CAPSTONE PROJECT

Secure data hiding in image using stegnography

Presented By: Mohammad Arslan Kana
Student Name : Mohammad Arslan Kana
College Name & Department : North Campus delina , Btech CSE

edunet foundation

OUTLINE

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

Problem Statement

- **Existing Issue:** Traditional steganography lacks robust encryption, exposing hidden data to extraction using “steg analysis”.

- **Solution:** Embed messages in images using XOR encryption and Least Significant Bit (LSB) steganography to ensure confidentiality even if the data is found by Steg analysis.

Technology used

- **Libraries:** OpenCV (image processing), struct (binary data handling), os (system operations).
- **Platform:** Python 3.x.
- **IDE :** vs code

Wow factors

- **Unique Features:**
  a. Dual Security: Combines XOR encryption + LSB steganography.
  b. Dynamic Key Handling: Repeats secret key for variable-length messages.
  c. Error Handling: Checks image size to prevent overflow.

End users

- **Target Audience:** Cybersecurity professionals, journalists, corporations handling sensitive data.
- **Use Case:** Securely share credentials or confidential notes via social media images.

Results : Message hidden in secret_image.png with no visual artifacts.

```python
    if img is None:
        print("Error: Couldn't load image")
        return

    # Encrypt message using XOR with secret key
    encrypted_data = []
    key_len = len(secret_key)
    for i, char in enumerate(message):#Each char represents one character from the secret message
        key_char = secret_key[i % key_len]#if key is hsorter then message then it makes it loop from start again of the key
        encrypted_data.append(ord(char) ^ ord(key_char))#ord(char): Converts the message character to its ASCII value.
                                                        #ord(key_char): Converts the corresponding key character to its ASCII value.

    # Add message length header
    data_length = len(encrypted_data)
    length_header = struct.pack('!I', data_length)

    # Embed data in image
    row, col, channel = 0, 0, 0
    img_row, img_col, _ = img.shape

    # Store length header , The first few pixels store the message length
    for byte in length_header:
        if row >= img_row or col >= img_col:
            print("Error: Image too small")
            return
        img[row, col, channel] = byte
        row += 1
        col += 1
        channel = (channel + 1) % 3

    # Store encrypted message
    for byte in encrypted_data:
        if row >= img_row or col >= img_col:
            print("Error: Image too small")
            return
        img[row, col, channel] = byte
        row += 1
        col += 1
        channel = (channel + 1) % 3

    cv2.imwrite('secret_image.png', img)
    print("Message hidden in secret_image.png")
    os.system('start secret_image.png')

def reveal_message(image_path):
    img = cv2.imread(image_path)
    if img is None:
        print("Error: Couldn't load image")
        return

    row, col, channel = 0, 0, 0
    img_row, img_col, _ = img.shape

    # Read length header
    length_bytes = []
    for _ in range(4):
        if row >= img_row or col >= img_col:
            print("Error: Invalid image format")
            return
        length_bytes.append(img[row, col, channel])
        row += 1
        col += 1
        channel = (channel + 1) % 3

    data_length = struct.unpack('!I', bytes(length_bytes))[0]

    # Read encrypted data
    encrypted_data = []
    for _ in range(data_length):
        if row >= img_row or col >= img_col:
            print("Error: Data corrupted")
            return
        encrypted_data.append(img[row, col, channel])
        row += 1
        col += 1
        channel = (channel + 1) % 3

    # Decrypt message
    secret_key = input("Enter decryption key: ")
    decrypted = []
    key_len = len(secret_key)
    for i, byte in enumerate(encrypted_data):
        key_char = secret_key[i % key_len]
        decrypted.append(chr(byte ^ ord(key_char)))

    print("Hidden message:", ''.join(decrypted))

# Main program
choice = input("Hide(h) or reveal(r) message? (h/r): ").lower()
if choice == 'h':
    img_path = input("Enter cover image path: ")
    msg = input("Enter secret message: ")
    key = input("Create encryption key: ")
    hide_message(img_path, msg, key)
```

```python
        key_char = secret_key[i % key_len]#if key is hsorter then message then it makes it loop from start again of the key
        encrypted_data.append(ord(char) ^ ord(key_char))#ord(char): Converts the message character to its ASCII value.
                                                        #ord(key_char): Converts the corresponding key character to its ASCII value.

    # Add message length header
    data_length = len(encrypted_data)
    length_header = struct.pack('!I', data_length)

    # Embed data in image
    row, col, channel = 0, 0, 0
    img_row, img_col, _ = img.shape

    # Store length header , The first few pixels store the message length
    for byte in length_header:
        if row >= img_row or col >= img_col:
            print("Error: Image too small")
            return
        img[row, col, channel] = byte
        row += 1
        col += 1
        channel = (channel + 1) % 3

    # Store encrypted message
    for byte in encrypted_data:
        if row >= img_row or col >= img_col:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\moham\OneDrive\Desktop\stego project> python -u "c:\Users\moham\OneDrive\Desktop\stego project\project.py"
Hide(h) or reveal(r) message? (h/r): r
Enter secret image path: secret_image.png
Enter decryption key: encrypt
Hidden message: hiii my name is arslan
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\moham\OneDrive\Desktop\stego project> python -u "c:\Users\moham\OneDrive\Desktop\stego project\project.py"
Hide(h) or reveal(r) message? (h/r): h
Enter cover image path: pic.jpg
Enter secret message: hiii my name is arslan
Create encryption key: encrypt
Message hidden in secret_image.png
```

edunet foundation

Conclusion

- **Summary:**
  - Successfully hides encrypted messages in images using lightweight, secure methods.
  - Addresses the gap in traditional steganography tools.

GitHub Link:

https://github.com/MohammadArslanKana/Stego_project.git

THANK YOU

edunet foundation