CAPSTONE PROJECT

SECURE DATA HIDING IN IMAGE USING STEGANOGRAPHY

Presented By: Aayush Sunil Patil Wankhede
Student Name: Aayush Sunil Patil Wankhede
College Name: Shri Guru Gobind Singh Ji Institute of Engineering and Technology, Nanded-431606
Department: Electronics and Telecommunication

edunet foundation

OUTLINE

- Introduction to Steganography
- Problem Statement
- Technology used
- Block diagram and Flowchart
- Wow factor
- End users
- Result
- Conclusion
- Git-hub Link
- Future scope

INTRODUCTION TO STEGANOGRAPHY

- **What is Steganography?**
  A technique for **hiding information** inside other data to **avoid detection**.
- **Project Focus**
  Implements **image-based steganography**, allowing **text messages** to be concealed within an image.
- **Key Advantage**
  Hidden data remains **undetectable** without the correct **decryption key**, ensuring **stealthy communication**.

PROBLEM STATEMENT

- Traditional encryption methods make data unreadable but do not hide its existence, making it susceptible to detection.
- **Steganography** provides a discreet method by embedding secret messages within images, ensuring security without raising suspicion.
- This project focuses on developing a **Python-based system** that can securely **hide and retrieve messages from images**, offering a seamless and covert method of data protection.
- **Key Points**

✓ **Hidden in Plain Sight** : Secret messages are embedded in images without altering their visible appearance.

✓ **Enhanced Security & Confidentiality** : Protects sensitive information from unauthorized access.

✓ **User-Friendly & Efficient** : Simple Python-based system for message hiding and extraction.

✓ **Real-World Applications**

✓ Secure communication for journalists & organizations.

✓ Protecting confidential data in digital media.

- 🚀 **Steganography + Python = A Smart & Secure Data-Hiding Solution!**

TECHNOLOGY USED

- **Platform:** Windows 11 : Ensures compatibility and smooth execution of the application.
- **Programming Language:** Python : A versatile and powerful language for image processing and GUI development.
- **Libraries:**

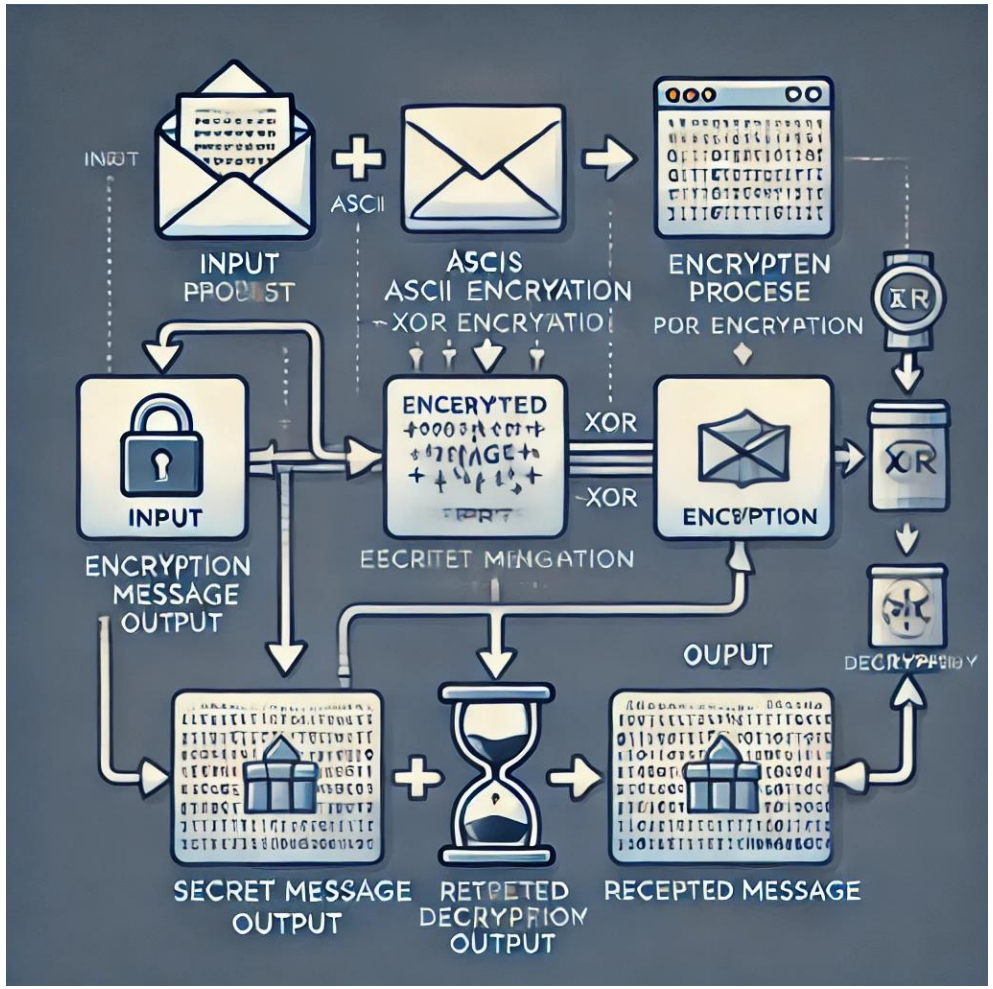
  OpenCV : For image processing and pixel manipulation.

