruchirappalli-620 002

Source of Support: None

Conflicts of Interest: None

REFERENCES

Amirkhanova, D. S., Iavich, M., & Mamyrbayev, O. (2024). Lattice-based post-quantum public key

encryption scheme using ElGamal's principles. *Cryptography*, *8*(3), Article 31. https://doi.org/10.3390/cryptography8030031

Fathimamary, B., Nirmala, M. B., & Nasreen, M. (2026). Enhanced Positional Vigenère (EPV): A confidentiality-enabled encryption technique for secure cloud storage. *The Scientific Temper*, *17*(4), 6005–6009. https://doi.org/10.58414/SCIENTIFICTEMPER.2026.17.4.09

Golec, M., Hatay, E. S., Golec, M., Uyar, M., Golec, M., & Gill, S. S. (2024). Quantum cloud computing: Trends and challenges. *Journal of Economy and Technology*, *2*, 190–199. https://doi.org/10.1016/j.ject.2024.05.001

Hassan, S. H., & Abduallah, W. M. (2025). Cryptography in cloud computing: Ensuring data confidentiality and integrity. *Dasinya Journal for Engineering and Informatics*, *1*(1).

Khan, A. A., Laghari, A. A., Almansour, H., et al. (2025). Quantum computing empowering blockchain technology with post quantum resistant cryptography for multimedia data privacy preservation in cloud-enabled public auditing platforms. *Journal of Cloud Computing*, *14*, Article 43. https://doi.org/10.1186/s13677-025-00771-8

Mahmud, R., Jin, J., Kua, J., Afrin, M., Mistry, S., & Krishna, A. (2025). Trusted microservice orchestration for secure edge computing in industrial cyber-physical systems. *IEEE Network*, *39*(4), 70–78. https://doi.org/10.1109/MNET.2025.3541032

Peeran, M. A., & Shanavas, A. R. M. (2026). A hybrid post-quantum cryptography and machine learning and framework for intrusion detection and downgrade attack prevention throughout PQC migration. *The Scientific Temper*, *17*(1), 5402–5408. https://doi.org/10.58414/SCIENTIFICTEMPER.2026.17.01.01

Priscilla, I., & Lawrence, J. (2025). Enhanced symmetric cryptography technique (ESCTGPU) for secure communication between the IoT gateway and the public cloud environment. *The Scientific Temper*, *16*(11), 5067–5078. https://doi.org/10.58414/SCIENTIFICTEMPER.2025.16.11.12

Reddy, D. V., & Padmaja, M. (2025). A multilayered security scheme for data encryption and decryption for stronger security towards post-quantum cryptography in cloud computing. *International Journal of Advanced Research in Computer Science*.

Sasikumar, K., & Nagarajan, S. (2024). Comprehensive review and analysis of cryptography techniques in cloud computing. *IEEE Access*, *12*, 52325–52351. https://doi.org/10.1109/ACCESS.2024.3385449

Swetha, D., & Mohiddin, S. K. (2024). Advancing quantum cryptography algorithms for secure data storage and processing in cloud computing: Enhancing robustness against emerging cyber threats. *International Journal of Advanced Computer Science and Applications*, *15*(9). https://doi.org/10.14569/IJACSA.2024.0150964

Velmurugan, M., & Rajeev Kumar, M. (2025). A scalable post-quantum secure blockchain framework with adaptive time consensus in cloud environments. *Scientific Reports*, *15*, Article 45090.