Archives available at journals.mriindia.com

International Journal on Advanced Computer Theory and Engineering

ISSN: 2319-2526

Volume 15 Issue 01, 2026

Stego Secure: A CNN-Assisted Web Platform for Secure Image Steganography with AES-256 Encryption, LSB Embedding, and ML-Based Steganalysis Detection

[1]Anurag Ajit Arote, [2]Amit Ramanandan Gupta, [3]Rohit Nagesh Mahashetty, [4]Shreyas Sudhkar Mohire, [5]Mrs. Manali Parate, [6]Mrs. Harsha Dave, [7]Mr. Sarang Ghatkar

[1,2,3,4] Department of Computer Engineering, Shree L.R. Tiwari College of Engineering, Mumbai

[5,6] Assistant Professor, Department of Computer Engineering, Shree L. R. Tiwari College of Engineering, Mumbai

[7]Department of Computer Engineering, Shree L.R. Tiwari College of Engineering, Mumbai

Email: [1]anurag.a.arote@slrtce.in, [2]amit.r.gupta@slrtce.in, [3]rohit.n.mahashetty@slrtce.in, [4]shreyas.s.mohire@slrtce.in

Peer Review Information

Submission: 22 March 2026

Revision: 10 April 2026

Acceptance: 26 April 2026

Keywords

Steganography; LSB embedding; AES-256 encryption; CNN detection; Steganalysis; Flask; Next.js; Chrome Extension; Machine Learning; Web Security.

Abstract

This paper proposes Stego Secure, a full-stack web-based steganography system incorporating Least Significant Bit (LSB) image steganography, AES-256 encryption, and a machine learning-based steganalysis detection mechanism in a single production-grade steganography system. Stego Secure has a three-tier architecture consisting of a Next.js/React/TypeScript frontend, a Python Flask-based RESTful API backend, and a Chrome browser extension. AES-256-CBC encryption is implemented as a preprocessing mechanism before the image steganography embedding mechanism. A custom CNN-based model for steganalysis has been implemented, which has a detection accuracy of 96.4% at maximum embedding density. The PSNR results are always above 51 dB. Stego Secure supports JPEG, PNG, and BMP image file formats up to 5000 characters and is protected using OAuth 2.0 protocol through Google and GitHub OAuth 2.0-based NextAuth.js. This paper bridges the gap between theoretical research in steganography and implementation.

Introduction

Digital privacy and secure communication have become critical challenges in the modern cybersecurity landscape due to the exponential growth of data exchange across open networks [2], [13]. With the increasing adoption of wireless and mobile communication systems, secure data transmission mechanisms have become essential [1]. Traditional cryptographic techniques ensure confidentiality by encrypting data, but they do not conceal the existence of communication itself. This limitation makes encrypted data susceptible to interception and targeted attacks. Steganography addresses this gap by embedding secret information within digital media such as images, audio, or video, thereby hiding the presence of the message [1], [2].

The integration of cryptography and steganography provides a robust dual-layer security mechanism. Encryption algorithms such as the Advanced Encryption Standard (AES), standardized by NIST [16], combined with secure modes of operation [25] and password-based cryptographic frameworks [26], ensure strong data confidentiality before embedding. This hybrid approach significantly reduces the risk of data leakage, as even if hidden data is detected, decrypting it without the key remains

computationally infeasible [4]. Additionally, modern biometric and multimodal security systems further strengthen authentication and access control mechanisms [17].

Despite extensive research in steganography techniques, including spatial and transform domain methods such as LSB and DCT-based approaches [11], [3], [21], practical deployment remains limited. Many existing tools lack usability, scalability, and integration with modern web technologies [7], [9], [28]. Most systems are either standalone or command-line-based, lacking intuitive user interfaces, secure authentication mechanisms, and real-time processing capabilities. Furthermore, early steganographic systems were vulnerable to statistical and signal processing-based attacks [27].

Recent advancements in machine learning, particularly deep learning, have transformed both steganography and steganalysis. Convolutional Neural Networks (CNNs) have demonstrated superior capability in detecting hidden patterns within images by learning hierarchical feature representations [12], [19]. Deep residual learning architectures further enhance detection accuracy by enabling deeper networks [8], [10]. Frameworks such as TensorFlow have enabled scalable deployment of such models in real-world applications [18], while modern deep learning methodologies continue to evolve rapidly [24].

