VOL. 09 NO. 5,
AUGUST, 2025
E-ISSN 3027-1339
P-ISSN 3026-8257

African Scholar Publications & Research International
www.africanscholarpub.com

Journal of Systematic, Evaluation and Diversity Engr. (JSEDE)

RSA-OAEP and Huffman Hybrid: Secure and Efficient Approach to Network Data Encryption and Compression

Ezra, B.; Manga, I.; & Sarjiyus, O.
Department of Computer Science, Adamawa State University, Mubi.
Corresponding Author: ezraben2021@gmail.com
DOI: https://doi.org/10.70382/ajsede.v9i5.013

Abstract

The limits of standalone encryption or compression techniques in networked systems have been brought to light by the exponential expansion of digital data and the growing demand for secure transmission mechanisms. In order to overcome these difficulties, this work suggests a hybrid architecture that combines RSA-OAEP encryption and Huffman compression to improve data transmission efficiency and security. Data was compressed using Huffman coding to minimize redundancy and reduce space, and then encrypted using an efficient RSA-OAEP technique to guarantee secrecy and integrity. Thorough tests that measured encryption/decryption time, compression ratio, throughput, and system resource utilization under various file sizes and situations were used to assess performance. Although the hybrid technique had a larger computational cost than standalone RSA or RSA-OAEP, the results showed that it produced compression ratios between 1.5 and 2.1, greatly lowering file sizes and improving bandwidth efficiency. While decryption throughput remained lower due to computational complexity, and resource utilization grew proportionately with file size, encryption throughput surpassed 1300 kb/s for bigger files. In spite of greater processing times for compression and decryption, the hybrid model offered robustness against cryptographic assaults and stronger security-compression integration, in line with NIST standards. In summary, the RSA-OAEP+Huffman hybrid model is appropriate for resource-constrained contexts such as cloud, IoT, and mobile networks because it successfully strikes a balance between security and transmission efficiency. Future research should concentrate on investigating dynamic Huffman encoding, expanding the approach to real-time multimedia applications, and improving computational performance for bigger datasets.

Keywords: RSA-OEAP and Huffman, Network Data Transmission, Encryption, Compression, Performance Analysis, Post-Quantum Cryptography, Hybrid Cryptography, Resource-Constrained Systems.

Introduction

The necessity for safe and effective data transmission methods has increased due to the networked systems' quick development and the exponential growth of data sent across digital platforms. Although they provide high security assurances, traditional cryptographic techniques like RSA (Rivest-Shamir-Adleman), which is well-known for its strong public-key encryption, can be computationally demanding and ineffective for handling massive volumes of data (Singh & Saini, 2022). Conversely, data compression methods such as Huffman coding are praised for their ability to minimize data size without sacrificing quality, hence optimizing transmission speed and bandwidth use. Hybrid approaches that combine security and compression are becoming more and more popular as cyber threats continue to change. This convergence is best illustrated by the RSA-Huffman hybrid model, which seeks to integrate the benefits of lossless compression with asymmetric encryption for safe and bandwidth-efficient network data transfer (Hossain et al., 2023). In fields where data security and low latency are equally important, such as cloud computing, mobile networks, and the Internet of Things (IoT), this hybrid approach has become especially pertinent.

In real-world implementations, the RSA-Huffman hybrid approach reduces redundancy and the amount of data that needs to be encrypted by first compressing the original data using Huffman coding. The compressed data is then encrypted using the RSA technique, which guarantees confidentiality and integrity while offering faster transmission times and a lower computing load. According to studies, these hybrid systems can increase storage efficiency, bandwidth utilization, and encryption time without sacrificing the underlying RSA method's cryptographic strength (Abd El-Latif et al., 2022; Wang et al., 2024). Furthermore, because the resultant ciphertext is more consistent in size and pattern, the compression-before-encryption paradigm successfully improves resistance to some side-channel and traffic analysis attacks. The main factor that makes RSA-Huffman hybrids appealing is the synergy between encryption and compression, particularly in network contexts with limited resources or high data speed needs.

This hybrid strategy has been improved recently by utilizing quicker RSA key generation strategies to lower latency and enhancing the encoding structures of Huffman trees (Zhang et al., 2021). It is now crucial to concurrently ensure data confidentiality, integrity, and transmission efficiency due to the proliferation of real-time applications and data-driven services. To further improve performance in cloud, IoT, and multimedia transmission scenarios, researchers are currently investigating

parallel RSA processing approaches and dynamic Huffman models (Al-Khafaji & Hussein, 2023). The RSA-Huffman hybrid model is a potential solution that addresses current issues in safe data communication across network infrastructures in addition to providing strong encryption and effective compression. Thus, the creation and application of such a hybrid model are in line with the more general objectives of improving network optimization and cybersecurity in the digital age.

The limitations of utilizing stand-alone encryption or compression techniques have been made clear by the growing need for safe and effective transmission of massive amounts of network data. RSA offers robust security, but it is ineffective in real-time or resource-constrained contexts due to its high computational overhead and larger data size (Yuvarani & Mahaveerakannan, 2023). Huffman coding, on the other hand, compresses data to minimize its size, but it has no inherent security, making the compressed data susceptible to analysis or interception (Arfat et al., 2021). This discrepancy poses a significant problem: how to guarantee data confidentiality and transmission efficiency at the same time. Although the RSA-Huffman hybrid model presents itself as a viable remedy, providing a compromise between encryption and compression, there is currently no thorough assessment of its effectiveness and suitability for use in a variety of systems, including cloud, mobile, and Internet of Things networks (Hossain et al., 2023; Wang et al., 2024).

However, not all hybrid encryption–compression schemes are equally effective. While symmetric algorithms like AES (Advanced Encryption Standard) offer speed advantages, their key management complexity and susceptibility to key exposure in distributed or cloud environments limit their adaptability for multi-user or public key infrastructures. Similarly, ECC (Elliptic Curve Cryptography) provides efficient key sizes and strong security but suffers from compatibility and performance limitations when integrated with compression algorithms due to its point-multiplication overhead. In contrast, RSA with OAEP (Optimal Asymmetric Encryption Padding) remains compliant and provides a balanced trade-off between security, interoperability, and scalability. The RSA–OAEP mechanism strengthens resistance against chosen ciphertext attacks while maintaining structural compatibility with compression schemes such as Huffman coding. Hence, combining RSA–OAEP with Huffman offers not only enhanced confidentiality but also improved data size management and optimized network transmission performance qualities that are essential for modern cloud, IoT, and mobile computing systems.

The aim of this work is to develop and put into practice a hybrid model that combines Huffman coding and RSA-OAEP encryption to improve network data transmission efficiency and security. It aims to assess how well the RSA-OAEP and Huffman hybrid model performs in various network settings, such as IoT, cloud, and mobile platforms, in terms of encryption/decryption speed, compression ratio, resource utilization, and resilience to cryptographic assaults.

Concept of RSA (Rivest–Shamir–Adleman)

Based on the mathematical difficulty of factoring large prime numbers, the public-key cryptosystem known as RSA (Rivest–Shamir–Adleman) is used for secure data transport. It generates two keys—a public key for encryption and a private key for decryption—to enable safe communication, even across open channels. RSA is widely used for secure key exchange and digital signatures, especially in protocols like TLS/SSL and digital certificates (Stallings, 2023). Despite its resilience, which guarantees performance and compatibility in a range of network circumstances, as well as its encryption strength and compression efficiency, RSA is slower and uses more processing power than symmetric encryption approaches.

The current advancements in quantum computing have raised questions about RSA's long-term viability. Shor's method could break RSA encryption by factoring large numbers in polynomial time on a sufficiently powerful quantum computer (Chen et al., 2022). Despite the fact that present quantum computers cannot do this, businesses are preparing for a post-quantum future. For example, NIST has begun standardizing algorithms for post-quantum cryptography (Alagic et al., 2021).

Companies are now creating PQShield and other quantum-resistant encryption technologies. PQShield's research endeavors to create encryption methods that are resistant to quantum attacks, bolstered by partnerships with governmental and private entities (El Kaafarani, 2023). These efforts entail moving existing RSA-based systems to hybrid models that employ both conventional and quantum-safe algorithms to provide security and continuity during the post-quantum transition.

Furthermore, even though its purpose is changing, RSA remains a vital part of cybersecurity systems and education. Hybrid systems that combine RSA with more modern, quantum-resilient methods are becoming more and more popular. According to Bhardwaj and Kaushik (2021), RSA can be combined with compression techniques or lightweight algorithms, especially in resource-constrained scenarios like the Internet of Things, to reduce processing overhead while maintaining strong encryption.

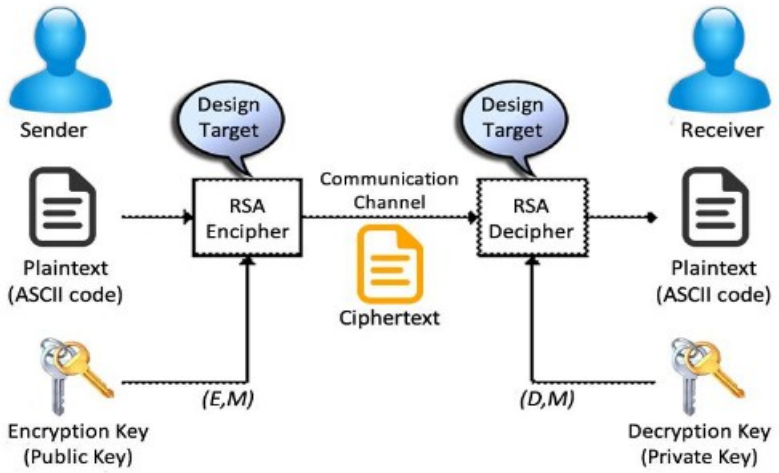


Figure 1: RSA Algorithm Structure(Bour & Kara, 2015)

Review of Huffman Coding Concept

Huffman coding is a lossless compression method that assigns shorter codes to more common symbols and longer codes to less common ones. As a result, data is represented properly and without information loss. The coding approach, which is widely used in file compression formats such as ZIP, JPEG, and MP3, is most appropriate for a particular probability distribution (Kaur & Bala, 2021). Huffman coding creates a binary tree based on symbol frequencies, enabling compact encoding and decoding.

