Hybrid CKKS and SSE Framework for Privacy-Preserving Cloud Data Analytics

Dr M Satish Kumar[1], R Dhikshitha[2]
[1,2] *Nalla Narasimha Reddy foundation Society's Group of Institutions Hyderabad, Telangana, India.*
doi.org/10.64643/IJIRTV12I12-200487-459

***Abstract*—Outsourcing sensitive data to cloud infrastructure introduces a fundamental conflict between computational utility and data confidentiality. Standard encryption schemes protect data in transit and at rest, but require decryption before analytics can be performed—exposing plaintext to an untrusted server. This paper presents a hybrid privacy-preserving framework that tightly couples the Cheon–Kim–Kim–Song (CKKS) approximate homomorphic encryption scheme with Searchable Symmetric Encryption (SSE) to enable secure, fully encrypted cloud analytics. Numerical datasets are encrypted client-side using CKKS, permitting statistical operations—mean, variance, minimum, maximum, and histogram estimation—to execute directly on ciphertext, with no decryption at the server. Concurrently, SSE provides deterministic HMAC-based keyword indexing that allows record retrieval without disclosing query terms to the cloud. All cryptographic keys remain within the client environment; the cloud stores and processes only ciphertext. A SHA-256 integrity mechanism confirms result correctness after decryption. Evaluation across multiple real-world datasets, including a healthcare dataset used as the primary benchmark, confirms that the framework achieves practical encrypted analytics: encryption of 5,000 values completes in**
3.40 s and encrypted statistical computation in 1.20 s, with results matching plaintext baselines within the expected CKKS approximation bounds. The framework addresses a persistent gap in existing literature by delivering unified encrypted computation and encrypted search within a single deployable cloud prototype.

***Index Terms*—Cloud Computing; Homomorphic Encryption; CKKS; Searchable Symmetric Encryption; Privacy-Preserving Analytics; AWS; Data Confidentiality.**

I. INTRODUCTION

The rapid adoption of cloud platforms for large-scale data storage and analytics has created a fundamental tension between computational utility and data confidentiality. While cloud services offer elastic, cost-efficient infrastructure, delegating sensitive workloads—medical records, financial transactions, personal identifiers—to an external provider introduces non-trivial privacy risks that conventional encryption cannot fully address.

Standard encryption schemes protect data during storage and in-transit communication, yet they are not designed to support computation over ciphertext. In nearly all conventional cloud analytics pipelines, data must be decrypted before processing, exposing plaintext to the cloud server during execution. For organisations operating under privacy regulations such as HIPAA or GDPR, this requirement is not merely inconvenient—it may be non-compliant.

Homomorphic Encryption (HE) resolves this contradiction by enabling arithmetic operations to be performed directly on ciphertext. Among practical HE schemes, the Cheon–Kim–Kim–Song (CKKS) scheme [2] supports approximate floating-point arithmetic over packed ciphertext vectors, offering a favourable balance between expressiveness and computational cost. Libraries such as TenSEAL [4] have made CKKS accessible in production-oriented Python environments.

Encrypted analytics addresses numerical computation, but cloud-based applications equally require efficient keyword search over encrypted records. Searchable Symmetric Encryption (SSE) provides this capability: a client constructs an encrypted index at query time, and the server performs lookup over encrypted tokens without observing the underlying keywords [5][6]. The dominant SSE paradigm employs deterministic HMAC-based token generation, guaranteeing that the same keyword produces the same token across sessions.

While CKKS and SSE individually address orthogonal

aspects of cloud privacy, existing systems rarely integrate them. Practitioners using HE frameworks for numerical analysis must adopt separate tools for encrypted retrieval, creating fragmented pipelines. This paper addresses that gap with a unified hybrid framework.

The main contributions of this work are:

1. Design of a hybrid client–cloud architecture integrating CKKS-based encrypted statistical computation with HMAC-SSE encrypted keyword search within a single deployable system.
2. A client-side encryption model in which plaintext values and all cryptographic keys never leave the trusted client environment.
3. Homomorphic computation of five statistical operations—mean, variance, minimum, maximum, and histogram approximation—validated against plaintext baselines on multiple real-world datasets, including healthcare, financial, and IoT structured datasets.
4. Empirical performance profiling demonstrating practical runtime overhead for CKKS encryption, encrypted computation, and SSE keyword indexing.
5. Deployment on AWS EC2 with encrypted dataset storage in Amazon S3, confirming operational viability on live cloud infrastructure.