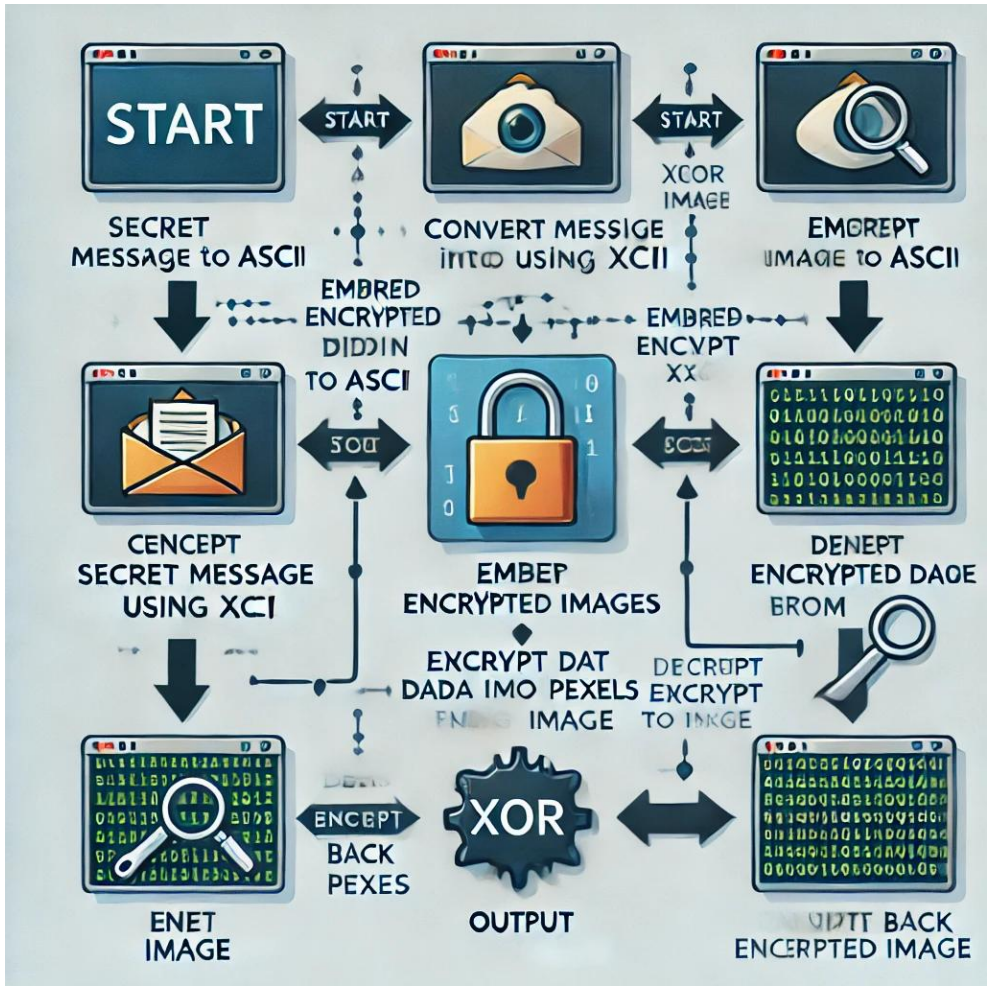
  NumPy : Efficient numerical computations for encoding and decoding.

