Periodico di Mineralogia Volume 95, No.3, 2026
ISSN: 0369-8963

SECURE NETWORK MESSENGER USING SSH FOR IMPROVED ANONYMITY

A.Neela Madheswari[1], K.Aloysius Rosario[2], M.Arunkumar[3], S.Aasif Hameed[4] and P.Anbarasu [5]

[1]CSE Department, Mahendra Engineering College, Namakkal, India
[2,3,4,5]Cyber Security Department, Mahendra Engineering College, Namakkal, India

ABSTRACT

Secure communication systems are essential in modern digital environments where privacy and confidentiality are critical. This paper presents the design of secure end to end network messenger that utilizes SSH based authentication and RSA key exchange for secure identity verification. After authentication, symmetric encryption algorithms including AES, Triple DES, RC4, Blowfish, and ChaCha20 are implemented for message confidentiality. The system also benchmarks the average encryption and decryption time for each algorithm to evaluate performance trade-offs. Experimental results demonstrate that modern algorithms such as AES and ChaCha20 provide better performance and security compared to legacy algorithms. The proposed system ensures secure communication while allowing comparative analysis of symmetric encryption efficiency.

KEYWORDS

Secure messaging, SSH authentication, RSA, DES, Symmetric encryption, Encryption benchmarking, Network security

1. INTRODUCTION

Secure communication over networks is a fundamental requirement in distributed systems. Traditional messaging systems often rely on centralized encryption mechanisms that may expose vulnerabilities. End to end encryption ensures that only communicating users can access the content of messages. This paper proposes a secure network messenger built using SSH based authentication using Paramiko library. RSA is used for secure key exchange, and multiple symmetric encryption algorithms are implemented to encrypt message payloads. The system also evaluates encryption performance by measuring average encryption and decryption times for different symmetric algorithms. The primary goal of this work is to analyse security strength and computational performance of commonly used symmetric encryption algorithms within a real time messaging system.

2. LITERATURE REVIEW

Secure communication systems require the integration of cryptographic mechanisms, authentication protocols, secure key exchange techniques, and protected message transmission channels. Several recent studies have addressed these components individually. This section reviews relevant work grouped into cryptographic security, authentication frameworks, secure transport mechanisms, secure messaging implementations, and forensic analysis.

2.1. Cryptographic mechanisms and Secure communication

Cryptographic protection forms the foundation of secure digital communication system. Kumar et al [1] proposed a comprehensive end-to-end web security solution integrating cryptography,

 https://doi.org/10.5281/zenodo.20313243

multi factor authentication, and secure communication protocols. Their framework ensure confidentiality and secure user verification, however, it primarily targets web based architectures and does not specifically focus on secure messaging transport mechanisms such as SSH.

Ramakrishna and Shaik [2] conducted a broad evaluation of cryptographic algorithms, comparing their security strength, computational efficiency, and future research challenges. Their comparative study provides essential guidance for selecting appropriate symmetric encryption algorithms in secure communication systems, though it does not present a unified messaging implementation.

Alam et al [3] implemented the ChaCha20 algorithm combined with Elliptic Curve Diffie Hellman (ECDH) to present Man-in-the-Middle (MITM) and reused key attacks. Their hybrid encryption model demonstrates the effectiveness of combining asymmetric key exchange with symmetric encryption for secure data transmission. However, the system is limited to a server monitoring context rather than a generalized secure messaging platform.

Tran et al [4] performed a formal analysis of a post-quantum hybrid key exchange mechanism within the SSH transport layer protocol. Their study strengthens key exchange resilience against emerging quantum threats but focuses primarily on theoretical validation rather than full messaging system implementation and performance benchmarking. Collectively, these studies reinforce the importance of encryption strength and secure key exchange, yet they largely treat cryptographic components independently rather than integrating them within a complete secure messaging architecture.

Liu et al [5] proposed enhancements to the Advanced Encryption Standard (AES) by optimizing key operations, demonstrating improved security resilience against cryptanalytic attacks. Similarly, Subaselvi et al. [6] explored hardware-level implementation of Triple-DES, highlighting performance–security trade-offs in block cipher deployment. These works validate the importance of robust symmetric encryption, which forms the foundation of the proposed hybrid encryption model.

Secure transaction frameworks integrating multi-factor authentication (MFA) and machine learning techniques have been studied extensively in financial and cloud environments [7], [8], [9]. Yusop et al. [10] further advanced password less authentication mechanisms, emphasizing the need to mitigate traditional credential-based vulnerabilities. While these systems enhance user authentication, they often rely heavily on centralized trust models. In contrast, the proposed system integrates authentication with end-to-end hybrid encryption, minimizing exposure of plaintext even to intermediate servers.

Transport layer security mechanisms such as SSL/TLS remain critical for secure data transmission [11], though vulnerabilities in their implementation have been documented [12]. These studies justify the dual-layer security approach adopted in this work, where SSH-based transport security is combined with application-layer encryption to defend against both passive and active attacks [13].

Emerging threats such as proactive eavesdropping [14] and quantum-computing risks [15] highlight the evolving landscape of communication security. Cloud security challenges [16] and fintech vulnerabilities [17] further reinforce the need for adaptable encryption frameworks capable of dynamic key exchange and secure relay. Additionally, authentication enhancements using wearable devices and smart environments [18], [19] demonstrate the increasing integration of cryptographic systems into real-world applications.

 https://doi.org/10.5281/zenodo.20313243

Unlike existing systems that rely solely on either transport security or symmetric encryption, the proposed architecture employs a hybrid encryption model combining RSA-based key encapsulation with symmetric encryption algorithms (AES, Triple-DES, Blowfish, RC4, and ChaCha20), ensuring confidentiality, scalability, and secure key distribution across clients.