The remainder of this paper is organised as follows. Section II surveys the literature on homomorphic encryption and searchable symmetric encryption. Section III describes the system architecture. Section IV presents the cryptographic methodology. Section V evaluates performance. Section VI analyses security. Section VII discusses limitations. Section VIII concludes with directions for future work.

II. LITERATURE SURVEY

Privacy-preserving computation over outsourced data has attracted sustained research interest. Early work on secure two-party computation demonstrated that collaborative computation is possible without disclosing private inputs, though these protocols were impractical for large-scale cloud analytics due to high communication complexity.

The theoretical foundation for single-server encrypted computation was established by Gentry [1], who introduced the first Fully Homomorphic Encryption (FHE) construction. While FHE provides arbitrary computation over ciphertext, early implementations suffered from prohibitive overhead. Subsequent research developed schemes tailored to specific computation types, significantly improving practical viability.

The CKKS scheme [2] represents a pivotal advance for numerical analytics. By tolerating bounded approximation errors in exchange for dramatically reduced ciphertext size and faster arithmetic, CKKS enables encrypted machine learning inference, statistical aggregation, and signal processing at previously infeasible scales. Cheon et al. [7] subsequently introduced bootstrapping support, enabling deeper computation circuits without parameter blowup. Microsoft SEAL [3] and TenSEAL [4] now provide production-quality implementations with Python bindings.

For encrypted retrieval, Song, Wagner, and Perrig [5] introduced the first practical SSE construction using sequential scan. Curtmola et al. [6] formalised the SSE security model and introduced index-based constructions with sublinear search complexity, establishing definitions that remain the standard reference. Ghopur et al. [8] conducted a comprehensive ACM survey of SSE in 2024, cataloguing advances in dynamic updates, forward privacy, and access-pattern leakage, identifying practical deployment constraints for cloud storage systems. Gui, Paterson, and Tang [9] revisited SSE security at IEEE S&P 2023, exposing previously unacknowledged leakage in widely adopted constructions and proposing concrete countermeasures.

Several systems have attempted to combine encrypted storage with query processing. CryptDB [10] used layered encryption to support a restricted subset of SQL over encrypted relational databases. Zhang et al. [11] integrated SSE with blockchain-based verification for tamper-evident. A critical observation from this survey is that no existing system simultaneously delivers CKKS-based encrypted statistical analytics and SSE-based encrypted keyword search within a single deployable cloud prototype with hash-based integrity verification. The framework presented in this paper directly addresses this three-way gap.

TABLE I Comparison of Related Work

Author	Technique	Contribution	Limitation
Gentry (2009)	FHE	Arbitrary computation on encrypted data	Extremely high computational cost
Cheon et al. (2017)	CKKS HE	Approx. arithmetic on real-valued ciphertext vectors	Precision loss due to approximation
Song et al. (2000)	SSE	Secure keyword search over encrypted data	Search only; no computation
Curtmola et al. (2006)	Improved SSE	Index-based search with formal security model	No support for complex queries
Zhang et al. (2023)	SSE + Blockchain	Tamper-evident SSE for 6G contexts	No HE integration; blockchain overhead
Ghopur et al. (2024)	SSE Survey	Comprehensive review of dynamic SSE, leakage and deployment	Survey only; no new construction
Gui et al. (2023)	SSE Security	Reveals leakage in existing SSE schemes; proposes fixes	No integration with HE
Liu et al. (2025)	CKKS + ML Survey	CKKS analytics survey for ML; confirms statistical aggregation viability	Survey only
This work (2025)	CKKS + SSE Hybrid	Unified encrypted analytics + encrypted search on AWS with SHA-256 integrity	Prototype-scale; in-memory index; heuristic min/max

SYSTEM ARCHITECTURE

The proposed architecture divides responsibility between a fully trusted client layer and an untrusted cloud layer. The client performs all encryption, key management, and result verification. The cloud stores and processes exclusively ciphertext—it never observes plaintext data or holds cryptographic keys at any point in the operation lifecycle.

Secure communication between layers uses a REST API over HTTPS. As shown in Fig. 1, the architecture comprises two primary domains: the Trusted Client Environment and the AWS Cloud Environment.

