Capstone Project Presentation: Image Steganography Using AES Encryption

This presentation details a Capstone Project on Image Steganography, leveraging AES Encryption for enhanced data security.

🔐 Image Steganography Using AES Encryption

Capstone Project – B.Tech Cybersecurity

Presented By:

Shreyanshu Bhartiya

College:

MBM University,Jodhpur

Organisation:-

Edunet Foundation

Problem Statement – Hiding Information in Images

- In today's digital world, data security is critical.
- This project uses **steganography** to hide secret messages inside images.
- Combined with **AES encryption**, it ensures message confidentiality and stealth.
- The output image looks identical to the original but carries hidden data.
- The recipient can decrypt and recover the hidden message using a secret key.

Technology & Tools Used

System Requirements:

- Python 3.10
- VS Code or Jupyter Notebook
- PNG image input/output

Libraries Used:

- opencv-python (cv2)
- pycryptodome (AES from Crypto.Cipher, pad/unpad from Crypto.Util.Padding, get_random_bytes from Crypto.Random)
- hashlib, string, os, numpy

Step-by-Step Workflow

1

1. Encrypt Message

Using AES with a secret key (CBC mode).

2

2. Convert to Binary & Add Header

Encrypted bytes converted to binary with a 32-bit data length header.

3

3. Embed in LSBs

Binary data embedded into the least significant bits of image pixels.

4

4. Save Stego-Image

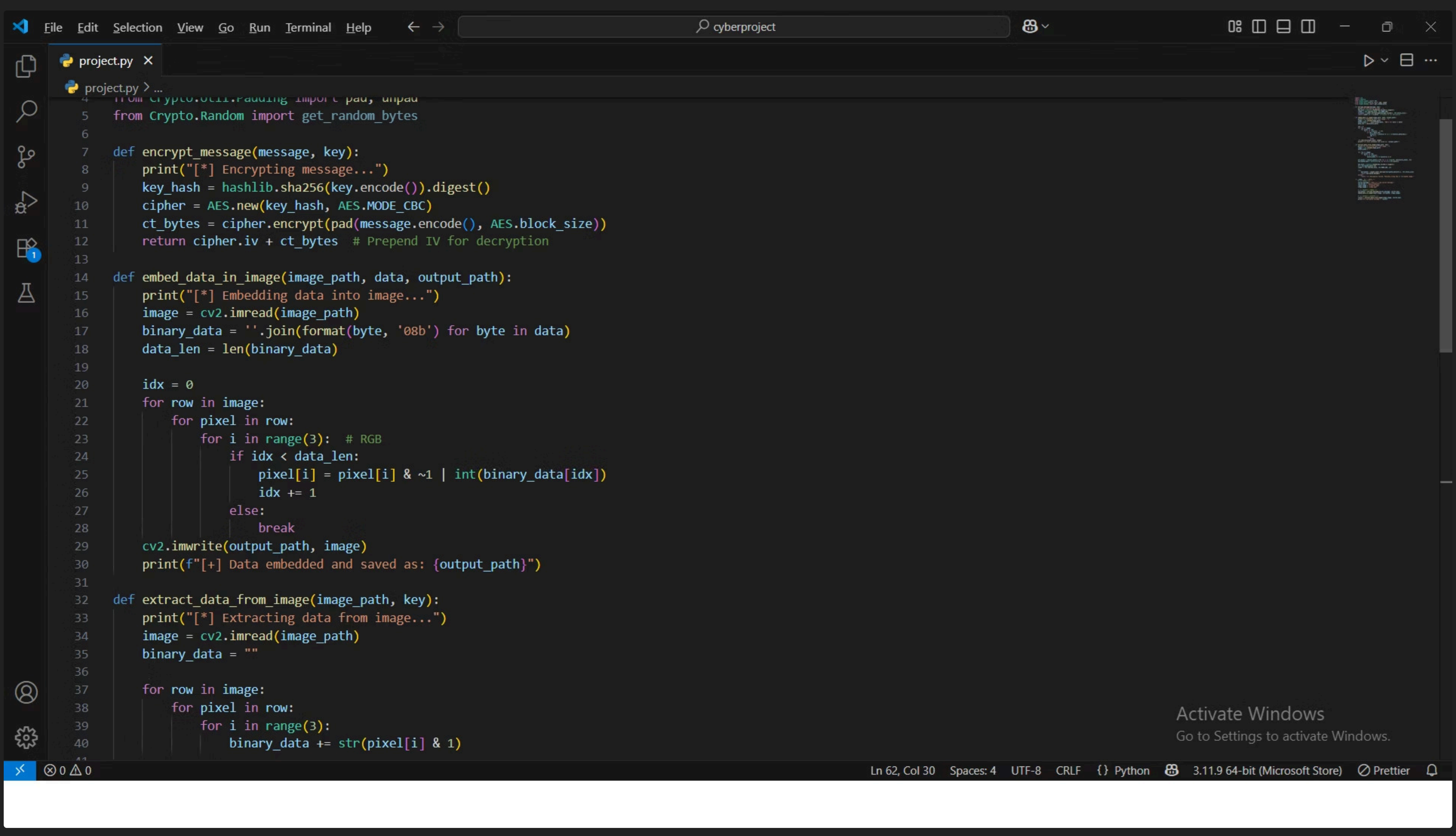
The image with the hidden data is saved.

