RANDOMIZED LEAST SIGNIFICANT BIT(LSB) EMBEDDING IMAGE STEGANOGRAPHY WITH ADVANCED ENCRYPTION STANDARD(AES) ALGORITHM

Team Details:

TEAM- 1

1. G M S Padmini (20EG105415)
2. B Vaishnavi (20EG105406)
3. CH Shreya (20EG105407)

Project Supervisor:

Dr.K Madhuri

Associate Professor

Introduction

- The **"Randomized Least Significant Bit(LSB) Embedding Image Steganography with Advanced Encryption Standard (AES)"** project is a combination of Image Steganography and AES Encryption technique along with Randomization.
- The developing environment is **Python 3.11** and **Command prompt.**
- It allows confidential information to be securely hidden within digital images, ensuring privacy and confidentiality during data transmission.
- The applications span various domains, including secure communication, digital forensics, privacy protection.

Problem Statement

Conventional methods of data transmission often lack the desired level of privacy, making sensitive information vulnerable to interception. These methods of sending data can be intercepted or compromised, posing a risk to sensitive information and are subjected to various security threats, including brute force attacks, Chosen Plaintext attacks, Known Plaintext attacks.

The objective of the Project is to address these privacy and security concerns by combining Steganography and Encryption methods. The aim is to create a Seamless and Secure channel for transmitting confidential information without arousing suspicion while also ensuring data integrity. This fusion of techniques seeks to provide a better solution for protecting sensitive information during transmission.

Proposed Method

- The proposed method is **"Randomized Least Significant Bit (LSB) Embedded Steganography with Advanced Encryption Standard(AES) Algorithm".**
-

INPUTS: Image
Message

Password.

Here, user has to decide the options 1 or 2 for :

1. Encrypt and Encode the Text
2. Decode and Decrypt the Text

Proposed Method

For the option 1: **Encrypt and Encode the Text**

- Firstly, check if the image is in **PNG Format.** If not we cannot use other Formats.
- We will check if the image is **RGB mode**, if not we need to convert it.
- The provided password must be of at least **12 characters.**
- With the given password, we will do Password Key Derivation where we use SHA-256 to get 256 bit key hash value which is used as encryption key for AES Encryption.
- We will also ask to give the message and we should do **Padding.**
- The padded plaintext is **Shuffled.**

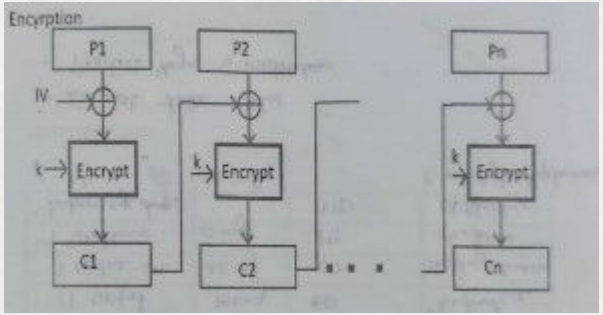
SHUFFLING THE BITS: (In this case we are using this method, we can use any other way of shuffling also)

The last 4 bits of the data are rearranged to occupy the first 4 positions, while simultaneously, the first 4 bits are moved to the last 4 positions.

After Shuffling the bits we will perform AES - 256 ENCRYPTION.

AES ENCRYPTION:

1.AES algorithm is used in Cipher Block Chaining (CBC) mode which takes Derived Key and Initialization Vector (IV) as inputs.
2. An initialization vector (IV) is an arbitrary number that can be used with a secret key for data encryption.
3. Initialization vector, XOR operation adds extra layer of security to use CBC mode.
4. Derived Key is 256 bit length.
5. AES works on 128 bit length fixed length block so the message is divided into 128 bit length blocks.



CBC MODE

The final cipher text is obtained is in the form of bytes where we convert the message to individual bits by keeping them in the form of a list.

APPENDING TO IV:

The resulting ciphertext is appended to IV.

BASE 64 ENCODING:

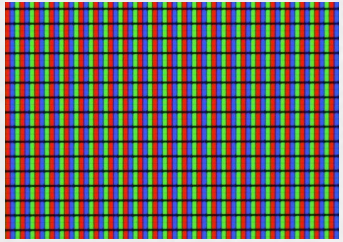
Then the ciphertext is finally encoded with BASE-64 encoding where 6 binary bits are grouped together and among the predefined 64 characters: **A-Z, a-z, +, /, 0-9.**

The Base 64 Encoding is done.