2.2. Authentication and multi factor security models

Djidjekh et al [20] proposed a lightweight protocol-agnostic security enhancement for SWIPT-enabled IoT systems. Their design emphasizes efficiency and reduced computational overhead in resource-constrained environments. However, it does not incorporate SSH-based secure communication or comparative cryptographic performance evaluation. These studies improve authentication robustness and efficiency, yet they operate independently of integrated SSH-based secure messaging systems with hybrid encryption benchmarking.

Authentication mechanisms are critical in preventing unauthorized access to communication systems. Iwamura and Kamal [21] proposed a secure user authentication scheme based on information-theoretic security using secret sharing-based secure computation. Their model enhances authentication robustness but introduces additional computational complexity and does not directly integrate with SSH-based transport systems. Sarower et al. [22] introduced SMFA, which strengthens multi-factor authentication using steganography to conceal authentication credentials. While this technique improves resistance against credential-based attacks, it primarily focuses on authentication enhancement without addressing encrypted message transmission performance or secure session establishment.

2.3. Secure Transport and Hybrid Encryption mechanism

Secure transport protocols ensure confidentiality and integrity during data transmission. Tran et al. [4] analyzed hybrid key exchange mechanisms within the SSH protocol, reinforcing the importance of secure transport-layer encryption. Similarly, Alam et al. [3] demonstrated that combining asymmetric and symmetric cryptographic techniques enhances resistance against MITM attacks. Although hybrid encryption approaches are well studied, limited research evaluates their practical performance within real-time secure messaging systems alongside symmetric algorithm benchmarking and packet-level validation.

2.4. Secure messaging and Digital Forensics

Secure instant messaging systems require reliable encryption and authentication mechanisms. Rodrigues et al. [23] proposed securing instant messages using hardware-based cryptography and authentication within a browser extension. Their approach improves message confidentiality and integrity; however, it depends on specialized hardware modules and does not evaluate comparative encryption performance.

From a forensic perspective, Sunardi, Herman, and S. R. Ardiningtias [24] conducted a comparative analysis of digital forensic investigation tools on Facebook Messenger applications. Their findings highlight variations in evidence recovery capabilities across forensic platforms. While the study emphasizes post-incident investigation, it does not address secure-by-design communication systems that minimize forensic exploitability.

2.5. Research gap

Although existing studies address cryptographic security, authentication mechanisms, hybrid encryption techniques, and SSH transport layer analysis, limited research integrates SSH-based secure transport, hybrid encryption implementation, symmetric algorithm performance evaluation, and real-time packet-level validation within a unified secure messaging architecture. This gap motivates the development of the proposed Secure Network Messenger, which

 https://doi.org/10.5281/zenodo.20313243

combines SSH communication, hybrid encryption, algorithm benchmarking, and experimental network analysis to enhance messaging confidentiality and anonymity.

3. System Architecture

The proposed system consists of three major components namely: i) Authentication and session management module, ii) Cryptographic processing module, and iii) Secure communication module. The server authenticates users using SSH password authentication. After successful authentication, public keys are exchanged between clients for secure RSA encryption of symmetric session keys. Messages are then encrypted using the selected symmetric algorithm before transmission.

4. Cryptographic Design

RSA with 2048 bit keys is used for secure key exchange, providing a balance between computational efficiency and cryptographic strength consistent with current security recommendations. Each client generates a public private key pair. The public key is shared via the server, while the private key remains confidential.

4.1. Symmetric encryption algorithms

The system supports the following symmetric algorithms: i) AES, ii) TripleDES, iii) RC4, iv) Blowfish, and v) ChaCha20. AES and ChaCha20 are modern secure algorithms. TripleDES and Blowfish are legacy algorithms included for performance comparison though it is considered insecure for modern systems. Figure 1 specifies the selection of algorithm to use for communication. Messages are encrypted using cipher feedback mode where applicable. For ChaCha20, stream cipher encryption is applied.

Figure 1. Algorithm selection option window

4.2. User Authentication and key generation

When a client starts, the following steps are performed:

a) The user provides authentication credentials such as username and password.

b) The client generates an RSA public-private key pair.

 https://doi.org/10.5281/zenodo.20313243

c) The public key is sent to the server and stored for future communication. The private key remains securely stored on the client side.

d) The server maintains a mapping between usernames and their corresponding public keys. This enables secure key exchange between users.

4.3. A Hybrid Encryption Mechanism

The messaging process uses a hybrid encryption model consisting of the following steps:

a) Symmetric key generation: For every message or file transfer, a unique random symmetric session key is generated dynamically. This ensures session-level confidentiality and prevents key reuse across communications.

b) Message encryption using optimized symmetric algorithm: The plaintext message or file content is encrypted using the selected symmetric algorithm like AES, Triple DES, RC4, Blowfish, or ChaCha20. Based on experimental benchmarking results, AES is used as the default algorithm due to its consistently lower encryption and decryption times across all tested data types. Symmetric encryption is employed for its high computational efficiency when handling large data such as multimedia and document files.

c) Secure Key Encapsulation using RSA: The generated symmetric session key is encrypted using the receiver's RSA public key. This ensures that only the intended recipient, possessing the corresponding RSA private key, can decrypt and retrieve the session key securely.

d) Secure Transmission over SSH: Both the encrypted data (cipher text) and the RSA-encrypted symmetric session key are transmitted through an SSH-protected channel. SSH provides transport-layer encryption, authentication, and protection against network-level attacks such as packet interception and man-in-the-middle attacks.

e) Decryption process and Data recovery: Upon receiving the message the recipient decrypts the symmetric session key using their RSA private key. The decrypted symmetric key is then used to decrypt the actual message. This approach ensures confidentiality while maintaining performance efficiency.