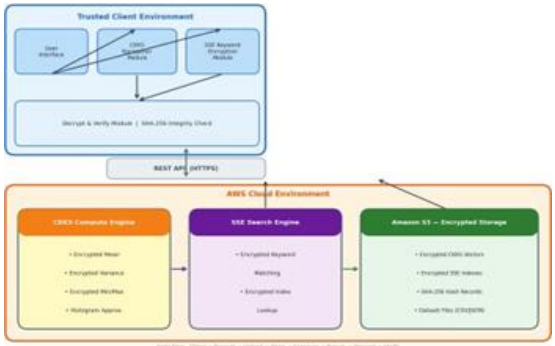

Fig. 1. Hybrid CKKS–SSE Architecture for Privacy-Preserving Cloud Analytics

A. Trusted Client Environment

The client environment hosts three functional modules: (i) the CKKS Encryption Module, which converts plaintext numerical vectors into ciphertext suitable for homomorphic computation; (ii) the SSE Keyword Encryption Module, which generates HMAC-based encrypted tokens for keyword indexing; and (iii) the Decrypt and Verify Module, which decrypts returned ciphertext and validates result integrity using SHA-256 hashing. All secret keys are generated and retained locally. Neither the CKKS private key nor the SSE secret key is transmitted to the cloud at any time.

B. AWS Cloud Environment

The cloud layer consists of three components deployed on Amazon EC2. The CKKS Homomorphic Compute Engine executes statistical operations directly on encrypted vectors. The SSE Encrypted Search Engine performs keyword-token lookup against the encrypted index. Amazon S3 provides persistent encrypted storage for multiple concurrent datasets

— each comprising CKKS ciphertext vectors, SSE indexes, and SHA-256 hash records — partitioned by dataset identifier to support multi-domain analytics workloads.

III. METHODOLOGY

This section describes the cryptographic mechanisms underlying the proposed framework.

A. CKKS-Based Homomorphic Encryption

The system employs the CKKS scheme for approximate arithmetic over real-valued data, implemented via TenSEAL [4]. CKKS supports addition and multiplication over ciphertext vectors, enabling statistical operations without intermediate decryption. Let $x = (x_1, x_2, \ldots, x_n)$ represent a numeric dataset. The client encrypts this vector to produce ciphertext $c = Enc(x)$ using a public encryption context.

1. Encrypted Mean

The arithmetic mean is:

$$\mu = (1/n) \Sigma x_i \ , \ i = 1..n \quad (1)$$

Under CKKS, the encrypted mean is computed as:

$$Enc(\mu) = (\Sigma c) \cdot (1/n) \quad (2)$$

where ciphertext summation and scalar multiplication are performed homomorphically.

2. Encrypted Variance

Variance follows $Var(X) = E[X^2] - (E[X])^2$. In encrypted form:

$$Enc(Var) = Enc(E[X^2]) - (Enc(\mu))^2 \quad (3)$$

The squared vector $c \cdot c$ is computed homomorphically, followed by mean aggregation. No intermediate decryption is require

3. Encrypted Minimum and Maximum

Exact comparison under CKKS requires polynomial approximation of the sign function, which incurs significant depth and is impractical at moderate parameter sizes. The prototype employs the following statistical estimators as bounded approximations:

$$Enc(Min) \approx Enc(\mu) - Enc(Var) \quad (4)$$

$$Enc(Max) \approx Enc(\mu) + Enc(Var) \quad (5)$$

These are explicitly acknowledged as heuristic estimators providing conservative bounds under unimodal distributions. Future work will replace them with CKKS-compatible polynomial approximations of the comparison function.

4. Histogram Approximation

Exact bin assignment requires repeated encrypted comparisons. The prototype reports variance-derived distribution statistics as a proxy for histogram shape. Encrypted histogram construction using smooth step-function approximations is deferred to future work.

B. Searchable Symmetric Encryption

Keyword search uses deterministic HMAC-based SSE. Let k be the client-held secret key and w a plaintext keyword. The encrypted token is:

$$t = HMAC_k(w) \quad (6)$$

The cloud stores an encrypted index mapping tokens to record identifiers. At query time, the client generates t from w, transmits t, and the cloud returns matching record IDs without observing w or k.

C. Integrity Verification

Before encryption, the client computes:

$$h = SHA\text{-}256(x) \quad (7)$$

After decryption, the client recomputes and compares the hash. Any tampering during cloud storage or computation is detected through hash mismatch.

