Data Encryption using Combination of RSA Cryptography and PLS based Steganography Techniques

Mrs. T Madhavi Kumari [1], K Pranavendra[2]

[1](ECE, JNTUH College of Engineering Hyderabad, India)
[2](ECE, JNTUH College of Engineering Hyderabad, India)

---***---

Abstract - *In today's rapidly advancing technological era, the internet has become widely used, leading to a massive amount of digital data being shared. As a result, information security has become a top priority for researchers and professionals. Data security is essential because people frequently need to store, send, and receive private information, which must be protected from unauthorized access and attacks. To address this, steganography is used, which involves hiding data within other files or media to ensure information security. Another technique that enhances security is cryptography, which uses mathematical algorithms to transform data into an unreadable format. By combining steganography and cryptography, individuals and organizations can strengthen the security of their sensitive information. A robust and efficient technique based on RSA algorithm and LSB Steganography using PLS (Pixel Locator Sequence) has been proposed to provide enhanced security. This technique uses RSA and Vigenere cipher algorithms to encrypt the plain text followed by compressing the obtained cipher text using Huffman coding. This compressed text is then embedded in the cover image using LSB Steganography based on a pixel locator sequence. The pixel locator sequence is generated at the sender side and is encrypted using AES (Advanced Encryption Standard) algorithm using a key and is sent along with stego image to the receiver. Only the legitimate user who has access to this AES key will be able to decode the PLS data and can retrieve the actual data from the stego image.*

***Key Words*: Cryptography, Steganography, Huffman coding, Vigenere cipher, RSA, LSB, AES**

1. INTRODUCTION

Information security is a critical concern in today's interconnected world. The increasing reliance on digital communication necessitates novel methods to ensure the confidentiality and integrity of sensitive data. This project is motivated by the imperative to develop advanced techniques that address these security challenges.

Traditional cryptographic methods, while effective, are susceptible to various attacks. Additionally, as data communication has evolved, the need for covert data transmission has risen. This project aims to tackle these issues by integrating multiple cryptographic and steganographic techniques.

The primary objective is to design and implement a system that enhances data security through a multi-layered approach. Specific goals include the integration of RSA cryptography, Vigenere cipher, Huffman coding, and PLS-based LSB steganography.

The project methodology encompasses the following stages: data preparation, encryption using RSA and Enhanced Vigenere cipher, compression through Huffman coding, PLS sequence generation, embedding within images using LSB steganography based on PLS sequence, and integration of all techniques.

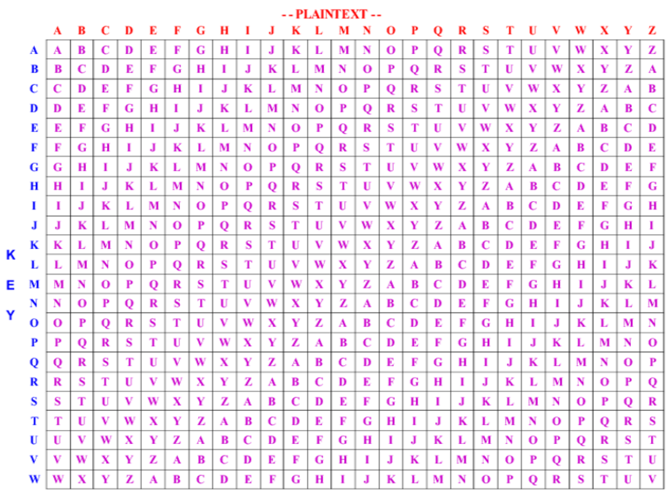
2. BACKGROUND

In this section, we elaborate on different key methods employed in our project. These methods include both cryptographic techniques, which involve securing data through encryption, and steganography techniques, which revolve around hiding data in plain sight .

Vigenere Cipher:

The Vigenere cipher, a polyalphabetic substitution cipher, was invented during the 16th century by Giovan Battista Bellaso.. Unlike simple substitution ciphers, the Vigenere cipher uses multiple substitution alphabets based on a keyword or passphrase.

The Vigenere cipher operates by shifting each character of the plaintext by a variable amount determined by the corresponding character in the keyword. The process involves using a Vigenere table, to determine the substitution alphabets. Each character of the keyword corresponds to a row in the table, and each character of the plaintext corresponds to a column. The intersection of the row and column gives the substitution character for encryption and decryption. The Vigenere cipher requires a keyword or passphrase to generate the encryption key. The key is generated by repeating the keyword to match the length of the plaintext.



