ISSN : 2455-135X

A Model for Enhancing Data Privacy in Cloud-Server Environment using Homomorphic Encryption

Akanwa A. O[1], Matthias D[2], Nwiabu N. D[3], Taylor O. E[4]

[1,2,3&4]*Department of Computer Science, Rivers State University, Port-Harcourt,*

Abstract - In recent years, multi-tenant cloud computing has become the default architecture for modern data services, yet the same shared infrastructure that makes it cost-effective also creates a persistent privacy risk, due to tenant data is typically encrypted only at rest and in transit, leaving it exposed in plaintext the moment it is processed. This gap creates opportunities for unauthorized access through misconfigured permissions, compromised hypervisors, or malicious co-tenants. This study addresses this gap by designing and implementing a cryptographic data isolation model using Homomorphic Encryption (HE) to protect tenant data throughout its entire lifecycle, including during computation. The model's design is grounded in the Brakerski-Fan-Vercauteren (BFV) scheme for lattice-based, quantum-resistant key management, addition, and multiplication over ciphertext, and was implemented and evaluated using the Cheon-Kim-Kim-Song (CKKS) homomorphic scheme within a Python application layer backed by a MySQL database Server. Furthermore, homomorphic key pairs were generated and used to encrypt and decrypt both user profile and financial transaction data, with encryption and decryption times measured across ciphertext sizes ranging from 4,800 to 50,000 bytes. Experimental results show that encrypted values remained completely unreadable regardless of message length, confirming strong confidentiality, while encryption and decryption times scaled predictably with data size, from 0.35ms/0.12ms at 4,800 bytes to 3.6ms/1.9ms at 50,000 bytes. A comparative analysis against Rivest-Shamir-Adleman (RSA), Data Encryption Standard (DES), Advanced Encryption Standard (AES), and Elliptic Curve Cryptography (ECC) further showed that although HE carries a heavier computational cost, as it uniquely allows computation directly on encrypted data, making it a suitable foundation for privacy-preserving multi-tenant cloud systems.

Keywords - *Cryptography, Homomorphic Encryption, Cloud Computing, Data Privacy Preservation, Server-Side Attacks.*

1. Introduction

The rapid migration of enterprise server workloads to cloud computing has fundamentally reshaped how organizations store, process, and share data [1]. Cloud service providers now host information belonging to hundreds or even thousands of distinct clients on shared physical infrastructure, this is a design principle commonly referred to as multi-tenancy. Multi-tenancy is, in many respects, the economic engine of cloud computing, as it allows providers to pool compute, storage, and network resources across tenants, driving down cost and improving scalability for everyone involved [2]. However, this architectural convenience comes with a lingering vulnerability. The same servers, hypervisors, and databases that make the cloud so efficient are also the very seams through which one customer's data can spill into another's, whether by design flaw, human error, or malicious intent.

Data privacy in this setting is not a peripheral concern; it is central to whether cloud computing can be trusted with sensitive information at all [3]. Financial institutions, healthcare providers, and government agencies are increasingly required by regulation to demonstrate that tenant data is isolated not just logically, but cryptographically, from other tenants and from the cloud provider itself [4]. In practice, however, most commercial multi-tenant systems still rely on access control lists, network segmentation, and encryption which is only applied to data at rest or in transit. The moment data needs to be processed i.e., queried, aggregated, or analyzed; it is typically decrypted somewhere inside the provider's infrastructure [5]. That decryption point is precisely where a curious administrator, a misconfigured permission, a compromised hypervisor, or a malicious co-tenant can gain access to information that was never meant to leave its owner's control.

In recent years, Homomorphic Encryption (HE) has offers a way out of this dilemma. Unlike conventional cryptographic schemes, homomorphic encryption allows arithmetic and logical operations to be carried out directly on ciphertext, producing an encrypted result that, once decrypted by the data owner, matches the result of the same operations performed on the original plaintext [6]. In other words, the cloud server can compute over data it can never actually read. For a multi-tenant environment, this property is transformative as provider can run analytics, aggregate queries, or shared computations across tenant workloads without the underlying data ever being exposed in usable form to the provider, to other tenants, or to an attacker who breaches the server [7].

However, despite this promise, existing multi-tenant cloud systems lack efficient, privacy-preserving mechanisms capable of ensuring secure data isolation and preventing unauthorized access to sensitive tenant data during both storage and transmission. Most privacy controls in production cloud platforms are enforced at the access-control layer rather than the cryptographic layer, meaning that isolation between tenants depends on correctly configured software rather than on mathematical guarantees [8].

ISSN : 2455-135X

This study sets out to build a cryptographic data isolation model using homomorphic encryption. The idea is to protect tenant data not just at rest or in transit, but while it's being processed. The model was implemented using Python programming language, with MySQL Server as the underlying data store, so that tenant records can be encrypted, stored, queried, and analyzed while staying cryptographically isolated from other tenants throughout their entire lifecycle.

