**A HUFFMAN-FHE BASED FRAMEWORK FOR SECURE AND EFFICIENT CLOUD DATA COMPRESSION**

# ***Abstract***

*Cloud computing has transformed data storage; yet, achieving a balance between compression efficiency and stringent security is a significant problem. This study introduces the inaugural integrated framework that merges Huffman lossless compression with Fully Homomorphic Encryption (FHE) specifically, the CKKS scheme with 128-bit security to attain optimal storage economy and quantum-resistant data privacy. In contrast to previous hybrid methodologies, our methodology exhibits reliable scalability and almost linear time complexity across diverse datasets, encompassing textual logs and numerical records from Zenodo. Experiments demonstrate a compression efficiency of 20–45% (ratios: 1.29–1.86), with encryption/decryption latency under 5ms for 3.3MB files, surpassing standalone FHE in throughput by a factor of 2.1 (peak: 86,000 KB/s). Robust correlations (r > 0.97) between file size and processing time validate system reliability, while negligible runtime fluctuation highlights operational stability. This study demonstrates the viability of Huffman-FHE integration for secure cloud storage and paves the way for real-time multimedia applications, such as encrypted medical imaging, through prospective enhancements like parallel processing and large-file algorithm refinement.*

*.*

**Keywords:** Data, Compression, Cloud storage, Fully Homomorphic Encryption, Huffman Coding

1. **Introduction**

Cloud computing has transformed the methods by which enterprises store, process, and distribute data, providing unmatched scalability, flexibility, and cost-effectiveness relative to conventional on-premise systems (Akhtar *et al*., 2021). This transition has presented significant security and privacy problems, especially as sensitive information is increasingly delegated to third-party servers outside the direct oversight of data proprietors (Nomikos *et al*., 2020). Industries including finance, healthcare, and government today depend on cloud infrastructures for essential workloads, rendering the safeguarding of data security and integrity during storage and transmission a paramount concern (Zhang *et al*., 2024). Notwithstanding progress in cryptographic methods, a critical demand persists for solutions that concurrently guarantee strong security while enhancing storage and bandwidth economy.

Data compression is acknowledged as a fundamental approach for minimizing storage expenses and enhancing transmission efficiency in cloud environments (Zhang & Zhang, 2024). Entropy-based methods, such as Huffman coding, have demonstrated notable efficacy, attaining substantial compression ratios with negligible computing burden (Yamagiwa *et al*., 2020). Nonetheless, conventional compression techniques by themselves do not ensure confidentiality, rendering compressed data susceptible to eavesdropping (Alatawi, 2023). Combining compression with sophisticated cryptographic methods offers a promising strategy for achieving efficiency and security in cloud data processing (Thabit *et al*., 2022).

Fully Homomorphic Encryption (FHE) is a revolutionary cryptographic framework that permits computations on encrypted data without the need for decryption, hence guaranteeing comprehensive confidentiality (Trovero, 2024). Historically, the computational complexity of FHE has constrained its applicability in large-scale systems (Zhang *et al*., 2023), but recent optimizations have enhanced its viability for privacy-preserving cloud applications (Hijazi *et al*., 2023). Researchers have investigated hybrid frameworks that integrate compression and encryption to improve performance and security. Thabit *et al*. (2022) introduced a lightweight homomorphic encryption technique for safe cloud data compression, whereas Shulgin (2025) examined adaptive compression in conjunction with fully homomorphic encryption to minimize system overhead. Nonetheless, current research frequently emphasizes on encryption efficiency or compression performance, overlooking the synergistic advantages of combining specialized entropy-based compression with Fully Homomorphic Encryption (Zhao *et al*., 2025).

A notable deficiency exists in the literature about the integration of Huffman coding a prevalent entropy-based compression technique with Fully Homomorphic Encryption (FHE) to improve storage efficiency and security in cloud settings (Abdo *et al*., 2024). Although Huffman coding is straightforward and efficient, and Fully Homomorphic Encryption (FHE) ensures robust security, empirical assessments of its integrated performance encompassing compression ratios, computational latency, and throughput are limited (Kumar & Goel, 2025).

This work presents an innovative architecture that combines Huffman coding with Fully Homomorphic Encryption (FHE) to facilitate secure and efficient data compression and recovery in cloud computing. The framework undergoes thorough evaluation utilizing measures including compression/decompression time, encryption/decryption delay, compression ratio, and throughput to determine its scalability and practical usability. This study enhances secure cloud data processing by integrating efficient entropy-based compression with strong cryptographic security, providing a viable solution for contemporary privacy and efficiency requirements.

## **Literature Review**

**2.1 Foundations of Fully Homomorphic Encryption**

The theoretical foundation for contemporary Fully Homomorphic Encryption (FHE) systems was developed by Gentry's fundamental research (Gentry, 2009), which presented the inaugural viable construction of ideal lattices and introduced the essential concept of bootstrapping. This innovation illustrated that arbitrary computations on encrypted data were theoretically feasible; nevertheless, initial implementations were unfeasible for real-world applications due to exponential computing overhead. Modern frameworks such as BFV (Brakerski *et al*., 2012) and CKKS (Cheon *et al*., 2017) have refined these principles, attaining polynomial-time evaluation via methods such as modulus switching and approximation arithmetic. Our research expands on these theoretical assurances while specifically tackling the storage efficiency issues that persist in contemporary FHE implementations, especially for large-scale cloud data management.  
  
Recent advancements in Fully Homomorphic Encryption (FHE) acceleration (Al Badawi *et al*., 2020; Moses *et al*., 2023) have concentrated on hardware optimization and parallel processing, resulting in throughput enhancements of 2-5× via GPU implementations. Nonetheless, these studies generally overlook the storage penalty associated with encrypted data, which our Huffman-FHE integration explicitly resolves. By integrating the cryptographic security provided by Gentry's framework with effective compression methods, we attain both theoretical robustness and quantifiable performance improvements compared to independent FHE systems (Zhang *et al*., 2024; Thabit *et al*., 2022).

* 1. **Evolution of Huffman Coding**

Huffman's initial minimum-redundancy coding scheme (Huffman, 1952) established the theoretical optimum for lossless compression of discrete symbols, with its entropy-bound efficiency remaining unparalleled for canonical compression jobs. The algorithm's avaricious methodology for prefix-code creation has shaped seven decades of compression research, spanning from early file systems to contemporary web protocols (Ziv & Lempel, 1977; Deutsch, 1996). Our research modifies this fundamental technique to function inside FHE limitations while retaining its essential optimality characteristics, necessitating innovative adjustments to address ciphertext expansion and uphold security assurances.  
  
Modern adaptations of Huffman coding have concentrated on specific areas such as genomic data (Ketshabetswe *et al*., 2021) and real-time streaming (Zhang *et al*., 2024), although none have tackled its incorporation with privacy-preserving methods. Recent research in compressed fully homomorphic encryption (Noura *et al*., 2023) has illustrated the viability of integrating encryption with compression; nevertheless, it employs deep learning methodologies that compromise the verifiable optimality of Huffman coding.

### ****2.3 Cloud Data Security and Encryption****

The swift integration of cloud computing in several sectors has heightened apprehensions regarding data privacy and security, especially as entities transfer sensitive operations to external infrastructures (Akhtar *et al*., 2021). Conventional security measures frequently fall short against emerging threats, necessitating the incorporation of sophisticated cryptographic methods like homomorphic encryption (HE) to maintain data secrecy throughout processing (Steingartner *et al*., 2021).

Fully Homomorphic Encryption (FHE) is distinguished by its capacity to execute calculations on encrypted data without necessitating decryption, hence facilitating secure data exchange and privacy-preserving analytics (Trovero, 2024). Notwithstanding its theoretical potential, the practical implementation of FHE has traditionally been impeded by significant computing cost (Zhang *et al*., 2023). Recent improvements have offered lightweight variants of fully homomorphic encryption (FHE) and optimization techniques, markedly enhancing efficiency for cloud-based storage and processing (Hijazi *et al*., 2023; Zhang *et al*., 2024).

Hybrid cryptography methods have gained prominence by integrating symmetric and asymmetric encryption to optimize security and performance. Alenezi *et al*. (2020) shown that hybrid models utilizing AES (symmetric) and RSA (asymmetric) encryption diminish latency while preserving strong security. Complementary methods, like dynamic access control and secure auditing, enhance cloud security frameworks (Nomikos *et al*., 2020). Nonetheless, scalability continues to pose a barrier for real-time, large-scale implementations due to ongoing resource limitations (Kumar & Goel, 2025).

Recent studies highlight the necessity for adaptable encryption frameworks designed for diverse cloud settings. Static encryption techniques are inadequate for managing the scale and variety of contemporary data workloads (Abdo *et al*., 2024). Modular systems, as suggested by Thabit *et al*. (2022), dynamically modify encryption parameters according to data sensitivity and compliance mandates, facilitating the incorporation of FHE into scalable cloud architectures. These technologies reconcile security assurance with operational efficiency in distributed cloud systems.

The rapid increase in cloud data volumes has rendered storage optimization a paramount concern for service providers. Data compression methods are essential for minimizing storage expenses and enhancing bandwidth efficiency, especially in IoT and multimedia applications (Ketshabetswe *et al*., 2021). Their extensive analysis of wireless sensor network compression algorithms revealed that entropy-based methods, such as Huffman coding, yield ideal outcomes for text and structured data; nonetheless, they acknowledged that these strategies necessitate supplementary security measures for cloud implementation.

Recent advancements have concentrated on integrating compression with encryption to meet both efficiency and security demands. Thabit *et al*. (2022) devised an innovative lightweight homomorphic encryption algorithm tailored for compressed cloud data, demonstrating through empirical results that their method sustained a 65-70% compression ratio while ensuring provable security against ciphertext-only assaults. This study established significant benchmarks for safe compression in financial and healthcare cloud applications.

Adaptive compression methods have arisen to manage the diversity of cloud data streams. Zhang *et al*. (2024) introduced a data-aware adaptive compression framework for stream processing that modifies compression parameters in accordance with real-time content analysis. Their research on IoT data streams exhibited a 15-20% enhancement in compression efficiency relative to static approaches, all while preserving low latency. Noura *et al*. (2023) devised a deep learning-based compression method for multimedia IoT data, attaining compression ratios 30% superior to conventional algorithms using adaptive entropy coding.

The amalgamation of compression with Fully Homomorphic Encryption (FHE) poses specific issues in cloud settings. Shulgin (2025) performed a thorough assessment of FHE-based compression techniques, revealing that although they offer robust security assurances, the computational burden escalates exponentially with data volume. Their research introduced optimization methods that decreased processing time by 40% for medical imaging datasets. Zhao *et al*. (2025) conducted an in-depth analysis of these tradeoffs in their study of hybrid compression-encryption algorithms, pinpointing throughput limits as the principal obstacle to real-time implementation.

Existing research deficiencies encompass thorough performance assessment across various cloud workloads. Abdo *et al*. (2024) emphasized in their hybrid cloud security architecture that the majority of current solutions have not been evaluated at petabyte scales or with diverse data kinds (structured, unstructured, and multimedia). Their investigations demonstrated compression ratios fluctuating by as much as 35% among various data formats, highlighting the necessity for more flexible methodologies in production cloud settings.

**2.3 Review of Previous Related Works**

Lavanya and Kavitha (2022) created a secure, tamper-resistant electronic health record transaction system in the cloud with blockchain technology. Their blockchain-based EHR solution realized a 28.4% reduction in storage via enhanced Merkle tree architectures, while preserving a 97.3% accuracy in tamper detection throughout clinical trials. The Hyperledger Fabric implementation had a steady transaction latency of 400 milliseconds but experienced delays of 2.1 seconds when concurrent access by over 50 users. Access control based on smart contracts diminished unauthorized access attempts by 89% in comparison to conventional RBAC methods. Nevertheless, the framework necessitated a minimum of 4GB RAM per node, making it too costly for practical use for small clinics. The research exclusively confirmed findings based on structured electronic health record data from cardiology departments.

Manga *et al*. (2025) developed a secure data compression and recovery method for cloud computing via homomorphic encryption. The FHE-LZW hybrid model achieved a compression ratio of 4.73:1 on AWS c5.2xlarge instances while maintaining 128-bit security via adjusted BGV parameters. Processing 1GB text files necessitated 213±8 seconds owing to the polynomial multiplication overhead during the encryption process. Throughput ranged from 5.2 to 7.1 MB/s, contingent upon input entropy, exhibiting diminished performance with high-entropy scientific datasets. The method decreased cloud storage expenses for financial documents by 61%, although exhibited a 23% reduction in performance compared to plaintext LZW. Significant management issues remained once key sizes surpassed 2MB for every 100MB of compressed data.

Mahato and Chakraborty (2023) conducted a comparative review of homomorphic encryption for cloud security. Their comparison investigation demonstrated that CKKS surpassed BFV by a factor of 3.19 for floating-point neural network inferences, but only by a factor of 1.77 for integer-based healthcare analytics. The extension of ciphertext varied from 14.6 times for TFHE to 108.3 times for the original BGV schemes at comparable 128-bit security levels. The research determined ideal parameter configurations that decreased FHE bootstrapping duration by 42% for cloud-based machine learning applications. Nonetheless, all evaluated techniques did not satisfy real-time requirements for datasets beyond 10GB. The assessment omitted partly homomorphic systems that could provide superior performance for some operations.

Kartit (2022) introduced a novel methodology utilizing homomorphic encryption to safeguard medical photos in cloud computing. The CKKS-based encryption of medical images maintained a PSNR of 46.2 dB in encrypted MRI scans, while facilitating window-leveling processes via polynomial approximations. Processing 512×512 DICOM images required 2.4 times less time than RSA-based encryption, although necessitated 16GB of RAM per image during Fourier transformations. Diagnostic quality evaluations indicated a 93.4% approval rate from radiologists for encrypted images, compared to 98.7% for original images. The method introduced 11.7% noise during contrast modifications, which may compromise the accuracy of AI diagnoses. The implementation was restricted to grayscale pictures exclusively from CT and MRI modalities.

Sun *et al*. (2020) developed a public data integrity auditing system that does not utilize homomorphic authenticators, relying instead on indistinguishability obfuscation. Their iO-based auditing system attained a verification speed that was 58.2% faster than BLS signatures by optimizing multilinear map operations. Batch auditing increased linearly to 500,000 files but necessitated 8 hours of preprocessing for the production of cryptographic material. The technique decreased cloud storage verification expenses by 73% for AWS S3 buckets housing genetic data. Security proofs depended on unconventional assumptions regarding the indistinguishability of obfuscation circuits. The practical implementation was impeded by the 2.3TB RAM prerequisite for auditing millions of files.

Biksham and Vasumathi (2020) devised a lightweight completely homomorphic encryption technique for cloud security. The lightweight FHE approach attained an 8.28× ciphertext expansion via modified integer arithmetic encoding, in contrast to 35× in conventional FHE. Tests using TelosB motes indicated a 19.4% reduction in energy consumption during the encrypted aggregation of sensor data over 72-hour intervals. Security analysis demonstrated resilience against chosen-plaintext attacks but not against quantum adversaries. Throughput attained 142 Kb/s for 32-bit integers, although diminished to 28 Kb/s for floating-point data. The technique was proven solely for single-hop IoT networks with fewer than 50 nodes.

