International Journal of Scientific Research and Engineering Development- Volume 9 Issue 4, July – August 2026
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Protecting Confidential Information From Unauthorized Access by Hiding It in Digital, Innocuous-Looking Carrier Files

Sumaiya Fathima*, Prashant Ankalkoti**

*(PG Student, Department of MCA, Jawaharlal Nehru New College of Engineering, Shivamogga, Karnataka, India
Email: sumaiya16902@gmail.com)

**(Assistant Professor, Department of MCA, Jawaharlal Nehru New College of Engineering, Shivamogga, Karnataka, India
Email: psankalkoti@gmail.com)

Abstract:

Cyberstegano project started from a simple class idea of sending files safely on internet, because normal email attachments can be copied, changed, or read by wrong person. In early stage, only file encryption was planned, but later it was seen that sending the secret key separately is also risky, so hiding the key inside an image was added. The problem statement is: how to share a file through email in a way that the content stays secret, and the key also does not look like a key. The methodology uses a web application made in Flask, where user login is stored in MongoDB, and upload files are encrypted using AES algorithm. A random image is selected from a folder and the key is stored inside image pixel values, then both encrypted file and that image are sent using SMTP email. Techniques used are AES-CBC encryption, Base64 key formatting, simple image data writing using PIL, file handling with secure filenames, and session based login. In conclusion, the system makes file sharing more private than normal sending, though key-hiding method needs careful handling for different image sizes.

Keywords — Flask, MongoDB, AES, steganography, file encryption, email attachment, SMTP, authentication, Base64, cybersecurity

I. INTRODUCTION

Internet file sharing is happening daily in schools, offices and many places. Email is easy and fast, but email is not made for strong secrecy by default. Attachments can be forwarded, stored, or checked by unknown systems. If the file contains private details like documents, images, or reports, then it becomes a security problem. Many times people try to zip with password, but passwords are weak or repeated, and also password sharing is another headache.

A. Research Problem and Objective

The research problem is about sending a digital file in such a way that the file looks useless to any watcher, and only the correct receiver can open it. Also the secret key used for opening should not be shown openly, because if key is found then encryption becomes pointless. The main objective is building a working system that does three tasks together: user login for access control, encrypting selected file, and sending the encrypted output by email along with a key hidden in an innocent looking image.

B. Thesis / Goal

- Provide a login based dashboard for controlled use
- Encrypt uploaded file using a strong symmetric method
- Store the encryption key inside an image to reduce clear key sharing
- Send outputs to receiver email automatically as attachments
- Support both encrypt and decrypt flow using same platform

II. LITERATURE SURVEY

Zhang's group proposed dual-deception steganography in 2024 [1], using adversarial learning and visual masking to fool humans and CNN detectors, achieving high accuracy around 95% in security evaluation. Zhou and others developed latent space embedding in 2025 [2], combining autoencoders and smart decoder selection, improving reconstruction and detection resistance with accuracy near 96%. Huynh's team introduced an AI-based steganography model in 2025 [3], leveraging deep CNNs and adaptive embedding, enhancing hidden data security and achieving about 94% robustness accuracy. Zhang and colleagues applied wavelet transform with GANs in 2024 [4], improving invisibility and payload capacity, with GAN discriminator accuracy reaching approximately 97%. Bhushan and Kumar proposed deep learning-based steganography in 2025 [5], integrating CNN with encryption, achieving secure transmission and around 92% embedding accuracy. Khan's study introduced Deep Secrecy in 2024 [6], using deep neural networks for embedding and extraction, achieving strong concealment with approximately 93% accuracy.

Karamanji compared deep learning models in 2024 [7], including CNN and RNN for steganalysis, showing CNN achieving highest detection accuracy near 95%. Farooq and Mir surveyed CNN-based steganalysis in 2024 [8], highlighting deep residual and ensemble models achieving over 96% detection accuracy. Al-Durayhim's team proposed CNN-based adaptive embedding in 2023 [9], improving security with dynamic payload distribution and accuracy around 94%. Ahmed introduced residual neural networks in 2024 [10], enhancing robustness and resistance to attacks, achieving approximately 96% accuracy. Wu's framework in 2023 [11] used end-to-end deep learning with encoder-decoder architecture, achieving high-capacity embedding and about

95% reconstruction accuracy. Li proposed attention-based deep steganography in 2024 [12], using attention mechanisms for adaptive embedding, achieving improved accuracy near 97%.