  Tkinter : GUI framework for an interactive user interface.
- **Techniques Used**
- **ASCII Value Mapping** : Converts text into numerical values for secure encoding.
- **Image Pixel Manipulation** : Embeds secret data into image pixels without noticeable distortion.
- **Passcode-Based Access Control** : Ensures only authorized users can extract hidden messages.
- **File Format**
- **PNG/JPG** : Supports lossless image storage, maintaining data integrity.
- **Tools Used**
- **Jupyter Notebook & PyCharm** : For coding, testing, and debugging.

  Tkinter : For building an intuitive and user-friendly GUI.
- 🚀 **A Blend of Cutting-Edge Tools & Techniques for Secure Data Hiding!**

BLOCK DIAGRAM AND FLOWCHART





WOW FACTORS

- **Security** – Passcode-based protection ensures that only authorized users can retrieve hidden messages.
- **Automation** – Intuitive **Tkinter-based GUI** makes the process seamless and user-friendly.
- **Scalability** – Can be extended to support **video and audio steganography** in future enhancements.
- **Error Handling** – Prevents **oversized messages** and **incorrect file selections**, ensuring smooth execution.
- **Multi-Platform Support** – Compatible with **Windows 11, Linux, and macOS**, offering cross-platform functionality.
- 🚀 **A Secure, Scalable, and Smart Approach to Steganography!**

END USERS

- **Cybersecurity Professionals** – Ensures **secure communication** and aids in **digital forensics**.
- **Journalists & Whistleblowers** – Enables **confidential data sharing** without raising suspicion.
- **Military & Intelligence Agencies** – Facilitates **covert message transmission** for secure operations.
- **General Users** – Protects **personal data** from unauthorized access and breaches.
- 🚀 **Empowering Secure & Undetectable Communication for Everyone!**