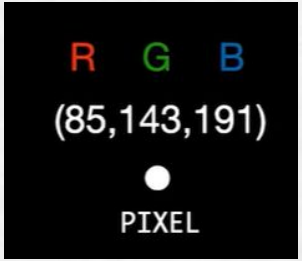
HEADER TEXT INSERTION:

- Header Text is encoded along with the cipher text.
- Header Text is is a predefined string that is used as a marker or identifier at the beginning of the hidden message.
- The length of the Header Text is 10 characters or bytes - "**HeaderText**" **.** It's purpose is to distinguish between the header (marker) and the actual encrypted message within the image.
- The header is shared among the Sender and Receiver.
- When decoding an image, the code checks whether the header text is present at the beginning of the decoded data. If it is not found or does not match the expected header text, the code considers the data invalid. This helps to ensure the integrity of the encoded message.

LSB ENCODING:

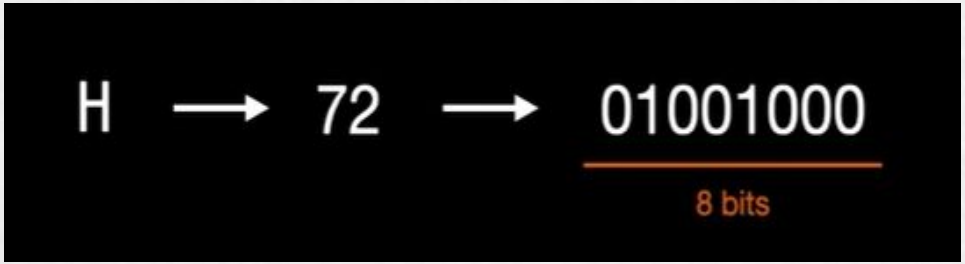


The image is made up of pixels

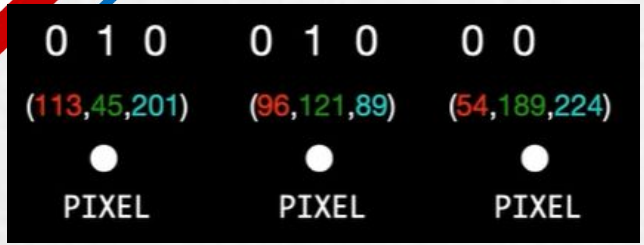


Each pixel has 3 colours - Red, Green and Blue

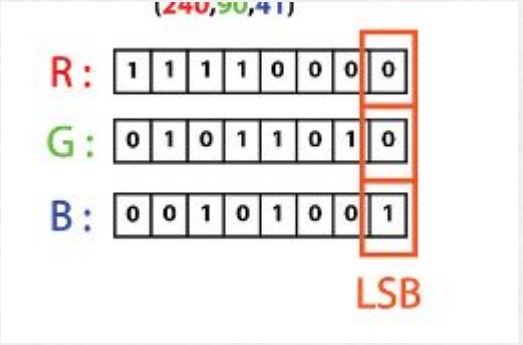
For example, in the steganography process, if the message bit to encode is “H” then the procedure is as follows:



Each character is converted to the corresponding ASCII value and converted to 8 bits.



Each character needs 3 pixels to encode it



LSB in each colour of the pixel

1	0	0	1	0	1	0	1

→ Message bits

1	0	1	1	0	1	1	0	1

→ 3 Pixels RGB Colours LSB

- If the message bit is '0' then value of LSB of the colour of pixel must be '0'
- If the message bit is '1' then value of LSB of the colour of pixel must be '1'
- 9th bit is called as **"STATUS BIT".** (3rd pixel's blue LSB)
- 9th bit should be '0' if the message continues or make it as '1' if the message ends with that character.

For the option 2: **Decode and Decrypt the Text**

LSB DECODING :

The code iterates through the pixel data of the encoded image. For each pixel, it reads the least significant bits (LSB) of the RGB values.

HEADER TEXT VALIDATION:

Header Text is validated from the decoded text. If the decoded text is having the headertext as "HeaderText". Then we continue to do further steps.

BASE-64 DECODING :

Then the ciphertext is finally decoded using BASE-64 encoding.

REMOVE THE IV:

The IV is removed from the ciphertext that is decoded.

DECRYPTING BY AES-256 DECRYPTION:

1. This process takes the same key used for encryption.
2. It extracts the IV from the beginning of the data and uses it to initialize.
3. The decrypted data is then unpadded to remove the padding bits which are added during encryption.
4. The bits are converted into bytes which in turn is converted to ASCII Text and the corresponding original message is obtained.

UNSHUFFLING THE BITS:

The last 4 bits of the data are rearranged to occupy the first 4 positions, while simultaneously, the first 4 bits are moved to the last 4 positions.

UNPAD THE TEXT:

We will Unpad the bits and finally display the Final Text.

This is the Proposed Method which includes Encrypting and Encoding the Text and also Decoding and Decrypting the Text.