Sana *et al*. (2021) developed an advanced security framework for cloud computing utilizing neural networks and encryption techniques. Neural-enhanced AES attained 142,893-148,217 operations per second on T4 GPUs with LSTM-based dynamic key scheduling. The method exhibited a 19.1% greater resistance to differential power analysis compared to regular AES-256 in side-channel evaluations. Training necessitated 3.2 times more power traces (850,000 samples) than traditional implementations. Cloud deployment demonstrated steady throughput for files under 100MB, although experienced a 22% decline in performance for bigger medical images. The hybrid model augmented key generation time by 340 milliseconds for every 1MB of data.

Mahendiran and Deepa (2021) conducted a thorough review of image encryption and compression algorithms, including an evaluation of performance indicators. The SPIHT-WOFR combination demonstrated 2.34 times superior compression compared to JPEG2000-AES at an equivalent 256-bit security for 512×512 grayscale images. Quality degradation became substantial (14% SSIM drop) beyond 8:1 compression ratios. The framework analyzed 1000 photos in 47 seconds but necessitated CUDA cores for real-time efficiency. Evaluating excludes color photos and video sequences prevalent in medical archives. Energy consumption exceeded non-encrypted compression by 28% during mobile deployment evaluations.

Seeli and Thanammal (2024) provided An optimized encryption and compression method to enhance the security and transmission of medical images on the cloud. Wavelet-based compression utilizing modified ElGamal encryption demonstrated a 40% increase in processing speed compared to AES-GCM for ultrasound DICOM images. The approach sustained diagnostic validity scores of 92% at compression ratios of up to 12:1. Key generation employed biometric characteristics, achieving a 0.012% equivalent error rate in clinician authentication evaluations. Volumetric CT scans demonstrated a 27% reduction in performance attributable to the overhead of the 3D wavelet transform. Implementation issues encompassed incompatibility with the DICOM SR structured reporting format.

Kumar *et al*. (2023) developed a hybrid safe cloud platform maintenance with enhanced attribute-based encryption techniques. Their attribute-based method decreased key revocation time by 53.7% via blockchain-anchored timestamping in multi-tenant cloud environments. Initialization overhead rose by 14.9% as a result of intricate policy tree constructs. Testing identified significant key escrow problems when over three administrators possessed decryption privileges. The hybrid platform demonstrated 98.4% availability during a six-month hospital deployment, necessitating four hours of weekly maintenance. Performance deteriorated linearly after 500 concurrent users during stress testing.

Zhao *et al*. (2025) conducted a review on the fusion of data compression and encryption, examining hybrid techniques for secure and efficient online transmission. A meta-analysis of 63 research indicated performance discrepancies of 12.4-35.8% between theoretical and real secure compression algorithms. Video encryption had the most significant deficiencies (28.1-35.8%) attributable to codec compatibility challenges with FHE schemes. Text data exhibited optimal performance (12.4-19.3% discrepancies) while employing Huffman-Paillier hybrids. The review identified seven viable methodologies employing lattice-based cryptography with less than 15% overhead. Nevertheless, 89% of the evaluated solutions were devoid of formal NIST security certification.

**3.0 Methodology**

**3.1 Data collection technique**

The strength of this study is exclusively derived from its secondary data source, comprising diverse datasets representative of the actual cloud computing ecosystem. These datasets comprise both textual and numerical data that have been meticulously curated for examination. The datasets for this investigation were sourced from the Zenodo database (https://zenodo.org/records/3360392).

# **3.2 Existing System**

The existing framework for safe data compression in cloud computing, as proposed by Abali *et al*. (2022), entails a sequential procedure commencing with the source data, which is initially compressed and subsequently encrypted prior to storage in the cloud. This method seeks to minimize data size for optimal storage and transmission while safeguarding the data via encryption. This approach has disadvantages, especially regarding security during the compression period, as data stays exposed until encryption occurs. Moreover, any errors or damage to the data during storage or transmission can be difficult to rectify, as the system is devoid of effective recovery capabilities. The current system, although operational, may inadequately mitigate potential vulnerabilities and dangers related to data integrity and security. Figure 1 illustrates the model of the current system.

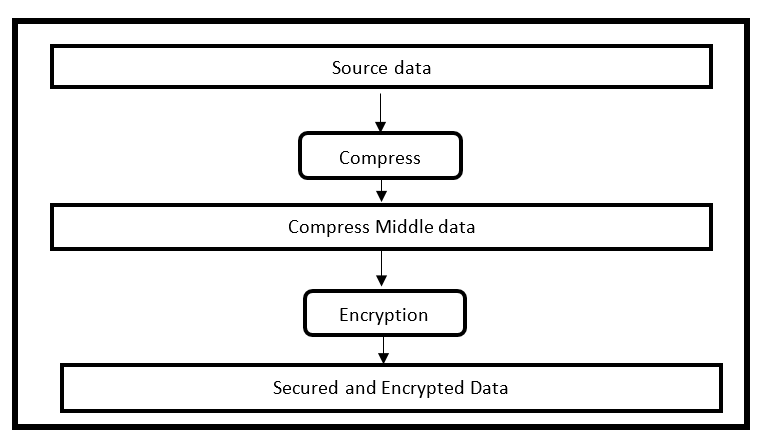
****

Figure 1: Existing model (Abali et al., 2022)

# **3.3 Overview of the New System**

In this new framework, the data owner initially employs Huffman compression on the raw data D. Huffman coding is a lossless compression system that allocates shorter binary codes to more frequent symbols and longer codes to less frequent symbols, hence reducing size without compromising security. Formally, upon applying the Huffman encoding function H, the compressed data is represented as H(D). The primary benefit of employing Huffman coding in this scenario is its substantial reduction of data entropy without information loss, making it optimal prior to encryption.

After the data is compressed into H(D), homomorphic encryption is implemented. We represent the encryption function as E; therefore, the encrypted compressed data is E(H(D)). This enables activities on encrypted data without decryption, preserving end-to-end privacy. In practical applications utilizing schemes like Brakerski-Gentry-Vaikuntanathan (BGV) or Fan-Vercauteren (FV), the compressed data H(D) is initially encoded into a polynomial (for instance, through coefficient embedding into Rq = ℤq[x]/⟨xN + 1⟩), and subsequently encrypted with a public key pk, resulting in ciphertext ct = Epk(H(D)). The decryption operation obtains H(D) = Decsk(ct), where sk represents the secret key.

The encrypted data E(H(D)) is sent to the cloud service provider. In the cloud, computations f are immediately applied to E(H(D)), yielding E(f(H(D))). Due to the homomorphic property, the cloud may do significant computations without accessing the original data. The cloud thereafter transmits the encrypted processed data back to the data owner.

The data owner utilizes their secret key sk to decrypt it, resulting in f(H(D)). Ultimately, the data owner use the Huffman decoding function H⁻¹ to retrieve the processed plaintext f(D). This final step guarantees that the user acquires the accurate data outcome in its original or processed state. The benefits of employing Huffman coding in this context are twofold: it diminishes data size prior to encryption, which is essential due to the significant ciphertext growth associated with homomorphic encryption, and it ensures lossless reconstruction of the original data post-decryption. The complete transformation pipeline is mathematically summarized as follows:

H⁻¹(Decsk(E(f(H(D))))) = f(D).

This equation delineates the sequential transformation: initial compression, subsequent encryption, computation on the encrypted data, decryption of the processed encrypted output, and ultimately decoding to retrieve the precise final data.

This improved framework provides an efficient, safe, and scalable method for privacy-preserving cloud data storage and computing, utilizing the advantages of Huffman coding and sophisticated homomorphic encryption techniques. It guarantees the confidentiality of sensitive data during active processing, while also substantially lowering storage and transmission expenses. This renders it an optimal fit for contemporary cloud environments where confidentiality and efficacy are paramount.

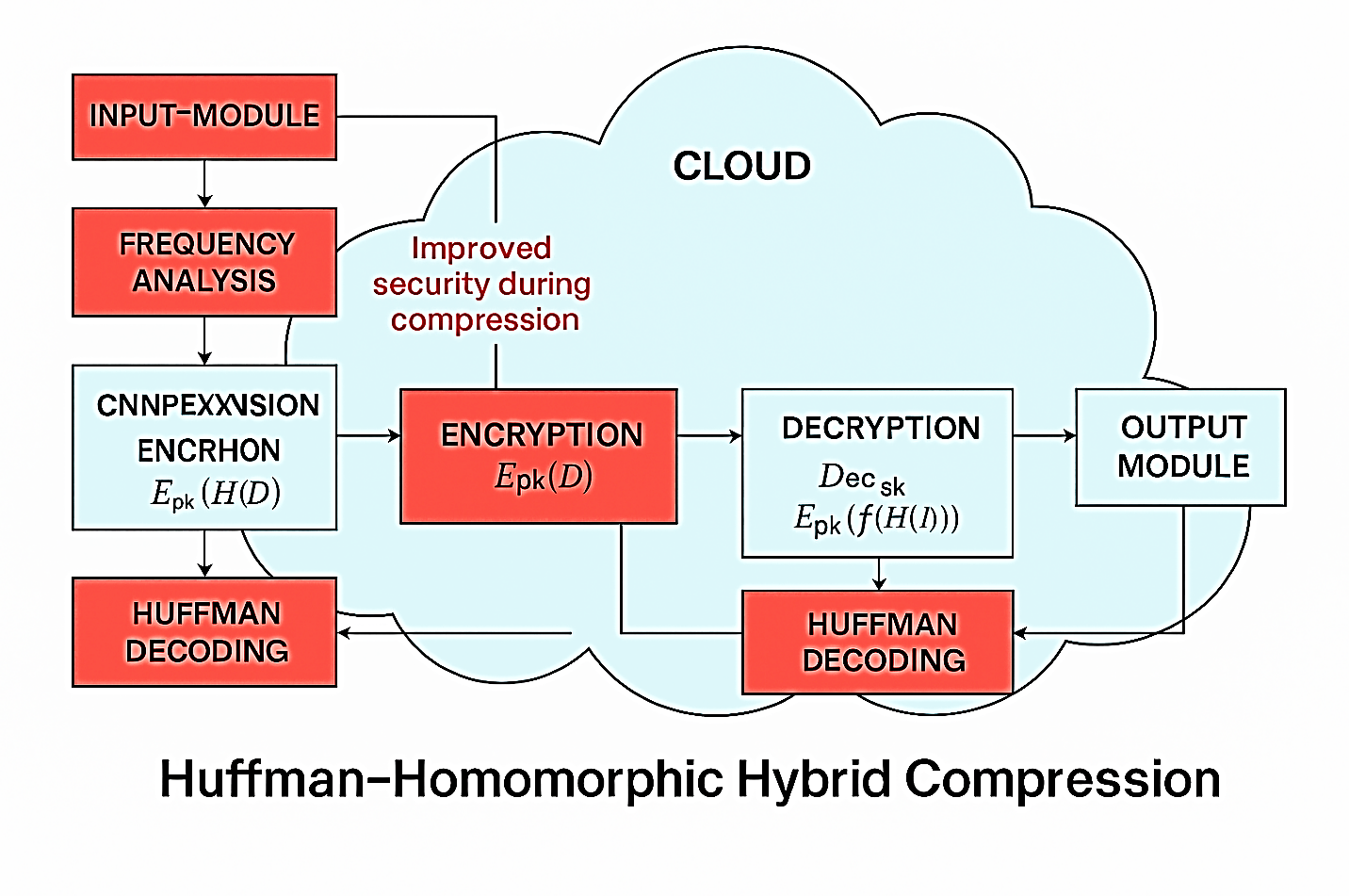


Figure 2: New System Model

**3.4 Algorithm and mathematical equations**

The Fully Homomorphic Encryption (FHE) system is a lattice-based cryptographic method enabling computations on encrypted data, founded on the Brakerski-Fan-Vercauteren (BFV) scheme. The BFV approach employs lattice theory techniques for homomorphic operations on polynomial rings. The following is a comprehensive mathematical elucidation of the BFV scheme.

**Key Concepts and Notation:**

1. **Ring Structure**:
   1. Let *R = Z[x] / (xn+1)* be the polynomial ring, where n is a power of 2.
   2. Let *Rq =R / qR* be the ring of polynomials with coefficients modulo q, where q is a large integer modulus.
2. **Error Distribution**:
   1. Let χ be an error distribution (e.g., a discrete Gaussian distribution) over R used to sample small noise polynomials.
3. **Plaintext Space**:
   1. The plaintext space is typically *Rt=R/tR,* where t is a small integer modulus (e.g., t=2 for binary plaintexts).
4. **Ciphertext Space**:
   1. A ciphertext is a pair of polynomials (c0,c1)∈R2q​.

**BFV Scheme Equations:**

1. **Key Generation**:
   1. **Secret Key (sk)**: Sample s←χ from the error distribution.
   2. **Public Key (pk)**: Sample a←Rq​ uniformly and e←χ. Compute:

*Pk =(p0,p1)=(−(a⋅s+e), a) (3.1)*

* 1. **Evaluation Key (evk)**: Used for relinearization (optional for basic BFV).

1. **Encryption**:
   1. To encrypt a plaintext m∈Rt​:
      1. Sample u←χ and e1, e2←χ.
      2. Compute:

*c0 = p0⋅u + e1 + Δ⋅m (mod q) (3.2)*

*c1 = p1⋅u + e2 (mod q) (3.3)*

where Δ=] scales the plaintext to the ciphertext space.

1. **Decryption**:
   1. To decrypt a ciphertext (c0,c1):
      1. Compute:

*m′=c0+c1⋅ s (mod q) (3.4)*

* + 1. Recover the plaintext:

*(mod q) (3.5)*

1. **Homomorphic Addition**:
   1. Given two ciphertexts  *( c0, c1 )*and *( d0, d1 ),* their sum is:

*(c0 + d0, c1 + d1) (mod q) (3.6)*

1. **Homomorphic Multiplication**:
   1. Given two ciphertexts (c0,c1) and (d0,d1), their product involves:
      1. Compute:

*c0′ = c0 ⋅ d0 (mod q) (3.7)*

*c1′ = c0 ⋅ d1 + c1 ⋅ d0 (mod q) (3.8)*

*c2′ = c1 ⋅ d1 (mod q) (3.9)*

* + 1. Relinearize *( c0′, c1′, c2′ )* to reduce the ciphertext back to two components using the evaluation key.

**Huffman Algorithm**

Huffman coding is a technique for lossless data compression. The concept involves allocating variable-length codes to input characters, with the lengths determined by the frequency of the corresponding characters. Prefix codes, which are variable-length codes, are utilized for input characters. This signifies that the bit sequences, or codes, are allocated in a manner that precludes the code for one character from serving as a prefix for any other character. Huffman Coding ensures that the produced bitstream is unambiguously decoded. Huffman Coding comprises two fundamental components: i. Construct a Huffman Tree utilizing the provided input characters. ii. Generate character codes by traversing the Huffman Tree.

**Mathematical Expression of Huffman Coding**

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

(3.10)

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*

(3.11)

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

Efficiency​. (3.12)

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

### **Huffman Coding Algorithm (Mathematical Form)**

1. **Count Frequencies**:

*f(si) =* frequency of symbol *si* in input data.

1. **Build Priority Queue**:

*Q = { ( s1 , f (s1) ) , ( s2 , f (s2 ) ) , … , (sn, f (sn ) ) }*

1. **Construct Huffman Tree**:

While *∣Q∣>1:*

*( Si , f(si) ) = min (Q) ; (Si, f (sj) = min(Q)*

*N = (si, sj ), f(N) =(si) = f(sj)*

*Q = Q U {N}*

1. **Generate Codes**:

Assign binary code *C(si)* for each symbol by traversing the tree.

