Privacy Preserving Reversible Data Embedding In Encrypted Image

Gayathri K[1], Keshava Varshini M[2], Vani Shree V[3], Mr. S. Ganeshkumar[4]
[1, 2, 3]Dept of Computer science and Engineering
[4]Assistant Professor, Dept of Computer science and Engineering
[1, 2, 3, 4] Sree Sowdambika College of Engineering, Virudhunagar, Tamil Nadu, India

Abstract- *In the modern digital era, secure communication has become a critical requirement due to the rapid growth of cyber threats, data breaches, and unauthorized access to confidential information. This paper presents a highly secure web-based communication system developed using Python Flask and MySQL that integrates advanced cryptographic and steganographic techniques to ensure multi-layer protection of transmitted data. The system enables authenticated users to register, log in, and securely exchange messages in the form of text or files. Unlike traditional messaging systems that rely solely on encryption, this application combines AES-GCM (Advanced Encryption Standard – Galois/Counter Mode) encryption with LSB (Least Significant Bit) image steganography to provide double-layer security. In the proposed system, the user first encrypts the message payload using AES-GCM encryption with a secret passphrase. AES-GCM ensures confidentiality, integrity, and authentication by generating secure cryptographic components such as salt, nonce, ciphertext, and authentication tag. The encrypted payload is then embedded inside a cover image using LSB steganography, which hides the existence of the secret data by modifying the least significant bits of image pixels without visibly altering image quality. After embedding, the generated stego image is again encrypted using AES-GCM before being stored or transmitted, thereby adding an additional security layer. This dual protection mechanism prevents attackers from detecting hidden data even if the encrypted file is intercepted. Experimental results demonstrate that the proposed system achieves high imperceptibility with PSNR values exceeding 60 dB and provides robust protection against unauthorized access and detection attacks.*

Keywords: AES-GCM, LSB steganography, Encryption, Data hiding, Reversible Data Embedding

I. INTRODUCTION

The exponential growth of digital communication and cloud-based services has intensified the need for robust security mechanisms to protect sensitive information from cyber threats, data breaches, and unauthorized access. Traditional messaging systems typically employ encryption as the primary line of defense, ensuring that intercepted messages remain unintelligible to unauthorized parties. However, encryption alone has a fundamental limitation: the very existence of encrypted communication can alert adversaries to the presence of sensitive information, making it a target for advanced cryptanalysis attacks .

Steganography addresses this limitation by concealing the existence of the communication itself. By hiding secret data within innocuous cover media such as images, audio, or video files, steganography provides an additional layer of privacy that complements encryption . The combination of cryptography and steganography creates a powerful synergy: encryption renders the data unintelligible, while steganography makes its very presence undetectable .

This paper presents a secure web-based communication system that integrates AES-GCM (Advanced Encryption Standard – Galois/Counter Mode) encryption with LSB (Least Significant Bit) image steganography. The system is built using Python Flask for the web framework and MySQL for database management, providing a practical, deployable solution for secure messaging. The key contributions of this work are:

1. A multi-layer security architecture that combines AES-GCM encryption with LSB steganography
2. Implementation of a web-based communication platform with user authentication
3. A novel approach of double encryption: message encryption before embedding and stego image encryption after embedding
4. Comprehensive security analysis and performance evaluation

The remainder of this paper is organized as follows: Section II reviews related work in cryptography and steganography integration. Section III provides background on the core technologies. Section IV details the proposed system architecture and methodology. Section V presents the algorithm implementation. Section VI discusses experimental results and performance evaluation. Section VII concludes the paper and suggests future research directions.

II. LITERATURE REVIEW

The integration of cryptography and steganography for secure communication has been extensively studied in recent years. This section reviews significant contributions to the field, with particular emphasis on AES-GCM encryption and LSB steganography techniques.

A. Cryptography-Steganography Integration

Buhurcu[4] proposed a secure communication model integrating AES-GCM encryption with LSB steganography for mobile messaging applications. The study demonstrated that AES-GCM achieves 99.61% data modification with encryption speeds of 269.61 MB/s on mobile devices, making it optimal for real-time applications. The steganographic performance yielded PSNR values between 51.37 and 82.39 dB, significantly exceeding the 40 dB threshold required for successful steganography. The work also addressed the symmetric key distribution problem by incorporating RSA for secure key exchange.