Experiment Environment

The environment used is **PYTHON IDLE & COMMAND PROMPT**

The packages used in our Python code is:

- **PILLOW PACKAGE:** Used for image processing, to open, manipulate and save various image formats.
- **OS PACKAGE:** To interact with the operating system.
- **PYCRYPTODOME PACKAGE:** Used in the code for AES, SHA-256, RANDOM
- **BASE-64 PACKAGE:** Used for particularly in image steganography without causing issues when we use character encoding or special characters.
- **TERMCOLOR PACKAGE:** Used to add the color text, various background colors to output console. Here, we use cprint() function which is used to print the text with different styles and colours.

- **PYFIGLET PACKAGE:** To create ASCII text and attractive headers using font and width
- **GETPASS MODULE:** To read passwords and ensure the passwords and sensitive data is not exposed.
- **SYS PACKAGE:** To work with Python runtime Environment
- **RANDOM PACKAGE:** For the generation of IV
- **TIME MODULE:** To calculate the encryption and decryption time
- **BINASCII MODULE:** To work with binary and ASCII encoded data
- **NUMPY PACKAGE:** To work with arrays or matrices
- **SCIKIT-IMAGE PACKAGE:** For calculating image quality metrics, specifically PSNR (Peak Signal-to-Noise Ratio) and MSE (Mean Squared Error).
- **OPENCV PACKAGE**: For computer vision tasks and image processing.
- **MATPLOTLIB.PYTPLOT**: To create plots and visualizations

Experiment Screenshots

[1]

[2]

```
Details about the Image:

Enter the image path with the extension:
>>hello.png

Original Image:

The Image u have selected to Encode the Message is:

Enter 'OPEN' to display the image:
open
```

[3]

[4]

```
The image is in PNG Format

Width of the image : 6016
Height of the image : 3384

Total number of pixels in the image:  20358144

Details about the message:

Message to be hidden is:
>>Anurag University
```

[5]

```
Details about the plaintext:

Plaintext Bits:

[
    0, 1, 0, 0, 0, 0, 0, 1,
    0, 1, 1, 0, 1, 1, 1, 0,
    0, 1, 1, 1, 0, 1, 0, 1,
    0, 1, 1, 1, 0, 0, 1, 0,
    0, 1, 1, 0, 0, 0, 0, 1,
    0, 1, 1, 0, 0, 1, 1, 1,
    0, 0, 1, 0, 0, 0, 0, 0,
    0, 1, 0, 1, 0, 1, 0, 1,
    0, 1, 1, 0, 1, 1, 1, 0,
    0, 1, 1, 0, 1, 0, 0, 1,
    0, 1, 1, 1, 0, 1, 1, 0,
    0, 1, 1, 0, 0, 1, 0, 1,
    0, 1, 1, 1, 0, 0, 1, 0,
    0, 1, 1, 1, 0, 0, 1, 1,
    0, 1, 1, 0, 1, 0, 0, 1,
    0, 1, 1, 1, 0, 1, 0, 0,
    0, 1, 1, 1, 1, 0, 0, 1,
]

Length of Plaintext in Bytes: 17
Length of Plaintext in Bits: 136
```

[6]

```
Details about the password:

Enter your password (at least 12 characters):
Password:

Re-confirm the password:
Password:
```

[7]

```
Details about the Message, Key and IV:

The  Plaintext after padding is: b'Anurag University\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f\x0f'

Length of the Plaintext after padding in Bytes:32
Length of the Plaintext after padding in Bits:256
```

[8]

```
Key is:b'\xa8\xf1\x96\x9c\x12\x8c\xbcV\x8cD}\r\xd0~\xde\xae\xd8\xd1\xc9[$\xc1\xefgM\xa3\xadW\xe2U\x8dN'
Length of Key in bytes:32
Length of Key in bits: 256
```

[9]

```
IV: b'\x17\x1f\xa5\xfa\x8a\xbf\xbf\xedT\x1c\xb7:WN\xc5\x90'
Length of IV in bytes: 16
Length of IV in bits: 128
```

[10]

```
Details about Shuffling the Bits of the Padded Plaintext:

Original Bits:

[
    0, 1, 0, 0, 0, 0, 0, 1,
    0, 1, 1, 0, 1, 1, 1, 0,
    0, 1, 1, 1, 0, 1, 0, 1,
    0, 1, 1, 1, 0, 0, 1, 0,
    0, 1, 1, 0, 0, 0, 0, 1,
    0, 1, 1, 0, 0, 1, 1, 1,
    0, 0, 1, 0, 0, 0, 0, 0,
    0, 1, 0, 1, 0, 1, 0, 1,
    0, 1, 1, 0, 1, 1, 1, 0,
    0, 1, 1, 0, 1, 0, 0, 1,
    0, 1, 1, 1, 0, 1, 1, 0,
    0, 1, 1, 0, 0, 1, 0, 1,
    0, 1, 1, 1, 0, 0, 1, 0,
    0, 1, 1, 1, 0, 0, 1, 1,
    0, 1, 1, 0, 1, 0, 0, 1,
    0, 1, 1, 1, 0, 1, 0, 0,
    0, 1, 1, 1, 1, 0, 0, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
]
```