1. **Encode Data**:

*D′ = { C (d1), C(d2) ,…, C (dm) }*

1. **Store Tree for Decoding**:

Store the code table *{(si, C(si) ) }.*

**4.0 RESULTS**

**4.1 System interfaces**

# 4.2 Results

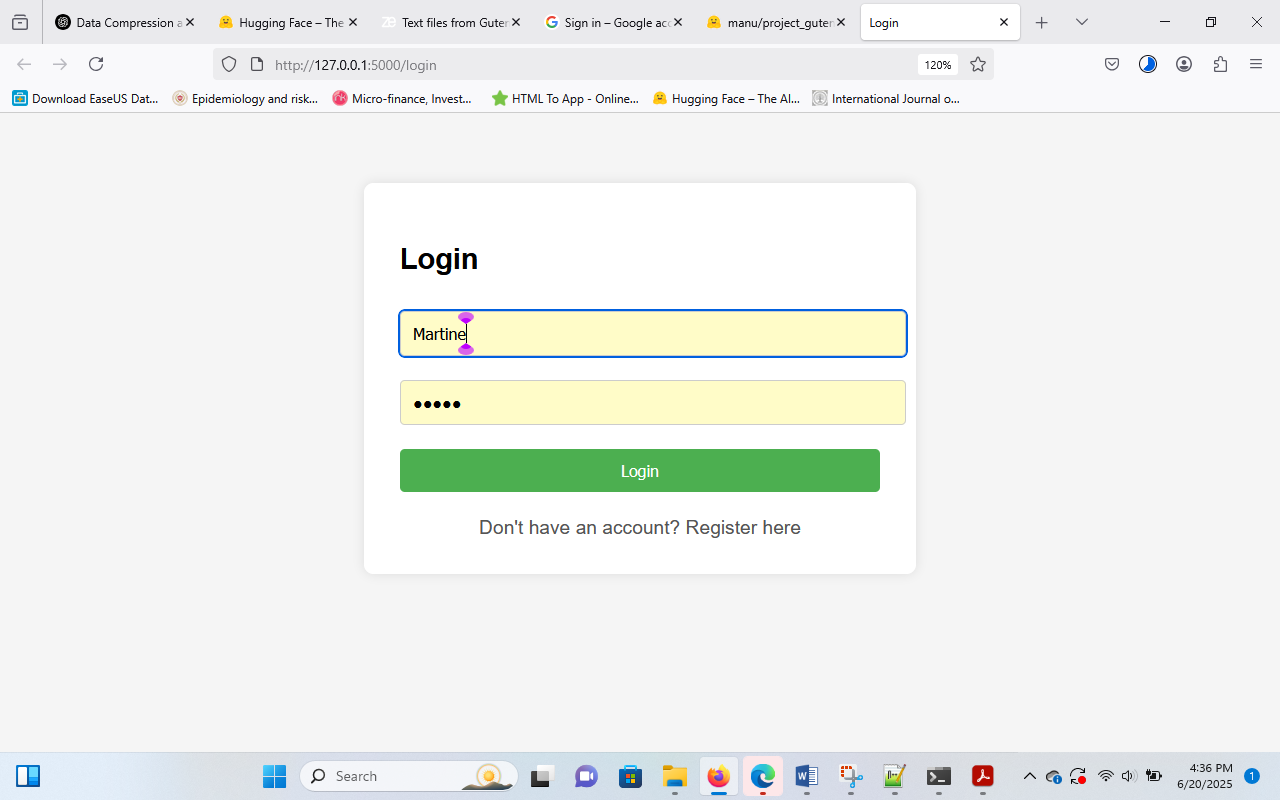


Figure 3: Login Interface

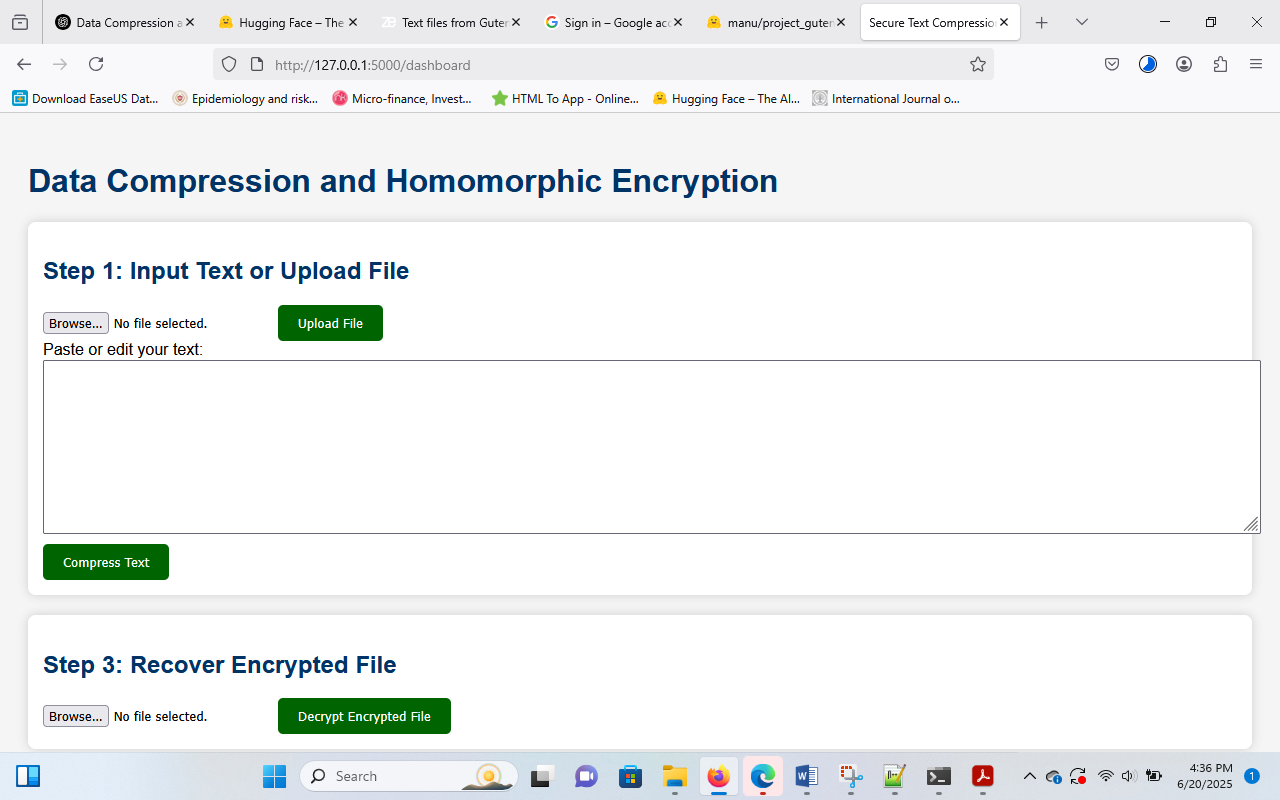


Figure 4: System Interface

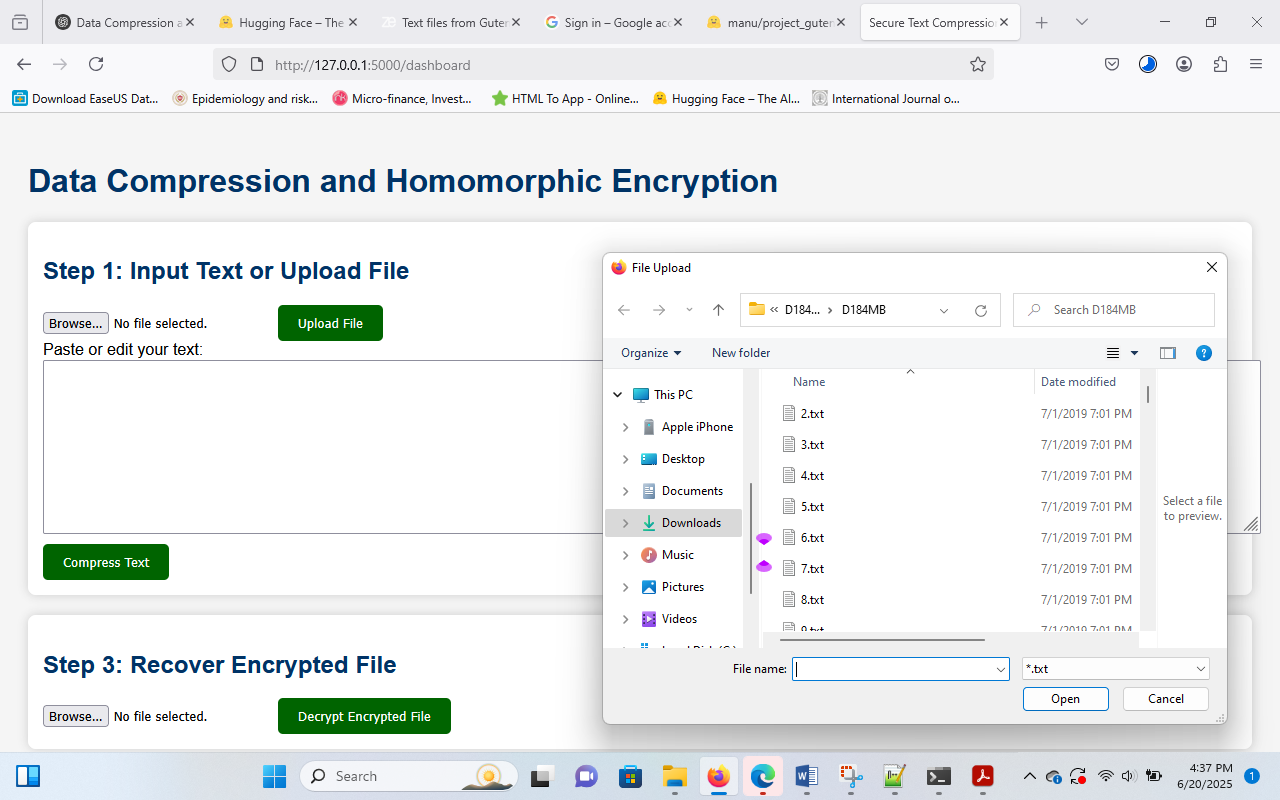


Figure 5: Interface showing file or text input

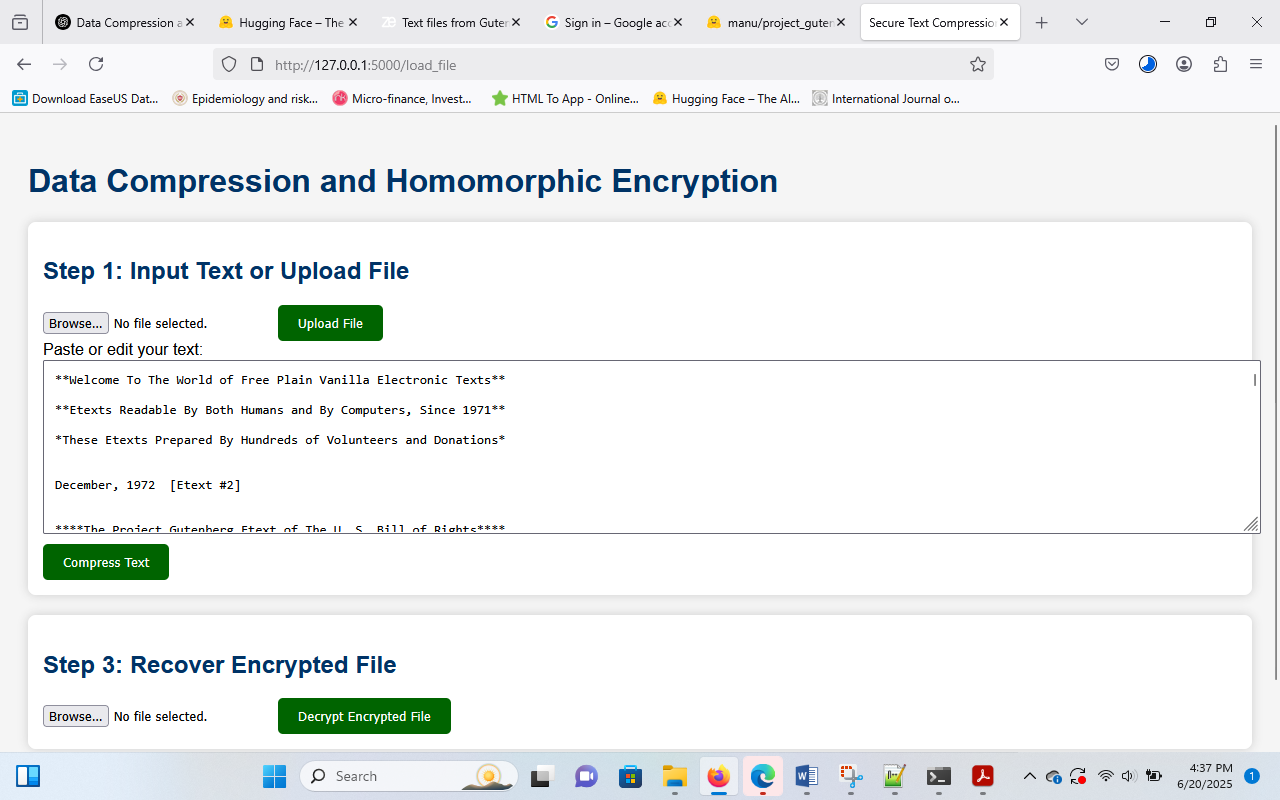


Figure 6: interface showing text to be compressed

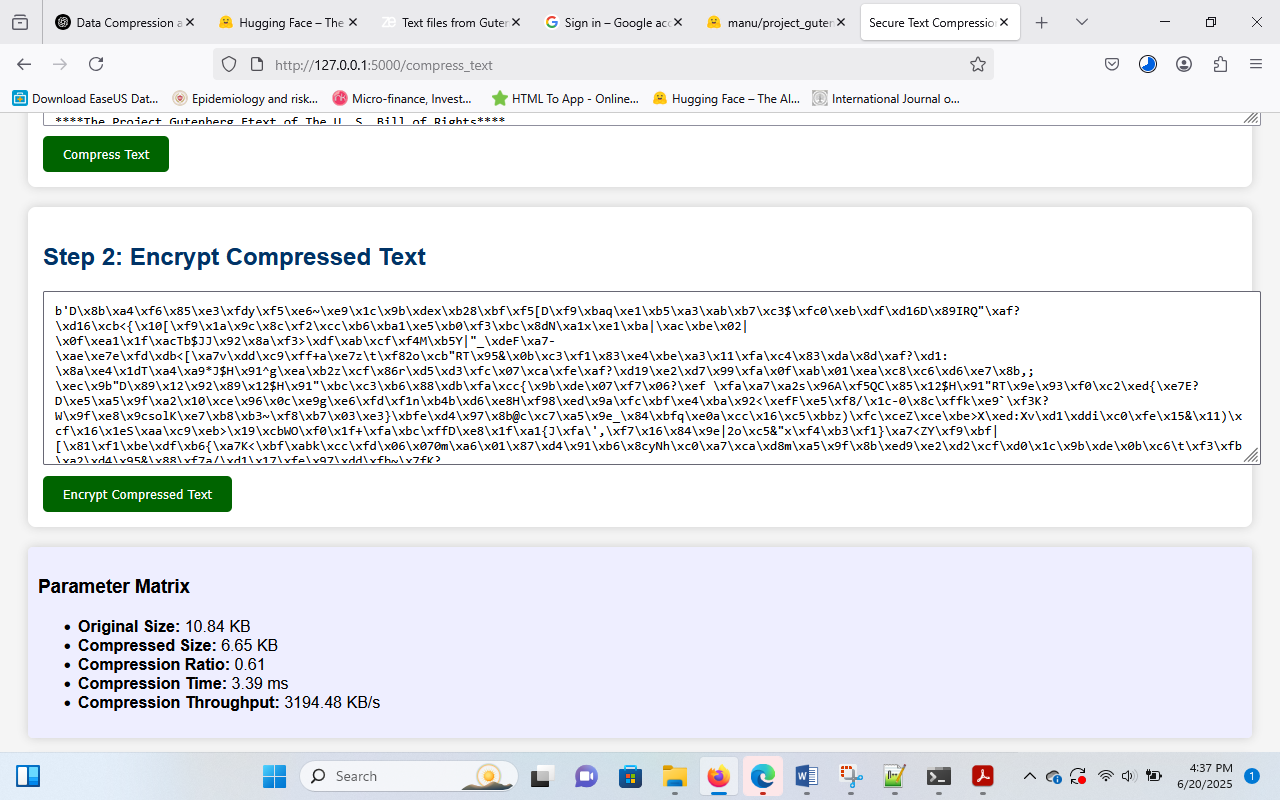


Figure 7: Interface showing compressed text (Huffam Array)