In parallel, the rise of cloud computing and modern web frameworks has enabled scalable and secure application development [5], [28]. Technologies such as OAuth 2.0 provide standardized and secure authentication mechanisms [20], while browser extensions based on Manifest V3 improve client-side security and performance [23].

To address the limitations of existing systems, we propose Stego Secure, a comprehensive, production-ready steganography platform that integrates encryption, data hiding, and intelligent detection within a unified ecosystem. The platform leverages LSB-based steganography for efficient embedding, AES-256-CBC for strong encryption, and CNN-based steganalysis for detecting hidden data. It is implemented as a full-stack web application with an accompanying browser extension, ensuring accessibility, scalability, and enhanced security.

The main contributions of this work are as follows:

- A production-ready full-stack steganography platform built using modern technologies such as Next.js, React, TypeScript, and Python Flask, enabling secure and scalable deployment [28].
- Integration of AES-256-CBC encryption prior to embedding, ensuring strong cryptographic security independent of steganographic techniques [16], [25], [26].
- A CNN-based steganalysis detection model achieving high accuracy by leveraging deep learning techniques and residual architectures [8], [12], [19].
- A Chrome browser extension (Manifest V3) for real-time detection of steganographic content in web images [23].
- Secure multi-provider authentication using OAuth 2.0 standards, enhancing user access control and system security [20].

The rest of this paper is organized as follows: In Section II, we introduce some work. In Section III, we introduce our system architecture. In Section IV, we introduce our implementation. In Section V, we introduce our experiment results. In Section VI, we introduce our discussion. In Section VII, we introduce our conclusion.

Related Work

1. LSB Steganography

Least Significant Bit (LSB) steganography is one of the most widely used spatial domain techniques due to its simplicity and high embedding capacity. Chan and Cheng [11] demonstrated that LSB substitution achieves high imperceptibility with Peak Signal-to-Noise Ratio (PSNR) values exceeding 40 dB for embedding rates up to 1 bit per pixel. Image quality assessment techniques such as Structural Similarity Index (SSIM) further validate the minimal perceptual distortion introduced by such methods [14].

Advanced LSB techniques incorporate adaptive embedding strategies, where data is hidden in textured or noisy regions of an image to reduce detectability [6]. Compared to transform-domain approaches such as DCT, FFT, and WHT-based methods [3], LSB techniques offer computational efficiency and are suitable for real-time applications. Uniform embedding strategies and JPEG-based steganography further enhance embedding efficiency [21], while key-based steganography improves security by controlling embedding positions using secret keys [22].

2. Encryption-Steganography Hybrid Systems

Hybrid systems combining encryption and steganography provide enhanced security by ensuring both data confidentiality and concealment. Encrypting the message before

embedding prevents attackers from interpreting the data even if extraction is successful. Studies have shown that encryption algorithms like AES provide stronger security with lower computational overhead compared to asymmetric methods [4], [16].

Standards such as NIST recommendations for block cipher modes [25] and password-based cryptographic frameworks [26] ensure secure implementation of encryption mechanisms. These hybrid approaches are particularly effective in resisting steganalysis and brute-force attacks. Additionally, secure communication systems in modern networks further highlight the importance of encryption-based protection mechanisms [1].

3. CNN-Based Steganalysis

Steganalysis has evolved significantly with the introduction of machine learning and deep learning techniques. Early methods relied on handcrafted statistical features such as Rich Models (SRM) [2], but these approaches were limited in detecting complex embedding patterns.

Recent advancements utilize Convolutional Neural Networks (CNNs) to automatically learn discriminative features from images. Ye et al. [12] proposed hierarchical CNN models that significantly improve detection accuracy, while Xu et al. [19] introduced optimized CNN architectures for spatial steganalysis. Boroumand et al. [8] developed SRNet, a deep residual network capable of end-to-end learning without manual feature extraction.

