Secure Network Messenger Using SSH for Improved Anonymity

Dr.A. Neela Madheswari
dept. Of Cyber Sceurity
Mahendra Engineering College Namakkal,India
neelamadheswaria@mahendra.info

K Aloysius Rosario
dept. Of Cyber Security
Mahendra Engineering College
Namakkal,India
aloysiusrosario6@gmail.com

M Arunkumar
dept. Of Cyber Security
Mahendra Engineering College
Namakkal, India
arunkumar772235@gmail.com

S Aasif Hameed
dept.of Cyber Security Mahendra Engineering College
Namakkal,India
aasifhameed2003@gmail.com

P Anbarasu
dept.of Cyber Security Mahendra Engineering College
Namakkal,India
anbarasu.cyber@gmail.com

***Abstract*—Secure communication systems are essential in modern digital environments where privacy and confidentiality are critical. This paper presents a 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, TripleDES, 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 tradeoffs. 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, AES, Symmetric encryption, Encryption benchmarking, Network security

I. 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 through the 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 analyze security strength and computational performance of commonly used symmetric encryption algorithms within a real time messaging system.

II. 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.

A. Cryptographic Mechanisms and Secure Communication

Cryptographic protection forms the foundation of secure digital communication systems. Kumar et al. [1] proposed a comprehensive end-to-end web security solution integrating cryptography, multi-factor authentication, and secure communication protocols. Their framework ensures 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 [7] 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. [8] implemented the ChaCha20 algorithm combined with Elliptic-Curve Diffie-Hellman (ECDH) to prevent 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. [9] 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.

B. Authentication and Multi-Factor Security Models

Authentication mechanisms are critical in preventing unauthorized access to communication systems. Iwamura and Kamal [4] 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. [5] 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.

Djidjekh et al. [2] 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.

C.Secure Transport and Hybrid Encryption Mechanisms

Secure transport protocols ensure confidentiality and integrity during data transmission. Tran et al. [9] analyzed hybrid key exchange mechanisms within the SSH protocol, reinforcing the importance of secure transport-layer encryption. Similarly, Alam et al. [8] 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.

D. Secure Messaging and Digital Forensics

Secure instant messaging systems require reliable encryption and authentication mechanisms. Rodrigues et al. [6] 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 [10] 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.

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

III. System Architecture

The proposed system consists of three major components:

1. Authentication and Session Management Module
2. Cryptographic Processing Module
3. 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.

IV. Cryptographic Design

A. Asymmetric Encryption

RSA-2048 is used for secure key exchange, providing a balance between computational efficiency and cryptographic strength consistent with current security recommendations.

B. Symmetric Encryption Algorithms

The system supports the following symmetric algorithms:
AES
Triple DES
RC4
BlowFish
ChaCha20
AES and ChaCha20 are modern secure algorithms. TripleDES and Blowfish are legacy algorithms included for benchmarking comparison. RC4 is included for performance comparison though it is considered insecure for modern system

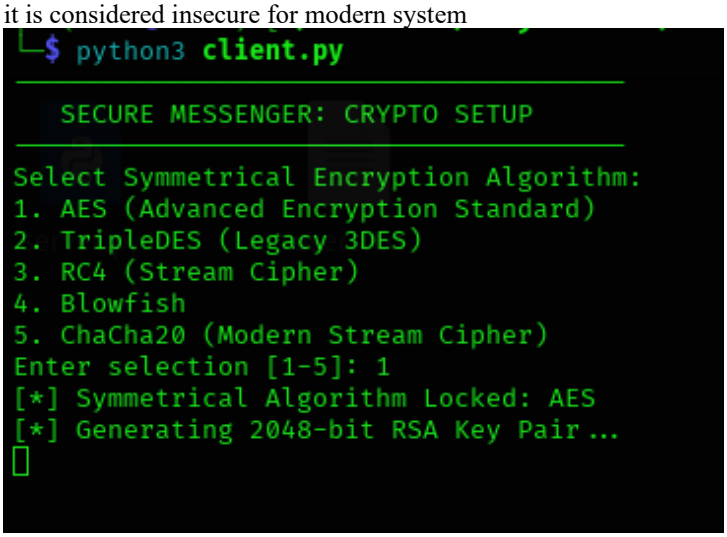

Fig.1. Algorithm Selection Options

We can enter the alloted integer to select an algorithm to use for the communication

Fig.2. After Selecting the Desired Algorithm

Messages are encrypted using Cipher Feedback mode where applicable. For ChaCha20, stream cipher encryption is applied.