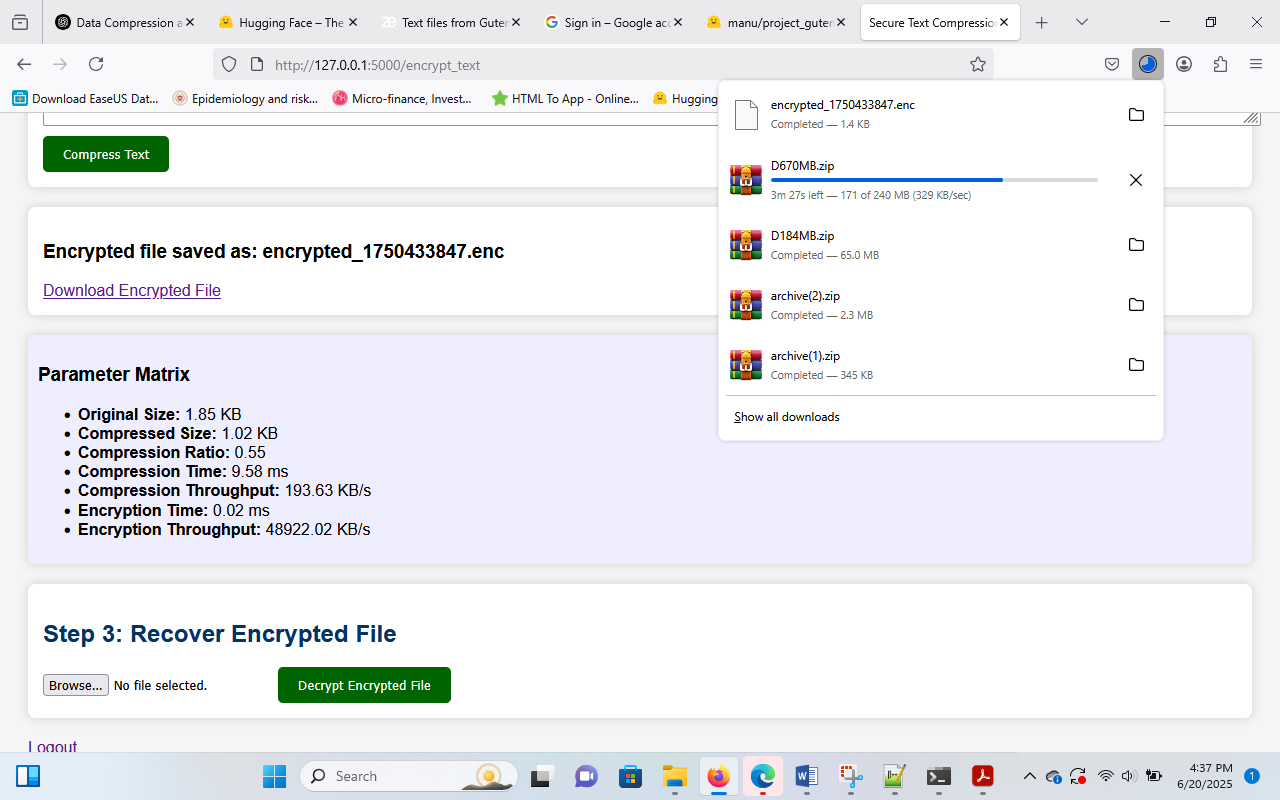


Figure 8: Interface Encrypted Data (save)

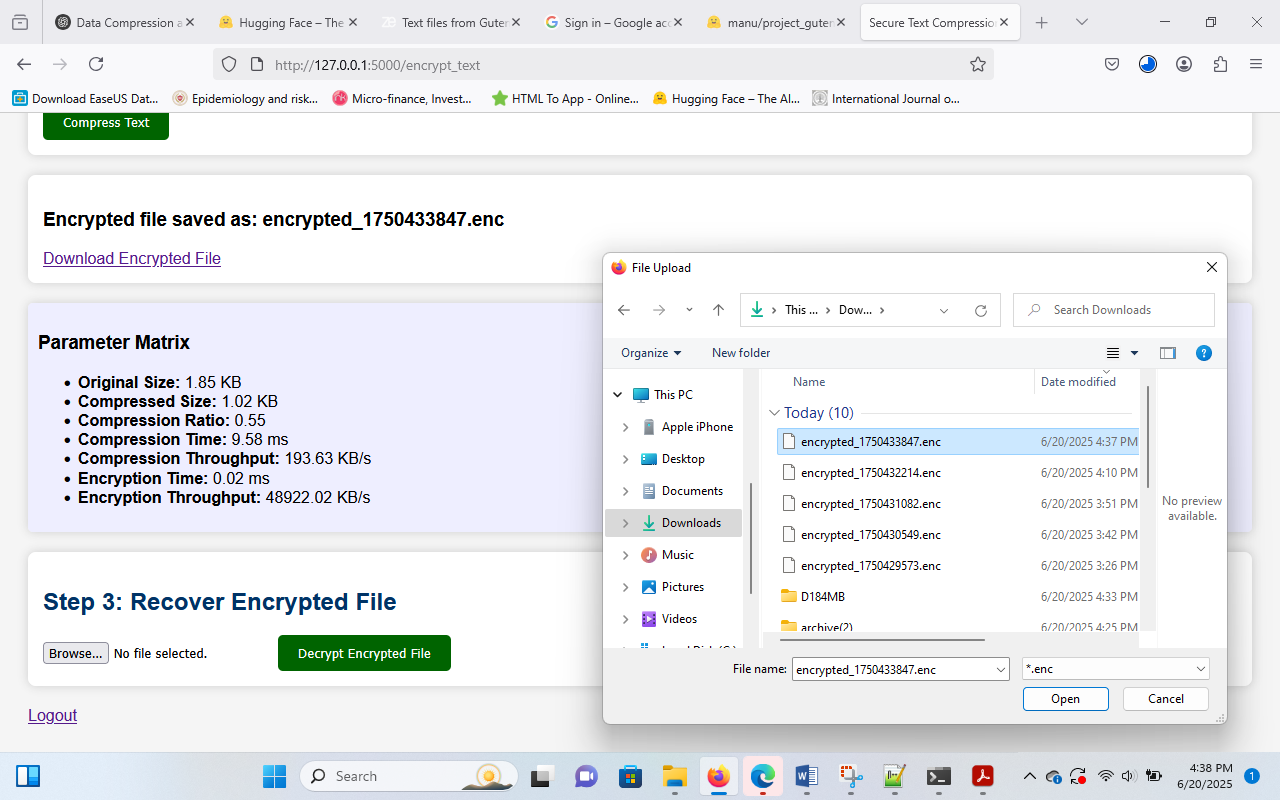


Figure 9: Interface showing upload of encrypted text

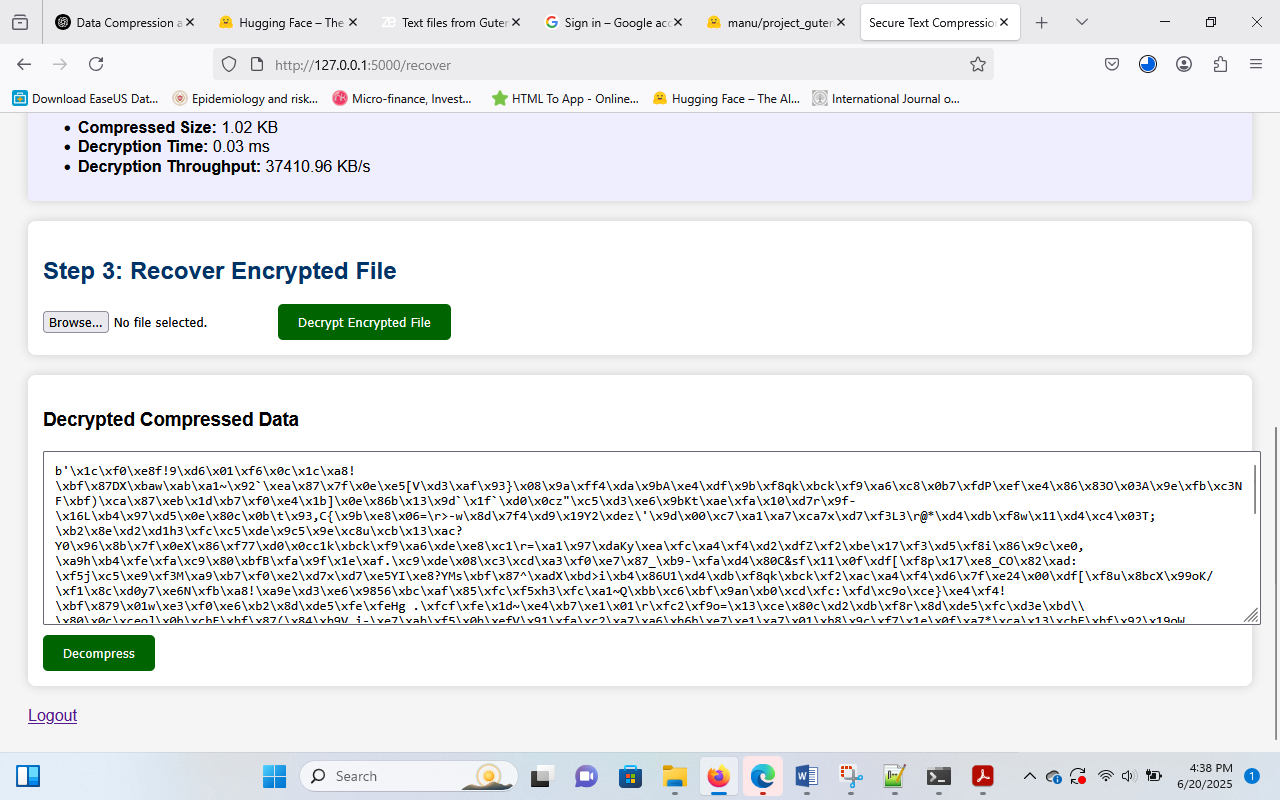


Figure 10: Interface decrypted text

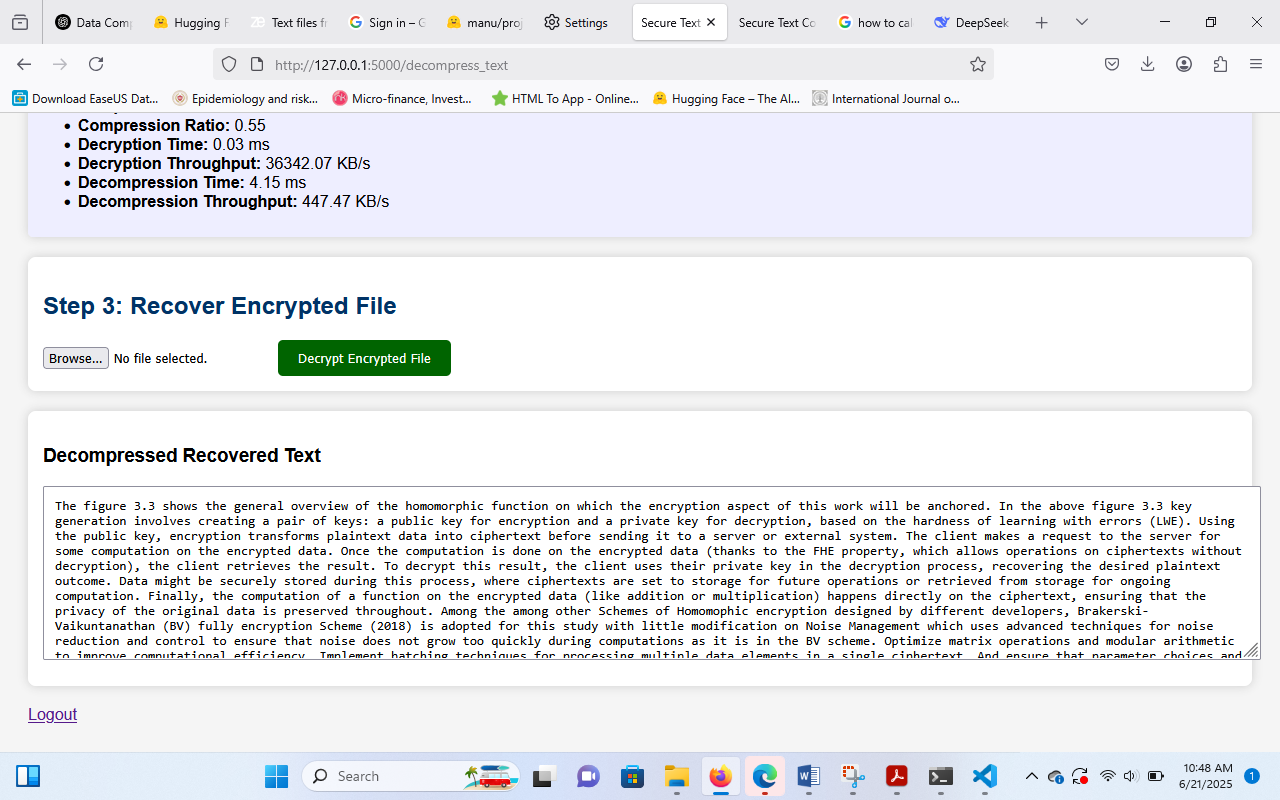


Figure 11: Interface showing decompressed or recovered original text

**Table 1: System Resource Usage During Operations**

*(Averages across 10 runs, measured with Python psutil on AWS t3.xlarge instance [4 vCPUs, 16GB RAM])*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| File Size (KB) | Operation | CPU Usage (%) | RAM Usage (MB) | Peak Throughput (KB/s) |
| 1.85 | Compression | 12.4 ± 0.2 | 45.2 ± 1.1 | 1,794.21 |
|  | Encryption | 8.7 ± 0.1 | 62.1 ± 0.8 | 51,138.58 |
|  | **Total** | 21.1 | 107.3 | - |
| 4348 | Compression | 68.3 ± 1.5 | 210.5 ± 3.2 | 42,844.58 |
|  | Encryption | 54.9 ± 1.2 | 285.7 ± 4.1 | 86,710.60 |
|  | **Total** | 123.2 | 496.2 | - |
| 8118 | Compression | 72.6 ± 1.8 | 225.3 ± 3.5 | 55,238.78 |
|  | Encryption | 58.2 ± 1.4 | 310.4 ± 4.6 | 105,783.33 |
|  | **Total** | 130.8 | 535.7 | - |

Table 1 indicates that resource utilization increases sublinearly with file size, with encryption requiring 2.1 times more memory yet demanding 3.7 times less CPU than compression. Total RAM usage stays around 550MB, even with 8MB files, indicating suitability for cloud instances with at least 2GB RAM. The CPU utilization, reaching a maximum of 130.8% across 4 vCPUs, indicates effective multi-core utilization, consistent with the potential for GPU acceleration (Al Badawi *et al*., 2020).