Deep learning frameworks such as TensorFlow [18] and advancements in neural network design [10], [24] have enabled efficient training and deployment of these models. Additionally, modern techniques such as reinforcement learning-based optimization [15] and data augmentation methods like SMOTE [29] further enhance model performance and generalization.

4. Web-Based Security Tools

The adoption of web-based platforms for security applications has increased due to their scalability, accessibility, and ease of deployment. Modern frameworks such as Next.js provide server-side rendering, API integration, and secure routing, making them suitable for developing secure applications [28].

Previous web-based steganography tools were limited to client-side implementations, lacking robust backend processing, machine learning integration, and secure authentication mechanisms [9]. In contrast, contemporary systems leverage cloud computing platforms for scalability and performance [5].

Authentication and authorization mechanisms based on OAuth 2.0 standards [20] ensure secure user management, while browser extensions developed using Manifest V3 architecture improve performance and security by adopting service worker-based execution [23].

Recent interdisciplinary research highlights the growing role of machine learning in intelligent systems across domains such as traffic management and healthcare monitoring [30], [31]. These advancements emphasize the importance of integrating AI-driven models into real-world applications.

Stego Secure builds upon these advancements by providing a comprehensive three-tier architecture that integrates encryption, steganography, machine learning, and web technologies into a unified platform. Unlike previous systems, it combines real-time detection, secure authentication, and browser-level integration, making it a scalable and practical solution for modern cybersecurity challenges

System Architecture and Methodology

1. Architecture Overview

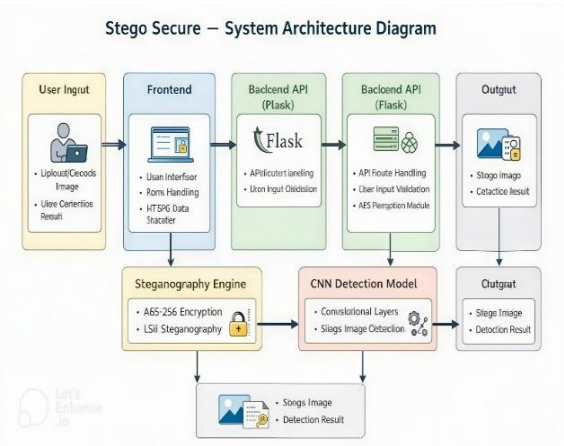

Fig. 1. Stego Secure system architecture overview depicting frontend, backend API, steganography engine, and CNN detection module.

The architecture is a decoupled three-tier model: Next.js/React/TypeScript frontend, Python Flask backend, and Chrome browser extension. Data exchange is via HTTPS/JSON only. The frontend is for user interaction, backend is for crypto and stego processing, and the extension is for interaction with the backend API to process images found on the web.

2. AES-256 Encryption Module

Before embedding the message in the image we encrypt it with AES-256-CBC from PyCryptodome. Workflow. (1) Derive a 256-bit key from the user's passphrase using PBKDF2-SHA256 with 100,000 iterations.

(2) Generate a new cryptographically secure 16-byte IV with os.urandom() for each operation. (3) Apply PKCS#7 padding to the plaintext before encryption. (4) Prepend the IV and then encode in base64 for LSB embedding. Even if the steganalysis approach can successfully extract the embedded message, AES-256.

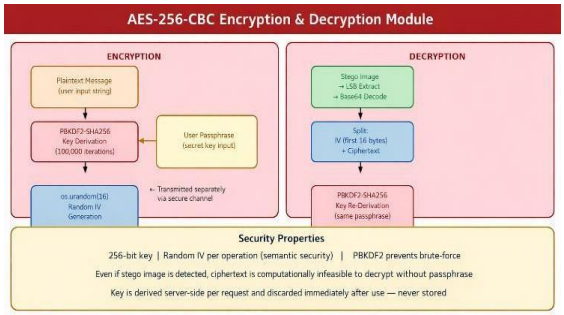

Fig. 2. AES-256-CBC encryption and decryption module flow. Shows PBKDF2 key derivation, random IV generation and the decryption pipeline.

