MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Faculty of Engineering and Technology

CSF20060: Project Based Learning-I

A.Y. 2025-26 Semester: III Panel: A Batch: A2

Project Report

Title: Huffman Coding + Image Steganography

Group Id: 4

1. Shreyas Newe - 1262241387
2. Sreeraj Rajeevan - 1262241335
3. Adwait Bhalerao - 126224169

Introduction

In today's digital era, the secure transmission of confidential data over the internet has become a growing concern. Encryption techniques like AES or RSA protect the content of a message but make the presence of hidden data obvious, which can attract the attention of potential attackers. Steganography, on the other hand, hides data in plain sight — embedding it within innocuous media such as images, videos, or audio files. By combining this with compression algorithms, we can not only secure the data but also reduce its size for efficient storage and transmission. This project presents a hybrid approach that integrates **Huffman Coding** and **Least Significant Bit (LSB) Steganography**. Huffman coding compresses the text message to minimize size, and LSB steganography embeds this compressed binary data into an image. The system is implemented using **Python** and **C++**, with a **Flask-based backend API** and optional **React frontend** for user interaction. This combination ensures the solution is efficient, modular, and suitable for modern web applications.

Problem Statement

In a world dominated by online communication, the confidentiality of personal and organizational data is constantly under threat. While encryption methods provide protection, they make the presence of secret data evident, which can trigger interception attempts.
Moreover, storing or transmitting large uncompressed messages can cause bandwidth inefficiencies.
The problem this project aims to solve is **how to securely hide and efficiently store secret text data within digital images** without compromising image quality or arousing suspicion. By combining **Huffman Coding** (for data compression) and **LSB Steganography** (for secure embedding), this project achieves both data protection and storage optimization.

Motivation

The motivation for this project stems from the increasing importance of **data privacy** and **secure communication** in the digital age. With incidents of cyber espionage, data leaks, and unauthorized surveillance on the rise, there is a strong need for techniques that can hide the very existence of secret information. Steganography provides a discreet form of security by embedding hidden data inside common files, while Huffman coding makes data smaller and less predictable, further reducing detectability.
By merging these two techniques, the project aims to create a more **robust**, **efficient**, and **practical** approach for secure communication and data storage.`

Objectives

The project's primary objectives are as follows:

- To **compress textual data** using Huffman coding, reducing its bit length before embedding.
- To **embed compressed binary data** into digital images using the Least Significant Bit technique.
- To ensure **accurate data retrieval** through Huffman decoding after extraction.
- To develop a **Flask-based backend** that automates compression, embedding, and decoding processes.
- To explore **Python–C++ integration** for optimal performance.
- To maintain **image quality** after embedding to avoid visual distortions.

Literature Review

Several researchers have worked on steganographic systems using various embedding techniques. The **LSB substitution method** is one of the most widely used approaches because it is simple, effective, and causes minimal perceptible changes to the image. However, it does not address efficiency in terms of message size.

Huffman coding, introduced by David Huffman, is a classic **lossless compression algorithm** used in text and image compression systems (like JPEG). It replaces frequent characters with shorter binary codes, reducing overall data size.

Earlier projects have used these two methods separately, but few have combined them. The integration of **Huffman coding with LSB steganography** enhances both the **security** and **efficiency** of data hiding. This project aims to demonstrate that combining compression and steganography yields better performance and reliability than using either technique alone.

System Architecture / Block Diagram

The architecture of the proposed system is divided into four major components:

1. **Frontend (React Interface)** – The user uploads an image and inputs a secret message.
2. **Flask Backend (Python)** – Receives input, coordinates the encoding/decoding process, and returns results.
3. **Huffman Coding Module (C++)** – Compresses and decompresses text messages.
4. **Steganography Module (Python)** – Embeds compressed binary data into the least significant bits of the image's pixels.

Workflow:

1. User uploads image and message → Flask /encode endpoint.
2. Huffman coding (C++) compresses the text to a binary stream.
3. LSB module (Python) embeds bits into the image.
4. Flask returns the stego-image to the user.
5. During decoding, the reverse process retrieves and decompresses the message.

This modular approach allows parallel development and provides flexibility for future extensions like encryption or cloud integration.

Technology Used

- **Python (Flask)** – Backend framework handling routes, requests, and integration.
- **C++** – Implements Huffman encoding and decoding logic for high-speed processing.
- **Pillow (PIL)** – Manages image manipulation and bit-level editing.
- **UUID** – Ensures unique session management for temporary files.
- **React.js (optional)** – Provides a user-friendly interface for encoding/decoding operations.