2. Materials and Methods

Technically, the growing body of recent literature has examined how homomorphic encryption can be applied to secure data in cloud and multi-tenant settings, and reviewing this work helps situate the contribution of the present study. [9] proposed HE-AO, which is considered to be an optimization-based encryption approach purpose-built for data delivery in multi-tenant cloud environments. Their study observe that although multi-tenancy is one of cloud computing's chief economic advantages, existing mechanisms for securing data delivery in shared environments repeatedly fall short on confidentiality, authenticity, and integrity. Their model applies homomorphic encryption to encode data before it leaves the cloud server, reporting improvements in delivery time alongside stronger confidentiality guarantees compared to prior schemes. The work is valuable in confirming that multi-tenant data delivery is a distinct problem from general cloud storage encryption, but it stops short of addressing how encrypted data can be queried or processed once inside the data store.

[10] presented an applied study on securing cloud environments with homomorphic encryption, examining both partially and fully homomorphic schemes and their trade-offs for practical cloud deployment. The study's contribution lies mainly in clarifying the operational difficulties organizations face when attempting to adopt homomorphic encryption at scale including computational cost, ciphertext expansion, and the difficulty of mapping real business operations onto the limited operation sets that many schemes support. While informative as a practitioner-oriented account of the barriers to adoption, the work remains largely descriptive, as it does not propose or implement a concrete architecture, leaving open the practical question of how these barriers might be engineered around in an actual multi-tenant deployment, which is precisely the space this study occupies.

[11] took a more implementation-focused route with LightPHE, a Python library that integrates partially homomorphic encryption schemes and benchmarks their performance across a range of cloud computing environments, from GPU-backed instances to lower-cost cloud setups. Their evaluation is particularly relevant here because it demonstrates, empirically, that homomorphic encryption can be embedded into a Python-based application layer with manageable performance costs depending on the deployment environment chosen. This supports the technology choices adopted in the present study, though their work is concerned with library-level benchmarking rather than with tenant isolation, access boundaries, or database integration.

[12] turned their attention to multi-key homomorphic encryption, identifying a previously unrecognized vulnerability in an existing multi-key scheme (referred to as CDKS) when it is applied to multiparty secure computation tasks such as privacy-preserving federated learning. They found that the existing construction could inadvertently leak plaintext information between parties who were each supposed to remain cryptographically isolated from one another, and proposed a corrected scheme, SMHE, that closes this leakage while adding only modest computational overhead. This work is a useful caution for any multi-tenant design that assumes multi-key homomorphic schemes provide isolation by default: isolation guarantees need to be verified rather than assumed.

[13] addressed a different but related bottleneck: the client-side computational burden that fully homomorphic encryption imposes when used for privacy-preserving machine learning, particularly on constrained or edge devices. Their hybrid homomorphic encryption approach known as "HHEML" combines symmetric encryption with Full Homomorphic Encryption (FHE) so that the expensive homomorphic operations are shifted toward the server while the client performs only lightweight symmetric encryption, reporting substantial reductions in client-side latency in their experiments. Although their target environment is edge inference rather than multi-tenant cloud storage.

At a broader level, [14] conducted a systematic literature review of homomorphic encryption research applied to cloud data privacy protection. Their review found that FHE is by a wide margin the most studied variant in the literature, but also confirmed that high computational overhead and implementation complexity remain the two most frequently cited obstacles to real-world deployment, particularly for latency-sensitive or real-time applications. This review is useful as a map of the field, but by design it surveys existing work rather than proposing a new architecture, and it does not focus specifically on multi-tenant isolation as a distinct requirement separate from general cloud data protection.

[15] proposed a privacy-protection optimization method for cloud platforms that combines federated learning with homomorphic encryption, explicitly targeting the privacy risks that arise from parameter interaction across multi-tenant, heterogeneous, high-concurrency cloud nodes during distributed model training. Their experiments reported meaningful reductions in both encryption and decryption time and in the number of communication rounds required, while maintaining high model accuracy and tenant participation rates. Their study reinforces that multi-tenant privacy risk is not confined to raw data storage but extends to any process, including collaborative computation that moves information across tenant boundaries, a principle that

ISSN : 2455-135X

shapes how the present study defines the scope of the privacy problem it is solving.

[16] introduced an innovative approach by advocating for the use of public key Elliptic Curve Cryptography (ECC) at the Data Link Layer, rather than relying on traditional Media Access Control (MAC) methods. The authors implemented a prototype of their proposed architecture using the microFourQ-MSP-IAR Embedded Workbench IDE-MSP430 7.12.4, demonstrating the practical applicability of their approach. Their findings indicate a direct correlation between the cost of key generation and the characteristics of the Elliptic Curve employed; specifically, the time required for key generation increases in proportion to the bit length of the curve. Given the proliferation of connected devices, the lack of effective security measures at the Data Link Layer raises significant concerns regarding network privacy, governance, and overall security. Their research underscores the necessity for robust security protocols, suggesting that integrating a public key cryptosystem based on Elliptic Curves can provide a more secure framework for data communication among Internet-connected devices. Their research study contribute d to the ongoing discourse on enhancing cybersecurity measures and offers a promising avenue for future research in securing lower layers of the OSI model.

