# CAPSTONE PROJECT

# SECURE DATA HIDING IN IMAGE USING STEGANOGRAPHY

**Student Name : Nandipati Avinash Chowdary**
**College Name & Department : GITAM University, CSE - Cyber Security**

edunet foundation

# OUTLINE

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

# PROBLEM STATEMENT

With the rise of cyber threats and data breaches, secure communication has become a significant challenge. Traditional encryption methods can attract attention, making them susceptible to attacks. This project explores **image steganography**, where secret messages are embedded within an image, making them invisible to unauthorized users.

# TECHNOLOGY USED

**Programming Language:** Python

- **Libraries Used:** OpenCV (cv2), NumPy, OS
- **Concepts Implemented:**
  - Image Processing
  - Pixel Manipulation
  - Steganography Techniques

# WOW FACTORS

- **Invisible Data Embedding:** The message is hidden within image pixels, making it undetectable.
- **Password Protection:** Only users with the correct passcode can retrieve the hidden message.
- **Simple yet Effective:** A lightweight solution for secure message transmission without raising suspicion.
- **Automation:** Image encryption and decryption occur seamlessly using a simple script.

# END USERS

**Journalists & Activists:** Secure communication without leaving digital traces.

- **Government & Military:** Safeguarding classified information.
- **Corporate Sector:** Protecting confidential business strategies.
- **General Users:** Sending private messages securely over public networks.

# RESULTS

- CODE

steg-1.py 2

steg-1.py > encode_message

```python
import cv2
import numpy as np

def encode_message(img, message, password):
    """Embeds a secret message into an image using simple pixel encoding."""
    # Convert message to ASCII values + delimiter
    message += "~"  # Special character to mark end of message
    ascii_values = [ord(ch) for ch in message]

    height, width, _ = img.shape
    total_pixels = height * width * 3  # RGB channels

    if len(ascii_values) > total_pixels:
        print("Error: Message is too long for this image!")
        return None

    flat_image = img.flatten()

    # Encode message into pixels
    for i, value in enumerate(ascii_values):
        flat_image[i] = value

    # Reshape image to original shape
    encoded_img = flat_image.reshape(height, width, 3)

    # Save encrypted image
    cv2.imwrite("encoded_image.png", encoded_img)
    print("Encryption complete. Image saved as 'encoded_image.png'")

def decode_message(img, password):
    """Extracts a secret message from an image using pixel decoding."""

    flat_image = img.flatten()
    decoded_chars = []

    for pixel in flat_image:
        char = chr(pixel)
        if char == "~":  # Stop at delimiter
            break
        decoded_chars.append(char)

    return "".join(decoded_chars)

if __name__ == "__main__":
```

Ln 21, Col 30 Spaces: 4 UTF-8 LF Python 3.13.2 Go Live

# RESULTS

- EXECUTION

```
(myenv) avi_chows@Avinashs-Mac VSCODE % LS
encoded_image.png       gym.html                myenv                   steg-1.py               steg.py
(myenv) avi_chows@Avinashs-Mac VSCODE % python3 steg-1.py
Enter 'e' to encode or 'd' to decode: e
Enter the secret message: avinash's project
Set a passcode: 123456
Encryption complete. Image saved as 'encoded_image.png'
(myenv) avi_chows@Avinashs-Mac VSCODE % python3 steg-1.py
Enter 'e' to encode or 'd' to decode: d
Enter passcode to decrypt: 123456
Decrypted message: avinash's project
(myenv) avi_chows@Avinashs-Mac VSCODE %
```

# CONCLUSION

- This project demonstrates how **steganography** can be used to hide sensitive data in images securely. By leveraging image processing techniques, we ensure confidential data transmission without raising suspicion. The approach is simple yet effective in securing communication.

# GITHUB LINK

- https://github.com/CyberEnchanter-0001/Stegnography

# THANK YOU

edunet
foundation