4.4. Units, Abbreviations and Acronyms

The following abbreviations and acronyms are used throughout this paper:

a) 2EE: End-to-End Encrypted

b) SSH : Secure Shell

c) RSA : Rivest-Shamir-Aldman public key cryptographic algorithm

d) AES : Advance Encryption Algorithm

e) CBC : Cyber Block Chaining

f) GUI : Graphical User Interface

g) TCP : Transmission Control Protocol

h) DES : Data Encryption Standard

 https://doi.org/10.5281/zenodo.20313243

When calculating encryption time and decryption time, the standard unit used is seconds (s), which is the SI unit of time. However, because cryptographic operations usually complete very quickly, the measurements are often expressed in smaller units such as milliseconds (ms) or microseconds (μs). For lightweight symmetric algorithms like AES, ChaCha20, Blowfish, or Twofish, encryption and decryption times are typically measured in milliseconds or even microseconds, especially when processing small messages.

5. System Performance Evaluation

To comprehensively evaluate the effectiveness and practicality of the proposed Secure Network Messenger, multiple quantitative performance metrics were analyzed. Since the system integrates SSH-based transport security with application-layer hybrid encryption (RSA + symmetric cryptography), both computational and network-level performance indicators were considered.

The primary evaluation metric is encryption time, which measures the computational duration required to transform plaintext into cipher text using the selected symmetric algorithm. This metric directly reflects the processing overhead introduced during secure message transmission. Similarly, decryption time measures the time required to recover the original plaintext from the cipher text at the receiver side. Together, these metrics determine the feasibility of real-time secure messaging and the responsiveness of the system under different cryptographic configurations.

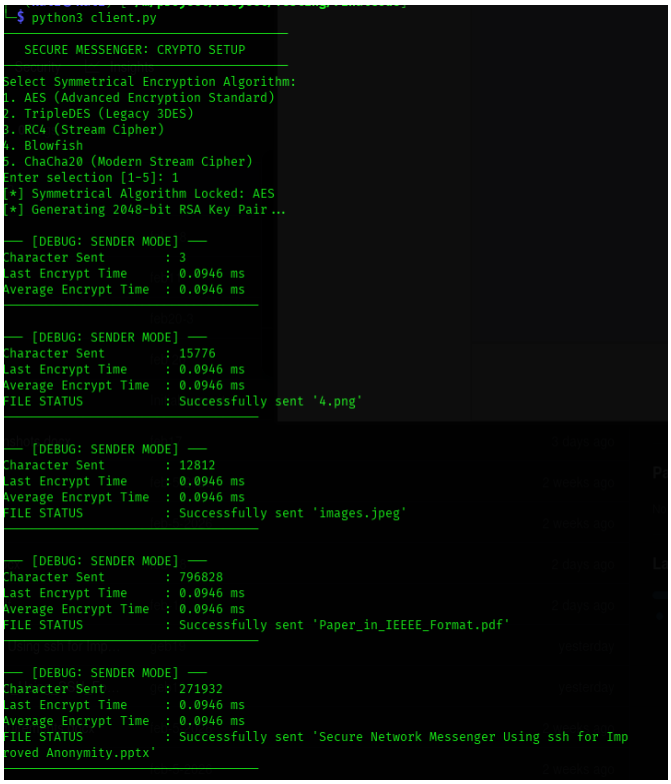

Figure 2. Encryption time calculation display

In addition to symmetric encryption performance, RSA key generation time and RSA key encapsulation time are analysed. Since the proposed system employs hybrid encryption, RSA is used to encrypt the symmetric session key during session establishment. The time required to generate RSA key pairs and encrypt/decrypt session keys represents the asymmetric overhead

 https://doi.org/10.5281/zenodo.20313243

introduced during secure initialization. Evaluating this metric is critical because asymmetric cryptography is computationally more intensive than symmetric algorithms.

Collectively these evaluation metrics provide a comprehensive assessment of computational cost, communication overhead, and practical deployment feasibility. By benchmarking multiple symmetric algorithms (AES, Triple-DES, Blowfish, RC4, and ChaCha20) within the same hybrid encryption framework, the study identifies the most efficient algorithm for secure real-time messaging while maintaining strong cryptographic protection.

When calculating encryption time and decryption time, the standard unit used is seconds (s), which is the SI unit of time. However, because cryptographic operations usually complete very quickly, the measurements are often expressed in smaller units such as milliseconds (ms) or microseconds (μs). One millisecond is equal to 10^{-3} seconds, and one microsecond is equal to 10^{-6} seconds. For lightweight symmetric algorithms like AES, ChaCha20, Blowfish, or Twofish, encryption and decryption times are typically measured in milliseconds or even microseconds, especially when processing small messages.

In practical benchmarking, time is measured using high- resolution timers provided by programming languages. For example, in Python, functions such as time.perf_counter() or time.time() return values in seconds as floating-point numbers. Even though the base unit returned is seconds, the result is usually multiplied by 1000 to convert it into milliseconds for clearer comparison.

Figure 3. Encryption time shown in ms

For accurate benchmarking, especially in cryptographic performance analysis, multiple runs are performed and the average time is calculated. This helps reduce measurement noise caused by system load, CPU scheduling, and background processes. The final reported value is typically the mean encryption or decryption time per message, expressed in seconds, milliseconds, or microseconds depending on the scale of the results.