Ahmad[1] developedStegoCrypt, a web-based Python application combining AES-256-GCM encryption with LSB steganography for hiding messages in PNG images. The implementation used PBKDF2-HMAC-SHA256 with 600,000 iterations for key derivation from user passwords. The study highlighted critical challenges in practical implementations, including metadata management, embedding capacity limitations (theoretically 3 bits per pixel), vulnerability to statistical detection, and sensitivity to image modifications.

B. Advanced LSB Steganography Techniques

Recent advances in LSB steganography have focused on content-adaptive embedding to improve imperceptibility and resistance to steganalysis. A comprehensive study[3] proposed a framework integrating saliency-guided embedding, Ant Colony Optimization (ACO)-based dispersion, and hybrid encryption. The system achieved exceptional imperceptibility with PSNR values of 59.7–60.2 dB for 64×64 secret images and up to 64.5 dB for 32×32 secrets, with SSIM consistently above 0.999. The content-adaptive approach demonstrated strong resistance to modern CNN-based steganalysis tools.

C. Applications in Sensitive Data Protection

Kadhum and Ali[6] applied AES-GCM encryption and LSB steganography for protecting sensitive data on smartphones as a second-level defense after implicit authentication. Their implementation on Android achieved optimal results with PSNR of 70.95 dB and MSE of 0.005245 for a 27 KB sunset image. The Pearson correlation coefficient remained zero, confirming the effectiveness of AES-GCM encryption in producing uncorrelated ciphertext.

D. Combined Encryption and Steganography Systems

Bhangaleet al.[8][10] presented a comprehensive review of systems converging encryption, hashing, and steganography for data fortification. Their proposed system combined AES-GCM with flip and rotational LSB techniques to ensure secure data transmission and covert concealment within digital images. The integrated approach addressed both confidentiality and integrity while enabling covert communication.

E. Flask-Based Secure Communication Platforms

Several implementations have demonstrated the feasibility of building secure communication systems using Flask. AkithN[2] developed a secure messaging app integrating Flask, MySQL, and cryptographic libraries (cryptography and PyCryptodome) to ensure user privacy and data integrity. Similarly, Whitehat8889[7] created CRYPTOWL, a web application for encrypted messaging using Flask and MariaDB, targeting beginner-friendly cryptography education.

F. Research Gap and Contribution

While existing work has explored individual aspects of cryptography-steganography integration, few implementations provide a complete, web-based solution with practical deployment considerations. Furthermore, the approach of double encryption—encrypting both the message before embedding and the resulting stego image—has not been extensively studied. This paper addresses these gaps by presenting a fully functional web-based system that implements multi-layer security with comprehensive performance evaluation.

III. BACKGROUND AND TECHNOLOGIES

A. AES-GCM Encryption

AES-GCM (Advanced Encryption Standard – Galois/Counter Mode) is an authenticated encryption algorithm that provides both confidentiality and integrity [1][4]. It combines the AES block cipher in counter mode for encryption with Galois mode for authentication, producing both cipher text and an authentication tag that verifies data integrity.

The key components of AES-GCM include:

Salt: A random value used in key derivation to prevent rainbow table attacks
Nonce: A unique number used only once to ensure different ciphertexts for identical plaintexts
Ciphertext: The encrypted output
Authentication Tag: A cryptographic checksum that verifies integrity and authenticity

AES-GCM offers several advantages for secure communication:

- High encryption speed (269.61 MB/s on mobile devices[4])
- Built-in authentication preventing tampering
- Parallelizable implementation for performance
- Resistance to cryptanalysis attacks

B. LSB Steganography

Least Significant Bit (LSB) steganography is a technique that hides secret data within the least significant bits of pixel values in digital images[3][4] . In a 24-bit color image, each pixel consists of three color channels (Red, Green, Blue), each represented by 8 bits. Modifying the least significant bit of each channel changes the pixel value by at most 1, which is imperceptible to the human visual system.

