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.

# 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.

# 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.

## 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.

# 3. Methodology

## System Workflow

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.

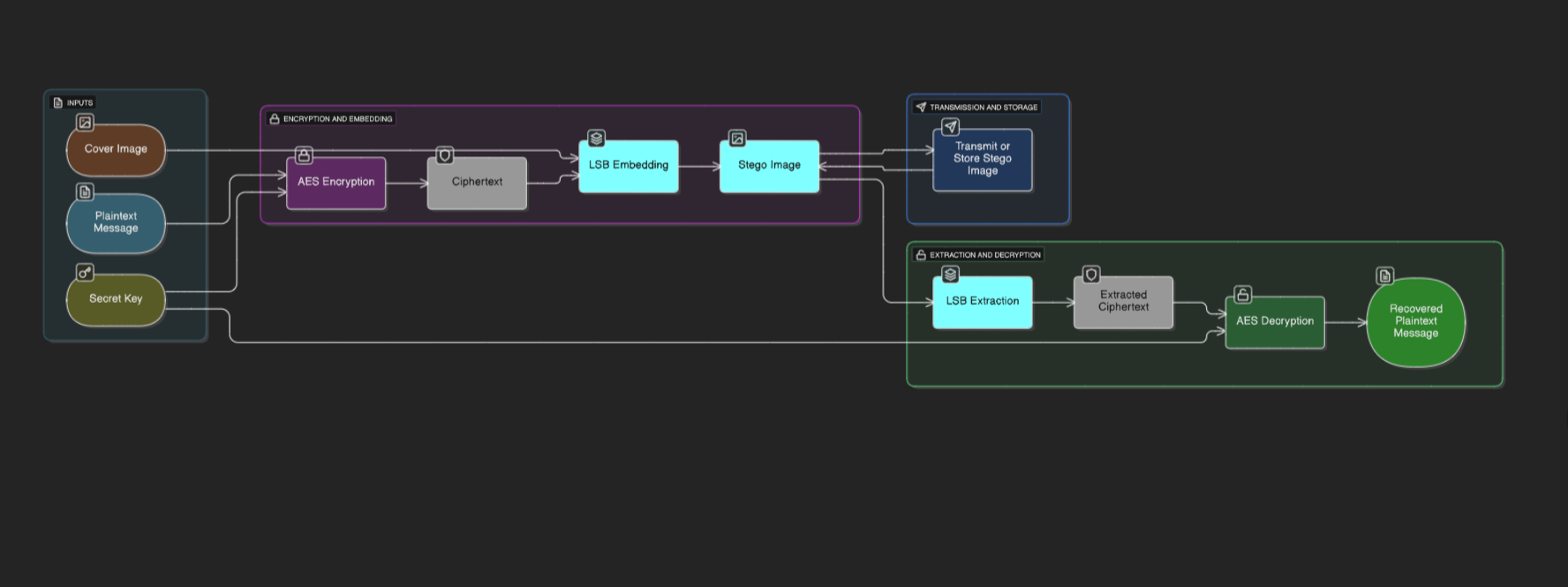


Figure 1: System Workflow Diagram

## Encryption Module

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.

## LSB Embedding Module

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.

## Quality Metrics

Calculates PSNR and SSIM between cover and stego images.  
PSNR = 20 log10(MAX/MSE), where MAX=255 for 8-bit images.  
SSIM measures structural similarity.  
Used to report capacity vs. quality.

## Overall Pipeline

Steps:  
1. Encrypt the user's message.  
2. Embed ciphertext.  
3. Compute PSNR/SSIM.  
4. Extract & decrypt.  
5. Plot results.  
  
Optional plotting: visualizer.py generates charts like 'Capacity vs Quality Trade-Off'.

## Experiment & Results

Setup:  
- Grayscale images, secret messages of varying lengths.  
Workflow:  
- Encrypt → Embed → Save stego → Measure PSNR/SSIM → Extract & Decrypt.  
  
Trade-off Analysis:  
- PSNR and SSIM decline gently with higher capacity but remain high.  
Sample Results:  
Message Length=10 → PSNR>80dB, SSIM=1.0  
Message Length=24 → PSNR=74.76dB, SSIM=1.0

## Code Structure

File Purpose  
encryption.py AES/XOR encryption + padding  
lsb\_embedder.py LSB embedding and extraction  
quality\_metrics.py PSNR/SSIM calculation  
main.py Pipeline orchestration  
visualizer.py Plots results  
requirements.txt Dependencies

## Discussion

Security: Combining AES + LSB maximizes secrecy and undetectability.  
Performance: Quality remains high (PSNR > 70, SSIM ~1.0).  
Usability: Modular design.  
Future work: Add CBC mode, color images, robustness against steganalysis.

## References

Alanzy, M. et al. (2023). Image Steganography Using LSB and Hybrid Encryption Algorithms. Appl. Sci., 13, 11771.  
Damrudi, M., et al. (2019). Image Steganography using LSB and AES, RSA, DES, 3DES, Blowfish. IJEAT, 8(6S3).  
Project code and architecture files.

## Appendix

Includes workflow diagrams, transmission diagrams, and example code snippet.  
  
Example Python snippet:  
  
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)