3. Methodology and Design

Isolating user, device or cloud server data is a needed practice of the proposed system for keeping sensitive information secured from unauthorized access and ensuring that data privacy is upheld. This practice is important in cloud environments where multiple tenants share resources, thereby increasing the risk of data breaches.

This study adopts homomorphic encryption technique to design and develop a robust data isolation model for preserving privacy in cloud-server settings. Here, the primary objective is to maintain the confidentiality, integrity, and availability of sensitive data in cloud servers. The design of the proposed model leverages the Brakerski-Fan-Vercauteren (BFV) homomorphic cryptographic scheme to protect data both at rest and in transit. This ensures that unauthorized access is prevented even during processing.

The BFV scheme is well-suited for applications requiring precise computations. This scheme is built on strong mathematical foundations, specifically lattice-based cryptography, which is considered secure against quantum attacks. The main reason for adopting the BFV Scheme is due to its ability to securely compute encrypted data without exposing it to other users. This enables secure data processing in the developed model, ensuring that even if multiple entities share the same infrastructure, while their data remains isolated and protected.

In this study, the data isolation procedure is implemented in the following steps:

Step 1 (Key Management): Key management is critical in the proposed study as it ensures the security and effectiveness of Homomorphic Encryption. Here, the key generation process involve creating a public and private key pair. The first step is to select parameters n(degree of the polynomial), q(modules), and t(the number of slots for encoding). Next is to select a secret polynomial s(x) randomly from a polynomial ring. This will be achieved as follows:

$$\frac{\mathbb{Z}[x]}{(x^n+1)} \tag{1}$$

In terms of generating the Public Key, we will choose a random error polynomial $e(x)$, and compute the public key HE_{PubKey} as follows:

$$HE_{PubKey} = \big(a(x), b(x)\big) = (r(x) \cdot s(x) + e(x) \ \ mod\ q) \tag{2}$$

Where $r(x)$ represent another random polynomial.

The HE key pair will be achieved as $HE_{PriKey} = s(x)$ for private key, and $HE_{PubKey} = (a(x), b(x))$ for public key.

Step 2 (Data Encryption): During the data encryption phase, the plaintext or user data is transformed into ciphertext using the public key. This step ensures that all sensitive data is encrypted with the adopted BFV HE scheme before it is stored or processed. This will effectively be safeguarding the information from exposure. For this study, it is achieved by representing the plaintext m as a polynomial $m(x)$ in the polynomial ring. Next, we selected a random polynomial $r(x)$ and compute the ciphertext as follows:

$$HE_{Cipher} = (c_1, c_2) = (b(x) + r(x) \cdot q, a(x) + r(x) \cdot s(x) + m(x) \ \ mod\ q) \tag{3}$$

Where HE_{Cipher} represent the generated HE ciphertext, c_1 and c_2 represents the components of the ciphertext

Step 3 (Data Decryption): In the Data Isolation Mechanism, the study utilizes the private key to retrieve the plaintext message from the ciphertext. However, once the necessary computations are completed, the result is decrypted to obtain the final output. This step ensures that sensitive data in cloud server remains confidential throughout the processing phase. In this study, we define the following to achieve this process.

Let $HE_{Cipher} = (c_1, c_2)$ be the received ciphertext. The intermediate value is then computed as follows:

$$u(x) = c_2 - c_1 \cdot s(x) \quad mod\ q \tag{4}$$

ISSN : 2455-135X

The plaintext polynomial $m(x)$ is retrieved as follows:

$$m(x) = u(x) \ mod \ q \quad (5)$$

Note, in the context of the BFV (Brakerski-Fan-Vercauteren) Homomorphic Encryption scheme adopted for this study's Data Isolation Mechanism, the addition and multiplication of ciphertexts are crucial for performing operations on encrypted data without exposing the underlying plaintext. Here, this is achieved using the mathematical equation as follows:

Homomorphic Addition: Given two ciphertexts c_1 and c_2:

$$c_1 = (c_{1,1}, c_{1,2}) = (b_1 + r_1 \cdot q, a_1 + r_1 \cdot s + m_1) \quad mod \ q \quad (6)$$

$$c_2 = (c_{2,1}, c_{2,2}) = (b_2 + r_2 \cdot q, a_2 + r_2 \cdot s + m_2) \quad mod \ q \quad (7)$$

The addition operation on the ciphertext is achieved as follows:

$$HE_{AddOperation} = c_1 + c_2 = (c_{1,1} + c_{2,1}, c_{1,2} + c_{2,2}) \quad mod \ q \quad (8)$$

This result in:

$$HE_{AddOperation} = ((b_1 + b_2 + (r_1 + r_2) \cdot q), (a_1 + a_2 + (r_1 + r_2) \cdot s + (m_1 + m_2)) \quad (9)$$