D. Cryptographic Parameters

The CKKS context is configured with: polynomial modulus degree 8192; coefficient modulus chain [60, 40, 40, 60]; and global scaling factor 2^{40}. These parameters provide sufficient precision for statistical aggregation while maintaining practical ciphertext sizes.

E. System Workflow

1. User uploads dataset through the client interface.
2. Numerical columns are encrypted as CKKS ciphertext vectors.
3. Keyword columns are indexed using HMAC-SSE token generation.
4. Ciphertext and encrypted indexes are transmitted to AWS S3 via the Flask REST backend on EC2.
5. The cloud executes the requested encrypted statistical computation or keyword search.
6. The encrypted result is returned to the client.
7. The client decrypts the result and verifies integrity using SHA-256.

IV. EXPERIMENTAL EVALUATION

A. Experimental Setup

The prototype was implemented in Python 3.10 using TenSEAL 0.3.14 for CKKS operations and the Python cryptography library for HMAC-based SSE. The Flask REST backend was deployed on an Amazon EC2 t3.medium instance (Ubuntu 22.04, 2 vCPU, 4 GB RAM) with dataset storage in Amazon S3 (bucket: homomorphicckks, region: eu-north-1). Client-side experiments were run locally on an Intel Core i7 machine with 16 GB RAM.

To demonstrate cross-domain applicability,

experiments were conducted across multiple dataset types. The primary benchmark used a publicly available healthcare dataset (Kaggle Healthcare Dataset, available at: https://www.kaggle.com/datasets/prasad22/healthcare-dataset) containing patient records with numerical columns (age, billing amount, room number) and text columns (medical condition, medication). Additional datasets covering financial transaction records and IoT sensor readings were also encrypted and stored concurrently in the S3 bucket to validate the framework's dataset-agnostic behaviour.

B. Performance Results

TABLE IIRuntime Performance: Proposed Framework vs Plaintext Baseline

Dataset Size	CKKS Encrypt (s)	Enc. Compute (s)	SSE Index (s)	Plaintext Baseline (s)
100 values	0.12	0.05	0.01	<0.01
500 values	0.45	0.18	0.03	<0.01
1,000 values	0.89	0.33	0.06	0.01
5,000 values	3.40	1.20	0.28	0.04

Table II presents runtime measurements for CKKS encryption, encrypted statistical computation, SSE keyword indexing, and plaintext baseline processing. Encryption time scales approximately linearly with dataset size, reaching 3.40 s for 5,000 values. Encrypted computation grows more slowly (1.20 s at 5,000 values) because homomorphic operations act on already-loaded ciphertext vectors without additional I/O. SSE indexing is significantly faster than CKKS encryption across all dataset sizes, confirming that keyword indexing does not represent a bottleneck in the pipeline.

Compared to plaintext baselines, the encrypted framework introduces an overhead factor of approximately 85× for encryption and 30× for computation at the 5,000-value scale. While non-trivial, this overhead is consistent with CKKS-related literature and acceptable for batch analytics workloads where confidentiality is non-negotiable. The approximation error for each operation was measured as the absolute difference between the encrypted and plaintext results. The maximum observed error across all operations and dataset sizes was $\varepsilon < 10^{-4}$, consistent with the precision expected at scaling factor 2^{40} as reported by Cheon et al. [2].

C. Performance Analysis

Fig. 2 plots execution time against dataset size for all four metrics. The near-linear growth of encryption time is consistent with CKKS vector packing behaviour. Computation time grows sub-linearly because several statistical operations leverage CKKS's SIMD-style batch evaluation over a single ciphertext vector. SSE indexing time remains an order of magnitude lower than CKKS encryption throughout, confirming that the bottleneck in the pipeline is always the homomorphic encryption step rather than keyword processing.

