I. J. Computer Network and Information Security, **2026, 4, 88-105**
Published Online on August 8, 2026 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2026.04.05

Integrity Checking Mechanism for Privacy-Preserved Auditing of Cloud Shared-Data

Deepshikha Chaturvedi*
Shah & Anchor Kutchhi Engineering College, Mumbai, 400088, India
Email: shikha1in@gmail.com
ORCID ID: https://orcid.org/0000-0002-6939-7566
*Corresponding author

Vidyullata Devmane
Shah & Anchor Kutchhi Engineering College, Mumbai, 400088, India
Email: vidyullata.devmane@sakec.ac.in
ORCID ID: https://orcid.org/0000-0002-2274-3943

Shashikant Radke
Shah & Anchor Kutchhi Engineering College, Mumbai, 400088, India
Email: shashikant.radke@sakec.ac.in
ORCID ID: https://orcid.org/0000-0002-4305-1831

Shahzia Sayyad
Shah & Anchor Kutchhi Engineering College, Mumbai, 400088, India
Email: shahzia.sayyad@sakec.ac.in
ORCID ID: https://orcid.org/0000-0001-5697-6449

Shreeshail Devmane
Shah & Anchor Kutchhi Engineering College, Mumbai, 400088, India
Email: shreeshail.devmane15544@sakec.ac.in
ORCID ID: https://orcid.org/0009-0008-1571-7940

Simran Patel
Shah & Anchor Kutchhi Engineering College, Mumbai, 400088, India
Email: simran.patel15566@sakec.ac.in
ORCID ID: https://orcid.org/0009-0002-8460-4276

Received: February 14, 2025; Revised: May 22, 2025; Accepted: September 2, 2025; Published: August 8, 2026

Abstract: As the cloud computing and mass data sharing develop, data integrity and privacy have become an imperative issue. Conventional remote data auditing techniques tend to reveal sensitive data or they have high computational cost. In order to overcome these shortcomings, the Fully Homomorphic Encryption enhanced Remote Method Invocation (FHEbRMI) mechanism that includes a combination of the Modified Least Squares (MLS) optimization model and the proposed cloud auditing security and efficiency is proposed in this paper. The suggested system provides an encrypted data auditing system, which involves RMI-based communication, to enable the client, server, and third-party auditor to perform their verification functions remotely without the disclosure of the plaintext data. An actual execution of the suggested structure is introduced, such as secure key generation, trapdoor-based dimensionality reduction, ciphertext multiplication, and optimized homomorphic functions. Moreover, the RMI interface provides a smooth communication among the distributed nodes and increases the scalability and minimizes transmission delays. A comparative study with the recent homomorphic-based auditing schemes like blockchain-assisted, certificateless and lattice-based FHE model reveals that the proposed FHEbRMI-MLS model has better performance in terms of encryption/decryption latency, computational cost, and encryption overhead. The experimental performance is indicative of an improvement of about 35–40% in encryption speed and computational efficiency compared with traditional FHE models. This paper presents a viable, privacy-friendly auditing framework of clouds which guarantees the end-to-end encrypted verification without sacrificing the efficiency.

Index Terms: Remote Method Invocation, Modified Least-Squares, Fully Homomorphic Encryption, Homomorphic Encryption.

1. Introduction

This has made cloud storage very popular with its amazing benefits which include flexibility, scalability, redundancy and high level security features [1]. Users can rely on cloud storage providers as they can encrypt their data when uploading or storing them, when individuals or organisations do so [2]. This encryption builds trust among consumers, who are confident that their data is protected [3]. Remote data access is the most important feature of cloud storage; however, the authenticity of data in the remote server is a significant issue [4].

It is very important that there are protocols to verify ownership of data and to detect its integrity so as to ease this concern [5]. The issue lies in the fact that the data secrecy may be compromised in the course of such verification processes, particularly, when hash matching and decryption are concerned [6]. In the dynamism of the cloud storage services market, achieving the optimal equilibrium of ensuring security, data integrity, and ensuring data confidentiality is an endless task [7]. In this paper, the researcher will look at how this paradox can be overcome and get into the specifics of this balance [8].

Many algorithms and techniques have been utilized in the study of integrity checking mechanisms for privacy-preserved auditing of cloud shared data. The most common algorithm (asymmetric) encryption technique is RSA [9], which is based on the factoring of large primes to be secure [10]. Key generation includes picking a prime and calculating a modulus [11]. ECC [12] (Elliptic Curve Cryptography) makes use of elliptic curves, which is a modern encryption methodology. Its power is in difficulty in solving discrete logarithm problems [13]. Multithreading allows performance and responsiveness to be optimized through the implementation of tasks concurrently [14]. Threads share resources yet work on their own and take advantage of multicore processors effectively [15]. Java supports distributed computing through Remote Method Invocation (RMI), which helps to invoke methods on remote objects [16]. It provides network communication between the client and server applications [17]. The cryptographic homomorphic [18, 19] property allows its encrypted data to behave as the result of operations on encrypted data. It allows safe calculation and data confidentiality is maintained [20].

The paper seeks to design a privacy-sensitive and efficient cloud auditing system based on Fully Homomorphic Encryption (FHE) combined with Remote Method Invocation (RMI) and Modified Least Squares (MLS) optimization. The most important points are (i) to be able to compute safe encrypted data, (ii) to minimize the ciphertext growth and the cost of computation with the help of MLS, (iii) to optimise processing performance via parallelism of RSA with multiple threads, and (iv) to give proper verification of integrity without disclosing plaintext data.

The originality of the work consists in the fact that FHE, RMI, and MLS are combined into one system that carries out distributed encrypted computation and verification with privacy protection. The proposed approach has end-to-end data confidentiality, shorter latency, and better scalability not seen by the current auditing models due to optimized encrypted processing. Such a combined mechanism offers secure, efficient, and scalable solution of the modern cloud environment.

The outline of the existing document is as follows. Section 2 provides a review of the literature, and Section 3 gives the definition of the system model and problem statements. Moreover, the proposed structure is presented in Section 4, whereas the results and comparisons are provided in Section 5. Section 6 is the conclusion of the paper.

2. Literature Works

2.1. Cloud Auditing and Checking on the integrity

Kshirsagar et al., [21] have suggested an attribute-based cloud integrity validation system combining Elliptic-Curve-Modified RSA and Modified MD5 hashing algorithm. A Third-Party Auditor (TPA) checks integrity of data through public-key encryption and homomorphic hashing by not revealing plaintext data. Ryu et al., [22] created a blockchain based EHR outsourcing model that encrypts the medical data and stores it in the Interplanetary File System (IPFS) to attain the computational unforgeability and reliability when such data is accessed by multiple users. Chaudhary et al., [23] used a Witness Zero-Knowledge Proof of Knowledge to strengthen Dynamic Integrity Checking (DIC) privacy to safeguard the file and authenticator in case of auditing. Following Wang et al., [24], a protocol called identity-based RDIC (Remote Data Integrity Checking) was developed to remove any certificate management and lessen the need of third-party auditors, enhancing efficiency and privacy. To guarantee privacy and timely verification in a healthcare network and telemedicine, Tang et al., [29] proposed a cloud-storage integrity-auditing scheme, which integrates encryption, a data sanitizer, and blockchain-based time-bound auditing. Kirubakaran et al., [30] have proposed the Privacy-Preserved Data Security Approach (PP-DSA) based on Group Signatures and an authentication model that is TPA-based, and guarantees integrity and confidentiality in the enterprise cloud storage and e-governance.

Goswami et al., [32] introduced a partial signature-based cloud auditing model that uses homomorphic encryption and hash commands to conduct privacy preserving integrity tests on dynamically changing cloud data.