5.1. Symmetric key generation

For each message, a random session key is generated as follows:

$$Ks=Random(n)$$

where Ks is the symmetric session key, and "n" is the key length in the form of 256 bits for AES, 128 bits for RC4, etc.

5.2. Message encryption using Symmetric encryption

The plain text message "M" is encrypted using a symmetric algorithm given:

$$C=Esym(M,Ks,IV)$$

where "C" is the cipher text, "Esym" is the selected symmetric encryption algorithm, "Ks" is the symmetric session key, "IV" is the initial vector.

 https://doi.org/10.5281/zenodo.20313243

5.3. RSA encryption of Session key

The session key is encrypted using the receiver's RSA public key:

$$Ke=ERSA(Ks,PUreceiver)$$

where *Ke* is the encrypted session key, *PUreceiver* is the receiver's RSA public key.

5.4. Transmission packet structure

The transmitted packet contains the following fields:

$$Packet=(Algorithm, IV, Ke, C)$$

where *Algorithm* specifies any type of encryption algorithm used, *IV* specifies initial vector, *Ke* specifies encrypted session key, and *C* is the Cipher text.

5.5. RSA decryption of session key

The receiver decrypts the symmetric key using their private key as follows:

$$Ks=DRSA(Ke, PRreceiver)$$

where *Ks* is the symmetric session key, *Ke* is the encrypted session key, *PRreceiver* is the receiver's RSA private key.

5.6. Message Decryption

The original message is recovered using the following formula:

$$M=Dsym(C, Ks, IV)$$

where *M* is the recovered original message. *Dsym* is the selected symmetric decryption algorithm, *Ks* is the symmetric session key, *IV* is the initial vector.

5.7. Performance Evaluation parameters

The system is evaluated using encryption time and decryption time, average encryption time and average decryption time. The total encryption time is given as follows:

$$Tenc=tendenc-tstartenc$$

where *Tenc* is the total encryption time, *tstartenc* is the time at which the encryption process begins, and *tendenc* is the time at which the encryption process completes. The encryption time is obtained by subtracting the encryption start time from the encryption end time and it represents the total computational time required to convert plain text into cipher text.

$$Tenc = (\frac{1}{m}) \sum_{k=1}^{m} k^m * Tenck$$

where *Tenc* is the average encryption time, *m* is the total number of encryption operations performed. *Tenck* represents the encryption time of the k^{th} message, and $\sum_{k=1}^{m} k^m$ denotes the summation of encryption times from the first encryption operation to the Nth encryption operation. The average encryption time is calculated by summing all individual encryption times and dividing the total by the number of encryption operations.

 https://doi.org/10.5281/zenodo.20313243

ISSN: 0369-8963

$$Tdec = (\frac{1}{N}) \sum_{i=1}^{N} Tdeci$$

where Tdec is the average decryption time, N is the total number of decryption operations performed. Tdeci represents the decryption time of the ith message, and $\sum_{i=1}^{N}$, is the summation of decryption times from the first decryption operation to the Nth decryption operation. The average decryption time is calculated by summing all individual decryption times and dividing the total by the number of decryption operations.

5.8. Complete Hybrid Encryption Model

The complete hybrid encryption model is represented as follows:

$$C=Esym(M, DRSA(ERSA(Ks, PU), PR), IV)$$

where *C* is the final cipher text produced by the hybrid encryption process, *Esym( )* represents the symmetric encryption function, *M* is the original plaintext message, *DRSA( )* denotes the RSA decryption operation, *ERSA( )* denotes the RSA encryption operation, *Ks* is the symmetric session key, *PU* is the receiver's public key, *PR* is the receiver's private key, and *IV* is the initialization vector used in the symmetric encryption algorithm.

In this hybrid encryption model, the symmetric session key *Ks* is first encrypted using RSA encryption with the receiver's public key *PU* through *ERSA(Ks, PU)*. The encrypted session key is then decrypted using the receiver's private key *PR* through *DRSA(ERSA(Ks, PU), PR)* to securely recover the symmetric key. Finally, the recovered symmetric key is used along with the initialization vector *IV* to perform symmetric encryption of the plaintext message *M*, producing the final cipher text *C*.

This model combines the security of asymmetric encryption (for secure key exchange) with the efficiency of symmetric encryption (for fast message encryption), forming a complete hybrid encryption framework.

