**<StegaSafe: Image-Based Message Encryption>**

**A Major Project Synopsis Submitted to**

****

**Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal**

**Towards Partial Fulfillment for the Award of**

**Bachelor of Technology**

**(Computer Science and Engineering)**

**Submitted By**

**Aaditya Panwar(0827CS211001)**

**Anusha Nagar(0827CS211029)**

**Anushka Patel(0827CS211030)**

**Under the Supervision of**

**Prof. Shraddha Sharma**



**Department of Computer Science and Engineering**

**Acropolis Institute of Technology & Research, Indore**

**July-Dec 2024**

# Introduction of the Project

"StegaSafe: Image-Based Message Encryption" is a project aimed at creating a secure communication tool based on steganography, which is the practice of hiding information within digital media. The objective is to develop an easy-to-use application where users can hide sensitive information within image files in a way that is invisible to the naked eye. This application employs Least Significant Bit (LSB) steganography, one of the most popular and effective methods for concealing data within images.

In a world where cybersecurity threats are increasing, the need for secure communication is more important than ever. Traditional encryption methods can attract attention and be intercepted. In contrast, steganography allows the information to remain concealed, blending into the digital media without raising suspicion. "StegaSafe" addresses challenges like unauthorized access, data tampering, and surveillance by allowing users to hide messages in a secure manner. This project contributes to fields like digital communication, privacy, and data security, ensuring that sensitive information can be shared securely without being detected.

# Objective

The primary objective of **"StegaSafe: Image-Based Message Encryption"** is to develop a secure and efficient method for hiding sensitive information within digital images, using steganography to ensure data privacy. By providing a user-friendly interface, the application will allow both technical and non-technical users to easily encode and decode hidden messages. The project aims to:

* Enable users to hide text messages inside image files with minimal distortion.
* Provide an accessible and intuitive interface for message encoding and decoding.
* Ensure secure communication by keeping the presence of hidden messages undetectable to unauthorized users.
* Facilitate privacy in digital communications where sensitive data must be kept confidential.

Ultimately, the goal is to create an application that not only simplifies steganography but also ensures a high level of data security without compromising the quality of the image files.

# Scope

The scope of **"StegaSafe: Image-Based Message Encryption"** revolves around developing a steganography-based application that allows users to hide and retrieve secret messages from image files. The core functionality will focus on using the Least Significant Bit (LSB) method, where the binary representation of the message is embedded within the least significant bits of the pixel data in images. The project will cover:

* **File Support**: The application will support common image formats, such as PNG and JPG, which are widely used in various platforms.
* **Encoding/Decoding**: Users will be able to encode messages into images and later retrieve them, ensuring the original image quality is retained.
* **User Interface**: A graphical user interface (GUI) built with Kivy will allow for simple user interactions, including image selection, message input, and message extraction.

While the project will focus on encoding text messages in images, it will not delve into advanced encryption algorithms or support for hiding data in other media, such as audio or video files. The project is also limited to local use, meaning that the application will run on individual machines without involving network-based features like remote message encoding/decoding.

# Study of Existing System (200 words)

In the development of **"StegaSafe: Image-Based Message Encryption"**, it is important to analyze existing steganography tools to understand their strengths, weaknesses, and potential gaps in functionality. The following is a comparative study of five popular steganography applications:

1. **OpenPuff**
   * **Problems Addressed**: OpenPuff supports hiding data in a wide range of media formats, including images, audio, and video files.
   * **Advantages**: The software offers robust security features, including multi-layer encryption, password protection, and flexible support for numerous file types.
   * **Disadvantages**: Its user interface is complex, requiring users to have some technical expertise. It may be difficult for beginners to navigate.
   * **Gaps Identified**: OpenPuff's complexity and lack of user-friendly features limit its accessibility for non-technical users.
   * **Reference**: <https://embeddedsw.net/OpenPuff_Steganography_Home.html>
2. **StegHide**
   * **Problems Addressed**: StegHide allows users to hide data within image and audio files, using password-based encryption for enhanced security.
   * **Advantages**: Offers a strong layer of security through password protection, making it suitable for secure data embedding.
   * **Disadvantages**: The absence of a graphical user interface limits its usability to command-line interactions, which can be daunting for less experienced users.
   * **Gaps Identified**: Lack of a user-friendly GUI limits its adoption among non-technical users.
   * **Reference**: <https://steghide.sourceforge.net/>
3. **SSuite Picsel**
   * **Problems Addressed**: A lightweight solution for embedding text into image files, without requiring external encryption tools.
   * **Advantages**: SSuite Picsel is simple to use, free, and lightweight, making it accessible to users who need basic steganography capabilities.
   * **Disadvantages**: It lacks encryption or advanced security features, making the hidden messages more vulnerable to detection.
   * **Gaps Identified**: The absence of encryption and limited steganography functionality restrict its effectiveness for sensitive data.
   * **Reference**: <https://www.ssuiteoffice.com/software/ssuitepicselsecurity.htm>
4. **Hide'N'Send**
   * **Problems Addressed**: Focuses on hiding messages inside images with minimal visual distortion, making the presence of hidden data hard to detect.
   * **Advantages**: It has a simple and intuitive interface, making it easy to use for beginners.
   * **Disadvantages**: The software is limited to specific image formats (PNG and BMP), restricting its flexibility.
   * **Gaps Identified**: The lack of support for popular formats like JPG and advanced encryption limits its functionality.
   * **Reference**: <https://github.com/ashwek/Hide_n_Seek>
5. **QuickStego**
   * **Problems Addressed**: A basic steganography tool designed for beginners to hide text inside images.
   * **Advantages**: Extremely easy to use, suitable for people with minimal technical skills.
   * **Disadvantages**: Only supports BMP images and lacks any form of encryption.
   * **Gaps Identified**: The lack of encryption and restricted image format support make it unsuitable for securing sensitive information.
   * **Reference**: <https://quick-stego.software.informer.com/1.2/>

By studying these existing systems, **"StegaSafe"** aims to fill the gaps left by these tools, particularly by offering a simplified interface along with support for widely used formats and enhanced security options for better data protection.

# Project Description (150 words)

**"StegaSafe: Image-Based Message Encryption"** is a Python-based application built to simplify the process of embedding secret messages within image files using Least Significant Bit (LSB) steganography. This method involves modifying the least significant bits of image pixels to store the binary data of a text message, creating minimal visual distortion in the image. The goal is to provide a secure communication tool that allows users to hide sensitive information in plain sight without raising suspicion.

The workflow of the project is as follows:

1. **Loading an Image**: Users select an image file (PNG or JPG) from their local device.
2. **Message Encoding**: Users input a text message, which is then embedded into the image using LSB steganography.
3. **Message Decoding**: Users can upload encoded images to extract hidden messages, ensuring that only those with access to the image can retrieve the message.

The application's user interface is built using Kivy, providing an intuitive experience where users can select images, input messages, and perform encoding/decoding operations with a few simple clicks. By integrating the **PIL (Python Imaging Library)** for image processing and the **stegano** library for steganography, **StegaSafe** offers a balance between simplicity, security, and functionality.

# Planning of the Project work (200 words)

The development of **"StegaSafe"** follows a structured, phase-wise approach to ensure steady progress and timely completion. The overall project plan includes the following stages:

1. **Research and Design**: Weeks 1-2
   * During the initial phase, research was conducted on different steganography techniques and existing applications. Based on this research, the project design was conceptualized, with a focus on selecting the LSB method for image-based message encryption. The design of the application's architecture, user interface, and data flow was finalized.
2. **Prototype Development**: Weeks 2-3
   * In this phase, a basic prototype of the application was created to test the encoding and decoding functionality of LSB steganography. The prototype focused on core functionalities like embedding text into image pixels and verifying that the message could be accurately decoded.
3. **User Interface Design**: Weeks 3-4
   * A user-friendly graphical interface was developed using Kivy. The interface includes image selection, message input fields, and buttons for encoding and decoding messages. The design focused on providing clear feedback to users and ensuring ease of use.
4. **Core Development**: Weeks 4-6
   * The core development stage involved integrating the steganography logic with the user interface. Features such as image handling, message encoding/decoding, and saving the encoded image files were fully implemented.
5. **Testing and Optimization**: Weeks 6-7
   * Rigorous testing was conducted to ensure the application could handle different image formats, file sizes, and message lengths. Performance optimization was done to ensure minimal visual distortion in encoded images and fast execution of encoding and decoding processes.
6. **Documentation and Finalization**: Week 8

* Comprehensive documentation, including user guides and technical details, was prepared. The final project was reviewed, and any remaining issues were addressed before the project was submitted for evaluation.