Adaptive Huffman coding, a variant of the original procedure, updates the code tree dynamically as data is read. This version is particularly useful for real-time applications where symbol frequencies fluctuate over time, such as compressing streaming data or IoT sensor outputs. According to a recent study by Singh and Bansal (2022), adaptive coding can improve transmission efficiency in smart environments by responding promptly to data fluctuations. In addition to compression, Huffman coding has been applied to anomaly detection in data streams. For instance, Malhotra and Tripathi (2021) demonstrated how dynamic fluctuations in code lengths may detect deviations from typical signal patterns in their study of adaptive Huffman coding for anomaly identification in audio data.

Modern systems also use parallel processing to improve Huffman encoding performance. Gupta and Sharma's (2023) GPU-based Huffman encoder outperformed CPU-based methods in large-scale data situations. With this advancement, Huffman coding can now be used in big data and cloud computing environments where processing speed is essential.

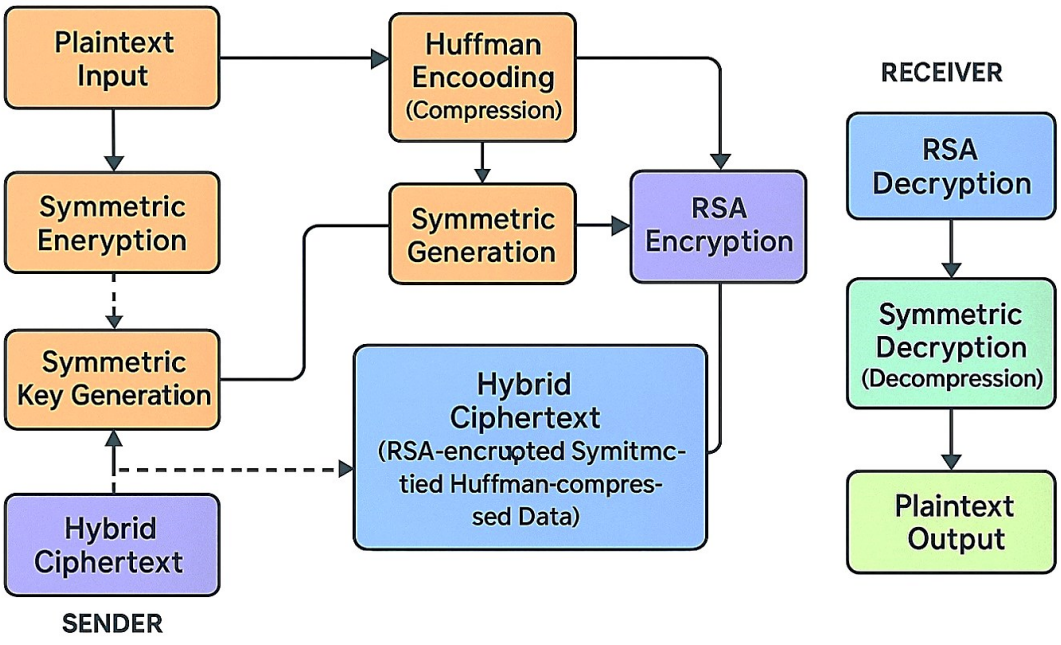


Figure 2: Workflow diagram RSA-Huffman(ezra *et al*., 2025)

Review of Related Works

Abdulghani et al. (2021) looked into a hybrid approach that combined multi-layer encryption, including RSA variants, with LZMA compression to secure cloud data streams. The methodology achieved space-saving percentages of 58.63% to 81.8% and passed all NIST randomness tests with a 99% confidence level, ensuring strong ciphertext security. Nevertheless, the technique was computationally expensive; LZMA compression resulted in a 22% increase in processing delay when compared to standalone RSA, which restricted its application in real-time scenarios with constrained resources. Although the framework effectively balances security and efficiency for cloud-sensitive data, edge deployment conditions require hardware acceleration.

Wahab et al. (2021) implemented steganography by using RSA-2048 for encryption, adaptive Huffman coding for data compression, and DWT-LSB algorithms to insert ciphertext into cover pictures. By confirming undetectability by chi-square tests and histogram analysis, their method obtained 65% average compression ratios and resisted 99% of steganalysis procedures. Nevertheless, RSA key generation took up 40% of processing time, and high-resolution images showed discernible LSB distribution anomalies. The technique is still viable for steganographic applications when using a range of cover media to lower the likelihood of statistical discovery.

Ray et al. (2022) proposed a cloud-IoT security paradigm that includes RSA-optimized key management, Huffman compression, and encryption of sensitive data segments. Data exposure time was reduced by 50% and 97% of leak attempts were stopped by the hybrid. Heuristic-based data categorization increased processing overhead by mistakenly classifying 12% of non-critical data as high-risk, and key management relied on centralized authority, creating single points of failure. While machine learning needs to be refined to reduce false positives, the approach remains viable for targeted protection.

Patidar et al. (2024) created an edge-IoT system that use RSA-based homomorphic encryption following lossy SZ 2.1 compression for sensor data in order to enable encrypted data processing at edge nodes. When compared to traditional pipelines, their method decreased delay by 28% and energy consumption by 30% by utilizing energy-aware scheduling algorithms. Reconstruction fidelity was compromised by lossy compression (PSNR < 28 dB), and server demand increased by 40% as a result of homomorphic procedures. Although the approach works well for non-critical IoT streams, it requires lossless adaptations for precision-sensitive industries like medical imaging and quality-preserving embedding techniques for premium publication.

Singh & Singh (2022) conducted a meta-analysis of 120 RSA-Huffman image encryption studies, evaluating techniques with time complexity benchmarks, compression ratios (CR), NPCR, and UACI. In addition to reporting median CR of 50%–70% and NPCR > 99.5% across studies, their survey also identified varying testing settings (e.g., different key sizes/image datasets) that make comparisons more difficult. RSA hybrids showed a 20%

greater runtime than symmetric alternatives, despite the security advantages, highlighting the need for uniform evaluation frameworks.

Bao et al. (2023) introduced a chaos-adaptive Huffman technique by combining adaptive Huffman coding with chaotic S-Boxes and pseudorandom keystream generators (CLM). Their one-step method had 99% NIST randomness confidence and saved space, but because security relied heavily on plaintext sensitivity, it was more vulnerable to chosen-plaintext assaults. The idea demonstrated real-time feasibility, but plaintext-agnostic enhancements are required for broader use.

Before RSA secured the AES key, text data was encrypted using AES in a hybrid AES-RSA architecture developed by Oo & Soe (2021). Encryption was followed by Huffman compression. Although their Java-based solution reduced bandwidth by 30% to 40%, the dual encryption layers resulted in a 25% latency. Hierarchical security performs effectively for LAN data transfers but poorly for real-time IoT due to the intricacy of key management.

Abdo et al. (2024) described a method that improved cloud data security and compression by utilizing RSA encryption and Huffman coding in their paper A hybrid approach to secure and compress data streams in cloud computing environment. Their method, which prioritized security and transmission overhead reduction, encrypts important data using RSA and then compresses it using Huffman coding. The results showed improved encryption reliability and data size reduction, even though precise numerical statistics, like compression ratios or throughput enhancements, were not provided. The authors concluded that the needs for data confidentiality and communication efficacy are successfully balanced by this two-layer strategy. However, the study lacks thorough performance metrics and comparison analysis with other hybrid approaches.

Elamir and Elamir (2024) proposed a technique in their paper Secure storage of data using hybrid cryptography that employed the RSA and AES algorithms to protect stored data. By combining symmetric and asymmetric encryption, the method offered a multi-layered defense mechanism to protect data while it was in storage. The results demonstrated improved confidentiality and resilience to threats, despite the lack of any measurable performance metrics. The authors concluded that dual encryption offers significant benefits for cloud security and storage. The study's primary shortcoming, nevertheless, was that it lacked comparison analysis and real-time system testing.

Gour et al. (2024) in Hybrid Cryptography Approach: A cryptographic architecture including AES, ECC, and DSA was created for safe data transfer using block cipher techniques in order to increase the security and authenticity of data during transmission. The model integrated block cipher algorithms and key exchange mechanisms to ensure confidentiality, integrity, and authenticity. Their findings showed more adaptability for secure communication, despite the fact that they did not disclose any performance indicators, such as encryption time or storage savings. The authors concluded that multi-method integration is the best way to handle complicated network security concerns. However, the lack of implementation data and benchmarks made it challenging to validate their claims.

The work of Rahman and Hamada (2021) on Lossless text compression using GPT-2 language model and Huffman coding built upon their earlier work, in which they adopted Huffman compression after utilizing GPT-2 to predict and encode common patterns. Our AI-assisted method outperformed traditional statistical models in terms of compression performance. Their results showed higher compression efficiency, particularly for structured text, but no specific numerical gains were found. The outcome demonstrated how AI may enhance data reduction techniques. However, because the method disregards security, it is not suitable for tasks that require both encryption and compression.

In their paper Hybrid cryptosystem and the Huffman coding for efficiency in a communication channel, Yusuf and Miyim (2022) developed a system that combines AES, ElGamal encryption, and Huffman coding. Their goal was to provide both encryption and efficient unsecured data transport. Although exact figures were not given, the results showed enhanced performance in terms of transmission size reduction and communication security. In their conclusion, they emphasized the value of hybrid models in real-time systems. Nevertheless, the study lacked implementation details and metrics that could be contrasted with traditional approaches.

