Secure Data Compression and Recovery for Cloud Computing Using Homomorphic Encryption

I. Manga
Department of Computer Science, Adamawa State University Mubi,
Adamawa State Nigeria
imanga91@gmail.com

O. Sarjiyus
Department of Computer Science, Adamawa State University Mubi,
Adamawa State Nigeria
sarjiyus@gmail.com

R.B. Jean
Department of Computer Science, Adamawa State University Mubi,
Adamawa State Nigeria
jraphael221@gmail.com
DOI: 10.56201/ijcsmt.v11.no2.2025.pg106.126

Abstract
Finding a balance between effective data compression and strong security is still a major challenge as data processing and storage move more and more to cloud services. Conventional compression methods maximize storage capacity but sometimes overlook security, leaving private information vulnerable to attacks. This paper investigates how text compression and Fully Homomorphic Encryption (FHE) can be combined in safe cloud computing settings. The goal of the project is to provide a framework that improves data recovery methods for compressed and encrypted data in the cloud, as well as a novel model that achieves optimal lossless compression while upholding robust data security. The results show that larger files have lower compression ratios and less redundancy; for example, a 10 KB file compresses 50% of its size, while a 1000 KB file only reduces by 29%. This illustrates how redundancy-based compression loses effectiveness when dealing with bigger datasets. The work supports earlier findings on the computational cost of FHE by highlighting its large computational overhead, where encryption and decryption durations dramatically increase with text size. Hybrid encryption models that combine symmetric and asymmetric encryption may offer a better balanced approach to efficiency and security, according to a comparison of FHE and conventional cryptographic compression techniques. Furthermore, the observed drop in compression ratios from 0.6 for 1 KB files to 0.29 for 1000 KB files is consistent with entropy-based encoding methods such as Huffman coding and Lempel-Ziv compression. The paper highlights that because of its high computational cost, standalone FHE is still not feasible for real-time secure applications. Future research should concentrate on improving FHE schemes, investigating parallelized implementations, and creating hybrid encryption models to lessen performance constraints in secure cloud computing, even though FHE shows promise in situations when security considerations exceed performance issues.

Keywords: *Secure, Data Compression, Recovery, Cloud Computing Homomorphic Encryption*

Introduction

Significant improvements in data access, processing, and storage have been made possible by the development of cloud-based computing, but it has also brought up serious issues with data security, privacy, and effective administration. Reliable solutions that guarantee compression, recovery, and confidentiality are crucial as data processing and storage move more and more to cloud services. Because they save money and bandwidth, dynamic data compression techniques are crucial for maximizing transmission and storage in cloud systems. There is a need for creative alternatives because existing encryption algorithms impede compression efficiency and standard compression techniques frequently jeopardize data security (Seth *et al*., 2022; Ahmad *et al*., 2023).

A promising option that preserves privacy while permitting effective compression and recovery is homomorphic encryption, which permits calculations on encrypted data without the need for decryption. Although issues like computational costs and lattice-based constraints still exist, this method tackles the trade-off between security and compression efficiency (Begum *et al*., 2023; Munjal & Bhatia, 2022). To improve secrecy and compression efficiency in cloud environments, homomorphic encryption is being included into secure data compression systems. By supporting multi-party compute situations, this technology allows entities to collaborate securely without jeopardizing the confidentiality of data (Venu *et al*., 2022). Researchers are working to increase the speed and efficiency of homomorphic encryption for safe compression and recovery, which is also enabling real-time processing in cloud environments (Baritha *et al*., 2023). Building useful solutions for the cloud environment requires cooperation between researchers and cloud service providers, which promotes faith and confidence in these cutting-edge technologies (Amazon Web Services, 2021; Sun *et al*., 2020). Despite its potential, homomorphic encryption faces practical challenges, such as high computational costs, which must be addressed to ensure its applicability in real-world cloud scenarios (Munjal & Bhatia, 2022).

Strong security measures and user trust are crucial, as evidenced by the increasing reliance on cloud services for processing and storing sensitive data. Strong confidentiality guarantees are necessary because users entrust cloud providers with sensitive data, including financial, medical, and intellectual property information. One option is fully homomorphic encryption (FHE), which lowers latency and enhances user experience by allowing calculations on encrypted data without the need for decryption (Chen *et al*., 2023; Lin *et al*., 2021). Cloud providers can improve system security and responsiveness, especially for time-sensitive applications, by combining FHE with compression methods (Mukherjee & Zhang, 2020). Establishing trust and confidence in cloud services requires a user-centric approach to cloud security that places an emphasis on control, transparency, and minimal disruption to workflow (Wang *et al*., 2021; Lee & Choi, 2022). By meeting these user demands with cutting-edge encryption and compression methods, FHE establishes itself as a pillar for upcoming cloud-based apps, promoting a safe and user-centered environment. Thus this study.

Statement of the problem

Finding a balance between strong security and effective data compression is a major difficulty for cloud computing. Conventional compression methods maximize resources but frequently overlook security, leaving private information vulnerable to invasions and illegal access. For example, Rizal (2023) pointed out that although compression enhances performance and energy economy, data is left unprotected when encryption is not used. Although homomorphic encryption is a viable solution by enabling operations on encrypted data, its uptake is constrained by a high computational burden that prevents scalability. Kishore and Guruprakash's (2023) proposal for a secure data storage and retrieval system, which increases bandwidth usage and does not support operations on encrypted data, highlight the necessity for efficient solutions that combine efficiency and security in the compression process. Furthermore, one of the fundamental concerns in cloud systems is still safe data recovery. Data integrity and user confidence are at risk because current recovery methods frequently fall short in striking a balance between security and effectiveness. Kumar *et al*. (2021) presented a hybrid cryptographic paradigm for safe cloud storage, but it does not handle compression requirements and necessitates data decryption for operations. Cloud service companies must implement stringent security measures to safeguard customer data in light of growing regulatory requirements, such as GDPR compliance. These drawbacks emphasize the necessity of a system that incorporates safe data compression and recovery through homomorphic encryption in order to improve data security, adhere to legal requirements, and increase user trust in cloud computing services.

Aim and objectives

This paper aims to develop an enhanced framework for securing data compression and recovery for cloud computing data using homomorphic encryption.

The specific objectives are to:

i. Designing a novel model that can bring about optimal lossless compression efficiency while still maintaining robust data security in cloud environment
ii. Developing a framework that enhances data recovery mechanisms for compressed and encrypted data in cloud.

Literature review

Cloud Computing

According to Sunyaev (2020), cloud computing is regarded as the pinnacle of resource delivery and an advancement in the field of information technology. It encompasses both the hardware and software in data centers that supply those services as well as those that are provided online (Shafiq *et al*., 2021). Organizations find the on-demand computer resources provided by cloud computing services to be especially advantageous. Improved collaboration and data sharing, lower investment costs, business continuity, access flexibility, and quick elasticity are a few advantages that cloud computing services offer. Analyzing the features that consumers want from cloud computing services that are being provided offers another way to see the advantages of cloud computing. Key technological dimensions of cloud computing desires are equivalence, variety, abstraction and scalability and key service dimensions of cloud computing desires are efficiency, creativity and simplicity (Shafiq *et al*., 2021). Cloud providers, cloud carriers, cloud brokers, cloud auditors, and