RESULTS

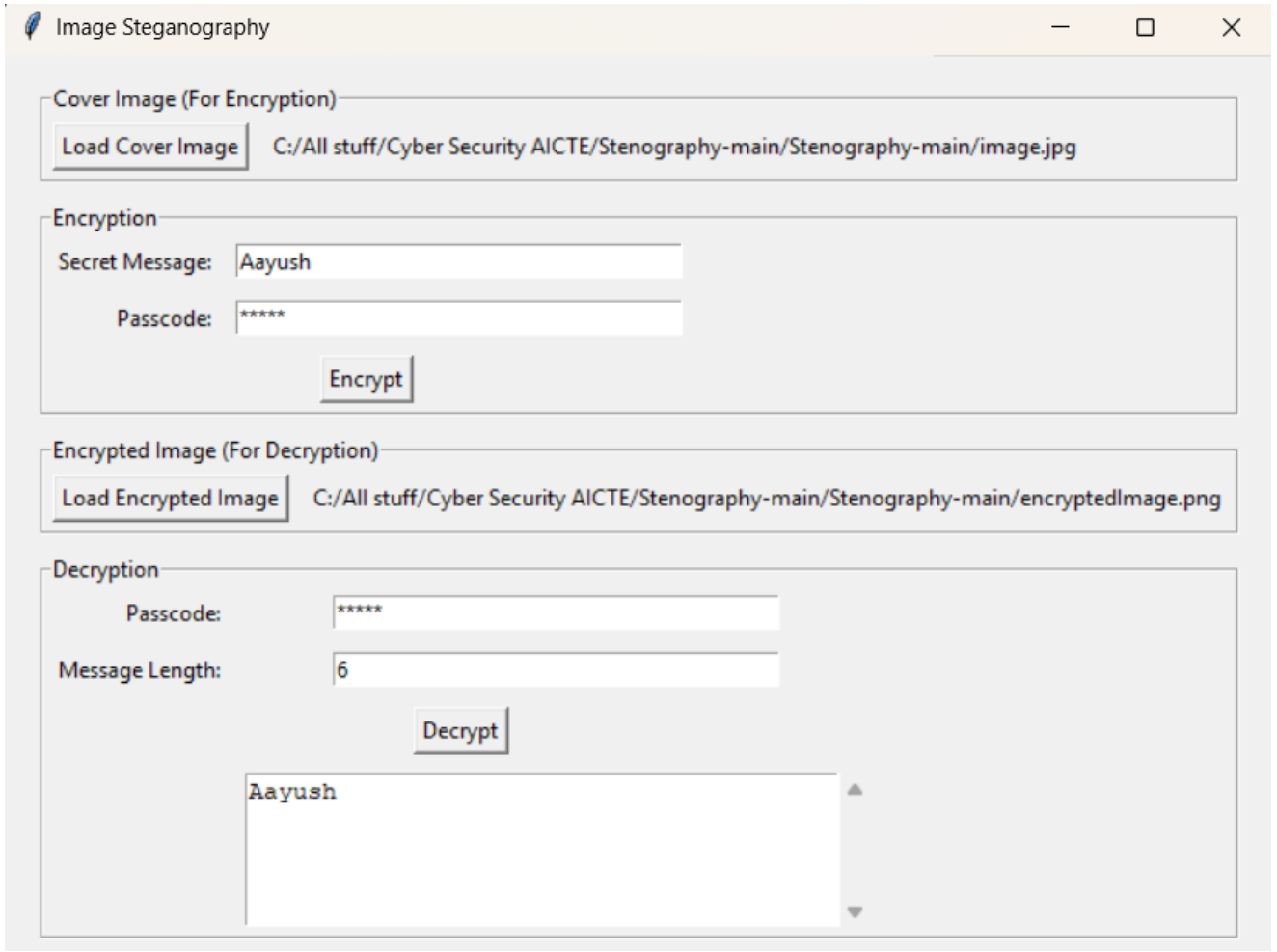
Project.py > ...

```python
import cv2
import os

# Load the image
img = cv2.imread("image.jpg")  # Replace with the correct image path
if img is None:
    print("Error: Image not found. Check the file path.")
    exit()

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

# Character to ASCII mappings
d = {chr(i): i for i in range(255)}
c = {i: chr(i) for i in range(255)}

# Get image dimensions
height, width, _ = img.shape
max_capacity = height * width * 3  # Maximum bytes we can store

# Ensure message fits in the image
if len(msg) + 4 > max_capacity:  # +4 bytes to store message length
    print("Error: Message too long for the image size.")
    exit()

# Convert message length to 4 bytes and store it first
msg_length = len(msg)
length_bytes = [(msg_length >> (i * 8)) & 0xFF for i in range(4)]

# Encode message length
n, m, z = 0, 0, 0
for byte in length_bytes:
    img[n, m, z] = byte
    n, m, z = n + 1, m + 1, (z + 1) % 3

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

# Save and open the encrypted image
cv2.imwrite("encryptedImage.jpg", img)
if os.name == "nt":  # Windows
    os.system("start encryptedImage.jpg")
else:  # Linux/Mac
    os.system("xdg-open encryptedImage.jpg" if os.name == "posix" else "open encryptedImage.jpg")

# Decryption
message = ""
n, m, z = 0, 0, 0

# Ask for the passcode
pas = input("Enter passcode for Decryption: ")
if password == pas:
    # Retrieve message length
    msg_length = sum(img[n + i, m + i, (z + i) % 3] << (i * 8) for i in range(4))
    n, m, z = 4, 4, 1  # Move past the length bytes

    # Retrieve the message
    for i in range(msg_length):
        message += c[img[n, m, z]]
        n, m, z = n + 1, m + 1, (z + 1) % 3

    print("Decryption message:", message)
else:
    print("YOU ARE NOT AUTHORIZED")
```

```
Enter secret message: Aayush
Enter a passcode: 12345
Enter passcode for Decryption: 12345
Decryption message: Aayush
```

edunet foundation

RESULTS



Input image

Output image

edunet foundation

CONCLUSION

- **Enhanced Security** – Steganography **hides** messages within images, adding an extra layer of protection.
- **Stealthy Communication** – Ensures **undetectable** data transfer without raising suspicion.
- **Practical Implementation** – Demonstrates how **image manipulation** can be leveraged for **secure messaging**.
- **Future Scope** – Can be extended to **video, audio, and real-time applications** for broader security needs.
- **Secure. Invisible. Reliable.**

GITHUB LINK

https://github.com/Aayush-PatilWankhede/Steganography.git

FUTURE SCOPE

- **Advanced Encryption :** Enhancing security by integrating **AES encryption** along with steganography.
- **Multi-Level Steganography :** Concealing data within **multiple layers** of an image for added security.
- **Video & Audio Steganography :** Expanding the technique beyond images to **video and audio formats**.
- **AI-Resistant Steganography :** Developing advanced techniques to counter **steganalysis using deep learning**.

THANK YOU!

edunet foundation