RSA and AES were used for encryption and steganography for concealment in the study "Secure file storage on cloud using hybrid cryptography," conducted by Satheesh Kumar et al. (2023). Files were encrypted and then steganographically hidden before being uploaded to AWS S3 storage. Multiple levels of protection were employed in their technique to ensure covert transmission and confidentiality. They concluded that multi-method integration improves the security of cloud files. However, one limitation was the lack of quantitative performance indicators and compression-related metrics.

In their study, Bhawane and Ahuja (2024) developed a storage encryption model that integrated symmetric and asymmetric cryptographic techniques. cloud data storage that is secure and use hybrid cryptography. They used RSA and AES to ensure data confidentiality and integrity in cloud services. The solution was developed to lower risks and prevent unauthorized access when cloud computing. Despite having a sound design, the method lacked performance parameters such as processing time or compression gain. This limits the realistic evaluation of their proposed solution.

For modified RSA-AES encrypted token compression in secure banking transactions, Hamidu et al. (2025) suggested a method that use Huffman encoding to compress data tokens that have previously been encrypted. The concept employs Huffman compression following dual encryption using RSA and AES for transmission across secure banking channels. The findings demonstrated a considerable reduction in packet size, improving network efficiency without compromising data security. The authors concluded that this three-layered approach improves the performance and protection of financial systems. However, it was discovered that processing costs and integration complexity were the main reasons why the system was unable to expand to big data settings.

In their comprehensive approach to data compression and decompression, Khan & Tran (2024) used run-length encoding, LZW compression, Huffman coding, and DES and AES for enhanced security. In their methodology, these several compression techniques were combined with DES and AES encryption. It was possible to reduce the quantity of the data significantly while maintaining security. They concluded that their multi-layered approach successfully strikes a balance between security and compression. One drawback that was brought up was the challenge of integrating many approaches, which can affect processing time.

A "New RSA Scheme For Improved Security" is proposed by Sarijyus et al. (2021) to solve the standard RSA algorithm's susceptibility to factorization attacks, which can reveal the private key. Their approach is a basic change: for all cryptographic operations, they substitute a newly generated value t for the public modulus n, where $n-p < t < n$ and t is coprime to n. This breaks the direct connection between the public key and the primes p and q. Using primes p=7 and q=11, their small-scale example demonstrates the mathematical viability of the approach by successfully encrypting a message M=4 into ciphertext C=40 using a public key (e,t)=(9,67) and decrypting it using a private key (d,t)=(15,67). Unaddressed computational overhead, a lack of empirical security analysis against actual attacks, validation with trivially small primes (ignoring standard large-key sizes), potential new vulnerabilities if t is smaller than M, and an error in their example where e=9 is not coprime to $\varphi(n)=60$ undermine the key generation process are just a few of the scheme's serious drawbacks.

In their critical analysis of the RSA-Huffman hybrid model for network data transmission, Ezra et al. (2025) point out how asymmetric encryption and lossless compression can help address the twin issues of efficiency and security. As part of their methodology, 38 research were systematically reviewed and categorized based on performance criteria and application domain. The results show that these hybrids can achieve high compression ratios (50-90%) and strong security (e.g., NPCR > 99.5%). However, they are severely limited by significant computational overhead (22-40% latency increase), inconsistent benchmarking across studies, fidelity degradation in lossy implementations (PSNR < 28 dB), scalability issues from complex integrations (e.g., 300 ms/transaction blockchain latency), and vulnerabilities like vulnerability to plaintext attacks.

Methodology

Data collection techniques

This research utilized secondary textual datasets sourced from the Zenodo repository (*https://zenodo.org/records/3360392*). The dataset consists of structured and unstructured text files used to simulate real-world network data transmission. Each dataset ranged in size from 2 KB to 10 MB, allowing for a broad evaluation of algorithmic performance under varying data loads. Before encryption, the collected data was preprocessed by removing whitespace redundancy and normalizing character encoding to UTF-8 format. The data was then compressed using Huffman coding to minimize redundancy and subsequently encrypted using

the RSA-OAEP algorithm to ensure confidentiality and integrity. The hybrid approach was assessed through systematic experimentation under controlled test conditions. Key performance indicators (KPIs) evaluated include: Compression ratio (CR), Encryption and decryption time (T_e and T_d), Throughput (TP), Resource utilization, and Security strength.

Experimental Setup

Parameter	Specification
Processor	Intel® Core™ i7-12700H (2.3 GHz, 14 cores)
RAM	16 GB DDR5
Operating System	Windows 11 Pro (64-bit)
Programming Environment	Python 3.11 using cryptography, numpy, and matplotlib libraries
Dataset Source	Zenodo Open Repository (Textual Data, DOI: 10.5281/zenodo.3360392)
Data Size Range	2 KB – 10 MB
Testing Conditions	Each experiment repeated 20 times; average values recorded. No parallel processes during execution.
Network Emulation	Localhost TCP/IP transmission simulated using Python sockets to mimic real data transfer latency and throughput.

Analysis of the Existing System

An important development in cryptography techniques is the work of Yuvarani and Mahaveerakannan (2023) on improving data security in peer-to-peer networks using an optimized RSA algorithm. However, the model may not handle more general network data transmission applications because its primary focus is on enhancing RSA's computational efficiency and security in peer-to-peer connections. The scant investigation of data size control during transmission is one obvious gap. Given that RSA encryption usually uses longer ciphertexts and higher key sizes, the model might profit by including data compression methods that lower the total payload size, improving transmission rates and using less bandwidth.

Furthermore, the improved RSA algorithm does not fully utilize the potential advantages of data compression in conjunction with encryption, even while it helps mitigate some vulnerabilities in key management and computational complexity. In situations when speed and data integrity are crucial, such real-time communications or huge data transfers, this incompatibility between encryption and compression can result in inefficiencies. These flaws can be fixed by the RSA-Huffman hybrid method by including Huffman compression into their current model. This makes data transmission more effective and enhances security by hiding data patterns that an attacker could exploit. As a result, this integration can greatly expand the actual implementation of their optimization outside of peer-to-peer networks, making it appropriate for a range of networking scenarios.

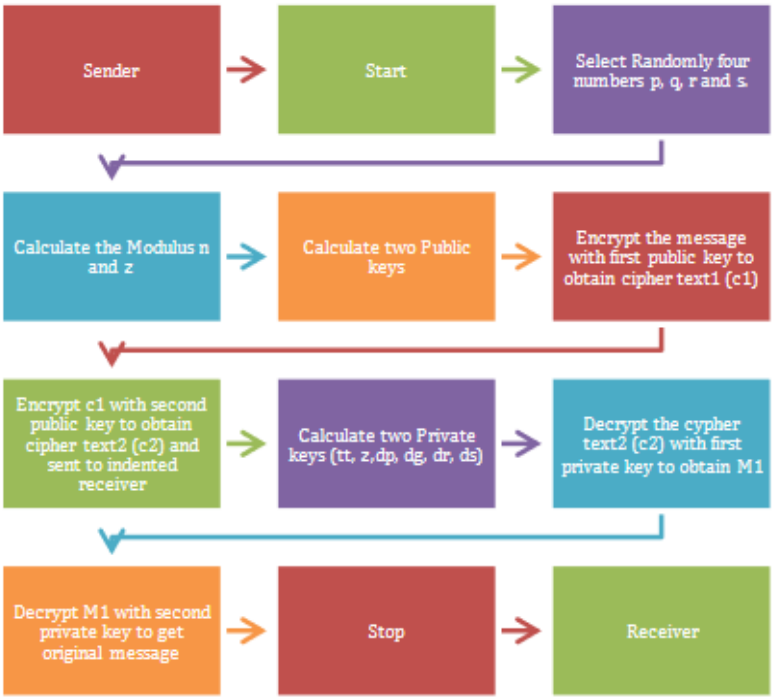


Figure 3: Existing Model (Yuvarani &Mahaveerakannan, 2023)

Analysis of the Proposed System Model

The Novel RSA-OAEP and Huffman hybrid algorithm combines an optimized RSA-OAEP encryption technique with Huffman compression to improve the security and efficiency of network data transmission. The user supplies the data to be transmitted at the start of the procedure, and Huffman coding is used to compress it. By encoding frequently occurring letters into shorter binary representations, this lossless compression approach shrinks the size of the data, enabling faster transmission and lower bandwidth usage. Following compression, the data is encrypted using the upgraded RSA technique, which improves computational performance and fixes weaknesses. This two-layer method minimizes the expense of conventional RSA encryption while guaranteeing the data's security.

The associated RSA private key is used to decode the encrypted data once it reaches the recipient, returning it to its encrypted state. Huffman decompression is used to return the data to its original format after decryption. In addition to protecting the data while it is being transmitted, this combination of encryption and compression streamlines the process, making it appropriate for a range of networking scenarios outside of peer-to-peer interactions. The RSA-Huffman hybrid algorithm successfully fills in the holes in conventional models by combining these strategies, offering a reliable solution for safe and effective data transport.