**Gantt Chart:**

* Research and Design: Week 1-2
* Prototype Development: Week 2-3
* User Interface Design: Week 3-4
* Core Development: Week 4-6
* Testing and Optimization: Week 6-7
* Documentation and Finalization: Week 8

# Features

**User Interface Features:**

* **Image Loading**: Users can load images from their local storage using a file picker dialog.
* **Text Input**: An input field is provided for entering the message to be encoded into the image.
* **Encoding/Decoding Buttons**: Clearly labeled buttons allow users to initiate encoding or decoding operations.
* **Result Display**: Feedback is provided to users regarding the success or failure of encoding/decoding operations.

**Functional Features:**

* **Message Encoding**: Users can encode text messages into image files using the Least Significant Bit (LSB) steganography technique.
* **Message Decoding**: The application can extract hidden messages from encoded images, enabling the retrieval of original content.
* **Image Saving**: Encoded images can be saved to disk, preserving the embedded message.

**Security Features:**

* **Data Concealment**: The application uses basic steganography to ensure that hidden messages are not easily detectable without decoding.
* **Privacy Assurance**: By hiding messages within image files, the application provides a layer of privacy that minimizes the risk of unauthorized access.

**Performance Features:**

* **Efficient Processing**: The application is designed to handle encoding and decoding with minimal impact on image quality and processing time.
* **Visual Integrity**: The hidden messages are encoded in a manner that maintains the original appearance of the image with minimal distortion.

**User Experience Enhancements:**

* **Clear Feedback**: Users receive clear indications of encoding/decoding success or failure, enhancing usability.
* **Progress Indicators**: A progress bar provides visual feedback during encoding and decoding processes, keeping users informed of ongoing operations.

# System architecture

The architecture of **"StegaSafe"** consists of three main components:

1. **User Interface (UI)**: Developed using Kivy, this component provides the interactive elements of the application, such as buttons, text fields, and image preview areas. The UI allows users to interact with the application and perform encoding/decoding operations.
2. **Steganography Core**: This component implements the Least Significant Bit (LSB) algorithm for embedding and extracting messages from image files. It handles the core functionality of message hiding and retrieval.
3. **File Handling System**: Responsible for managing the input and output of image files, this system ensures that images are correctly loaded for encoding and saved after processing. It handles file operations and ensures compatibility with supported image formats.

**System Flow:**

* The user interacts with the UI to select an image and input a message.
* The UI passes this information to the steganography core, which performs the encoding or decoding operation.
* The file handling system manages the image files, ensuring they are correctly processed and saved.

# User Interface (UI)

The **user interface** of **"StegaSafe"** is designed with a focus on simplicity and usability. It includes:

* **Image Preview Area**: Displays the selected image, allowing users to see the image before and after encoding.
* **Text Input Field**: Users can enter the message to be hidden within the image.
* **Load Image Button**: Opens a file picker dialog for selecting an image file from the local storage.
* **Encode Button**: Initiates the encoding process, embedding the message into the selected image.
* **Decode Button**: Starts the decoding process to extract hidden messages from encoded images.
* **Result Display Label**: Provides feedback on the success or failure of encoding/decoding operations.

# Technology Stack

**Programming Language**: Python

* Python is chosen for its simplicity and extensive library support, making it suitable for rapid development and integration of steganography features.

**Framework**: Kivy

* Kivy is used for designing the graphical user interface, providing a flexible and intuitive framework for building cross-platform applications.

**Steganography Library**: stegano

* The stegano library provides tools for implementing Least Significant Bit (LSB) steganography, allowing for message encoding and decoding.