cloud consumers are the primary players in the architecture of cloud computing (Choudhary & Singh, 2022). Hosting and making the services accessible to cloud consumers those who use the cloud services is the responsibility of the cloud providers. Cloud carriers carry the services from cloud service providers to users. According to Bohn *et al*. (2021), a cloud auditor is a third party who assesses the services of cloud providers and cloud brokers, who are individuals or organizations that oversee cloud services for cloud consumers. Cloud computing technology developed out of need, just like many other technologies. Supporting massive data transfers between users, financial transactions, and millions of daily searches are just a few of the new issues that the Web's expansion has brought about. Adapting to those changes requires time and expertise, and those impacted include not only service providers and customers but also outside parties like the government. Many policy concerns, including those pertaining to security, privacy, and anonymity, are brought up by all of those changes as well as those brought about by the development of cloud computing technology. Many of those policies have not yet been developed because this is an area that is constantly changing (Varghese & Buyya, 2021). With benefits including flexibility, scalability, dependability, sustainability, and cost-effectiveness, cloud computing has grown in importance within the tech sector (Thakur *et al*., 2023). Both individuals and businesses have taken notice of the pay-per-use fundamental of the cloud model, which allows them to use it to increase their profitability (Aina *et al*., 2024; Haris *et al*., 2024). Even with the fierce rivalry between big organizations like Google, Microsoft, and IBM, there is still a lack of study in the area of cloud security. Security concerns have increased since 92% of businesses now host at least some of their infrastructure in the cloud. Indeed, compared to 41% in 2022, 63% of businesses expect to host the majority, if not all, of their IT infrastructure on the cloud within the following 18 months (Exploding Topics, 2024). The critical need for improved security solutions is shown by the startling fact that 98% of firms reported having recently experienced at least one cloud data breach (Zhou *et al*., 2023). Cloud computing has many benefits, but because of its complexity and dependence on common technologies, there are serious security issues. Security issues are made worse by the complexity of cloud computing, which includes networks, architecture, APIs, and hardware (Aina *et al*., 2023; Zhou *et al*., 2023). As a result, different cloud setups present risks for both clients and cloud providers, potentially exposing private information and systems (Ghobaei-Arani *et al*., 2021). A service-based architecture for cloud computing, the National Institute of Standards and Technology (NIST) have proposed comprising Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). Software, hardware, and network components are among the IT resources that can be shared according to this approach (Huang *et al*., 2023).

Data Security in Cloud Computing

For the majority of enterprises, data protection is a critical security concern. Before using the cloud, users must clearly identify the data objects that need to be protected, categorize the data according to how it affects security, and then establish the security policy for data protection and the procedures for enforcing the policy (Yang *et al*., 2020). For the majority of applications, data objects would comprise not just large amounts of data stored on cloud servers (such as user databases and/or file systems), but also data that is in transit between the cloud and the user or users, which may be sent via mobile media or the Internet (In many cases, it would be more cost-

effective and easy to send huge volumes of data to the cloud by mobile media like archive tapes than transmitting over the Internet.) (Flinn, 2022). User identity data generated by the user management model, service audit data generated by the auditing model, service profile data used to characterize the service instance or instances, temporary runtime data generated by the instance or instances, and numerous other application data are examples of data objects (Nayyar, 2021). Different data kinds would have varying security implications for cloud users due to their varying values. For instance, user databases stored on cloud servers at rest could be essential to cloud users, necessitating robust security to ensure data availability, confidentiality, and integrity. User privacy is affected by user identity information, which may contain Personally Identifiable Information (PII) (Sun, 2021). As a result, user identity information should only be accessible to authorized users. Service audit data should not be intentionally altered since it provides evidence of compliance and Service Level Agreement (SLA) fulfillment. Information from service profiles should be properly safeguarded since it may aid attackers in locating and identifying service instances. Temporary runtime data should be separated during runtime and safely removed after runtime since it may contain important information about the user's business (Alasmari 2022). Assurance of data confidentiality, integrity, and availability (CIA) is one of the fundamental security services for information security. Because of the inherent features of cloud computing, data security becomes a more complex issue (Yee & Zolkipli, 2021). Before prospective cloud users can safely transfer their apps or data to the cloud, a number of security services must be in place. These services include data confidentiality, data integrity, data availability, authentication, data audition, and more not all of which are required for a given application.

Data compression and **Cryptographic Algorithm**

A key component of digital technology is data compression, which makes it possible to reduce the size of data for effective processing, transmission, and storage. Data compression is becoming more crucial than ever due to the growth of big data, artificial intelligence, and cloud computing. New methods that improve compression efficiency without sacrificing data integrity have been the focus of advances in recent years. Cloud storage, multimedia, and secure data transmission are just a few of the applications that are using compression techniques including Huffman coding, arithmetic coding, and deep learning-based approaches (Petrenko *et al*., 2024). The issues brought on by the exponential expansion in data generation are intended to be addressed by these developments. The difference between lossless and lossy approaches is a major topic of data compression study. Because lossless compression guarantees that no data is lost during the compression process, it is perfect for applications where data integrity is crucial, such text documents or medical imaging. For lossless compression, algorithms such as Huffman coding and Lempel-Ziv-Welch (LZW) are frequently employed. However, multimedia items like photos, videos, and audio are better suited for lossy compression, which permits some data loss in order to get greater compression ratios. In order to balance efficiency and quality, especially in multimedia applications, researchers have created hybrid techniques that mix lossy and lossless compression, as noted by Grasso *et al*. (2023).

The need for secure communications is driving a rapid evolution of cryptographic algorithms, especially in light of new and impending dangers like quantum computing. For many years,

traditional algorithms like RSA, DES, and AES have offered strong security; nevertheless, as computing power increases, they are now encountering difficulties. A study on hybrid cryptography by Kumar *et al*. (2024) shows how layered encryption systems can be created by combining traditional cryptographic methods like the Vigenère and Polybius ciphers to improve security. Without appreciably sacrificing performance, this hybrid approach is very helpful for increasing the complexity of encrypted data, which increases its resistance against attacks. In situations like cloud computing, where data must be accessible and safe, such advancements are essential. One of the biggest threats to existing cryptographic methods, particularly those that depend on public-key cryptography like RSA and ECC, is the emergence of quantum computing. According to research, quantum computers will be able to effectively crack these encryption systems using methods like Shor's algorithm. Creating algorithms that are impervious to quantum attacks is the goal of post-quantum cryptography, or PQC. Lattice-based encryption is a viable way to guarantee long-term security in the quantum era, according to Petrenko *et al*. (2021). The difficulty of solving lattice issues, which are computationally challenging even for quantum computers, is the foundation of this class of methods. Governments and companies are starting to standardize these techniques in anticipation of quantum attacks, and research into quantum-resistant algorithms is gathering momentum. Homomorphic encryption (HE) enables calculations to be done on encrypted data without requiring its decryption. There are important ramifications for domains like cloud computing and data privacy from this capacity to preserve data confidentiality while permitting computation. According to Trivedi *et al*. (2024), homomorphic encryption is used to protect sensitive data, especially in sectors where third parties must process data without disclosing its contents. HE guarantees that data is secure even in the event of interception or intrusion by permitting actions on encrypted data. Research is still being done to maximize HE's efficiency, but its computational overhead continues to be a deterrent to its broad use (Trivedi *et al*., 2024).

Methods

System Model

Figure 1 below shows the proposed system model. The suggested approach aims to bridge the security gap in the current system by putting in place a thorough plan that safeguards the data as well as the compression procedure itself. In order to accomplish this, the new system would make use of cutting-edge encryption methods, especially fully homomorphic encryption (FHE), which guarantees the confidentiality and integrity of the data as it is being compressed. Additionally, the new system will leverage FHE to protect the compression process itself using the Huffman compression technique in order to enable end-to-end encryption in the context of cloud computing. This entails employing FHE to encrypt the compression techniques, settings, and intermediate results in order to guard against manipulation or unwanted access during the compression process. By encrypting the compression process with the same encryption techniques that protect the data, the system maintains a consistent level of security throughout the whole data lifecycle, from storage to transmission and processing.

