**Project Report**

**Steganography Tool for Image/File Hiding**

**1. Introduction**

Steganography is the art and science of hiding information within other non-secret data. This project focuses on developing a graphical user interface (GUI) tool that enables users to embed text or files within digital images using the Least Significant Bit (LSB) technique. The tool provides both embedding and extraction capabilities while maintaining the visual integrity of the original image.

Unlike cryptography which makes the message unreadable, steganography hides the very existence of the message. This makes it particularly useful for secure communication where the presence of secret communication itself needs to be concealed.

**2. Abstract**

The Steganography Tool is designed to provide a user-friendly interface for hiding information within digital images. The system works by modifying the least significant bits of the image's pixel data to encode secret messages. The primary objectives include:

* Implementing LSB-based steganography for PNG and BMP formats
* Developing an intuitive GUI with drag-and-drop functionality
* Supporting both text and file embedding
* Providing extraction capabilities with optional decryption
* Maintaining image quality while maximizing payload capacity

The tool demonstrates practical applications of steganography in digital security and serves as an educational resource for understanding data hiding techniques.

**3. Tools Used**

**Python Libraries**

* PIL (Python Imaging Library) - For image processing
* stepic - For steganography operations
* tkinter - For GUI development
* os/sys - For file system operations

**Key Features**

* Drag-and-drop interface
* Support for multiple image formats
* Text and binary file embedding
* Optional password protection
* Visual quality preservation

**4. Steps Involved in Building the Project**

1. **Setting Up the Environment**:
   * Install necessary libraries: **tkinter** for GUI, **PIL** (Pillow) for image processing, **stepic** for steganography, and **pycryptodome** for encryption.
   * Ensure Python is installed and set up on the development machine.
2. **Creating the Main Application Class**:
   * Define the **SteganographyApp** class to encapsulate the application logic and GUI components.
   * Initialize the main window with a title, size, and resizable properties.
3. **Defining Variables**:
   * Create **StringVar** and **BooleanVar** instances to hold user inputs such as image path, secret message, password, and encryption/decryption options.
4. **Setting Up the User Interface (UI)**:
   * Use **ttk.Notebook** to create tabs for embedding and extracting messages.
   * Design the "Embed Message" tab with fields for selecting an image, entering a secret message, and options for encryption.
   * Design the "Extract Message" tab with fields for selecting a stego image and options for decryption.
5. **Implementing Image Selection**:
   * Create functions (**browse\_image** and **browse\_extract\_image**) to allow users to select images from their file system using **filedialog**.
   * Display the selected image in the GUI using **PIL** to open and resize the image.
6. **Implementing Message Embedding**:
   * Create the **embed\_message** method to handle the embedding process:
     + Check for valid inputs (image path and message).
     + Encrypt the message if the user opts for encryption using the **encrypt\_message** method.
     + Use the **stepic** library to embed the message into the image.
     + Save the modified image to a specified location.
7. **Implementing Message Extraction**:
   * Create the **extract\_message** method to handle the extraction process:
     + Check for a valid image path.
     + Use the **stepic** library to extract the hidden message from the image.
     + If the message is encrypted, prompt for a password and decrypt the message using the **decrypt\_message** method.
     + Display the extracted message in the GUI.
8. **Implementing Encryption and Decryption**:
   * Create methods for encrypting (**encrypt\_message**) and decrypting (**decrypt\_message**) messages using AES encryption.
   * Implement a key derivation function (**derive\_key**) to generate a secure key from the user-provided password.
9. **Handling User Interactions**:
   * Implement toggle functions (**toggle\_encrypt\_fields** and **toggle\_decrypt\_fields**) to show or hide password fields based on user selections.
   * Provide feedback to the user through status messages and error handling using **messagebox**.
10. **Running the Application**:
    * Instantiate the **SteganographyApp** class and start the Tkinter main loop to run the application.

**5. Conclusion**

The Steganography Tool successfully demonstrates the practical implementation of LSB-based image steganography. The project highlights several key achievements:

* Effective hiding of data within images with minimal visual impact
* User-friendly interface that simplifies complex steganography operations
* Support for multiple data types (text and files)
* Robust extraction process that accurately recovers hidden data

Future enhancements could include support for additional image formats, improved encryption methods, and capacity optimization techniques. The tool serves as both a practical utility for secure communication and an educational demonstration of steganography principles.