CAPSTONE PROJECT

STENOGRAPHY IN PYTHON

Presented By:
Student Name : Niharika S Nair
College Name & Department : Mar Baselios College of Engineering and Technology,B-Tech in Electrical and Electronics

edunet
foundation

OUTLINE

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

PROBLEM STATEMENT

The task is to securely hide sensitive data within an image using steganography, ensuring both confidentiality and protection against unauthorized access. Unlike conventional encryption methods, which can make hidden data more apparent, steganography embeds information directly into the image's pixels, leaving the image visually unchanged. The aim is to create an effective algorithm that allows for seamless embedding and extraction of data, while preserving the image’s quality and maintaining strong security.

TECHNOLOGY USED

Language Used: Python

Libraries Used: Standard Libraries such as NumPy, OpenCV, OS, Tkinter, string etc.

Platforms:

Operating System: Windows 10

IDE: JetBrains PyCharm

File Format: BMP (Bitmap Image), JPEG

Additional Points:

Steganography Method: Least Significant Bit (LSB) Encoding

Data Handling: Embeds and extracts text data from images

Ability to upload image directly from the file manager without having to specify the path.

WOW FACTORS

- **Seamless Image-Based Encryption:**
  Secret messages are securely embedded within image pixels, making the data nearly invisible to unauthorized users.
- **Minimal Image Distortion:**
  Text is embedded without altering the image's visual appearance, ensuring the encrypted image looks identical to the original.
- **Dual-Layer Security with Passcode Protection:**
  Even if the modified image is extracted, a passcode is required to decrypt the hidden message, adding an extra layer of security.
- **Efficient and Simple Algorithm:**
  A lightweight pixel-mapping technique is used for encoding, ensuring quick and effective encryption and decryption.
- **Intuitive File Selection:**
  A user-friendly GUI-based file dialog allows easy selection of images for encryption.
- **Cross-Format Compatibility:**
  Supports multiple image formats (JPEG, PNG, BMP), making it adaptable for a wide range of use cases.
- **Automatic Image Preview After Encryption:**
  The encrypted image opens automatically, so users can immediately confirm that the file appears unchanged.
- **Platform-Agnostic Functionality:**
  Works seamlessly across different operating systems, requiring only minor adjustments (e.g., replacing os.system("start...") for macOS/Linux).

edunet
foundation

END USERS

Journalists & Whistleblowers

Can discreetly hide sensitive information within images to avoid detection when sharing critical reports or evidence.

Cybersecurity Professionals
Ideal for training, research, and the development of techniques in digital forensics, data protection, and steganalysis.

Government & Intelligence Agencies
Enables secure, covert communication and classified data transmission, ensuring secrecy without raising suspicion.

Businesses & Corporations
Safeguards confidential business documents or trade secrets by embedding them within images for added security.

Privacy-Conscious Individuals
Provides a way for everyday users to store private notes, passwords, or messages hidden within images.

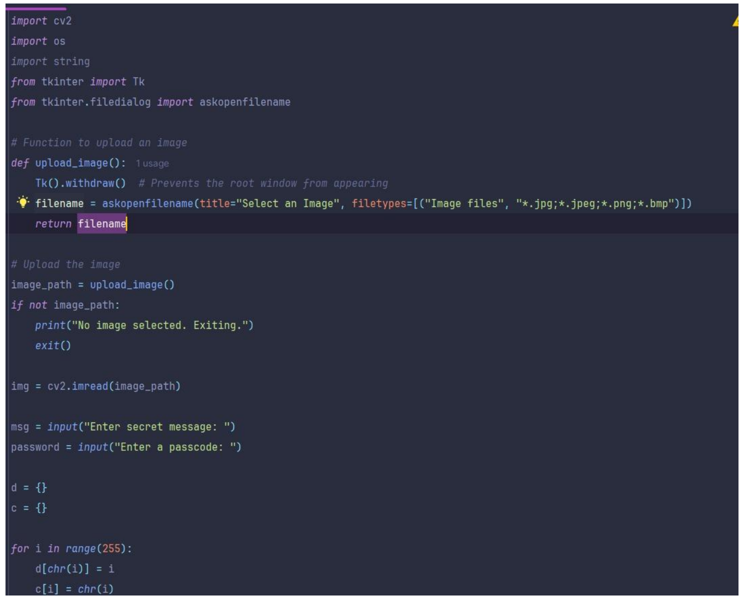
Artists & Digital Creators
Allows embedding of hidden signatures or copyrights within artwork, helping to prevent plagiarism and establish authenticity.