We decrypted the cipher addition operation using the decryption formula as follows:

$$Decrypt(HE_{AddOperation}) = m_1 + m_2 \quad (10)$$

Homomorphic Multiplication: Given the multiplication of the ciphertexts c_1 and c_2. This is performed as follows:

$$HE_{MultOperation} = c_1 \cdot c_2 = (c_{1,1} \cdot c_{2,1}, c_{2,2} + c_{1,2} \cdot c_{2,1}) \quad mod \ q \quad (11)$$

Here, the first component is calculated as:

$$HE_{MultOperation,1} = c_{1,1} \cdot c_{2,1} = (b_1 + r_1 \cdot q) \cdot (b_2 + r_2 \cdot q) \quad (12)$$

Further, the second component is the computed as:

$$\begin{aligned} HE_{MultOperation,2} &= c_{1,1} \cdot c_{2,2} + c_{1,2} \cdot c_{2,1} \\ &= (b_1 + r_1 \cdot q) \cdot (a_2 + r_2 \cdot s + m_2) \\ &\quad + (a_1 + r_1 \cdot s + m_1) \end{aligned}$$

After performing the polynomial multiplication, we reduce each component by modulo of q. The encrypted result is then represented as:

$$Decrypt(HE_{MultOperation}) = m_1 \cdot m_2 \quad (13)$$

For this study, these operations are deemed essential for maintaining data integrity and confidentiality within the Data Isolation Mechanism, allowing for secure computations on encrypted data in cloud environments.

The integration of HE into the Data Isolation Mechanism enhances multiple layers of security (Data in Transit and Data at Rest) for in proposed system. In terms of Data at Transit, when data is been transmitted between services, it will remain encrypted, thereby preventing interception and unauthorized access. For Data at Rest, our Data Isolation Mechanism ensures that stored data in cloud servers are encrypted. This ensures that even if the storage system is compromised, sensitive information will remain protected.

The proposed Data Isolation Mechanism is designed utilizing Homomorphic Encryption (HE) and robust key management practices. This is essential for ensuring the confidentiality and integrity of sensitive data in cloud computing environments. By allowing computations on encrypted data while securely managing keys, this mechanism will provide robust protection against unauthorized access and maintains compliance with data protection regulations. Implementing this mechanism will enhance the overall security posture, safeguarding sensitive information against evolving cyber threats and ensuring trust in cloud services.

4. Results and Discussion

Presented in this section are the results of the proposed system. The studies utilized Homomorphic Encryption (HE) technique for designing the system and enhance data privacy (both in-transit and at-rest). The study employed the CKKS homomorphic cryptographic scheme, which supports computations on encrypted floating-point data without requiring decryption. This adds an extra layer of security for both data-in-transit and data-at-rest.

The CKKS is known to be a powerful property of the homomorphic cryptosystem and is employed by various robust cryptographic systems to enhance security of data in cloud settings. The system generates homomorphic key pairs, which are used for encryption and decryption.

The result of the generated key pairs and their representations is presented in Table 1. While Fig 1 represents a screenshot proof of the generated key pairs (Public and Secret Key) from the developed system.

ISSN : 2455-135X

Table 1 Generated Homomorphic Public and Secret Key Pair

Key-Pair Name	Public Key (Length in Byte: 80)	Secret Key (Length in Byte: 112)
Allweel-Keys	XqEQBAECAAD8KgcAAAAAACi1L/2 gYQAIAKxdDm79X+hyLRCgvGZ44v2J xm8pxI3LfqUR0ONLqwz/UtONf/xTEQ YJOM5FXRDZ3d3d3d29U4WElYQvY WMnnOIC19RUxTg6AL3CbbGIrIw06p KmTwoD0qU3VLo2sBfIQPZdEY19T8A 5XdLLdthmAEJmIoidzGictm/iNLHFdsO TbZTu4qBsWxUfpDSCZdtLegVsQEBR SEvoiZIMg5BQ2AbiqFO1wGsAVk0gD 5oOMXi8x7tliJ5DKzmSpX9Y8QMMZz AN6RwOw3vEQQL3A4BAQ==	XqEQBAECAAB6lwMAAAAAACi1L/2gSA AEAHRcDp7+H+ByLRCgvGZHqLSKIan4g/c 9RwBq1IKpeq8GwHkEARLPkvS7ULCNTi67 u7u7u7t7p2qEhoRPYXwASRcrqVGiUFhlretj mCKH3uB3UtmIL1ukSFnF6CDUoyKH9boCu +DOn7VMlcmVpI4OmlvrDyZtzrBJrMIlrFQA ShYMGVVNX5k3qIpYbCkN8lEhS4hplEujxi O7A1xRvCBhNO25lMUb2Uh24jsGoEoIzBlG BbS4eAtVURxBF2YG1hAJIguSYLTWRVEQ Ml1hM4dL5

