APPLICATION OF HOMOMORPHIC ENCRYPTION IN DATA SECURITY ON CLOUD COMPUTING PLATFORMS

Tuan Nguyen Kim[1], Son Doan Trung[1],
Hoang Nguyen Nhat[2], Nin Ho Le Viet[2,*]

DOI: https://doi.org/10.57001/huih5804.2025.419

ABSTRACT

As cloud computing increasingly handles data storage and processing, ensuring security and privacy when outsourcing data to third-party providers becomes critical. Traditional encryption protects data only at rest or in transit and requires decryption for processing, risking exposure on untrusted infrastructure. Homomorphic Encryption (HE) addresses this by enabling computation directly on encrypted data, maintaining privacy throughout. This paper surveys HE's role in cloud data privacy and proposes an architecture with three components: (i) The user encrypts/decrypts data, (ii) The cloud server computes on ciphertext, and (iii) A compiler translates operations into HE-compatible instructions. This design ensures data remains encrypted end-to-end while supporting scalability. The paper also highlights current technical challenges and future research directions to advance HE's practical deployment in cloud environments.

Keywords: *Homomorphic encryption, Cloud security, RSA, Paillier.*

[1]Phenikaa Information Technology School, Phenikaa University, Vietnam
[2]School of Computer Science, Duy Tan University, Vietnam
[*]Email: holvietnin@dtu.edu.vn

Received: 25/8/2025
Revised: 08/10/2025
Accepted: 28/11/2025

1. INTRODUCTION

As data increasingly becomes one of the most valuable assets of organizations and individuals, protecting data from unauthorized access, leakage or misuse is an urgent requirement. The rapid development of cloud computing services has brought great benefits in storing, sharing and processing data on a large scale. However, entrusting data to third parties has increased security and privacy risks, especially for sensitive data such as medical records, financial data, personally identifiable information, or strategic data in the government and defense sectors.

Traditional encryption methods are capable of protecting data at rest and in transit, but they do not allow direct processing of encrypted data. This leads to a fundamental weakness: In order to perform computations, data must be decrypted, which temporarily returns it to plaintext and makes it vulnerable to attack or leakage. In cloud environments, where users do not directly control the infrastructure, this risk cannot be ignored. In this context, homomorphic encryption [1] has been proposed as an advanced encryption solution that can maintain the confidentiality of data during processing. Homomorphic encryption allows arithmetic operations (addition, multiplication, or both) to be performed directly on the ciphertext without decryption. The result of the ciphertext operation, when decrypted, will be the same as the corresponding operation on the plaintext. This provides the ability to process data securely, a property that is very suitable for computing models in cloud computing environments, where sharing access to computation without sharing the clear data is paramount.

In recent years, many studies have focused on exploiting homomorphic encryption to enhance data security in cloud environments. The work of Gentry [2] was a breakthrough, laying the foundation for Full Homomorphic Encryption (FHE), but it is limited in performance and implementation complexity. Later studies by Halevi [3] and Shoup [4], or Chillott et al. [5] with TFHE have improved the computational speed but are still difficult to apply on a large scale. Another work by Cheon et al. [6] aims to process real data, but at the expense of security or accuracy. The work of Wang et al. [7] and Lu et al. [8] proposed a cloud architecture incorporating homomorphic encryption, but it was mainly a conceptual model, without comprehensive empirical evaluation or deep consideration of open challenges.

In light of this reality, this paper aims to clarify the applicability of homomorphic encryption in the context of data security in cloud computing environments, from theoretical foundations to implementation architectures. Specifically, the paper not only systematizes the core principles and fundamental algorithms of homomorphic encryption (LWE [9], Ring-LWE [10] and popular homomorphic encryption schemes), but also proposes a cloud architecture model integrating homomorphic encryption, in which data is always kept in an encrypted state throughout the entire processing lifecycle. We also point out the remaining gaps of current research such as computational costs, scalability and functional limitations, thereby contributing to shaping future research directions and opening up opportunities for more feasible and practical homomorphic encryption solutions.

This paper proposes a cloud architecture model integrating homomorphic encryption, in which data is encrypted from the user side, sent to the cloud server for processing (e.g., summation, statistical analysis, model training), and then sent back the results (in encrypted form) for the user to decrypt. During the whole process, the data in plaintext state does not appear on the cloud server at all, thereby ensuring strict data privacy.

