CAPSTONE PROJECT

SECURE THE DATA TO PREVENT HACKERS.

Presented By:
Student Name : Aryan Raj
College Name & Department : St. Aloysis (Autonomous) College, Jabalpur (M.P.)

edunet foundation

OUTLINE

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

PROBLEM STATEMENT

SECURE DATA HIDING IN IMAGES USING STEGANOGRAPHY

TECHNOLOGY USED

Libraries :

- OpenCV(cv2) : Used for image processing and manipulation.
- Operating System(os) : Provides a way of using operating system-dependent functionality like reading or writing to the file system, managing environment variables, and handling processes.
- String : Provides a collection of string operations and constants that can be very useful for various text processing tasks.

Platforms :

- Python : The primary programming language used for implementing steganography projects.
- GitHub : A platform for hosting and sharing code repositories, where many steganography projects are available.

WOW FACTORS

Steganography, the art of hiding data within images, offers several "wow factors" that make it an impressive technique for securing data.

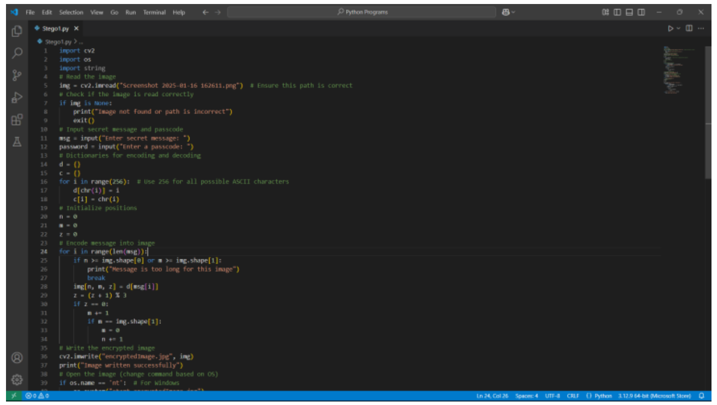
- **Invisibility** : Steganography embeds the secret data within the image in such a way that it remains invisible to the naked eye. This makes it a highly discreet method of communication and data protection.
- **Security** : By combining steganography with encryption, the security of the hidden data is significantly enhanced. Even if someone detects the hidden data, they would still need the encryption key to access the content.
- **Data Integrity** : The hidden data in steganography can be designed to be robust against common image manipulations, such as resizing and compression, ensuring the integrity of the hidden information.
- **Simplicity of Use** : Modern steganography tools and libraries make it relatively easy to embed and extract data, even for users without extensive technical knowledge.
- **Academic and Research Interest** : The technique continues to be a topic of interest in academic and research circles, leading to continuous improvements and innovations in the field.
- **Real-world Use Cases** : Steganography has been employed in real-world scenarios, such as :
  - **Covert Communication** : Used by intelligence agencies for discreet communication.
  - **Digital Forensics** : Embedding data within images for tracking and legal purposes.

END USERS

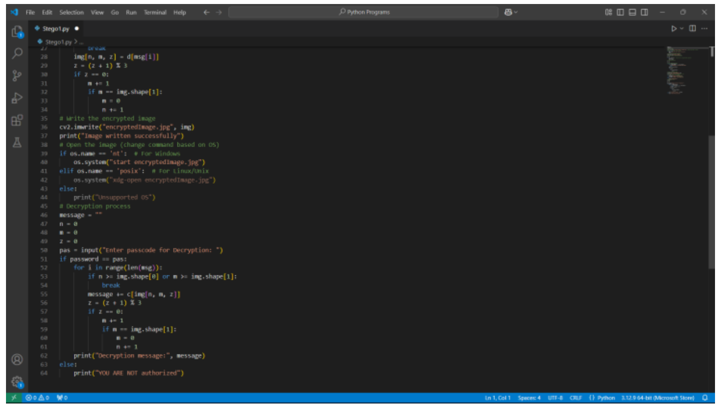
- **Military :**
  - <u>Secure Messaging</u> : Ensuring that strategic and operational communications are protected from interception.
  - <u>Intelligence Operations</u> : Concealing mission-critical information within images to prevent enemy detection.
- **Healthcare Providers :**
  - <u>Patient Data Security</u> : Embedding patient information within medical images to ensure confidentiality and compliance with regulations such as HIPAA.
  - <u>Research Data Protection</u> : Safeguarding sensitive research data from unauthorized access.
- **Financial Institutions :**
  - <u>Secure Transactions</u> : Protecting transaction data and financial information from cyber threats.
  - <u>Anti-Fraud Measures</u> : Embedding anti-counterfeiting measures within images on financial documents.
- **Educational Institutions :**
  - <u>Research Data Security</u> : Protecting research data and findings from unauthorized access and ensuring academic integrity.
  - <u>Student Information Protection</u> : Embedding student information within images to comply with data protection regulations.
- **Journalists and Media Professionals :**
  - <u>Protecting Sources</u> : Concealing the identities and information of confidential sources within images.
  - <u>Secure Reporting</u> : Ensuring sensitive information related to investigative journalism is kept confidential.
- **Individuals :**
  - Personal Data Security: Protecting personal documents and sensitive information by embedding them within personal images.
  - <u>Secure Communication</u> : Ensuring private messages and data are kept confidential when shared over the internet.