V. User Authentication And Key Generation

When a client starts, the following steps are performed:

1.The user provides credentials (username and password).

2.The client generates an RSA public–private key pair.

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

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

A. Hybrid Encryption Mechanism

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

1. Symmetric Key Generation
   For every message, a random symmetric session key is generated.
2. Message Encryption
   The plaintext message is encrypted using the selected symmetric algorithm (e.g., AES, ChaCha20, or Blowfish). Symmetric encryption is chosen for its computational efficiency.
3. Key Encryption Using RSA
   The generated symmetric session key is encrypted using the receiver's RSA public key. This ensures that only the intended recipient can decrypt the session key.
4. Secure Transmission
   The encrypted message and encrypted session key are transmitted together to the server, which forwards them to the intended recipient.
5. *Decryption Process*
   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.

B. Abbreviations and Acronyms

The following abbreviations and acronyms are used throughout this paper:

1. E2EE: End-to-End Encrypted
2. SSH : Secure Shell

   RSA : Rivest-Shamir-Aldman public key cryptographic algorithm
4. AES : Advance Encryption Algorithm
5. CBC : Cyber Block Chaining
6. GUI : Graphical User Interface

TCP : Transmission Control Protocol

DES : Data Encryption Standard

All acronyms are defined at their first occurrence in the text to maintain clarity and consistency with IEEE formatting standards.

C. Units

When calculating encryption time and decryption time, the standard unit used is **seconds (s)**, which is the SI unitof 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.

D. Performance Evaluation Metrics

To evaluate the effectiveness of the proposed secure messaging system, several performance metrics were considered. The primary metric is encryption and decryption time, which measures the computational cost of each cryptographic operation. Additionally, key generation time is analyzed to determine the overhead introduced by RSA during session initialization. Memory usage is also observed to assess resource consumption on the client side. Network overhead is evaluated by measuring the size of transmitted ciphertext compared to the original plaintext, particularly in the hybrid encryption model where both the encrypted message and encrypted session key are sent together. These metrics provide a comprehensive assessment of security-performance trade-offs and help determine the most efficient symmetric algorithm for real-time secure communication.

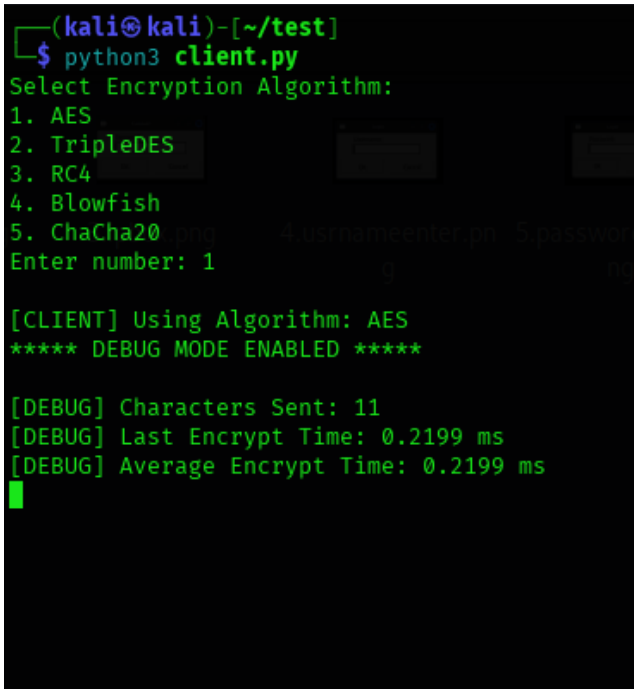

Fig.3. Encryption data

- 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 inseconds as floating-point numbers. Even though the base unit returned is seconds, the result is usuallymultiplied by 1000 to convert it into milliseconds for clearer comparison.
- 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.