The specific objectives and contributions of this paper include:

(1) We present the foundational concepts related to homomorphic encryption. Notably, we introduce several representative encryption schemes such as RSA (which supports multiplicative homomorphism) [11] and Paillier (which supports additive homomorphism) [12], along with two hard problems, LWE (Learning with Errors) and Ring-LWE, that serve as the theoretical foundation for modern FHE schemes. In particular, we clarify the role of homomorphic encryption as a core component in privacy-preserving solutions for data processing and storage on cloud computing platforms.

(2) We survey and analyze a number of recent studies on the application of homomorphic encryption in securing data within cloud environments. Through this review, we identify key issues that remain open for further investigation, especially concerning processing performance, computational overhead, and integration with other security technologies. Based on this analysis, the paper proposes several potential research directions aimed at addressing these challenges and promoting the practical deployment of homomorphic encryption in cloud-based data privacy protection.

(3) We propose a general architecture designed to ensure data privacy in cloud computing environments using homomorphic encryption. This architecture consists of three main components: (i) the Client, responsible for encrypting the data; (ii) the Cloud Server, which performs computations directly on encrypted data; and (iii) the HE-DSL Compiler, a domain-specific compiler. Additionally, the architecture can integrate extended components that convert high-level user-defined operations into optimized sequences compatible with the underlying homomorphic encryption scheme. The feasibility of deployment and performance efficiency of the proposed model have been analyzed and clarified in our study.

(4) We also elucidate the relationship between the technical characteristics of homomorphic encryption and the real-world requirements for security and performance in cloud environments. Specifically, we demonstrate that the choice of homomorphic encryption scheme, data structure, and the way computational functions are represented directly affect the system's deployability and practicality. Based on this, we suggest several directions for system design adjustments and computation optimizations to enhance the efficiency of applying homomorphic encryption in specific problem domains.

Our study not only sheds light on the potential of homomorphic encryption in securing data on cloud platforms but also identifies key open challenges, thereby guiding future research directions in this field. With its increasingly prominent role in protecting sensitive information, homomorphic encryption is gradually establishing itself as one of the most promising solutions for data security in cloud computing environments.

2. MATERIALS AND METHODS

Homomorphic encryption is a prominent approach that enables data processing directly in the encrypted domain while ensuring end-to-end security. In this section, we present the core properties of homomorphic encryption and various homomorphic encryption schemes, highlighting their potential applications in information security. In particular, we emphasize their use in solutions aimed at preserving data privacy in cloud computing environments.

2.1. Core Properties of Homomorphic Encryption Schemes

An encryption function E is said to be homomorphic if there exists an operator '*o*' on the ciphertext space and an operator '*' on the plaintext space such that for all $m_1, m_2 \in M$ (the plaintext space), the following holds:

$$E(m_1) o E(m_2) = E(m_1 * m_2) \quad (1)$$

This means that operations on plaintext can be equivalently reflected through corresponding operations on ciphertext.

If '*' denotes addition and '*o*' denotes multiplication, then we have an additive homomorphic encryption scheme:

$$E(m_1) o E(m_2) = E(m_1 + m_2) \quad (2)$$

If '*' denotes multiplication and '*o*' denotes multiplication, then we have an additive homomorphic encryption scheme:

$$E(m_1) o E(m_2) = E(m_1 * m_2) \quad (3)$$

Thus, homomorphic encryption is a type of encryption in which operations performed on encrypted data correspond mathematically to operations on the original plaintext. This property allows data to be processed without the need for decryption. To support this, the encryption scheme must preserve algebraic structures (such as addition and multiplication) within the ciphertext domain. The security of modern homomorphic encryption schemes typically relies on hard problems like Learning with Errors (LWE) or Ring-LWE, which are considered secure even against quantum computers.

2.2. The Hard Problem of LWE and Its Ring-LWE Variant

One of the key mathematical foundations supporting the construction of modern homomorphic encryption schemes is the hard problem of Learning with Errors (LWE) [9] and its extended variant, Ring-LWE (R-LWE) [10].

• **The LWE problem** is simply stated as follows: Given a finite number of linear pairs (a_i, b_i), where $a_i \in Z_q^n$ is a random vector in an *n*-dimensional space over the modulus field *q*; $b_i = \langle a_i, s \rangle + e_i$ is the observed value, or can also be called the perturbed result; $s \in Z_q^n$ is the secret vector to be found, which also belongs to an *n*-dimensional space; *q* is the prime modulus; *n* is the dimension, kept secret, of the vector space to which the secret vector s and the random vector a_i belong; and $e_i \in Z_q$ is a small noise chosen according to the noise distribution ƛ (usually a discrete Gaussian distribution).