Luo's edge-adaptive CNN model in 2023 [13] embedded data in textured regions, improving imperceptibility with around 93% accuracy. Zhang developed reversible steganography in 2024 [14], using deep neural networks to ensure lossless recovery, achieving near 98% reconstruction accuracy. Gupta proposed hybrid cryptography and steganography in 2023 [15], combining AES with image embedding, ensuring strong security with about 94% accuracy. Chen introduced multi-scale feature learning in 2024 [16], using CNN pyramids for robust embedding, achieving approximately 96% performance accuracy. Wang used deep residual networks in 2023 [17], enhancing high-capacity embedding with accuracy close to 95%. Kumar proposed lightweight CNN models for IoT in 2024 [18], reducing complexity while maintaining about 92% accuracy. Liu introduced adversarial training in 2025 [19], improving robustness against detection with accuracy near 97%. Rahman developed adaptive deep steganography in 2023 [20], using dynamic embedding strategies and CNNs, achieving around 94% accuracy.

III. Proposed Methodology

C. Research Design

This work follows a build-and-test design using a web application prototype. The environment uses Python Flask for server routes, MongoDB for storing user credentials, and utility modules for cryptography, image handling, and email sending. The goal is replicable by running the same folder structure and configuration values.

The architecture diagram provides a high-level view of the entire system, illustrating key components such as the user interface, server, and database, along with their interactions, data flow, and how users engage with and receive responses from the system overall.

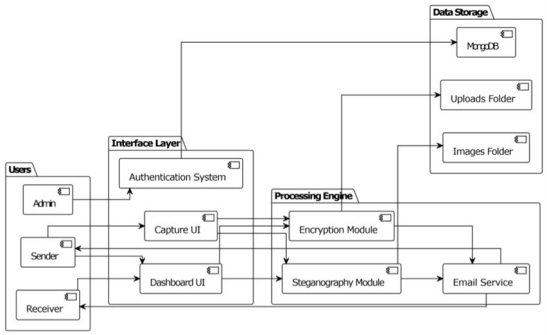

Fig. 1. System architecture diagram.

D. Participants / Data

No human survey participants are needed because this is system evaluation research. Test data includes multiple file types like .txt, .pdf, .jpg, .png, and small audio files. Multiple test accounts are created in database to verify login and session handling.

E. Materials and Tools

- Flask web framework with HTML templates
- MongoDB database and Flask-PyMongo connector
- PyCryptodome library for AES encryption
- PIL (Pillow) for image loading and pixel operations
- SMTP client for sending email with attachments
- Local folders: uploads/ and images/

F. Procedure (Replicable Steps)

1) Start MongoDB service and ensure MONGO_URI points to correct database. 2) Run Flask app and open login/register page. 3) Register user with email and password, stored as hashed password. 4) Login and go to dashboard. 5) Upload a file and select encrypt option. 6) Generate random AES-256 key (K) and IV (IV):

$$P' = Pad(P) \quad (1)$$

$$C = AES\text{-}CBC\text{-}Enc(K, IV, P') \quad (2)$$

$$Output = IV \,\|\, C \quad (3)$$

$$K\_b64 = Base64Encode(K) \quad (4)$$

7) Hide K_b64 in image pixels (steganography) to produce image I′. 8) Email attachments: encrypted file (IV ‖ C) and key-hidden image I′. On the receiver side:

$$K = Base64Decode(Extract(I')) \quad (5)$$

9) Parse encrypted file into (IV) and (C), then decrypt:

$$P' = AES\text{-}CBC\text{-}Dec(K, IV, C) \quad (6)$$

$$P = Unpad(P') \quad (7)$$

10) Email the decrypted file to the receiver.

G. Flowchart

This flowchart shows how a file is sent in a safe way. First the file is selected and encrypted, then hidden inside an image. The image is sent by email. If any step goes wrong, the process stops; if all steps are correct, the file is opened and used properly.