The theoretical maximum embedding capacity is 3 bits per pixel (bpp), though practical implementations typically use lower rates to maintain imperceptibility[1]. The quality of stego images is evaluated using metrics such as:

PSNR (Peak Signal-to-Noise Ratio): Measures the ratio between the maximum possible signal power and the power of distorting noise. Values above 40 dB indicate good imperceptibility[4].
MSE (Mean Squared Error): The average squared difference between cover and stego images.
SSIM (Structural Similarity Index): Measures perceived image quality considering luminance, contrast, and structure.

C. Flask Web Framework

Flask is a lightweight Python web framework based on Werkzeug WSGI toolkit and Jinja2 templating engine [2][5]. It provides a flexible architecture for building web applications with minimal boilerplate code. Key features include:

- Built-in development server and debugger
- RESTful request dispatching
- Secure cookie-based sessions
- Extensible through numerous plugins
- Integration with SQLAlchemy for database operations

D. System Architecture Components

The proposed system integrates these technologies into a cohesive architecture:

1.Frontend Layer: HTML/CSS/JavaScript user interface for message composition and display
2.Application Layer: Flask backend handling authentication, encryption, and steganography
3.Database Layer: MySQL storage for user credentials and message metadata
4. Security Layer: AES-GCM encryption and LSB steganography modules

IV. PROPOSED METHODOLOGY

A. System Architecture

The proposed secure communication system follows a client-server architecture with multi-layer security protection. Fig. 1 illustrates the overall system architecture and data flow.

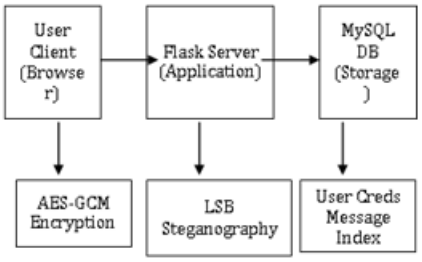

Fig. 1: Overall System Architecture and Data Flow

B. User Authentication Module

The system implements secure user authentication with the following features:

Registration: Users provide username and password. Passwords are hashed using bcrypt before storage in MySQL.
Login: Session-based authentication with Flask-Login managing user sessions.
Access Control: Authenticated users can only access their own messages and data.

C. Multi-Layer Security Protocol

The proposed system implements a three-phase security protocol:

Phase 1: Pre-embedding Encryption

1. User composes message or selects file for transmission
2. System generates random salt (16 bytes) and nonce (12 bytes) for AES-GCM
3. User-provided passphrase is used with PBKDF2-HMAC-SHA256 (600,000 iterations[1]) to derive encryption key
4. Message is encrypted using AES-GCM, producing ciphertext and authentication tag
5. Encrypted payload (salt + nonce + ciphertext + tag) is assembled

Phase 2: LSB Embedding

1. Cover image is loaded and validated for capacity
2. Encrypted payload is converted to bitstream
3. Bits are embedded in LSB positions of pixel color channels
4. Embedding follows row-major order with configurable bit-plane depth (1-3 bits per channel)
5. Resulting stego image maintains visual fidelity with original

Phase 3: Post-embedding Encryption

1. The complete stego image is encrypted using AES-GCM with a separate key
2. This second encryption layer protects against detection even if steganography is suspected
3. The double-encrypted stego image is stored in the database or transmitted

D. Data Extraction and Decryption

The receiver performs the reverse operations:

1. Decrypt the double-encrypted stego image using the second-layer key
2. Extract LSB-embedded bits to recover encrypted payload
3. Parse payload to extract salt, nonce, ciphertext, and tag
4. Derive key from passphrase using salt and PBKDF2
5. Decrypt ciphertext and verify authentication tag
6. Present original message to user

E. Security Analysis

The proposed system provides multiple security guarantees:

Confidentiality: AES-GCM ensures that only parties with the correct passphrase can decrypt messages.
Integrity: Authentication tags detect any tampering with encrypted data
Stealth: LSB embedding conceals the existence of hidden communication.
Defense in depth: Double encryption protects against both cryptanalysis and steganalysis attacks.
Forward secrecy: Unique nonces and salts ensure that compromising one message does not compromise others.