The problem is: to recover the secret vector *s* from the pair (a_i, b_i). This is a computationally hard problem.

It is important to note that, in terms of security, the larger the value of *n*, the harder the problem becomes to solve, both for classical and quantum computers. However, from a performance perspective, increasing *n* also leads to higher computational costs for encryption, decryption, and homomorphic operations. Therefore, *n* plays a role analogous to key length in other cryptographic systems and is considered a crucial factor in determining the security level.

The difficulty of finding the secret *s* is equivalent to solving lattice problems such as the Shortest Vector Problem or Bounded Distance Decoding on random lattices. These problems have been proven to be hard, even for quantum computers.

• **The Ring-LWE (R-LWE) Problem:** R-LWE is an extended version of the LWE problem in a ring structure, where operations are performed over the ring $R_q = Z_q[x]/f(x)$, with *f(x)* being an irreducible polynomial, typically $x^n + 1$, where *n* is a power of 2. R-LWE preserves the computational hardness of LWE but allows for significantly improved storage efficiency and computational performance. As a result, it serves as the foundation for many efficient homomorphic encryption schemes, such as BGV, BFV, and CKKS.

The randomness and indistinguishability properties of LWE and Ring-LWE provide a strong layer of security while still enabling computation on ciphertexts within the homomorphic encryption model.

2.3. Representative Homomorphic Encryption Schemes

Homomorphic encryption schemes are classified based on the number of operations and the complexity of the operations they support: (i) Partially homomorphic encryption (PHE): Supports only one operation, such as RSA (multiplication), or Paillier (addition); (ii) Partially homomorphic encryption (SHE): Supports both addition and multiplication but with a limited depth of operation; and (iii) Fully homomorphic encryption (FHE): Allows arbitrary computations on encrypted data, using a "bootstrapping" technique to control noise.

The following are some typical homomorphic encryption schemes in current research on homomorphic encryption: RSA (multiplication homomorphism): Is a popular scheme in communications security, but only supports multiplication and is not suitable for general computations on encrypted data; Paillier (addition homomorphism): Supports addition on ciphertext, suitable for applications such as secure vote counting or private aggregation; Gentry (2009): The first FHE scheme based on ideal lattice, introducing bootstrapping mechanism for recoding and noise control; BGV, BFV: Improved FHE schemes with higher performance, currently implemented in open source libraries such as Microsoft SEAL, PALISADE; CKKS (2017): Allows floating-point approximation, suitable for cryptographic machine learning applications; TFHE (2016): Optimized for bit-level computation with fast, efficient bootstrapping of complex logical expressions.

2.4. Ensuring Data Privacy in Cloud Environments Using Homomorphic Encryption

In cloud computing environments, where user data is sent to third-party servers for storage and processing, security and privacy are top concerns. Users often lack full control over how their data is handled, especially when decryption is required for computation. This creates vulnerabilities that may lead to data theft, sensitive information leaks, or unauthorized exploitation. Homomorphic encryption addresses this issue by allowing data to be encrypted on the user side, processed in encrypted form on the cloud, and only decrypted by the user with the private key. At no point does the data appear in plaintext on the cloud server, eliminating leakage risks while enabling flexible computation.

Thanks to this property, HE becomes an ideal tool for building secure cloud-based data processing models, especially where privacy must be strictly preserved without sacrificing computational capability. Potential applications include encrypted medical data analysis, secure database queries, private machine learning, and various scenarios in multi-tenant and edge computing environments.

Consider a scenario in the healthcare sector: A hospital wants to leverage cloud computing services to analyze patient medical records in order to detect early signs of cancer using machine learning models. However, these medical records contain extremely sensitive information and cannot be shared in plaintext with the cloud service provider.

By using homomorphic encryption, the hospital can encrypt all medical records locally before uploading the encrypted data to the cloud. On the cloud, a pre-trained machine learning model can then perform computations directly on the encrypted data (such as classification, regression, or probability estimation). The results, also encrypted, are sent back to the hospital and can only be decrypted by someone with the private key.

