CAPSTONE PROJECT

SECURE DATA HIDING IN IMAGES USING STEGANOGRAPHY

Presented By: Sarthak .P. Jadhav
Student Name : Sarthak Pandit Jadhav
College Name & Department : MES Abasaheb Garware

edunet
foundation

OUTLINE

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

PROBLEM STATEMENT

Conventional encryption techniques draw potential attackers because they make data transfer evident. Steganography is used in secure data hiding to embed private information in pictures so that unauthorized users cannot see it. This method maintains picture fidelity while guaranteeing secret communication. It covers the requirement for safe, imperceptible data transfer in digital forensics, cybersecurity, and private correspondence.

TECHNOLOGY USED

****Technology Used:-**

- **Language**: Python
- **Libraries**: OpenCV, NumPy, PIL (Pillow)
- **Algorithm**: LSB (Least Significant Bit) Steganography
- **File Formats**: PNG, JPEG for better security

WOW FACTORS

•**Invisible Data Embedding** :- The secret message is concealed within image pixels, rendering it invisible to the naked eye.

• **Secure Access**: Data can only be retrieved with the correct passcode, adding an extra layer of security.

•**Minimal Distortion**: Suspicion is avoided because the original image quality is essentially preserved.

•**User-Friendly**: Simple Python encryption and decryption using NumPy and OpenCV.

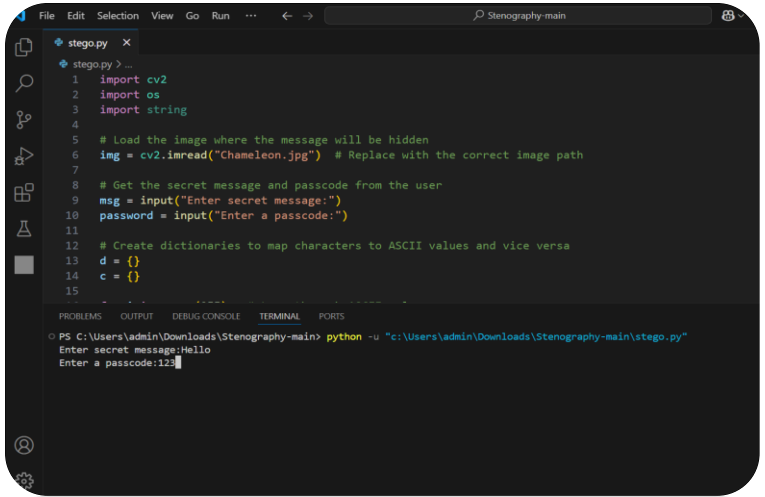
•**Practical Applications**Helpful for digital watermarking, safe communication, and safeguarding private data.

END USERS

- Who benefits from this technology?
- Professionals in cybersecurity :- For safe communication and data security.
- Law enforcement organizations :- To conceal private communications and sensitive intelligence.
- Journalists & Activists :– Ensures safe information exchange in high-risk areas.
- Government & Military :- Protects secret operations and classified communications.
- Everyday Users :– Protects personal messages and sensitive information from unauthorized access.

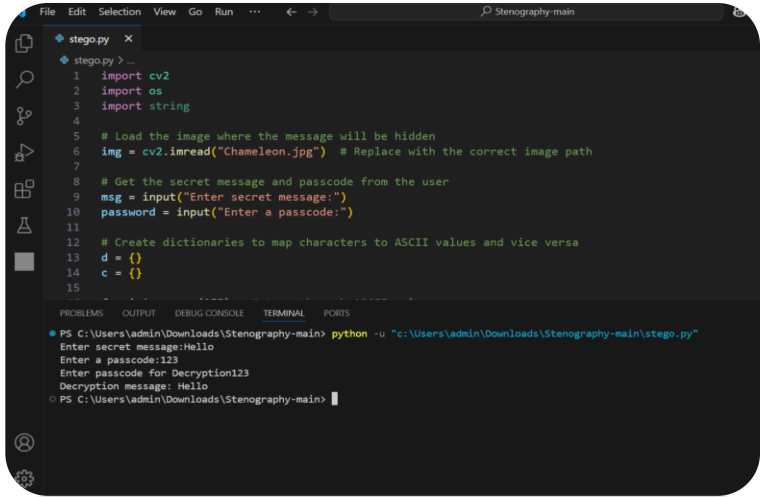
RESULTS

- Screenshots of the outcome (min 3)


File Edit Selection View Go Run ···
Stenography-main
stego.py
stego.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

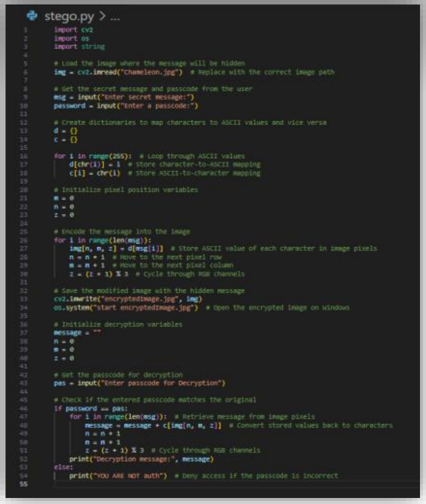
```python
import cv2
import os
import string

# Load the image where the message will be hidden
img = cv2.imread("Chameleon.jpg")  # Replace with the correct image path

# Get the secret message and passcode from the user
msg = input("Enter secret message:")
password = input("Enter a passcode:")

# Create dictionaries to map characters to ASCII values and vice versa
d = {}
c = {}