2.2. Homomorphic encryptions and Privacy-Preserving Cryptography

The researchers introduced DPP by Flavia et al., [25], which was a Paillier homomorphic encryption protocol that allowed safe cloud computing at a reasonable latency and compatibility. Zhou et al. [26] compared various partially homomorphic encryption (PHE) schemes Paillier, ElGamal, ASHE and Symmetria based on a study of trade-offs in key size versus ciphertext security. Ramesh et al., [27] compared Multithreaded RSA to perform parallel encryption and decryption and showed that it obtained a substantial improvement in speed when compared to data-intensive workloads. Oppermann et al., [28] extended the Lib-Scarab Fully Homomorphic Encryption (FHE) library and evaluated the synchronisation overheads in field-widening evaluations in multi-threaded polynomial computations. Gajmal and Udayakumar [31] suggested a data-sharing model of EHR based on blockchain, cloud computing, IPFS, and a CAViaR-based Bird Swarm Algorithm (BSA), which generates privacy-sensitive coefficients to share medical information safely. Oluwafemi [33] compared Homomorphic Encryption (HE) and Secure Multi-Party Computation (MPC) as complementary privacy-preserving distributed computation methods in finance and healthcare that are highly secure, but with severe computational complexity. Geva et al., [34] have created a multi-party FHE-based analytics system of cooperative medical data analytics (e.g., survival and logistic regression) that does not require exposing raw data. Xu et al., [35] suggested a homomorphic-encryption-based smart-metering system to facilitate the calculation of billing information with encryptions and ensure that users remain anonymous with smart grids. Ma et al., [36] proposed a more efficient multi-key FHE protocol, named xMK-CKKS, that uses the federated learning of an IoT and mobile network, and ensures the security of model updates with the minimum cost of communication by employing collaborative decryption.

2.3. Research Motivation and Gaps Identified

The analyzed literature indicates a high level of advancement of cloud auditing and privacy-preserving computation but still, some major gaps exist.

- Weak Privacy Guarantees: Most integrity checking schemes are good at making sure that correctness is met, but they do not guarantee that all privacy is maintained during auditing.
- Scalability Limitations: Most of them fail to scale in large-scale or multi-user cloud environments.
- Third Party Dependence: TPA reliance is also accompanied with extra trust and security issues.
- Computational Overhead: Homomorphic and blockchain-based schemes are prone to high processing and storage costs.
- Data Support that is dynamic: Not many frameworks are effective in responding to updates in data that is often updated in real-world clouds.

The proposed FHE-based Remote Method Invocation (FHEbRMI) mechanism incorporates Fully Homomorphic Encryption (FHE), Elliptic Curve Cryptography (ECC) and Multithreaded RSA as a way of addressing such challenges. This hybrid architecture will provide end to end data confidentiality, parallel scalability and reduce the dependence on third party auditors and offer a balanced trade-off between security, efficiency and deployability to cloud data integrity verification.

Few key contributions to the proposed works are listed below,

- Propose an FHE-based RMI protocol that would allow ensuring integrity verification can be performed on encrypted data, preserving data secrecy and eliminating the need to decrypt.
- FHE using Remote Method Invocation (RMI) in addition to secure communication and auditing.
- Installed a multi-threaded design of data chunks processing in a collection of servers. Eliminated the need to decrypt it by supporting operations on encrypted data.
- Created a universal key generation, encryption, homomorphic operations and decryption algorithm.
- Developed a structure to manage the risk of using untrusted Cloud Service Providers (CSPs) and to have timely audits.

3. System Model and Problem Statement

The two main issues facing the system are the delayed (procrastinating) TPA and the cloud being under attack. In addition to hostile assaults that can expose sensitive data and result in identity theft, privacy violations, or reputational harm, the cloud storage server may be compromised by software bugs, hardware faults, or human error. Furthermore, a tardy TPA can work with the cloud server to conceal data damage, postpone audits, or re-use old evidence rather than conducting audits on time. User confidence and data integrity may be jeopardized by this improper supervision, which

may lead to undetected data loss or corruption.

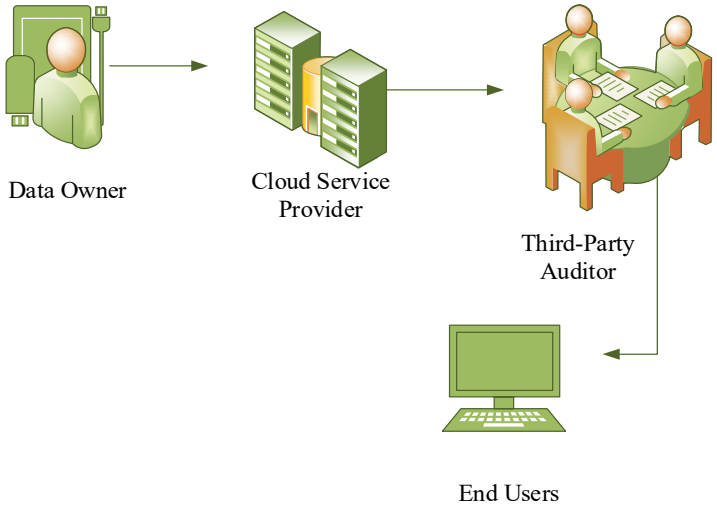

Fig. 1. Basic System Model.

The fundamental system model is depicted in Figure 1. Using homomorphic encryption and hashing algorithms, the Data Owner ensures confidentiality when uploading encrypted and tagged data to the Cloud Service Provider (CSP). Since the CSP is semi-trusted, a Third-Party Auditor (TPA) verifies data integrity without decrypting it, using cryptographic methods like homomorphic encryption and zero-knowledge proofs. The End Users securely access the shared data as authorized. This model ensures data confidentiality, integrity, and transparency, with auditing operations maintaining privacy and trust in the cloud environment.

4. Proposed Methodology

The FHEbRMI framework suggested incorporates Fully Homomorphic Encryption (FHE), Remote Method Invocation (RMI), Modified Least Squares (MLS) and Multithreaded RSA to construct a cohesive, reliable, and effective data auditing system on the clouds. Each of the components deals with a certain weakness of the current auditing schemes. FHE allows processing encrypted data without decryption and, therefore, ensures end-to-end privacy and avoids data exposure to the cloud server. RMI enables invoking secure computational functions on distributed servers remotely, facilitating privacy-sensitive processing on the cloud-server, while maintaining seamless communication between the client and the server. The MLS algorithm acts as an optimization layer, dimension reduction on encrypted data is done to reduce ciphertext growth as well as to minimize computational cost during homomorphic calculations. In the meantime, Multithreaded RSA is an RSA implementation that increases the efficiency of system throughput by performing encryption and decryption on many threads, thereby decreasing latency and increasing processing efficiency. Combined, these methods create a unified system where RMI coordinates distributed encrypted operations, FHE maintains the confidentiality of data, MLS optimizes the encrypted data computation, and multithreading speeds up the cryptographic computations. This synergistic integration enables the suggested FHEbRMI framework to have desired secure, scalable, and high-performance cloud auditing and deal with privacy, performance, and efficiency challenges concurrently.

As depicted in Figure 2, any input file type (text, image, PDF or document) is converted to an integer array format which can be encrypted. The converted array is zero-padded and subdivided into equal parts to be consistent and allow parallel processing. These fragments are allocated to more than one server and in each server the portion assigned to it is allotted to a number of threads to be processed in parallel. Each thread applies encryption with an FHE scheme (like CKKS or BFV) and implement the secure computations with the help of the RMI interface without any decryption in the course of the work, so all the transmitted and processed data is encrypted at any point of the work. An MLS algorithm is subsequently used on the encrypted outputs to reduce dimensions to optimize the size of ciphertext and compute complexity without (significantly) affecting the integrity of the data. Multithreading also has a great effect on the throughput of the system since it is able to perform multiple encryption, computation, and verification processes. All distributed thread encrypted results are finally assembled, proven correct with homomorphic hashing and elliptic curve cryptography (ECC), and decrypted locally to recreate the correct, proven result. This combined methodology guarantees complete preservation of privacy, beyond-scalable parallel processing, and verifying integrity correctly without divulging any plaintext information. The proposed system is effective and secure in cloud-based integrity auditing of dynamic data environments due to FHE (privacy), RMI (distributed execution), MLS (efficiency), and multithreading (performance).