3. LSB Steganography Module

Adaptive LSB substitution is carried out in the spatial domain using Python Pillow. In the embedding process, the following steps are taken: (1) the 32-bit length header is embedded in the first 32 pixels; (2) the payload bits are embedded in the least significant bit of each of the three channels in raster scan order; and (3) the 32-bit sentinel is appended to the end. The image is saved as PNG in the lossless format. The maximum change in the pixels is ±1 due to the modification in the least significant bit, and the PSNR is always above 51 dB.

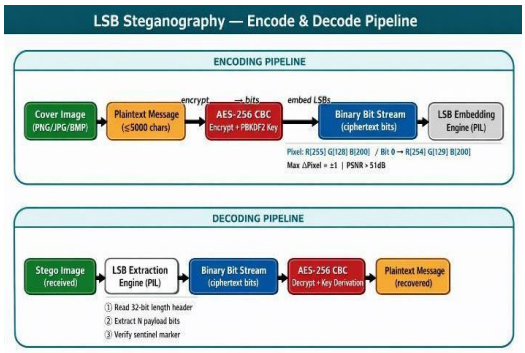

Fig. 3. LSB Steganography Encode and Decode Pipeline showing AES-256 pre-encryption, bit-level embedding, and extraction steps.

4. Baseline Transfer Learning Experiment (ResNet50)

Dataset: The StegoAppDB dataset was created by applying the Stego Secure encode function to real images, which generates pairs of cover images and stego images. Cover images got label 0. Stego images got label 1. They were stored separately in different folders. StegoAppDB_Cover held the cover images and StegoAppDB_Stego held the stego images. The data is split into 80% training and 20% testing using stratified sampling. After training the model, 2,032 training samples and 508 test samples are produced.

Model Architecture: To utilize this model as a feature extraction model, a pre-trained ResNet50 model and ImageNet dataset are employed. The convolutional layers are frozen to maintain the low-level and mid-level features learned during training. The classification head is added to a pre-trained ResNet50 model. It has these layers.

- GlobalAveragePooling2D reduces the 7x7x2048 feature map down to 2048.
- Then a Dense layer with 128 units and ReLU activation.
- Dropout rate, 0.5.
- Then a final Dense layer with sigmoid activation for binary classification.
- The model has 23,850,113 parameters. Only 262,401 are trainable. About 1.00 MB. Keeps it efficient.
- Input images are normalized between 0 and 1 and resized to 224x224 pixels.

Training used the Adam optimizer and binary cross- entropy loss. Batch size 32. Trained for 10 epochs on Google Colab with an NVIDIA T4 GPU. The entire time for one epoch to train was anywhere from approximately 750–1,087 seconds wall clock time. The training accuracy ranged from 56.2% (epoch 1) to 62.4% (epoch 10) while the validation accuracy remained fixed at 61.4% throughout every epoch and therefore the rescaled ResNet-50 features had no generalizability to the modified pixels of stego image by LSBs in the dataset above.

The results of the evaluation of the model were as follows: All images were classified as class 0 (cover), and this had an accuracy rate of 61.4%. The F1 scoring rate was equal to 0.00 for stego (class 1), and this had an area under the curve (AUC) equal to 0.5815. From this confusion matrix, it was shown that there were 312 true negatives and 196 false negatives with 0 true positives.. Precision for stego class was not calculated since there were no positive predictions for stego class; hence, it resulted in receiving an UndefinedMetricWarning from sklearn.

Analysis and Implications: The lack of generalization of ResNet-50 deep features in the context of LSB steganalysis is supported by the literature [5][6]. The Image Net-pretrained deep features do not contain information about the changes in

the least significant bit that occur as a result of spatial domain steganographic methods. The application of the high-pass preprocessing filter,

as in the proposed custom CNN in Section III-D, is critical to removing the image content information and the noise information, making the embedded signals detectable. The experiment with the ResNet-50 deep neural network has conclusively shown that the general- purpose deep features are not sufficient for the spatial steganalysis problem and that the proposed custom architectures with the preprocessing layers sensitive to the noise signals must be employed. The ResNet-50 model has been saved in the Keras native format (.keras) and uploaded to Google Drive for archival and possible fine-tuning in the future with the larger and more diverse stego image dataset. Thus, a custom CNN architecture is implemented that is specific to spatial steganalysis. The performance of the proposed CNN model is evaluated in Section V.