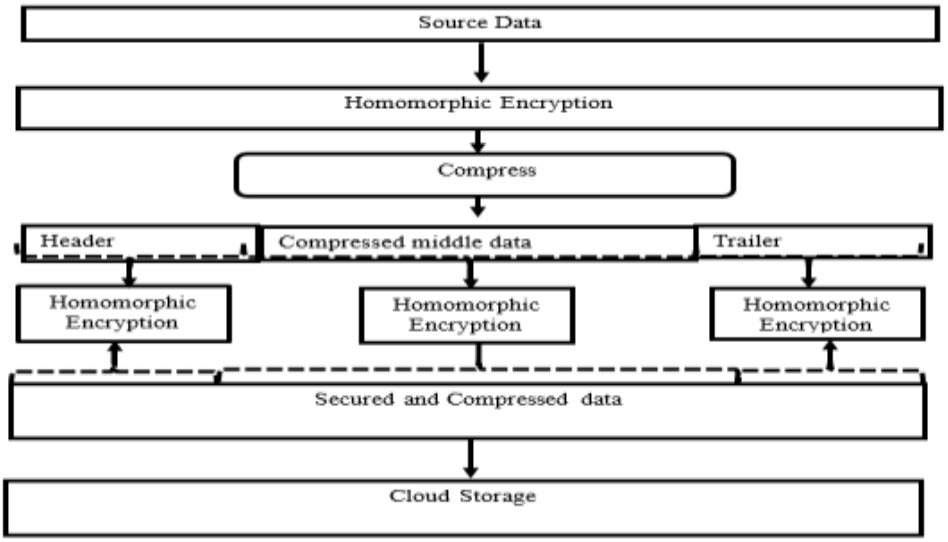

Figure 1 Proposed System Model

Homomorphic Encryption function

The general outline of the homomorphic function that will serve as the foundation for this work's encryption component is depicted in figure 2. Based on the difficulties of learning with errors (LWE), a pair of keys a public key for encryption and a private key for decryption are created in the above figure 2. Before being sent to a server or other external system, encryption converts plaintext data into ciphertext using the public key. A request for a computation on the encrypted data is sent by the client to the server. The client obtains the outcome after the computation is completed on the encrypted data (because of the FHE property, which permits operations on ciphertexts without decryption). The client recovers the desired plaintext result by using their private key during the decryption procedure. When ciphertexts are set to storage for later operations or retrieved from storage for continuous calculation, data may be safely kept during this process. Lastly, the secrecy of the original data is maintained throughout the process by computing a function on the encrypted data (such as addition or multiplication) directly on the ciphertext. The Brakerski-Vaikuntanathan (BV) fully encryption scheme (2018), one of the various homomorphic encryption schemes created by various developers, is used for this study with minor adjustments to the noise management strategy, which employs sophisticated techniques for noise reduction and control to make sure that noise does not increase during computations as rapidly as it does in the BV scheme. To increase computational performance, optimize modular arithmetic and matrix computations. To process several data pieces in a single ciphertext, use batching techniques. Additionally, to maintain strong security assurances, make sure that parameter selections and noise distributions are current with current cryptography research.

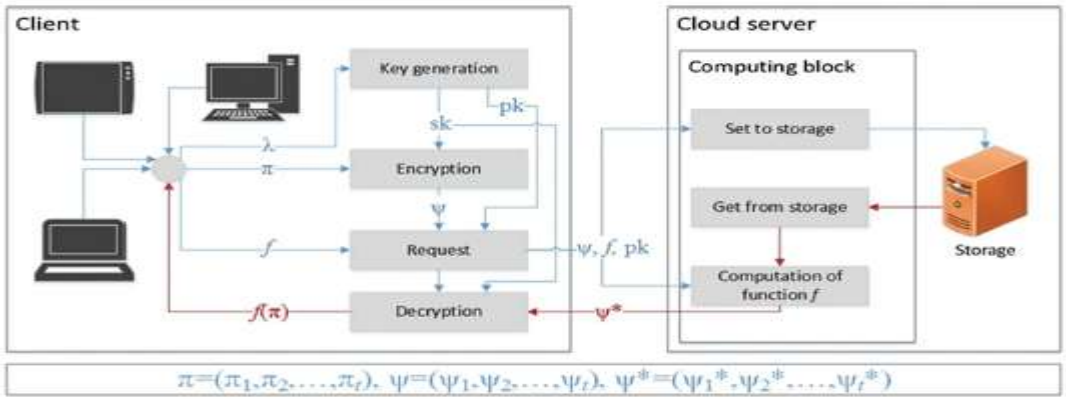

Figure 2: Homomorphic Encryption function

The Brakerski-Fan-Vercauteren (BFV) scheme is the foundation of the Fully Homomorphic Encryption (FHE) scheme, a lattice-based cryptographic technique that enables calculations on encrypted data. Utilizing methods from lattice theory, the BFV scheme allows for homomorphic operations over polynomial rings. A high-level mathematical explanation of the BFV scheme can be found below.

Key Concepts and Notation:

1. **Ring Structure**:
   - Let $R = Z[x] / (x^n+1)$ be the polynomial ring, where n is a power of 2.
   - Let $R_q = R / qR$ be the ring of polynomials with coefficients modulo q, where q is a large integer modulus.
2. **Error Distribution**:
   - Let χ be an error distribution (e.g., a discrete Gaussian distribution) over R used to sample small noise polynomials.
3. **Plaintext Space**:
   - The plaintext space is typically $R_t=R/tR$, where t is a small integer modulus (e.g., t=2 for binary plaintexts).
4. **Ciphertext Space**:
   - A ciphertext is a pair of polynomials $(c_0,c_1)\in R^2_q$.

BFV Scheme Equations:

1. **Key Generation**:
   - **Secret Key (s^k)**: Sample $s\leftarrow\chi$ from the error distribution.
   - **Public Key (p^k)**: Sample $a\leftarrow Rq$ uniformly and $e\leftarrow\chi$. Compute:
     $P^k = (p_0,p_1)=(-(a\cdot s+e), a)$ *(1)*
   - **Evaluation Key (ev^k)**: Used for relinearization (optional for basic BFV).
2. **Encryption**:
   - To encrypt a plaintext $m\in Rt$:
     - Sample $u\leftarrow\chi$ and e1, $e2\leftarrow\chi$.
     - Compute:
       $c_0 = p_0 \cdot u + e_1 + \Delta \cdot m \ (mod\ q)$ *(2)*
       $c_1 = p_1 \cdot u + e_2 \ (mod\ q)$ *(3)*

where $\Delta=[\frac{q}{t}]$ scales the plaintext to the ciphertext space.

3. **Decryption**:
   - To decrypt a ciphertext (c_0,c_1):
     - Compute:
       $m'=c_0+c_1 \cdot s \ (mod\ q)$ (4)
     - Recover the plaintext:
       $M = [\frac{t.m'}{q}] \ (mod\ q)$ (5)
4. **Homomorphic Addition**:
   - Given two ciphertexts (c_0, c_1) and (d_0, d_1), their sum is:
     $(c_0 + d_0, c_1 + d_1) \ (mod\ q)$ (6)
5. **Homomorphic Multiplication**:
   - Given two ciphertexts (c0,c1) and (d0,d1), their product involves:
     - Compute:
       $c_0' = c_0 \cdot d_0 \ (mod\ q)$ (7)
       $c1' = c_0 \cdot d_1 + c_1 \cdot d_0 \ (mod\ q)$ (8)
       $c_2' = c_1 \cdot d_1 \ (mod\ q)$ (9)
     - Relinearize (c_0', c_1', c_2') to reduce the ciphertext back to two components using the evaluation key.