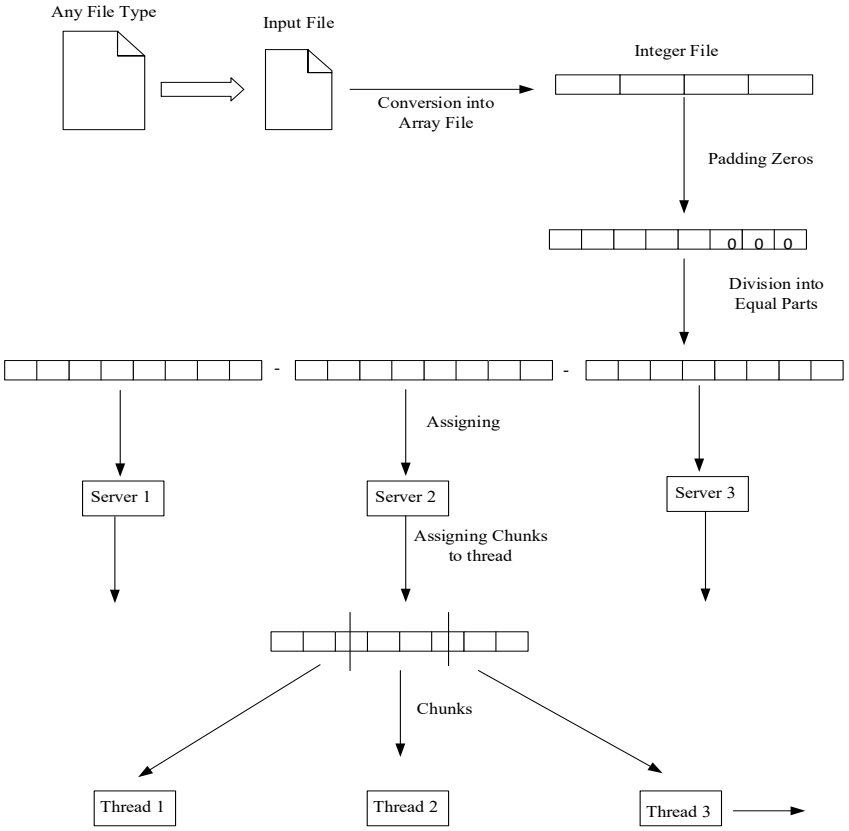

Fig. 2. Proposed Block Diagram.

4.1. Process of proposed Method

The proposed method enhances homomorphic encryption by addressing ciphertext inflation and enabling efficient secure computations. It begins with key generation, where secret and shared key components are defined, followed by trapdoor calculation to facilitate dimensionality reduction. Encryption is performed by generating subciphertexts while preserving plaintext order. Ciphertext multiplication is implemented with dimensionality reduction to maintain a manageable ciphertext size. Decryption reconstructs the plaintext by reversing dimensionality reduction using trapdoors and key information. This process ensures secure, noise-free operations with improved computational efficiency and scalability for encrypted data processing.

4.2. Modified Least-Squares (MLS) Scheme in Homomorphic Encryption

The Privacy-Preserving cloud shared data uses the Modified Least-Squares (MLS) scheme, which significantly improves on the original Least-Squares (LS) homomorphic encryption technique. Two main goals guided the development of this plan:

- *Addressing Ciphertext Inflation:*

Ciphertext inflation is a problem with homomorphic encryption, especially when multiplication operations are involved. This happens when there are inefficiencies in processing and storage because the ciphertext's size grows disproportionately with each multiplication. In order to address this problem, the MLS technique adds trapdoors to the encryption procedure. Special methods known as "trapdoors" enable specific actions on encrypted data without requiring decryption, preserving efficiency and reducing the amount of the ciphertext.

- *Providing an Ordering Feature:*

By introducing an ordering feature for ciphertexts, the MLS technique makes it possible to compare encrypted data. The ω-concept, which adds a "gap" between subciphertexts, accomplishes this. Because of this gap, for the same plaintext value, many ciphertexts can be generated, preserving the order of the data but permitting different representations of the same value. Crucially, data equality is not ensured even while data ordering is preserved, improving the scheme's privacy feature.

The original Least-Squares (LS) scheme's essential features and homomorphic qualities are maintained by the Modified Least-Squares (MLS) scheme, which guarantees noise-free operation and supports a wide range of arithmetic operations. It allows ciphertexts to be added $(\oplus)$ and multiplied $(\otimes)$, as well as to be multiplied with plaintext $(\circledast)$. It

also allows subtraction ($\ominus$) by multiplying ciphertexts with the negation of a plaintext value and then adding. By designating its message space as R (real values) and its ciphertext space as R^m (ciphertext space), respectively, the system makes it possible to directly encrypt real values and expands its use in safe data processing across a range of fields.

4.2.1. Secure Key Generation

Our suggested FHEbRMI approach incorporates improvements catered to our particular needs while maintaining a Secret Key (SK) configuration that closely adheres to the original scheme's structure. The Secret Key is structured as following equation (1),

$$SK(n) = \left[\left(a_1, b_1, c_1\right), \ldots, \left(a_n, b_n, c_n\right)\right] \tag{1}$$

Distinguishing clearly between the "secret key" and "shared key" components. The data owner safely keeps the secret key, while the shared key is used to create trapdoors, reduce dimensionality effectively, and handle any ciphertext inflation problems that arise during homomorphic operations. Randomly choosing values for $SK(n)$ that satisfy the following requirements is part of the key generation process:

- The method generates a minimum of $'n'$ subciphertexts where $n \geq 3$.
- The total sum is the following equation (2) used to ensure valid key construction.,

$$a_n + b_n + c_n \neq 0 \tag{2}$$

- Both a_i and b_i are positive integers, with $1 \leq i \leq n$, and their GCDs (Greatest Common Divisors) must be greater than 1 and not equal to each other.
- There is a unique element e (where $1 \leq e < n$) such that $c_e \neq 0$, allowing secure comparisons between data. In our method should be the following equation (3),

$$c_e = \left(b_e + a_e\right) * \omega \tag{3}$$

where ω represents a numeric gap designed to maintain ordering without revealing data equality. Additionally, we utilize a list of random numbers using the equation below for the encryption process.

$$T = \left[t_1, \ldots, t_{n-1}\right] \tag{4}$$

These random numbers are chosen as positive non-zero integers between 1 and ω, ensuring that t_e, corresponding to the e-th The i-th element of the secret key is larger than any other random value. In addition to improving security, this methodical approach maximizes the effectiveness of data processing and integrity checking in cloud systems.

4.2.2. Trapdoors Calculation

The suggested approach addresses ciphertext inflation by reducing dimensionality using trapdoors. A series of trapdoors is given by the following equation (5),

$$Trap = \left[trap_1, \ldots, trap_n\right] \tag{5}$$

It corresponds to the secret key, with the final element, $trap_n(abc)$, calculated separately due to its significance. The process begins with the calculation of the Greatest Common Divisors (GCD) of subkeys, followed by defining Trap and shared key components $(Shared\ b\ and\ Shared\ a)$ for trapdoor generation. Iterative calculations refine the shared keys and trapdoors, with abc the value computed by the following equation (6),

$$abc = a_n + b_n + c_n \tag{6}$$

This approach ensures optimized ciphertext handling and efficient encryption performance.

4.2.3. Encryption

The following equation (7) describes the suggested encryption function that transforms a plaintext value into x m subciphertexts using the secret key $SK(n)$.

$$E(x) = \{E_1,...E_n\} \tag{7}$$

Where,

$$E_1 = \frac{(a_1 \times c_1 \times x + b_1 + a_1 \times (t_i - t_{i-1}))}{b_i} \tag{8}$$

$$E_n = a_n + b_n + c_n \tag{9}$$

The order of the plaintext in the designated e^{th} subcipher (e_e). The process involves iterative steps, including the use of a ciphertext level counter (l) to track the dimensionality reductions applied, ensuring scalability for multiple levels. The encryption adheres to conditions from the key generation phase, ensuring secure and efficient ciphertext generation while preserving plaintext ordering for encrypted comparisons.

4.2.4. Decryption

The decryption function reconstructs the plaintext value x from the ciphertext , as given by equation (10),

$$E = \{E_1,....,E_n\} \tag{10}$$

The value is calculated as in the following equation (11) C ,

$$c = \sum_{i=1}^{n-1} c_i \tag{11}$$

After the cyphertext's dimensionality has been decreased, the new subciphertext value for each subcipher E_i in E is as following equation (12). To generate an accurate decryption, this step is necessary.

