Project Report: LSB Steganography Tool for Image/File Hiding

Intern: Muhammad Ali Khan **Date:** June 28, 2025 **Organization**: ELEVATE LABS

1. Introduction

In the digital age, secure communication and data protection are vital. Steganography hides information within other, non-secret data, making its presence imperceptible. This report details the development of a Graphical User Interface (GUI) tool. It uses Least Significant Bit (LSB) steganography to hide and extract information within image files. This project aims to provide practical insight into data concealment for cybersecurity.

2. Abstract

This report outlines the Python-based LSB Steganography Tool. The GUI allows users to embed text or files into PNG/BMP images and extract them later. Its core functionality modifies least significant bits of pixel color channels, causing minimal visual distortion. The application features user-friendly controls, including browsing and drag-and-drop. It demonstrates fundamental steganographic principles, delivering a functional tool for data embedding and extraction across various file types.

3. Tools Used

The LSB Steganography Tool was developed using:

- **Python 3.3.5:** Primary language for logic and GUI.
- **Pillow (PIL Fork):** For image manipulation (pixel access, saving lossless formats).
- **Tkinter:** Python's standard GUI toolkit for building the user interface.
- **tkinterdnd2:** A Python wrapper for enhanced drag-and-drop functionality from the OS.
- **PyInstaller:** Used for packaging the Python application into a standalone executable.
- **Inno Setup:** Utilized for creating a professional installer for the Windows operating system.

4. Steps Involved in Building the Project

Project development followed these phases:

a. Understanding LSB Steganography: Learned LSB technique: altering the least significant bit of image pixels. This causes tiny, imperceptible visual changes to hide data. This method works by altering the least significant bit of each color channel (Red, Green, Blue, and Alpha) in an image's pixels

b. Core Steganography Logic (LSB Implementation):

- **Binary Conversion:** Functions convert text/files to binary and vice versa.

- **Data Embedding:** The hide_data_in_image function embeds binary data into image pixel LSBs. A unique binary *delimiter* marks the data's end. Output is always **PNG** for lossless preservation.
- **Data Extraction:** The extract_data_from_image function reads LSBs, reconstructing data until the delimiter is found.

c. File Handling and Binary Conversion: Utility functions (file_to_bytes, bytes_to_file) handle reading any file as raw binary for embedding and saving extracted binary data back to a file.

d. GUI Development (Tkinter): The user interface was built for ease of use:

- **Tabbed Interface:** "Hide Data" and "Extract Data" tabs organize functions.
- **Input/Output:** Fields for paths and text display, linked with tk.StringVar.
- **File Browsing:** Standard file dialog for easy file selection.
- **Dynamic UI:** Radio buttons control visibility of text/file input fields.

e. Integrating Drag-and-Drop (DND) Functionality:

- Initial direct tkdnd bindings caused compatibility errors. Switched to tkinterdnd2 Python wrapper for reliable DND.
- **Specific Drop Targets:**
    - "Input Image Path" (hide tab): Accepts only *PNG/BMP images*.
    - "File to Hide" (hide tab): Accepts *any file type*.
    - "Extraction Image Path" (extract tab): Accepts only *PNG images*.
    - General tab areas: Route drops to the appropriate image input field.

f. Error Handling and Enhancements: Robust error handling was integrated:

- **Validation:** Checks for missing inputs, empty messages, and data exceeding image capacity.
- **User Feedback:** Clear success, error, and warning message box pop-ups.
- **Lossless Enforcement:** Output images are always PNG to prevent data corruption.

5. Conclusion

The LSB Steganography Tool successfully meets its objective: a functional, user-friendly application for hiding and extracting data in images. It demonstrates steganographic principles through its LSB algorithm and intuitive Tkinter GUI with robust DND. While effective, future enhancements could include data **encryption**, error **correction** mechanisms, and exploring alternative techniques for formats like JPEG. This project provided me a valuable cybersecurity tool development experience.