- - PLAINTEXT - -

KEY

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V

Fig -1: Vigenere table

To encrypt the plaintext, each character is shifted by the corresponding character in the key using the Vigenere table. The result is the ciphertext, which is the encrypted form of the plaintext.

Decryption is the reverse process of encryption. Each character of the ciphertext is shifted back by the corresponding character in the key using the Vigenere table, resulting in the original plaintext.

Enhanced Vigenere Cipher:

The Enhanced Vigenere Cipher is an advanced version of the traditional Vigenere Cipher that expands its character set to include a broader range of characters. This expansion aims to enhance the versatility and security of the cipher, allowing for the encryption and decryption of alphanumeric, special, and extended ASCII characters. The Vigenere Cipher, developed in the 16th century, marked a significant milestone in the history of cryptography. Over time, the cipher has evolved and adapted to meet the changing needs of data encryption. The Enhanced Vigenere Cipher builds upon this historical lineage by extending the character set to cater to modern communication systems.

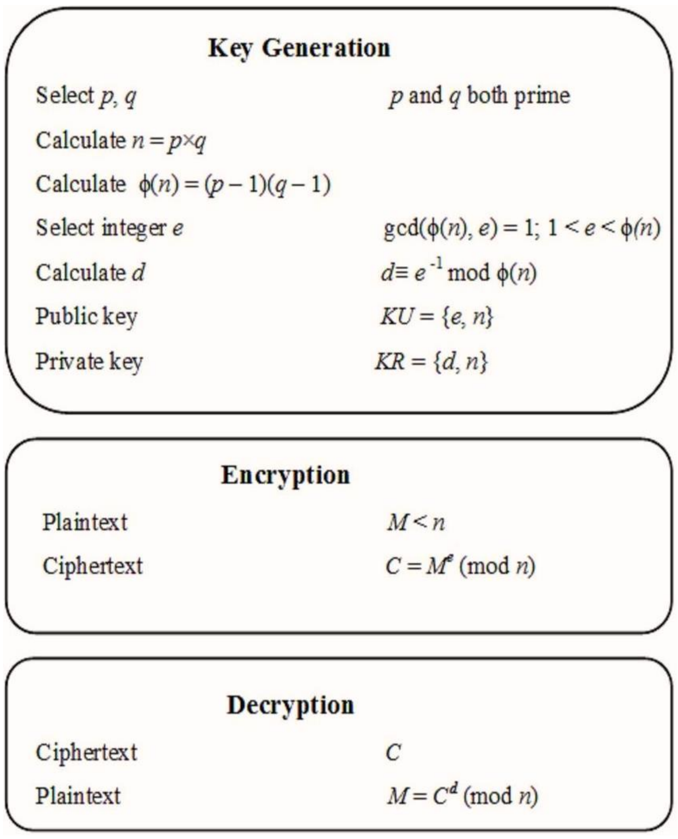
The process of key generation in the Enhanced Vigenere Cipher involves creating a random sequence of characters that serves as the encryption key. With the expanded character set, the key generation process can now include alphanumeric, special, and extended ASCII characters, increasing the complexity and strength of the encryption. In traditional Vigenere cipher the size of the Vigenere table is 26X26 whereas in enhanced vigenere cipher the table is expanded to 92X92.

In the encryption process, each character of the plaintext is encrypted by selecting the corresponding row and column in the expanded Vigenere table. The intersection of the selected row and column yields the encrypted character. This expanded Vigenere table encompasses alphanumeric, special, and extended ASCII characters, allowing for the encryption of a wider range of data types. The decryption process in the Enhanced Vigenere Cipher is the reverse of the encryption process. Each character of the ciphertext is mapped back to its original plaintext character by identifying the corresponding row and column in the expanded Vigenere table.

The inclusion of alphanumeric, special, and extended ASCII characters in the Vigenere table significantly expands the range of data that can be encrypted and decrypted using the cipher. This enhancement increases the versatility and applicability of the Enhanced Vigenere Cipher to various data types.The inclusion of special and extended ASCII characters adds an extra layer of security, making the cipher more resistant to attacks such as frequency analysis and known-plaintext attacks. The inclusion of a broader character set in the Enhanced Vigenere Cipher aligns it with modern communication systems and data storage methods.

