# Project Report: Data Hider - LSB Steganography Tool

**Introduction:**
This project, "Data Hider," presents a graphical user interfiace (GUI) application designed to implement steganography, the art and science ofi hiding a message within another message or object, such that the presence ofi the message is not immediately obvious. Unlike cryptography, which scrambles a message to make it unreadable, steganography aims to conceal the communication entirely. The primary goal ofi this application is to provide a user-firiendly interfiace fior embedding secret text messages into image files and extracting them later. The tool utilizes the Least Significant Bit (LSB) insertion technique, a common and effective method fior digital steganography.

**Abstract:**
The Data Hider is a Python-based desktop application developed using the Tkinter library fior the user interfiace and the stegano library fior core steganographic fiunctionality. It successfiully implements the Least Significant Bit (LSB) steganography technique to hide arbitrary text data within the pixel data ofi a host image (prefierably PNG). The application provides fiour core fiunctions: image loading, data hiding, data revealing, and saving the stego-image (the image containing the hidden data). The modular design ensures clarity between the GUI elements and the underlying data manipulation logic, creating a robust, educational, and practical cyber security utility.

**Tools Used:**

| Tool/Library | Role in Project |
|---|---|
| Python 3.x | The core programming language used fior development. |
| Tkinter | The standard Python interfiace to the Tcl/Tk GUI toolkit, used to build the entire desktop application interfiace. |
| Pillow (PIL) | Python Imaging Library, used fior opening, manipulating, and displaying image files within the Tkinter environment. |
| stegano | A dedicated Python library fior steganography. Specifically, the lsb module was used fior the encoding and decoding logic. |
| Operating System (os) | Used fior managing file paths and providing access to the system's file dialog fior image selection. |

**Steps Involved in Building the Project:**

The project construction fiollowed a standard modular approach, fiocusing first on the user interfiace and then on implementing the core logic fiunctions.

1. GUI Design and Initialization:
   - Window Setup: A Tkinter main window (root) was initialized with a fixed size, title ("Steganography - Hidden Text inside Image"), and a dark background (#2fi4155) fior visual appeal.
   - Frame Layout: The application was divided into fiour main firames (fi, fi2, fi3, fi4) to organize the image display, text input area, and control buttons.
     - fi: Dedicated to displaying the selected image.
     - fi2: Dedicated to the text area fior message input and display, equipped with a scrollbar.
     - fi3 & fi4: Frames fior grouping control buttons (Open/Save and Hide/Show, respectively).
2. Image and File Handling:
   - showimage(): This fiunction uses filedialog.askopenfilename to allow the user to select an image (.png, .jpg). The selected image is loaded using PIL.Image.open and converted to a Tkinter-compatible object (ImageTk.PhotoImage) fior display in the lbl widget.
   - save(): This fiunction saves the image output ofi the steganography process (stored in the global secret variable) as a PNG file named hidden.png.
3. Core Steganography Logic:
   - Hide():
     1. The fiunction retrieves the message string firom the text input area (text1.get(1.0, END)).
     2. It calls lsb.hide(str(filename), message) to embed the retrieved message into the selected image file.
     3. The resulting stego-image object is stored in the global secret variable, ready to be saved.
   - Show():
     1. The fiunction uses the currently loaded image's filename (filename).
     2. It calls lsb.reveal(filename) to extract the hidden message.
     3. The extracted clear_message is then cleared firom the text area (text1.delete) and inserted fior the user to view (text1.insert).
4. Event Binding:
   - Buttons were created fior "Open Image," "Save Image," "Hide Data," and "Show Data" and bound to their respective Python fiunctions (showimage, save, Hide, Show) using the command attribute, making the application fiully interactive.

**Conclusion:**

The Data Hider project successfiully meets its objective ofi creating a fiunctional, cross-platform steganography tool using pure Python libraries. By implementing the LSB method and encapsulating the fiunctionality within a simple GUI, the project serves as an excellent demonstration ofi infiormation security concepts in a practical setting. Potential fiuture work could involve extending fiunctionality to support more robust steganographic algorithms, adding password protection, or incorporating a visual comparison tool to highlight the minimal differences between the host and stego-images. The current implementation is efficient and user-firiendly, providing a solid fioundation fior fiurther development.