If $E.l \neq 0$,

$$E_i = E_i \times \frac{\left(Secret\, b^{E.l} \Big/ Secret\, a^{E.l} \right)}{c^{E.l}} \tag{12}$$

$$b = \frac{E_n}{a_n + b_n + c_n} \tag{13}$$

Finally, the decoded value is calculated as in the following equation (14) x ,

$$x = \frac{\sum_{i=1}^{n-1} \left((E_i \times b_i) - (b \times b_i) \Big/ a_i \right)}{c} \tag{14}$$

By reversing the dimensionality reductions applied during encryption. This process ensures accurate and secure decryption while handling the modifications introduced by dimensionality reduction.

4.2.5. Ciphertext Multiplication and Dimensionality Reduction

Ciphertext multiplication in the proposed method follows the MLS scheme, where two ciphertexts, E_1 *and* E_2, are multiplied to produce a new ciphertext of size; to address the ciphertext inflation issue m^2. , dimensionality

reduction is applied using trapdoors and the a_{bc} value, reducing the ciphertext size back to n without decryption. The reduction process involves iterating over the n^2 subciphertexts, recalculating subciphertexts with trapdoor-based transformations, and storing them in a reduced ciphertext RE. This ensures computational efficiency and scalability while preserving the homomorphic properties. The operation combining multiplication and dimensionality reduction is denoted by η, while plaintext-ciphertext multiplication is denoted by $\otimes$.

The proposed method for FHEbRMI begins with key generation, where a public key (PK) is distributed to the client and a secret key (SK) is securely stored on the server. The client prepares input data and encrypts it using the PK to obtain ciphertext. This encrypted request is sent to the server, which, without decrypting the ciphertext, applies homomorphic calculations to it. The server returns the encrypted results to the client after finishing the calculations. The original data is kept secret during the process while the client uses the secret key to decrypt SK the response and acquire the plaintext output. Additionally, error handling and security measures are implemented to ensure data integrity and privacy. Both the proposed FHEbRMI method and its flowchart are displayed in Figure 3.

Algorithm Proposed FHEbRMI		
Start		
{		
	Key generation ()	
	{	
		Int $SK(n), a_1, b_1, c_1$;
		$SK(n) = [(a_1, b_1, c_1), ..., (a_n, b_n, c_n)]$; //using equation. (1)
		// Generate the secret and shared keys based on the MLS scheme
		$T = [t_1, ..., t_{n-1}]$; // using equation (4)
		// For encryption Process
	}	
	Trapdoor Calculation ()	
	{	
		Int, $Trap, trap_1, trap_n$;
		// Compute trapdoor values corresponding to the secret keys
		$Trap = [trap_1, ..., trap_n]$; // using equation (5)
		// dimensionality reduction
	}	
	Encryption phase ()	
	{	
		Int $E(x), E_1, E_n$;
		$E(x) = \{E_1, ...E_n\}$; // using equation (7)
		// plaintext into a sequence of sub-ciphertexts using the MLS
		$E_n = a_n + b_n + c_n$; // using equation (9)
		// for encrypted comparisons
	}	
	Decryption Phase ()	
	{	
	Int E ;	
		$E = \{E_1, ..., E_n\}$; // using equation (10)
		// ciphertext back to its original plaintext.

		$x = \frac{\sum_{i=1}^{n-1}\left((E_i \times b_i) - \left(b \times b_i\right)/a_i\right)}{c}$ // using equation (14)
		//calculate the decoded decryption
	}	
	Ciphertext Multiplication and Dimensionality Reduction ()	
	{	
		// Reconstruct the data accurately even after dimensionality reduction
		// Complete the secure data processing cycle with encrypted computations.
	}	
}		
Stop		

The suggested FHEbRMI works in such a way that secure key generation using the Modified Least-Squares (MLS) scheme to optimize coefficients is the starting point of the mechanism. The encrypted blocks of data are passed over RMI interfaces so that the server can perform homomorphic operations (addition and multiplication) on encrypted data without decrypting it, and guarantees that privacy is maintained along the chain of computation. After processing the client is given back the encrypted result. The client then uses his secret key to decode the output to arrive at the final output. It ends with the security considerations where there is protection of privacy of the data in the RMI process as the sensitive information is encrypted during the transmission and processing.

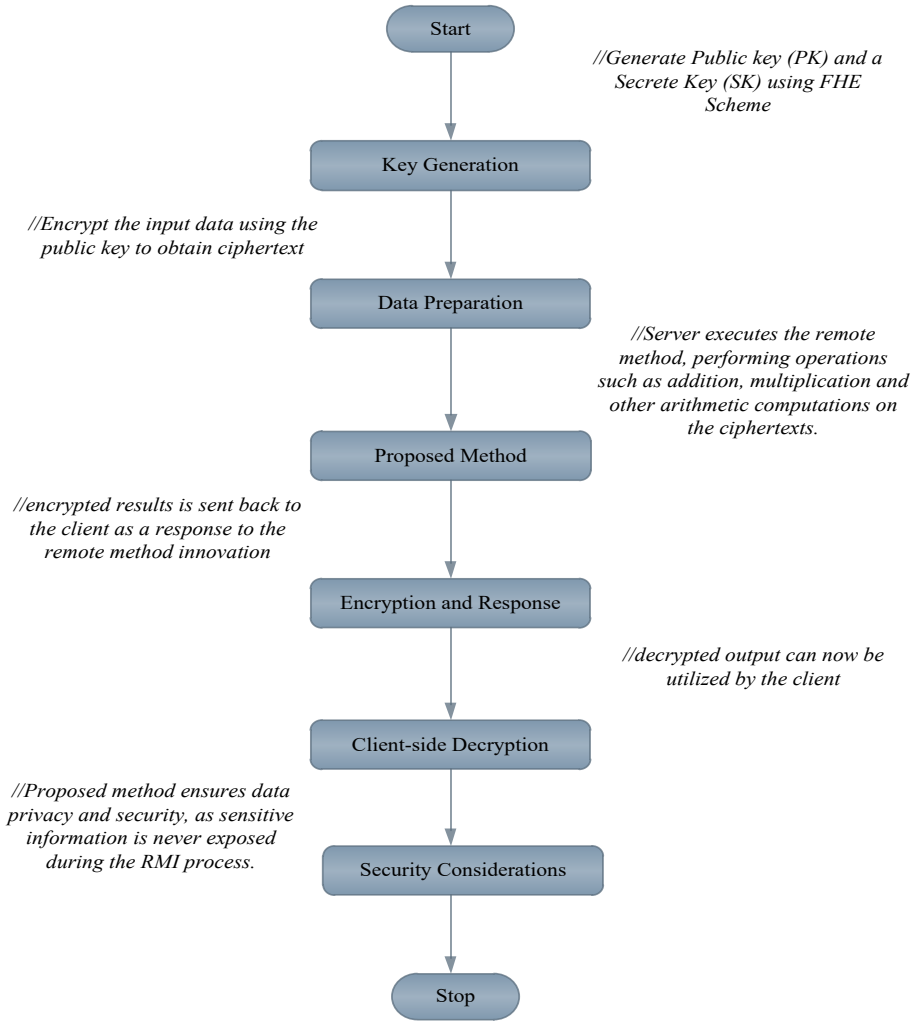

Fig. 3. Flow chart of proposed FHEbRMI.

The ultimate advantage of this method is that it can be used to safely do remote computing where the server will be able to handle encrypted data and will never be shown the actual data so the privacy of the data is preserved throughout the process.

5. Results and Discussion

The details of the parameters are presented in Table 1.

Table 1. Parameter Description.

Parameter	Specification
Processor	Intel(R) Core(TM) i5-3570 CPU @ 3.40GHz, 4 cores
Installed RAM	8.00 GB (7.88 GB usable)
Operating System	Windows 10 Pro,
Version	22H2
Development Platform	NetBeans IDE
Java Version	Java 8.02
Security Level (sec)	128 bits
Lattice / vector dimension (n)	2048
Modulus size (bit-length) $\lvert q \rvert$	4096 bits
Number of subciphertexts (m)	8
Threads per node (T)	min(num_chunks_node, physical_cores) threads per node (baseline: 1 thread per physical core).
Chunk size per thread	8 MB

5.1. Performance estimation and comparison

The proposed work parameters are compared with existing protocols such as Multithreaded RSA, Simple RSA, and Distributed Computing.