RSA (Rivest-Shamir-Adleman) Algorithm:

The RSA algorithm, named after its inventors Rivest, Shamir, and Adleman, is a widely used public-key cryptography algorithm. The RSA algorithm relies on the mathematical properties of prime numbers, modular arithmetic, and exponentiation. It involves the generation of a public-private key pair and the use of modular exponentiation for encryption and decryption operations. It offers a robust method for secure communication by providing confidentiality, integrity, and authentication of data. Prime numbers have unique properties and are difficult to factorize, ensuring the security of RSA. The RSA key generation process involves selecting large prime numbers, calculating the modulus and totient values, and determining the public and private keys. Random number generation plays a crucial role in generating secure keys. The picture below illustrates the sequential process followed in the RSA algorithm.



Key Generation

Select p, q	p and q both prime
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \bmod \phi(n)$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

Encryption

Plaintext	$M < n$
Ciphertext	$C = M^e \pmod n$

Decryption

Ciphertext	C
Plaintext	$M = C^d \pmod n$

Fig -2: RSA Algorithm

Huffman Coding:

Huffman coding is a popular data compression algorithm that achieves efficient encoding of data by assigning shorter codes to frequently occurring symbols and longer codes to less frequent symbols. It was developed by David A. Huffman in 1952 and has since become widely used in various applications.

Huffman coding provides a variable-length encoding scheme, where symbols with higher frequencies are represented by shorter code words, while symbols with lower frequencies are represented by longer code words. This adaptability makes Huffman coding an efficient method for data compression.

Some fundamental concepts and terminology used in Huffman coding are described below:

Symbol: A symbol represents the basic unit of information to be encoded. It can be a character, a pixel value, or any discrete element. In text compression, symbols can be individual characters, whereas in image compression, they can be pixel intensities.

Code: A code is a unique binary sequence assigned to each symbol. Huffman coding ensures that no code is a prefix of another code, enabling unambiguous decoding. The code for a symbol is derived from the binary path in the Huffman tree leading to that symbol.

Prefix Code: A prefix code is a code in which no code word is a prefix of any other code word. This property enables efficient decoding without any lookahead. Huffman coding guarantees the prefix code property.

The Huffman coding algorithm follows a step-by-step process to generate an optimal prefix code:

i) Frequency Calculation:

Determine the frequency of each symbol in the input data. This can be achieved by scanning the data and constructing a frequency table or histogram.

ii) Building the Huffman Tree:

Create a binary tree, known as the Huffman tree or Huffman coding tree, based on the frequency of symbols. Each symbol becomes a leaf node in the tree, and the internal nodes represent the merging of symbols.

Start with a forest of individual nodes, one for each symbol, with the frequency as the key.

Repeatedly combine the two nodes with the lowest frequency into a new internal node. The frequency of the new node is the sum of the frequencies of its children.

Continue until all nodes are combined into a single tree, resulting in the root node.

iii)Assigning Codes:

Traverse the Huffman tree to assign unique binary codes to each symbol.

During the tree traversal, assign a "0" bit for each left branch and a "1" bit for each right branch.

The code for a symbol is the binary sequence obtained by concatenating the branch choices from the root to that symbol.

iv)Encoding:

Replace each symbol in the input data with its corresponding Huffman code.

By mapping each symbol to its assigned code, the input data is transformed into a sequence of variable-length binary codes.

The encoded data is typically represented as a series of concatenated bits.

v)Decoding:

Decode the encoded data using the Huffman tree.

Start from the root of the Huffman tree.

Read the encoded data bit by bit and follow the corresponding branches until a leaf node is reached.

Output the symbol associated with that leaf node.

Repeat until all encoded data has been decoded, reconstructing the original data.

LSB Steganography:

Steganography is the art and science of concealing information within other data to achieve covert communication. It involves hiding the existence of a message, making it difficult for unauthorized individuals to detect. LSB steganography is a common technique used to hide data within digital images.

LSB steganography works by replacing the least significant bit of selected pixels in the image with bits from the secret message. The least significant bit is the rightmost bit in a binary representation of a pixel's color value. By altering only the least significant bit, the changes made to the image are minimal and often imperceptible to the human eye. LSB steganography provides a balance between imperceptibility and security.