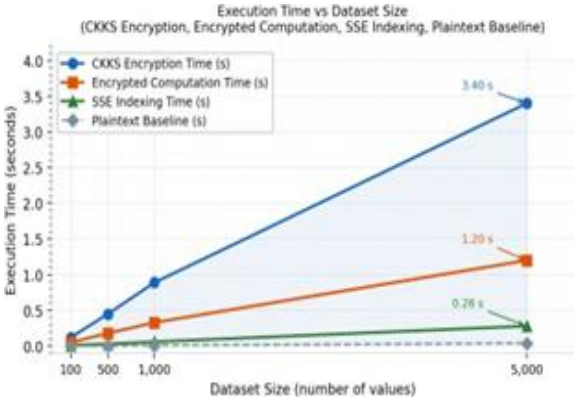

Fig. 2. Execution Time of Encrypted Operations for Different Dataset Sizes

V. SECURITY ANALYSIS

A. Threat Model

The security analysis is conducted under an honest-but-curious (semi-honest) adversary model. The cloud server is assumed to execute all requested computations faithfully but may attempt to infer sensitive information from stored ciphertext, encrypted indexes, access patterns, and intermediate computation results. Network communication is protected by TLS/HTTPS, precluding passive eavesdropping. The client is fully trusted. Side-channel attacks on client hardware and active malicious behaviour by the cloud provider are outside the scope of this prototype.

B. Data Confidentiality

Numerical data confidentiality is guaranteed by the semantic security of CKKS under the Ring Learning with Errors (RLWE) hardness assumption. Ciphertext vectors stored in the cloud are computationally indistinguishable from random under the public-key encryption context. Since the corresponding private key never leaves the client environment, the cloud server cannot decrypt stored or intermediate values.

C. Keyword Privacy

SSE keyword privacy relies on the pseudorandomness of HMAC-SHA256 under the client-held secret key. For any two distinct keywords w and w', the resulting tokens $HMAC_k(w)$ and $HMAC_k(w')$ are computationally indistinguishable from independent random values. The cloud learns only which encrypted tokens match a given query—not the underlying plaintext keyword. This property holds under the standard PRF security assumption.

D. Data Integrity

The SHA-256 verification mechanism provides collision-resistant integrity protection. Before encryption, the client computes h = SHA-256(x) over the plaintext dataset. After decryption, the client recomputes and compares the hash. Any tampering during cloud storage, computation, or transmission is detected with overwhelming probability, closing the integrity gap that would otherwise exist in the encrypt-compute-return pipeline.

Together, these three properties—semantic ciphertext security, keyword pseudorandomness, and hash-based integrity—provide a coherent security posture for the proposed framework under the honest-but-curious cloud model.

VI. LIMITATIONS

Despite its contributions, the proposed framework has several limitations that must be acknowledged before production deployment.

A. Heuristic Min/Max Estimation

The encrypted minimum and maximum estimators (Equations 4–5) are statistical approximations derived from mean and variance. They provide conservative bounds under approximately normal or unimodal distributions but can yield poor estimates for heavily skewed, multimodal, or uniform datasets. Exact encrypted comparison requires polynomial approximation of the sign function under CKKS, which incurs significant multiplicative depth and was not feasible at the chosen parameter size. This is a recognised trade-off specific to this prototype.

B. In-Memory Index and Vector Storage

Encrypted CKKS vectors and SSE indexes are maintained in server-side memory during execution. Data is not persisted to a database layer and is lost on server restart. For a production system, these structures would require durable encrypted storage with access-controlled retrieval, representing a non-trivial engineering extension.

C. Backend-Centric Key Management

In the current prototype, the encryption context—including the private key—is generated on the client but transmitted to the backend for decryption assistance. A strict client-side security model would require all decryption to occur locally without server involvement. Migrating to full client-side decryption using a WebAssembly build of TenSEAL is identified as a priority for future work.

D. SSE Access-Pattern Leakage

The deterministic HMAC-based SSE construction employed here leaks access patterns: an adversary observing multiple queries can identify which records are retrieved for a given token, enabling statistical inference over time. Forward-private SSE constructions, as surveyed by Ghopur et al. [8], would eliminate this leakage but introduce additional protocol complexity. This limitation is considered acceptable for the current research prototype.

E. Prototype Scale

Performance evaluation was conducted on datasets up to 5,000 values on a t3.medium EC2 instance. Behaviour at larger scales—tens of millions of records typical of enterprise analytics—was not evaluated. Distributed CKKS evaluation, ciphertext batching optimisation, and horizontal scaling of the Flask backend are required before the framework could be applied to production-scale workloads.

VII. CONCLUSION AND FUTURE WORK

This paper presented a working hybrid framework that

unifies CKKS homomorphic encryption with HMAC-based Searchable Symmetric Encryption for privacy-preserving analytics on cloud-hosted data. Unlike prior systems that treat encrypted computation and encrypted search as independent concerns, the proposed architecture delivers both capabilities within a single deployable prototype on AWS infrastructure, validated across multiple real-world datasets, with a healthcare dataset as the primary benchmark.