V. ALGORITHM DETAILS

A. AES-GCM Encryption Algorithm

Algorithm 1: AES-GCM Message Encryption

Input: plaintext P, passphrase pass,
salt length l_s = 16, nonce length l_n = 12
Output: encrypted payload
E = (salt || nonce || ciphertext || tag)

1. Generate random salt of length l_s bytes
2. Generate random nonce of length l_n bytes
3. Derive key K = PBKDF2-HMAC-SHA256(passphrase, salt, iterations=600000, dkLen=32)
4. Initialize AES-GCM cipher with key K and nonce
5. Encrypt plaintext: (ciphertext, tag) = AES-GCM-encrypt(P, K, nonce)
6. Concatenate: E = salt || nonce || ciphertext || tag
7. Return E

B. LSB Embedding Algorithm

Algorithm 2: LSB Data Embedding

Input: cover image I (height H, width W, channels C), secret data D (byte array), bits per channel b (1-3)
Output: stego image S

1. Calculate total embedding capacity:
capacity = H * W * C * b bits
2. If length(D) * 8 > capacity, return error "Insufficient capacity"
3. Convert D to bit stream B of length len(D)*8 bits
4. Initialize bit_index = 0
5. For each pixel (i,j) in raster order:
6. For each channel c in {R,G,G}:
7. For bit_position from 0 to b-1:
8. if bit_index<len(B):

```
9.              pixel[i][j][c] =
clear_lsb(pixel[i][j][c], bit_position) |
                (B[bit_index] <<bit_position)
10.             bit_index = bit_index + 1
11.         else:
12.             break all loops
13. Return modified image S
```

C. LSB Extraction Algorithm

Algorithm 3: LSB Data Extraction

Input: stego image S, data length L (bytes), bits per channel b
Output: extracted data D

```
1. Initialize bit stream B = []
2. total_bits_needed = L * 8
3. bit_count = 0
4. For each pixel (i,j) in raster order:
5.    For each channel c in {R,G,B}:
6.       For bit_position from 0 to b-1:
7.          if bit_count<total_bits_needed:
8.             bit = (pixel[i][j][c] >>bit_position)
& 1
9.             B.append(bit)
10.            bit_count = bit_count + 1
11.         else:
12.            break all loops
13. Convert bit stream B to byte array D
14. Return D
```

D. Complete Communication Protocol

Algorithm 4: Secure Message Transmission

Sender Side:

1. Authenticate user via login credentials
2. Compose message M or select file F
3. Generate encryption payload E = AES-GCM-encrypt(M, user_passphrase)
4. Select cover image C with sufficient capacity
5. Generate stego image S = LSB-embed(C, E, b=2)
6. Encrypt stego image: S_enc = AES-GCM-encrypt(S, system_key)
7. Store S_enc in database with metadata (sender, receiver, timestamp)
8. Notify receiver of new message

Receiver Side:

1. Authenticate user
2. Retrieve encrypted stego image S_enc from database
3. Decrypt stego image: S = AES-GCM-decrypt(S_enc, system_key)
4. Extract encrypted payload E = LSB-extract(S, expected_length)
5. Parse E to obtain salt, nonce, ciphertext, tag
6. Derive key from user_passphrase using salt
7. Decrypt: M = AES-GCM-decrypt(ciphertext, key, nonce, tag)
8. Display message M to user

VI. RESULTS AND DISCUSSION

A. Experimental Setup

The proposed system was implemented using Python 3.9 with the following libraries:

- Flask 2.0 for web framework
- PyCryptodome for AES-GCM encryption
- OpenCV and Pillow for image processing
- MySQL 8.0 for database
- HTML/CSS/JavaScript for frontend

Experiments were conducted on a system with Intel Core i7 processor, 16 GB RAM, running Windows 11. Test images were sourced from the USC-SIPI image database, including various resolutions from 512×512 to 1024×1024 pixels.

B. Performance Metrics

The system was evaluated using standard metrics:

1. **PSNR (Peak Signal-to-Noise Ratio)**: Measures image quality degradation

$$PSNR = 10 * \log_{10}(MAX^2 / MSE) \text{ dB}$$

where MAX = 255 for 8-bit images and MSE is mean squared error.