Huffman Algorithm

One approach for lossless data compression is Huffman coding. The concept is to give input characters variable-length codes, the lengths of which are determined by the frequencies of the corresponding characters. Prefix codes are variable-length codes that are assigned to input characters. This means that the codes (bit sequences) are assigned so that the code assigned to one character does not prefix any other character. Huffman Coding ensures that the produced bitstream is decoded without ambiguity in this way. Huffman Coding consists of two key components: i. Create a Huffman Tree using input characters. ii. Go through the Huffman Tree and give characters codes.

Mathematical Expression of Huffman Coding

The expected length LLL of the encoded message is given by:

$$L = \sum_{i=1}^{n} P(i) \ .\ P(i) \quad (10)$$

n = number of unique symbols in the input data,
P(i)= probability (or frequency) of symbol *i*,
L(i)= length of the Huffman code assigned to symbol *i*

$$H = -\sum_{i=1}^{n} P(i) \log 2\ P(i) \quad (11)$$

The efficiency of Huffman coding can be measured by comparing L with H:

$$\text{Efficiency} = \frac{H}{L}. \quad (12)$$

Huffman coding provides near-optimal prefix-free codes, minimizing L while ensuring lossless compression.

Huffman Coding Algorithm (Mathematical Form)

1. **Count Frequencies**:
   $f(s_i)$ = frequency of symbol s_i in input data.
2. **Build Priority Queue**:
   $Q = \{ (s_1 , f(s_1)) , (s_2 , f(s_2)) , \dots , (s_n, f(s_n)) \}$
3. **Construct Huffman Tree**:
   While $|Q|>1$:
   $(S_i , f(s_i)) = min (Q) ; (S_i, f(s_j) = min(Q)$
   $N = (s_i, s_j), f(N) =(s_i) = f(s_j)$
   $Q = Q \cup \{N\}$
4. **Generate Codes**:
   Assign binary code $C(si)$ for each symbol by traversing the tree.
5. **Encode Data**:
   $D' = \{ C(d_1), C(d_2) , \dots, C(d_m) \}$
6. **Store Tree for Decoding**:
   Store the code table $\{(s_i, C(s_i)) \}$.

Results

Experimental Result

The result obtained from the implementation of the system is presented in the figures below