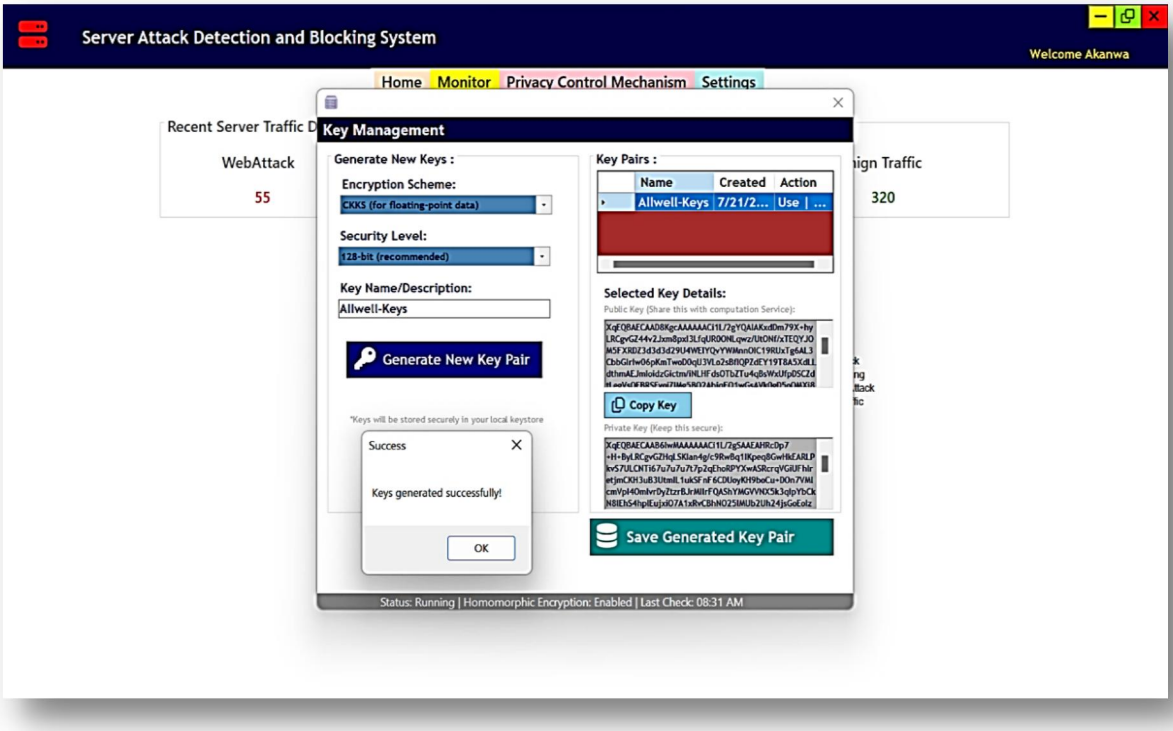

Fig 1: Homomorphic Encryption Key-Pair (Public and Private Key Generation

Fig 1 captured the homomorphic key generation process of the developed system. It captures the key management form used for generating the key-pair. A user is required to select the encryption scheme from the provided combo box which in this case, the CKKS scheme was selected, and also the user is expected to fill in the key name and description before clicking on the Generate New Keys button. When clicked, the public and private key pair will be displayed in their respective textboxes as shown in Figure 1 above.

To evaluate the effectiveness of the homomorphic encryption (CKKS scheme) in preserving privacy, the system was tested using plain text messages. Encryption was performed using the generated public key, while the decryption was achieved using the private key. The results of the original plaintext of both user and sample transaction data together with that of their encrypted ciphertext are presented in Table 2.

ISSN : 2455-135X

Table 2 Sensitive Original Plain Text vs. Encrypted Ciphertext

Field	Plaintext Message	Encrypted Message (CKKS Format)	Ciphertext Size
User Data	Fullname: Akanwa Allwell Ononiwu National ID: 2567894321098765 BVN: 22445566778 PhoneNumber: +2348032330317 Email: allwell.akanwa@email.com Gender: Male BloodGroup: O+ MedicalCondition: Asthma InsuranceNumber: HMO-227874567	XqEQBAECAABnGAUAAAAAACi1L/2gYQAGADxaDo79n81yLRCgvGaWGJVVjiZeebXvYUV62uCkbil5AOt1vPt8azkyFvs7My2yu7u7u7t7p1mEfIQiYRnyHIGAiWIBCZH5q0O0LbNFz5YCdxI52aSkRU++Ph4BHQ2dit55spnCTyJEJnuI2yaqlZOsFDTrz5fLo7w1Pe9DEZ3IyjpmPtdZxX3zT8x4ISlpRBU7V2rvf1ImyMgF1oq6uLdCLp1lVLno5yy5aAbX3bgSN16zxSglrqH2ZY6LYl8po57vLOJUtx1OaApbOtmltkSJR1/N5iVFlkmosRKjsKat+wjHLlxyiwhwYbJGo6wurrzdgWVRN4UCAMOFEY+mLF+	5,200 Byte
Transaction Data	Transaction ID: 20507210001 AccountNumber: 1122334455 BVN: 22445566778 BankName: GTBank Amount: 600,000 PaymentMethod: Card BalanceBefore: 900,000 BalanceAfter: 300,000 BankCode: 058	XqEQBAECAADRGQ/2gYQAGAPxZDo79n8tyLRCgvGb8rJdNUEPP/k72EeADsBg2vMXvhny3LzcdOPzDtdxE4Sq7u7u7u7t7p1qEc4QeYXyGcnjUwE6/hD2h5TGKRqJdadKBwFYqiit/Lt6kRKUYbbn0IYfW1Y+sr5EAGYzuBM8vjygzLGkhTY4ovGPFKvtV42sFhK3AwZAswtKxCON2LQXRiLAhU5ZPoI8yNx4CfQ8ZcFXyMCLiSZmSyDmS4c1U8MvLTeqBUNYOWgzZVe1X/auwrIt+UWqXfMUwTBTB+DXiVFJjw4jcT9UGRoKCn4yoiEejnwGWb4c8MVs9v1qdDEAsc+g03KbMjMGOMaxb4=	4,800 Byte