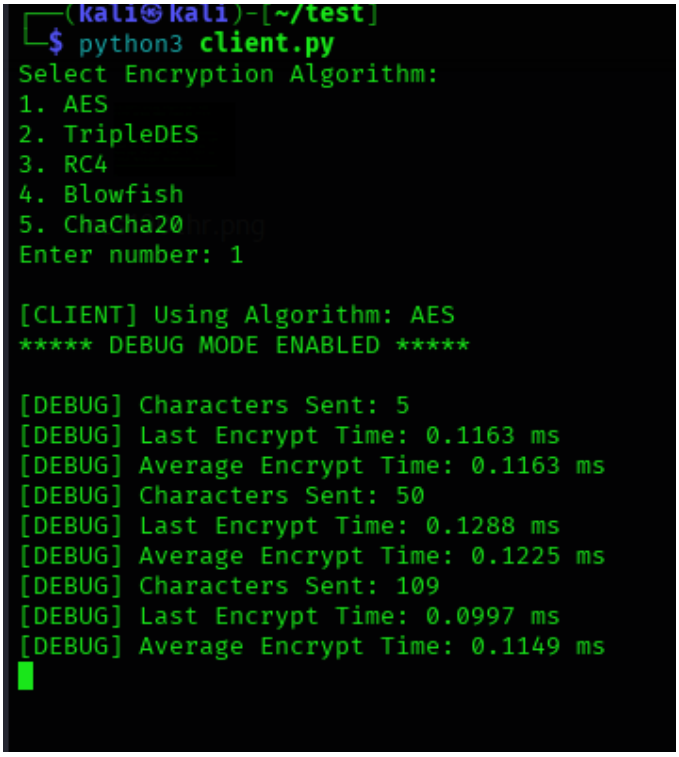

Fig. 4. Time units are mentioned in ms.

E. Equations

The equations are an exception to the prescribed specifications of this template. You will need to determine whether or not your equation should be typed using either the Times New Roman or the Symbol font (please no other font). To create multileveled equations, it may be necessary to treat the equation as a graphic and insert it into the text after your paper is styled.

Number equations consecutively. Equation numbers, within parentheses, are to position flush right, as in (1), using a right tab stop. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

1. Encryption Time:

$$Tenc=tend− tstart$$

Where:

· tstartt_{start}tstart = Time before encryption

· tendt_{end}tend = Time after encryption

2. 2.Decryption Time:

$$Tdec=tend− tstart$$

3. Average Encryption Time:

If encryption is performed n times:

$$Tavg = n\sum i=1nTi/n$$

Where:

- Ti = Time taken for each encryption
- n = Total number of run

Measuring both encryption and decryption time ensures that the proposed system not only maintains strong security through end-to-end encryption but also achieves acceptable performance levels suitable for real-time messaging applications.

F. *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 plaintext information such as usernames, message content, or cryptographic keys exposed in the network stream.

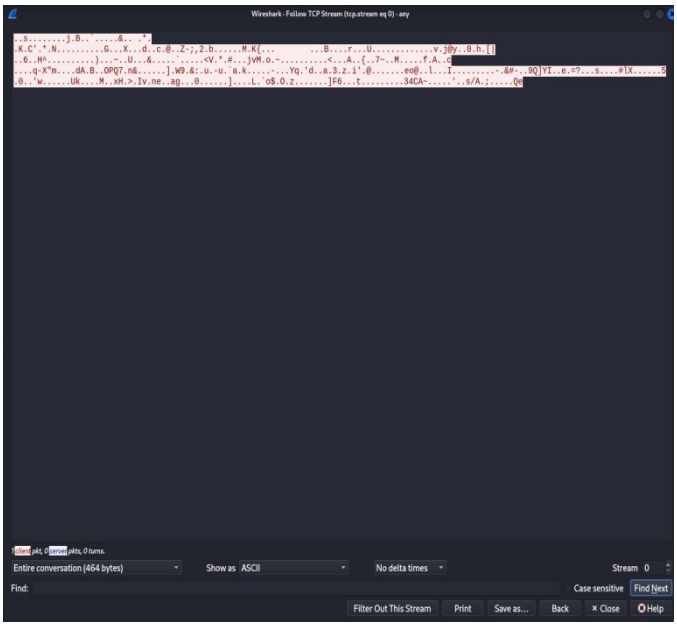

Fig.6.Wire Shark Inspection

Further inspection of the packet payload confirmed that the transmitted data consisted entirely of ciphertext 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.

G. *H. Tables*

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 ENCRYPTION/DECRYPTION FOR TEXT

S.no	**Average Encryption Time**				
	Table column subhead	***50 chars***	***100chars***	***250chars***	***500chars***
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
S.no	**Average DecryptionTime**				
	Table column subhead	***50 chars***	***100chars***	***250chars***	***500chars***
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
24	BlowFish	1.8540ms	1.770ms	1.982ms	2.139ms
5	ChaCha20	2.688ms	3.7371ms	3.159ms	3.429ms

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 2 ENCRYPTION/DECRYPTION FOR .PNG

	Algorithms	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	1.023ms

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-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 3 ENCRYPTION/DECRYPTION FOR .JPEG

	Algorithms	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	1.223ms
5	ChaCha20	1.2122ms	0.902ms

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 3 ENCRYPTION/DECRYPTION FOR .PDF

	Algorithms	Encryption Time	Decryption Time
1	AES	0.1708ms	0.2343ms
2	Triple DES	0.2721ms	0.062ms
3	RC4	0.9181ms	0.9759ms
4	BlowFish	1.489ms	1.3859ms
5	ChaCha20	1.582ms	1.829ms

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

	Algorithms	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

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 4 ENCRYPTION/DECRYPTION FOR .PPTX

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.