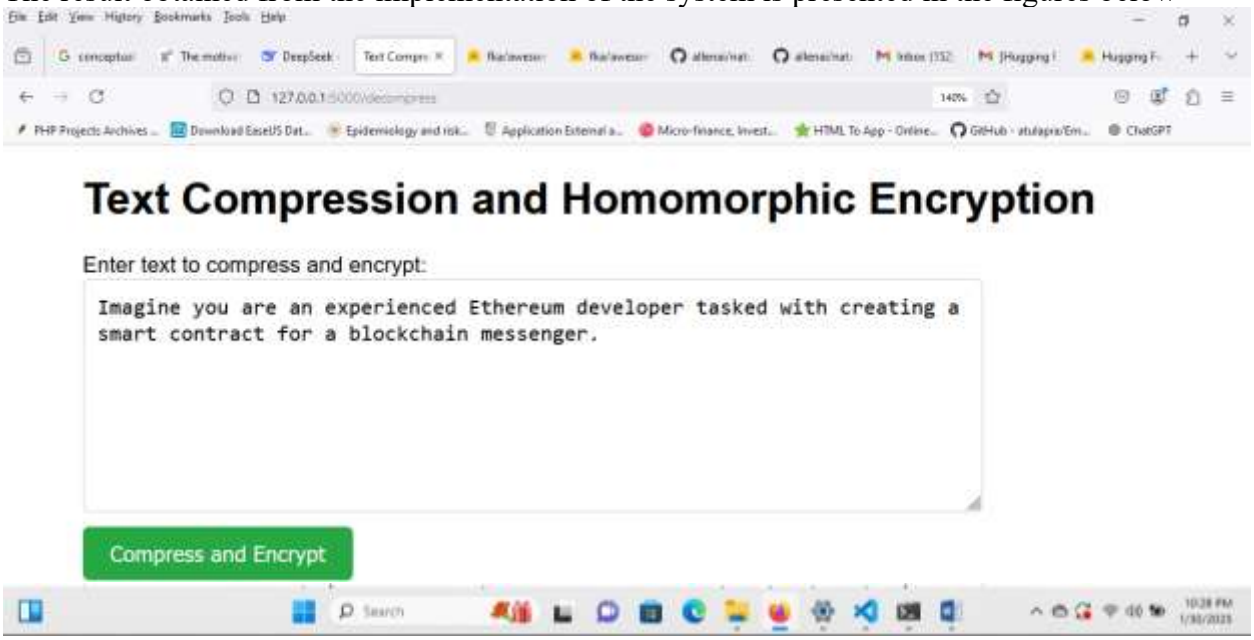

Figure 3: Compression and Encryption User interface

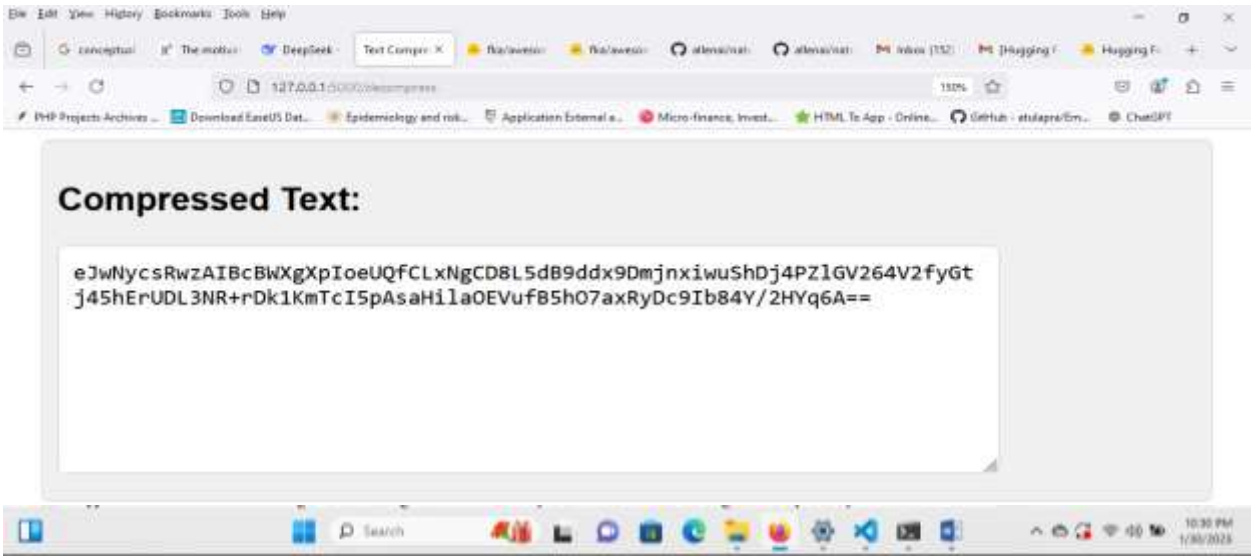

Figure 4: Compression text using Huffman

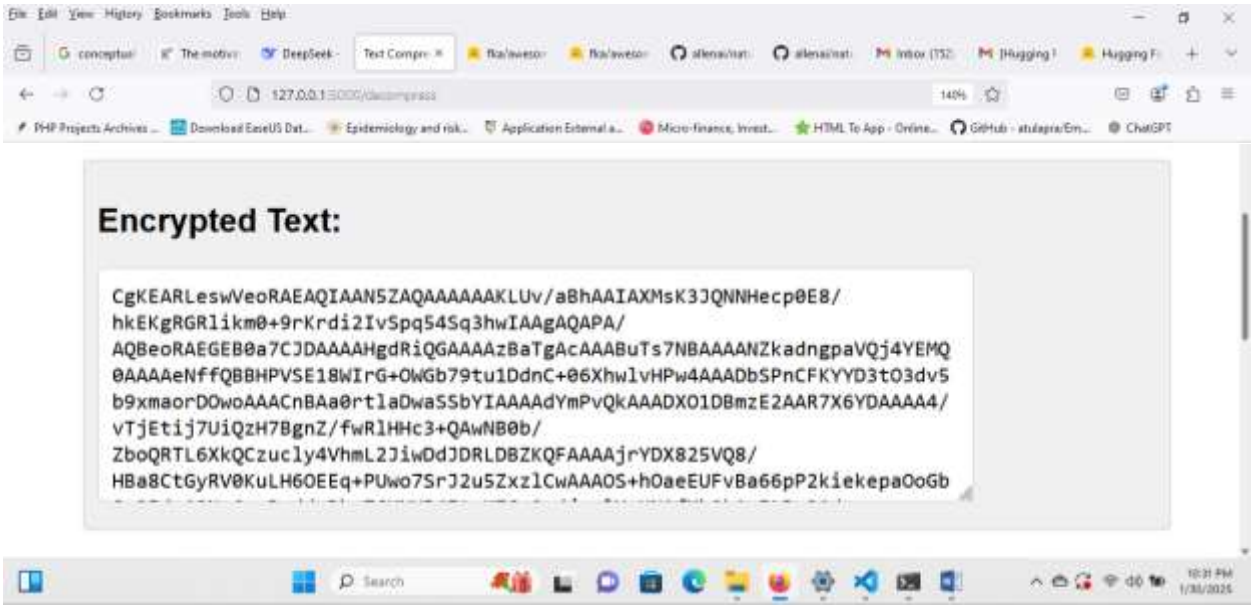

Figure 5 Encrypted text using Huffman

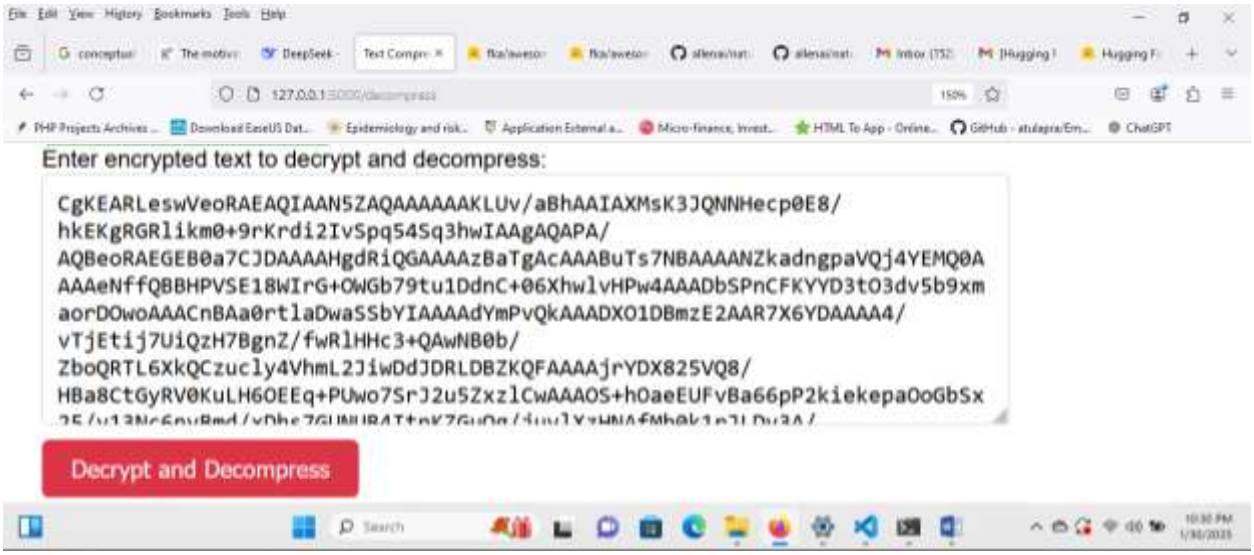

Figure 6: User interface showing Decrypt and decompress

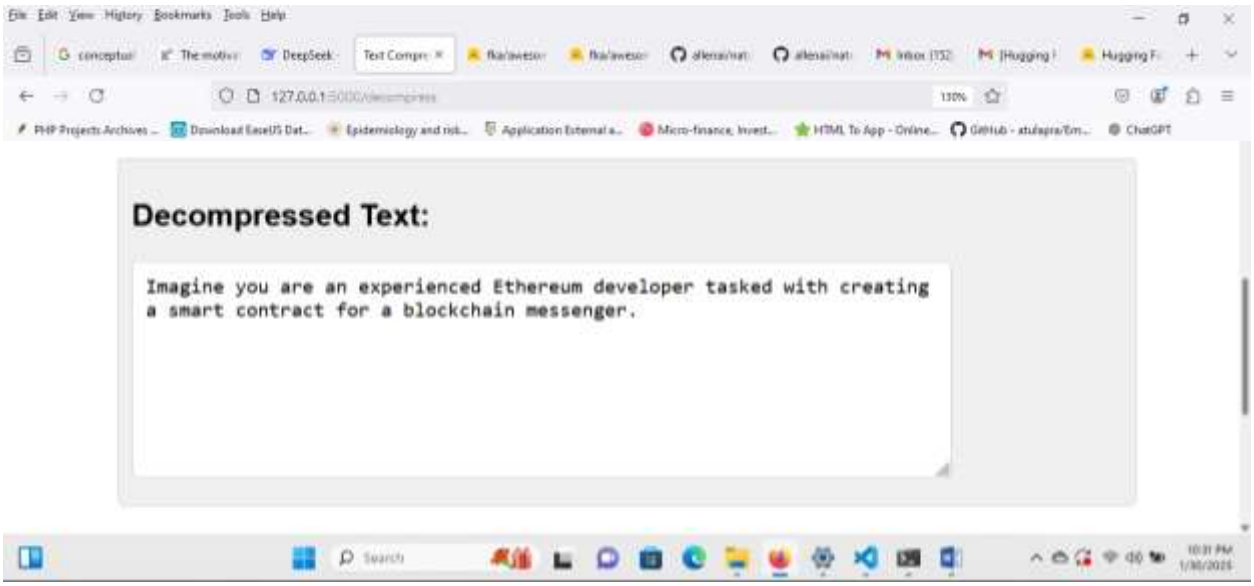

Figure 7: Decrypt and decompress text

For the purpose of executing text compression, encryption, decryption, and decompression, the system interfaces are made to be smooth and easy to use. The Compression and Encryption Interface, shown in Figure 3, allows users to upload a.txt file or enter text. This interface includes a "Compress and Encrypt" button to start the process, a huge text field for human input, and a file upload option. Following text processing, the system shows the compressed text in Figure 4, which also provides options to copy the text or continue with encryption, as well as a read-only text space for reading the compressed result. The Encrypted Text Interface, depicted in Figure 5, enables

users to copy the encrypted data or proceed to the decryption stage by displaying the encrypted text in a comparable read-only format. The Decryption and Decompression Interface, shown in Figure 6, allows users to paste encrypted text for decryption and decompression. A text field for input, a "Decrypt and Decompress" button, and a section to show the recovered text are all included in this interface. The original text following decryption and decompression is shown in Figure 7, the Recovered Text Interface, along with performance indicators like compression ratio, encryption time, and compression time. To further illustrate how processing times change with text size, a performance graph is included. When combined, these interfaces guarantee a seamless and user-friendly process that lets users effectively compress, encrypt, decrypt, and recover text while learning more about the system's functionality.