From the tabulated in Table 2, it was observed that the encrypted values were completely unreadable, ensuring strong data privacy protection. Nonetheless, the ciphertext a size varies based on the message length. This is due to the fact the CKKS operates on floating-point approximations. The developed system successfully encrypts and decrypts messages without the sensitive data content being exposed to threat. This ensures a robust data isolation for both the data in-transit and at-rest. Fig 2 capture the screenshot proof of the encryption process using the generated key-pair name “Allwell-key”.

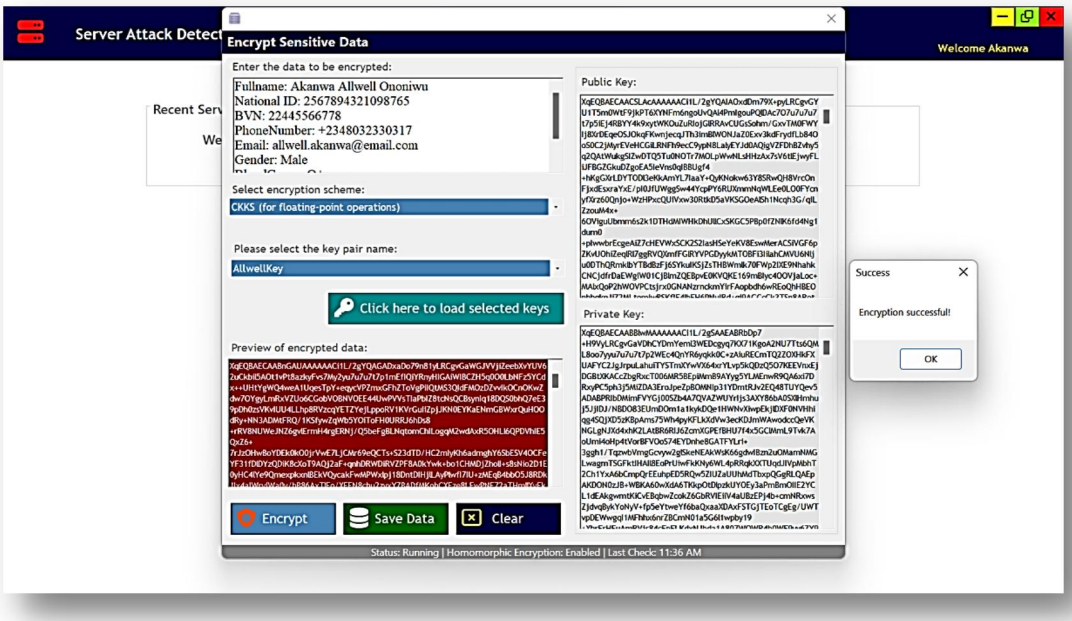

Fig 2: Plaintext Encryption Process using the generated Key Pair

ISSN : 2455-135X

The To evaluate the computational efficiency of the developed system using the scheme (CKKS), we measured its encryption time, decryption time, and ciphertext size for different plaintext message lengths and tabulate them in Table 3.

Note, for a ciphertext size of 4,800 bytes, the developed system completed a HE encryption and decryption operations in 0.35ms and 0.12ms respectively. For a ciphertext size of 5,200 bytes, the model completed the operations in 0.42ms and 0.24ms. However, similar trends were observed for larger cipher sizes. The Time taken for the developed system to perform the encryption and decryption operations on the various ciphertext sizes using the recommended 128-bits security level is tabulated in Table 3, while Figure 3 captures a line graph representation of the various encryption and decryption time of the listed data sizes using the CKKS Homomorphic Encryption (HE) scheme.