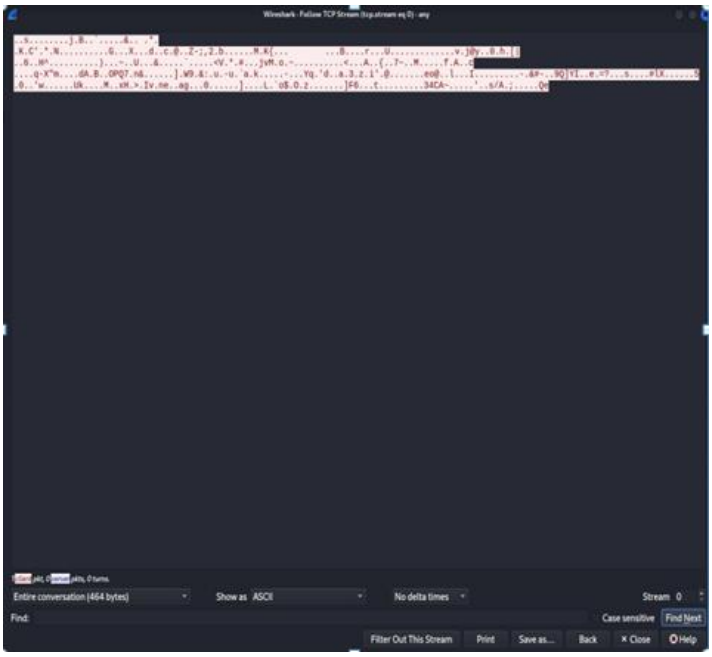

Figure 4. Wireshark inspection page

 https://doi.org/10.5281/zenodo.20313243

6. NETWORK LEVEL SECURITY VALIDATION

To validate the security of the proposed end-to-end encrypted messenger at the network level, packet inspection was performed using Wireshark during live communication between two authenticated clients. Traffic was captured on the server communication port while multiple encrypted messages were exchanged. The captured packets were analyzed in both summary and detailed payload views. The results showed that all transmitted data appeared as encrypted SSH protocol segments, with no readable plain text information such as usernames, message content or cryptographic keys exposed in the network stream.

Further inspection of the packet payload confirmed that the transmitted data consisted entirely of cipher text generated by the Secure Shell (SSH) transport layer combined with the application-level encryption mechanism implemented in the system. Even under deep packet inspection, the message contents remained unintelligible. This demonstrates that the proposed architecture provides strong protection against packet sniffing, eavesdropping, and unauthorized interception. The results confirm that both transport-layer encryption and application-layer end-to-end encryption work together to ensure comprehensive network-level security.

7. EXPERIMENTAL RESULTS AND OBSERVATIONS

The following table presents the experimentally measured average encryption and decryption times for different symmetric cryptographic algorithms implemented in the proposed secure messaging system. The benchmarking was conducted using two message sizes, namely 50 characters and 100 characters, to evaluate the computational efficiency and scalability of each algorithm. The results are reported in milliseconds (ms), representing the average time required to complete the respective cryptographic operation across multiple iterations. This comparison enables a clear performance evaluation of AES, Triple DES, RC4, Blowfish, and ChaCha20 under identical system conditions.

Table 1. Average encryption time for text file

S.No	Encryption algorithm	50 characters	100 characters	250 characters	500 characters
1.	AES	0.1288ms	0.0997ms	0.1108ms	0.1172ms
2.	Triple DES	0.0852ms	0.1233ms	0.1423ms	0.1532ms
3.	RC4	0.1235ms	0.1262ms	0.1264ms	0.1437ms
4.	BlowFish	0.1768ms	0.1805ms	0.2116ms	0.2398ms
5.	ChaCha20	0.1393ms	0.1633ms	0.1864ms	0.1892ms

Table 2. Average decryption time for text file

S.No	Decryption algorithm	50 characters	100 characters	250 characters	500 characters
1.	AES	2.3237ms	1.7857ms	0.0683ms	0.0432ms
2.	Triple DES	1.9717ms	2.1457ms	2.525ms	2.432ms
3.	RC4	1.8650ms	1.9699ms	2.183ms	2.582ms
4.	BlowFish	1.8540ms	1.770ms	1.982ms	2.139ms
5.	ChaCha20	2.688ms	3.7371ms	3.159ms	3.429ms

 https://doi.org/10.5281/zenodo.20313243

Based on the experimental results obtained from the benchmarking analysis, a comparative evaluation of five symmetric encryption algorithms—AES, Triple DES, RC4, Blowfish, and ChaCha20—was performed using average encryption and decryption times for 50-character, 100-character, 250-character, and 500-character message sizes. The results indicate that Triple DES demonstrated the lowest average encryption time for 50-character inputs (0.0852 ms), whereas AES achieved comparatively lower encryption times for 100-character (0.0997 ms) and 250-character (0.1108 ms) inputs. For larger inputs such as 500 characters, AES maintained a competitive encryption time of 0.1172 ms, while Blowfish and ChaCha20 showed higher encryption times across all message sizes, with Blowfish reaching 0.2398 ms for 500-character inputs.

In terms of decryption performance, Blowfish consistently exhibited lower average decryption times for 50-character and 100-character messages (1.854 ms and 1.770 ms respectively), while AES showed significantly faster decryption times for larger inputs, with 0.0683 ms for 250 characters and 0.0432 ms for 500 characters. ChaCha20, despite being secure and modern, recorded the highest decryption times across all message sizes, peaking at 3.7371 ms for 100-character inputs.

Considering both computational efficiency and cryptographic security, AES demonstrates the most balanced performance across different message sizes, making it highly suitable for secure messaging systems requiring both speed and reliability. Blowfish and RC4 provide marginally faster decryption for smaller inputs, but their encryption performance is comparatively slower. Overall, from a practical deployment perspective in end-to-end encrypted communication systems, AES remains the most optimal choice due to its consistent and secure performance across varying message lengths.

Table 3. Encryption and decryption time for png file

S.No	Algorithm Type	Encryption time	Decryption time
1.	AES	0.0946ms	0.0626ms
2.	Triple DES	0.3224ms	0.4010ms
3.	RC4	1.2395ms	1.5123ms
4.	BlowFish	1.4233ms	1,2323ms
5.	ChaCha20	1.1422ms	1.0230ms

Based on the experimental results obtained from the benchmarking analysis, a comparative evaluation of five symmetric encryption algorithms—AES, Triple DES, RC4, Blowfish, and ChaCha20—was performed using average encryption and decryption times for PNG image files. The results indicate that AES demonstrated the fastest encryption time (0.0946 ms) as well as the lowest decryption time (0.0626 ms), making it highly efficient for handling image data. In contrast, Triple DES showed moderate performance with 0.3224 ms for encryption and 0.4010 ms for decryption, while RC4 and Blowfish recorded significantly higher encryption and decryption times, with Blowfish reaching 1.4233 ms for encryption and RC4 peaking at 1.5123 ms for decryption. ChaCha20, though modern and secure, exhibited slower times compared to AES, with 1.1422 ms for encryption and 1.023 ms for decryption.