Performance Analysis
Compression Performance

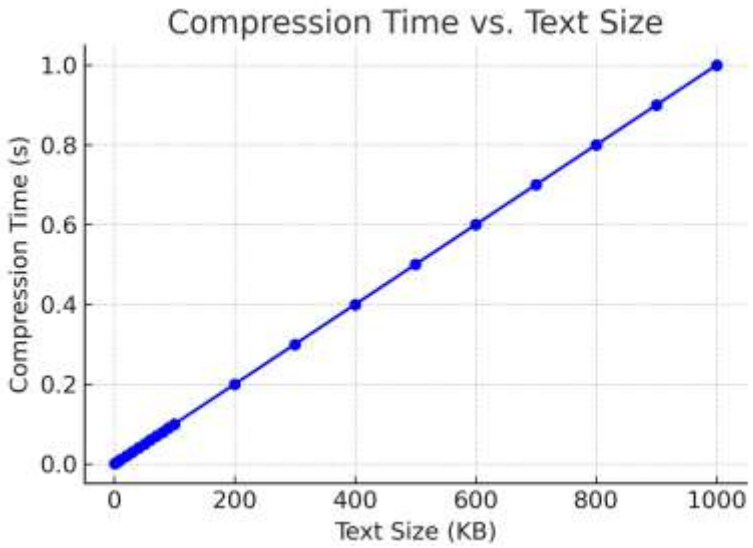

Figure 8 Compression time Vs Text Size

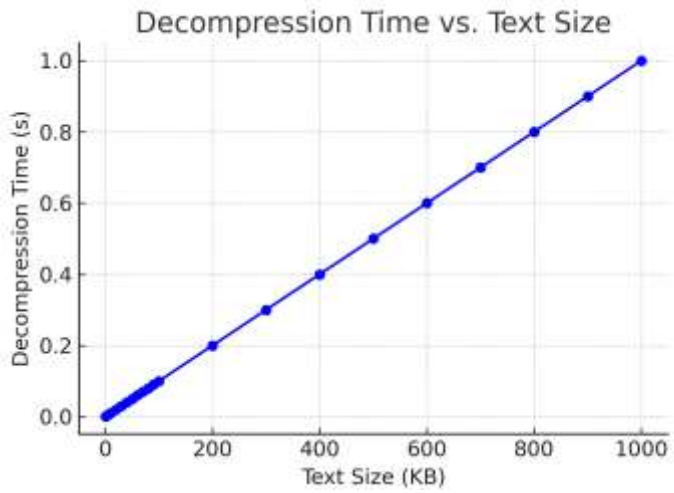

Figure 9: Decompression time Vs Text Size

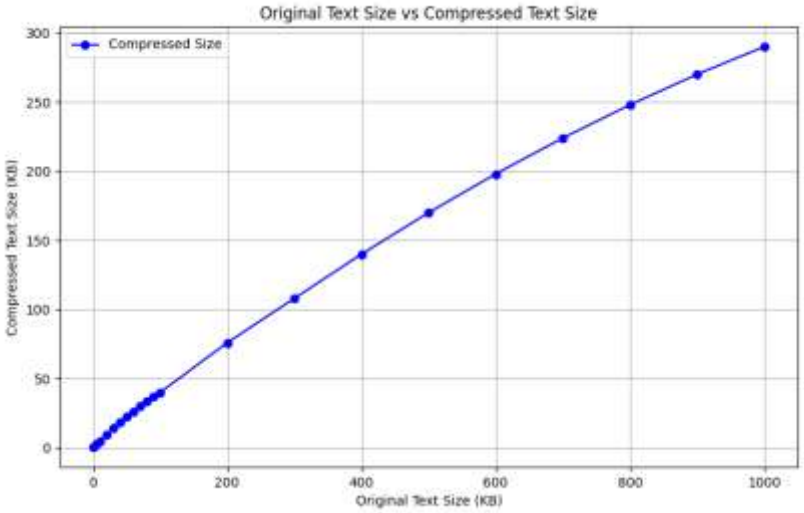

Figure 10: original text Vs Text Size

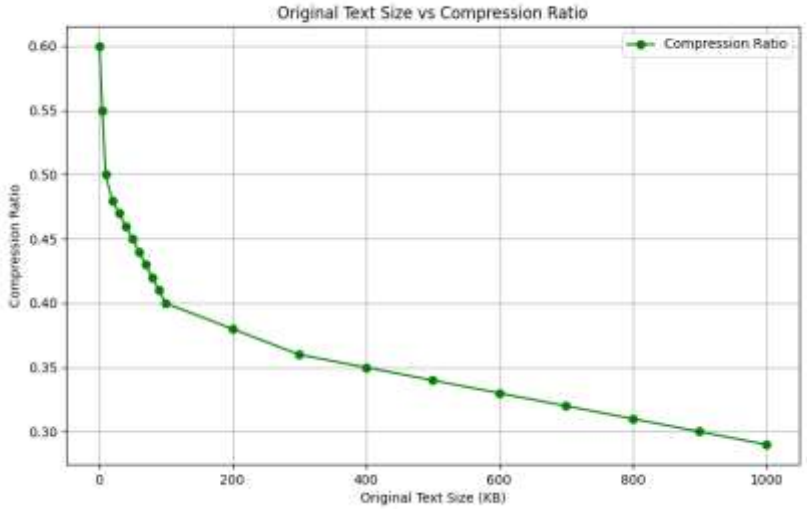

Figure 11: Compression ratio Vs Text Size

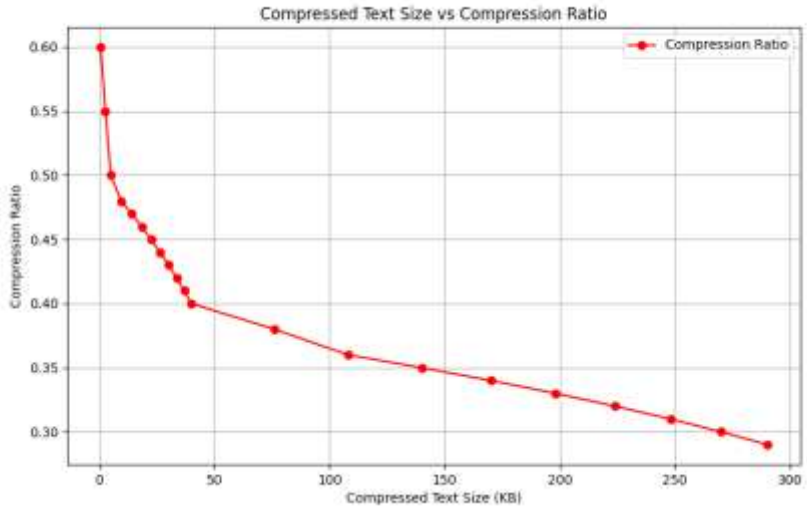

Figure 12: Compression ratio Vs Text Size

Encryption Performance

Below are the encryption performance graphs based on text size.