5

5. Extract & Decrypt

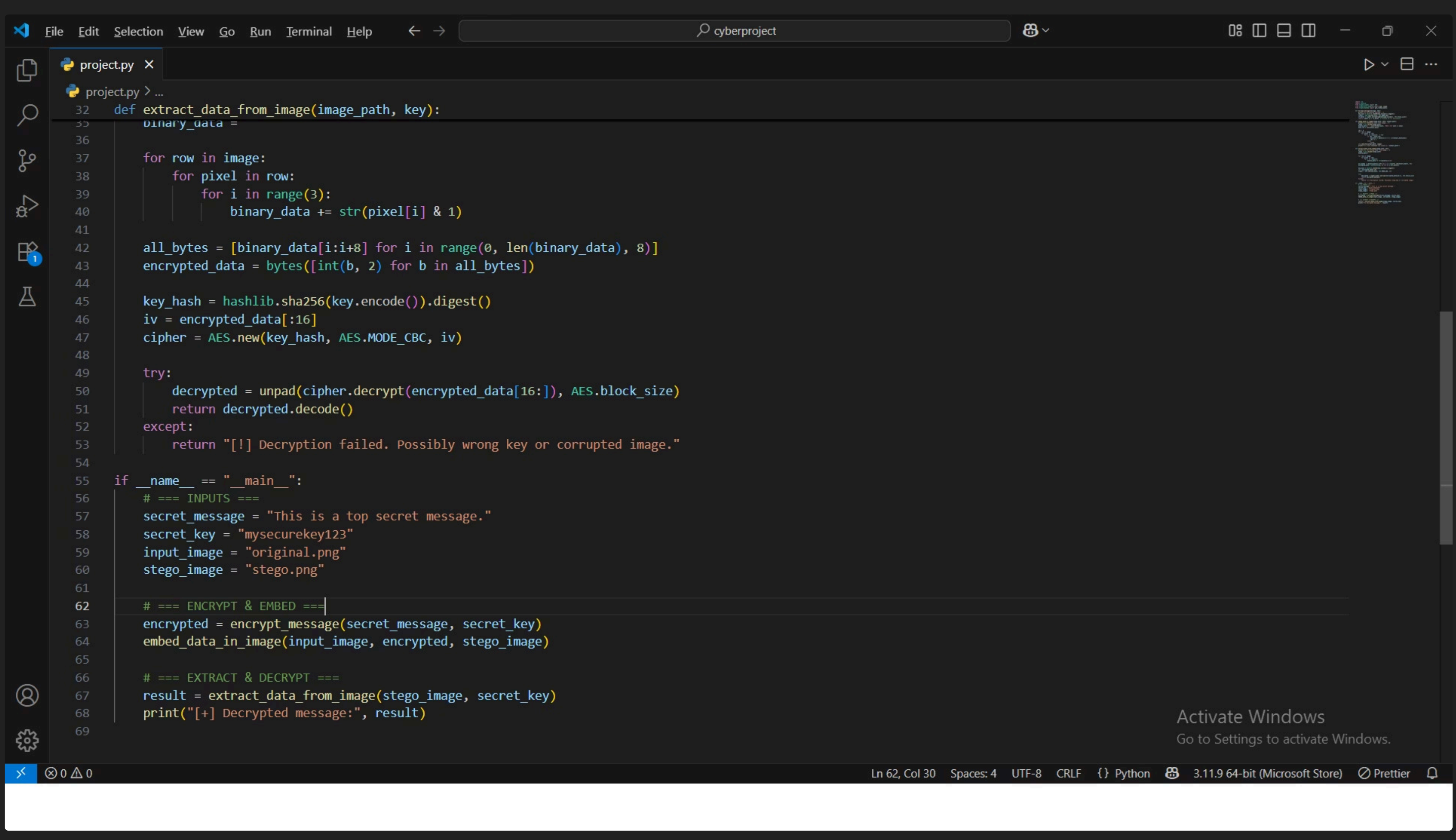
LSBs extracted, header read, data reconstructed, and decrypted with the same key.