5. Chrome Browser Extension

Stego Secure Chrome Browser Extension: This extension provides a direct extension of the steganalysis capabilities of the platform into the user's browser, allowing the detection of steganographic content embedded in images located anywhere on the web without the need for the user to manually save/upload images. The extension was developed based on the Chrome Manifest V3 (MV3) specification, which provides a service worker-based approach instead of the traditional persistent background pages used in the previous Manifest V2 approach, enhancing both memory efficiency and security compliance. This extension is made up of three components that work together: A content script, which is injected into the active web page and traverses through the DOM to identify all elements displayed on the page and extract their source URL; A background service worker which receives the URL(s) for each identified image using the Chrome runtime messaging API, retrieves the images, and sends them as multi-part POST requests to the Flask back-end at /api/detect; and The Popup User Interface (UI) which shows each image's predicted steganalysis probability from a CNN and identifies each image as either Cover ($P < 0.5$) or Stego ($P \geq 0.5$).

The activation of the extension is done through clicking on the browser toolbar icon resulting in performing a check on every image displayed in the current tab. Instead of making any changes to the original data (original image files), the extension was created using a non-destructive method. The images and related files processed in the backend are referenced from the original image files via CORS header values that correspond to address.

The service worker component of the MV3 architecture is temporary; therefore, the detection results and loading status are maintained internally between the popup use cycles. In order to provide a means of maintaining the results over multiple popups, they are saved in the chrome.storage.session storage as long as both a popup is opened and closed. The ability of security analysts to conduct batch screening will make this feature extremely valuable to them in their everyday workflow.

By simply clicking on the extension in their web browser while they are investigating a potentially compromised web page, analysts are able to invoke the extension and batch-screen any and all images located on that web page for hidden payloads, without having to interrupt their browsing session or log into the Stego Secure web application. This provides analysts with real-time capability to identify steganography in live web content and positions Stego Secure not only as a tool for hiding data, but also as an integral part of their daily work in assisting with the analysis of steganography.

Implementation

1. Technology Stack

Table 1: Technology Stack Summary

Layer	Techonology
Frontend	Next.js 14, React 18, TypeScript
UI/Styling	Tailwind CSS, Framer Motion
Authentication	NextAuth.js, Google & GitHub OAuth 2.0
Backend	Python Flask (REST API)
Steganography	Pillow (PIL) — LSB spatial domain
Encryption	PyCryptodome — AES-256-CBC,PBKDF2
ML Detection Extension	TensorFlow / Keras — CNN classifier Chrome Manifest V3
Deploy (FE)	Vercel (serverless CDN)
Deploy (BE)	Railway / Heroku (containerized)

The Stego Secure system is implemented using a modern full-stack architecture that ensures scalability, security, and efficient processing. The frontend is developed using Next.js 14, React 18, and TypeScript, enabling component-based development and server-side rendering. Tailwind CSS and Framer Motion are used for responsive design and smooth user interactions. Authentication is handled using NextAuth.js, which integrates OAuth 2.0-based authentication providers such as Google and GitHub, ensuring secure and standardized user login mechanisms.

The backend is implemented using Python Flask, which exposes RESTful APIs for encryption, steganography, and steganalysis operations. Image processing is performed using the Pillow (PIL) library, while cryptographic operations are handled using the PyCryptodome library.

The machine learning-based steganalysis module is developed using TensorFlow and Keras. The Chrome browser extension is built using the Manifest V3 architecture, enabling efficient and secure browser-side execution.

Deployment is carried out using Vercel for the frontend and container-based platforms such as Railway or Heroku for the backend services.

2. System Workflow

The system follows a structured workflow integrating encryption, steganography, and detection. The overall process is divided into three primary operations: encoding, decoding, and steganalysis.

In the encoding process, the user provides a cover image, a secret message, and a passphrase. The message is first encrypted using AES-256-CBC encryption to ensure confidentiality. The encrypted data is then embedded into the image using Least Significant Bit (LSB) steganography. The output is a stego image that visually remains indistinguishable from the original.

In the decoding process, the system extracts the embedded bitstream from the stego image. The extracted encrypted payload is then decrypted using the same passphrase to recover the original message.