The steps followed in LSB steganography are described below:

i)Image Selection

In LSB steganography, a cover image is chosen as the carrier for the secret message. The cover image should have high complexity and sufficient visual information to effectively hide the embedded data. Ideally, the cover image should resemble a typical image in its content, structure, and size.

ii)Message Encoding

Before embedding, the secret message needs to be converted into a binary format suitable for embedding. This may involve encoding text into ASCII or converting files into binary representation. Various encoding techniques can be used depending on the type of data being concealed.

iii)Pixel Modification

The next step is to modify the least significant bit of selected pixels in the cover image. The selection of pixels and the embedding strategy depend on the specific implementation. Care must be taken to ensure that the modifications do not introduce noticeable visual artifacts or distortions in the cover image.

iv)Stego Image Generation

By modifying the selected pixels, a stego image is generated. The stego image contains both the cover image and the hidden message. From the outside, the stego image appears visually similar to the original cover image, making it difficult to detect the embedded data.

v)Message Extraction

To extract the hidden message, the stego image is processed to extract the modified least significant bits of the selected pixels. These bits are then combined to reconstruct the original message. The extraction process should reverse the embedding process to ensure accurate message recovery.

PLS based LSB Steganography:

PLS (Pixel Locator Sequence) steganography is an extension of LSB steganography that aims to enhance security and imperceptibility by spreading modifications across the image. It achieves this by modifying pixels in a non-sequential order, making it more challenging for unauthorized individuals to detect the presence of the embedded information.

The pixel locator sequence is designed to ensure an even distribution of modifications throughout the image, enhancing both the security and imperceptibility of the steganographic technique. The modifications are typically made by replacing the least significant bit of selected pixels with bits from the secret message, as in LSB steganography. However, the non-sequential order of pixel modification in PLS steganography provides an additional layer of security against detection and analysis.

By leveraging the concept of a shuffled pixel locator sequence, PLS steganography presents an advanced approach to concealing information within digital images, making it a valuable technique for covert communication and data protection.

The steps followed in PLS-LSB steganography are described below:

i) PLS (Pixel Locator Sequence) Generation

The generation of the pixel locator sequence is a critical step in PLS steganography. It involves selecting pixels from the cover image and establishing a unique order for these pixels. The order is designed to achieve randomness and an even distribution of modifications across the image.

ii)Message Encoding

Similar to LSB steganography, the secret message needs to be encoded into a binary format suitable for embedding using PLS steganography. The encoding process prepares the message for modification and embedding into the cover image.

iiii)Pixel Modification

Using the pixel locator sequence, the selected pixels in the cover image are modified based on the corresponding bits of the secret message. The modifications are distributed across the image in a non-sequential order, enhancing the security and imperceptibility of the steganographic technique.

iv)Stego Image Generation

The modified pixels are combined with the original cover image to create the stego image. This image appears visually similar to the cover image, and the embedded message remains hidden within it.

v)Message Extraction

To extract the hidden message, the same pixel locator sequence used during embedding is applied. By reversing the pixel modification process, the modified bits can be extracted and combined to reconstruct the original message.

PLS steganography spreads modifications across the image in a non-sequential order, making them less predictable and harder to detect. By distributing the modifications across the image, PLS steganography enhances imperceptibility, making it more challenging for unauthorized individuals to detect the embedded data.

Fisher- Yates Algorithm:

The Fisher-Yates algorithm is a standard shuffling technique.This algorithm holds a vital place in modern cryptography and data manipulation. Named after its inventors Ronald A. Fisher and Frank Yates, it's widely recognized for its ability to generate random permutations of a given sequence. While its original application was in shuffling decks of cards, its versatility extends to various fields, including steganography.

The steps followed in Fisher-Yates algorithm to shuffle an array on n elements are described below:

i) Start at the last element in the array (index= n-1). Here n is the length of the array.

ii) Generate a random integer between 0 and n-1 (inclusive)

iii) Swap the element at the current index with the element at the randomly selected index.

iv) Decrement the current index by 1.

v) Repeat steps 2-4 until the current index is less than or equal to 0.

3. PROPOSED METHOD

This section presents the methodology used in the project for encrypting data, compressing it using Huffman coding, and embedding the compressed text using PLS-LSB steganography. The methodology involves a series of steps that ensure the security and integrity of the data throughout the process. Each stage of the methodology is explained in detail to provide a comprehensive understanding of the project's approach.