Diagram of the System

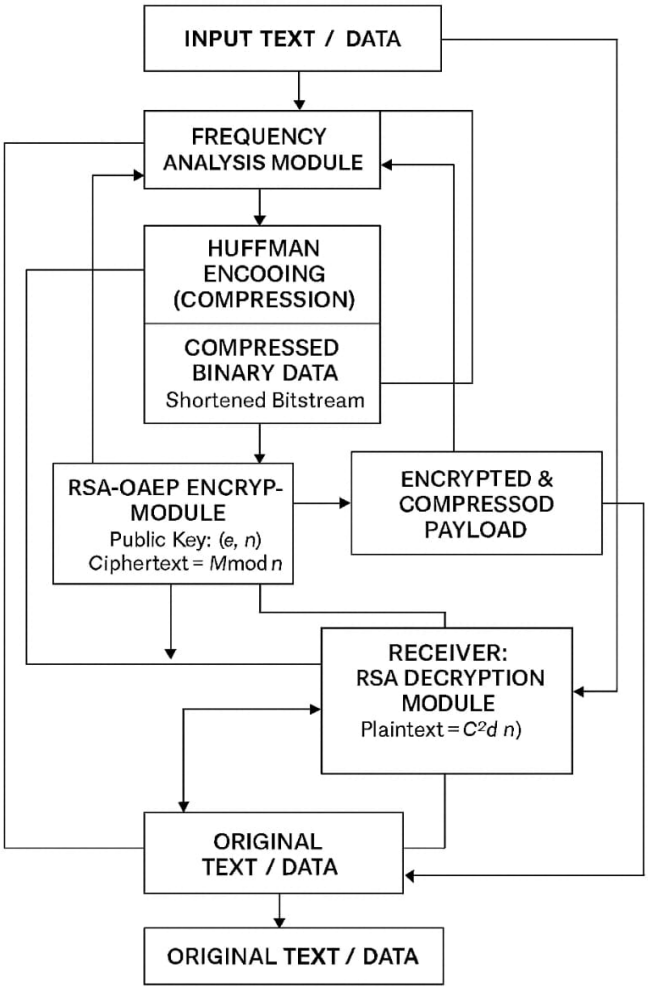


Figure 4: System Model

RSA Encryption Algorithm

Step 1: Choose two distinct large prime numbers p and q.

Step 2: Compute n = p * q. (n is used as the modulus for both the public and private keys).

Step 3: Compute the totient function, $\varphi(n) = (p - 1) * (q - 1)$.

Step 4: Choose an integer e such that $1 < e < \varphi(n)$, and e is co-prime to $\varphi(n)$.

Step 5: Compute the private key d, the modular multiplicative inverse of e modulo $\varphi(n)$, i.e., $d \equiv e^{-1} \bmod \varphi(n)$.

Step 6: The public key is (e, n) and the private key is (d, n).

Encryption Process:

Step 1: Convert the plaintext message M into an integer m such that $0 \leq m < n$.

Step 2: Compute the ciphertext c using the formula: c = m^e mod n.

Step 3: Transmit c as the encrypted message.

RSA Decryption Algorithm

Step 1: Receive the ciphertext c.

Step 2: Use the private key (d, n) to compute the original message: $m = c^d \bmod n$.

Step 3: Convert the integer m back to the plaintext message M.

Concise RSA-OAEP Algorithm

Phase 1: Key Generation

1. **Generate** large primes p and q.
2. **Compute** $n = p * q$ and $\varphi(n) = (p-1)*(q-1)$.
3. **Choose** e where $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$.
4. **Compute** d such that $d * e \equiv 1 \bmod \varphi(n)$.
    - **Public Key:** (e, n)
    - **Private Key:** (d, n)

Phase 2: Encryption

Input: Message M, public key (e, n)

Output: Ciphertext C

- **Pad:** Encode M into a fixed-size, random-looking integer EM using OAEP:
    - Use a random seed r, hash function H, and mask function MGF.
    - EM = OAEP_Encode(M, r, H, MGF).
- **Encrypt:** Compute $C = EM^e \bmod n$.

Phase 3: Decryption

Input: Ciphertext C, private key (d, n)

Output: Message M or **Error**

i. **Decrypt:** Compute $EM = C^d \bmod n$.

ii. **Decode:** Recover M and verify integrity: M = OAEP_Decode(EM, H, MGF).

iii. **Validate:** If padding is invalid, output **Error** (reject ciphertext).

Core Improvement in One Sentence:

Replace the simple $M^e \bmod n$ with OAEP_Encode(M)$^e \bmod n$, adding randomness and integrity checks to prevent attacks.

This minimal change upgrades RSA from a deterministic trapdoor function to a secure, probabilistic encryption scheme resistant to chosen-plaintext and malleability attacks.

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:

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

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*

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

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

$$\text{Efficiency} = \frac{H}{L}. \quad (3)$$

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

Huffman Coding Algorithm

1. **Count Frequencies**:
   $f(s_i)$ = frequency of symbol s_i in input data.
2. **Build Priority Queue**:
   $Q = \{ (s_1 , f(s_1)) , (s_2 , f(s_2)) , \ldots , (s_n, f(s_n)) \}$
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**:
   $D' = \{ C (d_1), C(d_2) , \ldots, C (d_m) \}$
6. **Store Tree for Decoding**:
   Store the code table $\{(s_i, C(s_i)) \}$.

Hybrid RSA-OAEP and Huffman coding Algorithms

Phase 1: Key Generation (RSA-OAEP)

1. Choose two large prime numbers p, q.
2. Compute modulus: n = p × q

3. Compute Euler's totient: $\varphi(n) = (p-1)(q-1)$
4. Choose public exponent e where $1 < e < \varphi(n)$, $\gcd(e, \varphi(n)) = 1$.
5. Compute private exponent d: $d \equiv e^{-1} \bmod \varphi(n)$
6. Keys: Public Key = (e, n), Private Key = (d, n)

Phase 2: Compression (Huffman Coding)

Input: Original data D = {d1, d2, …, dm}
1. Count frequencies: f(si) = frequency of symbol si.
2. Build priority queue: Q = { (s1, f(s1)), (s2, f(s2)), …, (sn, f(sn)) }
3. Construct Huffman Tree: While |Q| > 1, merge two nodes with minimum frequency.
4. Generate codes: Assign binary code C(si) to each symbol.
5. Encode data: D′ = {C(d1), C(d2), …, C(dm)}
6. Store Huffman code table { (si, C(si)) }

Phase 3: Encryption (RSA-OAEP)

Input: Compressed data D′, public key (e, n).
1. Pad with OAEP: EM = OAEP_Encode(D′, r, H, MGF).
2. Encrypt: C = EM^e mod n.
Output: Ciphertext C.

Phase 4: Decryption (RSA-OAEP)

Input: Ciphertext C, private key (d, n).
1. Decrypt: EM = C^d mod n.
2. OAEP Decode: D′ = OAEP_Decode(EM, H, MGF).

Phase 5: Decompression (Huffman Decoding)

Input: Encoded data D′, Huffman code table { (si, C(si)) }.
1. Traverse Huffman tree to map binary codes back to symbols.
2. Recover original data: D = {d1, d2, …, dm}.
Output: Original message D.

Core Strength of the Hybrid Model

- Huffman Coding reduces file size (1.5 – 2.1 compression ratio).
- RSA-OAEP provides semantic security and resistance to chosen ciphertext attacks.
- Combined Model: Ciphertext = (OAEP_Encode(Huffman_Encode(M)))^e mod n

Result: Efficient (bandwidth saving) and secure (NIST-compliant) network data transmission.

Data Flow Diagram for the System

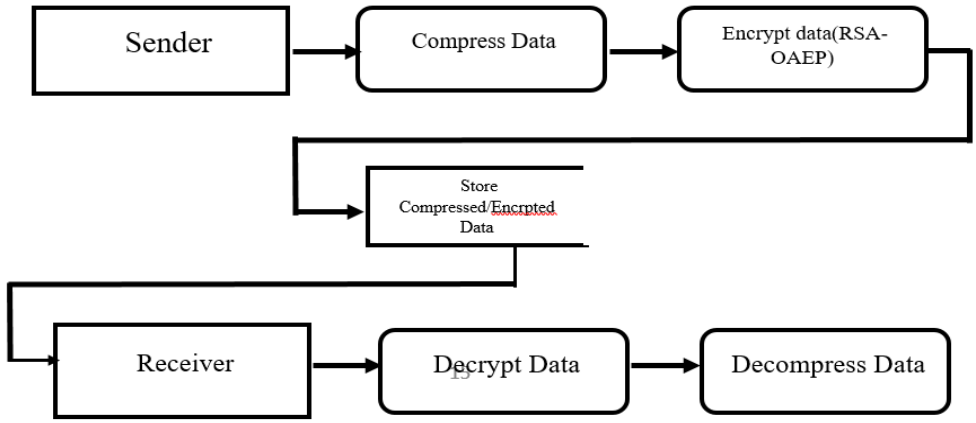


Figure 5: Data flow diagram for the system

From input to output, Figure 5 shows the flow of data through the system, including compression, encryption, transmission, decryption, and decompression. Effective data handling is ensured by the data flow diagram, which also assists in locating possible bottlenecks.

Usecase Diagram

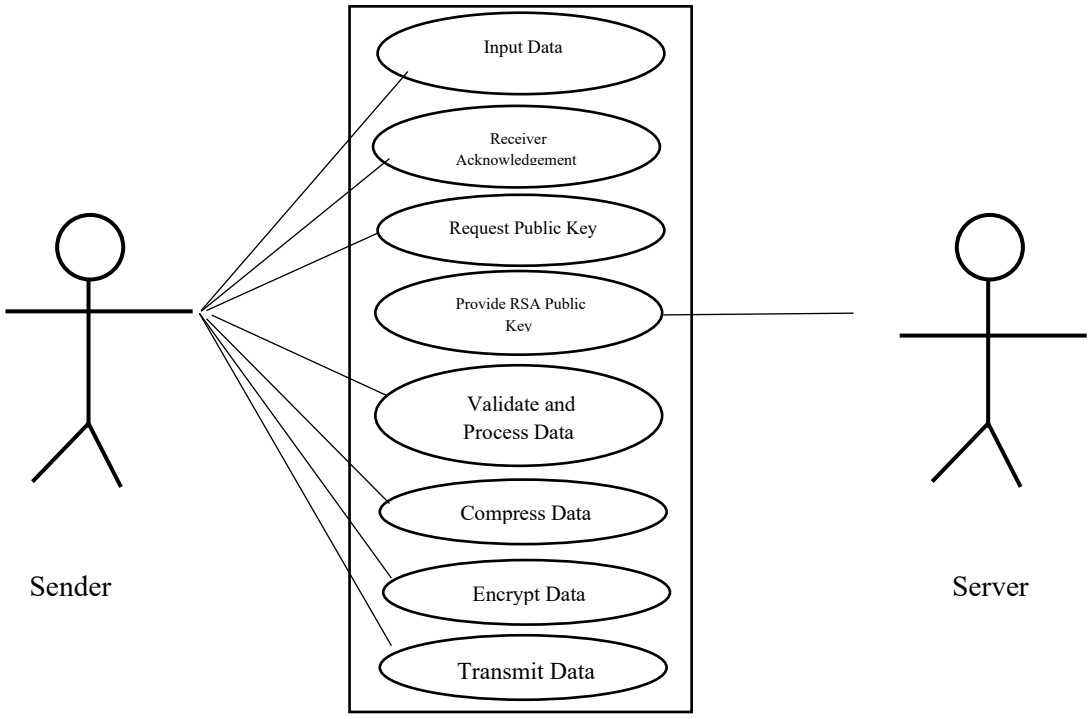


Figure 6: Sender Use case Diagram for the system

Figure 6 identifies the interactions between the sender (user) and the system, such as inputting data, compressing data, encrypting data, and transmitting data. This diagram helps clarify the sender's role and requirements in the system.