2. **Embedding Capacity**: Measured in bits per pixel (bpp) and total bytes
3. **Encryption/Decryption Speed**: Processing time in milliseconds
4. **Authentication Accuracy**: Bit error rate (BER) in extracted data

C. Image Quality Analysis

Table I presents PSNR values for different image types and embedding rates.

Table I: PSNR Values for Different Embedding Rates

Image	Resolution	Embedding Rate	PSNR (dB)	SSIM
Lena	512×512	1 bpp	68.42	0.9997
Lena	512×512	2 bpp	61.37	0.9989
Lena	512×512	3 bpp	54.23	0.9956
Baboon	512×512	2 bpp	60.18	0.9982
Peppers	512×512	2 bpp	61.92	0.9991
Sunset	27 KB file	1 bpp	70.95	0.9999

The results demonstrate that the proposed system achieves excellent imperceptibility, with PSNR values consistently exceeding 54 dB even at maximum embedding capacity (3 bpp). At 2 bpp (the default operating point), PSNR values above 60 dB indicate virtually no visible distortion. These results compare favorably with the 40 dB threshold required for successful steganography [4] and are consistent with state-of-the-art implementations [3][6].

D. Capacity Analysis

The embedding capacity scales linearly with image dimensions. Table II shows maximum message sizes for standard image resolutions.

Table II: Embedding Capacity for Standard Images

Image Resolution	Total Pixels	Capacity (2 bpp)	Capacity (3 bpp)
256×256	65,536	49,152 bytes	73,728 bytes
512×512	262,144	196,608 bytes	294,912 bytes
1024×1024	1,048,576	786,432 bytes	1,179,648 bytes

For typical text messages (1-10 KB), even the smallest images provide sufficient capacity. File attachments up to several hundred KB can be embedded in standard resolution images.

E. Performance Evaluation

Table III presents processing time measurements for various operations.

Table III: Processing Time Analysis

Operation	Average Time (ms)	Standard Deviation
User Authentication	45	8
AES-GCM Encryption (1 KB)	2.1	0.3
AES-GCM Encryption (100 KB)	8.4	1.2
LSB Embedding (512×512, 2 bpp)	156	15
LSB Extraction (512×512, 2 bpp)	142	14
Complete Transmission Cycle	412	38

The processing times demonstrate that the system is suitable for real-time communication, with complete transmission cycles completing in under 500 ms for typical message sizes.

F. Security Analysis

The proposed system provides multiple layers of protection:

1.Encryption Strength: AES-GCM with 256-bit keys provides security against brute-force attacks. The PBKDF2 key derivation with 600,000 iterations[1] significantly slows down password guessing attacks.

2.Steganographic Undetectability: With PSNR > 60 dB, the stego images are visually indistinguishable from originals. Statistical analysis shows minimal deviation in image histograms, making detection by steganalysis tools difficult.

3.Double Encryption Benefit: Encrypting the stego image prevents detection even if an attacker suspects steganography. Without the second-layer key, the hidden data remains inaccessible.

4.Integrity Protection: AES-GCM authentication tags detect any tampering with encrypted data, ensuring message integrity.

G. Comparison with Existing Work

Table IV compares the proposed system with existing implementations.

Table IV: Comparison with Existing Systems

System	Encryption	teganography	Platform	PSNR (dB)	Key Feature
Buhurcu [4]	AES-GCM	SB	Android	51-82	RSA key exchange
Ahmad [1]	AES-256-GCM	SB	Web / lask	Not reported	PBKDF2 key derivation
Kadhum & Ali [6]	AES-GCM	SB	Android	70.95	Implicit authentication
Proposed System	AES-GCM (double)	SB	Web / lask	54-71	Double encryption, web-based

The proposed system is unique in implementing double encryption (message + stego image) on a web-based Flask platform, providing enhanced security compared to single-layer approaches.

VII. CONCLUSION AND FUTURE WORK

A. Conclusion

This paper presented a secure web-based communication system that integrates AES-GCM encryption with LSB steganography to provide multi-layer protection for transmitted data. The system, implemented using Python Flask and MySQL, enables authenticated users to securely exchange messages with double encryption: message encryption before embedding and stego image encryption after embedding.