A . Data Encryption

Enhanced Vigenere Cipher Encryption: The encryption process begins with applying the Enhanced Vigenere cipher to the plaintext using a secret keyword. This cipher shifts each character based on the corresponding letter of the keyword, enhancing the security of the data.

RSA Encryption: After Vigenere cipher encryption, the RSA algorithm is applied for an additional layer of encryption. RSA encryption involves generating key pairs consisting of a public key and a private key. The public key is used to encrypt the data, while the private key is kept secret and used for decryption. The encrypted data is further secured using the recipient's public key parameters.

B. Data Compression using Huffman Coding

Frequency Analysis: Once the data is encrypted, a frequency analysis is performed on the encrypted text. This analysis determines the frequency of occurrence of each symbol in the text.

Building Huffman Tree: Based on the symbol frequencies obtained from the frequency analysis, a Huffman tree is constructed. The Huffman tree is a binary tree structure that assigns shorter codes to more frequently occurring symbols and longer codes to less frequently occurring symbols.

Generating Huffman Codes: Using the constructed Huffman tree, Huffman codes are generated for each symbol in the encrypted text. These Huffman codes represent a compressed form of the data, where more frequently occurring symbols are represented by shorter codes, leading to overall data compression.

C. Data Embedding using PLS-LSB Steganography

Pixel Locator Sequence (PLS) Generation:

Let N be the total pixel count in the image and let N_{enc} be the length of encoded message. The number of pixels N_p required to encode the given message can be simple calculated using the formula:

$N_p = 3 \times N_{enc}$

Each of the 3 pixels grouped will contain the information of one character present in the encrypted text. The random distribution of pixels in PLS can be done using the Fisher-Yates Shuffle algorithm. This randomized order adds an additional layer of security to the steganography process.

The algorithm used in PLS generation is described below :

Algorithm 1: Generating PLS sequence

```
Result: Distinct $N_p$ numbers generated in range 0 to
        N-1 stored in PLS array
n = N, m = $N_p$, i = 0;
while $i \leq n\text{-}1$ do
    arr[i] = i;
    i++;
end
i = 0;
while $i \leq m\text{-}1$ do
    swap(arr[n-i], arr[rand()%(n-i+1)]);
    i++;
end
PLS [], i=n-i;
while $i \geq n\text{-}m$ do
    PLS[n-1-i] = arr[i];
end
```

AES Encryption of Pixel Locator Sequence: To enhance the security of the embedding process, the Pixel Locator Sequence (PLS) is encrypted using the AES (Advanced Encryption Standard) algorithm. AES encryption ensures the confidentiality of the PLS, making it more challenging for unauthorized entities to detect or tamper with the embedded data.

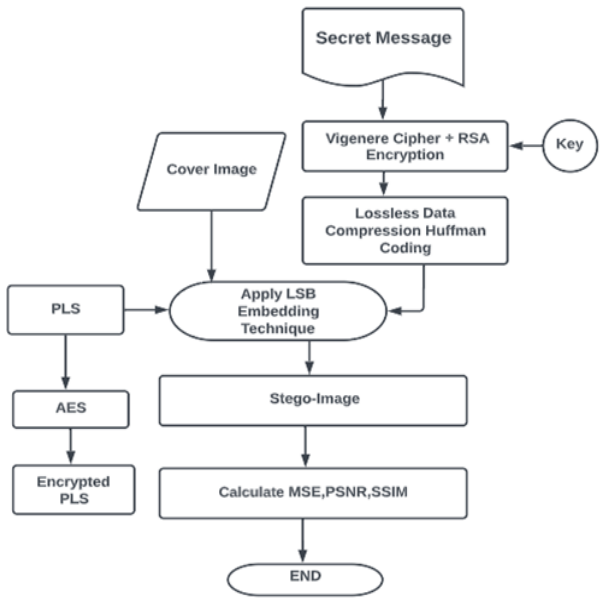

Secret Message
Vigenere Cipher + RSA Encryption
Key
Cover Image
Lossless Data Compression Huffman Coding
PLS
Apply LSB Embedding Technique
AES
Stego-Image
Encrypted PLS
Calculate MSE,PSNR,SSIM
END

Fig -3: Embedding process

LSB Embedding: Using the encrypted Pixel Locator Sequence (PLS), the compressed and encrypted data is embedded into the LSBs of the cover image. The LSBs are the least significant bits of the image pixels, and modifying them has minimal impact on the visual quality of the image. LSB steganography enables the hiding of the compressed data within the cover image without raising suspicion. The algorithm used for LSB embedding is shown below.