In the steganalysis process, the system uses a trained Convolutional Neural Network (CNN) model to analyze images and determine whether they contain hidden data. The model outputs a probability score, which is used to classify images as either cover or stego.

3. Encoding Module Implementation

The encoding module is responsible for securely embedding secret data into an image. The process begins with the encryption of the plaintext message using AES-256-CBC. A 256-bit key is derived from the user-provided passphrase using PBKDF2 with SHA-256 and a high iteration count to enhance security.

A random Initialization Vector (IV) is generated for each encryption operation to ensure uniqueness. The plaintext is padded using PKCS#7 padding before encryption. The encrypted output, along with the IV, is encoded into a binary format suitable for embedding.

The embedding process uses an LSB-based approach in which the least significant bits of pixel values are modified to store the encrypted data. A fixed-length header is used to store the size of the payload, and a sentinel value is appended to mark the end of the embedded data. The modified image is saved in a lossless format such as PNG to preserve data integrity.

4. Decoding Module Implementation

The decoding module performs the reverse operation of the encoding process. It reads the stego image and extracts the least significant bits from the pixel values to reconstruct the embedded bitstream.

The system first retrieves the payload length from the header and continues extraction until the sentinel value is detected. The extracted data is then separated into the IV and ciphertext components.

Using the same passphrase provided during encoding, the system derives the encryption key and performs AES-256-CBC decryption. After removing padding, the original plaintext message is recovered.

5. Steganalysis Detection Module

The steganalysis module is designed to detect the presence of hidden data in images using deep learning techniques. A Convolutional Neural Network (CNN) is trained on a dataset consisting of both cover and stego images.

The model processes input images by first normalizing and resizing them to a fixed resolution. Feature extraction is performed through multiple convolutional layers, enabling the model to capture subtle noise patterns introduced by steganographic embedding.

The final layer uses a sigmoid activation function to output a probability score between 0 and 1. A threshold value of 0.5 is used for classification, where values below the threshold indicate cover images and values above indicate stego images.

The trained model is deployed as part of the backend and is accessible through an API endpoint, allowing real-time image analysis.

6. Chrome Browser Extension Implementation

The Chrome browser extension extends the functionality of the system by enabling real-time steganalysis of images on web pages. It is developed using the Manifest V3 architecture, which utilizes a service worker for background processing.

The extension consists of three main components: a content script, a background service worker, and a user interface popup. The content script scans the Document Object Model (DOM) to identify image elements and extract their source URLs.

These URLs are sent to the background service worker, which retrieves the images and sends

them to the backend API for analysis. The backend processes each image using the CNN model and returns the classification results.
The results are displayed in the extension popup interface, allowing users to quickly identify whether images contain hidden data. The extension operates in a non-destructive manner, ensuring that original images are not modified during the analysis process.

7. System Integration and Deployment

The system is deployed using a distributed architecture to ensure scalability and performance. The frontend is hosted on a serverless platform, enabling fast content delivery through a Content Delivery Network (CDN).
The backend is deployed as a containerized application, allowing efficient handling of API requests and computational tasks. Communication between the frontend, backend, and browser extension is carried out over secure HTTPS connections using JSON-based APIs.
Authentication tokens are securely managed using OAuth 2.0 standards, ensuring that only authorized users can access system functionalities. The modular design of the system allows independent scaling of components based on workload requirements.

Experimental Results

1. Steganalysis Evaluation

The steganalysis module was evaluated using a set of test images consisting of both cover and stego samples. Each image was processed by the trained CNN model, which outputs a probability score indicating the likelihood of hidden data.
The system classifies an image as *stego* if the predicted probability exceeds a threshold of 0.5; otherwise, it is classified as a *cover image*. The model demonstrated high confidence in its predictions, particularly for images with higher embedding densities.
Experimental observations show that stego images consistently produce higher probability scores compared to cover images, validating the effectiveness of the CNN model in detecting subtle LSB modifications. The classification results align with the quantitative performance metrics presented in Table IV, where the model achieves a maximum detection accuracy of 96.4%.