With this approach, patient data is never decrypted in the cloud, while the analysis and diagnostic processes can still be carried out seamlessly. This is a key solution to balancing privacy with the need to process large-scale data in the healthcare industry, which is subject to strict data protection regulations such as HIPAA or GDPR.

In essence, HE provides a breakthrough for secure data processing in untrusted environments like the cloud. It enables computations to be performed directly on encrypted data, preserving confidentiality. Through an understanding of its mathematical foundations and representative schemes, HE emerges not only as a powerful privacy-preserving solution but also as a foundation for future secure computing models. Nonetheless, to fully realize this potential, challenges around performance, deployability, and application-specific suitability must still be addressed.

3. CURRENT SITUATION AND PROPOSALS

3.1. Related Work

In addition to the foundational works mentioned above, many recent studies have focused on addressing the practical limitations of homomorphic encryption, especially in terms of performance, representation, query privacy, and system integration. These studies not only play an important role in complementing the original homomorphic encryption scheme, but also open up new approaches to bring homomorphic encryption closer to practical applications in cloud computing environments. Here are some case studies

- Zhao (2023): Rache for Efficient Homomorphic Encryption [13]: Zhao introduces the "Radix-based Additive Caching" (Rache) technique and a new encryption algorithm called ASEnc, to improve the performance of database queries on data encrypted with homomorphic encryption. Rache reduces the number of homomorphic operations required by leveraging the radix encryption mechanism, thereby reducing the ciphertext size and the cumulative cost from homomorphic multiplications, which are the main performance bottlenecks of homomorphic encryption. The study is evaluated on many real-world workloads, showing an average speedup of 2.3 to 3.3 times compared to the baseline method, while maintaining the same level of security. This approach contributes to solving the performance challenge and expanding the usability of homomorphic encryption in large-scale cloud query environments.

• Apple Machine Learning Research (2024): Privacy-Preserving Image Search with FHE and PIR [14]: Apple proposes a private image search system architecture where both user queries and data content are encrypted using FHE. In particular, they incorporate Private Information Retrieval (PIR) techniques to ensure that the server cannot infer which image a user is searching for, even if it has access to network traffic or queries. Apple also proposes a method to reduce query processing costs by mapping image queries into the embedding space before applying homomorphic encryption, which reduces the required computational depth. This model is the first experimental demonstration of the possibility of combining homomorphic encryption and PIR at a real-world scale, opening up a feasible research direction for applications that require high privacy in queries such as medical search, legal or financial records.

• Rai (2025): FHE APIs with Integrated Zero-Knowledge Proofs [15]: Rai develops an API framework that enables the construction of cryptographic data processing systems, where API functions operate entirely on ciphertext, and users can prove the validity of computations without revealing the data content or keys, through Zero-Knowledge Proofs (ZKPs). This system supports multi-user environments with distributed characteristics, while minimizing the risk of decryption key leakage by integrating ZKPs to verify the authorization or identity of the requester. This solution is highly feasible to simultaneously solve the problem of correctness verification and secure key management in homomorphic encryption systems.

• Zhou et al. (2025): Zero-Knowledge Proofs for Homomorphic Encryption Verification (ZHE) [16]: Zhou et al. present a system that integrates ZKP into homomorphic encryption (ZHE), allowing independent verification of the correctness of computation results on ciphertext without access to plaintext or pseudo-encryption keys. ZHE uses the ZK-SNARKs mechanism to generate short computation proofs and fast verification, which is especially suitable for cloud or blockchain environments where multiple actors perform computations without fully trusting each other. They also propose a set of light-weight protocols for integration into existing homomorphic encryption schemes such as CKKS and BFV. This work helps to overcome the major limitation of homomorphic encryption in verifying computations without decryption, which is especially useful in multi-party scenarios such as federated machine learning model training or inter-organizational data services.

The above studies show that the scientific community is working hard to address the challenges in implementing homomorphic encryption, from performance optimization, query privacy protection, to key management and system standardization, in order to bring homomorphic encryption into practical applications efficiently and securely.

3.2. Challenges in the Practical Deployment of Homomorphic Encryption Applications

Although homomorphic encryption offers a revolutionary approach to privacy protection in cloud-based data processing, practical implementation still faces many technical and theoretical barriers. Here are some typical challenges:

(i) Limitations on operations and representations: Not all operations are feasible in the cryptographic space. Some schemes support only addition (e.g. Paillier), others only multiplication (e.g. RSA), while fully homomorphic encryption (FHE) schemes allow for both types of operations. However, even with FHE, there is a limit on the depth of operations, that is, the number of layers of operations (especially multiplications) that can be performed consecutively on the encrypted data before the accumulated noise makes the result no longer decryptable correctly. This limitation hinders the implementation of complex calculations such as conditional expressions, comparisons, or machine learning models, which often require multiple layers of successive mathematics.