Researchers & Academics
Serves as a valuable tool for studying data-hiding techniques, cryptography, and information security in academic research and education.

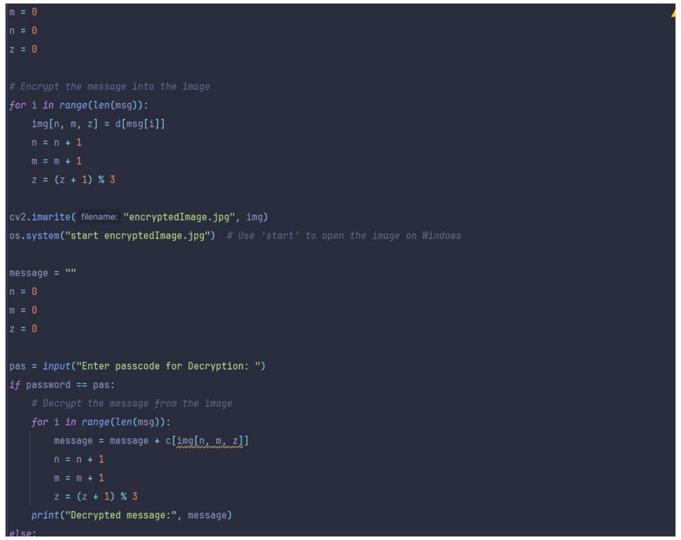
Hackathon Participants & Developers
Provides a foundation for building innovative security solutions in hackathons or cybersecurity competitions.

.

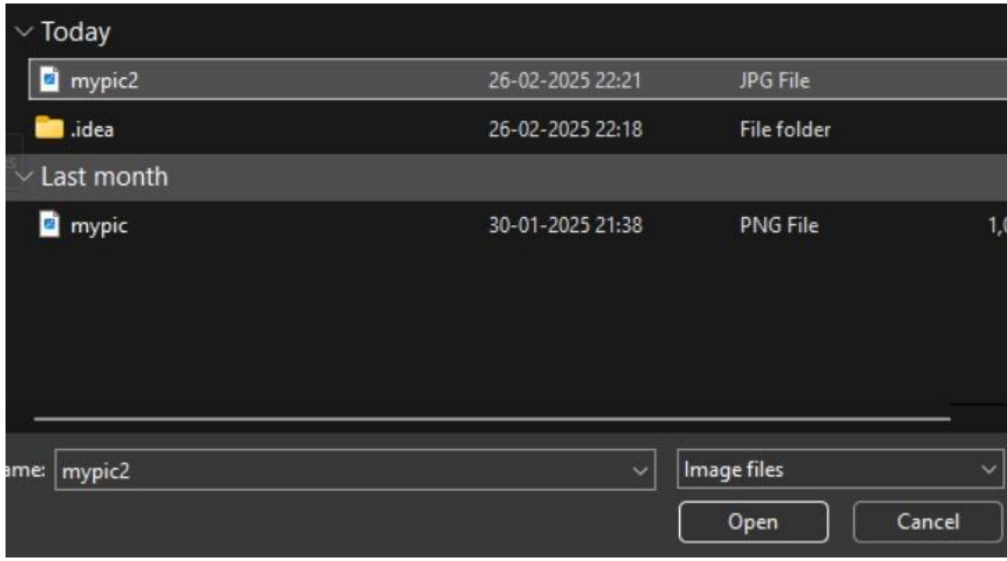
RESULTS



```
import cv2
import os
import string
from tkinter import Tk
from tkinter.filedialog import askopenfilename

# Function to upload an image
def upload_image():  1 usage
    Tk().withdraw()  # Prevents the root window from appearing
    filename = askopenfilename(title="Select an Image", filetypes=[("Image files", "*.jpg;*.jpeg;*.png;*.bmp")])
    return filename

# Upload the image
image_path = upload_image()
if not image_path:
    print("No image selected. Exiting.")
    exit()

img = cv2.imread(image_path)

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

d = {}
c = {}

for i in range(255):
    d[chr(i)] = i
    c[i] = chr(i)
```

