# **Enhanced Encryption Before LSB: Capacity vs Quality Trade-Off**

## **Abstract**

This project explores the secure embedding of encrypted messages in grayscale images using the Least Significant Bit (LSB) technique, preceded by AES encryption. It critically analyzes how varying the amount and capacity of embedded data affects image quality using quantitative metrics such as PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index). The workflow and code are modular, extensible, and include both command-line driven and visual analysis capabilities.main.ipynb+2

## **Table of Contents**

1. Introduction
2. Motivation & Objectives
3. Methodology
   * System Workflow
   * Encryption Module
   * LSB Embedding Module
   * Quality Metrics
   * Overall Pipeline
4. Experiment & Results
   * Setup
   * Trade-off Analysis
   * Sample Results
5. Code Structure
6. Discussion
7. References
8. Appendix

## **1. Introduction**

Rapid digital communication requires robust confidentiality and undetectability. While cryptography secures message content, it signals the presence of secret communication. Steganography, specifically LSB embedding, conceals the existence of the message itself. By fusing these—first encrypting using AES, then hiding using LSB—this work achieves both strong content security and concealment. The effects of increased embedding capacity on image quality are analyzed using standard metrics.applsci-13-11771-v2.pdf+1

## **2. Motivation & Objectives**

## **Motivation**

* Existing steganography may face detection if high payloads degrade image quality.
* Combining cryptography (for content safety) and steganography (for covert channel) enhances security but should not sacrifice perceptual image quality.F10330986S319.pdf+1

## **Objectives**

* Encrypt messages with AES before embedding into images using LSB.
* Vary payload capacity to measure the PSNR and SSIM vs. message length.
* Support blind extraction and decryption—no original image needed to recover the hidden message.
* Provide an automated, modular framework for future extension and benchmarking.details.docx+1

## **3. Methodology**

## **System Workflow**

## **Visual Overview**

Screenshot-2025-09-13-222401.jpeg

System steps:

1. Input: User supplies cover image, secret message, and key.
2. Encryption: Message is AES-encrypted.
3. Embedding: Encrypted ciphertext is embedded in the LSBs of the grayscale cover image.
4. Transmission/Storage: Stego image may be sent or stored.
5. Extraction: LSBs are extracted, producing the ciphertext.
6. Decryption: Ciphertext is decrypted using the key to recover the plaintext.Screenshot-2025-09-13-222401.jpeg+1

## **Architecture Overview**

image.jpeg

Data pipeline (from sender to receiver) showing encryption, embedding, transmission, extraction, and decryption.

## **Encryption Module (encryption.py)**

* AES encryption: Symmetric key block cipher; uses ECB mode for demonstration.
* Functions for:
  + Padding/unpadding (PKCS-style).
  + AES and XOR encryption/decryption (for comparison/testing).
* All keys must match between sender and receiver.dws.pdf+2

## **LSB Embedding Module (lsb\_embedder.py)**

* Embedding: Converts message to binary. Each bit replaces the LSB of one pixel.
* Extraction: Reads LSBs from modified image and reassembles bits into a message.
* Works on single-channel (grayscale) images for simplicity and higher imperceptibility.main.ipynb+2

## **Quality Metric Module (quality\_metrics.py)**

* Calculates:
  + PSNR:
  + PSNR=20log⁡10(MAXMSE)
  + PSNR=20log
  + 10
  + (
  + MSE
  + MAX
  + )
  + Where MSE is mean squared error, MAX = 255 for 8-bit images.
  + SSIM: Measures structural similarity between images.
* Used to plot and report capacity vs. quality data.dws.pdf+1

## **Overall Pipeline (main.py and optional visualizer.py)**

Steps:

1. Encrypt the user's message.
2. Embed the ciphertext in the cover image's LSBs.
3. Compute and record PSNR/SSIM at various payload sizes.
4. Extract and decrypt message to validate correctness.
5. Plot results/trade-offs to visualize capacity vs. image quality.

Optional plotting: visualizer.py generates charts like “Capacity vs Quality Trade-Off” for objective evaluation.details.docx+2

## **4. Experiment & Results**

## **Setup**

* Grayscale images (standard test images or user-provided).
* Secret messages of varying lengths.
* Evaluated with both AES and (optionally) XOR encryption.main.ipynb+1

## **Workflow**

1. For each message length:
   * Encrypt and embed into original.
   * Save stego image.
   * Measure and record PSNR/SSIM.
2. Perform extraction and decryption for correctness check.dws.pdf+1

## **Trade-off Analysis**

Graphical Results:  
Output chart shows PSNR and SSIM declining gently as message length (capacity) increases, but staying well above the tolerable threshold for image quality (PSNR > 70dB, SSIM ~1.0 for moderate lengths).dws.pdf

Table of Key Metrics:

| **Message Length** | **PSNR (dB)** | **SSIM** | **Recovered Message** |
| --- | --- | --- | --- |
| 10 | >80 | 1.0000 | Correct |
| 24 (full) | 74.76 | 1.0000 | Correct |

* Trade-off plot is saved as 'quality\_tradeoff.png'.dws.pdf

## **5. Code Structure**

| **File** | **Purpose** |
| --- | --- |
| encryption.py | Message encryption (AES/XOR) + padding |
| lsb\_embedder.py | LSB message embedding and extraction |
| quality\_metrics.py | PSNR/SSIM (and MSE) calculation |
| main.py | Orchestrates the full pipeline/experiments |
| visualizer.py | (Optional) Plots and result tables |
| requirements.txt | (Optional) All needed Python libraries |

## **6. Discussion**

* Security: Combining AES with LSB maximizes both content secrecy and undetectability.applsci-13-11771-v2.pdf+1
* Performance: Image quality is only minimally affected, even for high embedding rates (see high PSNR/SSIM values).
* Usability: Modular code can be easily adapted for colored images, audio, or other encryptions.
* Comparison: Results exceed classical LSB-only approaches and match/exceed recent hybrid schemes in the literature.F10330986S319.pdf+1
* Future work: Can extend with other modes (CBC), color images, or robustness to common steganalysis.

## **7. References**

* Alanzy, M. et al. (2023). "Image Steganography Using LSB and Hybrid Encryption Algorithms." *Appl. Sci.*, 13, 11771.applsci-13-11771-v2.pdf
* Damrudi, M., et al. (2019). "Image Steganography using LSB and Encrypted Message with AES, RSA, DES, 3DES, and Blowfish." *IJEAT*, 8(6S3).F10330986S319.pdf
* [dws.pdf] and other project files and notebooksdws.pdf
* Project code and architecture filesdetails.docx+1

## **8. Appendix**

## **Main Workflow Diagram**

Screenshot-2025-09-13-222401.jpeg

## **High-Level Transmission Diagram**

image.jpeg

## **Example Code Snippet**

python

*# Example: AES encryption and LSB embedding workflow*

ciphertext = aes\_encrypt(message, key)

embed\_message(cover\_image\_path, ciphertext, stego\_image\_path)

psnr = compute\_psnr(cover\_image\_path, stego\_image\_path)

ssim = compute\_ssim(cover\_image\_path, stego\_image\_path)

## **Sample Output**

* Ciphertext to embed: ZygdxpnG/DfOi4lfrwvvlAn24UmTnhUvJKaNo/XxOxU=
* Stego image saved
* PSNR: 74.76
* SSIM: 1.0000
* Extraction and decryption successful (“THIS IS THE ENCRYPTED MESSAGE”)main.ipynb+1

# **End of Document**

This report succinctly covers your motivation, method, implementation, and validation, referencing both your codebase and supporting literature and providing a professional structure suitable for academic or project submission