[11]

```
Shuffled Bits:
[
  0, 0, 0, 1, 0, 1, 0, 0,
  1, 1, 1, 0, 0, 1, 1, 0,
  0, 1, 0, 1, 0, 1, 1, 1,
  0, 0, 1, 0, 0, 1, 1, 1,
  0, 0, 0, 1, 0, 1, 1, 0,
  0, 1, 1, 1, 0, 1, 1, 0,
  0, 0, 0, 0, 0, 0, 1, 0,
  0, 1, 0, 1, 0, 1, 0, 1,
  1, 1, 1, 0, 0, 1, 1, 0,
  1, 0, 0, 1, 0, 1, 1, 0,
  0, 1, 1, 0, 0, 1, 1, 1,
  0, 1, 0, 1, 0, 1, 1, 0,
  0, 0, 1, 0, 0, 1, 1, 1,
  0, 0, 1, 1, 0, 1, 1, 1,
  1, 0, 0, 1, 0, 1, 1, 0,
  0, 1, 0, 0, 0, 1, 1, 1,
  1, 0, 0, 1, 0, 1, 1, 1,
  1, 1, 1, 1, 0, 0, 0, 0,
  1, 1, 1, 1, 0, 0, 0, 0,
  1, 1, 1, 1, 0, 0, 0, 0,
  1, 1, 1, 1, 0, 0, 0, 0,
  1, 1, 1, 1, 0, 0, 0, 0,
  1, 1, 1, 1, 0, 0, 0, 0,
  1, 1, 1, 1, 0, 0, 0, 0,
  1, 1, 1, 1, 0, 0, 0, 0,
  1, 1, 1, 1, 0, 0, 0, 0,
  1, 1, 1, 1, 0, 0, 0, 0,
  1, 1, 1, 1, 0, 0, 0, 0,
  1, 1, 1, 1, 0, 0, 0, 0,
  1, 1, 1, 1, 0, 0, 0, 0,
]

After Shuffling the bits, the shuffled text is:b"\x14\xe6W'\x16v\x02U\xe6\x96gV'7\x96G\x97\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0"
```

[12]

[13]

```
Details about encrypting the shuffled text using AES-256 encryption :

The shuffled bits are encrypted and the ciphertext is: b'\xbafFYsdk\x9e\x95[\xd5\xb1v\x9b&6\x16\xc8/\x15\xf0\xacmd\x13\xf3\xdc\x8ds\xee~"'

Length of CipherText in bytes: 32
Length of CipherText in bits: 256
```

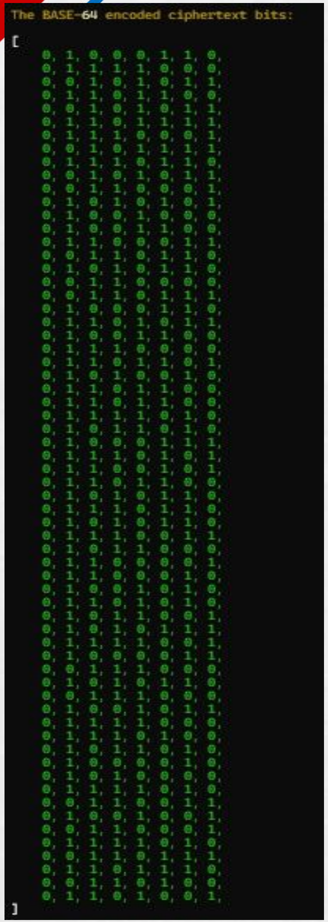
```
Cipher Text bits:

[
    1, 0, 1, 1, 1, 0, 1, 0,
    0, 1, 1, 0, 0, 1, 1, 0,
    0, 1, 0, 0, 0, 1, 1, 0,
    0, 1, 0, 1, 1, 0, 0, 1,
    0, 1, 1, 1, 0, 0, 1, 1,
    0, 1, 1, 0, 0, 1, 0, 0,
    0, 1, 1, 0, 1, 0, 1, 1,
    1, 0, 0, 1, 1, 1, 1, 0,
    1, 0, 0, 1, 0, 1, 0, 1,
    0, 1, 0, 1, 1, 0, 1, 1,
    1, 1, 0, 1, 0, 1, 0, 1,
    1, 0, 1, 1, 0, 0, 0, 1,
    0, 1, 1, 1, 0, 1, 1, 0,
    1, 0, 0, 1, 1, 0, 1, 1,
    0, 0, 1, 0, 0, 1, 1, 0,
    0, 0, 1, 1, 0, 1, 1, 0,
    0, 0, 0, 1, 0, 1, 1, 0,
    1, 1, 0, 0, 1, 0, 0, 0,
    0, 0, 1, 0, 1, 1, 1, 1,
    0, 0, 0, 1, 0, 1, 0, 1,
    1, 1, 1, 1, 0, 0, 0, 0,
    1, 0, 1, 0, 1, 1, 0, 0,
    0, 1, 1, 0, 1, 1, 0, 1,
    0, 1, 1, 0, 0, 1, 0, 0,
    0, 0, 0, 1, 0, 0, 1, 1,
    1, 1, 1, 1, 0, 0, 1, 1,
    1, 1, 0, 1, 1, 1, 0, 0,
    1, 0, 0, 0, 1, 1, 0, 1,
    0, 1, 1, 1, 0, 0, 1, 1,
    1, 1, 1, 0, 1, 1, 1, 0,
    0, 1, 1, 1, 1, 1, 1, 0,
    0, 0, 1, 0, 0, 0, 1, 0,
]
```