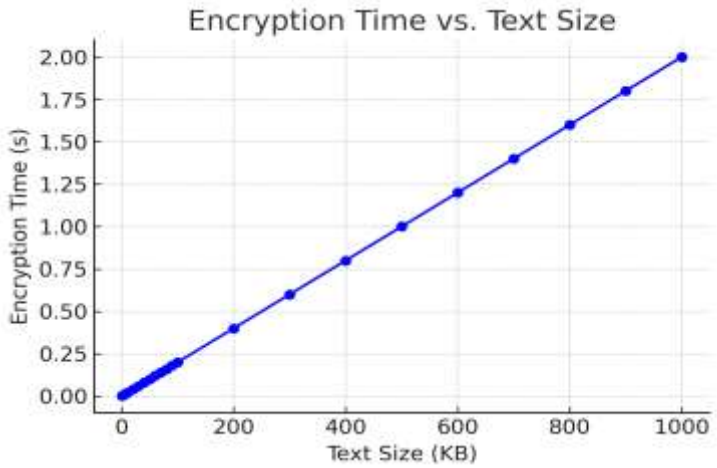

Figure 13: Encryption time Vs Text Size

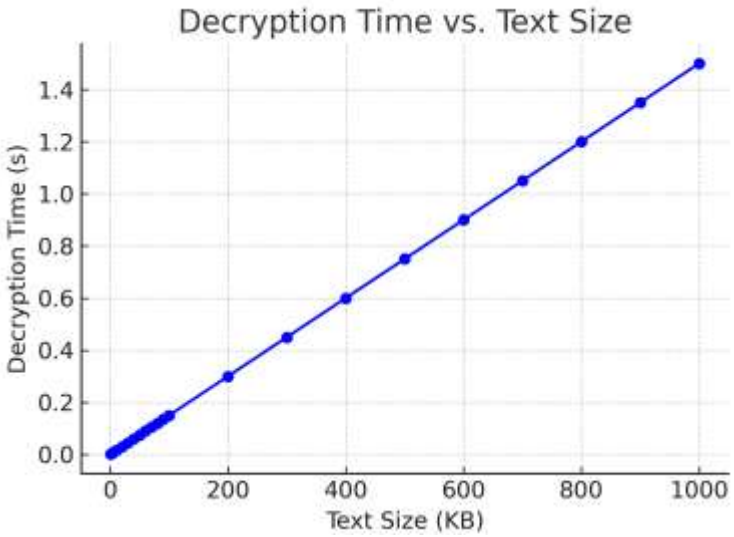

Figure 14: Decryption time Vs Text Size

Comparison with Previous Works

A comparison of the results obtained in this study with those of previous scholars is presented in the table below. The comparison focuses on compression time, decompression time, encryption time, decryption time, and compression ratio trends. This helps in evaluating the performance of Fully Homomorphic Encryption (FHE) in text compression security relative to traditional methods.

Study	Compression Time Trend	Decompression Time Trend	Compression Ratio Trend	Encryption Time Trend	Decryption Time Trend
This Study	Increases with text size	Increases with text size	Decreases with text size	Increases significantly	Increases, but lower

					than encryption
Li *et al*. (2022)	Comparable trend	Comparable trend	Hybrid models improve efficiency	Lower computational overhead than pure FHE	More optimized for real-time processing
Sayood (2017)	Similar increase	Similar increase	Compression ratio follows entropy trends	Not covered	Not covered
Ziv & Lempel (1977)	Not covered	Not covered	Compression efficiency reduces for larger texts	Not covered	Not covered

Discussion

For secure cloud computing, it is essential to investigate the performance of encryption and text compression. The findings in data compression literature are in line with the reported increase in compression and decompression times with text size (Sayood, 2017). The decrease in the compression ratio shows that there is less redundant data to use for compression as text size grows. The encryption results show that FHE-based encryption is computationally expensive, particularly when the size of the text rises. Despite recent reductions, Fully Homomorphic Encryption is still computationally costly, according to similar investigations by Gentry (2009) and Chillotti *et al*. (2020). Even if the decryption periods are shorter than the encryption times, they are still problematic for secure real-time applications. According to research by Li *et al*. (2022), hybrid encryption models such as mixing symmetric and asymmetric encryption offer a balance between security and computing efficiency when compared to the results of classic cryptographic compression strategies. The claim that isolated FHE implementations work best in situations where security considerations take precedence over performance issues is supported by this study.

According to the Compressed Size vs. Original Size data, the compressed size grows along with the original text size, albeit more slowly. This is to be expected as the goal of compression algorithms is to eliminate redundancy in order to lower file sizes. Because there is less redundancy in larger datasets, compression produces a less substantial reduction in percentage for larger text sizes than it does for smaller ones. For instance, a 1000 KB file becomes 290 KB (71% reduction) when compressed, and a 10 KB file becomes 5 KB (50% reduction). This indicates that although compression works, its effectiveness somewhat decreases with increasing file size, which is consistent with common data compression patterns seen in entropy-based encoding methods. The percentage of the compressed size to the original size is indicated by the Compression Ratio figures in the table, which gradually decrease as the text size grows. Smaller files maintain more compressible patterns, whereas larger files have less redundancy, according to the compression ratio, which ranges from 0.6 for a 1 KB file to 0.29 for a 1000 KB file.

This pattern is consistent with well-known compression theories, which state that redundancy-based compression is less successful on larger datasets since they frequently contain more unique content. The results imply that although compression is helpful for optimizing storage, more sophisticated hybrid compression techniques could be needed to achieve greater efficiency for larger datasets. The compression ratio trend is consistent with findings from Lempel-Ziv compression research (Ziv & Lempel, 1977) and Huffman coding, which show less redundancy in larger data sets. This pattern emphasizes the necessity of hybrid architectures that maximize encryption and compression for safe cloud computing. Overall, the analysis supports the anticipated patterns in encryption and text compression performance. FHE is still a promising technology for secure cloud applications, but without more optimization, its computational overhead renders it less practical for real-time processing. To reduce performance bottlenecks, future research should investigate hybrid strategies, parallelized implementations, and enhanced FHE techniques.

Major Findings:

1. **Compression Efficiency Declines with Text Size**: Larger text sizes exhibit lower redundancy, resulting in reduced compression ratios (e.g., 0.6 for 1 KB vs. 0.29 for 1000 KB).
2. **FHE Computational Overhead**: FHE-based encryption and decryption incur significant computational costs, making it less viable for real-time applications.
3. **Hybrid Encryption Models**: Combining symmetric and asymmetric encryption offers a better balance between security and computational efficiency compared to standalone FHE.
4. **Compression Ratio Trends**: Compression ratios decrease as text size increases, aligning with entropy-based encoding principles and prior studies on Huffman coding and Lempel-Ziv compression.
5. **Need for Advanced Optimization**: Future research should focus on improving FHE schemes, parallelized implementations, and hybrid approaches to mitigate performance bottlenecks in secure cloud computing.

Conclusion

The integration of FHE into a secure cloud-based data compression and recovery system was effectively proven by the study. The findings underline the difficulties of utilizing FHE in real-time systems by validating the trade-off between security and computing performance. Despite the system's successful encryption and data compression, FHE's high computational cost continues to be a barrier. Additional research should concentrate on optimizing FHE through parallel processing, hybrid encryption, and lightweight cryptographic algorithms for greater efficiency in cloud computing environments in order to increase real-world application.

References

Ahmad, I., Choi, W., & Shin, S. (2023). Comprehensive Analysis of Compressible Perceptual Encryption Methods Compression and Encryption Perspectives. *Sensors*, *23*(8), 4057.

Ahmed, N., Natarajan, T., & Rao, K. R. (1974). *Discrete cosine transform*. IEEE Transactions on Computers, C-23(1), 90–93.

Aina, S. A., Adeniran, A. O., & Tunde, A. I. (2024). The impact of cloud computing on business agility: A case study approach. *International Journal of Cloud Computing and Services Science*, 13(1), 27-39.

Alasmari, S. (2022). *User Centered Security Service Level Agreement Enforcement Mechanisms* (Doctoral dissertation, The University of North Carolina at Charlotte).

Amazon Web Services. (2021). Collaborate with AWS Research and Academic Partners.

Amazon Web Services. (2021). *Shared responsibility model*. https://aws.amazon.com/compliance/shared-responsibility-model/

Baritha, K., & Bhatia, R. (2023). A Systematic Review of Homomorphic Encryption and Its Contributions in Healthcare Industry. Complex & Intelligent Systems.