edunet foundation

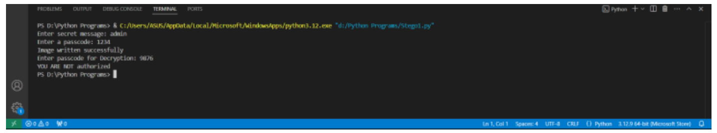
RESULTS



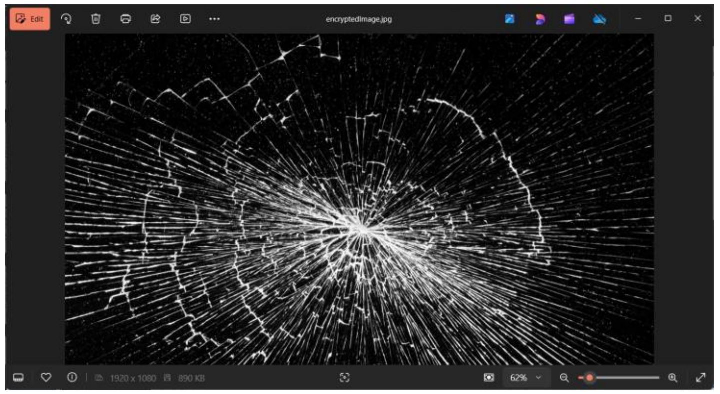
```python
import cv2
import os
import string
# Read the image
img = cv2.imread("Screenshot 2025-01-16 162611.png")  # Ensure this path is correct
# Check if the image is read correctly
if img is None:
    print("Image not found or path is incorrect")
    exit()
# Input secret message and passcode
msg = input("Enter secret message: ")
password = input("Enter a passcode: ")
# Dictionaries for encoding and decoding
d = {}
c = {}
for i in range(256):  # Use 256 for all possible ASCII characters
    d[chr(i)] = i
    c[i] = chr(i)
# Initialize positions
n = 0
m = 0
z = 0
# Encode message into image
for i in range(len(msg)):
    if n >= img.shape[0] or m >= img.shape[1]:
        print("Message is too long for this image")
        break
    img[n, m, z] = d[msg[i]]
    z = (z + 1) % 3
    if z == 0:
        m += 1
        if m == img.shape[1]:
            m = 0
            n += 1
# Write the encrypted image
cv2.imwrite("encryptedImage.jpg", img)
print("Image written successfully")
# Open the image (change command based on OS)
if os.name == 'nt':  # For Windows
```



```python
    img[n, m, z] = d[msg[i]]
    z = (z + 1) % 3
    if z == 0:
        m += 1
        if m == img.shape[1]:
            m = 0
            n += 1
# Write the encrypted image
cv2.imwrite("encryptedImage.jpg", img)
print("Image written successfully")
# Open the image (change command based on OS)
if os.name == 'nt':  # For Windows
    os.system("start encryptedImage.jpg")
elif os.name == 'posix':  # For Linux/Unix
    os.system("xdg-open encryptedImage.jpg")
else:
    print("Unsupported OS")
# Decryption process
message = ""
n = 0
m = 0
z = 0
pas = input("Enter passcode for Decryption: ")
if password == pas:
    for i in range(len(msg)):
        if n >= img.shape[0] or m >= img.shape[1]:
            break
        message += c[img[n, m, z]]
        z = (z + 1) % 3
        if z == 0:
            m += 1
            if m == img.shape[1]:
                m = 0
                n += 1
    print("Decryption message:", message)
else:
    print("YOU ARE NOT authorized")
```



```
PS D:\Python Programs> & C:/Users/ASUS/AppData/Local/Microsoft/WindowsApps/python3.12.exe "d:/Python Programs/Stego1.py"
Enter secret message: admin
Enter a passcode: 1234
Image written successfully
Enter passcode for Decryption: 9876
YOU ARE NOT authorized
PS D:\Python Programs>
```



CONCLUSION

Securing data hiding in images using steganography provides a sophisticated and robust method for protecting sensitive information. This technique leverages the power of invisibility, security, and versatility to ensure that data remains confidential and secure from unauthorized access. By embedding data within images, steganography offers a layer of protection that is difficult to detect, making it an ideal solution for covert communication, data integrity, and intellectual property protection.

Incorporating encryption alongside steganography further enhances security, ensuring that even if the hidden data is discovered, it remains inaccessible without the proper decryption key. This combination of techniques provides a comprehensive and reliable approach to safeguarding information in various fields, including government, military, healthcare, financial institutions, and more.

As technology continues to evolve, the applications and methodologies of steganography will also advance, providing even more robust and innovative solutions for data protection. By understanding and implementing these techniques, organizations and individuals can stay ahead of potential threats and maintain the confidentiality and integrity of their sensitive information.

GITHUB LINK

- https://github.com/matrix-77/AICTE-B4-2025.git

THANK YOU