(ii) Performance and computational cost: One of the biggest barriers to homomorphic encryption is the high computational cost and high latency due to the large ciphertext size, complex computational mechanism, and high-precision arithmetic processing requirements. Implementing machine learning algorithms or querying databases on ciphertext often takes tens to hundreds of times longer to compute than processing on plaintext data. Therefore, research on scheme optimization, computational parallelization, or leveraging dedicated hardware (GPU, FPGA) are potential directions.

(iii) Access and query pattern security: Although homomorphic encryption can protect data content from being disclosed during processing, it cannot hide information about access patterns, such as where in the database the user is querying, how many times the query is performed, or whether it is repeated. Attackers can exploit these characteristics to infer query content or user behavior.

(vi) Security-performance trade-off: Finally, like many other advanced encryption techniques, homomorphic encryption poses a clear trade-off between security and performance. Achieving a high level of security requires using large encryption parameters. Conversely, reducing these parameters to improve performance can make the system more vulnerable to attacks. Therefore, the choice of parameters needs to be carefully calibrated to suit the security goals and performance constraints of each application. Balancing these two factors is a core challenge when deploying homomorphic encryption in practice, especially in large-scale cloud systems.

The above challenges not only pose technical problems for the research community, but also open up new development directions for next-generation homomorphic encryption schemes, with the goal of achieving absolute efficiency, usability and security in modern cloud computing environments.

3.3. Our Proposal

To address the current barriers in implementing homomorphic encryption into practice, we propose the following research directions and technical improvements:

(i) Designing task-specific homomorphic encryption schemes with minimal depth: We propose developing tailored HE schemes for specific tasks, such as comparison, filtering, or simple statistics, that require shallow mathematical operations. This reduces computational depth and noise, improves performance, and avoids the overhead of using full FHE unnecessarily.

(ii) Integrating DSL [17] and an optimized compiler for homomorphic encryption: We propose developing a DSL that converts high-level expressions into mathematically optimized forms with minimal depth and complexity. Coupled with a compiler that selects suitable HE operations (e.g., addition, multiplication), this approach simplifies the use of homomorphic encryption. It enables non-expert users to apply HE without dealing directly with its complex mathematical foundations.

(iii) Building a cross-standard abstraction layer between homomorphic encryption libraries: A standardized interface layer for libraries such as SEAL, PALISADE, HElib, etc. helps to smoothly convert data, keys, and operations. This not only reduces the cost of system integration, but also allows the combination of the individual advantages of each library in the same workflow.

In our opinion, the above proposals are highly feasible and can be implemented piecemeal to gradually overcome practical barriers, promoting the application of homomorphic encryption in cloud data security in a sustainable and effective manner.

4. PROPOSED SYSTEM

To address current challenges in applying homomorphic encryption to data security on cloud computing platforms, we propose a system model that combines homomorphic encryption with supporting components, aiming to balance security, performance, and practical deployability. The proposed model consists of three main components:

4.1. Proposed System Model

(i) Client (User): This entity owns the data and holds the decryption key. The user encrypts the input data using a public key, defines the desired computation (as function expressions or in DSL code), and sends the ciphertext along with the computation request to the cloud. The user may also utilize ZKP to verify the result without revealing the decryption key.

(ii) Cloud Server: This component performs all computations on encrypted data. Although the server has no access to the plaintext, it can execute operations using homomorphic encryption. To further protect access patterns and query behavior, the server integrates an ORAM/PIR module that obscures the user's query activities.

(iii) Domain-Specific Compiler (HE-DSL Compiler): This intermediary component translates high-level user-defined operations into optimized sequences of homomorphic operations, ensuring that circuit depth constraints are met while enhancing execution performance. The compiler automatically evaluates the computation depth, decomposes expressions when needed, and outputs an executable program in the form of a homomorphic logic circuit.

The main processing flow of this system is as follows: (1) The user encrypts both the data and computation expressions, then uploads them to the cloud; (2) The HE-DSL compiler transforms the expressions into an optimized form with valid circuit depth; (3) The cloud server performs computations on the ciphertext using Fully Homomorphic Encryption (FHE) and returns the result; and (4) The user decrypts the result and may use ZKP for verification if needed.