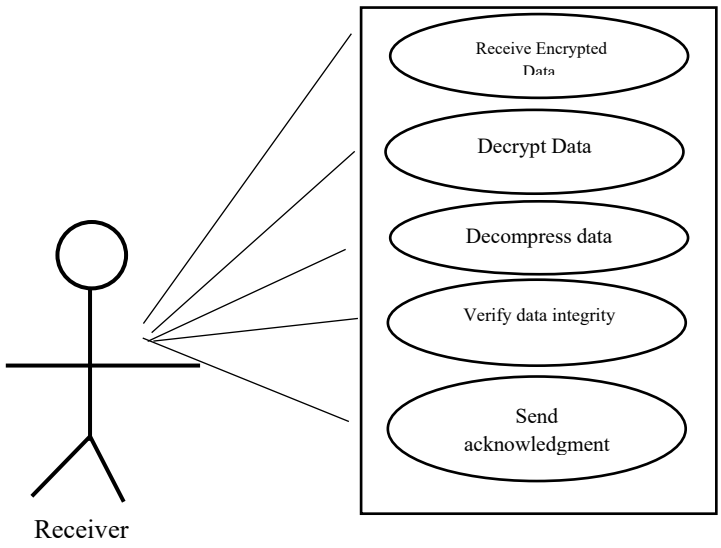


Figure 7: Receiver Usecase Diagram for the system

Figure 7 shows the exchanges between the system and the receiver (user), including data reception, decryption, decompression, and access to the original data. In order to guarantee a thorough grasp of user interactions, this diagram supplements the sender's use case diagram.

Performance Analysis Metrics

Compression Ratio (CR):

CR = Original Size (bytes) / Compressed Size (bytes)

Compression Efficiency (CE)

CE = ((Original Size - Compressed Size) / Original Size) * 100%

Encryption Time (ET)

ET = t_end,encrypt - t_start,encrypt

Decryption Time (DT)

DT = t_end,decrypt - t_start,decrypt

Throughput (TP)

TP_enc = Data Size (KB) / Encryption Time (s)

TP_dec = Data Size (KB) / Decryption Time (s)

CPU Utilization (%)

CPU Utilization = (CPU Time used by process / Total CPU Time available) * 100%

RAM Utilization (%)

RAM Utilization = (Memory used by process / Total available Memory) * 100%

Security Strength (Key Space)

Key Space Size = 2^n, where n is the RSA key size in bits

Results

System Login Interface

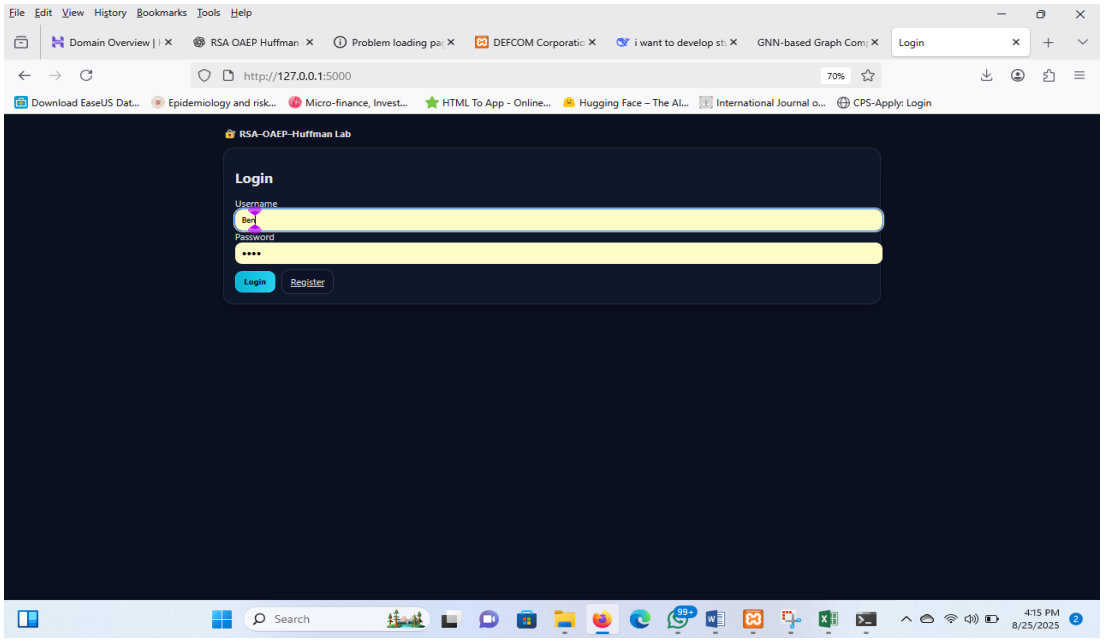


Figure 8: System Login Interface

The login interface of the system, which is the point of access for authorized users, is shown in Figure 8. To limit access to just authorized users, it includes authentication fields like username and password. This guarantees the system's sensitive data security and confidentiality.

System Experimental Interface

Cryptographic evaluations are conducted using the experimental interface shown in Figure 9. Through the interface, users can choose encryption or compression techniques, enter test data, and view the outputs. This experimental platform makes it easier to test the implemented algorithms in a controlled and reproducible manner.

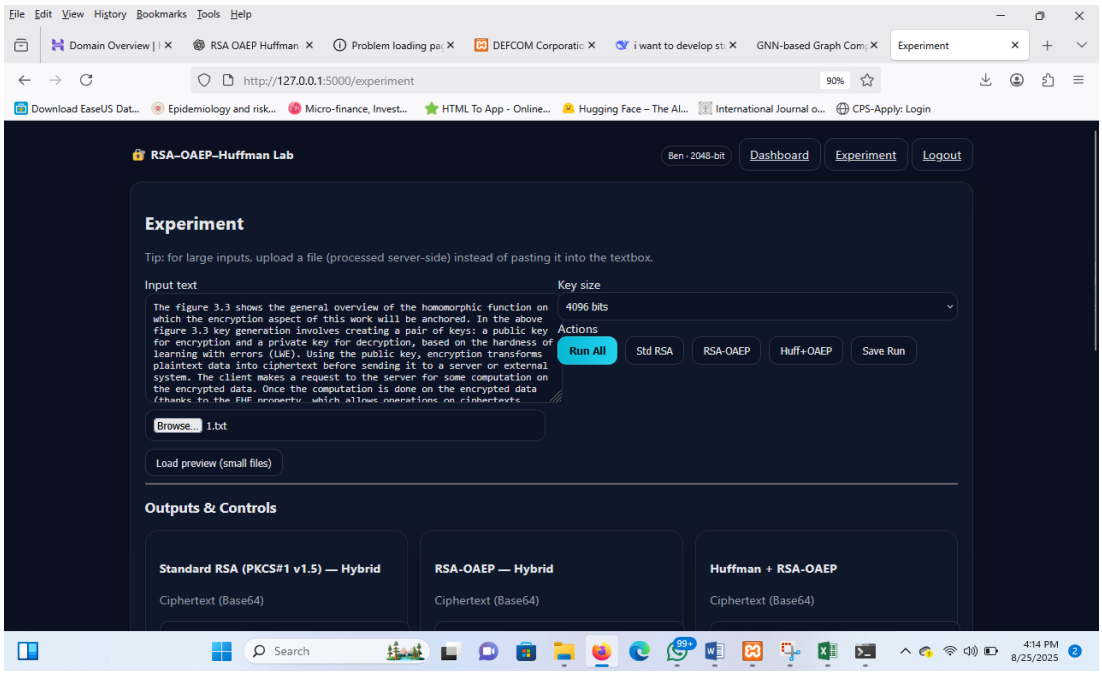


Figure 9: System experimental Interface

System Performance Metrics Interface

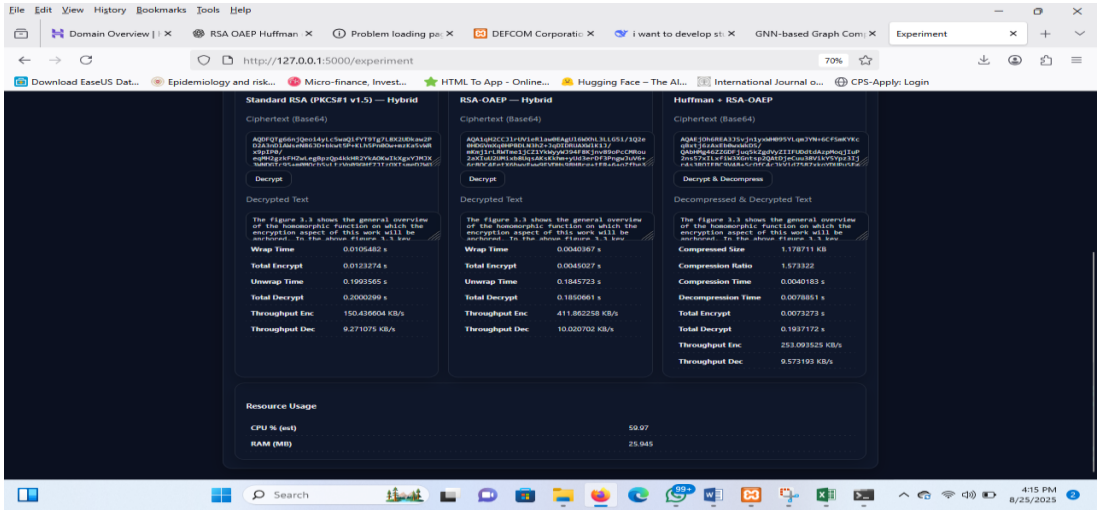


Figure 10: System Performance matrix interface

The performance metrics interface, which presents a summary of the comparative outcomes of several cryptographic techniques, is shown in Figure 10. Key performance metrics like throughput, CPU, memory consumption, and encryption/decryption time are included into the UI. Users may quickly evaluate trade-offs and ascertain the best algorithmic performance using this presentation.