Table 3 Encryption and Decryption Time for various Data Sizes

Data Size in Byte	Encryption Time	Decryption Time	HE Scheme
4,800	0.35ms	0.12ms	CKKS
5,200	0.42ms	0.24ms	CKKS
10,000	0.6ms	0.42ms	CKKS
20,000	1.2ms	0.94ms	CKKS
50,000	3.6ms	1.9ms	CKKS

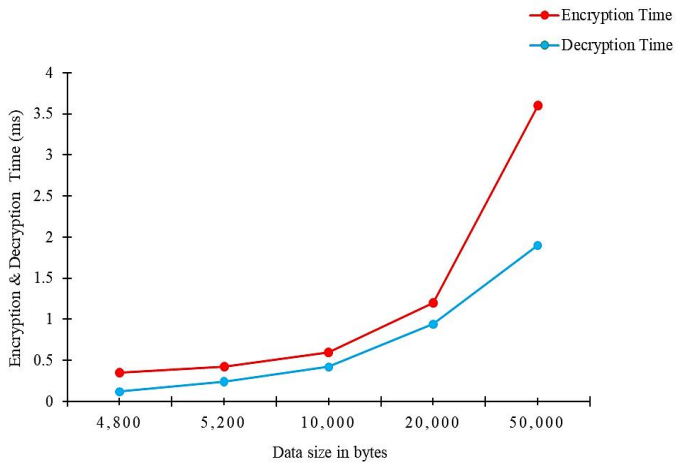

Fig 3: Line Graph Representation of the various Encryption and Decryption Time

In this study, a performance analysis was conducted to demonstrate the efficacy of Homomorphic Encryption (CKKS) and its role in ensuring privacy preservation in the proposed system. The results of the analysis conducted are captured and tabulated is in Table 4. The analysis highlights the effectiveness of Homomorphic Encryption (HE) in securing sensitive data (both in transit and at rest) while allowing computations on encrypted data, making it a robust solution for cloud-based security. Despite its computational overhead, CKKS was adopted due to its ability to process encrypted information without decryption, thereby minimizing exposure to cyber threats. This confirms the HE as a suitable cryptographic solution for upholding data isolation in the developed server-side attack detection and blocking system.

Table 4 Performance Analysis of Symmetric and Asymmetric Key Cryptosystem

Method	Encryption	Decryption	Distribution	Complexity	Security	Nature
RSA	Slow	Slow	Decentralized	O(N^3)	High	Asymmetric
DES	Fast	Fast	Centralized	O(LogN)	Moderate	Symmetric
AES	Fast	Fast	Centralized	O(LogN)	High	Symmetric
ECC	Fast	Fast	Centralized	O(N^3)	High	Symmetric
HE	Very slow	Slow	Decentralized	O(N^4)	Extreme	Asymmetric

5. Conclusion

This study addressed one of the most significant security challenges in multi-tenant cloud computing environment by developing a privacy-preserving cryptographic model capable of protecting tenant data throughout its entire lifecycle. Although conventional encryption techniques provide adequate protection for data at rest and during transmission, they require data to be decrypted before processing, thereby exposing sensitive information within the cloud environment. To overcome this limitation, this study integrated a homomorphic encryption model into the cloud data processing workflow, enabling computations to be performed directly on encrypted data without revealing the underlying plaintext. By eliminating the need for decryption during processing, the proposed model strengthens tenant isolation and significantly enhances data confidentiality within shared cloud infrastructures.

If there is no conflict of interest, authors should state that “The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

The proposed model was implemented using the BFV and CKKS homomorphic encryption scheme in a Python environment with MySQL serving as the database management system. Experimental evaluation confirmed the successful generation of homomorphic public and private key pairs, while user identity information and financial transaction records were securely transformed into ciphertext that remained unintelligible without the corresponding decryption key. The decryption process consistently reconstructed the original plaintext without any loss of data integrity, demonstrating the correctness, reliability, and effectiveness of the proposed cryptographic framework.

The experimental results further demonstrated that encryption and decryption times increased gradually and predictably as the size of the input data increased. Even for

ISSN : 2455-135X

larger message sizes, processing times remained within a few milliseconds, indicating that the computational overhead introduced by HE is manageable for realistically sized cloud data records. Comparative evaluation with the Rivest-Shamir-Adleman (RSA), Data Encryption Standard (DES), Advanced Encryption Standard (AES), and Elliptic Curve Cryptography (ECC) further highlighted the strengths and limitations of our proposed approach. Although HE incurred greater computational cost than these conventional encryption algorithms, it remained the only technique capable of performing computations directly on encrypted data while preserving data confidentiality throughout the computation process. This capability represents a significant advantage for multi-tenant cloud environments where preventing unauthorized access to tenant information during processing is a fundamental security requirement.

The findings of this study demonstrate that HE provides a practical and effective foundation for privacy-preserving cloud computing. Rather than depending solely on access control mechanisms or organizational security policies, the proposed model embeds privacy protection directly within the cryptographic process itself. This approach substantially reduces the opportunities for data exposure arising from insider threats, system misconfigurations, unauthorized access, or malicious attacks, thereby providing stronger guarantees for tenant data confidentiality in shared computing environments.

