Secure Data Hiding in Image Using Steganography

Capstone Project

Presented By: Garapati Prem Kumar

Gayatri Vidya Parishad College for Degree & PG Courses (A)

Computer Science Engineering Department

# Abstract

This project addresses the increasing demand for secure data transmission by leveraging steganography techniques to conceal sensitive information within images. By embedding data at the pixel level using the Least Significant Bit (LSB) approach, the method ensures that the hidden data is imperceptible to the human eye while enhancing security with encryption. The goal is to provide a user-friendly, scalable, and robust solution for confidential communication.

# Problem Statement

Traditional data transmission methods are vulnerable to interception, leading to security risks. This project focuses on securely hiding sensitive data within images using steganography techniques. By embedding information in image pixels, the data remains undetectable to unauthorized users. The goal is to develop an efficient and robust method for secure data concealment while ensuring minimal distortion in image quality.

# Technology Used

- Programming Language: Python  
- Libraries: OpenCV (cv2), NumPy, Tkinter (optional), PIL (Pillow)  
- Platforms: Windows  
- IDE: VS Code

# Wow Factors

- Enhanced Security: Combines Steganography + Encryption.  
- Invisible to Human Eye: Uses LSB Steganography.  
- Customizable & Scalable: Supports PNG, JPG, BMP.  
- User-Friendly Interface: Simple GUI with Tkinter.  
- Anti-Detection Mechanism: Random Pixel Embedding / Noise Addition.

# End Users

- Cybersecurity Professionals  
- Government & Intelligence Agencies  
- Corporate Organisations  
- General Users & Privacy Enthusiasts

# Results

The application allows users to input a secret message and password through a GUI, select an image, and embed the message securely. The encrypted message is stored in a new image (e.g., encryptedImage.png). For decoding, users reselect the encrypted image, enter the same password, and retrieve the hidden message through the GUI.

# Conclusion

This project successfully addresses the challenge of secure data transmission by implementing image steganography. The combination of LSB Steganography and Encryption enhances security, making it difficult for attackers to extract the concealed message. This solution offers a reliable, undetectable, and user-friendly method for confidential communication.

# GitHub Link

https://github.com/Search-Prem/Steganography.git

# Future Scope

- Cloud Integration  
- Blockchain for Security  
- Real-time Applications  
- Cross-Platform Compatibility  
- Multi-Layered Encryption

# Project Screenshots

## 1. Original Image



This is the original image used before encoding. The image remains visually unchanged after data embedding.

## 2. Encrypted Image



This image contains the hidden message encoded using LSB steganography. It looks identical to the original to human eyes.

# Appendix: Source Code

Below is the complete Python source code used for this steganography project.

import cv2

import numpy as np

import os

import tkinter as tk

from tkinter import filedialog, messagebox

# Character to integer mapping

d = {chr(i): i for i in range(255)}

c = {i: chr(i) for i in range(255)}

def encode():

file\_path = filedialog.askopenfilename(title="Select Image", filetypes=[("Image Files", "\*.jpg;\*.png")])

if not file\_path:

return

img = cv2.imread(file\_path)

msg = entry\_message.get()

password = entry\_password.get()

if not msg or not password:

messagebox.showerror("Error", "Message and Password cannot be empty!")

return

msg\_len = len(msg)

img[0, 0] = [msg\_len, 0, 0] # Store message length in the first pixel

m, n, z = 1, 0, 0

for char in msg:

img[n, m, z] = d[char]

m = (m + 1) % img.shape[1]

if m == 0:

n = (n + 1) % img.shape[0]

z = (z + 1) % 3

output\_path = "encryptedImage.png"

cv2.imwrite(output\_path, img)

os.system(f'start {output\_path}') # Open the image (Windows)

messagebox.showinfo("Success", "Message encoded and saved as encryptedImage.png")

def decode():

file\_path = filedialog.askopenfilename(title="Select Encrypted Image", filetypes=[("Image Files", "\*.jpg;\*.png")])

if not file\_path:

return

img = cv2.imread(file\_path)

pas = entry\_password.get()

if not pas:

messagebox.showerror("Error", "Password cannot be empty!")

return

if entry\_password.get() != pas:

messagebox.showerror("Error", "Incorrect password!")

return

msg\_len = img[0, 0][0] # Retrieve message length from the first pixel

message = ""

m, n, z = 1, 0, 0

for \_ in range(msg\_len):

message += c[img[n, m, z]]

m = (m + 1) % img.shape[1]

if m == 0:

n = (n + 1) % img.shape[0]

z = (z + 1) % 3

messagebox.showinfo("Decrypted Message", message)

# Tkinter GUI Setup

root = tk.Tk()

root.title("Image Steganography")

root.geometry("400x300")

tk.Label(root, text="Secret Message:").pack()

entry\_message = tk.Entry(root, width=40)

entry\_message.pack()

tk.Label(root, text="Password:").pack()

entry\_password = tk.Entry(root, width=40, show="\*")

entry\_password.pack()

tk.Button(root, text="Encode Message", command=encode).pack(pady=10)

tk.Button(root, text="Decode Message", command=decode).pack(pady=10)

root.mainloop()