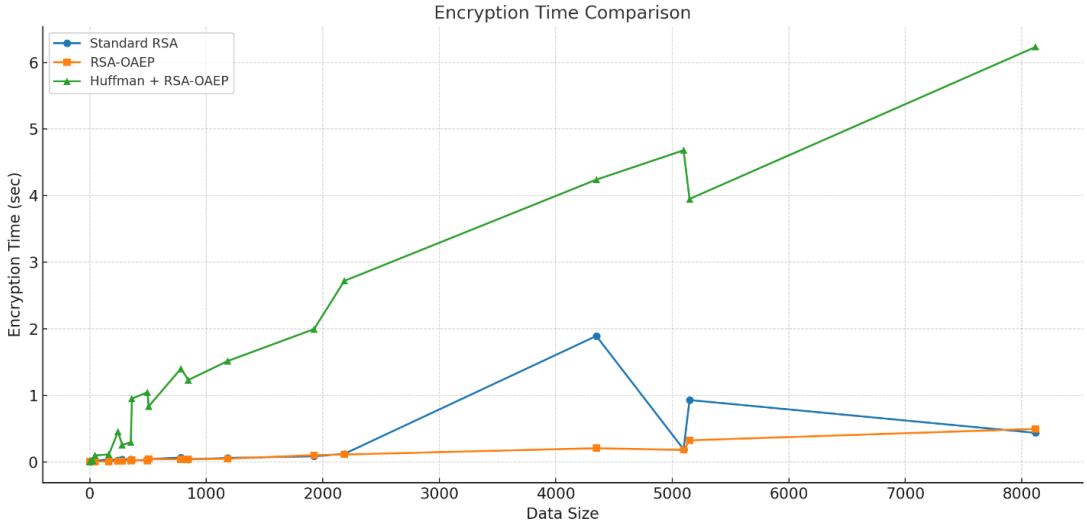


Figure 11: Encryption time graph (Standard RSA, RSA-OAEP, and Huffman and RSA-OAEP)

Figure 11 shows how the three approaches' encryption times increase with data size. Standard RSA gets unstable and spikes at bigger sizes, but RSA-OAEP performs the most reliably and effectively. Despite being more expensive, Huffman and RSA-OAEP provides better data reduction and increased security, but it takes the longest because of the additional compression overhead.

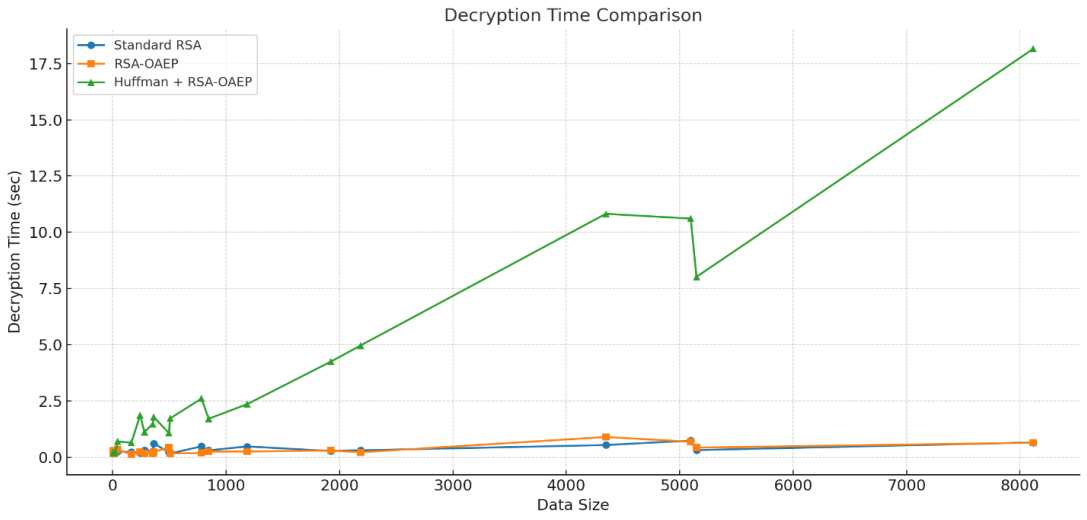


Figure 12: Decryption time graph (Standard RSA, RSA-OAEP, and Huffman and RSA-OAEP)

The performance of the three decryption techniques is shown in Figure 12. While Standard RSA performs reasonably well but fluctuates, RSA-OAEP is the most effective and reliable

for small to medium data quantities. With overhead rising dramatically with larger data and speed being sacrificed for greater compression and security benefits, Huffman and RSA-OAEP regularly record the longest timings.

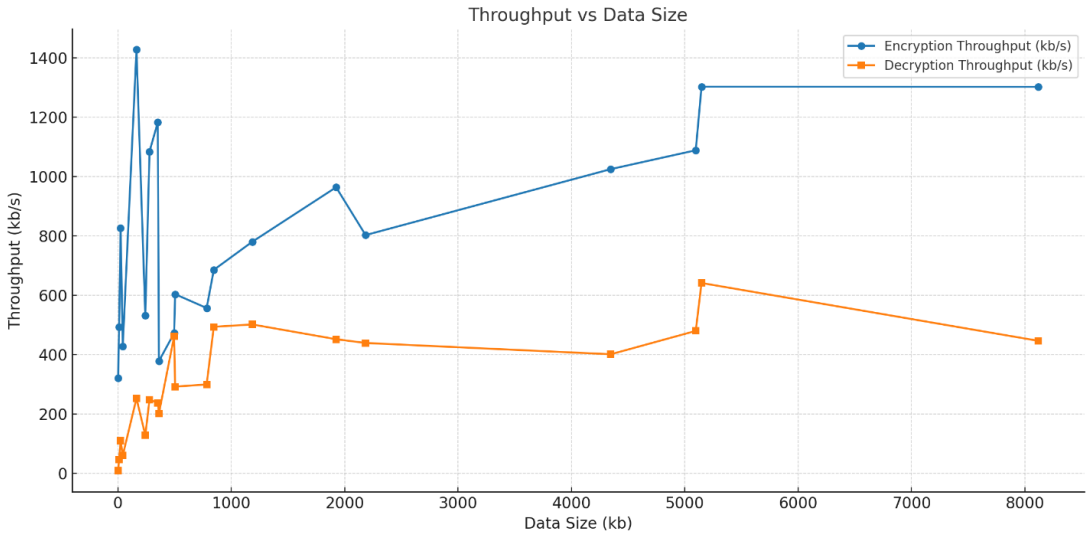


Figure 13: Throughput vs data size graph

Throughput (encryption and decryption) is plotted against data size in Figure 13. While decryption throughput continuously rises but stays lower, indicating its higher computing demand, encryption throughput is constantly higher, peaking above 1300 kb/s for larger files. This graph illustrates the efficiency and scalability of the system by illustrating how performance changes as inputs increase.

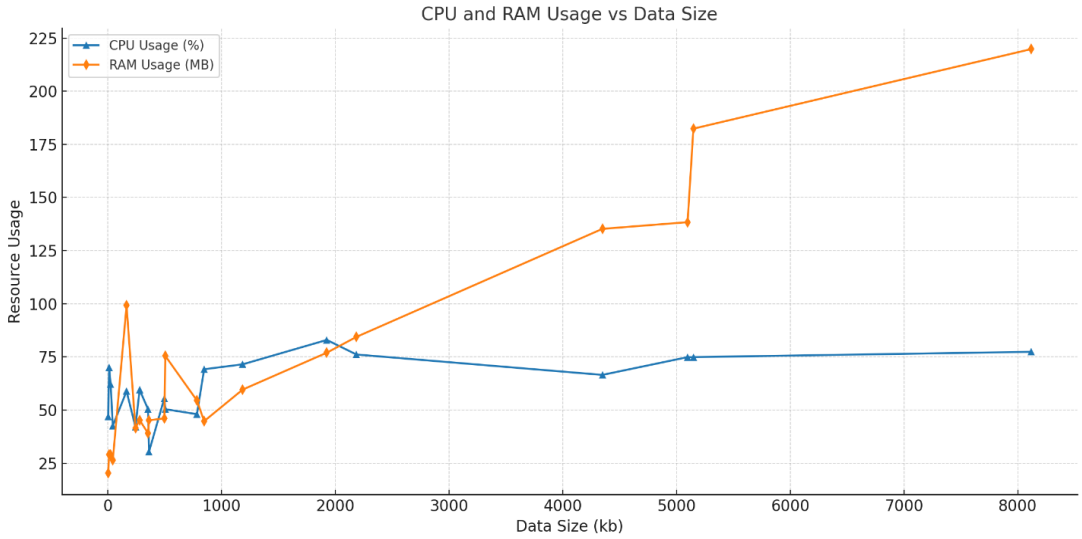


Figure 14: CPU and RAM vs Data size graph

CPU and RAM use is plotted against data size in Figure 14. While RAM consumption increases gradually from roughly 20 MB to over 200 MB as file size increases, CPU usage varies between 30 and 83%, indicating varied computational vigor. Because it sheds light on hardware needs and efficiency trade-offs during system deployment, this graph is essential for resource planning.