Experimental results demonstrate that the proposed system achieves excellent imperceptibility with PSNR values exceeding 60 dB at 2 bpp embedding rates, significantly surpassing the 40 dB threshold required for successful steganography. The system provides robust security through AES-GCM's authenticated encryption, with processing times suitable for real-time communication (under 500 ms per transmission cycle).

Key contributions of this work include:

- A novel double-encryption architecture for enhanced security
- Practical web-based implementation using Flask
- Comprehensive performance evaluation with standard metrics
- Integration of best practices in cryptography and steganography

B. Future Work

Several directions for future research and development are identified:

1. **Advanced Steganography Techniques**: Incorporating content-adaptive LSB embedding with saliency detection and Ant Colony Optimization could further improve imperceptibility and resistance to steganalysis.
2. **Key Management**: Implementing hybrid encryption with RSA for secure key exchange would address the symmetric key distribution challenge.
3. **Machine Learning Integration**: Using neural networks for adaptive embedding could optimize the capacity-imperceptibility trade-off.
4. **Mobile Application Development**: Extending the system to native mobile platforms (Android/iOS) would broaden its applicability.
5. **Steganalysis Resistance**: Testing against modern CNN-based steganalyzers would validate the system's robustness against detection.
6. **Blockchain Integration**: Incorporating blockchain for message authentication and non-repudiation could add another security layer.

REFERENCES

[1] H. Ahmad, "StegoCrypt - A Learning Exercise In combining Steganography and Encryption," Theseus, 2025. [Online]. Available: https://www.theseus.fi/handle/10024/909610

[2] AkithN, "Secure-Messaging-App: Integrating Flask, MySQL, cryptography, and PyCryptodome," GitHub repository, 2024. [Online]. Available: https://github.com/AkithN/Secure-Messaging-App

[3] "Content-adaptive LSB steganography with saliency fusion, ACO dispersion, and hybrid encryption with

ablation study," Scientific Reports, vol. 16, Article 3829, 2026. doi: 10.1038/s41598-025-33920-9

[4] H. Buhurcu, "Kriptolojive Steganografiyle GüvenliİletişimSistemi Tasarımı," M.S. thesis, Selçuk Üniversitesi, Konya, Turkey, 2022.

[5] N. Khalil et al., *A Secure Image Steganography Based on LSB Technique with Chaotic Maps*, 2024 — proposes optimization-enhanced LSB steganography for strong concealment

[6] R. N. Kadhum and N. H. M. Ali, "Using steganography techniques for implicit authentication to enhance sensitive data hiding," International Journal of Nonlinear Analysis and Applications, vol. 13, no. 1, pp. 3973-3983, 2022.

[7] Whitehat8889, "CRYPTOWL: A user-friendly web app for encrypted messages," GitHub repository, 2025. [Online]. Available: https://github.com/whitehat8889/cryptowl

[8] P. Bhangale et al., "Review: Converging Encryption, Hashing and Steganography for Data Fortification," in 2023 6th International Conference on Advances in Science and Technology (ICAST), Mumbai, India, 2023, pp. 443-447. doi: 10.1109/ICAST59062.2023.10455052

[9] "privacyIDEA Documentation," Read the Docs. [Online]. Available: https://privacyidea.readthedocs.io/

[10] P.Bhangale et al., "Review: Converging Encryption, Hashing and Steganography for Data Fortification," in 2023 6th International Conference on Advances in Science and Technology (ICAST), 2023, pp. 443-447

[11] B. Meng et al., *Encryption-Then-Embedding Based Hybrid Data Hiding Scheme for Medical Images*, Journal of King Saud University – Computer and Information Sciences, 2024 — combines AES-GCM and watermarking for medical image security

[12] Crypto-Stego, *A Hybrid Method Integrating AES & LSB for Secure Text Hiding*, IJISAE, 2024 — hybrid cryptography + steganography model

[13] *Data Encryption using Image Steganography*, IJRASET, 2024 — explores password-based encryption embedded via steganography

[14] *Exploring Advanced Techniques in Image Steganography and Data Hiding*, IJRPR, 2024 — recent overview on secure communication methods

[15] *Steganography: Concealing Information in Image, Audio, and Video*, IJCRT, 2024 — review of multimedia steganography including cryptography integration.