5.1.1. Text File

A text file is a digital file that is usually encoded using character encoding standards like ASCII or UTF-8 and contains plain text data that is human-readable and editable. Documents, source code, configuration files, and more may all be found in text files.

The size of a text file can be estimated using the following equation (15):

$$File\ Size\ =\ Number\ of\ Characters\ \times\ Bytes\ Per\ Character \tag{15}$$

where the number of characters is the total number of characters in the text file, including letters, numbers, spaces and punctuation. Bytes per character is the number of bytes used to encode each character.

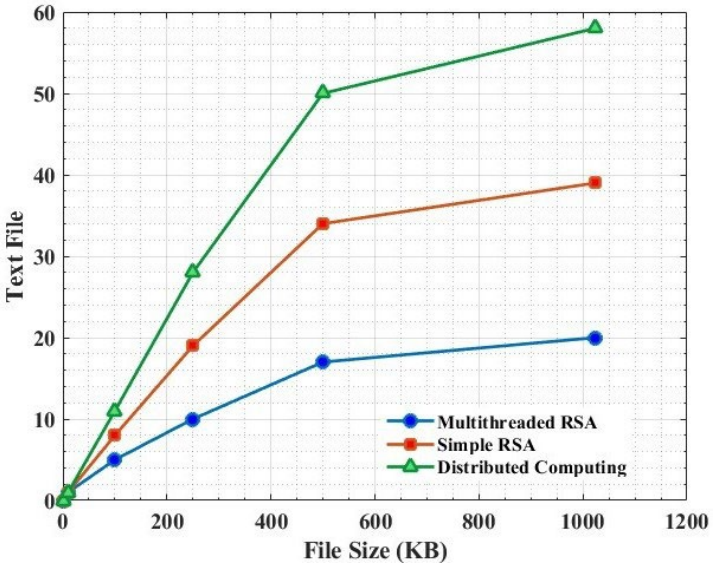

Fig. 4. Comparison of Text File.

Figure 4 shows a performance comparison between three computing methods: Multithreaded RSA, Simple RSA, and Distributed Computing, when handling text files of different sizes. Distributed Computing consistently outperforms the other two methods, reaching the highest efficiency, while Multithreaded RSA shows the lowest performance. All three approaches show a similar pattern where performance increases rapidly at first but then levels off as file sizes grow larger.

5.1.2. Word File as Input

A document produced using Microsoft Word or a comparable word processing program is referred to as a Word file. These files usually contain formatted text, graphics, tables, and other material and have extensions like.doc or.docx. The quantity of text, formatting complexity, embedded objects, and other document components are some of the variables that affect a Word document's size. Although formatting and embedded information might affect a Word document's precise size, the following equation (16) is a basic estimate of the file size:

$$File\ Size\ =\ \left(Number\ of\ Characters \times Bytes\ \ per\ Character\right) + Overhead \tag{16}$$

Where, Number of Characters is the total number of characters in the document, including letters, numbers, spaces, and punctuation. Bytes per Character is the average number of bytes used to encode each character, which can depend on the encoding (typically 1 byte for ASCII or an average of 2 bytes for UTF-16). Overhead is a set amount that takes into consideration the extra data needed by the file format (headers, metadata, and formatting information). The intricacy of the document might have a substantial impact on this cost.

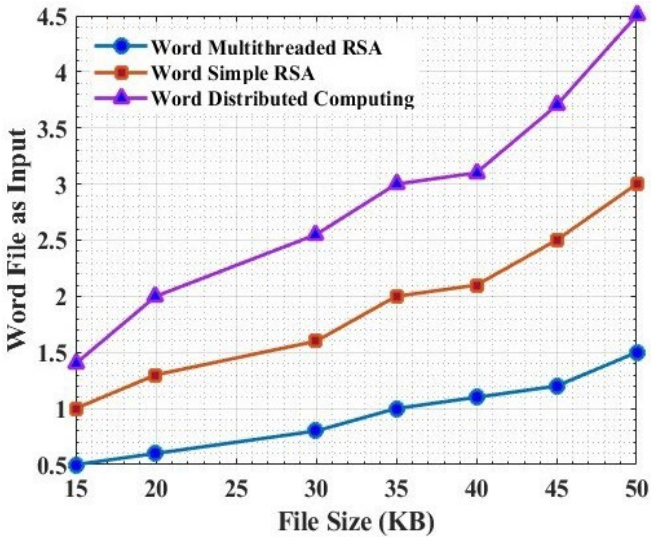

Fig. 5. Comparison of Word File Input.

Figure 5 compares the same three computing methods (Word Multithreaded RSA, Word Simple RSA, and Word Distributed Computing) but specifically for Word file inputs. The file sizes range from 15 to 50 KB, and the performance is measured in terms of input processing. Similar to the previous graph, Word Distributed Computing (purple line) shows the best performance, reaching about 4.5 units, followed by Word Simple RSA (red line) at around 3 units, while Word Multithreaded RSA (blue line) performs the lowest at approximately 1.5 units.

5.1.3. PDF File

The amount of digital storage space that a PDF (Portable Document Format) file takes up on a storage device is known as its file size, and Bytes (B), kilobytes (KB), megabytes (MB), or gigabytes (GB) are the most frequent units of measurement. The size of a PDF file can be estimated using the following equation (17):

$$PDF\ \ Size\ =\ Text\ Size\ +\ \mathrm{Im}age\ Size\ +\ Font\ Size\ +\ Other\ Elements\ +\ Metadata \tag{17}$$

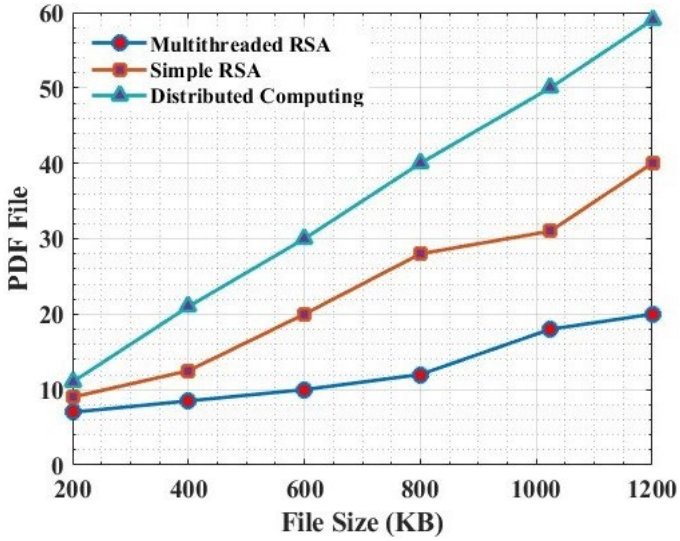

Fig. 6. Comparison of PDF File.

Figure 6 illustrates the performance comparison of three methods (Multithreaded RSA, Simple RSA, and Distributed Computing) when processing PDF files ranging from 200 to 1200 KB. Following the pattern seen in previous figures, Distributed Computing (blue line with triangles) shows superior performance, reaching about 60 units at maximum file size, while Simple RSA (red line with squares) achieves moderate performance around 40 units, and Multithreaded RSA (blue line with circles) demonstrates the lowest efficiency at about 20 units. The graph shows linear growth for Distributed Computing, while the other two methods show more gradual increases.

5.1.4. Image File

The size of an image file is often measured in bytes, such as gigabytes (GB), megabytes (MB), and kilobytes (KB), is the amount of digital storage space that an image file takes up on a storage media. A number of variables affect the file size, such as the picture format (e.g., JPEG, PNG, GIF), color depth (bits per pixel), and image dimensions (width and height).

For uncompressed images, the file size can be calculated using the following equation (18):

$$File\,Size = Width \times Height \times \left(\frac{Color\,Depth}{8}\right) \tag{18}$$

Where width is the image's width expressed in pixels. Height is the image's height expressed in pixels. The amount of bits required to describe a single pixel's color is known as color depth; typical values are 32 bits for true color with transparency and 24 bits for true color.