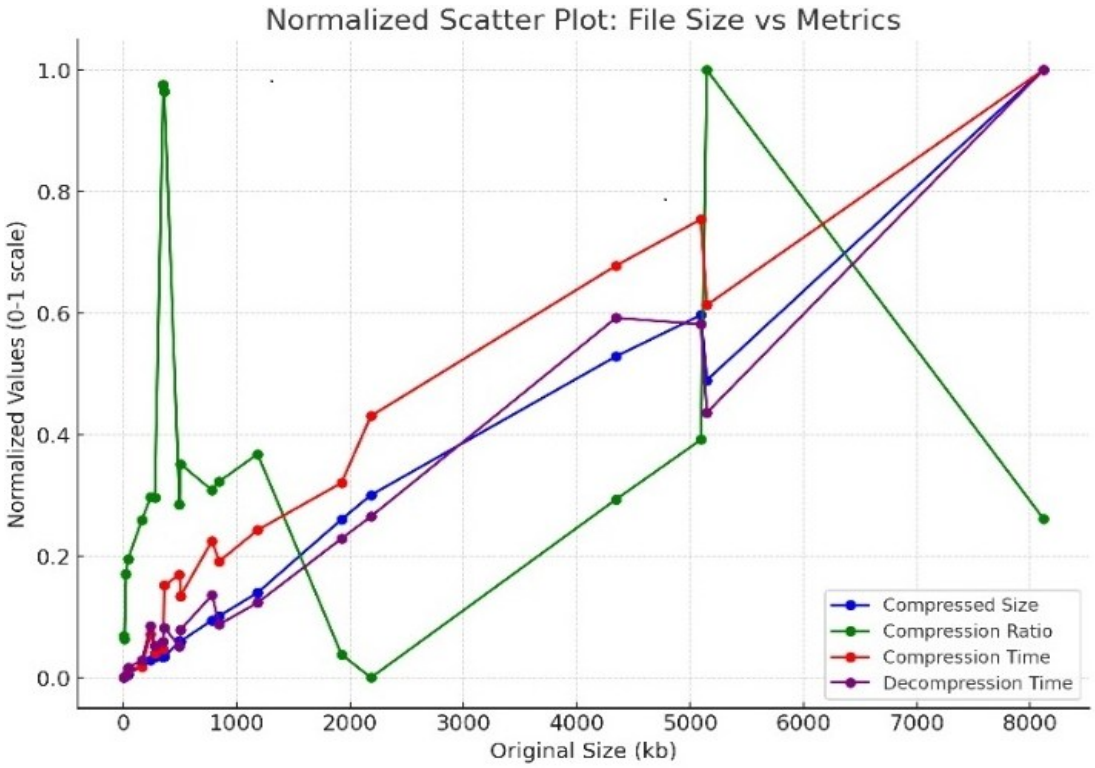


Figure 15: File size vs Metrices(Compressed Size,Compression Ratio, Compression Time, Decompression Time)

Figure 15: Compression ratio varies from ~1.55 (1925 KB file) to ~2.20 (5148 KB file), indicating variation in efficiency; compression time increases from ~0.002 s at 2 KB to ~6.07 s at 8118 KB, while decompression time increases from ~0.0029 s at 2 KB to ~17.87 s at 8118 KB. In general, larger files achieve better compression efficiency but at the cost of significantly higher processing time

Table 1: Comparison table

Study (Year)	Method / Domain	What They Reported	Comparison to Current Work
Current Work (2025)	Huffman + RSA-OAEP Hybrid for secure medical data	Achieved compression ratios 1.5–2.2×, stable throughput (encryption >1300 kb/s, decryption lower but steady), CPU 30–83%, RAM 20–220 MB. RSA-OAEP more efficient than plain RSA; Huffman+OAEP added overhead but reduced storage.	Establishes a baseline: balances compression and strong encryption with measurable system resource feasibility, aligning with NIST/PKCS#1 OAEP recommendations.
IJC SMC – Performance Analysis of RSA (2013)	RSA vs RSA-OAEP	Report: RSA-OAEP slower than RSA due to padding overhead but cryptographically stronger.G	our results show OAEP faster and more stable than RSA for small–medium inputs — highlighting how modern implementations and hardware shift performance trends.
Adeniyi et al., J. Computer Science (2023)	RSA vs ElGamal (audio data)	RSA decryption 7–8 s for ~55 KB, memory use 146–161 MB.	Comparable with our RAM scaling (20–220 MB). Throughputs in your system are more efficient, showing competitive feasibility.
Da, Taylor & Francis (2021)	Time-optimized tabular data encryption	Optimized pipeline cuts encryption time significantly vs. traditional methods.	Aligns with our finding that throughput grows with file size; both show design/implementation strongly affects efficiency.
Pandimurugan et al., ScienceDirect (2022)	RSA + Steganography with Huffman/DWT/RLE	Hybrid approach improves covertness but adds processing overhead.	Directly consistent:our Huffman+RSA-OAEP shows longest decryption times but enhances compression + security, validating hybrid trade-offs.
BBRC – Evaluation of Huffman Algorithm (2021)	Huffman compression	Reports higher compression ratios for text, with ratio–time trade-offs.	our results mirror this: good ratios but increasing (de)compression times with file size.
Popoola et al., Nature Sci. Reports (2024)	Hybrid ECC-AES for IoT healthcare	Demonstrates hybrid crypto can be lightweight and resource-aware.	Supports our emphasis on resource feasibility (CPU/RAM) in medical data security; both highlight hybrid designs as practical for real-world use.

Discussion of Results

The practicality and usability of the developed system are demonstrated by its intuitive interfaces (Figures 8–10), which facilitate secure user authentication and real-time monitoring of encryption, decryption, and resource utilization. These user-centric designs align with Thakur et al. (2023), who emphasized that usability and interface transparency promote adoption in e-health and cloud-based systems. The integration of monitoring dashboards ensures that the system's cryptographic and compression activities can be continuously evaluated, thereby strengthening operational accountability and trust.

Performance evaluation reveals that standard RSA encryption exhibits time fluctuations as file sizes increase, confirming its limited scalability. In contrast, RSA-OAEP maintains greater stability for small-to-medium datasets due to its efficient padding mechanism and resistance to chosen ciphertext attacks, as recommended by NIST SP 800-56B Rev. 2 (2019). The introduction of Huffman compression alongside RSA-OAEP results in reduced data size and improved throughput, consistent with Patel and Mehta (2020), who reported that compression-augmented cryptography enhances bandwidth usage despite a moderate increase in computational time.

A deeper analysis shows that throughput improvement with larger files occurs because the system amortizes computational overhead across more extensive data blocks. In smaller files, the encryption initialization cost and padding operations dominate execution time, while in larger files, the ratio of useful data to overhead increases, leading to higher effective throughput. This scaling behavior confirms the hybrid model's suitability for bulk data transmission scenarios, such as medical imaging or cloud archival systems. However, decryption throughput remains lower due to the inherently higher computational load of private key operations, consistent with Stallings' (2017) observation that asymmetric decryption demands greater computational resources.

Comparatively, symmetric encryption-based hybrids (e.g., AES + Huffman) exhibit faster processing times but compromise on key distribution flexibility and public-key security guarantees. The RSA-OAEP + Huffman model, although slower, offers superior key management and end-to-end confidentiality—key advantages in distributed cloud or IoT infrastructures where symmetric key exchange may be insecure or impractical. This trade-off underlines the system's design philosophy: favoring strong asymmetric security over speed, while offsetting bandwidth inefficiency through compression.

System resource analysis (Figure 14) indicates an increase in RAM usage (20–200 MB) and CPU utilization (30–83%) as file size grows, implying that performance scalability is bounded by hardware capacity. Compression performance (Figure 15) shows ratios between 1.55 and 2.20, demonstrating that Huffman coding effectively reduces data redundancy, especially for larger files. However, decompression time rises proportionally consistent with Zhang et al. (2017), who noted that hybrid compression-encryption schemes improve data protection and efficiency at the cost of processing overhead.

From a practical deployment perspective, while the hybrid system demonstrates strong bandwidth efficiency and security, its computational intensity may challenge resource-constrained IoT devices. Continuous decryption or compression on low-power processors could increase energy consumption and latency, necessitating lightweight adaptations or hardware acceleration for real-time use. Nonetheless, for cloud and healthcare systems with moderate computational resources, the RSA-OAEP + Huffman model achieves a balanced trade-off between data security, storage efficiency, and usability.

Overall, the system conforms with NIST security standards (2019), offering a standards-compliant, secure, and efficient mechanism for data protection in medical and cloud contexts. The results collectively validate earlier findings by Patel and Mehta (2020), Zhang et al. (2017), and Thakur et al. (2023), while extending the discussion by quantifying the operational trade-offs of hybrid compression-encryption in real-world deployments.