Although the proposed model achieved its intended objectives, the evaluation was conducted under controlled experimental conditions using individual records and did not consider large-scale concurrent processing involving multiple tenants. Consequently, the performance of the model under production-scale cloud workloads remains an area for further investigation. Future studies should evaluate the scalability of the proposed model in large multi-tenant cloud environments, investigate hybrid cryptographic approaches that combine homomorphic encryption with efficient symmetric encryption techniques to reduce computational overhead, and assess the model against a broader range of sophisticated cyber threats, including side-channel and inference attacks. Such investigations would provide additional evidence of the model's suitability for deployment in real-world cloud computing environments.

This study has demonstrated that the computational overhead associated with Homomorphic Encryption (HE) is outweighed by its unique ability to preserve data confidentiality during computation. The proposed model therefore represents a significant advancement toward secure and privacy-preserving multi-tenant cloud computing by providing cryptographic protection that extends beyond storage and transmission to include active data processing. Consequently, the study offers a practical and scalable framework for strengthening tenant isolation and improving trust in cloud computing environments where data privacy remains a critical concern.

References

[1] M. Kansara, "Advancements in cloud database migration: Current innovations and future prospects for scalable and secure transitions," *Sage Science Review of Applied Machine Learning,* vol. 7, pp. 127-143, 2024.

[2] W. Hashim and N. A.-H. K. Hussein, "Securing cloud computing environments: An analysis of multi-tenancy vulnerabilities and countermeasures," *Shifra,* vol. 2024, pp. 8-16, 2024.

[3] B. A. Sekti, "Data Privacy and Cybersecurity in Wind Power Systems," in *AI-Powered Analysis, Modeling, and Monitoring of Wind Energy Systems*, ed: IGI Global Scientific Publishing, 2026, pp. 335-366.

[4] N. S. Gaddapuri, *Cloud Computing in Regulated Financial and Healthcare Systems (Architecture Patterns, Security, and Compliance Frameworks)*: Geh press, 2026.

[5] B. R. Louassef, N. Chikouche, H. Mrabet, and S. Belguith, "A privacy-preserving scheme for iot healthcare systems using blockchain and lattice-based cryptography," *The Journal of Supercomputing,* vol. 82, p. 295, 2026.

[6] O. E. Taylor and I. N. Davies, "A Model for Enhancing Security and Privacy in Pervasive Computing using Homomorphic Encryption " *International Journal of Computer Sciences and Engineering,* vol. 13, pp. 21-29, 2025.

[7] B. O. Eboseremen, A. O. Ogedengbe, E. Obuse, O. Oladimeji, J. O. Ajayi, A. O. Akindemowo*, et al.*, "Secure data integration in multi-tenant cloud environments: Architecture for financial services providers," *Journal of Frontiers in Multidisciplinary Research,* vol. 3, pp. 579-592, 2022.

[8] S. K. Henge, R. Rajakumar, P. Prasanna, A. Parivazhagan, Y.-C. Hu, and W.-L. Chen, "Multi-layered access control based auto tuning relational key implications in enterprise-level multi-tenancy," *Multimedia Tools and Applications,* vol. 84, pp. 17805-17836, 2025.

[9] P. Kumar and A. K. Bhatt, "HE-AO: An Optimization-Based Encryption Approach for Data Delivery Model in A Multi-Tenant Environment," *Wireless Personal Communications,* vol. 138, pp. 1329-1350, 2024.

[10] P. Mehta, "SECURING CLOUD ENVIRONMENTS WITH HOMOMORPHIC ENCRYPTION," *International Research Journal of Advanced Engineering and Technology,* vol. 1, pp. 25-30, 2024.

ISSN : 2455-135X

[11] S. I. Serengil and A. Ozpinar, "Lightphe: Integrating partially homomorphic encryption into python with extensive cloud environment evaluations," *arXiv preprint arXiv:2408.05219,* 2024.

[12] J. Wu, T. Sun, F. Luo, H. Wang, and W. Zhang, "Secure multi-key homomorphic encryption with application to privacy-preserving federated learning," *arXiv preprint arXiv:2506.20101,* 2025.

[13] Y. H. Chan, H. Yang, S. Shen, X. Fan, S. Lyu, P. S. Hung*, et al.*, "HHEML: Hybrid Homomorphic Encryption for Privacy-Preserving Machine Learning on Edge," *arXiv preprint arXiv:2510.20243,* 2025.

[14] M. A. Junior, P. Appiahene, O. Appiah, and K. Adu, "Cloud data privacy protection with homomorphic algorithm: a systematic literature review," *Journal of Cloud Computing,* pp. 1-26, 2025.

[15] J. Wang and Y. Wang, "Privacy Protection Optimization Method for Cloud Platforms Based on Federated Learning and Homomorphic Encryption," *Sensors,* vol. 26, p. 890, 2026.

[16] D. S. Ene, I. N. Davies, G. F. Lenu, and I. B. Cookey, "Implementing ECC on Data Link Layer of the OSI Reference Model," *SSRG International Journal of Computer Science and Engineering,* vol. 8, pp. 12-16, 2021.