Screenshots – Code Execution & Output


File Edit Selection View Go Run Terminal Help
cyberproject
project.py
project.py > ...
Activate Windows
Go to Settings to activate Windows.
Ln 62, Col 30 Spaces: 4 UTF-8 CRLF {} Python 3.11.9 64-bit (Microsoft Store) Prettier

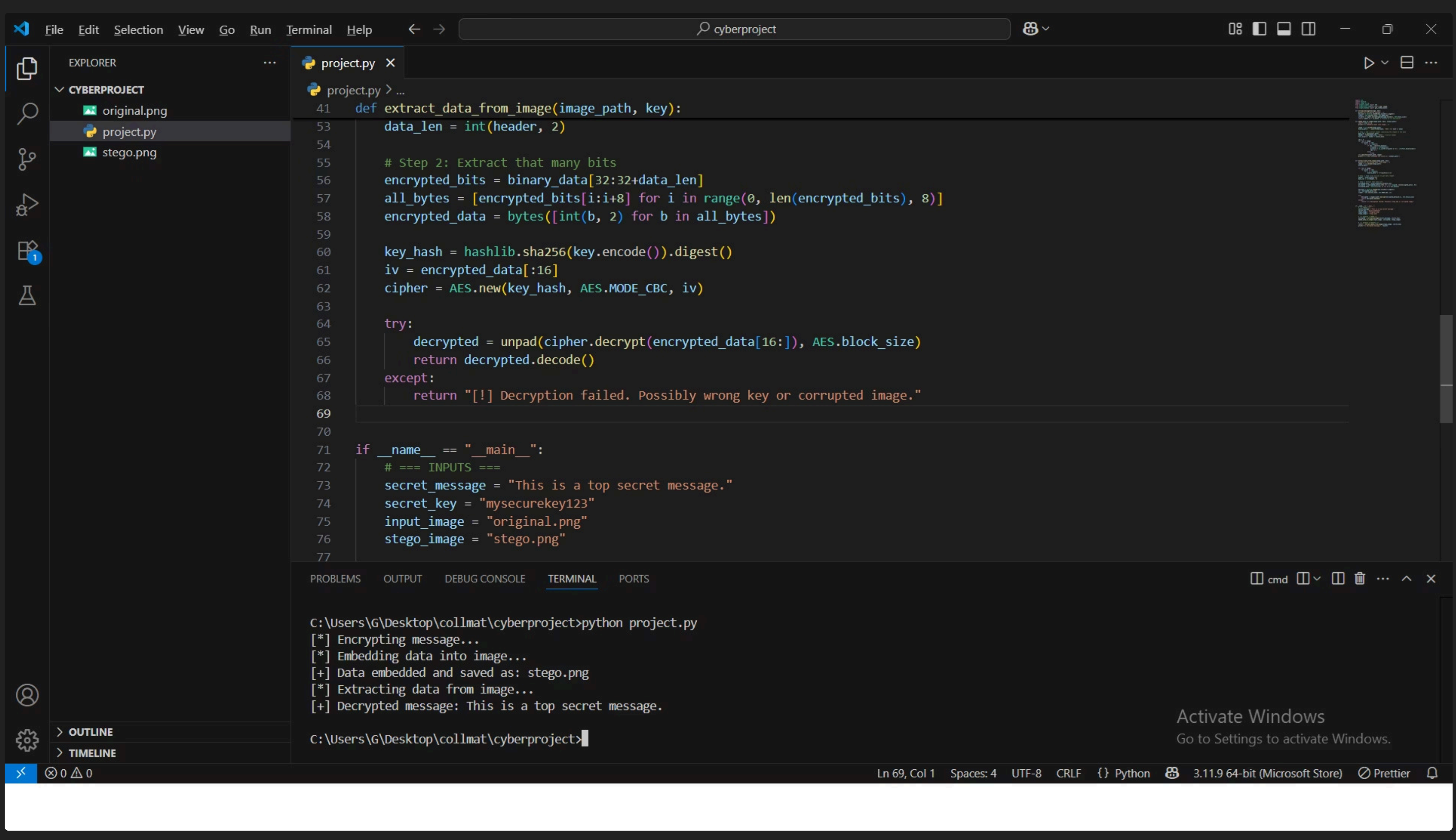
```python
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes

def encrypt_message(message, key):
    print("[*] Encrypting message...")
    key_hash = hashlib.sha256(key.encode()).digest()
    cipher = AES.new(key_hash, AES.MODE_CBC)
    ct_bytes = cipher.encrypt(pad(message.encode(), AES.block_size))
    return cipher.iv + ct_bytes  # Prepend IV for decryption

def embed_data_in_image(image_path, data, output_path):
    print("[*] Embedding data into image...")
    image = cv2.imread(image_path)
    binary_data = ''.join(format(byte, '08b') for byte in data)
    data_len = len(binary_data)

    idx = 0
    for row in image:
        for pixel in row:
            for i in range(3):  # RGB
                if idx < data_len:
                    pixel[i] = pixel[i] & ~1 | int(binary_data[idx])
                    idx += 1
                else:
                    break
    cv2.imwrite(output_path, image)
    print(f"[+] Data embedded and saved as: {output_path}")

def extract_data_from_image(image_path, key):
    print("[*] Extracting data from image...")
    image = cv2.imread(image_path)
    binary_data = ""

    for row in image:
        for pixel in row:
            for i in range(3):
                binary_data += str(pixel[i] & 1)
```

Code(2)