Considering both computational efficiency and security, AES emerges as the most optimal algorithm for PNG image encryption and decryption, offering strong cryptographic protection with minimal processing overhead. While Blowfish and RC4 may provide competitive performance in certain contexts, their higher processing times make them less suitable for real-

 https://doi.org/10.5281/zenodo.20313243

time or large-scale secure image transmission. Overall, AES provides the best balance of speed and security, making it the preferred choice for secure handling of image files in end-to-end encrypted systems.

Table 4. Encryption and decryption time for jpeg file

S.No	Algorithm Type	Encryption time	Decryption time
1.	AES	0.0946ms	0.0626ms
2.	Triple DES	0.2721ms	0.3013ms
3.	RC4	1.3212ms	1.5233ms
4.	BlowFish	1.4238ms	1.2230ms
5.	ChaCha20	1.2122ms	0.9020ms

Based on the experimental results obtained from the benchmarking analysis, a comparative evaluation of five symmetric encryption algorithms—AES, Triple DES, RC4, Blowfish, and ChaCha20—was performed using average encryption and decryption times for JPEG image files. The results indicate that AES demonstrated the fastest encryption time (0.0946 ms) and the lowest decryption time (0.0626 ms), showing consistently high efficiency for JPEG images. Triple DES performed moderately with encryption and decryption times of 0.2721 ms and 0.3013 ms respectively, while RC4 and Blowfish exhibited significantly higher processing times, with RC4 reaching 1.3212 ms for encryption and 1.5233 ms for decryption. ChaCha20, despite being modern and secure, showed slower performance relative to AES, with 1.2122 ms for encryption and 0.902 ms for decryption. Considering both computational speed and security, AES proves to be the most optimal algorithm for JPEG image encryption and decryption, providing strong cryptographic protection with minimal processing overhead. While Blowfish and RC4 may still offer competitive security in some scenarios, their slower performance makes them less suitable for real-time or high-volume secure image transmission. Overall, AES offers the best combination of speed and reliability, making it the preferred choice for end-to-end encrypted handling of JPEG images.

Table 5. Encryption and decryption time for pdf file

S.No	Algorithm Type	Encryption time	Decryption time
1.	AES	0.1708ms	0.2343ms
2.	Triple DES	0.2721ms	0.0620ms
3.	RC4	0.9181ms	0.9759ms
4.	BlowFish	1.4890ms	1.3859ms
5.	ChaCha20	1.5820ms	1.8290ms

Based on the experimental results obtained from the benchmarking analysis, a comparative evaluation of five symmetric encryption algorithms—AES, Triple DES, RC4, Blowfish, and ChaCha20 was performed using average encryption and decryption times for PDF files. The results indicate that AES exhibited competitive encryption and decryption performance with 0.1708 ms and 0.2343 ms respectively, maintaining efficiency across larger and structured document files. Triple DES showed a faster decryption time of 0.062 ms, but its encryption time (0.2721 ms) was higher than AES. RC4 and Blowfish recorded moderately higher processing times, with Blowfish reaching 1.489 ms for encryption and 1.3859 ms for decryption.

 https://doi.org/10.5281/zenodo.20313243

ChaCha20, while secure and modern, had the slowest performance, with 1.582 ms for encryption and 1.829 ms for decryption.

Considering both speed and cryptographic security, AES emerges Aas a strong choice for PDF file encryption and decryption, offering a balanced performance suitable for secure document handling. Although Triple DES provides very fast decryption, its higher encryption time may limit real-time use cases. RC4, Blowfish, and ChaCha20, due to their comparatively slower processing, are less optimal for large-scale or high-frequency secure PDF transmissions. Overall, AES provides the best trade-off between efficiency and security, making it the preferred algorithm for end-to-end encrypted handling of PDF files.

Table 6. Encryption and decryption time for pptx file

S.No	Algorithm Type	Encryption time	Decryption time
1.	AES	0.0923ms	0.0792ms
2.	Triple DES	0.6833ms	0.1211ms
3.	RC4	0.9123ms	0.8193ms
4.	BlowFish	1.2849ms	1.9283ms
5.	ChaCha20	2.4892ms	3.0828ms

Based on the experimental results obtained from the benchmarking analysis, a comparative evaluation of five symmetric encryption algorithms—AES, Triple DES, RC4, Blowfish, and ChaCha20—was performed using average encryption and decryption times for PPTX files. The results indicate that AES demonstrated the fastest encryption (0.0923 ms) and decryption (0.0792 ms) times, confirming its efficiency for presentation files. Triple DES showed a moderate performance with 0.6833 ms for encryption and 0.1211 ms for decryption, while RC4 and Blowfish exhibited higher processing times, with Blowfish reaching 1.2849 ms for encryption and 1.9283 ms for decryption. ChaCha20, despite its modern cryptographic design, recorded the slowest performance with 2.4892 ms for encryption and 3.0828 ms for decryption.