[14]

```
The above shuffled CipherText is appended to the IV.

 The resultant Cipher Text after it is appended to IV is :
b'\x17\x1f\xa5\xfa\x8a\xbf\xbf\xedT\x1c\xb7:WN\xc5\x90' + b'\xbafFYsdk\x9e\x95[\xd5\xb1v\x9b&6\x16\xc8/\x15\xf0\xacmd\x13\xf3\xdc\x8ds\xee~"'

b'\x17\x1f\xa5\xfa\x8a\xbf\xbf\xedT\x1c\xb7:WN\xc5\x90\xbafFYsdk\x9e\x95[\xd5\xb1v\x9b&6\x16\xc8/\x15\xf0\xacmd\x13\xf3\xdc\x8ds\xee~"'

The obtained ciphertext is encoded using BASE-64 encoding:

Fx+l+oq/v+1UHLc6V07FkLpmRllzZGuelVvVsXabJjYWyC8V8KxtZBPz3I1z7n4i
```

[15]

[16]




```
Length of the BASE-64 encoded text in bits : 512

Encryption Time: 0.30148 seconds


Image Mode: RGB
```

[17]

```
Details about the header text and header bits:

This header text is combined with the base64 encoded ciphertext and is encoded

Header Text is : HeaderText

Length of the Header Text in Bytes is : 10

Length of the header bits in Bits is: 80
```

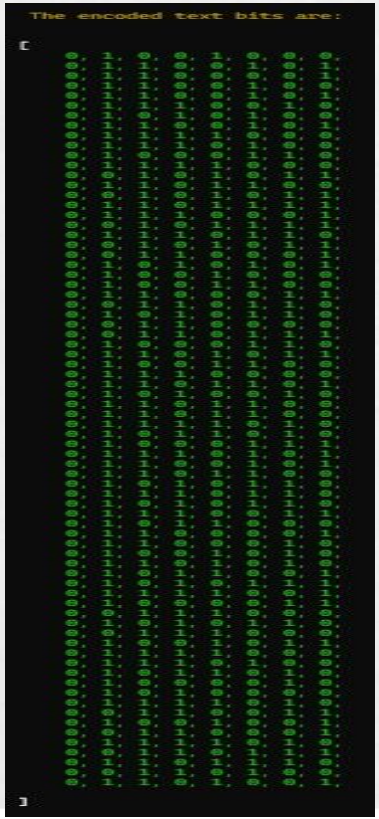
[18]

```
Header bits:

[
    0, 1, 0, 0, 1, 0, 0, 0,
    0, 1, 1, 0, 0, 1, 0, 1,
    0, 1, 1, 0, 0, 0, 0, 1,
    0, 1, 1, 0, 0, 1, 0, 0,
    0, 1, 1, 0, 0, 1, 0, 1,
    0, 1, 1, 1, 0, 0, 1, 0,
    0, 1, 0, 1, 0, 1, 0, 0,
    0, 1, 1, 0, 0, 1, 0, 1,
    0, 1, 1, 1, 1, 0, 0, 0,
    0, 1, 1, 1, 0, 1, 0, 0,
]
```

[19]

[20]

```
The header text that is used to combine with the ciphertext before encoding the total message :
 HeaderText

The text that is encoded into the image is:
HeaderTextFx+l+oq/v+1UHLc6V07FkLpmRllzZGuelVvVsXabJjYWyC8V8KxtZBPz3I1z7n4i
```



[21]

```
The length of the total encoded text in Bytes is : 74

The length of the total encoded text in Bits is : 592

Number of status bits that are used here are : 74
" Encoding image......
```

[22]

```
Details while Encoding the message to the image:
Original File: hello.png
Encoding image......Image encoded and saved as hello-encoded.png
Encoding image......
Comparison between Original and Encoded Image:

Total Pixels in the Image: 20358144

Number of Pixels Changed: 194

Percentage Change: 0.0009529356%
Encoding image......Image encoded and saved as hello-encoded.png
The Encoded Image is:
```

[23]