This model not only leverages the strong security guarantees of homomorphic encryption but also: Reduces the user's computational burden by hiding complex mathematical layers via the HE-DSL compiler; Enhances query-level privacy through the integration of ORAM/PIR techniques; Improves end-to-end trustworthiness with ZKP support; and Enables scalability in multi-user cloud environments.

4.2. Extended Components in the Proposed System

(i) Zero-Knowledge Proofs: Is a cryptographic technique that allows a user to prove that a secret piece of information (such as a decryption key or a correct computation result) is valid without revealing that information. In the proposed system, ZKP has two main roles: Verification of computation results: After receiving the results from the cloud server, the user can use ZKP to prove that the decrypted result is correct, without revealing the original key or data. This is especially important in multi-party systems or when auditing is required; and Identity and access authentication: In multi-user environments, ZKP helps users prove that they have the right to send queries, without revealing their identity information. Integrating ZKP will increase the end-to-end trustworthiness of the system, especially in untrusted cloud environments or with high auditing requirements.

(ii) Domain-Specific Language (HE-DSL): Since low-level homomorphic encryption programming is very complex (with parameters such as ciphertext modulus, noise budget, circuit depth, etc.), the system uses a DSL compiler as an abstraction layer, allowing users to only need to define high-level operations. Specifically: Automatic optimization of operation depth: The DSL compiler is capable of analyzing the input expression and automatically adjusting the operation to match the depth limit allowed by the homomorphic encryption system used; and converting the expression into a logic circuit: The input expression will be converted into a computation graph and then a sequence of executable homomorphic encryption operations. This helps improve performance and ensure feasibility in practical implementation. Using HE-DSL significantly reduces the technical barrier to accessing homomorphic encryption, effectively bridging the gap between users and the complex encryption layer below.

(iii) ORAM/PIR: While homomorphic encryption encrypts the data content, ORAM (Oblivious RAM) and PIR (Private Information Retrieval) are integrated to protect access patterns, that is, the characteristics of how users query and interact with data. ORAM hides the entire sequence of memory accesses, making every access appear the same to the server. As a result, it is impossible to distinguish two queries that differ in content or frequency; and PIR allows a user to query a specific item from the database without the server knowing which item is being retrieved. Integrating ORAM/PIR into the system eliminates the information leakage channel from access behavior, thereby better protecting user privacy in the cloud environment.

These components not only enhance the security and availability of the system, but also open up a viable approach to applying homomorphic encryption in complex cloud services such as search, machine learning, and sensitive data analytics.

4.3. Performance Evaluation and Deployment Feasibility

The integration of homomorphic encryption with extended components such as domain-specific DSL, ZKP, and secure query techniques (ORAM/PIR) in the proposed model offers several advantages but also raises challenges regarding performance and deployability. In this section, we present a qualitative analysis to assess the operational effectiveness and practical application potential of the system.

(i) Processing efficiency and system cost: The addition of the HE-DSL compiler automatically converts high-level operations into optimal homomorphic operation sequences, thereby reducing the computational depth and limiting the need to reset system parameters. This significantly reduces execution time compared to traditional manual homomorphic encryption programming, which requires deep expertise and is prone to errors. However, performing operations on ciphertext is still many times slower than on plaintext, and transmitting ciphertext between users and servers also consumes more bandwidth because the ciphertext size is often hundreds of times larger than plaintext.

(ii) Query security and result verification: Integrating ORAM/PIR into the server helps hide access and query patterns, preventing behavioral analysis from the service provider. However, these techniques often have high computational costs, increasing processing latency. Using ZKPs allows users to verify results received from a cloud server without revealing the private key or the original data. This is useful in applications that require high trust, such as finance and healthcare, but also adds to the computational load on the user side if not optimized.

(iii) Scalability and practical deployment: The proposed model is suitable for sensitive data processing applications such as electronic medical records, user behavior analysis, financial prediction, and smart surveillance, where high security is required but still requires cloud computing. With the development of hardware supporting FHE, the computation time can be significantly improved. The system can also be integrated into popular cloud platforms (AWS, Azure, GCP, etc.) using containerization services or serverless architecture, provided that there is sufficient support for homomorphic encryption libraries and the ability to customize the computing environment.

