

SECURE IMAGE STEGANOGRAPHY SYSTEM

CRYPTO FINAL PROJECT REPORT

1. INTRODUCTION

In today's digital world, protecting confidential information is very important. Traditional cryptography secures data by encrypting it, but it does not hide the existence of the message. Anyone can see that encrypted data is being transmitted.

Steganography is a technique used to hide secret information inside digital media such as images, audio, or video. This project combines cryptography and steganography to create a Secure Image Steganography System. The secret message is encrypted with a password and then hidden inside an image using the Least Significant Bit (LSB) technique.

2. OBJECTIVES

The main objectives of this project are:

- To hide secret text messages inside digital images
 - To protect hidden messages using password-based encryption
 - To combine cryptography and steganography concepts
 - To provide both command-line and graphical user interfaces
 - To improve basic security by clearing sensitive data in the GUI
-

3. SYSTEM OVERVIEW

The Secure Image Steganography System allows users to hide and reveal secret messages in images. The system supports two interfaces:

1. Command-Line Interface (CLI)
2. Graphical User Interface (GUI)

The system uses a password to encrypt the message before hiding it inside the image. Only users with the correct password can retrieve the original message.

4. TECHNOLOGIES USED

Programming Language:
Python 3.10 or higher

Libraries Used:

- Pillow (PIL) for image processing
- Tkinter for graphical user interface

Techniques Used:

- Least Significant Bit (LSB) steganography
 - XOR-based password encryption
-

5. PROJECT STRUCTURE

The project consists of the following files:

- main.py : Command-Line Interface
 - gui.py : Graphical User Interface
 - encoder.py : Encrypts and hides messages
 - decoder.py : Extracts and decrypts messages
 - README.md : Project documentation
 - requirements.txt : Required libraries
 - Encrypted-Images folder : Stores encoded images
 - test.jpg / test.png : Demo images
-

6. COMMAND-LINE INTERFACE (main.py)

The Command-Line Interface provides a menu-based system with three options:

1. Hide a secret message in an image
2. Reveal a hidden message from an image
3. Exit the program

Users enter the image path, secret message, password, and output image name through the terminal. This interface is suitable for users familiar with command-line operations.

7. GRAPHICAL USER INTERFACE (gui.py)

The Graphical User Interface is built using Tkinter to make the system easy to use. It provides Encode and Decode modes with buttons and input fields.

Main GUI Features:

- Image file selection using file browser
- Password input with hidden characters
- Encode and Decode mode switching
- Popup messages for success and errors
- Output images saved in the Encrypted-Images folder

Security

Improvement:

When switching from Encode Mode to Decode Mode, sensitive information such as the image path and password is automatically cleared. This prevents unauthorized users from seeing or reusing the password.

8. ENCODING PROCESS (encoder.py)

The encoding process works as follows:

1. The user enters a secret message and password
 2. The message is encrypted using XOR encryption
 3. An <END> marker is added to the encrypted message
 4. The message is converted into binary format
 5. Binary data is embedded into the least significant bits of image pixels
 6. The encoded image is saved without visible distortion
-

9. DECODING PROCESS (decoder.py)

The decoding process works as follows:

1. The encoded image is loaded
 2. LSB values are extracted from image pixels
 3. Binary data is converted back into characters
 4. Extraction stops when the <END> marker is found
 5. The encrypted message is decrypted using the password
 6. The original secret message is displayed
-

10. SECURITY ANALYSIS

The system provides the following security features:

- Password-based encryption
- Hidden communication using steganography
- Automatic clearing of sensitive fields in GUI
- End marker to ensure correct message extraction

Note: XOR encryption is used only for educational purposes and is not suitable for high-security systems.

11. LIMITATIONS

- XOR encryption is weak compared to modern encryption algorithms
 - Message size depends on image resolution
 - Lossy image formats may damage hidden data
 - No message integrity or tampering detection
-

12. FUTURE IMPROVEMENTS

Possible future enhancements include:

- Use AES or RSA encryption
 - Add message integrity verification
 - Support audio or video steganography
 - Implement user authentication
 - Add password strength checking
-

13. CONCLUSION

This project successfully demonstrates how cryptography and steganography can be combined to securely hide information inside images. The system supports both CLI and GUI interfaces and follows basic security practices. Although the encryption method is simple, the project effectively explains core concepts of secure data hiding and information security.