[24]

```
Choose one from below:

1. Encrypt and Encode the Text
2. Decode and Decrypt the Text

>>2

Enter the image path with the extension:
>>hello-encoded.png

The image is in PNG Format
```

[25]

```
Details about the password:

Enter password that is entered at encode option:

Password:

The Password is Correct ! ! !

The image that needs to be decoded is:
```

[26]

```
Enter 'OPEN' to display the image:
open
```

[27]

[28]

```
Details about decoding of the image:

The Decoded Text from the image is:
HeaderTextFx+l+oq/v+1UHLc6V07FkLpmRllzZGuelVvVsXabJjYWyC8V8KxtZBPz3I1z7n4i

The Length of the Encoded Text in the image in Bytes: 74

Decoding is successfully completed.
```

[29]

```
Validating the header ! ! !

The header decoded from the image is: HeaderText

 Header is validated......You can proceed to Decrypt the data.

The Decoded Text (Not including HeaderText) from the Image is :
Fx+l+oq/v+1UHLc6V07FkLpmRllzZGuelVvVsXabJjYWyC8V8KxtZBPz3I1z7n4i
```

[30]

```
Details about decrypting the message:

The text is in BASE-64 Encoding format. So after Decoding it, The Text is:
b'\x17\x1f\xa5\xfa\x8a\xbf\xbf\xedT\x1c\xb7:WN\xc5\x90\xbafFYsdk\x9e\x95[\xd5\xb1v\x9b&6\x16\xc8/\x15\xf0\xacmd\x13\xf3\xdc\x8ds\xee~"'

Key is b'\xa8\xf1\x96\x9c\x12\x8c\xbcV\x8cD}\r\xd0~\xde\xae\xd8\xd1\xc9[$\xc1\xefgM\xa3\xadW\xe2U\x8dN'

IV is b'\x17\x1f\xa5\xfa\x8a\xbf\xbf\xedT\x1c\xb7:WN\xc5\x90'

The Decrypted text is : b"\x14\xe6W'\x16v\x02U\xe6\x96gV'7\x96G\x97\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0\xf0"
```

[31]

```
Details about De-Shuffling the bits:

The bits that need to be deshuffled are :
[
    0, 0, 0, 1, 0, 1, 0, 0,
    1, 1, 1, 0, 0, 1, 1, 0,
    0, 1, 0, 1, 0, 1, 1, 1,
    0, 0, 1, 0, 0, 1, 1, 1,
    0, 0, 0, 1, 0, 1, 1, 0,
    0, 1, 1, 1, 0, 1, 1, 0,
    0, 0, 0, 0, 0, 0, 1, 0,
    0, 1, 0, 1, 0, 1, 0, 1,
    1, 1, 1, 0, 0, 1, 1, 0,
    1, 0, 0, 1, 0, 1, 1, 0,
    0, 1, 1, 0, 0, 1, 1, 1,
    0, 1, 0, 1, 0, 1, 1, 0,
    0, 0, 1, 0, 0, 1, 1, 1,
    0, 0, 1, 1, 0, 1, 1, 1,
    1, 0, 0, 1, 0, 1, 1, 0,
    0, 1, 0, 0, 0, 1, 1, 1,
    1, 0, 0, 1, 0, 1, 1, 1,
    1, 1, 1, 1, 0, 0, 0, 0,
    1, 1, 1, 1, 0, 0, 0, 0,
    1, 1, 1, 1, 0, 0, 0, 0,
    1, 1, 1, 1, 0, 0, 0, 0,
    1, 1, 1, 1, 0, 0, 0, 0,
    1, 1, 1, 1, 0, 0, 0, 0,
    1, 1, 1, 1, 0, 0, 0, 0,
    1, 1, 1, 1, 0, 0, 0, 0,
    1, 1, 1, 1, 0, 0, 0, 0,
    1, 1, 1, 1, 0, 0, 0, 0,
    1, 1, 1, 1, 0, 0, 0, 0,
    1, 1, 1, 1, 0, 0, 0, 0,
]
```

[32]

```
Original Bits:
[
    0, 1, 0, 0, 0, 0, 0, 1,
    0, 1, 1, 0, 1, 1, 1, 0,
    0, 1, 1, 1, 0, 1, 0, 1,
    0, 1, 1, 1, 0, 0, 1, 0,
    0, 1, 1, 0, 0, 0, 0, 1,
    0, 1, 1, 0, 0, 1, 1, 1,
    0, 0, 1, 0, 0, 0, 0, 0,
    0, 1, 0, 1, 0, 1, 0, 1,
    0, 1, 1, 0, 1, 1, 1, 0,
    0, 1, 1, 0, 1, 0, 0, 1,
    0, 1, 1, 1, 0, 1, 1, 0,
    0, 1, 1, 0, 0, 1, 0, 1,
    0, 1, 1, 1, 0, 0, 1, 0,
    0, 1, 1, 1, 0, 0, 1, 1,
    0, 1, 1, 0, 1, 0, 0, 1,
    0, 1, 1, 1, 0, 1, 0, 0,
    0, 1, 1, 1, 1, 0, 0, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
    0, 0, 0, 0, 1, 1, 1, 1,
]
```

