# CAPSTONE PROJECT

# SECURE DATA HIDING IN IMAGES USING STEGANOGRAPHY

Presented By:- Aman Pushpak

Student Name: Aman Pushpak
College Name: International Institute of Professional Studies
Department : Information Technology

edunet foundation

# OUTLINE

- Problem Statement
- Technology used
- Wow factor
- End users
- Result
- Conclusion
- Git-hub Link
- Future scope

# PROBLEM STATEMENT

- Traditional encryption methods are easily detectable and can raise suspicion.
- Need for a secure way to hide confidential messages within images.
- Ensuring data security while maintaining the integrity of the cover image.
- Making encryption and decryption accessible to non-technical users through a simple interface.

# TECHNOLOGY USED

- **Programming Language:** Python
- **Image Processing Library:** OpenCV
- **Graphical User Interface (GUI):** Tkinter
- **File Handling & Security:** Basic encryption logic for password protection
- **Additional Libraries:** Pillow (for image handling)
- **Platform:** Windows 11

# WOW FACTORS

- Uses **steganography** to embed a message into an image without noticeable changes.
- **User-friendly GUI** for easy encryption and decryption.
- Lossless data hiding using **pixel value manipulation** instead of traditional cryptographic techniques.
- Works on **any standard image file format** (PNG recommended for best results).

# END USERS

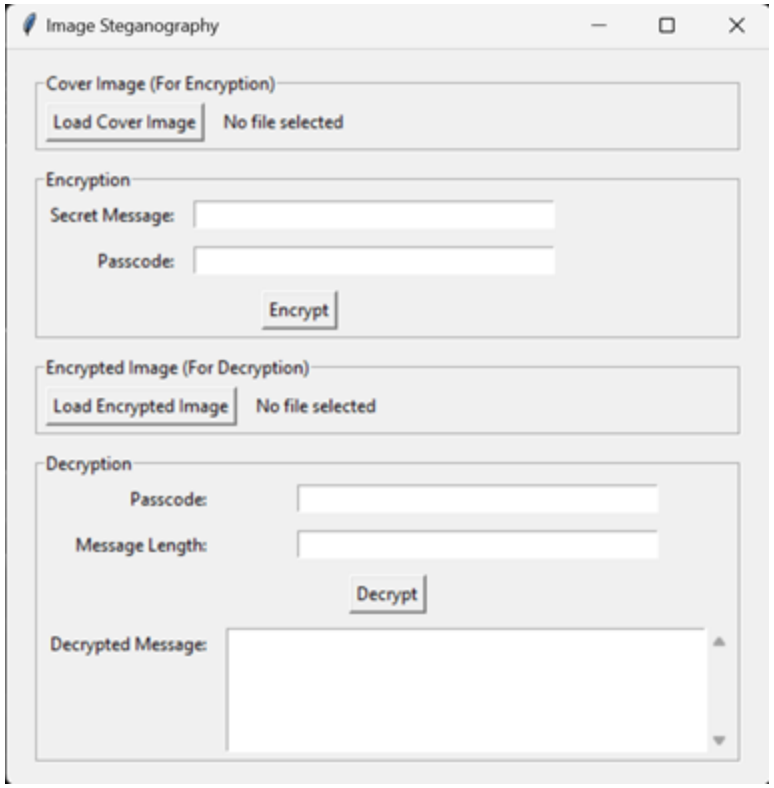
- **Cybersecurity Enthusiasts –** Exploring secure communication techniques.
- **Government & Defense –** Secure message transmission without raising suspicion.
- **Journalists & Activists –** Concealing sensitive information in images to avoid surveillance.
- **Software Developers –** Learning steganography concepts and their applications.

# RESULTS

## Encryption Code:

```
import cv2
import os

# Load cover image
img = cv2.imread("mypic.jpg")  # Ensure this image exists in your working directory
if img is None:
    print("Cover image not found!")
    exit()

# Input secret message and password
msg = input("Enter secret message: ")
password = input("Enter a passcode: ")

# Save the password to a file (for demonstration purposes only)
with open("pass.txt", "w") as f:
    f.write(password)

# Embed the message into the image.
# We will write each character's ASCII value into a pixel channel.
n = 0  # row index
m = 0  # column index
z = 0  # channel index (0=Blue, 1=Green, 2=Red in OpenCV)

for char in msg:
    # Check if we are within image boundaries
    if n >= img.shape[0] or m >= img.shape[1]:
        print("Message is too long for the image!")
        break
    # Write the ASCII value of the character into the chosen pixel channel
    img[n, m, z] = ord(char)
    n += 1
    m += 1
    z = (z + 1) % 3  # Cycle through the three channels

# Save the encrypted image as PNG to avoid lossy compression
cv2.imwrite("encryptedImage.png", img)
os.system("start encryptedImage.png")  # For Windows; on macOS use 'open' and on Linux 'xdg-open'
print("Secret message embedded into image!")
```

## GUI:




## Decryption Code:

```
import cv2

# Load the encrypted image (lossless PNG format)
img = cv2.imread("encryptedImage.png")
if img is None:
    print("Encrypted image not found!")
    exit()

# Retrieve the stored password from file
try:
    with open("pass.txt", "r") as f:
        correct_pass = f.read().strip()
except FileNotFoundError:
    print("Password file not found!")
    exit()

# Ask user for the decryption passcode
pas = input("Enter passcode for Decryption: ")
if pas != correct_pass:
    print("Incorrect passcode. Access denied!")
    exit()

# Ask user for the secret message length.
# (In this simple example, you need to remember or note the message length.)
try:
    length = int(input("Enter secret message length: "))
except ValueError:
    print("Invalid length input!")
    exit()

message = ""
n = 0  # row index
m = 0  # column index
z = 0  # channel index

# Read the embedded message from the image
for i in range(length):
    # Check if we are within image boundaries
    if n >= img.shape[0] or m >= img.shape[1]:
        print("Reached image boundary before reading the full message.")
        break
    # Read the pixel channel value and convert it back to a character
    message += chr(img[n, m, z])
    n += 1
    m += 1
    z = (z + 1) % 3

print("Decrypted message:", message)
```

## Input Image File:

## Output Image File:

edunet foundation

# CONCLUSION

- Steganography provides a **covert** way of communicating sensitive information.
- This project showcases a **simple yet effective** implementation of message hiding in images.
- The GUI makes encryption and decryption **accessible** even to non-technical users.
- Future improvements can make it more **robust, secure, and scalable.**

# GITHUB LINK

- https://github.com/Amanpushpak/Steganography.git

# FUTURE SCOPE

- **Advanced Encryption Techniques –** Integrate AES encryption before embedding text in images.
- **Support for Audio & Video Steganography –** Expanding beyond images.
- **AI-based Detection Prevention –** Ensuring messages stay undetectable from modern forensic tools.
- **Mobile App & Web Version –** Expanding accessibility beyond desktops.
- **Multi-Layer Security –** Combining steganography with blockchain for ultra-secure communication.

# THANK YOU

edunet foundation