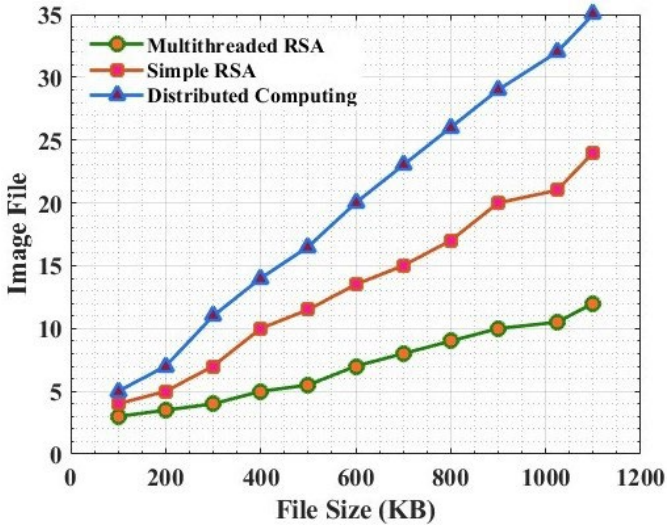

Fig. 7. Comparison of the Image File.

Figure 7 shows how three different computing methods handle image files of varying sizes from 0 to 1200 KB. Distributed Computing (blue line with triangles) consistently performs the best, reaching about 35 units, while Simple RSA (red line with circles) shows medium performance at around 24 units, and Multithreaded RSA (green line with circles) performs the lowest at about 12 units. All three methods show fairly linear growth as file size increases, though Distributed Computing maintains the steepest and most consistent upward trend.

5.1.5. Latency

Figures 8 and 9 show a comparative study of the encryption and decryption time of the proposed Homomorphic Hybrid Encryption (HHE) scheme and the Martino Homomorphic Encryption algorithm. The x-axis is used to plot the size of data (in KB) and the y-axis is used to plot the encryption/decryption time (in milliseconds). Based on the findings, it is clear that the suggested HHE always occupies shorter time latencies when performing both encryption and decryption tasks with respect to the sizes of different data. In particular, with the data size growing beyond 50 KB to 500 KB, the encryption time of the Martino scheme grows at an alarming rate beginning with around 300 ms and then to 2500 ms, but the encryption time of the proposed HHE grows at a slow rate starting with 200 ms and ending with around 1500 ms, or more exactly, by forty percent. On the same note, decryption time of the Martino scheme is up to 2000 ms with 500 KB, whereas the proposed HHE has a much lower time of approximately 1200 ms. This latency decrease is mainly attributed to the dimensionality reduction and optimized trapdoor computation that is incorporated into the HHE design which reduces the ciphertext growth and enhances the computational efficiency. Therefore, the proposed approach shows better scalability and speed of processing data which allows the application to be more applicable in cloud-based applications that need real-time encryption and decryption of data in large sizes.

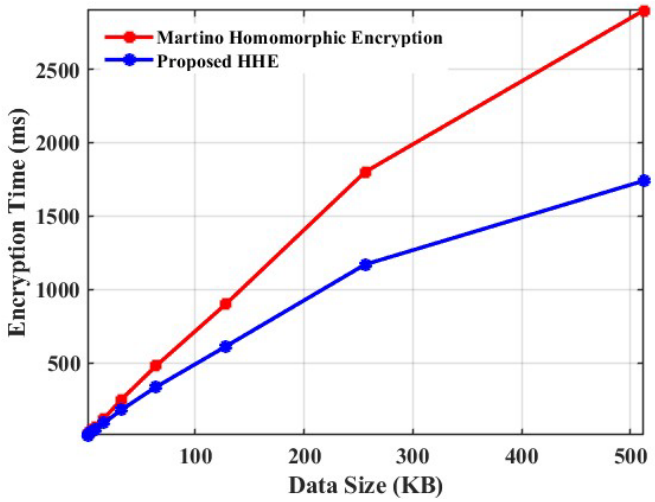

Fig. 8. Comparison of Encryption Latency.

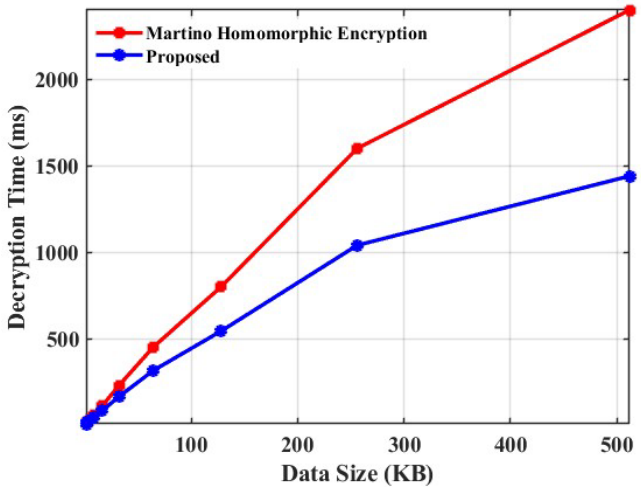

Fig. 9. Comparison of decryption Latency.

5.1.6. Computational cost per operation

Figure 10 provides a comparative study of the computational cost per operation of three encryption schemes CSIAS, PPPAE, and the proposed Homomorphic Hybrid Encryption (HHE) scheme. The x-axis is the file size (in MB) and the y-axis is the time (in seconds) of computation time of each operation. The computational cost of all three methods is proportional to the file size, that is, the larger the file size, the higher the computing cost, but the slower rate of change in the proposed HHE. Computational overhead is the highest in CSIAS scheme (almost 1000 seconds per 1 GB file), and PPPAE is not the worst (approximately 600 seconds). Conversely, the proposed HHE comes out as the most valuable one and it takes only approximately 400 seconds to operate with the same amount of data implying that it would have improved the computational efficiency by about 35–40 percent. This has largely been due to the optimal reduction of the dimensionality and efficient trapdoor-based key operations in the HHE scheme which ensures minimal redundant computation in the encryption and the homomorphic evaluation. Therefore, proposed method has better scalability, reduced computation and greater applicability to the large-scale and cloud-based storage and secure data auditing system.

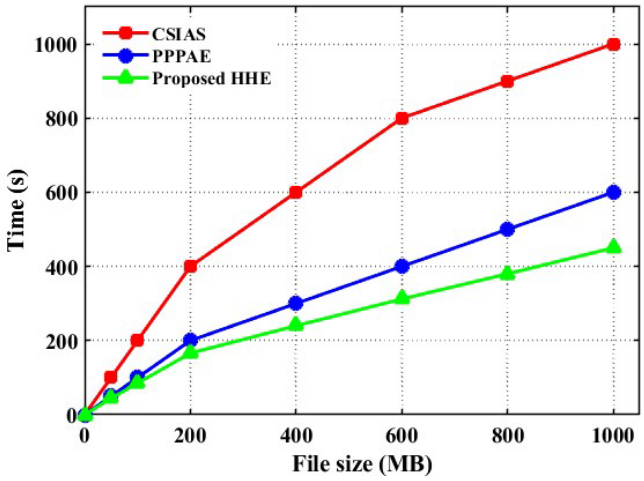

Fig. 10. Comparison of Computational cost per operation

5.1.7. Encryption overhead

The encryption overhead of the schemes HHE, PSO-OK-HECCFHE, JAYA-OK-HECCFHE, ASMO-OK-HECCFHE, HPB-OK-HECCFHE, and the Proposed Approach is compared for file sizes of 500 KB, 500 KB, and 5000 KB as shown in Figure 11. The y-axis will be used to show the encryption overhead (in milliseconds) and the x-axis will be used to show the file size (in KB). Based on the graph, one can note that there is an increase in the encryption overhead as the file size increases in all the methods. Nevertheless, the Proposed

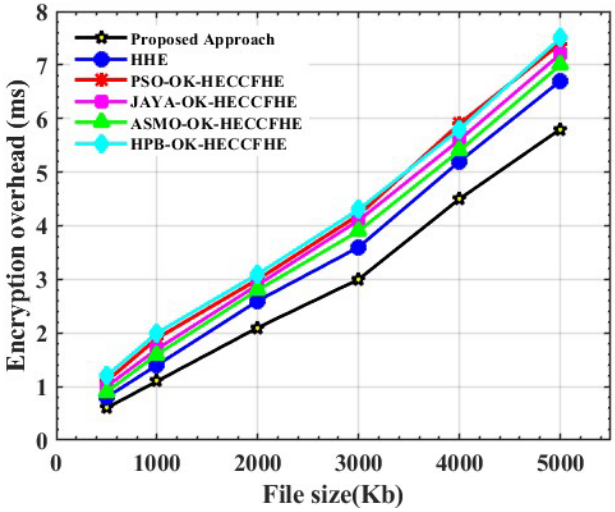

Fig. 11. Comparison of Encryption Overhead..