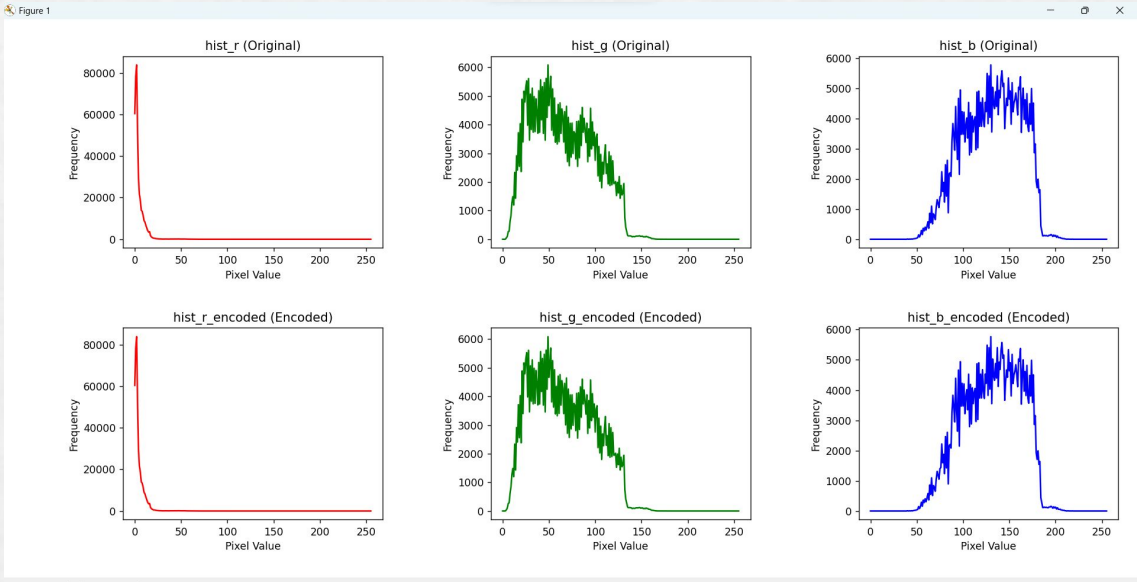
[33]

```
Decryption Time: 0.04742 seconds

The Decoded Text should be Decrypted....

After Decryption, the Decrypted Text:  b'Anurag University'

Decryption successful.....! ! !


The Final Text that receiver received is :
Anurag University
```

[34]

Experiment Results

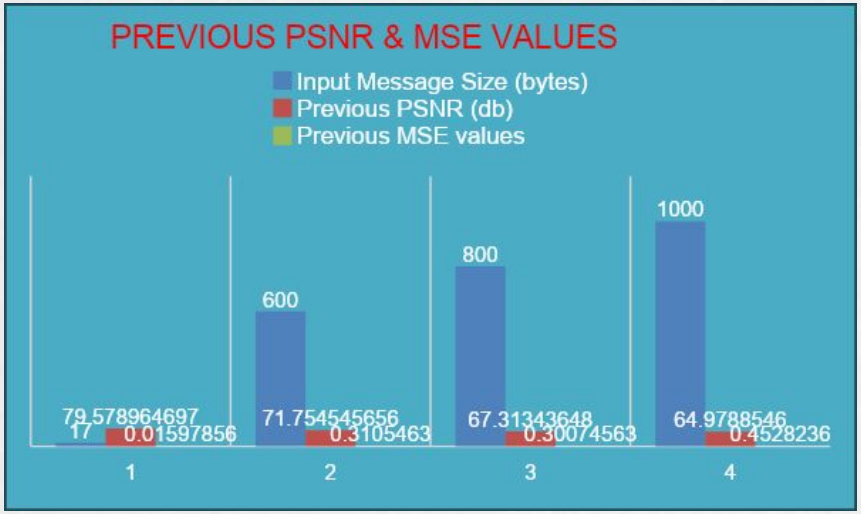
```
Original Image Size: 195.45 KB
Encoded Image Size: 195.45 KB
Text Size: 17 bytes
PSNR: 84.07974928 dB
MSE: 0.00025416
```