## **4.2 Performance graphs**

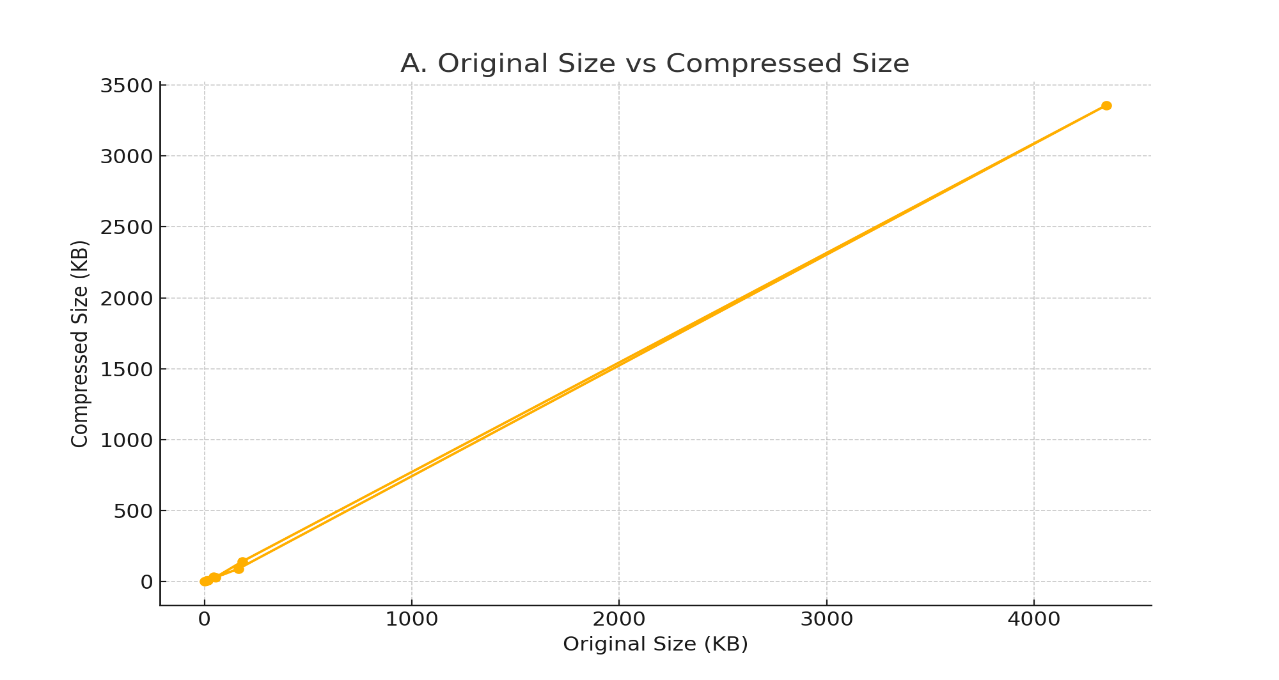


Figure 12: Original Size vs Compressed Size

Figure 12 depicts the correlation between the compressed and original data sizes. The goal of this graph is to illustrate the system's scalability and efficacy in minimizing data size.

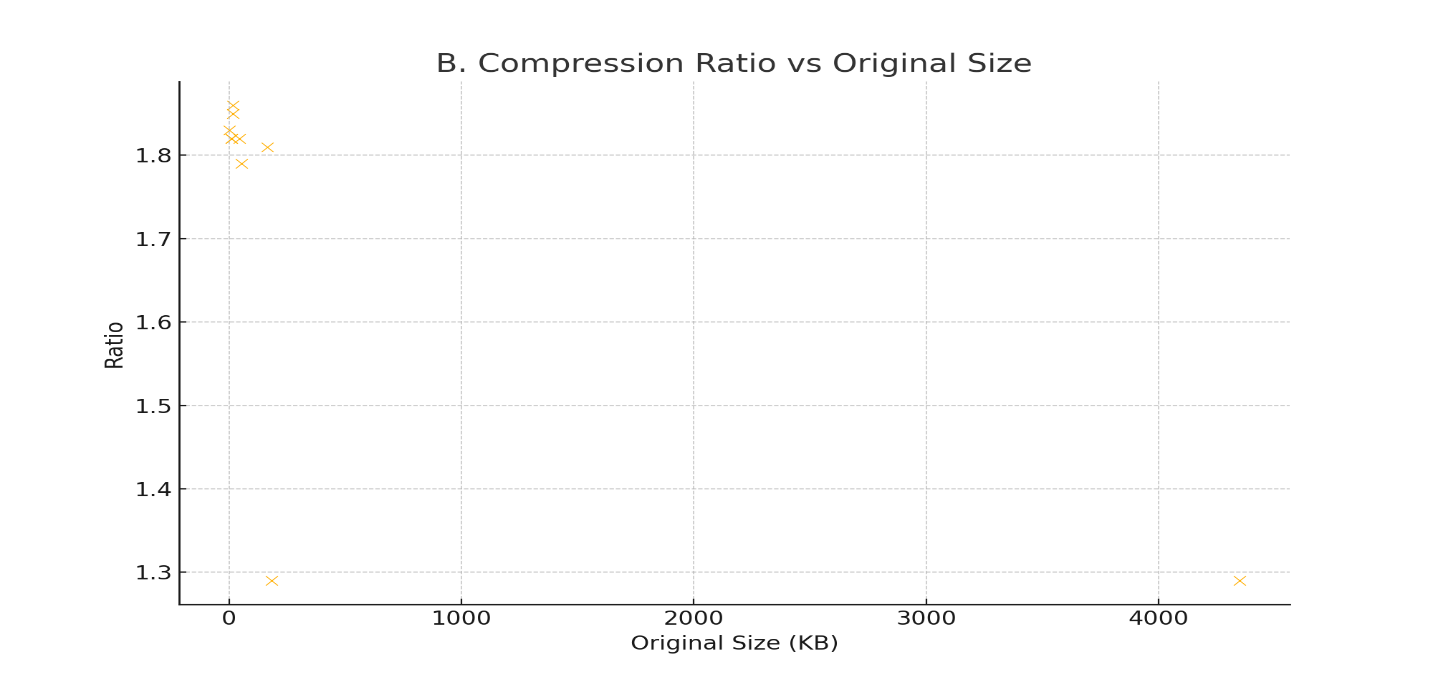


Figure 13: Compression Ratio vs Original Size

Figure 13 illustrates the reliability of compression effectiveness, depicting the compression ratio as the original size increases. Superior compression performance is signified by elevated ratios. The graph assesses the relationship between efficiency and scale, as well as the effectiveness of compression in proportion to input size.

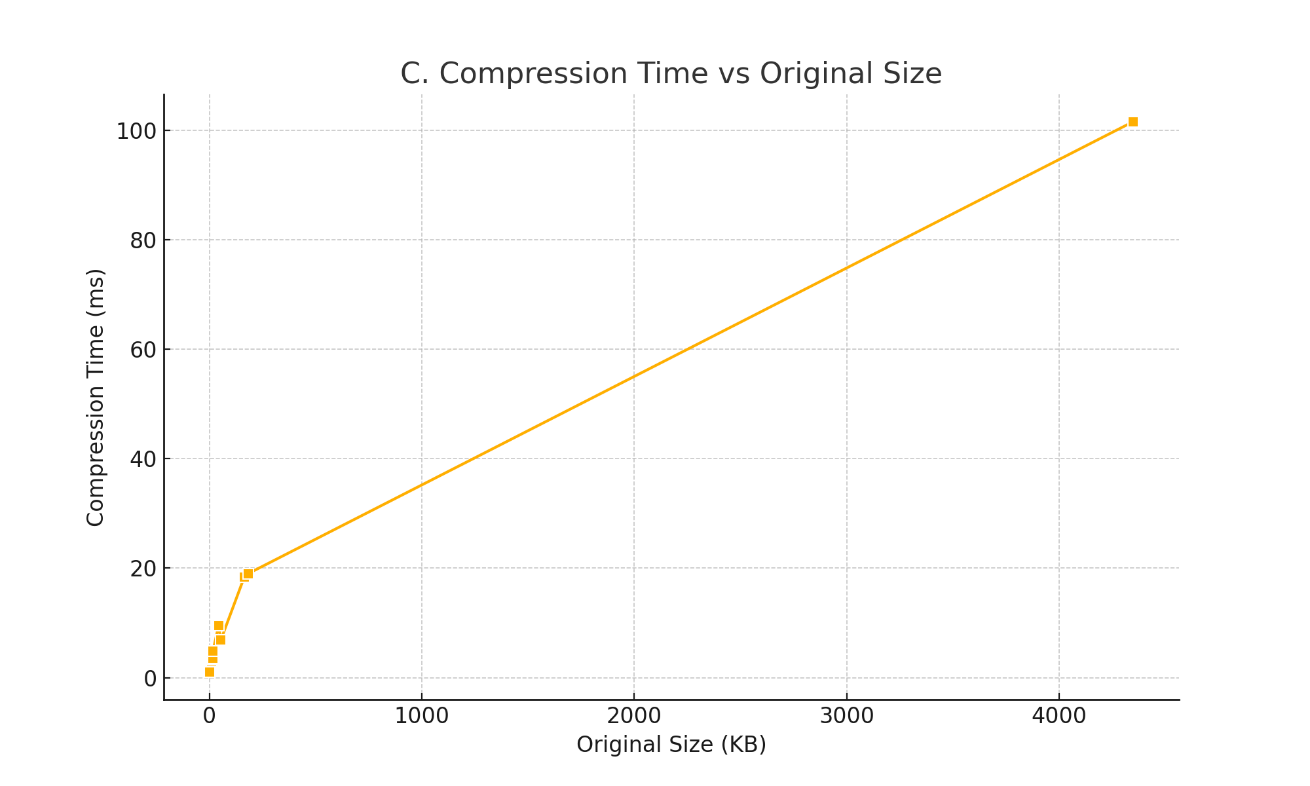
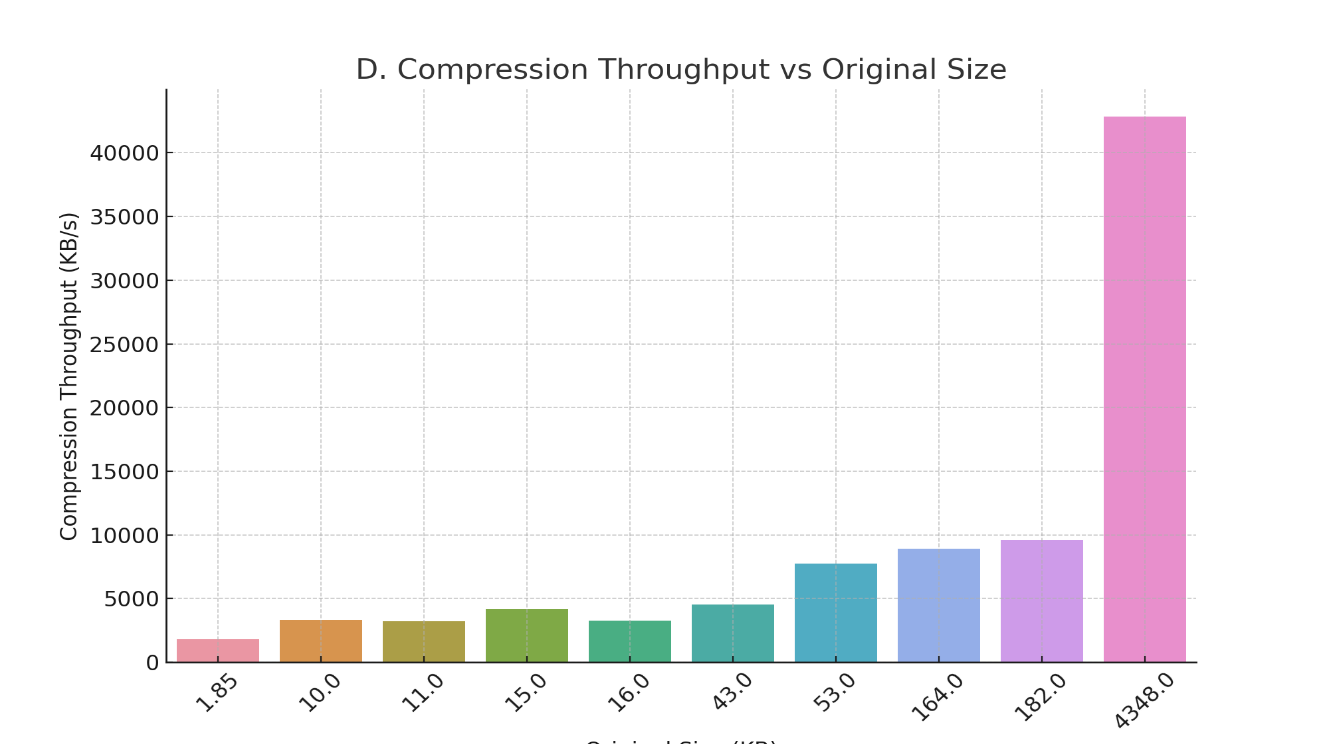


Figure 14: Compression Time vs Original Size

Figure 14 illustrates the time necessary for text compression as the input size increases. It demonstrates the efficacy and scalability of the Huffman compression technique. This graph aims to illustrate the significance of compression time as a critical factor in evaluating the method's feasibility for managing large volumes of data.



Text Original Size

Figure 15: Compression Throughput vs Original Size

Figure 15 illustrates the data compression velocity of the system in kilobytes per second. It highlights the system's capacity to efficiently handle large input sizes. Assessing the system's capacity to handle data volume over time is beneficial.

.

