

---

# **CAPSTONE PROJECT**

## **SECURE DATA HIDING IN IMAGE USING STEGANOGRAPHY**

**Presented By: Tejas. P**  
**Student Name : Tejas. P**  
**College Name & Department : ATME College of Engineering**  
**[Computer Science and Engineering]**

---

## OUTLINE

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

---

# PROBLEM STATEMENT

This project focuses on secure data hiding using steganography and AES encryption. Sensitive data is encrypted and embedded in images using the Least Significant Bit (LSB) technique. The goal is to ensure confidentiality, imperceptibility, and data integrity. Even if intercepted, the hidden data remains secure without the correct decryption key.

---

# TECHNOLOGY USED

## ❑ Programming Language:

- **Python** – Used for implementing steganography and encryption.

## ❑ Cryptography:

- **AES (Advanced Encryption Standard)** – Ensures secure message encryption.
- **cryptography** library – Provides AES encryption and padding.

## ❑ Image Processing:

- **Steganography (Least Significant Bit - LSB)** – Hides encrypted data in images.
- **OpenCV (cv2)** – Reads, modifies, and saves images.
- **NumPy** – Efficient pixel manipulation.

## ❑ Hashing:

- **SHA-256 (hashlib)** – Derives a secure encryption key from a password.

## ❑ Platforms & Environments:

- **Windows**– Runs on OS with Python support.
- **Jupyter Notebook**– Recommended development environments

## ❑ File Handling & System Integration:

- **OS Module** – Used for file handling, generating random IV (Initialization Vector) and opening encrypted images.

---

# WOW FACTORS

## ☐ **AES Encryption + Steganography = Double Security**

- Unlike basic steganography, this project **encrypts the message using AES** before embedding it in an image.
- Even if someone extracts hidden data, they cannot decrypt it without the **correct password**.

## ☐ **Least Significant Bit (LSB) Steganography for Invisible Data Hiding**

- The **LSB technique ensures imperceptibility**, meaning the image looks unchanged to the human eye.
- It avoids noticeable distortions, making detection extremely difficult.

## ☐ **High Efficiency & Minimal Image Distortion**

- Embeds data **without significantly increasing file size or affecting image quality**.
- Uses **optimal pixel selection** for better security and reduced modification.

## ☐ **Strong Key Derivation for Enhanced Security**

- Uses **SHA-256 hashing** to derive a strong AES key from the password.
- Prevents weak key attacks, ensuring **robust encryption**.

## ☐ **User-Friendly Input & Output**

- Simple **command-line interface** for encoding and decoding.
- Clear **error messages and warnings** if an image is too small or the wrong password is entered.

## ☐ **Resistance to Steganalysis (Detection Attempts)**

- Since AES encryption randomizes data patterns, **it avoids common statistical detection methods**.
- Unlike traditional LSB steganography, the encrypted payload appears as random noise, **making detection much harder**.

## **What Makes This Project Stand Out?**

- **Dual-layer security** (AES + LSB Steganography)
- **Minimal image quality loss**
- **Cross-platform & automated image opening**
- **Hard to detect using standard steganalysis tools**
- **Supports large encrypted messages with optimized pixel storage**

---

## END USERS

### ☐ **Cybersecurity & Intelligence Agencies**

- Used for **covert communication** in intelligence operations.
- Helps securely transmit sensitive messages without attracting attention.

### ☐ **Journalists & Whistleblowers**

- Can **hide confidential information** within images to avoid censorship.
- Ensures secure sharing of critical news reports or evidence.

### ☐ **Government & Military Organizations**

- Useful for **secure transmission of classified information**.
- Enhances communication security in sensitive operations.

### ☐ **Corporate & Business Professionals**

- Protects **trade secrets and confidential data** from corporate espionage.
- Ensures **secure internal communications** within an organization.

### ☐ **Privacy-Conscious Individuals**

- Can be used by individuals to **safeguard personal data** from cyber threats.
- Helps in securely **hiding passwords, private messages, or sensitive documents** in images.

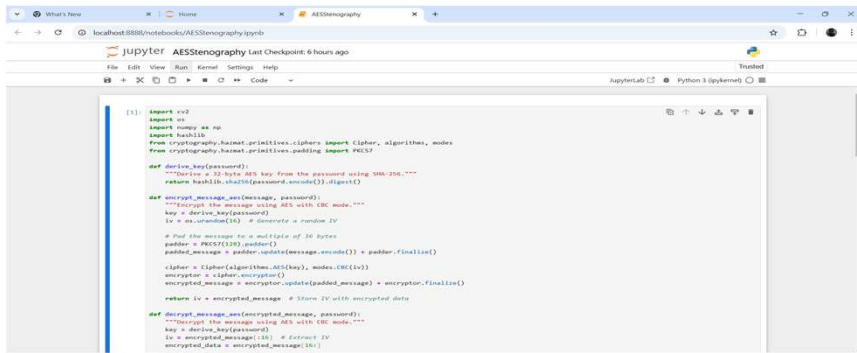
### ☐ **Digital Forensics & Law Enforcement**

- Helps **track hidden messages in images** sent by criminals.
- Can be used in **investigations to retrieve embedded evidence**.

### ☐ **Ethical Hackers & Security Researchers**

- Useful for **testing steganographic security methods**.
- Can be enhanced to **evaluate and improve modern data-hiding techniques**.

