Steganography Tool for Text/File Hiding

Abstract

This project is a simple steganography tool that lets users hide secret text messages or files inside digital images. It uses the Least Significant Bit (LSB) method, which makes the hidden data invisible to the human eye. The tool supports both encoding (hiding) and decoding (retrieving) through an web interface built with Flask.

Introduction

With the rise of digital communication, secure data transfer has become essential. Steganography hides information inside normal media file, making the data invisible instead of just unreadable like encryption.

This project hides text or files within images by modifying the Least Significant Bits (LSBs) of image pixels, which makes almost no visible change while securely hiding the data. This allows safe communication and secret file storage in a simple way.

Tools Used

- Programming Language: Python
- Framework:
  - Flask (for backend server and routing)
- Libraries:
  - Pillow (for image processing),
  - Struct (data packing/unpacking),
  - OS & datetime (file management)
- Frontend: HTML, CSS, JavaScript

Steps Involved in Building the Project

1. Project Setup
   - A Flask project structure is created with separate folders for uploads, encoded images, and decoded files.
   - The frontend is designed using HTML and CSS to provide a clean and user-friendly interface.
2. Backend Development (Flask)
   1. Implemented Flask routes:
      - / → Displays homepage
      - /encode → Handles encoding of text/file into an image
      - /decode → Extracts hidden data from stego images
   2. Helper functions written to:
      - Convert between bytes and bits
      - Embed payload bits into pixel values (encoding)
      - Extract bits and rebuild the hidden data (decoding)
   3. File handling saves outputs without overwriting.

3. Encoding (Hiding Data)
    - User uploads a cover image (PNG/BMP recommended).
    - Payload can be entered as text or uploaded as a file.
    - Data is packed with metadata (length and filename if needed).
    - Payload bits are hidden in the image’s pixel LSBs.
    - The stego image is generated and downloaded directly into encoded images folder.
4. Decoding (Extracting Data)
    - User uploads a stego image.
    - Backend extracts LSBs, rebuilds the payload, and unpacks it.
        - If text was hidden → it is shown in the browser.
        - If a file was hidden → it is saved in the decoded folder.
5. User Interface
    - Encode and Decode sections provided.
    - Radio buttons let users choose text or file mode.
    - Flash messages show success or error feedback.

Conclusion

This project shows how steganography can be used to hide sensitive text or files inside images. Using the LSB method, data is hidden without noticeable changes to the cover image. The Flask backend manages the core logic, while the frontend offers a simple and interactive interface.

The implementation demonstrates steganography as a useful complement to encryption for secure communication. It also highlights the need for lossless image formats (PNG/BMP) to prevent data loss or corruption.