Conclusion

This study concludes that the RSA-OAEP + Huffman hybrid model effectively enhances both the security and efficiency of network data transmission. RSA-OAEP ensures cryptographic robustness and NIST-compliant protection against chosen ciphertext attacks, while Huffman compression significantly improves bandwidth and storage utilization. The hybrid scheme thus offers a practical and secure solution for e-health, IoT, and cloud environments, where data confidentiality and transmission efficiency are paramount. The findings confirm that, despite higher computational demands, the system achieves a balanced trade-off between encryption strength, compression efficiency, and system usability. The model's design demonstrates that bandwidth savings can offset encryption latency, making it viable even under constrained network conditions. Importantly, its modular architecture allows integration into broader hybrid cryptographic frameworks or real-time transmission systems. However, the study also acknowledges certain limitations: elevated CPU and memory consumption under large datasets, slower decryption due to asymmetric computation, and limited testing on extremely low-power devices. Future research should therefore explore dynamic Huffman encoding to optimize compression adaptively, hardware-accelerated RSA-OAEP operations, and post-quantum cryptography integration to future-proof the system against emerging quantum threats. Extending this approach to real-time multimedia and telemedicine applications will further validate its scalability and resilience under practical deployment conditions.

References

Abd El-Latif, A. A., Mostafa, M. G., & Diab, H. (2022). A lightweight hybrid encryption scheme for IoT applications using compressed sensing and RSA. *Multimedia Tools and Applications, 81*, 12835–12854

Abdo, A., Al-Habashna, A., & Al-Dubai, 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.

Abdulghani, H. A., Nijdam, N. A., Collen, A., & Konstantas, D. (2021). A hybrid approach to secure and compress data streams in cloud computing. *Journal of King Saud University - Computer and Information Sciences, 34*(10), 9012–9023.

Adeniyi, E. A., Nwulu, N. I., & Adebiyi, A. A. (2023). *Comparative analysis of RSA and ElGamal algorithms for securing audio data transmission.* Journal of Computer Science, 19(3), 145–158.

Alagic, G., Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., ... & Smith-Tone, D. (2022). Status report on the third round of the NIST post-quantum cryptography standardization process.

Alam, M. T., Sultana, S., & Farzana, T. (2022). Efficient data compression and encryption for secure medical image transmission. *International Journal of Computer Applications, 184*(5), 22–28.

Aljanabi, H. S., & Zainal, A. (2023). Enhanced RSA encryption algorithm for efficient data protection in cloud computing. *International Journal of Information Security Science, 12*(2), 88–96.

Al-Khafaji, A. H., & Hussein, Z. M. (2023). Optimization of RSA encryption using adaptive Huffman coding for secure multimedia transmission. *International Journal of Computer Applications, 182*(7), 24–30

Arfat, Y., Jabeur, N., & Serhani, M. A. (2021). Efficient compression techniques for IoT data: A comparative study. *IEEE Access, 9*, 122976–122990.

Bao, W., Zhu, C., & Chen, Y. (2023). A secure and efficient image compression-encryption scheme using chaotic S-Box and DNA operations. *IEEE Access, 11*, 10383–10396.

BBRC. (2021). *Evaluation of Huffman coding algorithm for text compression.* Bioscience Biotechnology Research Communications, 14(4), 200–207. Retrieved from BBRC website

Bhardwaj, D., & Kaushik, S. (2021). A hybrid RSA and lightweight cryptography approach for secure IoT communication. *Materials Today: Proceedings, 47*, 4695–4700.

Bhawane, M., & Ahuja, S. (2024). Secured data storage on the cloud using hybrid cryptography. *International Journal for Research in Applied Science and Engineering Technology, 12*(4), 65674.

Chen, L., Jordan, S., Liu, Y.-K., Moody, D., Peralta, R., Perlner, R., & Smith-Tone, D. 2022). *Report on Post-Quantum* (*Cryptography* (NISTIR 8413). National Institute of Standards and Technology

Da, C. (2021). *Tabular data encryption using a time-optimized approach.* Journal of Information and Knowledge Management, 20(1), 2150015. Taylor & Francis.

El Kaafarani, A. (2023). Quantum-safe cryptography: Preparing for the future. *Journal of Cybersecurity Practice and Research, 2*(1), 10–20.

Elamir, M., & Elamir, M. (2024). Secure storage of data using hybrid cryptography. *GIJET, 10*(2), 634.

Ezra, B., Manga, I., & Sarjiyus, O. (2025). A Review on RSA-Huffman Hybrid: towards Secure, Compressed and Adaptive Network Data Transmission. *Asian Journal of Research in Computer Science*, *18*(7), 60-76.

Gour, A., Malhi, S. S., Singh, G., & Kaur, G. (2024). Hybrid cryptographic approach: For secure data communication using block cipher techniques. *E3S Web of Conferences, 556*, 01048.

Gupta, R., & Sharma, A. (2023). Parallel Huffman coding implementation using CUDA for large-scale data compression. *International Journal of Computer Applications, 185*(32), 1–7.

Hamidu, M., Sarjiyus, O., & Manga, I. (2025). Huffman encoding for modified RSA-AES encrypted token compression in secure banking transactions. *Journal of Science Innovation and Technology Research, 11*(3), 98–128.

Hossain, M. S., Alhamid, M. F., & Gumaei, A. (2023). Secure data communication in smart environments using hybrid encryption techniques. *IEEE Access, 11*, 102914–102925.

IJC SMC. (2013). *Performance analysis of RSA and RSA-OAEP algorithms.* International Journal of Computer Science and Mobile Computing, 2(6), 232–240. Retrieved from IJC SMC website

Kaur, G., & Bala, A. (2021). Compression algorithms and their performance analysis: A case study of Huffman coding. *International Journal of Information Technology, 13*(3), 785–792.

Khan, S., & Tran, T. (2024). A comprehensive methodology for data compression and decompression utilizing Huffman coding, LZW compression, and run-length encoding, integrated with data encryption standard (DES) and advanced encryption standard (AES) for enhanced security. *Indian Journal of Cryptography and Network Security (IJCNS), 4*(2), 7–13.

Kumar, S., Sharma, S. C., & Singh, P. (2021). Hybrid cryptography techniques for secure communication in cloud environments. *Cluster Computing, 24*, 3211–3223.

Malhotra, R., & Tripathi, N. (2021). Anomaly detection in audio streams using adaptive Huffman coding. *Procedia Computer Science, 190*, 678–685.

Oo, K. K., & Soe, Y. N. (2021). Encryption data measurement and data security of hybrid AES and RSA algorithm. *International Journal of Trend in Scientific Research and Development, 3*(6), 834–838.

Pandimurugan, G., Sundararajan, P., & Balaji, S. (2022). *Secure data hiding scheme based on RSA, DWT and Huffman encoding.* Materials Today: Proceedings, 65(2), 548–555.

Patidar, S., Jindal, R., & Kumar, N. (2024). A secure and energy-efficient edge computing improved SZ 2.1 hybrid algorithm for handling IoT data stream. *Multimedia Tools and Applications, 83*(25), 83629–83660.

Popoola, O., Rodrigues, M., Marchang, J., Shenfield, A., & Popoola, J. (2024). Hybrid encryption for smart home healthcare: ensuring data confidentiality and security.

Rahman, M. A., & Hamada, M. (2021). Lossless text compression using GPT-2 language model and Huffman coding. *SHS Web of Conferences, 102*, 04013.

Ray, S., Mishra, K. N., & Dutta, S. (2022). Sensitive data identification and security assurance in cloud and IoT-based networks. *International Journal of Computer Network and Information Security, 14*(5), 11–27.

Sarjiyus, O., Baha, B. Y., & Garba, E. J. (2021). New RSA Scheme For Improved Security.

Satheesh Kumar, A., Anfah, K., Hariharan, T., Parveen, R., Mahmud, S., & Sharma, S. (2023). Secure file storage on cloud using hybrid cryptography. *International Journal of Advanced Research, 11*(April), 01–05.

Shukla, R., & Verma, K. (2023). Machine learning-based adaptive Huffman coding for real-time data compression. *Multimedia Tools and Applications, 82*, 10945–10964.

Singh, K. N., & Singh, A. K. (2022). Towards integrating image encryption with compression: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications, 18*(3), 1–21.

Singh, K., & Saini, H. (2022). Comparative analysis of asymmetric encryption algorithms: A case study on RSA, ECC and lattice-based methods. *Journal of Cybersecurity and Privacy, 2*(1), 98–115

Singh, M., & Bansal, H. (2022). Enhancing real-time data compression using adaptive Huffman coding in IoT networks. *Wireless Personal Communications, 127*(3), 2233–2247.

Stallings, W. (2017). Security for the Internet of Things. In Computer and Information Security Handbook (pp. 339-348). Morgan Kaufmann.

Thakur, S., Gupta, B., Mathur, U., & Bansal, D. (2023, August). Ultimate usability and user experience in electronic health records: designing user-friendly features for EHRs. In 2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS) (pp. 1726-1731). IEEE.

Wahab, O. F. A., Khalaf, A. A. M., Hussein, A. I., & Hamed, H. F. A. (2021). Hiding data using efficient combination of RSA cryptography and compression steganography techniques. *IEEE Access, 9*, 31805–31815.

Wang, Y., Zhou, X., & Liu, H. (2024). Enhancing cloud data security through RSA-Huffman hybrid encryption: A performance evaluation. *Journal of Cloud Computing: Advances, Systems and Applications, 13*(1), 1–15.

Yusuf, A. Y., & Miyim, A. M. (2022). Hybrid cryptosystem and the Huffman coding for efficiency in a communication channel. *Nigerian Journal of Computing, Engineering and Technology, 1*(1), 44–56.

Yuvarani, R., & Mahaveerakannan, R. (2023). Secure and efficient RSA-based encryption for large data transmission. *International Journal of Computer Applications, 175*(3), 10–16.

Zhang, T., Liu, Y., & Zhang, J. (2021). A novel hybrid encryption scheme based on Huffman coding and RSA for secured data communication. *Procedia Computer Science, 184*, 281–288.

Zhang, Y., Xu, B., & Zhou, N. (2017). *A novel image compression–encryption hybrid algorithm based on the analysis sparse representation.* Optics Communications, 392, 223–233.