**Image Processing**: PIL (Python Imaging Library)

* PIL is used for image manipulation tasks, such as loading and saving images, ensuring compatibility with various image formats.

**IDE**: PyCharm

* PyCharm is utilized for its powerful development features, including debugging and code management, which aid in the efficient development of the application.

# Testing Plan

The testing plan for **"StegaSafe"** involves several testing methodologies to ensure the application's functionality and reliability:

**Test Cases:**

* **Test 1**: Encoding a Short Message
  + **Objective**: Verify that a short text message can be encoded into a PNG image and that the image remains visually unchanged.
  + **Method**: Encode a predefined short message into a PNG image and compare the original and encoded images for visual differences.
* **Test 2**: Decoding Messages
  + **Objective**: Ensure that hidden messages can be accurately decoded from images.
  + **Method**: Decode messages from encoded images and compare the retrieved messages with the original ones to ensure accuracy.
* **Test 3**: Unsupported Image Formats
  + **Objective**: Test the application's error handling for unsupported image formats.
  + **Method**: Attempt to load and process images in formats not supported by the application and verify that appropriate error messages are displayed.

**Testing Strategies:**

* **Unit Testing**: Test individual components of the application, such as encoding and decoding functions, to ensure they work correctly.
* **Integration Testing**: Test the interaction between different components, including the UI, steganography core, and file handling system.
* **User Acceptance Testing**: Conduct testing with actual users to validate the application's usability and functionality in real-world scenarios.

# Expected Outcome (100-150 words)

The expected outcome of **"StegaSafe: Image-Based Message Encryption"** is to deliver a fully functional application that allows users to securely hide and retrieve messages within digital images. The application will offer:

* **User-Friendly Interface**: An intuitive GUI that simplifies the process of encoding and decoding messages, making it accessible to users with varying technical backgrounds.
* **Data Privacy**: A reliable method for ensuring that sensitive information remains concealed within images, enhancing privacy and security.
* **Enhanced Communication Security**: A tool that supports secure communication by allowing users to share hidden messages discreetly.

By achieving these outcomes, **"StegaSafe"** will contribute to better data security and privacy in digital communications, addressing the growing need for secure and unobtrusive methods of data protection.

# Resources and Limitations (100 words)

**Resources:**

* **Hardware**: Development will be carried out on a laptop or PC with sufficient processing power and storage capacity.
* **Software**: The project will utilize Python, Kivy, PIL, and the stegano library for development.
* **Data**: Test image files in PNG and JPG formats will be used for encoding and decoding operations.

**Limitations:**

* **File Types**: The project is limited to image files and does not support audio or video files for message hiding.
* **Security**: LSB steganography, while useful, does not include additional encryption layers, making it vulnerable to certain types of attacks. Future improvements could include incorporating encryption for enhanced security.

# Conclusion (100-150 words)

The **"StegaSafe: Image-Based Message Encryption"** project represents a practical and effective approach to secure communication by embedding hidden messages within image files using steganography. The project successfully combines technical efficiency with an accessible user interface, providing a tool that is easy to use while ensuring the confidentiality of sensitive information.

By focusing on simplicity and usability, **StegaSafe** offers a valuable solution for secure communication, especially in contexts where data privacy is paramount. Although the project has some limitations, such as its focus on image files and the lack of advanced encryption, it lays the foundation for future enhancements. The successful implementation of **"StegaSafe"** will contribute to personal data security and privacy, serving as a stepping stone for more advanced steganography solutions.

# References

1. OpenPuff Steganography Tool. [Online] Available at: https://embeddedsw.net/OpenPuff\_Steganography\_Home.html
2. StegHide Documentation. [Online] Available at: https://steganosaur.us/stegohide/
3. SSuite Picsel. [Online] Available at: <https://www.ssuitesoft.com>
4. Hide'N'Send GitHub Repository. [Online] Available at: <https://github.com/hide-n-send>
5. QuickStego. [Online] Available at: <https://quickstego.com>
6. Kivy Documentation. [Online] Available at: https://kivy.org/doc/stable/
7. Stegano Library. [Online] Available at: <https://github.com/cedricbonhomme/Stegano>