Table 7. Encryption and decryption time for docx file

S.No	Algorithm Type	Encryption time	Decryption time
1.	AES	0.0902ms	0.0815ms
2.	Triple DES	0.8349ms	0.4691ms
3.	RC4	0.7139ms	0.5173ms
4.	BlowFish	1.6019ms	2.1029ms
5.	ChaCha20	2.7910ms	4.9342ms

Based on the experimental results obtained from the benchmarking analysis, a comparative evaluation of five symmetric encryption algorithms—AES, Triple DES, RC4, Blowfish, and ChaCha20—was conducted using average encryption and decryption times for DOCX files. The results indicate that AES achieved the fastest encryption time (0.0902 ms) as well as the lowest decryption time (0.0815 ms), demonstrating superior efficiency for document-based file handling. Triple DES recorded moderate performance with 0.8349 ms for encryption and 0.4691 ms for decryption, while RC4 showed slightly better encryption performance than Triple DES (0.7139 ms) but comparable decryption time (0.5173 ms). Blowfish and ChaCha20 exhibited significantly higher processing times, with Blowfish reaching 1.6019 ms for

 https://doi.org/10.5281/zenodo.20313243

encryption and 2.1029 ms for decryption, and ChaCha20 showing the highest overhead at 2.7910 ms for encryption and 4.9342 ms for decryption.

Considering both computational efficiency and cryptographic strength, AES clearly emerges as the most optimal algorithm for DOCX file encryption and decryption. While Triple DES and RC4 provide acceptable performance, their slower speeds compared to AES may limit their suitability for real-time or high-frequency secure document transmission. Blowfish and ChaCha20, due to their comparatively higher processing times, are less efficient for practical deployment in secure messaging systems handling DOCX files. Overall, AES provides the best balance of speed, reliability, and security, making it the preferred algorithm for end-to-end encrypted document communication systems.

8. Conclusion and Future Scope

This paper presented the design and implementation of a secure end-to-end encrypted messenger system developed using Secure Shell (SSH) for transport-layer security and hybrid cryptography for application-layer protection. The system integrates RSA for secure key exchange and multiple symmetric encryption algorithms—including AES, Triple DES, RC4, Blowfish, and ChaCha20—to ensure confidentiality of text messages and various file formats such as PNG, JPEG, PDF, PPTX, and DOCX. Comprehensive performance benchmarking was conducted to evaluate encryption and decryption times across different data types and file formats. The experimental results consistently demonstrate that AES achieves the lowest encryption and decryption times across almost all tested scenarios, including text messages of varying lengths and multimedia/document files. While Triple DES, RC4, Blowfish, and ChaCha20 show moderate to high computational overhead depending on the file type, AES maintains a balanced performance with minimal processing delay and strong cryptographic security. The results clearly indicate that AES provides the most optimal trade-off between efficiency and security for secure messaging applications. Furthermore, packet-level validation confirmed that all transmitted data remains encrypted during transmission, ensuring resistance against interception and unauthorized access. Overall, the proposed system successfully delivers secure, reliable, and scalable encrypted communication suitable for practical deployment.

Future enhancements can focus on strengthening cryptographic resilience and expanding system capabilities. Advanced security features such as Perfect Forward Secrecy (PFS), digital signatures for message authentication, hash-based message integrity verification, and automated key rotation mechanisms can be incorporated to enhance protection against emerging threats. The implementation of secure group messaging, encrypted cloud-based file storage, and real-time multimedia encryption would further extend the functionality of the system. Additionally, migrating from a desktop-based architecture to a cross-platform or mobile-based deployment would significantly improve accessibility and usability. Performance optimization through hardware-accelerated cryptographic operations, load balancing, and large-scale stress testing can further enhance real-world applicability. With these improvements, the proposed prototype can evolve into a production-ready secure communication platform capable of supporting enterprise-level secure messaging environments.

References

[1] K. J. Kumar, K. A. Sai, A. D. Reddy, C. P. D. Reddy, and S. M. Rajagopal (2025) “A comprehensive end-to-end solution for web security with cryptography, multi-factor authentication, and secure communication”, *in Proc. 2025 3rd Int. Conf. Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*.

 https://doi.org/10.5281/zenodo.20313243

[2] D. Ramakrishna and M. A. Shaik (2024) "A comprehensive analysis of cryptographic algorithms: Evaluating security, efficiency, and future challenges", *IEEE Access*, vol. 13, pp. 11576–11593.

[3] D. E. R. Alam, H. Al Fitrah, M. Ayunasari, and D. Ogi (2023) “Implementation of ChaCha20 algorithm with elliptic-curve-Diffie-Hellman in server room monitoring system to prevent MITM and reused key attack", *in Proc. 2023 IEEE Int. Conf. Cryptography, Informatics, and Cybersecurity (ICoCICs)*, Bogor, Indonesia.

[4] D. D. Tran, K. Ogata, S. Escobar, S. Akleylek, and A. Otmani (2023) “Formal analysis of post-quantum hybrid key exchange SSH transport layer protocol”, *IEEE Access*, vol. 12, pp. 1672–1687.

[5] S. Liu, Y. Li, and Z. Jin (2023) “Research on enhanced AES algorithm based on key operations”, *in Proc. 2023 IEEE 5th Int. Conf. Civil Aviation Safety and Information Technology (ICCASIT)*, Dali, China.

[6] S. Subaselvi, C. Mytheesh, R. Sanjay, G. Parithi Malavan, and S. D. Ragunath (2023) “VLSI implementation of Triple-DES block cipher”, *in Proc. 7th Int. Conf. Computing Methodologies and Communication (ICCMC)*, Erode, India.

[7] A. M. Aburbeian and M. Fernandez-Veiga (2024) “Secure internet financial transactions: A framework integrating multi-factor authentication and machine learning”, *AI*, vol. 5, pp. 177–194.

[8] A. K. Verma, M. Rakhra, A. Bhattacherjee, A. Maan, T. Sarkar, and V. K. Pandey (2024) “A suggested model for using multi-factor authentication framework in cloud computing for SME” *in Proc. Int. Conf. Cybernation and Computation (CYBERCOM)*, Phagwara, India.