```

```
PS C:\Users\admin\Downloads\Stenography-main> python -u "c:\Users\admin\Downloads\Stenography-main\stego.py"
Enter secret message:Hello
Enter a passcode:123
```

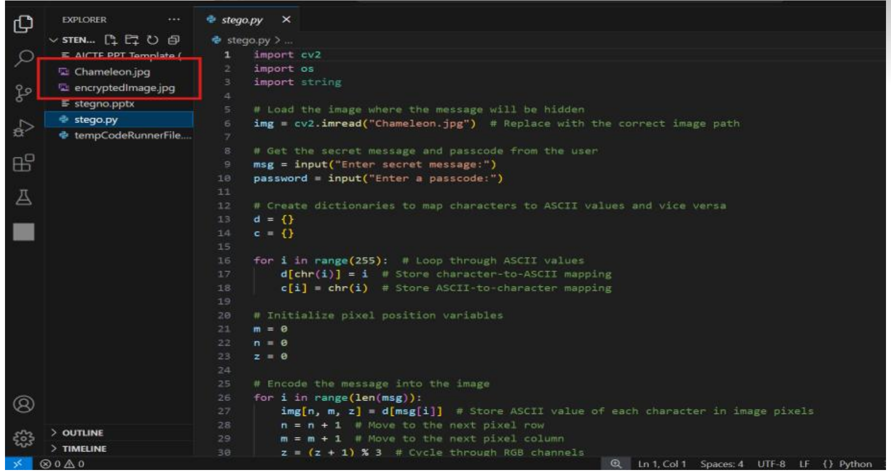

File Edit Selection View Go Run ···
Stenography-main
stego.py
stego.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```python
import cv2
import os
import string

# Load the image where the message will be hidden
img = cv2.imread("Chameleon.jpg")  # Replace with the correct image path

# Get the secret message and passcode from the user
msg = input("Enter secret message:")
password = input("Enter a passcode:")

# Create dictionaries to map characters to ASCII values and vice versa
d = {}
c = {}

```

```
PS C:\Users\admin\Downloads\Stenography-main> python -u "c:\Users\admin\Downloads\Stenography-main\stego.py"
Enter secret message:Hello
Enter a passcode:123
Enter passcode for Decryption123
Decryption message: Hello
PS C:\Users\admin\Downloads\Stenography-main>
```



```python
import cv2
import os
import string

# Load the image where the message will be hidden
img = cv2.imread("Chameleon.jpg")  # Replace with the correct image path

# Get the secret message and passcode from the user
msg = input("Enter secret message:")
password = input("Enter a passcode:")

# Create dictionaries to map characters to ASCII values and vice versa
d = {}
c = {}

for i in range(255):  # Loop through ASCII values
    d[chr(i)] = i  # Store character-to-ASCII mapping
    c[i] = chr(i)  # Store ASCII-to-character mapping

# Initialize pixel position variables
m = 0
n = 0
z = 0

# Encode the message into the image
for i in range(len(msg)):
    img[n, m, z] = d[msg[i]]  # Store ASCII value of each character in image pixels
    n = n + 1  # Move to the next pixel row
    m = m + 1  # Move to the next pixel column
    z = (z + 1) % 3  # Cycle through RGB channels

# Save the modified image with the hidden message
cv2.imwrite("encryptedImage.jpg", img)
os.system("start encryptedImage.jpg")  # Open the encrypted image on Windows

# Initialize decryption variables
message = ""
n = 0
m = 0
z = 0

# Get the passcode for decryption
pas = input("Enter passcode for Decryption")

# Check if the entered passcode matches the original
if password == pas:
    for i in range(len(msg)):  # Retrieve message from image pixels
        message = message + c[img[n, m, z]]  # Convert stored values back to characters
        n = n + 1
        m = m + 1
        z = (z + 1) % 3  # Cycle through RGB channels
    print("Decryption message:", message)
else:
    print("YOU ARE NOT auth")  # Deny access if the passcode is incorrect

```



```python
import cv2
import os
import string

# Load the image where the message will be hidden
img = cv2.imread("Chameleon.jpg")  # Replace with the correct image path

# Get the secret message and passcode from the user
msg = input("Enter secret message:")
password = input("Enter a passcode:")

# Create dictionaries to map characters to ASCII values and vice versa
d = {}
c = {}

for i in range(255):  # Loop through ASCII values
    d[chr(i)] = i  # Store character-to-ASCII mapping
    c[i] = chr(i)  # Store ASCII-to-character mapping

# Initialize pixel position variables
m = 0
n = 0
z = 0

# Encode the message into the image
for i in range(len(msg)):
    img[n, m, z] = d[msg[i]]  # Store ASCII value of each character in image pixels
    n = n + 1  # Move to the next pixel row
    m = m + 1  # Move to the next pixel column
    z = (z + 1) % 3  # Cycle through RGB channels
```

CONCLUSION

- Conclude your project concerning your problem statement
- Important Takeaways: Steganography conceals information within pictures so that unauthorized users cannot see it.
- Enhanced Security: Uses encryption and covert communication to safeguard private data.
- Restrictions:-

*Can be found with the aid of steganalysis instruments.

*Images have limited capacity to store data.Hidden data may be corrupted by compression or modification.

GITHUB LINK

- Make sure that there should be readme file:-
- https://github.com/Sarthak6421/Data-Hiding-Using-Steganography.git

THANK YOU

edunet
foundation