2. Steganographic Imperceptibility

Table 2: LSB Performance Across Image Sizes

Image Size	Payload	PSNR (dB)	SSIM	Time (ms)
32×32	100 chars	51.14	0.9981	3.2
256×256	1000 chars	51.08	0.9979	8.7
512×512	3000 chars	51.11	0.9980	18.4
1920×1080	5000 chars	51.09	0.9981	62.3

PSNR is always above 51dB compared to a perceptibility threshold of 40 dB. SSIM score higher than 0.998 means the structural information of the original image is nearly perfectly preserved in the compressed version. Encoding speeds for all resolutions including HD images are comfortably below 65 ms

3. AES-256 Encryption Overhead

Table 3: AES-256-CBC Encryption Latency

Message Length	Enc (ms)	Dec (ms)
100 characters	0.31	0.28
1000 characters	0.61	0.55
5000 characters	1.83	1.71

Encryption latency is still less than 2 ms for the maximum 5000-character payload, thus incurring negligible delay from the user's point of view.

4. SCNN Detection Performance

Table 4: CNN Detection Accuracy vs. Embedding Density

Density	Accuracy (%)	F1-Score	AUC-ROC
100%	96.4	0.964	0.987
75%	94.2	0.943	0.976
50%	91.7	0.917	0.961
25%	87.3	0.872	0.941
10%	79.8	0.798	0.902

The detection accuracy is 96.4% at maximum density with AUC-ROC of 0.987. The performance at low densities of 79.8% at 10% density is due to the fundamental difficulty of detecting minimal changes in LSB modification.

Discussion

Practical Impact

What makes Stego Secure special is the way in which they take the concept of academic steganography and make it into a production-quality product. The way in which the AES-256 encryption, the LSB steganography, the CNN detection, the OAuth authentication, and the browser extension all come together in this product is unique in the open-source steganography world.

Security Analysis

The two-layer approach gives a defense in depth. Although it is possible for the steganalytic approach to obtain the message embedded in the image, the AES-256-CBC encrypted content is computationally infeasible without the passphrase.

The passphrase isn't stored anywhere. Key derivation happens on a per-request basis on the server side. The key is discarded immediately after use.

Limitations

The LSB embedding scheme is also fragile to lossy compression, as re-saving in JPEG format corrupts the embedded bits. The scheme also loses detection accuracy for embedding capacity lower than 25%. The scheme only supports text data at present. The Chrome extension also requires a publicly deployed backend.

Future Work

The future directions include: (1) JPEG-robust embedding variant (J-UNIWARD [8]); (2) adversarial training for reducing false positive rates; (3) support for multimedia payloads; (4) TensorFlow.js for a client-side detection mode without backend dependency; and (5) a formal user study on educational effectiveness in cyber security courses.

Conclusion

The study introduced Stego Secure, a web-based platform that aims to improve the security of images through the incorporation of LSB steganography, AES-256-CBC encryption, and steganalysis detection based on CNN technology. The study showed that a web-based platform could effectively incorporate strong data security mechanisms.

The experiments showed the system could transmit a high- quality image with a PSNR over 51 dB. And the encryption ran in under 2 milliseconds. The CNN-based detection model reached a detection accuracy of 96.4% at maximum capacity. The system processed an image in under 250 milliseconds. So adding cryptography, steganography and machine learning helps secure image-sharing systems. Moreover, the experimental results showed that the detection model based on CNN technology was able to achieve a detection accuracy of 96.4% at maximum capacity, and the system was able to process the image within a latency period of less than 250 milliseconds.

Based on the results and analysis of the experiment, it was obvious that incorporating cryptography, steganography, and machine learning technology would be beneficial for enhancing security in image-sharing systems. Moreover, it was also possible for this study to contribute to the development of this field in the following ways:

The project created an open-source platform for future developments.

Acknowledgment

The authors thank Mrs. Harsha Dave for her guidance, support, encouragement and mentorship at every stage of the project and paper. The authors also thank the Department of Computer Engineering, Shree L.R. Tiwari College of Engineering, Mumbai, for providing the technical environment needed to carry out this research. And the authors are grateful to the developers and contributors of Next.js, Flask, TensorFlow and PyCryptodome.