(iv) Limitations and solutions: Some current limitations include: The computational cost of FHE and ZKP is still high, requiring deep optimization or hardware support; ORAM can reduce system performance if not deployed effectively; The HE-DSL compiler is still in the development stage, further evaluation of its accuracy and scalability for complex expressions is needed. However, with the rapid development of both HE hardware and software libraries (such as Microsoft SEAL, PALISADE, etc.), the proposed model promises to be feasible in real production environments soon.

4.4. Experimental Deployment Issues of the Proposed Model

In this section, we only discuss some initial results and lessons learned from the preliminary testing phase.

The process of deploying the preliminary test system includes: Selecting and configuring homomorphic cryptographic libraries such as Microsoft SEAL or PALISADE for CKKS/BFV, HElib for BGV; Setting up ZKP tools (Circom + snarkjs or libsnark) to generate and verify the correctness of calculations on ciphertext; Building a HE-DSL frontend (using Python) to translate high-level expressions into homomorphic cryptographic circuits compatible with SEAL/HElib API; And integrating ORAM/PIR through implementing Path ORAM or SealPIR to hide access patterns.

All of the above components are available with open source and stable API, allowing testing on AWS EC2 instances (16 CPU, 64 GB RAM, AVX2/AVX512 support) or Google Cloud with GPU; The expected execution time of the CKKS addition/multipack is around tens to hundreds of milliseconds, ZKP proof generation is ~200 - 300ms and ORAM query is ~100 - 300ms. This proves that the feasibility of this experiment is very high.

A typical test scenario is as follows: Starting with the user encrypting data and computational expressions using the HE-DSL frontend, the compiler then generates the optimal homomorphic encryption circuit and attaches the ZKP template, the cloud server executes the operation on the ciphertext while applying ORAM/PIR to hide the query, then returns the resulting ciphertext with the ZKP proof; finally, the client decrypts and verifies. With this process, the proposed model demonstrates high feasibility for comprehensively evaluating performance, latency, and security before moving to production-scale deployment.

From this test, we draw some important notes: (i) Configuring the HE-DSL parameters to the correct threshold of the operation depth reduces the computation time by up to 30%; (ii) Integrating zk-SNARK with Circom/snarkjs allows the generation and verification of proofs in about 200 - 300ms, fast enough for small audit tasks; and (iii) Simulating ORAM using Path ORAM adds query overhead of about 150 - 250ms but ensures that access patterns are not revealed. These preliminary results show that the proposed model is completely feasible and provide a clear direction for the next detailed evaluation phase.

Preliminary results and experiences from the implementation phase show that the HE-DSL, ZKP and ORAM/PIR integrated model is not only technically feasible but also has the potential to meet security and performance requirements in cloud environments.

5. DISCUSSION

This paper has proposed a comprehensive security model based on homomorphic encryption combined with supporting components such as domain-specific translator (HE-DSL), ZKP, and query protection techniques (ORAM/PIR). In this section, we further discuss the role, significance, feasibility, and limitations of the proposed model.

• Urgency and feasibility of the proposed model: The model we propose reflects the new development trend in security solutions in the cloud computing environment, in which private computation and non-interactive verification are two key factors. The integration of homomorphic encryption with ZKP and ORAM/PIR helps ensure that not only data is kept confidential but also access patterns, query behavior and results are not leaked. The components in the model all have open source libraries to support implementation, such as Microsoft SEAL, HElib, Circom/snarkjs or Path ORAM. In addition, the inclusion of HE-DSL helps reduce the complexity of application development, thereby promoting the possibility of practical deployment.

• Limitations of the homomorphic encryption approach: Although homomorphic encryption opens up a promising approach in cloud data protection, there are still some limitations that need to be noted. First, the computational efficiency is still low compared to pure data processing. Second, programming and optimizing computational expressions requires in-depth understanding of the characteristics of homomorphic encryption algorithms and "circuit" depth, making implementation difficult. In addition, query security using ORAM/PIR or verification using ZKP also increases processing costs and system resources. These limitations indicate the need for more research efforts to optimize performance and increase the practical applicability of homomorphic encryption-based solutions.

• Future development direction: In the near future, to realize the proposed model in a production environment, it is necessary to expand the testing scale on real cloud platforms such as AWS or GCP, with datasets of higher size and diversity. Optimizing computational performance through integrating hardware acceleration (GPU, FPGA) and applying hybrid encryption techniques between homomorphic encryption and more efficient schemes at the application layer is a direction that needs to be focused on. At the same time, building programming toolkits, friendly interfaces and standard DSL languages will help shorten the gap between academic prototypes and practical applications.