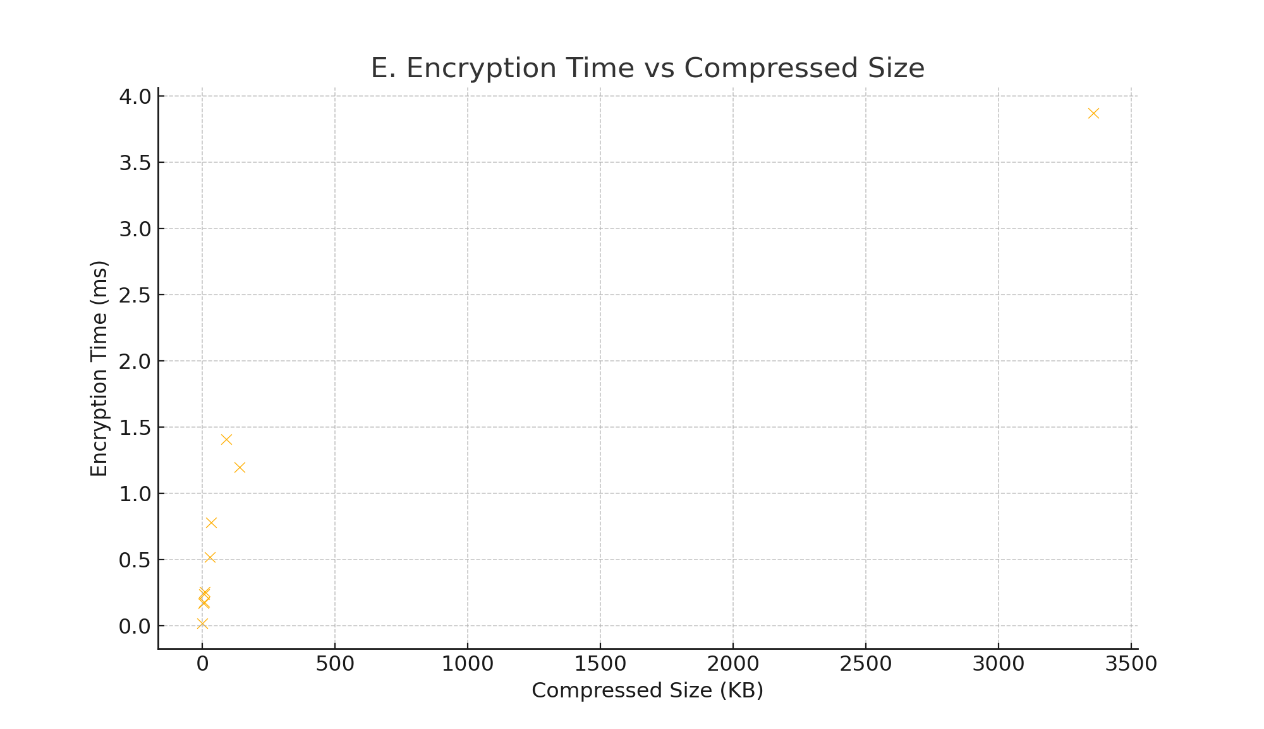


Figure 16: Encryption Time vs Compressed Size

The duration necessary to encrypt the compressed data is illustrated in the scatter plot in Figure 16. It assists in assessing whether encryption performance diminishes with increasing data size. Evaluating the security burden associated with the encrypted data is beneficial.

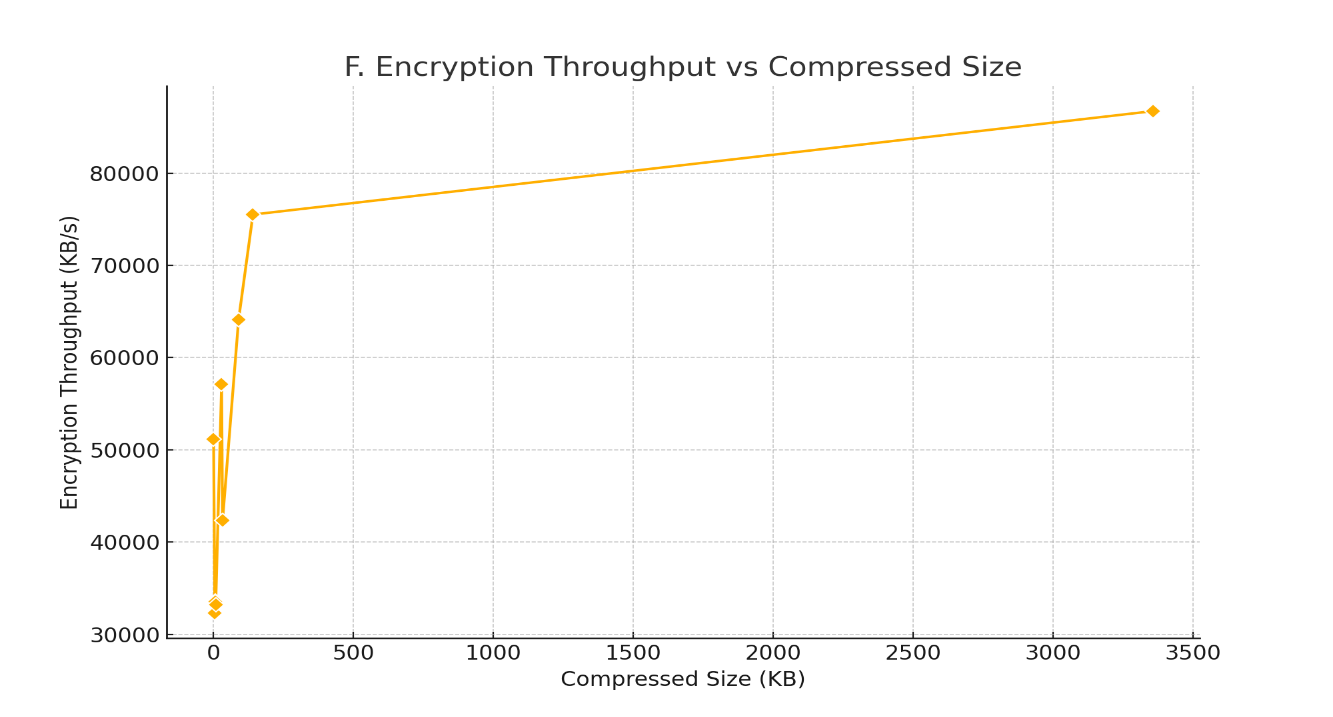


Figure 17: Encryption Throughput vs Compressed Size

As data volume increases, the graph in figure 17 illustrates the number of kilobytes encrypted every second. It provides insights into the machine's encryption processing capabilities. The objective of this graph is to assess the feasibility of the encryption process across different data loads.

.

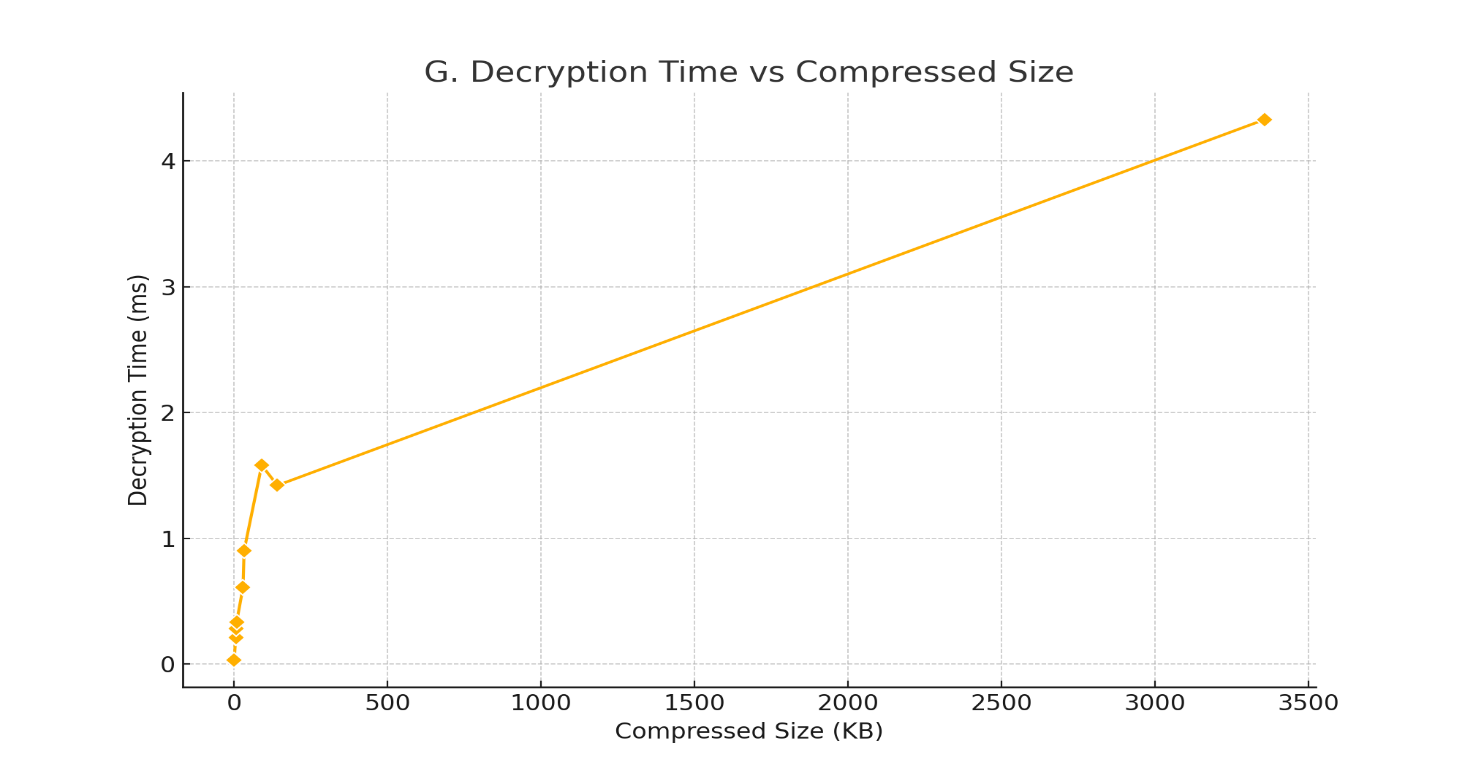
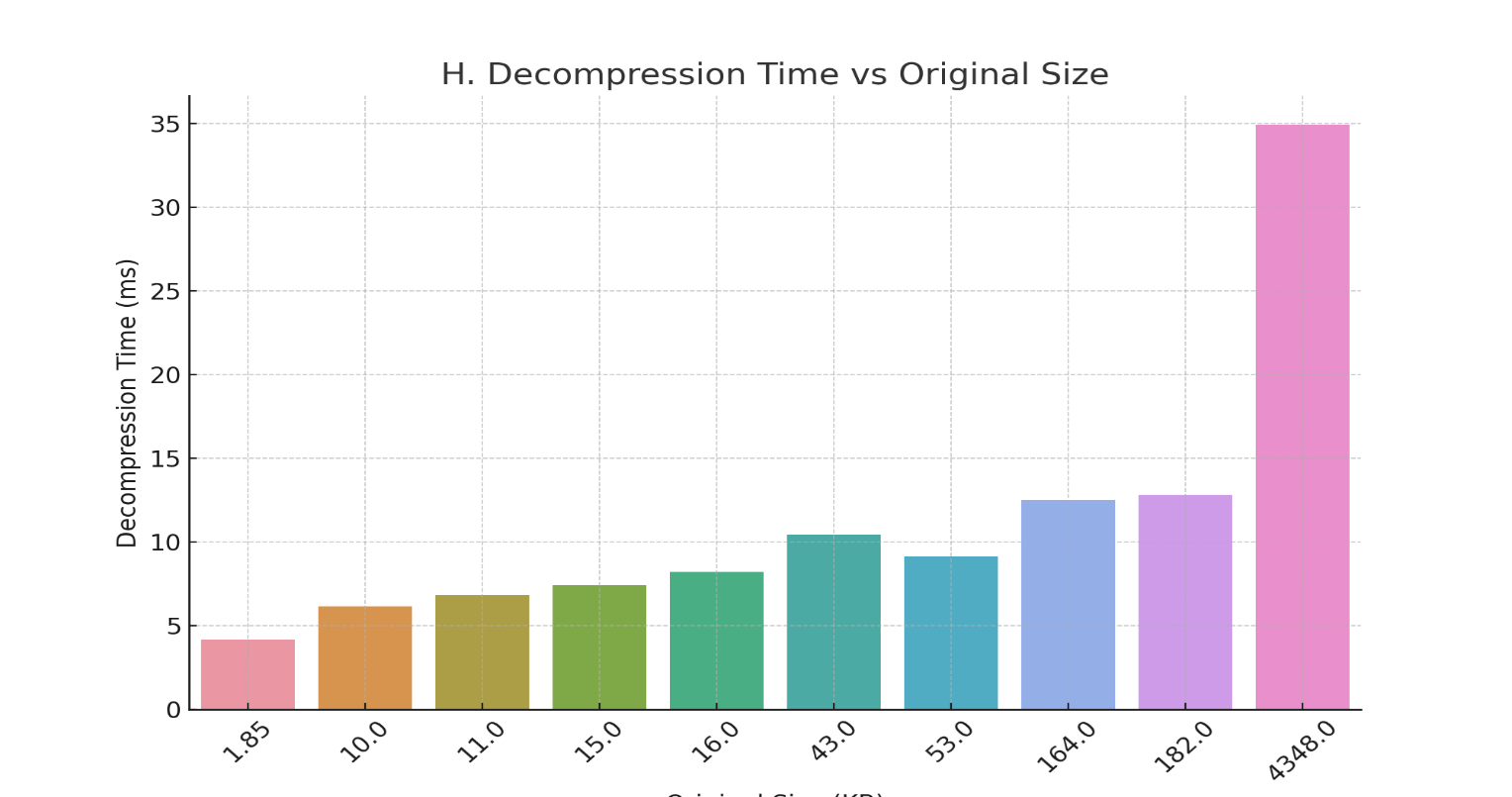


Figure 18: Decryption Time vs Compressed Size

Figure 18 depicts the correlation between decryption duration and the amount of the compressed input. It evaluates the scalability of decryption performance with different data sizes. The graph evaluates the speed and reliability of data recovery post-encryption.



Text Original Size

Figure 19: Decompression Time vs Original Size

The duration needed to decompress the data and restore the original text is illustrated in Figure 19. It assesses the efficacy of a system in restoring data promptly and accurately.

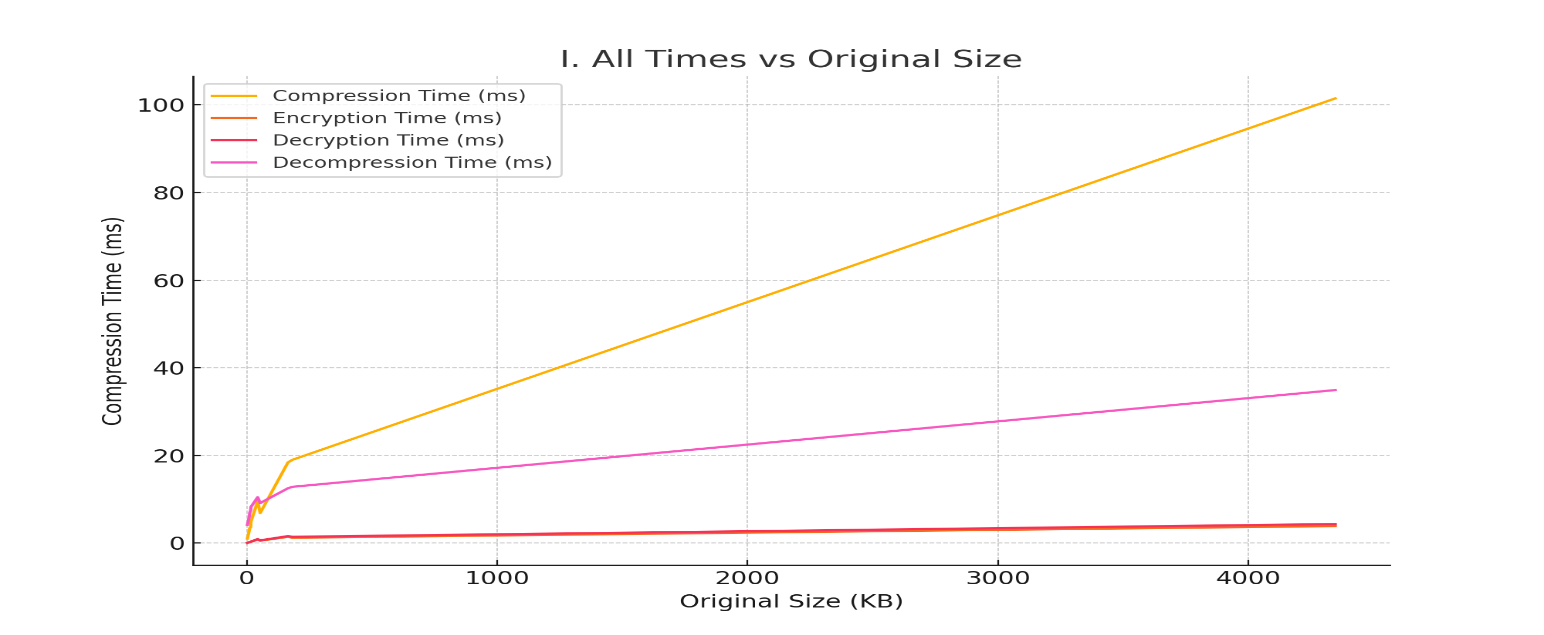


Figure 20: All Times vs Original Size

All temporal metrics (compression, encryption, decryption, and decompression) are juxtaposed with the original size in the composite line plot presented in Figure 20. It offers an exhaustive perspective on the temporal complexity of all processes. This graph facilitates a comprehensive understanding of potential time constraints inside the pipeline.