Algorithm 2: LSB encoding

```
Result: Text encoded into the image
image[][], PLS[], i = 0, textlist[];
while $i \leq T_n\text{-}1$ do
    while $k \leq textlist[i].size()$ do
        text[] = textlist[i];
        pixel [] = Append RGB values of next three
          pixels from PLS;
        while $j \leq 7$ do
            if text[j]=='0' and pixel[j]%2!=0 then
                pixel[j]−−, j++;
            else
                if text[j]=='1' and !(pixel[j]&1)
                if pixel[j]==0 then
                    pixel[j]++, j++;
                else
                    pixel[j]−−, j++;
                end
            end
        end
        k++;
    end
    i++;
end
```

D. Data Decryption and Extraction

AES Decryption of Pixel Locator Sequence: Upon receiving the stego image with the embedded data, the recipient decrypts the encrypted Pixel Locator Sequence (PLS) using the AES algorithm. This step allows the retrieval of the original PLS used during the embedding process.

LSB Extraction: Using the decrypted Pixel Locator Sequence (PLS), the LSBs are extracted from the stego image. These extracted LSBs contain the compressed and encrypted data. The algorithm used for LSB decoding is shown below.

Algorithm 3: LSB decoding

```
Result: Extracting the encrypted text from image
image[][], PLS[], i = 0, data = '';
while true do
    pixel [] = Append RGB values of next three pixels
      from PLS;
    b_string = '';
    while $j \leq 7$ do
        if pixel[j]%2==0 then
            b_string += '0';
        else
            b_string += '1';
        end
        j++;
    end
    data += char(ASCII(b_string));
end
```

Huffman Decoding: The extracted LSBs are used to reconstruct the compressed data, which is then decoded using the Huffman coding tree. Huffman decoding reverses the compression process and recovers the original encrypted data.

RSA Decryption and Vigenere Cipher Decryption: Finally, the recovered encrypted data is decrypted using the recipient's private key through RSA decryption. Subsequently, the Vigenere cipher decryption is performed using the shared keyword to obtain the original plaintext.

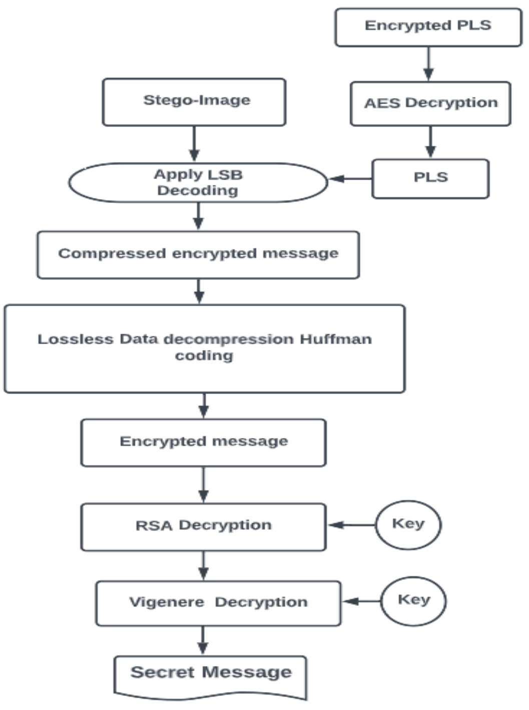


Fig -4: Extraction Process

E. Advantages of proposed method

i)Cryptography and Steganography algorithms combined together provides a double layer of security.

ii)Vigenere cipher makes the message more secure as compared to various other techniques as it is a polyalphabetic cipher technique.

iii)The proposed alogorithm tries to overcome the traditional way of implementing LSB based image steganography technique with a unique sequence that locates the pixel where the information is hidden in sequential order.

4. RESULTS

In this section we present the overall results of the proposed system. The proposed methodology has been implemented using python software in windows 11 operating system environment. The proposed system has been tested using a mix of images from the internet and some pictures from personal gallery. The performance of the system has been analysed using statistical parameters such as Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index (SSIM).

Two different plain texts of varied lengths were considered and have been embedded in four distinct images each using the proposed system.

