STEGANOGRAPHY TOOL FOR IMAGE/FILE HIDING **PROJECT REPORT**

CREATED BY – YASH G JADHAV

INTRODUCTION

Steganography is the technique of hiding information within non-secret data to avoid detection. Unlike cryptography, which makes data unreadable, it conceals the message’s presence. In today’s digital world, image steganography plays a key role in secure communication and data protection. This project develops an easy-to-use tool that lets users hide and extract text within image files for covert communication.

ABSTRACT

This project develops a steganography tool using Python that allows users to hide and extract text messages within images. It employs the Least Significant Bit (LSB) technique to embed data without affecting the image’s appearance. The user-friendly interface built with Tkinter supports formats like PNG, BMP, and JPEG. The tool offers both hiding and extraction features, making it useful for learning and secure communication through images.

TOOLS USED

- **Python 3.x** - Core programming language for application development
- **Tkinter** - Built-in GUI framework for creating the user interface
- **PIL (Python Imaging Library)** - Image processing and manipulation library
- **OS Module** - File system operations and path handling

STEPS INVOLVED IN BUILDING THE PROJECT

Step 1: Environment Setup and Library Installation - The Python environment was prepared, and essential libraries like PIL for image processing and Tkinter for the GUI were installed to support the project.

Step 2: GUI Design and Layout Creation - A user-friendly interface was designed with a left panel for image upload and right sections for hiding and extracting text, styled to appear modern and professional.

Step 3: Image Upload and Display Functionality - Users can select images through file dialogs, and the application displays a preview to confirm the chosen image before performing steganography operations.

Step 4: Text-to-Binary Conversion System - User messages are converted into binary format with proper character encoding and end delimiters to ensure accurate embedding and retrieval.

Step 5: LSB Steganography Implementation - The core algorithm modifies the least significant bits of image pixels to hide binary data while keeping the visual appearance of the image intact.

Step 6: Message Hiding Functionality - Input text is encoded into the selected image using LSB, with error handling to manage oversized messages and options for saving the modified image.

Step 7: Message Extraction Algorithm - The tool reads the least significant bits from image pixels to reconstruct the binary data, which is then converted back into readable text.

Step 8: Testing and Error Handling - Extensive testing ensures the tool works with various image formats and message sizes, while error handling manages file, processing, and input issues.

CONCLUSION

The steganography tool showcases the effective use of LSB-based techniques to hide text in images without visible changes. It combines steganography concepts with practical Python development, resulting in a functional and easy-to-use application. The project serves both educational and real-world purposes, demonstrating how powerful solutions can be built with minimal code.

KEY ACHIEVEMENTS

- Developed a user-friendly GUI with modern styling and easy-to-use drag-and-drop functionality.
- Enabled multi-format support for images including PNG, BMP, and JPEG with seamless handling.

Security Note: This tool is developed strictly for educational and learning purposes to demonstrate steganography concepts. It should not be used for malicious or unauthorized data hiding.