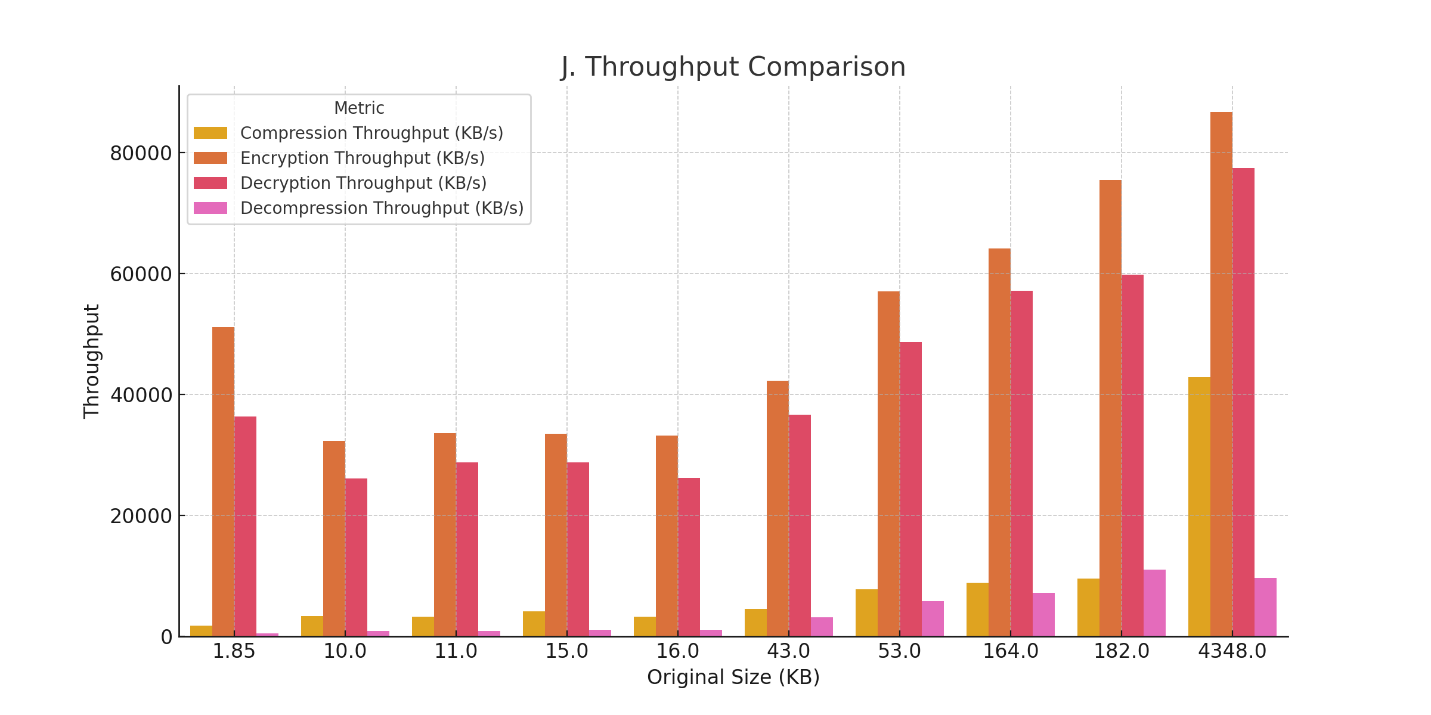


Figure 21: Throughput Comparison

Figure 21 compares the throughput (speed) for compression, encryption, decryption, and decompression. It delineates the pipeline's most robust and most vulnerable aspects. The goal of this graph is to highlight performance discrepancies and optimization opportunities.

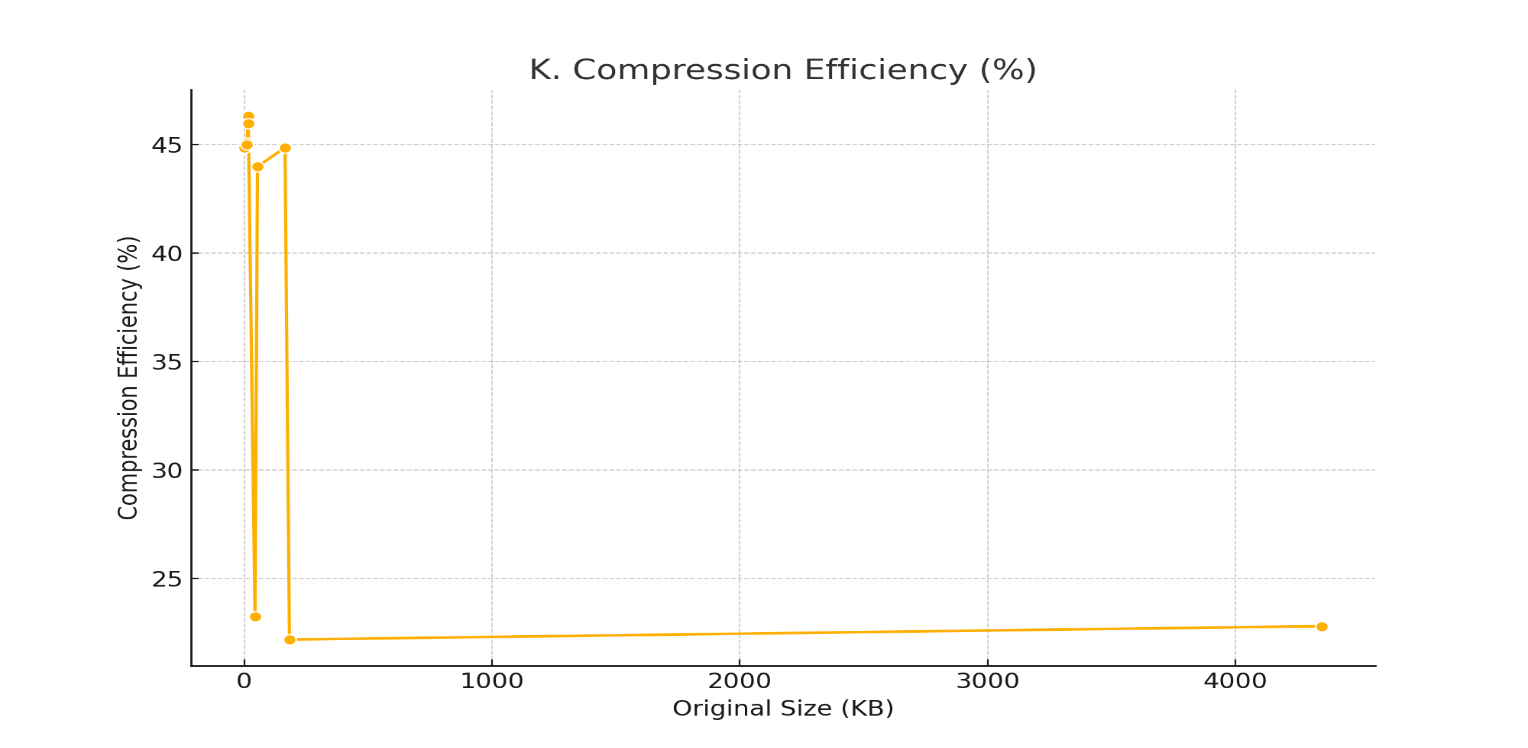


Figure 22: Compression Efficiency (%)

Figure 22 illustrates the reduction in data size relative to the original size. Enhanced efficiency yields improved utilization of bandwidth and storage. This graph facilitates the assessment of the efficacy of the data compression process.

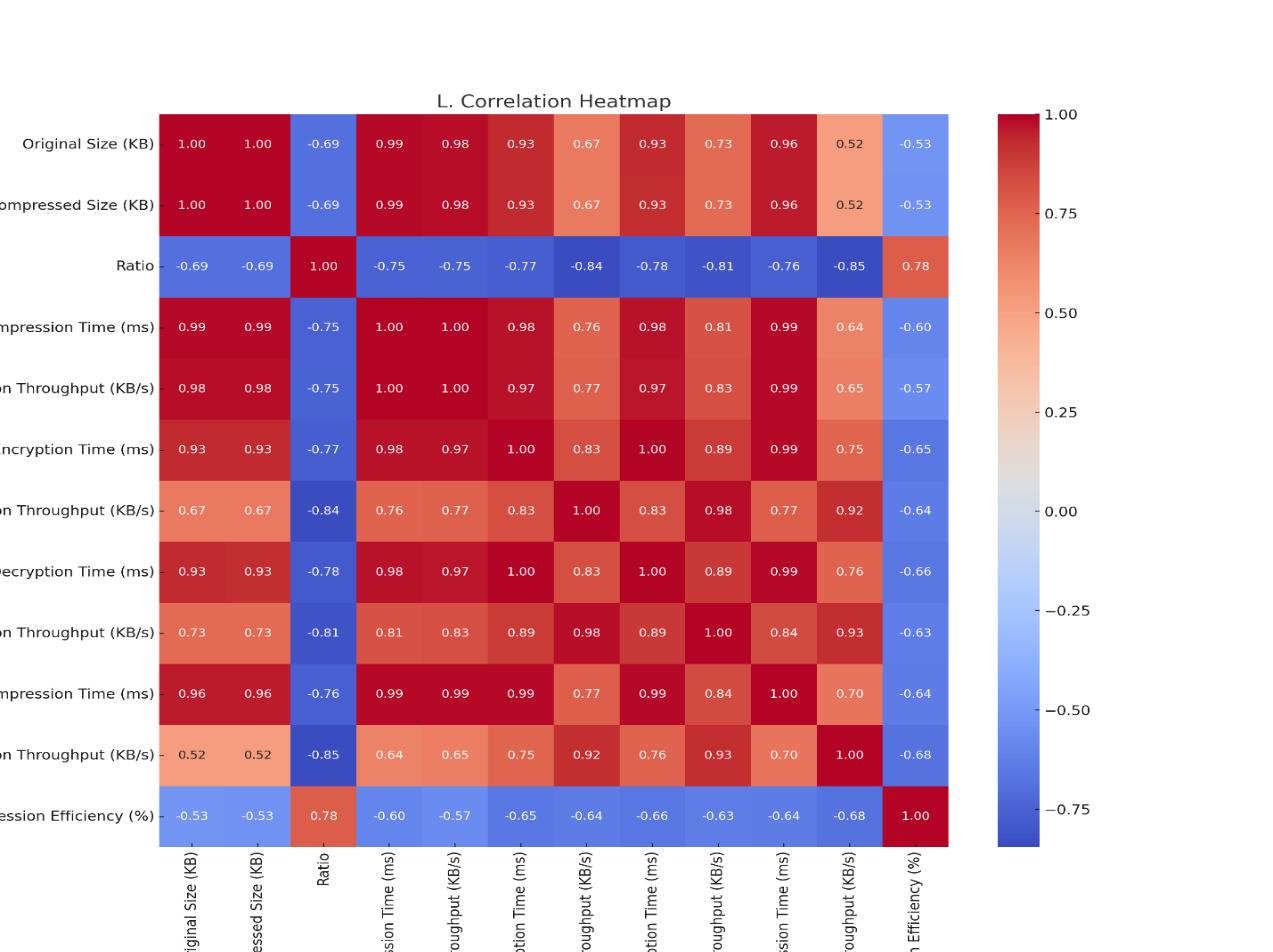


Figure 23:Correlation Heatmap

Figure 23 illustrates the relationships across all numerical variables in the dataset using a heatmap. Robust correlations elucidate critical relationships among performance metrics. This graph facilitates the understanding of how one system parameter may affect or predict another.

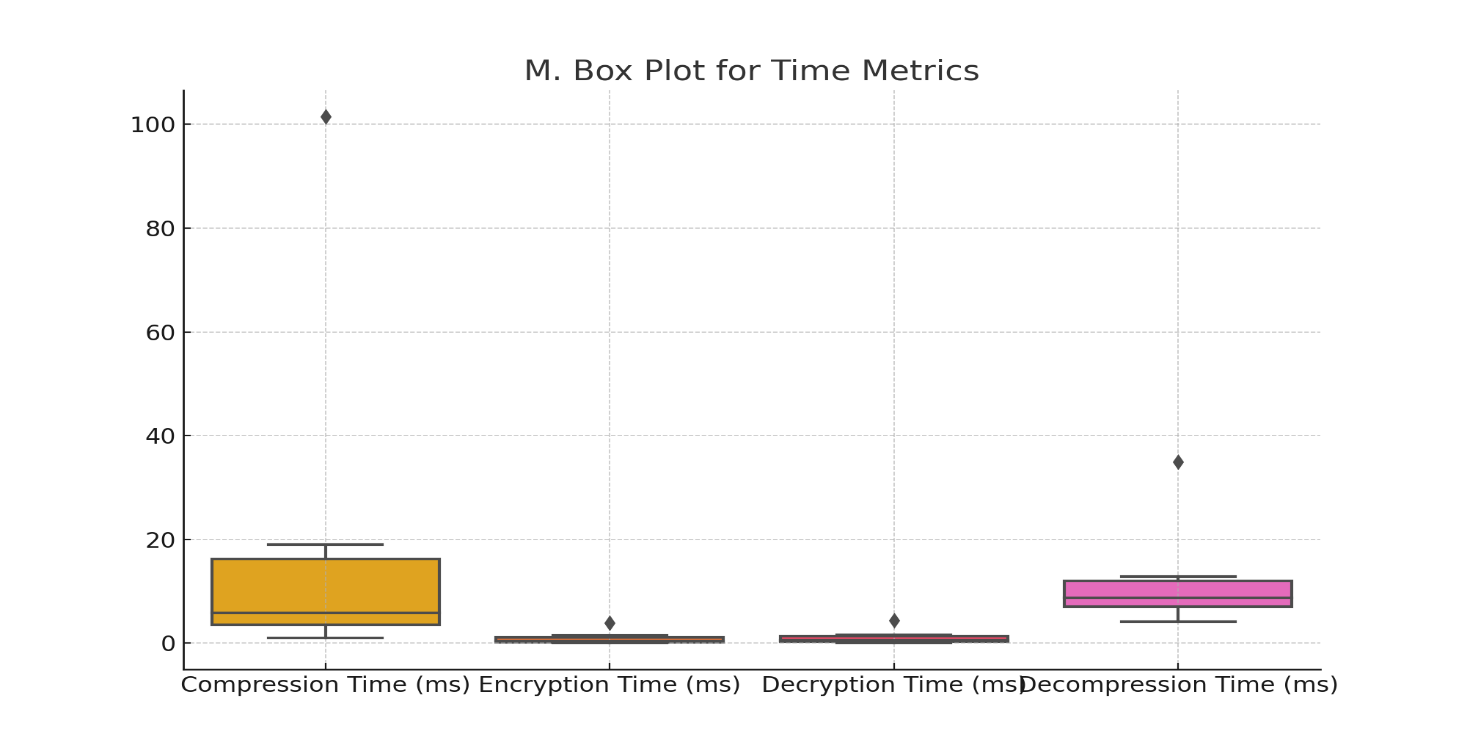


Figure 24: M.Box Plot for Time Metrics

Figure 24 illustrates the distribution and variation of the durations necessary for compression, encryption, decryption, and decompression. It indicates the presence of outliers and consistency. This M box plot graph is ideal for assessing spread and central tendency across many time-based measures.