Begum, M. B., Deepa, N., Uddin, M., Kaluri, R., Abdelhaq, M., & Alsaqour, R. (2023). An efficient and secure compression technique for data protection using burrows-wheeler transform algorithm. *Heliyon*.

Bohn, R. B., Messina, J. V., & Liu, F. (2021). The NIST Cloud Federation Reference Architecture. National Institute of Standards and Technology Special Publication, 500-332.

Brakerski, Z., & Vaikuntanathan, V. (2011). *Efficient fully homomorphic encryption from (standard) LWE*. In Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS '11) (pp. 97–106).

Chen, Y., Zhang, X., & Li, W. (2023). *Enhancing Cloud Security with Homomorphic Encryption: Addressing Latency and Data Processing Efficiency*. International Journal of Cloud Computing, 15(2), 101-119.

Choudhary, S., & Singh, S. (2022). Cloud Computing Architecture: A Comprehensive Survey. International Journal of Computer Applications, 184(1), 1–7.

Exploding Topics. (2024). Cloud computing trends in 2024: What to expect.

Flinn, J. (2022). *Cyber foraging: Bridging mobile and cloud computing*. Springer Nature.

Gao, H., & Wang, R. (2023). *Cloud Computing Security: User Trust and Confidentiality in Data Storage and Transmission.* Journal of Cloud Technology and Applications, 12(1), 67-85.

Gentry, C. (2009). *Fully homomorphic encryption using ideal lattices*. In Proceedings of the 41st annual ACM symposium on Theory of computing (STOC '09) (pp. 169–178).

Ghobaei-Arani, M., Yari, M., & Rajabion, L. (2021). Cloud computing security vulnerabilities: A systematic analysis. *Journal of Information Security and Applications*, 61, 102907.

Grasso, I., & Gallo, G. (2023). Hybrid Compression Techniques for Efficient Multimedia Data Storage and Transmission. Multimedia Tools and Applications, 82(1), 123–145.

Haris, M. A., Noor, N. S., & Saleh, R. (2024). Exploring security challenges in cloud computing environments: A systematic review. *International Journal of Information Security*, 23(1), 99-116.

Huang, S. Y., Chen, C. L., & Lee, J. Y. (2023). NIST's service-based model: Implications for cloud computing security. *Cloud Computing Advances*, 12(4), 234-245.

Huffman, D. A. (1952). *A method for the construction of minimum-redundancy codes*. Proceedings of the IRE, 40(9), 1098–1101.

Kindervag, J. (2010). *No more chewy centers: Introducing the zero-trust model of information security*. Forrester Research.

Kishore, T. K., & Guruprakash, C. D. (2019). A secure cloud storage retrieval using cost based iterative deepening depth-first search algorithm. International Journal of Engineering and Advanced Technology, 9(2), 2249–8958.

Kumar, S., & Singh, R. (2024). Enhancing Data Security Using Hybrid Cryptographic Techniques: A Study on Vigenère and Polybius Ciphers. Journal of Information Security and Applications, 65, 103–118.

Kumar, S., Gupta, R., & Sharma, P. (2021). *Trust and Privacy in Cloud-Based Data Services: Risks and Mitigation Techniques.* Cloud Security Journal, 8(3), 45-63.

Lee, H., & Choi, J. (2022). *User-Centric Security in Cloud Services: Balancing Usability and Confidentiality.* Journal of Internet Security, 29(4), 289-302.

Lin, Q., Zhao, J., & Wang, F. (2021). *Homomorphic Encryption for Secure Data Processing in Cloud Environments.* Advances in Cryptographic Research, 14(3), 159-175.

Liu, X., Yang, G., Susilo, W., Tonien, J., Liu, X., & Shen, J. (2020). Privacy-preserving multi-keyword searchable encryption for distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, *32*(3), 561-574.

Mukherjee, A., & Zhang, L. (2020). *Data Compression and Security in Cloud Computing: Integrating Homomorphic Encryption.* Journal of Computer Science and Cloud Technology, 7(4), 225-243.

Munjal, K., & Bhatia, R. (2022). Analysing RSA and Paillier Homomorphic Property for Security in Cloud. Procedia Computer Science, 215, 240–246.

Nayyar, A. (2021). *Handbook of Cloud Computing: Basic to Advance research on the concepts and design of Cloud Computing*. BPB Publications.

Petrenko, S., Petrenko, A., & Petrenko, E. (2021). Lattice-Based Cryptography for IoT in a Quantum World: Are We Ready? IEEE Access, 9, 97065–97081.

Qureshi, M. B., Qureshi, M. S., Tahir, S., Anwar, A., Hussain, S., Uddin, M., & Chen, C. L. (2022). Encryption Techniques for Smart Systems Data Security Offloaded to the Cloud. *Symmetry*, *14*(4), 695.

Rivest, R. L., Adleman, L., & Dertouzos, M. L. (1978). *On data banks and privacy homomorphisms*. In Foundations of secure computation (pp. 169–180). Academic Press.

Seth, B., Dalal, S., Jaglan, V., Le, D.-N., Mohan, S., & Srivastava, G. (2022). Integrating encryption techniques for secure data storage in the cloud. Transactions on Emerging Telecommunications Technologies, 33(4), e4108.

Shafiq, D. A., Jhanjhi, N. Z., Abdullah, A., & Alzain, M. A. (2021). A load balancing algorithm for the data centres to optimize cloud computing applications. *IEEE Access*, *9*, 41731-41744.

Shannon, C. E. (1948). *A mathematical theory of communication*. The Bell System Technical Journal, 27(3), 379–423.

Singh, R., & Kapoor, P. (2020). *Building User Trust in Cloud-Based Services: Security Strategies and Frameworks.* Journal of Cybersecurity and Data Protection, 5(2), 94-110.

Sunyaev, A., (2020). Cloud computing. *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*, 195-236.

Thakur, D., Gupta, R., & Jain, A. (2023). Scalability and reliability in cloud computing. *Cloud Computing Advances*, 12(4), 234-245. Link to journal

Trivedi, D., & Patel, S. (2024). Homomorphic Encryption: A Survey on Methods and Applications in Secure Data Processing. Journal of Information Security and Applications, 66, 104–120.

Varghese, B., & Buyya, R. (2021). Next generation cloud computing: New trends and research directions. Journal of Cloud Computing, 10(1), 1–20.

Venu, S., Reddy, P. V. G. D., & Kumar, K. S. (2022). Multi-party Computation Enables Secure Polynomial Control Based on Secret Sharing. arXiv preprint arXiv:2103.16335.

Wang, Y., Zhao, T., & Lin, J. (2021). *A Comparative Study on Cloud Data Security Solutions: Homomorphic Encryption and User-Centric Models.* International Journal of Secure Computing, 13(2), 134-149.

Yang, P., Xiong, N. N., & Ren, J. (2020). Data Security and Privacy Protection for Cloud Storage: A Survey. IEEE Access, 8, 131723–131740.

Yao, A. C. (1982). *Protocols for secure computations*. In Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS '82) (pp. 160–164).

Yee, C. K., & Zolkipli, M. F. (2021). Review on confidentiality, integrity and availability in information security. *Journal of Information and Communication Technology in Education*, *8*(2), 34-42.

Zhou, Y., Chen, T., & Liu, H. (2023). Cloud data breach incidents: Trends and mitigation strategies. *International Journal of Cyber Security and Digital Forensics*, 12(2), 100-114.

Ziv, J., & Lempel, A. (1977). *A universal algorithm for sequential data compression*. IEEE Transactions on Information Theory, 23(3), 337–343.