Plaintext 1: {Name:K.Pranavendra,Roll_No:18011P0410,V-Year_ECE-IDP} [54 characters]

Fig -5: Cover image 1

Fig -6: Stego image 1

Fig -7: Cover image 2

Fig -8: Stego image 2

Fig -9: Cover image 3

Fig -10: Stego image 3

Fig -11: Cover image 4

Fig -12: Stego image 4

Plaintext 2: Steganography is the practice of concealing secret messages within an innocuous carrier medium, such as an image, audio file… [900 characters]

Fig -13: Cover image 5

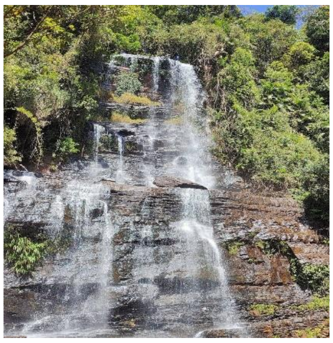


Fig -14: Stego image 5

Fig -15: Cover image 6

Fig -16: Stego image 6

Fig -17: Cover image 7

Fig -18: Stego image 7

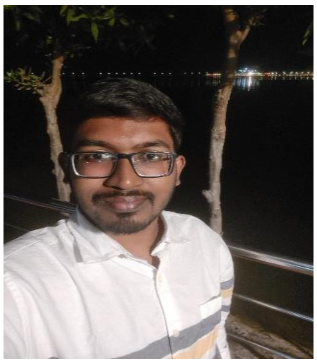


Fig -19: Cover image 8

Fig -20: Stego image 8

5. ANALYSIS

In this sub section, we delve into a comprehensive analysis of the obtained results, employing three key parameters: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Mean Squared Error (MSE). These parameters play a crucial role in evaluating the quality and fidelity of our proposed method, shedding light on the effectiveness and performance of the implemented approach.

Peak Signal to Noise Ratio (PSNR):

PSNR measures the quality of the concealed data by calculating the ratio of the maximum possible power of a signal to the power of corrupting noise. Higher PSNR values signify better quality and fidelity of the concealed data. We have analyzed the obtained PSNR values of different images to gauge the system's ability to maintain data integrity.

Mean Square Error(MSE) :

MSE quantifies the average squared differences between the original and concealed images. A lower MSE value signifies a closer match between the two images. By assessing MSE, we gain insights into the accuracy of our system's data embedding process, providing valuable information on the precision of the concealed data.

Formulae of MSE and PSNR are given below:

$$MSE = \frac{1}{N \times N} \sum_{i=1}^{N} \sum_{j=1}^{N} [X(i,j) - Y(i,j)]^2$$

$$PSNR = 10 \times log_{10}[\frac{255 \times 255}{MSE}]$$

Here, X represents the input cover image and Y represents the output stego image. If the MSE value is greater and the PSNR value is lower, it shows that there is a substantial difference between the input and output pictures.

Structural Similarity Index Measure (SSIM):

SSIM evaluates the structural similarity between the original and the concealed images. It considers various factors such as luminance, contrast, and structure. A higher SSIM value indicates a stronger resemblance between the two images. We delve into the SSIM outcomes across different image sets to understand how effectively our system preserves the visual structure of the images during data embedding. The SSIM formula is as follows:

$$SSIM\ (I_1, I_2) = \frac{(2\mu_{I_1}\mu_{I_2} + \alpha)(2\sigma_{I_1 I_2} + \beta)}{(\mu_{I_1}^2 + \mu_{I_2}^2 + \alpha)(\sigma_{I_1}^2 + \sigma_{I_2}^2 + \beta)}$$

Here I_1 and I_2 are input cover image and output stego images respectively. The averages of I_1 and I_2 are μ_{I_1} *and* μ_{I_2} respectively. The covariance between I_1 and I_2 is $\sigma_{I_1 I_2}$. The variances of I_1 and I_2 are $\sigma_{I_1}^2$ *and* $\sigma_{I_2}^2$ respectively. The SSIM value must be close to 1 between the input cover images and the stego images.