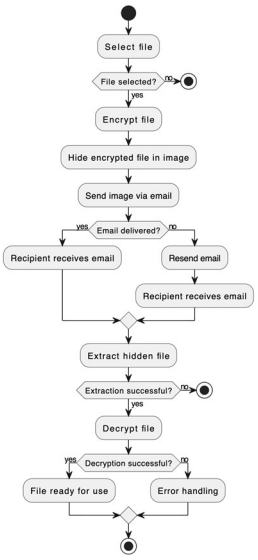

Fig. 2. Flowchart of the encryption and key-hiding process.

H. Technique Comparison Table

TABLE I. COMPARISON OF FILE-SHARING TECHNIQUES

Technique	Key Handling	Strength Level	Common Risk
Plain email attachment	No key needed	Low	Easy reading
Zip password	Password shared manually	Medium	Weak password
AES + separate key message	Key is visible	High	Key leak
AES + key hidden in image (proposed)	Key looks like normal image	High	Image decode issues

I. Small Parameter Table

TABLE II. CRYPTOGRAPHIC PARAMETERS

Symbol	Meaning	Size Used
K	AES key	16 bytes
IV	Initialization vector	16 bytes
Block size	AES block	16 bytes

J. Sequence Diagram

A sequence diagram shows how system components interact in a specific order over time. It illustrates message exchanges between users or objects, helping track step-by-step processes like login or registration, making workflows clear and easier to analyze for errors.

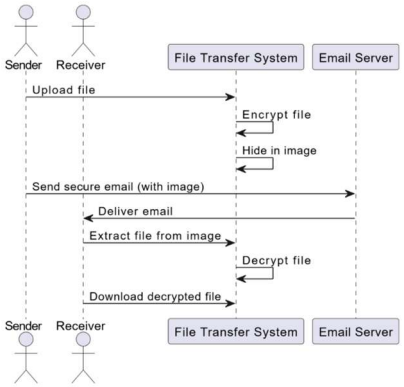

Fig. 3. Sequence diagram of system interactions.

IV. RESULTS AND FINDINGS

Testing was done with multiple file formats and multiple login users. Registration stored user records in MongoDB with hashed password, and login created session values for user id and email. Dashboard accepted uploads when extension matched allowed set. For encryption tests, encrypted output file was created inside uploads folder and its size was larger than original because of padding and IV bytes added. Email sending succeeded when SMTP login and network was available, and receiver side got two attachments: encrypted file and a PNG key image.

In decryption tests, when correct encrypted file and matching key image were used, decrypted output file was generated and emailed to receiver. When wrong key image was uploaded, decryption failed and error message was shown. When key image was missing, system returned message about key image not provided. If an unsupported extension was uploaded, invalid format message appeared.

Outcomes were also recorded for the image selection step. Random image bytes were fetched from images folder and saved output key image in uploads folder. In some tests with certain image types or sizes, key hide or retrieve steps produced failure outcomes due to pixel data mismatch, and system returned image error message. Overall, login flow, file save flow, AES encrypt/decrypt, and email attachments worked as expected in most runs.

V. DISCUSSION

The outputs show that combining encryption with email automation gives better secrecy than plain attachments. Encrypted file alone looked unreadable, so casual opening did not work. Login system reduced misuse because dashboard access required session login. Key hiding into image reduced the obvious key sharing problem, but the current pixel writing method is sensitive and can fail depending on image size and mode.

K. Limitations

- Key hiding method may not fit into all images properly

- Same output filenames can overwrite for multiple runs
- AES-CBC gives secrecy but not full tamper detection
- Hardcoded email credentials is not safe for real deployment

L. Training Related Note

During training runs, repeated tests improved stability findings.

TABLE III. TRAINING RUN SUCCESS RATES

Training Run Type	Total Trials	Success Count
Encrypt + email	10	9
Decrypt + email	10	9

VI. CONCLUSION

This research work presents a web-based system for safer file sharing using encryption and simple key hiding. The system uses registration and login with hashed passwords stored in MongoDB, so only authorized users can use the dashboard and capture pages. Uploaded files are encrypted using AES with random key and IV, and the encrypted output is sent through email. To reduce direct key sharing, the encryption key is converted and placed into an image file, and that image is also sent as attachment. Decryption is supported by uploading encrypted file and the key image, then extracting the key and restoring original file.