Approach always has the smallest encryption overhead among others. An example of this is at a file size of 5000 KB the proposed method will record an overhead of about 5.5 ms compared to the other methods such as HPB-OK-HECCFHE and ASMO-OK-HECCFHE which record overheads of nearly 7-7.5 ms. The key management is optimized and the dimensionality reduction methods in the proposed homomorphic encryption framework are based on the trapdoor technique which is the major contributor to this reduction in the overhead. These optimizations reduce unnecessary calculation when encrypting data and maximise processing rate without affecting the security of the data. In general, the Proposed HHE-based solution proves to be more efficient as it reduces the encryption overhead significantly thus enhancing computational efficiency and scalability and applicability to real-time cloud data audits.

6. Conclusion

This research presented a Fully Homomorphic Encryption based Remote Method Invocation (FHEbRMI) mechanism integrated with a Modified Least-Squares (MLS) optimization scheme to ensure privacy-preserving data auditing in cloud environments. The proposed approach enables secure computation directly on encrypted data, ensuring that sensitive information remains protected during storage, transmission, and processing. Experimental results demonstrated that FHEbRMI achieves improved efficiency and scalability compared to traditional methods such as Simple RSA, Multithreaded RSA, and Distributed Computing across multiple file types. However, despite its strong privacy guarantees, the framework still faces practical limitations such as high computational overhead, increased key size, and processing latency inherent to FHE operations, which can hinder large-scale deployment in real-time applications. Future enhancements will focus on optimizing encryption efficiency, reducing ciphertext expansion, and integrating lightweight cryptographic techniques and distributed workload management to improve practicality. With these optimizations, FHEbRMI has the potential to evolve into a viable and secure framework for privacy-preserving computation in emerging domains such as IoT, healthcare, and large-scale data analytics.

All the Declarations and Statements

Author Contributions Statement

Deepshikha A. Chaturvedi – Conceptualization, Methodology, Software, Validation, Formal analysis,Investigation, Resources, Data Curation, Writing - Original Draft.
Vidyullata Vinayak Devmane – Writing - Review & Editing, Visualization, Supervision, Project administration, Funding acquisition.
Shashikant Sudhakar Radke – Writing - Review & Editing, Visualization, Supervision, Project administration, Funding acquisition.
Shahzia Sayyad – Writing - Review & Editing, Visualization, Supervision, Project administration, Funding acquisition.
Shreeshail Devmane – Writing - Review & Editing, Visualization, Supervision, Project administration, Funding acquisition.
Simran Patel – Writing - Review & Editing, Visualization, Supervision, Project administration, Funding acquisition.

All authors have read and agreed to the published version of the manuscript.

Conflict of Interest Statement

The authors declare no conflicts of interest.

Funding Declaration

This research received no funding.

Data Availability Statement

Data sharing not applicable to this article.

Ethical Declarations

This article does not contain any studies with human participants or animals performed by any of the authors.

Acknowledgments

None

Declaration of Generative AI in Scholarly Writing

The technical work is fully done by the authors of the manuscript, and we have revised the paper to remove repetition and make the writing clearer and more specific.

Abbreviations

The following abbreviations are used in this manuscript:

FHEbRMI - Fully Homomorphic Encryption enhanced Remote Method Invocation
MLS - Modified Least Squares
FHE - Fully Homomorphic Encryption
RMI - Remote Method Invocation

APPENDIX A\B\C…, with appendix tile

None

References

[1] M.K. Bali, A. Mehdi, and M. Singh, "Implementation of AWS Cloud Infrastructure," In 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), pp.1-8, 2023, July. IEEE.

[2] I. Gupta, A.K. Singh, C.N. Lee, and R. Buyya, "Secure data storage and sharing techniques for data protection in cloud environments: A systematic review, analysis, and future directions," IEEE Access, vol.10, pp.71247-71277, 2022.

[3] U. Agarwal, V. Rishiwal, S. Tanwar, R. Chaudhary, G. Sharma, P.N. Bokoro, and R. Sharma, "Blockchain technology for secure supply chain management: A comprehensive review," IEEE Access, vol.10, pp.85493-85517, 2022.

[4] I. El Ghoubach, R.B. Abbou, and F. Mrabti, "A secure and efficient remote data auditing scheme for cloud storage," Journal of King Saud University – Computer and Information Sciences, vol.33, no.5, pp.593-599, 2021.

[5] P. Bhatt, S. Singh, S.K. Sharma, and V. Kumar, "Blockchain technology applications for improving quality of electronic healthcare system," In Blockchain for Healthcare Systems, pp.97-113, 2021. CRC Press.

[6] H.M. Rai, K.K. Shukla, L. Tightiz, and S. Padmanaban, "Enhancing data security and privacy in energy applications: Integrating IoT and blockchain technologies," Heliyon, vol.10, no.19, 2024.

[7] P. Goswami, N. Faujdar, S. Debnath, A.K. Khan, and G. Singh, "Investigation on storage level data integrity strategies in cloud computing: classification, security obstructions, challenges and vulnerability," Journal of Cloud Computing, vol.13, no.1, pp.45, 2024.

[8] M. Berti, and M.P.E. Cunha, "Paradox, dialectics or trade-offs? A double loop model of paradox," Journal of Management Studies, vol.60, no.4, pp.861-888, 2023.

[9] R. Imam, Q.M. Areeb, A. Alturki, and F. Anwer, "Systematic and critical review of rsa based public key cryptographic schemes: Past and present status," IEEE Access, vol.9, pp.155949-155976, 2021.

[10] A. Philips, J. Jayaraj, J. FT, and V. P, "Enhanced RSA key encryption application for metering data in smart grid," International Journal of Pervasive Computing and Communications, vol.17, no.5, pp.596-610, 2021.

[11] S. Banupriya, K. Kottursamy, and A.K. Bashir, "Privacy-preserving hierarchical deterministic key generation based on a lattice of rings in public blockchain," Peer-to-Peer Networking and Applications, vol.14, pp.2813-2825, 2021.

[12] S. Ullah, J. Zheng, N. Din, M.T. Hussain, F. Ullah, and M. Yousaf, "Elliptic Curve Cryptography: Applications, challenges, recent advances, and future trends: A comprehensive survey," Computer Science Review, vol.47, pp.100-530, 2023.

[13] A.S. Alali, R. Ali, M.K. Jamil, J. Ali, and Gulraiz, "Dynamic S-Box Construction Using Mordell Elliptic Curves over Galois Field and Its Applications in Image Encryption," Mathematics, vol.12, no.4, pp.5-87, 2024.

[14] P. Thomadakis, C. Tsolakis, and N. Chrisochoides, "Multithreaded runtime framework for parallel and adaptive applications," Engineering with Computers, vol.38, no.5, pp.4675-4695, 2022.

[15] Z. Que, H. Nakahara, H. Fan, H. Li, J. Meng, K.H. Tsoi, X. Niu, E. Nurvitadhi, and W. Luk, "Remarn: a reconfigurable multi-threaded multi-core accelerator for recurrent neural networks," ACM Transactions on Reconfigurable Technology and Systems, vol.16, no.1, pp.1-26, 2022.

[16] R. Kumar, A. Sachan, and B. Ghoshal, "SLePaaS: An Embedded Platform-as-a-Service Facilitating Research on Thermal Management of Embedded Platforms," IEEE Access, vol.10, pp.90827-90846, 2022.

[17] G. Ramesh, J. Logeshwaran, and A.P. Kumar, "The Smart Network Management Automation Algorithm for Administration of Reliable 5G Communication Networks," Wireless Communications and Mobile Computing, vol.2023, no.1, p.7626803, 2023.