# RESULTS



```
[1]: import sys
import os
import sys as sys
import hashlib
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.primitives.padding import PKCS7

def derive_key(password):
    """Derive a 32-byte AES key from the password using SHA-256"""
    return hashlib.sha256(password.encode()).digest()

def encrypt_message(msg, password):
    """Encrypt the message using AES with CBC mode"""
    key = derive_key(password)
    iv = os.urandom(16) # Generate a random IV

    # Pad the message to a multiple of 16 bytes
    padder = PKCS7(16).padder()
    padded_message = padder.update(msg.encode()) + padder.finalize()

    cipher = Cipher(algorithms.AES(key), modes.CBC(iv))
    encryptor = cipher.encryptor()
    encrypted_message = encryptor.update(padded_message) + encryptor.finalize()

    return iv + encrypted_message # Store IV with encrypted data

def decrypt_message(msg, password):
    """Decrypt the message using AES with CBC mode"""
    key = derive_key(password)
    iv = msg[:16] # Extract IV
    encrypted_data = msg[16:]
    decryptor = Cipher(algorithms.AES(key), modes.CBC(iv)).decryptor()
    decrypted_message = decryptor.update(encrypted_data) + decryptor.finalize()
    return decrypted_message.decode()

if __name__ == '__main__':
    if len(sys.argv) != 3:
        print('Usage: python3 stenography.py [message] [password]')
        sys.exit(1)
    message = sys.argv[1]
    password = sys.argv[2]
    if message[0] == 'e':
        encrypted_message = encrypt_message(message, password)
        print('Encrypted message: %s' % encrypted_message.hex())
    elif message[0] == 'd':
        decrypted_message = decrypt_message(message, password)
        print('Decrypted message: %s' % decrypted_message)
```

Figure 1: Source code.



```
Enter secret message: Hi My Name is Tejas
Enter password: 54321
Message encrypted successfully!
Enter passcode for Decryption: 54321
Decrypted message: Hi My Name is Tejas

[ ]:
```

Figure 3: Output of the project.



Figure 3: Input image Mypic.png

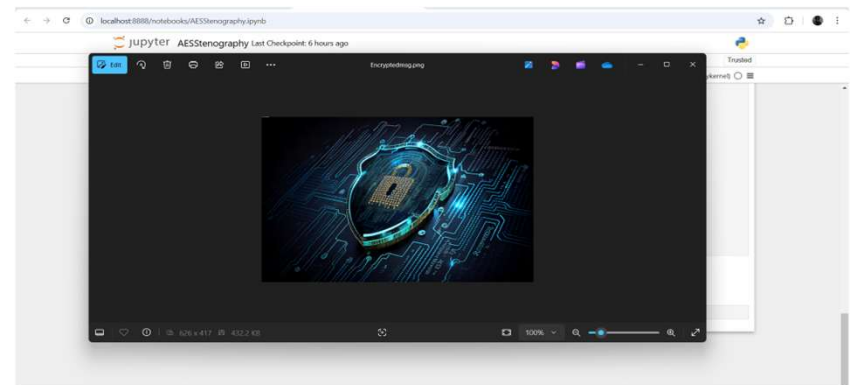


Figure 4: Output image Encryptedmsg.png

---

## CONCLUSION

- This project addresses the problem of **secure communication** by implementing **data hiding in images using steganography and AES encryption**. Traditional encryption alone may not be enough to prevent interception, but by embedding encrypted messages into images using the **Least Significant Bit (LSB) technique**, this system ensures both **confidentiality and imperceptibility**.
- The integration of **AES encryption** before embedding enhances security, ensuring that even if the hidden data is extracted, it remains unreadable without the correct key. This method provides a **robust, efficient, and undetectable** way to transmit sensitive information securely. It can be applied in **cybersecurity, intelligence operations, digital forensics, and secure communication**.
- In the future, this project can be expanded with **advanced steganography techniques, AI-driven detection resistance, and multi-image encoding** to further enhance security and usability.



---

## GITHUB LINK

<https://github.com/TejasP15/Stenography.git>

---

# FUTURE SCOPE

## ❑ Advanced Steganography Techniques

- Implement **adaptive steganography** to hide data more securely by analyzing image properties.
- Use **edge-based or frequency-domain steganography (DCT, DWT)** to make data embedding more robust against compression and attacks.

## ❑ Support for Multiple File Formats

- Extend support beyond PNG to **JPEG, BMP, GIF, and TIFF** for wider usability.
- Optimize encoding for **compressed formats** without degrading security.

## ❑ Enhanced Security with Multi-Layer Encryption

- Integrate **Elliptic Curve Cryptography (ECC) or RSA** for key exchange.
- Use **hybrid encryption (AES + RSA)** to improve security while maintaining efficiency.

## ❑ AI & Machine Learning for Detection Resistance

- Develop **AI-powered steganographic methods** that adapt based on detection algorithms.
- Use **GANs (Generative Adversarial Networks)** to generate stego - images that are nearly impossible to detect.

## ❑ Cross-Platform & Mobile Application

- Build a **GUI-based desktop application** for ease of use.
- Develop a **mobile app (Android/iOS)** for real-time secure communication using steganography.

## ❑ Cloud-Based Secure Steganography

- Implement a **cloud-based secure messaging system** using encrypted steganography.
- Enable users to securely upload and retrieve stego - images from a cloud platform.

## ❑ Resistance to Steganalysis Attacks

- Improve robustness against **LSB steganalysis, histogram analysis, and statistical detection techniques**.
- Implement **randomized bit embedding** to make detection more difficult.

By incorporating these future enhancements, your project can evolve into a **highly secure, AI-driven, and user-friendly steganographic communication system** with real-world applications in **cybersecurity, intelligence, and private communication**.

---

**THANK YOU**