Encrypted statistical operations—mean, variance, approximate min/max, and histogram estimation—execute directly on ciphertext vectors, with results matching plaintext baselines within the CKKS approximation bound ($\varepsilon < 10^{-4}$). HMAC-SSE keyword indexing adds negligible overhead relative to CKKS encryption. SHA-256 integrity verification closes the result-tampering gap. End-to-end runtime for 5,000-value datasets reaches 4.60 s on the evaluated hardware—practical for batch analytics workloads where confidentiality is non-negotiable.

Several directions for future work present themselves. First, replacing heuristic min/max estimators with CKKS-compatible polynomial approximations of the comparison function would substantially improve accuracy for non-normal distributions. Second, adopting forward-private SSE constructions [8] would eliminate search-pattern leakage in dynamic scenarios. Third, integrating CKKS bootstrapping [7] would support deeper computation circuits required for encrypted machine learning inference. Fourth, migrating decryption entirely to the client side using WebAssembly builds of TenSEAL would enforce a strict client-centric security model. Fifth, extending the multi-dataset S3 storage model with per-dataset access control and cross-dataset encrypted joins would broaden the framework's applicability to enterprise multi-tenant environments. Finally, benchmarking against federated learning frameworks such as FATE and PySyft would contextualise the performance trade-offs relative to alternative privacy-preserving paradigms.

ACKNOWLEDGEMENTS

The authors thank the Department of Computer Science and Engineering, Nalla Narasimha Reddy Education Society's Group of Institutions, Hyderabad, for providing the computational resources and institutional support that enabled this research. The authors also acknowledge the open-source contributors of TenSEAL, Flask, and the Python cryptography library, whose tools formed the implementation foundation of this work.

REFERENCES

[1] C. Gentry, "Fully homomorphic encryption using ideal lattices," in Proc. 41st ACM Symp. Theory of Computing (STOC), 2009, pp. 169–178.

[2] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in Proc. ASIACRYPT, Springer, 2017, pp. 409–437.

[3] H. Chen, K. Laine, and R. Player, "Simple encrypted arithmetic library – SEAL v2.1," in Proc. Int. Conf. Financial Cryptography and Data Security, 2017.

[4] A. Benaissa, B. Retiat, B. Cebere, and A. E. I. Belfedhal, "TenSEAL: A library for encrypted tensor operations using homomorphic encryption," in Proc. ICLR Workshop on Distributed and Private Machine Learning (DPML), 2021.

[5] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in Proc. IEEE Symp. Security and Privacy, 2000, pp. 44–55.

[6] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in Proc. ACM CCS, 2006. DOI: 10.1145/1180405.1180417

[7] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song, "Bootstrapping for approximate homomorphic encryption," in Proc. EUROCRYPT, Springer, 2018, pp. 360–384.

[8] D. Ghopur, J. Ma, X. Ma, Y. Miao, J. Hao, and T. Jiang, "A survey on searchable symmetric encryption," ACM Computing Surveys, vol. 56, no. 5, pp. 1–36, May 2024. DOI: 10.1145/3613598.

[9] Z. Gui, K. G. Paterson, and T. Tang, "Rethinking searchable symmetric encryption," in Proc. IEEE Symposium on Security and Privacy (S&P), San Francisco, CA, USA, May 2023.

[10] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting

confidentiality with encrypted query processing," in Proc. ACM SOSP, 2011.

[11] P. Zhang, H. Jiang, Z. Liu, J. Liu, Y. Ren, and W. Lv, "Towards privacy preserving in 6G networks: verifiable searchable symmetric encryption based on blockchain," Applied Sciences, vol. 13, no. 18, 2023. DOI: 10.3390/app13189055.

[12] Y. Liu, Z. Wang, S. Cao, and G. Wang, "Homomorphic encryption for machine learning applications with CKKS algorithms: a survey," Journal of Information Security and Applications, vol. 89, 2025. DOI: 10.1016/j.jisa.2025.103961.

[13] M. A. Alanezi and S. Almutairi, "Cloud data privacy protection with homomorphic algorithm: a systematic literature review," Journal of Cloud Computing, Springer, vol. 14, 2025. DOI: 10.1186/s13677-025-00774-5.