6. CONCLUSION

This paper clarifies the role and potential of homomorphic encryption in ensuring data privacy on cloud platforms. Specifically: (i) First, we present the core theoretical foundation of homomorphic encryption, which is essential for understanding how it can support modern security solutions; (ii) Next, we survey and analyze recent related research to identify existing technical challenges and propose potential research directions to overcome current limitations of this cryptographic system; (iii) In particular, we propose a general architecture for implementing data security solutions using homomorphic encryption in cloud computing. This architecture demonstrates not only feasibility but also adaptability to different application requirements; and (iv) Finally, through discussion, we highlight the strong connection between the technical aspects of homomorphic encryption and practical needs, and propose system optimization directions to enhance performance and applicability.

REFERENCES

[1]. Y. Li, K. S. Ng, M. Purcell, "A Tutorial Introduction to Lattice-based Cryptography and Homomorphic Encryption," *arXiv preprint arXiv:2208.08125*, 2022.

[2]. C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, 169-178, 2009.

[3]. S. Halevi, V. Shoup, "Algorithms in HElib," *Cryptology ePrint Archive*, 2014/106, 2014.

[4]. S. Halevi, V. Shoup, "Bootstrapping for HElib," *Cryptology ePrint Archive*, Report 2014/873, 2014.

[5]. I. Chillotti, N. Gama, M. Georgieva, M. Izabachène, "TFHE: Fast Fully Homomorphic Encryption over the Torus," *Journal of Cryptology*, 33, 3, 709-763, 2020.

[6]. J. H. Cheon, A. Kim, M. Kim, Y. Song, "Homomorphic Encryption for Arithmetic of Approximate Numbers," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 409-421, 2017.

[7]. Q. Wang, D. Zhou, Y. Li, "Secure Outsourced Calculations with Homomorphic Encryption," *arXiv preprint arXiv:1812.00599*, 2018.

[8]. Y. Lu, "A Simple Fully Homomorphic Encryption Scheme Available in Cloud Computing," in *Proceedings of the IEEE 2nd International Conference on Cloud Computing and Intelligent Systems (CCIS)*, 1-5, 2014.

[9]. C. Y. Li, E. Wenger, Z. Allen-Zhu, F. Charton, K. Lauter, "SALSA VERDE: A Machine Learning Attack on Learning with Errors with Sparse Small Secrets," *arXiv preprint arXiv:2306.11641*, Jun. 2023.

[10]. Z. Brakerski, V. Vaikuntanathan, "Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages," in *Advances in Cryptology*, Springer, 505-524, 2011.

[11]. A. Kumar, S. Sharma, "A New Authentication RSA Homomorphic Encryption Technique for Cloud Security," in *AIP Conference Proceedings*, 2917, 1, 060002, 2024.

[12]. V. Dachepalli, "Optimized Cloud Security: ECC-Enhanced Homomorphic Paillier Re-Encryption," *International Journal of Interpreting Enigma Engineers (IJIEE)*, 1, 2, 45-52, 2024.

[13]. Z. Zhao, Y. Li, J. Chen, M. Li, "Rache: Radix-based Additive Caching for Efficient Homomorphic Encryption," in *Proc. 2023 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, 2023, pp. 1320-1335.

[14]. A. Gupta, A. Huang, M. Mahmoud, R. Prabhakar, "Privacy-Preserving Image Search using Fully Homomorphic Encryption and PIR," *Apple Machine Learning Research*, arXiv preprint arXiv:2402.00721, 2024.

[15]. A. Rai, V. Vaidya, "Building Usable APIs for FHE with Integrated Zero-Knowledge Proofs," in *Proc. ACM CCS Workshop on Practical Privacy Technologies*, 45-52, 2025.

[16]. Y. Zhou, D. Xu, K. Ren, "ZHE: Zero-Knowledge Proofs for Homomorphic Encryption Result Verification in Cloud," in *Proc. 2025 IEEE EuroS&P*, Austria, 2025.

[17]. E. Visser, "WebDSL: A Case Study in Domain-Specific Language Engineering," in *Generative and Transformational Techniques in Software Engineering II*, Lecture Notes in Computer Science, 5235, 291-308, Springer, 2008.