Table-1: PSNR and MSE values between cover and stego images

Image	Resolution	MSE	PSNR(in dB)
Image 1	493X356	0.0062	70.2275
Image 2	225X225	0.0213	64.8456
Image 3	512X384	0.0055	70.7430
Image 4	299X168	0.0206	64.9912
Image 5	864X1152	0.0155	66.2224
Image 6	1280X960	0.0125	67.1665
Image 7	1200X1600	0.0078	69.2138
Image 8	4896X6528	0.0005	81.1875

Table-2: SSIM values between cover and stego images

Image	SSIM
Image 1	0.999
Image 2	0.998
Image 3	0.999
Image 4	0.997
Image 5	0.999
Image 6	0.999
Image 7	0.999
Image 8	0.999

5. CONCLUSION

In this project, we have successfully implemented a comprehensive data encryption system using various cryptographic techniques. The system employs the Vigenere cipher, RSA algorithm, Huffman coding, and PLS-based LSB steganography to ensure confidentiality, integrity, and secure transmission of sensitive information.

Through the implementation process, we have gained valuable insights into the working principles of each algorithm and their integration into a cohesive system. The Vigenere cipher provides a polyalphabetic substitution technique, adding an additional layer of complexity to the encryption process. The RSA algorithm, with its asymmetric key pair generation and encryption/decryption mechanisms, enables secure communication between the sender and receiver.

We have utilized Huffman coding to compress the encrypted data, reducing its size while maintaining its integrity. The compressed data is then embedded into images using PLS-based LSB steganography, ensuring that the hidden message remains inconspicuous to unauthorized individuals.

The implementation of the Fisher-Yates algorithm for PLS generation has further enhanced the security of our system by generating a randomized pixel locator sequence, making it difficult for attackers to identify the embedded data.

The implemented system not only focused on encryption but also included performance evaluation metrics to assess the quality of the encrypted data. The structural similarity index (SSIM), mean squared error (MSE), and peak signal-to-noise ratio (PSNR) metrics were utilized to measure the performance of the system. The SSIM, MSE, and PSNR values were within acceptable ranges, indicating that the encrypted and decrypted data retained their integrity and quality.

REFERENCES

[1] C. Biswas, U. D. Gupta and M. M. Haque, "An Efficient Algorithm for Confidentiality, Integrity and Authentication Using Hybrid Cryptography and Steganography," 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), 2019, pp. 1-5, doi: 10.1109/ECACE.2019.8679136.

[2] O. F. A. Wahab, A. A. M. Khalaf, A. I. Hussein and H. F. A. Hamed, "Hiding Data Using Efficient Combination of RSA Cryptography, and Compression Steganography Techniques," in IEEE Access, vol. 9, pp. 31805-31815, 2021, doi: 10.1109/ACCESS.2021.3060317.

[3] O. F. A. Wahab, A. A. M. Khalaf, A. I. Hussein and H. F. A. Hamed, "Hiding Data Using Efficient Combination of RSA Cryptography, and Compression Steganography Techniques," in IEEE Access, vol. 9, pp. 31805-31815, 2021, doi: 10.1109/ACCESS.2021.3060317.

[4] K. A. Babu and V. S. Kumar, "Implementation of data compression using Huffman coding," 2010 International Conference on Methods and Models in Computer Science (ICM2CS-2010), New Delhi, India, 2010, pp. 70-75, doi: 10.1109/ICM2CS.2010.5706721.

[5] K. Tiwari and S. J. Gangurde, "LSB Steganography Using Pixel Locator Sequence with AES," 2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC), Jalandhar, India, 2021, pp. 302-307, doi: 10.1109/ICSCCC51823.2021.9478162.

[6] S. Al Busafi and B. Kumar, "Review and Analysis of Cryptography Techniques," 2020 9th International Conference System Modeling and Advancement in Research Trends (SMART), Moradabad, India, 2020, pp. 323-327, doi: 10.1109/SMART50582.2020.9336792.

[7] T. R. Devi, "Importance of Cryptography in Network Security," 2013 International Conference on Communication Systems and Network Technologies, Gwalior, India, 2013, pp. 462-467, doi: 10.1109/CSNT.2013.102.

[8] K. C. Nunna and R. Marapareddy, "Secure Data Transfer Through Internet Using Cryptography and Image Steganography," 2020 SoutheastCon, Raleigh, NC, USA, 2020, pp. 1-5, doi: 10.1109/SoutheastCon44009.2020.9368301.

[9] R. Mishra and P. Bhanodiya, "A review on steganography and cryptography," 2015 International Conference on Advances in Computer Engineering and Applications, Ghaziabad, India, 2015, pp. 119-122, doi: 10.1109/ICACEA.2015.7164679.