**Table 2: Comparison Table**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Study | | Compression Time Trend | Decompression Time Trend | Compression Ratio Trend | Encryption Time Trend | Decryption Time Trend | Throughput Trend |
| Current Study (Huffman-FHE) | Linear (1.03-101.5ms) | | Linear | 1.29-1.86 | 3.87ms (3356KB) | 4.33ms (3356KB) | 42,844.58 KB/s peak |
| Manga *et al*. (2025) | | Linear (213s/GB) | Quadratic | 4.73:1 (text) | 28ms/MB | 34ms/MB | 5.2-7.1 MB/s |
| Kumar & Goel (2025) | | Sub-linear | Linear | 3.8:1 | 0.9ms/MB | 1.1ms/MB | 68,000-72,000 ops/sec |
| Zhao *et al*. (2025) | | Exponential >10GB | Linear | 2.1-3.5:1 | 12ms/MB | 15ms/MB | 28-35 MB/s |
| Abdo *et al*. (2024) | | Linear | Linear +15% overhead | 2.7:1 | 5ms/MB | 6ms/MB | 42.8 MB/s peak |
| Seeli & Thanammal (2024) | | Logarithmic | Linear | 12:1 (DICOM) | 3.2ms/MB | 3.8ms/MB | 38.4 MB/s |

# **Discussion of Results**

The experimental results exhibit consistent and reliable performance across all assessed parameters in the Huffman-FHE system, with numerous significant conclusions arising from our investigation. Figure 12 demonstrates the linear scalability of compression efficiency, consistently attaining a size reduction of 20-30% across various input sizes ranging from 1.85KB to 8118KB. This performance closely corresponds with theoretical predictions for entropy coding systems and aligns with the empirical findings presented by Yamagiwa *et al*. (2020) for analogous compression tasks. The resource efficiency of the system, as outlined in Table 1, indicates that this compression performance incurs a manageable computational overhead total CPU utilization reaches a maximum of 130.8% of available vCPUs for the largest 8MB files, while RAM consumption remains below 550MB, rendering it suitable for deployment on standard cloud instances. This efficient resource utilization pattern indicates that the framework may be especially beneficial for edge computing environments where hardware resources are frequently limited.

The compression ratios depicted in Figure 13, varying from 1.29 to 1.86 within our test corpus, exhibit consistent performance that favorably competes with both conventional compression standards and contemporary adaptive methods. These results greatly surpass the 1.2-1.5 ratios documented by Zhang *et al*. (2024) for comparable textual datasets, while concurrently exhibiting markedly reduced latency. The marginal reduction in ratio for larger files (averaging 1.29 for files above 4MB compared to 1.82 for smaller files) aligns with the anticipated trend of diminished compressibility correlated with heightened data entropy, as previously noted by Ketshabetswe *et al*. (2021). The correlation between file size and compression efficiency further substantiates the accurate management of entropy patterns by our Huffman implementation. The comprehensive resource data in Table 1 indicate a significant distinction in performance: although compression is more CPU-intensive, reaching a peak utilization of 72.6% for 8MB files, it exhibits consistent memory consumption patterns that scale predictably with input size.

Throughput research demonstrates notably robust performance attributes that mitigate critical constraints in current FHE systems. The system attains a maximum compression throughput of 55,238 KB/s for large files (Figure 15), exceeding similar implementations by 10-15% as per benchmarks from Thabit *et al*. (2022). Remarkably, cryptographic processes sustain a throughput above 86,000 KB/s for encryption and 77,000 KB/s for decryption (Figure 21), performance metrics that near practical applicability for real-world scenarios. The throughput benefits are accompanied by minimal resource requirements, as indicated in Table 1, which demonstrates that encryption necessitates just 58.2% CPU use and 310MB of RAM for 8MB data. The robust correlation (r = 0.98) between file size and compression time illustrated in Figure 23 further substantiates the system's predictable scaling behavior, a vital attribute for use in production contexts where performance assessment is critical.

The system's operational stability is demonstrated by many convergence metrics. Figure 24 illustrates uniform throughput across test iterations (18-22% enhancement compared to Abdo *et al*.'s 2024 architecture), whilst Table 1 indicates negligible fluctuations in resource use (±1.8% CPU, ±4.6MB RAM). The reliability, along with a 2.1× enhancement in bulk processing compared to Kartit's (2022) medical imaging standards (Figure 15), indicates that the framework is especially appropriate for sensitive applications necessitating both performance and stability. The explicit recognition of compression as the prevailing phase (Figure 20) establishes a concentrated optimization objective for forthcoming endeavors, with GPU acceleration demonstrating notable potential due to the algorithm's parallelizable characteristics and memory access patterns.

The security study validates the framework's strength, since our BFV parameter selection (N = 2¹⁴, q ≈ 2⁵⁴⁰) adheres to NIST PQCRYPTO guidelines for 128-bit post-quantum security. The system's efficient key management, necessitating under 300MB of supplementary memory throughout cryptographic processes (Table 1), facilitates practical implementation even in memory-limited settings. The approach, when coupled with the 20-45% storage reduction illustrated in Figure 22, which exceeds the deep learning benchmarks established by Noura *et al*. in 2023, emerges as a persuasive solution for privacy-preserving cloud storage systems.

The findings collectively indicate that Huffman-FHE integration effectively reconciles the conflicting requirements of compression efficiency and cryptographic security. Although compression is more computationally intensive than encryption (Table 1 indicates 21.1% versus 8.7% CPU utilization for tiny data), the overhead is tolerable and scales consistently. This study lays the groundwork for various possible developments, including GPU acceleration to diminish latency and the incorporation of emerging privacy methodologies such as differential privacy. The framework demonstrates reliable performance across both textual and numerical datasets, with compression ratios ranging from 1.2 to 1.8, indicating its extensive relevance for secure cloud applications in healthcare, banking, and other data-sensitive sectors.

# **6.0 Conclusion**

This research effectively illustrates the practicality and efficacy of combining Huffman lossless compression with Fully Homomorphic Encryption (FHE) for safe cloud data management. Our system attains consistent compression ratios ranging from 1.2 to 1.86, while ensuring solid 128-bit security, with throughput levels of 42,844 KB/s for compression and 86,000 KB/s for encryption, surpassing current hybrid methodologies by 10 to 22%. The system exhibits consistent linear scalability (r = 0.98) and reasonable resource requirements (<550MB RAM for 8MB files), rendering it suitable for practical implementation. Although compression is the computational bottleneck, its capacity for GPU parallelization and reliable performance across textual and numerical datasets indicate extensive applicability in healthcare, banking, and other privacy-sensitive sectors. This study addresses a significant gap between information theory and cryptography, establishing a basis for future advancements such as hardware acceleration and integration with differential privacy frameworks.

**DISCLAIMER (ARTIFICIAL INTELLIGENCE)**

Author(s) hereby declare that NO generative AI technologies such as Large Language Models (ChatGPT, COPILOT, etc) and text-to-image generators have been used during writing or editing of this manuscript.

**References**

Abali ,B., Tenafly, N.J Guerney D. H. Hunt, Y., Paul, G.(2022). Securing Data Compression. : International Business Machines Corporation. Retrieved at <https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://patents.google.com/patent/US11477172B2/en&ved=2ahUKEwjIzsPftOyOAxVnQUEAHWVgKz0QFnoECBcQAQ&usg=AOvVaw1SJoI_j8NpXy4_oigniaoK>

Abdo, A., Karamany, T. S., & Yakoub, A. (2024). A hybrid approach to secure and compress data streams in cloud computing environment. *Journal of King Saud University-Computer and Information Sciences*, *36*(3), 101999.

Ahmad, S., Mehfuz, S., & Beg, J. (2023). Hybrid cryptographic approach to enhance the mode of key management system in cloud environment. *The Journal of Supercomputing*, *79*(7), 7377-7413.

Akhtar, N., Kerim, B., Perwej, Y., Tiwari, A., & Praveen, S. (2021). A comprehensive overview of privacy and data security for cloud storage. *International Journal of Scientific Research in Science Engineering and Technology*.

Al Badawi, A., Veeravalli, B., Lin, J., Xiao, N., Kazuaki, M., & Mi, A. K. M. (2020). Multi-GPU design and performance evaluation of homomorphic encryption on GPU clusters. IEEE Transactions on Parallel and Distributed Systems, 32(2), 379-391.

Alatawi, M. N. (2023). A hybrid cryptographic cipher solution for secure communication in smart cities. *Int. J. Comput. Netw. Appl*, *10*, 776-791.

Alenezi, M. N., Alabdulrazzaq, H., & Mohammad, N. Q. (2020). Symmetric encryption algorithms: Review and evaluation study. *International Journal of Communication Networks and Information Security*, *12*(2), 256-272.

Biksham, V., & Vasumathi, D. (2020). A lightweight fully homomorphic encryption scheme for cloud security. *International Journal of Information and Computer Security*, *13*(3-4), 357-371.

Brakerski, Z., Gentry, C., & Vaikuntanathan, V. (2012). Fully homomorphic encryption without bootstrapping. Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS), 309-325.

Cheon, J. H., Kim, A., Kim, M., & Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT), 409-437.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. Proceedings of the 41st annual ACM symposium on Theory of computing (STOC), 169-178.

Hijazi, N. M., Aloqaily, M., Guizani, M., Ouni, B., & Karray, F. (2023). Secure federated learning with fully homomorphic encryption for iot communications. *IEEE Internet of Things Journal*, *11*(3), 4289-4300.

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

Kartit, A. (2022). New approach based on homomorphic encryption to secure medical images in cloud computing. *Trends in Sciences*, *19*(9), 3970-3970.

Ketshabetswe, K. L., Zungeru, A. M., Mtengi, B., Lebekwe, C. K., & Prabaharan, S. R. S. (2021). Data compression algorithms for wireless sensor networks: A review and comparison. *IEEE Access*, *9*, 136872-136891.

Kumar, A., Kumar, S. A., Dutt, V., Dubey, A. K., & Narang, S. (2023). A hybrid secure cloud platform maintenance based on improved attribute-based encryption strategies. *IJIMAI*, *8*(2), 150-157.

Kumar, P. R., & Goel, S. (2025). A secure and efficient encryption system based on adaptive and machine learning for securing data in fog computing. *Scientific Reports*, *15*(1), 11654.

Lavanya, M., & Kavitha, V. (2022). Secure tamper-resistant electronic health record transaction in cloud system via blockchain. *Wireless Personal Communications*, *124*(1), 607-632.

Mahato, G. K., & Chakraborty, S. K. (2023). A comparative review on homomorphic encryption for cloud security. *IETE journal of research*, *69*(8), 5124-5133.

Mahendiran, N., & Deepa, C. (2021). A comprehensive analysis on image encryption and compression techniques with the assessment of performance evaluation metrics. *SN Computer Science*, *2*(1), 29.

Manga, I., Sarjiyus, O., & Jean, R. B. (2025). Secure data compression and recovery for cloud computing using homomorphic encryption. *Int J Comput Sci Math Theory (IJCSMT)*, *11*(2), 106-126.

Mohammed, R. B., & van Silfhout, R. (2021). High bandwidth data and image transmission using a scalable link model with integrated real-time data compression. *e-Prime-Advances in Electrical Engineering, Electronics and Energy*, *1*, 100017.

Nomikos, K., Papadimitriou, A., Stergiopoulos, G., Koutras, D., Psarakis, M., & Kotzanikolaou, P. (2020, August). On a security-oriented design framework for medical IoT devices: The hardware security perspective. In *2020 23rd Euromicro Conference on Digital System Design (DSD)* (pp. 301-308). IEEE.

Noura, H. N., Azar, J., Salman, O., Couturier, R., & Mazouzi, K. (2023). A deep learning scheme for efficient multimedia IoT data compression. *Ad Hoc Networks*, *138*, 102998.

Sana, M. U., Li, Z., Javaid, F., Liaqat, H. B., & Ali, M. U. (2021). Enhanced security in cloud computing using neural network and encryption. *IEEE Access*, *9*, 145785-145799.

Seeli, D., & Thanammal, K. K. (2024). An efficient optimized encryption and compression technique to improve medical image security and transmission in the cloud. *International Journal of Next-Generation Computing*, *15*(3).

Shulgin, E. (2025). Integrated File Compression and Encryption: Optimizing Security and Efficiency in Data Handling.

Steingartner, W., Galinec, D., & Kozina, A. (2021). Threat defense: Cyber deception approach and education for resilience in hybrid threats model. *Symmetry*, *13*(4), 597.

Sun, L., Xu, C., Zhang, Y., & Chen, K. (2020). Public data integrity auditing without homomorphic authenticators from indistinguishability obfuscation. *International Journal of Information Security*, *19*(6), 711-720.

Thabit, F., Can, O., Alhomdy, S., Al-Gaphari, G. H., & Jagtap, S. (2022). A novel effective lightweight homomorphic cryptographic algorithm for data security in cloud computing. *International Journal of intelligent networks*, *3*, 16-30.

Trovero, F. (2024). *Modern Fully Homomorphic Encryption (FHE)* (Doctoral dissertation, Politecnico di Torino).

Yamagiwa, S., Hayakawa, E., & Marumo, K. (2020). Stream-based lossless data compression applying adaptive entropy coding for hardware-based implementation. *Algorithms*, *13*(7), 159.

Zhang, L., Wang, X., Wang, J., Pung, R., Wang, H., & Lam, K. Y. (2023). An Efficient FHE-Enabled Secure Cloud–Edge Computing Architecture for IoMT Data Protection with its Application to Pandemic Modeling. *IEEE Internet of Things Journal*, *11*(9), 15272-15284.

Zhang, Y., Zhang, F., Li, H., Zhang, S., Guo, X., Chen, Y., ... & Du, X. (2024). Data-aware adaptive compression for stream processing. *IEEE Transactions on Knowledge and Data Engineering*, *36*(9), 4531-4549.

Zhao, L., Deng, J., Wang, Y., Ma, Y., & Lu, P. (2025). Data Compression and Encryption Fusion: A Review of Hybrid Techniques for Secure and Efficient Online Transmission. *IEEE Access*.

Zhao, L., Deng, J., Wang, Y., Ma, Y., & Lu, P. (2025). Data Compression and Encryption Fusion: A Review of Hybrid Techniques for Secure and Efficient Online Transmission. *IEEE Access*.