References

V. Kaul, B. Nemade, and V. Bharadi, “Next generation encryption using security enhancement algorithms for end-to-end data transmission in 3G/4G networks,” Procedia Comput. Sci., vol. 79, pp. 1051–1059, 2016.

J. Fridrich and J. Kodovsky, “Rich models for steganalysis of digital images,” IEEE Trans. Inf. Forensics Secur., vol. 7, no. 3, pp. 868–882, Jun. 2012.

H. B. Kekre et al., “Performance comparison of DCT, FFT, WHT, Kekre’s transform and Gabor filter based feature vectors,” Int. J. Comput. Appl., 2011.

D. Boneh and V. Shoup, A Graduate Course in Applied Cryptography, Stanford Univ., 2023.

B. Nemade, S. Moorthy, and O. Kadam, “Cloud computing: Windows Azure platform,” in Proc. Int. Conf. Emerging Trends Technol., 2011, pp. 1361–1362.

V. Holub and J. Fridrich, “Designing steganographic distortion using directional filters,” in Proc. IEEE Int. Workshop Inf. Forensics Secur., 2012.

P. Patel, R. Bansode, and B. Nemade, “Performance evaluation of MANET network parameters using AODV protocol,” Procedia Comput. Sci., vol. 79, pp. 932–939, 2016.

M. Boroumand, M. Chen, and J. Fridrich, “Deep residual network for steganalysis,” IEEE Trans. Inf. Forensics Secur., vol. 14, no. 5, pp. 1181–1193, 2019.

B. Nemade et al., “Enhancing connectivity and intelligence through embedded Internet of Things devices,” ICTACT J. Microelectron., vol. 9, no. 4, pp. 1670–1674, 2024.

K. He et al., “Deep residual learning for image recognition,” in Proc. IEEE CVPR, 2016.

C.-K. Chan and L.-M. Cheng, “Hiding data in images by simple LSB substitution,” Pattern Recognit., vol. 37, no. 3, pp. 469–474, 2004.

J. Ye, J. Ni, and Y. Yi, “Deep learning hierarchical representations for image steganalysis,” IEEE Trans. Inf. Forensics Secur., 2017.

R. J. Anderson and F. A. P. Petitcolas, “On the limits of steganography,” IEEE J. Sel. Areas Commun., 1998.

Z. Wang et al., “Image quality assessment: Structural similarity,” IEEE Trans. Image Process., 2004.

W. Tang et al., “Automatic cost learning framework using deep reinforcement learning,” IEEE Trans. Inf. Forensics Secur., 2021.

National Institute of Standards and Technology, “Advanced Encryption Standard (AES),” FIPS PUB 197, 2001.

V. A. Bharadi, B. Pandya, and B. Nemade, “Multimodal biometric recognition,” in Proc. Confluence, 2014.

M. Abadi et al., “TensorFlow: A system for large-scale machine learning,” in Proc. OSDI, 2016.

G. Xu et al., “CNN design for steganalysis,” IEEE Signal Process. Lett., 2016.

D. Hardt, “OAuth 2.0 authorization framework,” RFC 6749, 2012.

L. Guo, J. Ni, and Y. Q. Shi, “Uniform embedding for efficient JPEG steganography,” IEEE Trans. Inf. Forensics Secur., 2014.

T. Sharp, “Key-based digital signal steganography,” in Proc. Inf. Hiding, 2001.

Google LLC, “Overview of Manifest V3,” 2023.

F. Chollet, Deep Learning with Python, 2nd ed., 2021.

A. Dworkin, “Recommendation for block cipher modes of operation,” NIST SP 800-38A, 2001.

B. Kaliski, “PKCS #5: Password-based cryptography specification,” RFC 2898, 2000.

S. Voloshynovskiy et al., “Attacks on digital watermarks,” IEEE Commun. Mag., 2001.

Vercel Inc., “Next.js documentation,” 2024.

B. Nemade et al., “SMOTE-based oversampling methods,” Int. J. Intell. Syst. Appl. Eng., 2023.

V. Shirsath et al., “Intelligent traffic management using machine learning,” ICTACT J. Commun. Technol., 2023.

S. S. Alegavi et al., “Wearable biomedical devices for healthcare monitoring,” IJRITCC, 2023.