File Edit Selection View Go Run Terminal Help
cyberproject
project.py
project.py > ...
Activate Windows
Go to Settings to activate Windows.
Ln 62, Col 30 Spaces: 4 UTF-8 CRLF {} Python 3.11.9 64-bit (Microsoft Store) Prettier

```python
def extract_data_from_image(image_path, key):
    binary_data = ""

    for row in image:
        for pixel in row:
            for i in range(3):
                binary_data += str(pixel[i] & 1)

    all_bytes = [binary_data[i:i+8] for i in range(0, len(binary_data), 8)]
    encrypted_data = bytes([int(b, 2) for b in all_bytes])

    key_hash = hashlib.sha256(key.encode()).digest()
    iv = encrypted_data[:16]
    cipher = AES.new(key_hash, AES.MODE_CBC, iv)

    try:
        decrypted = unpad(cipher.decrypt(encrypted_data[16:]), AES.block_size)
        return decrypted.decode()
    except:
        return "[!] Decryption failed. Possibly wrong key or corrupted image."

if __name__ == "__main__":
    # === INPUTS ===
    secret_message = "This is a top secret message."
    secret_key = "mysecurekey123"
    input_image = "original.png"
    stego_image = "stego.png"

    # === ENCRYPT & EMBED ===
    encrypted = encrypt_message(secret_message, secret_key)
    embed_data_in_image(input_image, encrypted, stego_image)

    # === EXTRACT & DECRYPT ===
    result = extract_data_from_image(stego_image, secret_key)
    print("[+] Decrypted message:", result)
```

Execution


File Edit Selection View Go Run Terminal Help
cyberproject
EXPLORER
CYBERPROJECT
original.png
project.py
stego.png
OUTLINE
TIMELINE
project.py
project.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Activate Windows
Go to Settings to activate Windows.
Ln 69, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.11.9 64-bit (Microsoft Store) Prettier

```python
def extract_data_from_image(image_path, key):
    data_len = int(header, 2)

    # Step 2: Extract that many bits
    encrypted_bits = binary_data[32:32+data_len]
    all_bytes = [encrypted_bits[i:i+8] for i in range(0, len(encrypted_bits), 8)]
    encrypted_data = bytes([int(b, 2) for b in all_bytes])

    key_hash = hashlib.sha256(key.encode()).digest()
    iv = encrypted_data[:16]
    cipher = AES.new(key_hash, AES.MODE_CBC, iv)

    try:
        decrypted = unpad(cipher.decrypt(encrypted_data[16:]), AES.block_size)
        return decrypted.decode()
    except:
        return "[!] Decryption failed. Possibly wrong key or corrupted image."


if __name__ == "__main__":
    # === INPUTS ===
    secret_message = "This is a top secret message."
    secret_key = "mysecurekey123"
    input_image = "original.png"
    stego_image = "stego.png"
```

```
C:\Users\G\Desktop\collmat\cyberproject>python project.py
[*] Encrypting message...
[*] Embedding data into image...
[+] Data embedded and saved as: stego.png
[*] Extracting data from image...
[+] Decrypted message: This is a top secret message.

C:\Users\G\Desktop\collmat\cyberproject>
```

Original Photo

Output Image

GitHub Repository

Access the full source code and comprehensive documentation on our GitHub repository:

hhttps://github.com/ShreyanshuBhartiya/Steganographyproject.git

The repository includes:

- Full source code
- Screenshots of execution
- requirements.txt for dependencies
- Detailed README.md for setup and usage

Conclusion & Learnings

Key Achievements:

- Successfully implemented steganography using AES in Python.
- Achieved stealth and confidentiality by combining cryptography and image-based hiding.
- Effective for PNG images; avoids distortion or compression loss.

Challenges & Improvements:

- **Challenges faced:** Bit overflow errors in pixel values, message extraction hanging without a header.
- **Improvements:** Add support for JPEG/DCT-based steganography, GUI or web interface for user-friendly use.

References & Resources

- PyCryptodome Documentation https://pycryptodome.readthedocs.io/
- OpenCV Library Docs https://docs.opencv.org/
- Johnson, N. F., & Jajodia, S. (1998). *Exploring Steganography: Seeing the Unseen.*
- ResearchGate – Image Steganography using LSB technique https://www.researchgate.net/
- GeeksforGeeks – Image-based Steganography in Python

🙏 Thank You!

Thank you for your time and attention.

Please feel free to ask any questions.

Contact: **shreyanshubhartiya@gmail.com**