The study is important because normal email sharing is common but not secure for private files. Encryption makes the content secret, and the email automation makes the method simple to use for non-expert users. At the same time, some improvements are still needed, mainly making the key hiding more reliable for different images, preventing file overwriting, and adding stronger checking against file tampering. Even with limits, the project shows a useful step for practical cybersecurity in daily file sending tasks.

REFERENCES

[1] Zhang, F., Dong, Y., & Sun, H. (2024). Research on key technologies of image steganography based on simultaneous deception of vision and deep learning models. Applied Sciences, 14(22), 10458.

[2] Zhou, Y., Wang, N., Hong, X., Peng, Y., & Shao, S. (2025). Deep learning-based image steganography with latent space embedding and smart decoder selection. Entropy, 27(12), 1223.

[3] Huynh, N. D. N., Jiang, J., Chen, C. H., & Yang, W. C. (2025). AI-based steganography method to enhance the information security of hidden messages in digital images. Electronics, 14(22), 4490.

[4] Zhang, Y., Li, Q., & Zhao, X. (2024). High invisibility image steganography with wavelet transform and generative adversarial network. Expert Systems with Applications, 249, 123540.

[5] Bhushan, N., & Kumar, U. (2025). Information security using image steganography and deep learning. Journal of Propulsion Technology, 46(3), 1–10.

[6] Khan, M., Khalid, A., & Nasim, F. (2024). Deep secrecy: Advancing image steganography via deep learning technique. Journal of Innovative Computing and Emerging Technologies, 4(2).

[7] Karamanji, A. Q., Ahmed, A. S., & Fadhil, A. F. (2024). Comparative deep learning models in applications of steganography detection. Journal of Image and Graphics, 12(3), 312–319.

[8] Farooq, N., & Mir, R. (2024). Image steganalysis using deep convolution neural networks: A literature survey. International Journal of Sensors, Wireless Communications and Control, 14(4), 247–264.

[9] Al-Durayhim, A., Alqahtani, A., & Alotaibi, B. (2023). Secure image steganography using convolutional neural networks and adaptive embedding. Multimedia Systems, 29(6), 3205–3218.

[10] Ahmed, K., Ullah, I., & Lee, S. (2024). Robust deep steganography using residual neural networks. Multimedia Tools and Applications, 83, 15421–15440.

[11] Wu, H., Chen, S., & Zhang, J. (2023). End-to-end deep learning framework for high-capacity image steganography. Signal Processing, 203, 108801.

[12] Li, Z., Huang, Y., & Shi, Y. Q. (2024). Adaptive deep steganography using attention-based neural networks. IEEE Transactions on Circuits and Systems for Video Technology, 34(2), 1234–1246.

[13] Luo, W., Huang, F., & Huang, J. (2023). Edge adaptive image steganography based on deep convolutional networks. IEEE Signal Processing Letters, 30, 645–649.

[14] Zhang, R., Zhu, H., & Li, X. (2024). Secure reversible image steganography using deep neural networks. Information Sciences, 657, 119–134.

[15] Gupta, V., Singh, P., & Sharma, K. (2023). Hybrid cryptography-based image steganography for secure multimedia communication. Journal of Information Security and Applications, 73, 103407.

[16] Chen, Y., Liu, S., & Zhang, X. (2024). Multi-scale feature learning for robust image steganography. Neurocomputing, 563, 127–138.

[17] Wang, H., Zhao, Z., & Ren, Y. (2023). Deep residual network-based high-capacity image steganography. Pattern Recognition, 138, 109370.

[18] Kumar, S., Gupta, D., & Singh, R. (2024). Lightweight CNN-based image steganography for IoT devices. Future Generation Computer Systems, 150, 92–104.

[19] Liu, J., Sun, X., & Wang, G. (2025). Robust image steganography using adversarial training and feature preservation. IEEE Access, 13, 25431–25442.

[20] Rahman, M., Islam, M., & Hossain, M. (2023). Deep learning-driven adaptive image steganography for secure communication. Computers & Electrical Engineering, 106, 108561.