This combination leverages the simplicity of Python for API development and the computational efficiency of C++ for data compression.

Methodology:

- **Input Phase:**
The user uploads an image and provides a secret text message.

- **Compression Phase:**
The message is passed to the C++ Huffman encoder. It analyzes character frequencies, constructs a binary tree, and assigns variable-length codes to each symbol.

- **Embedding Phase:**
The compressed binary data is embedded into the least significant bits of the image's RGB pixel values. Each pixel stores up to three bits (one per channel). The first 24 bits store the length of the message.

- **Output Phase:**
The modified image (stego-image) is saved and sent back to the user.

- **Decoding Phase:**
The LSB extraction module retrieves the bits from the image. The Huffman decoder then reconstructs the original text using the stored frequency table.

- **API Integration:**
Flask provides RESTful endpoints /encode and /decode, connecting the entire process seamlessly between the frontend and backend.

Limitations:

- The system may not perform well on **lossy image formats** like JPEG, as compression can destroy hidden data.

- Embedding large messages in small images can lead to **insufficient storage capacity**.

- Huffman code tables are **session-based**, so decoding requires consistent mapping from encoding.

- The project currently lacks **encryption**, making data theoretically extractable if steganography is detected. Processing time may increase for very large image or message sizes.

Future Scope

- Introduce **encryption algorithms** (like AES) before embedding to provide dual-layer protection.
- Enable **persistent Huffman code storage** for faster encoding/decoding across sessions.
- Expand compatibility with multiple image formats such as BMP, PNG, and JPEG.
- Incorporate **cloud-based storage** for secure file management.
- Develop a **complete web interface** with visual indicators for compression ratio and embedding depth.
- Apply this system to **real-world applications** such as digital watermarking, copyright protection, and secure document transfer

Conclusion

This project successfully demonstrates a secure and efficient data-hiding mechanism through the combination of **Huffman Coding** and **LSB Steganography**. Huffman compression reduces the size of the secret message, allowing more data to be embedded, while LSB steganography conceals it invisibly within image pixels. The system maintains high image quality, efficient data retrieval, and modularity through the use of Python and C++. Flask serves as a robust backend interface, making the entire workflow automated and extensible. Overall, this hybrid approach proves that the integration of data compression and steganography is an effective solution for modern secure communication systems.

Team Contributions

Shreyas Newe – Implemented the encoding and decoding algorithms for image-based steganography.

Sreeraj Rajeevan - Worked on the user interface design, documentation, and final report preparation

Adwait Bhalerao - Researched encryption techniques, integrated data hiding modules, and tested output accuracy

Each team member collaborated throughout the design, development, and testing stages of our project. The combined efforts ensured efficient data hiding using LSB encoding and retrieval with minimal image distortion .

References

1.” A Huffman code LSB based image steganography technique using multi-level encryption and achromatic component of an image” - https://www.nature.com/articles/s41598-023-41303-1 - Scientific Reports, Nature, 2023

Key Contribution : Proposes improved LSB technique using Huffman codes for better security and data hiding capacity

2. “A novel steganography method for image based on Huffman Encoding” - https://ieeexplore.ieee.org/abstract/document/6203290/ - 2012 3rd National Conference on Emerging Trends and Applications in Computer Science, IEEE

Key Contribution : Improves security and quality of stego images using Huffman encoding

3. “Image steganography using LSB and hybrid encryption algorithms”-

https://www.mdpi.com/2076-3417/13/21/11771 - Applied Sciences, MDPI, 2023

Key Contribution Multi-layered security (MLS) algorithm with AES and Blowfish encryption

4. “Huffman coding-based data reduction and quadristego logic for secure image steganography”

https://www.sciencedirect.com/science/article/pii/S2215098625000886 - Engineering Science and Technology, an International Journal, Elsevier, 2025

Key Contribution : Novel quadristego logic for improved stego image quality

5. “ A novel technique for image steganography based on Block-DCT and Huffman Encoding”

https://arxiv.org/abs/1006.1186 - arXiv preprint arXiv:1006.1186, 2010

Key Contribution : Large data hiding capacity using DCT and Huffman encoding

6. “ An image steganography approach based on k-least significant bits (k-LSB)”

https://ieeexplore.ieee.org/abstract/document/9089566/ - 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies

Key Contribution : k-LSB method that extends traditional LSB steganography

7. “Implementation of LSB steganography and its evaluation for various bits”

https://ieeexplore.ieee.org/abstract/document/4221886/ - 2006 1st International Conference on Digital Information Management, IEEE

Key Contribution : Comprehensive evaluation of 2, 4, 6 LSB techniques