Conclusion And Future Scope

This paper presented the design and implementation of a secure end-to-end encrypted messenger system built using Secure Shell (SSH) for transport-layer protection and hybrid cryptography for application-layer security. The system integrates RSA for secure key exchange and multiple symmetric encryption algorithms, including AES, TripleDES, RC4, Blowfish, and ChaCha20, for message confidentiality.

Performance benchmarking was conducted to evaluate encryption and decryption times across different algorithms and message sizes. The experimental results demonstrate that AES provides a balanced combination of strong security and efficient performance, while other algorithms exhibit varying trade-offs between speed and computational cost. Network- level validation using packet analysis further confirmed that all transmitted data remains encrypted and resistant to interception. Overall, the proposed system successfully achieves secure, efficient, and scalable encrypted communication.

Future enhancements can focus on improving both security robustness and system scalability. Advanced features such as Perfect Forward Secrecy (PFS), digital signatures for message authentication, and key rotation mechanisms can be integrated to strengthen cryptographic resilience. The implementation of secure group messaging, file transfer encryption, and multimedia message protection would expand system capabilities. Additionally, migrating from a desktop-based architecture to a cross-platform or mobile-based deployment would improve usability and accessibility. Performance optimization through hardware-accelerated cryptography and large-scale stress testing can further enhance real-world applicability. These improvements would transform the current prototype into a production-grade secure communication platform suitable for broader deployment

ACKNOWLEDGMENT

The authors would like to thank the faculty members of the Department of CyberSecurity for their valuable guidance and technical support throughout the development of this project. Special thanks are extended to the project supervisor for continuous encouragement and constructive feedback.

REFERENCES

[1] K. J. Kumar, K. A. Sai, A. D. Reddy, C. P. D. Reddy, and S. M. Rajagopal, “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), Feb. 2025, doi: 10.1109/IDCIOT64235.2025.10915145.

[2] T. E. Djidjekh, A. Takacs, G. Loubet, L. Sanogo, and D. Dragomirescu, “Lightweight protocol-agnostic security enhancement for SWIPT-enabled IoT systems,” IEEE Internet Things J., Early Access, Feb. 11, 2026, doi: 10.1109/JIOT.2026.3663551.

[3] A. Mahida, “An intellectual zero trust security framework using deep reinforcement learning for predictive threat mitigation in AI-based fraud detection systems,” IEEE Access, Early Access, Feb. 13, 2026, doi: 10.1109/ACCESS.2026.3664389.

[4] K. Iwamura and A. A. A. M. Kamal, “Secure user authentication with information theoretic security using secret sharing-based secure computation,” IEEE Access, vol. 13, pp. 9015–9031, Jan. 2025, doi: 10.1109/ACCESS.2025.3526632.

[5] A. H. Sarower, T. Bhuiyan, M. Hassan, M. S. Arefin, and G. Hossain, “SMFA: Strengthening multi-factor authentication with steganography for enhanced security,” IEEE Access, vol. 13, pp. 43593–43606, Feb. 2025, doi: 10.1109/ACCESS.2025.3545769.

[6] G. A. P. Rodrigues et al., “Securing instant messages with hardware-based cryptography and authentication in browser extension,” IEEE Access, vol. 8, pp. 93387–93402, 2020, doi: 10.1109/ACCESS.2020.2993774.

[7] D. Ramakrishna and M. A. Shaik, “A comprehensive analysis of cryptographic algorithms: Evaluating security, efficiency, and future challenges,” IEEE Access, vol. 13, pp. 11576–11593, Dec. 2024.

[8] D. E. R. Alam, H. Al Fitrah, M. Ayunasari, and D. Ogi, “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, Aug. 2023, doi: 10.1109/ICoCICs58778.2023.10277110.

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

[10] Sunardi, Herman, and S. R. Ardiningtias, “A comparative analysis of digital forensic investigation tools on Facebook Messenger applications,” J. Cyber Security Mobility, vol. 11, no. 5, pp. 655–672, Sep. 2022, doi: 10.13052/jcsm2245-1439.1151.

TABLE 5 ENCRYPTION/DECRYPTION FOR .DOCX

	Algorithms	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