[18] K. Limniotis, "Cryptography as the means to protect fundamental human rights," Cryptography, vol.5, no.4, pp.34, 2021.

[19] M. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. Lauter and S. Lokam, "Homomorphic encryption standard," Protecting privacy through homomorphic encryption, pp.31-62, 2021.

[20] B. Seth, S. Dalal, V. Jaglan, D.N. Le, S. Mohan, and G. Srivastava, "Integrating encryption techniques for secure data storage in the cloud," Transactions on Emerging Telecommunications Technologies, vol.33, no.4, p.e4108, 2022.

[21] A. Kshirsagar, N. Gupta, and B. Verma, "Real Time Facial Emotions Detection of Multiple Faces Using Deep Learning," In Pervasive Computing and Social Networking: Proceedings of ICPCSN 2022, pp.363-376, 2022. Singapore: Springer Nature Singapore.

[22] J. Ryu, K. Kim, and D. Won, "A Study on Partially Homomorphic Encryption," In 2023 17th International Conference on Ubiquitous Information Management and Communication (IMCOM), pp.1-4, 2023, January. IEEE.

[23] P. Chaudhary, R. Gupta, A. Singh, and P. Majumder, "Analysis and Comparison of Various Fully Homomorphic Encryption Techniques," In 2019 International Conference on Computing, Power and Communication Technologies (GUCON), New Delhi, India, 2019, pp.58-62.

[24] J. Wang, F. Wu, T. Zhang, and X. Wu, "DPP: Data Privacy-Preserving for Cloud Computing based on Homomorphic Encryption," In 2022 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), pp.29-32, 2022, October. IEEE.
[25] B.J. Flavia, and B.J. Chelliah, "BO-LCNN: butterfly optimization based lightweight convolutional neural network for remote data integrity auditing and data sanitizing model," Telecommunication Systems, vol.85, no.4, pp.623-647, 2024.
[26] R. Zhou, M. He, and Z. Chen, "Certificateless public auditing scheme with data privacy preserving for cloud storage," In 2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA), pp.675-682, 2021, April. IEEE.
[27] D. Ramesh, R. Mishra, P.K. Atrey, D.R. Edla, S. Misra and L. Qi, "Blockchain based efficient tamper-proof EHR storage for decentralized cloud-assisted storage," Alexandria Engineering Journal, vol.68, pp.205-226, 2023.
[28] A. Oppermann, F.G. Toro, F. Thiel, and J.P. Seifert, "Secure cloud computing: Reference architecture for measuring instrument under legal control," Security and Privacy, vol.1, no.3, p.e18, 2018.
[29] F. Tang, Y. Li, Y. Zhang, W. Susilo, and B. Li, "Real-time privacy-preserved auditing for shared outsourced data," Computer Standards & Interfaces, vol.92, p.103927, 2025.
[30] S.S. Kirubakaran, V.P. Arunachalam, S. Karthik, and S. Kannan, "Towards Developing Privacy-Preserved Data Security Approach (PP-DSA) in Cloud Computing Environment," Computer Systems Science & Engineering, vol.44, no.3, 2023.
[31] Y.M. Gajmal and R. Udayakumar, "Privacy and utility-assisted data protection strategy for secure data sharing and retrieval in cloud system," Information Security Journal: A Global Perspective, vol.31, no.4, pp.451-465, 2022.
[32] P. Goswami, N. Faujdar, G. Singh, K.P. Sharma, A.K. Khan, and S. Debnath, "Stub signature-based efficient public data auditing system using dynamic procedures in cloud computing," IEEE Access, vol.12, pp.58502-58518, 2024.
[33] B. Oluwafemi, "Privacy-preserving computation (homomorphic encryption, MPC)," Journal of Contemporary Educational Research, vol.7, pp.111-122, 2025.
[34] R. Geva, A. Gusev, Y. Polyakov, L. Liram, O. Rosolio, A. Alexandru, N. Genise, M. Blatt, Z. Duchin, B. Waissengrin, and D. Mirelman, "Collaborative privacy-preserving analysis of oncological data using multiparty homomorphic encryption," Proceedings of the National Academy of Sciences, vol.120, no.33, p.e2304415120, 2023.
[35] W. Xu, J. Sun, R. Cardell-Oliver, A. Mian, and J.B. Hong, "A privacy-preserving framework using homomorphic encryption for smart metering systems," Sensors, vol.23, no.10, p.4746, 2023.
[36] J. Ma, S.A. Naas, S. Sigg, and X. Lyu, "Privacy-preserving federated learning based on multi-key homomorphic encryption," International Journal of Intelligent Systems, vol.37, no.9, pp.5880-5901, 2022.

Authors' Profiles

Deepshikha A. Chaturvedi is an Assistant Professor at the Department of Computer Engineering of the Shah & Anchor Kutchhi Engineering College, Mumbai. She received her M.Tech degree in Information Technology from Mumbai University. Her research interests include Machine Learning, Natural Language Processing, Security and Distributed systems. She has more than 20 years of teaching experience and is author of several international conference and journal papers. She has ten copyrights registered, two patents and one book published for her technical research. She is a member of the Indian Society for Technical Education (ISTE) and SAKEC Student Branch Coordinator (SBC) for the Computer Society of India (CSI).

Vidyullata Devmane is a Professor and Head of the Department of Computer Engineering at Shah & Anchor Kutchhi Engineering College, Mumbai. She holds a Ph.D. in Computer Engineering with a specialization in Data Security in Remote Storage. With over 25 years of teaching experience, Dr. Devmane is an approved guide for undergraduate, postgraduate, and doctoral research under the University of Mumbai. She is a prolific researcher, having authored numerous papers in reputed international conferences and journals. Her contributions to the field are further recognized through her role as a reviewer for several esteemed publications. Dr. Devmane holds eleven registered copyrights and has four published patents. Actively engaged in bridging the gap between academia and industry, she collaborates with software industry experts to provide students with hands-on training, internships, and project mentorship. In recognition of her academic excellence and social work, she was honored with the Ideal Teacher Award by the Lion Club Ichalkaranji Pride, Kolhapur District.

Shashikant Radke is an Assistant Professor at the Department of Computer Engineering of the Shah & Anchor Kutchhi Engineering College, Chembur, Mumbai, Maharashtra, India – 400 088. His PhD is in Computer Engineering – Securing Data Access and Automatic Exchanges in a Heterogeneous Ecosystem. He has more than 20 years of teaching experience. He is an approved guide of UG under Mumbai University. He is author of several international conference and journal papers. Also, reviewer of many reputed journals. He has eight copyrights registered and two patents published for his technical research. He is associated with some software industry experts to provide training, internships, project guidance to the students to bridge industry/academia gap.

Shahzia Sayyad received the B.E. and M. Tech. degree in Computer Science & Engineering from the Visvesvaraya Technological University, Belgaum, Karnataka, in 2003 and 2011, respectively. She is an Assistant Professor in Shah and Anchor Kutchhi Engineering College, University of Mumbai. Her research interests include Network Security, Privacy Preservation, Cloud Computing. She is an author of several international conference and journal papers. She has eight copyrights registered, two book chapters and two patents published for her technical research. Shahzia Sayyad is a member of the Indian Society for Technical Education (ISTE).

Shreeshail Devmane is a student of Computer Engineering at Shah & Anchor Kutchhi Engineering College, Mumbai, where he is developing a strong foundation in core computing principles, software development, and emerging technologies. As an enthusiastic learner, he is actively engaged in academic and technical activities that enhance his knowledge and practical skills in the field of computer engineering.

Simran Patel is a student of Computer Engineering at Shah & Anchor Kutchhi Engineering College, Mumbai, where she is building a solid foundation in computing fundamentals, software development, and modern technological advancements. As a dedicated learner, she actively participates in academic and technical activities to enhance her knowledge and practical skills in the field of computer engineering.

How to cite this paper:

Deepshikha Chaturvedi, Vidyullata Devmane, Shashikant Radke, Shahzia Sayyad, Shreeshail Devmane, Simran Patel, "Integrity Checking Mechanism for Privacy-Preserved Auditing of Cloud Shared-Data", International Journal of Computer Network and Information Security(IJCNIS), Vol.18, No.4, pp. 88-105, 2026. DOI:10.5815/ijcnis.2026.04.05