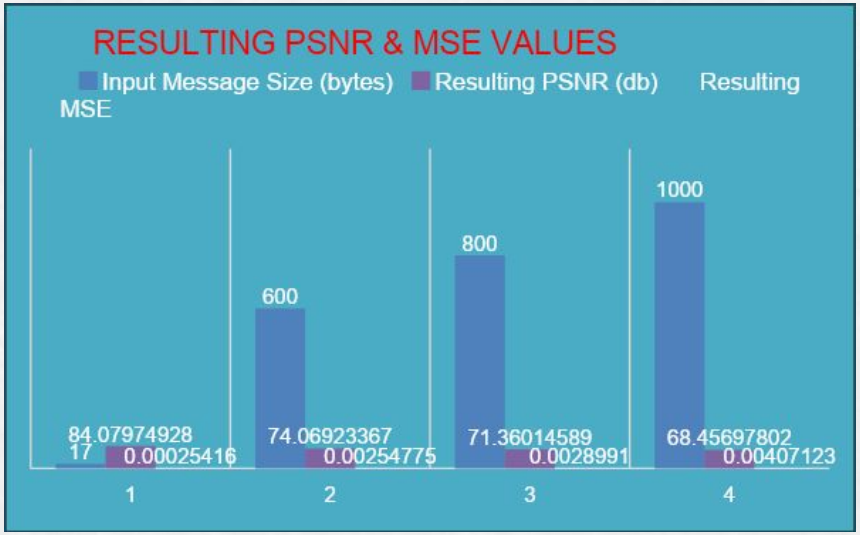


Experiment Results

IMAGE SIZE	Input Message Size (bytes)	Previous PSNR (db)	Previous MSE values	Resulting PSNR (db)	Resulting MSE
195.45 KB	17	79.578964697	0.01597856	84.07974928	0.00025416
195.45 KB	600	71.754545656	0.31054630	74.06923367	0.00254775
195.45 KB	800	67.313436 48	0.30074563	71.36014589	0.00289910
195.45 KB	1000	64.978854 60	0.45282360	68.45697802	0.00407123

Experiment Results





Finding

- We found that “Randomized Image Steganography with AES Algorithm” is **secure** against Chosen-Plaintext attack, Differential Cryptanalysis, Known- Plaintext attacks.

Example: If the message that is to be encoded without shuffling is “HELLO” then The ASCII values of the characters in "HELLO" are [72, 69, 76, 76, 79]. In the original image, you replace the least significant bit (LSB) of each color channel (R, G, B) with the corresponding bit of the ASCII values. But it is **not possible** to succeed these attacks as we are doing “Randomization through Shuffling”, “BASE-64 Encoding”, etc.

- Here, we are using Randomization through Shuffling the bits.

- Through these AES-256 Encryption and Decryption functions we are able to achieve Privacy

- We can use other modes of image to convert into RGB such as CMYK, GREYSCALE, RGBA.

Justification

Parameters	Formula	Previous value	Result value	Justification
RGB Mode of the Image	———	We are only using RGB mode of the images in Image Steganography.	In this project, we are also able to use **GREYSCALE, CMYK,RGBA** Modes of Images also	We are using a function which converts the other forms of Image to RGB Mode of the Image
AES-256	Using 256 bit Key	Using only AES-128 or DES	Usage of AES-256	Increases complexity by longer key size, as there are more number of round keys. AES-256 has 14 rounds

BASE-64 Encoding	Mapping 6 bits to the corresponding BASE 64 characters	Not used	The result is in right format.	Does not promote Security but it is the right format to use while encoding when the errors arise during Special Characters or the Encoded Characters
Randomization using Shuffling the bits	First 4 bits are shifted to the places of last 4 bits and vice versa	Not used	Confusion and Diffusion	Known-Plaintext, Chosen-Plaintext attacks are completely resisted.

METRICS TO MEASURE THE IMAGE QUALITY:

PSNR (Peak Signal-to-Noise Ratio):

PSNR = 10 * log10((peak^2) / MSE) FORMULA

- PSNR is a measure of the quality of a encoded image compared to an original reference image.
- PSNR values above 30 dB are generally considered high and indicative of good image quality.

MSE (Mean Squared Error):

MSE = (1 / (m * n)) * Σ(Σ(I(x, y) - K(x, y))^2) FORMULA

- **m** and **n** are the dimensions of the image (e.g., width and height).
- **I(x, y)** represents the pixel value of the original image at position (x, y).
- **K(x, y)** represents the pixel value of the reconstructed (or compressed) image at the same position.
- MSE measures the average squared difference between the pixel values of the original image and encoded image.
- MSE values close to zero suggest minimal distortion and high image fidelity.

COMPARISION OF THE METRICS:

IMAGE SIZE	Input Message Size (bytes)	Previous PSNR (db)	Previous MSE values	Resulting PSNR (db)	Resulting MSE
340.74 KB	17	90.56478977	0.0004690	101.53335594	0.00004570
340.74 KB	600	88.754545656	0.00054630	91.79645076	0.00004300
340.74 KB	800	84.31343648	0.00074563	89.68511923	0.00006991
340.74 KB	1000	82.97885460	0.00082360	85.45697802	0.00007123