[9] V. R., N. Harini, and N. M. R. (2023) “Multi-factor authentication system with ID card credentials for secure transactions”, *in Proc. 2023 14th Int. Conf. Computing Communication and Networking Technologies (ICCCNT)*, Delhi, India.

[10] M. I. M. Yusop, N. H. Kamarudin, N. H. S. Suhaimi, and M. K. Hasan (2025) “Advancing passwordless authentication: A systematic review of methods, challenges, and future directions for secure user identity”, *IEEE Access*, vol. 13, pp. 13919–13938.

[11] D. D. Kumar, J. D. Mukharjee, C. V. D. Reddy, and S. M. Rajagopal (2024) “Safe and secure communication using SSL/TLS”, *in Proc. Int. Conf. Emerging Smart Computing and Informatics (ESCI)*, Pune, India.

[12] F. Bozkurt, M. Kara, M. A. Aydın, and H. H. Balik (2023) “Exploring the vulnerabilities and countermeasures of SSL/TLS protocols in secure data transmission over computer networks”, *in Proc. 12th IEEE Int. Conf. Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Dortmund, Germany.

[13] A. Raich and V. Gadicha (2021) “Various threats and challenges to information security via active and passive attack”, *in Proc. Int. Conf. Smart Generation Computing, Communication and Networking (SMART GENCON)*, Pune, India.

[14] G. Hu, J. Si, Y. Cai, and N. Al-Dhahir (2022) “Proactive eavesdropping via jamming over multiple suspicious links with wireless-powered monitor”, *IEEE Signal Processing Letters*, vol. 29, pp. 354–358.

[15] Y. Hariprasad, S. S. Iyengar, and N. K. Chaudhary (2024) “Securing the future: Advanced encryption for quantum-safe video transmission”, *IEEE Trans. Consumer Electronics*, vol. 70, no. 1, pp. 1–10.

[16] A. Ajith, A. S. Mathew, and R. S. (2023) “A brief study on cloud security,” *in Proc. 14th Int. Conf. Computing Communication and Networking Technologies (ICCCNT)*, Delhi, India.

[17] S. Sruthi, U. Kumaran, P. K. Oyyavuru, S. Emadaboina, S. P. Machavarapu, and S. Balasubramanian (2024) “Securing financial technology: Mitigating vulnerabilities in fintech applications”, *in Advances in Information Communication Technology and Computing (AICTC 2024), Lecture Notes in Networks and Systems*, vol. 1074, pp. 205–214.

 https://doi.org/10.5281/zenodo.20313243

[18] K. Nimmy, S. Sankaran, and K. Achuthan (2018) “A novel multi-factor authentication protocol for smart home environments”, *in Information Systems Security (ICISS 2018), Lecture Notes in Computer Science*, vol. 11281, pp. 44–63.

[19] S. Bhardwaj, R. Dhaksana, K. A. Varshini, and N. Harini (2024) “Wearable security—Authentication using smartwatches”, *in Proc. 2nd Int. Conf. Advancement in Computation & Computer Technologies (InCACCT),* Gharuan, India.

[20] T. E. Djidjekh, A. Takacs, G. Loubet, L. Sanogo, and D. Dragomirescu (2026) "Lightweight protocol-agnostic security enhancement for SWIPT-enabled IoT systems", *IEEE Internet Things J., Early Access*, Feb. 11, 2026.

[21] K. Iwamura and A. A. A. M. Kamal (2025) “Secure user authentication with information theoretic security using secret sharing-based secure computation”, *IEEE Access*, vol. 13, pp. 9015–9031.

[22] A. H. Sarower, T. Bhuiyan, M. Hassan, M. S. Arefin, and G. Hossain (2025) "SMFA: Strengthening multi-factor authentication with steganography for enhanced security", *IEEE Access*, vol. 13, pp. 43593–43606.

[23] G. A. P. Rodrigues et al (2020) “Securing instant messages with hardware-based cryptography and authentication in browser extension”, *IEEE Access*, vol. 8, pp. 93387–93402.

[24] Sunardi, Herman, and S. R. Ardiningtias (2022) “A comparative analysis of digital forensic investigation tools on Facebook Messenger applications”, *J. Cyber Security Mobility*, vol. 11, no. 5, pp. 655–672.

Authors

Dr.A.Neela Madheswari completed B.E. during 2000, completed M.E. during 2006, completed Ph.D. during 2014. Her research interests are Web Technologies, Parallel and Distributed systems, Cloud Computing, Cyber Security and Mobile Networks. She has 5 SCIE publications, 30 Scopus publications, 3 books and 16 patents publications,

K. Aloysius Rosario, B.E Cyber Security during 2022-2026. His research interests are Cryptography, Secure Communications Methodologies, Game Development and 3D Development. He has one journal publication under IJET.

M.Arun Kumar, B.E Cyber Security during 2022-2026. His interests are Linux Systems, Ethical Hacking, python development, Cloud Computing, Data Structures, Windows Security. He has one journal publication under IJSET.

 https://doi.org/10.5281/zenodo.20313243

S.Aasif Hameed, B.E Cyber Security during 2022-2026. His interests are Ethical Hacking, Linux Systems, C Sharp Development, Python Development, Windows File Systems and Web Development. He has one journal publication under IJEST.

P.Anbarasu , B.E. Cyber Security during 2022-2026. His Interests are Ethical Hacking, Shell Programming, Linux File Systems, App Development and Web Technologies. He has one journal publication under IJSET.

https://doi.org/10.5281/zenodo.20313243