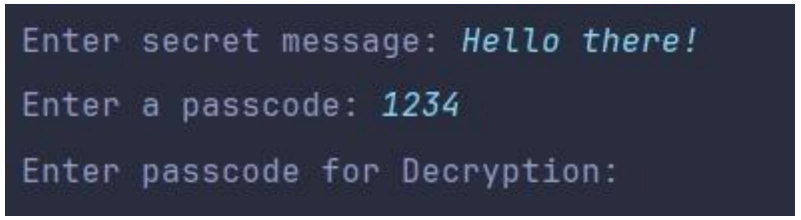


```
m = 0
n = 0
z = 0

# Encrypt the message into the image
for i in range(len(msg)):
    img[n, m, z] = d[msg[i]]
    n = n + 1
    m = m + 1
    z = (z + 1) % 3

cv2.imwrite( filename: "encryptedImage.jpg", img)
os.system("start encryptedImage.jpg")  # Use 'start' to open the image on Windows

message = ""
n = 0
m = 0
z = 0

pas = input("Enter passcode for Decryption: ")
if password == pas:
    # Decrypt the message from the image
    for i in range(len(msg)):
        message = message + c[img[n, m, z]]
        n = n + 1
        m = m + 1
        z = (z + 1) % 3
    print("Decrypted message:", message)
else:
```



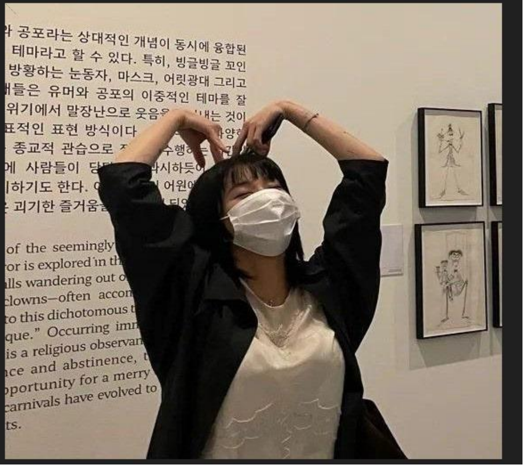
Today		
mypic2	26-02-2025 22:21	JPG File
.idea	26-02-2025 22:18	File folder
Last month		
mypic	30-01-2025 21:38	PNG File

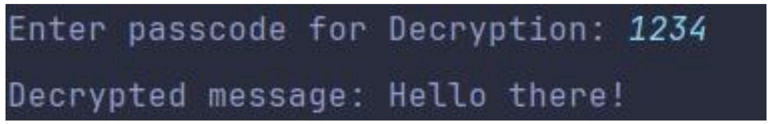
ame: mypic2 · Image files



```
Enter secret message: Hello there!
Enter a passcode: 1234
Enter passcode for Decryption:
```

encryptedImage.jpg





```
Enter passcode for Decryption: 1234
Decrypted message: Hello there!
```

CONCLUSION

This steganography-based image encryption system offers a straightforward yet powerful method for concealing sensitive information within images, all while preserving their visual integrity. By embedding text within pixel values and securing access with a passcode, the system adds an extra layer of protection, making it ideal for confidential communication and data security.

With its user-friendly interface, cross-format compatibility, and seamless encryption-decryption process, the program is well-suited for applications in cybersecurity, journalism, corporate security, and personal privacy. Although effective for basic steganography, future improvements, such as stronger encryption algorithms, enhanced storage efficiency, and AI-driven resistance to steganalysis, could further boost its security.

GITHUB LINK

- https://github.com/nihanair/steno

FUTURE SCOPE(OPTIONAL)

Enhanced Security with Advanced Encryption
Incorporate AES or RSA encryption alongside steganography to provide multiple layers of protection, fortifying data security.

Advanced Image Processing Techniques
Utilize methods like LSB (Least Significant Bit) substitution combined with random pixel selection to make the detection of hidden information even more difficult.

Increased Data Capacity
Adopt more efficient encoding techniques, such as DCT (Discrete Cosine Transform) or Wavelet Transformation, to accommodate larger messages without noticeable distortion in the image.

Multi-Format Support & Compression Handling
Broaden compatibility to include formats like GIFs and videos, while implementing strategies to minimize data loss due to image compression.

Steganalysis Resistance
Develop AI-driven algorithms to combat steganalysis techniques, making it more challenging for attackers to detect hidden messages.

Real-Time Communication Integration
Extend the functionality to popular messaging apps, enabling the secure exchange of encrypted images via email or chat platforms.

Blockchain Integration for Authentication
Incorporate blockchain technology to authenticate steganographic messages, ensuring tamper-proof data transmission and enhancing trust.

THANK YOU

edunet foundation