ABSTRACT:

Steganography is the technique of hiding secret data within an ordinary, non-secret, file or message in order to avoid detection; the secret data is then extracted at its destination. Image steganography is a technique used to hide secret data within an image and it is combined with encryption as an extra step for hiding or protecting data. The main objective of this proposed work is to hide the secret message inside the image using Least Significant Bit (LSB) technique and provide security for the hidden message using Advanced Encryption Standard (AES) Algorithm. Additionally the RGB values of each pixel in an image is extracted and saving those values into a CSV (Comma-Separated Values) file.

KEYWORDS - AES, Cryptography, Image steganography, LSB.

TABLE OF CONTENTS

LIST OF FIGURES

1.INTRODUCTION

1.1 CRYPTOGRAPHY

Cryptography is the practice of securing communication and data by transforming it into a format that is unintelligible to unauthorized individuals. It involves various techniques and algorithms to ensure confidentiality, integrity, authentication, and non-repudiation of information.

Cryptography is technique of securing information and communications through use of codes so that only those person for whom the information is intended can understand it and process it. Thus preventing unauthorized access to information. The prefix "crypt" means "hidden" and suffix "graphy" means "writing". In Cryptography the techniques which are use to protect information are obtained from mathematical concepts and a set of rule based calculations known as algorithms to convert messages in ways that make it hard to decode it. These algorithms are used for cryptographic key generation, digital signing, verification to protect data privacy, web browsing on internet and to protect confidential transactions such as credit card and debit card transactions.

Techniques used For Cryptography:

In today's age of computers cryptography is often associated with the process where an ordinary plain text is converted to cipher text which is the text made such that intended receiver of the text can only decode it and hence this process is known as encryption. The process of conversion of cipher text to plain text this is known as decryption.

Features Of Cryptography are as follows:

Confidentiality: Information can only be accessed by the person for whom it is intended and no other person except him can access it.

Integrity: Information cannot be modified in storage or transition between sender and intended receiver without any addition to information being detected.

Non-repudiation: The creator/sender of information cannot deny his intention to send information at later stage.

Authentication: The identities of sender and receiver are confirmed. As well as destination/origin of information is confirmed

1.2 TYPES OF CRYPTOGRAPHY

There are several types of cryptography, each with its own approach and purpose. Here are some commonly recognized types of cryptography:

1. **Symmetric Key Cryptography**:

   Also known as secret-key or single-key cryptography, this type uses the same key for both encryption and decryption. The sender and the recipient must possess and securely share the secret key. Symmetric key cryptography is generally faster than asymmetric key cryptography but requires a secure key distribution mechanism.

2. **Asymmetric Key Cryptography**:

   Also referred to as public-key cryptography, this type uses a pair of mathematically related keys: a public key for encryption and a private key for decryption. The public key can be freely distributed, while the private key remains secret. Asymmetric key cryptography enables secure communication and key exchange without requiring prior key distribution.

3. **Hash Functions**:

   Hash functions take input data and produce a fixed-size string of characters, known as a hash value or digest. These functions are designed to be one-way, meaning it is computationally infeasible to derive the original input from the hash value. Hash functions are used for data integrity checks, password storage, and digital signatures.

4. **Message Authentication Codes (MAC)**:

   MAC is a cryptographic technique that uses a secret key to generate a unique tag for a message or data. This tag acts as a cryptographic checksum, ensuring the integrity and authenticity of the message. MACs are commonly used in network protocols and data authentication.

5. **Digital Signatures**:

   Digital signatures provide a way to verify the authenticity, integrity, and non-repudiation of digital documents or messages. They use asymmetric key cryptography to create a unique digital signature that can be verified using the

corresponding public key. Digital signatures are widely used for secure email, document signing, and authentication purposes.

6. **Elliptic Curve Cryptography (ECC)**:

   ECC is a form of public-key cryptography based on elliptic curves over finite fields. It offers the same level of security as other public-key algorithms with smaller key sizes, making it more efficient for resource-constrained devices. ECC is commonly used in applications such as secure communication and mobile devices.

7. **Quantum Cryptography**:

   Quantum cryptography is a field that explores cryptographic techniques based on the principles of quantum mechanics. Quantum cryptography offers secure key exchange and encryption methods that are resistant to quantum attacks. Quantum key distribution (QKD) is a notable example of quantum cryptography.

1.2.1 Symmetric Key Cryptography:

Symmetric key cryptography is a type of encryption scheme in which the similar key is used both to encrypt and decrypt messages. Such an approach of encoding data has been largely used in the previous decades to facilitate secret communication between governments and militaries.

Symmetric-key cryptography is called a shared-key, secret-key, single-key, one-key and eventually private-key cryptography. With this form of cryptography, it is clear that the key should be known to both the sender and the receiver that the shared. The complexity with this approach is the distribution of the key.

Symmetric key cryptography schemes are usually categorized such as stream ciphers or block ciphers. Stream ciphers work on a single bit (byte or computer word) at a time and execute some form of feedback structure so that the key is repeatedly changing.

1.2.2 Asymmetric Key Cryptography:

Asymmetric cryptography is a second form of cryptography. It is called a Public-key cryptography. There are two different keys including one key is used for encryption and only the other corresponding key should be used for decryption. There is no other key can decrypt the message and not even the initial key used for encryption. The style of the design is that

every communicating party needs only a key pair for communicating with any number of other communicating parties.

Asymmetric cryptography is scalable for use in high and ever expanding environments where data are generally exchanged between different communication partners. Asymmetric cryptography is used to exchange the secret key to prepare for using symmetric cryptography to encrypt information.

In the case of a key exchange, one party produce the secret key and encrypts it with the public key of the recipient. The recipient can decrypt it with their private key. The remaining communication would be completed with the secret key being the encryption key. Asymmetric encryption is used in key exchange, email security, Web security, and some encryption systems that needed key exchange over the public network.

1.3 APPLICATIONS OF CRYPTOGRAPHY:

Computer passwords: Cryptography is widely utilized in computer security, particularly when creating and maintaining passwords. When a user logs in, their password is hashed and compared to the hash that was previously stored. Passwords are hashed and encrypted before being stored. In this technique, the passwords are encrypted so that even if a hacker gains access to the password database, they cannot read the passwords.

Digital Currencies: To safeguard transactions and prevent fraud, digital currencies like Bitcoin also use cryptography. Complex algorithms and cryptographic keys are used to safeguard transactions, making it nearly hard to tamper with or forge the transactions.

Secure web browsing: Online browsing security is provided by the use of cryptography, which shields users from eavesdropping and man-in-the-middle assaults. Public key cryptography is used by the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols to encrypt data sent between the web server and the client, establishing a secure channel for communication.

Electronic signatures: Electronic signatures serve as the digital equivalent of a handwritten signature and are used to sign documents. Digital signatures are created using cryptography and can be validated using public key cryptography. In many nations, electronic signatures are enforceable by law, and their use is expanding quickly.

Authentication: Cryptography is used for authentication in many different situations, such as when accessing a bank account, logging into a computer, or using a secure network. Cryptographic methods are employed by authentication protocols to confirm the user's identity and confirm that they have the required access rights to the resource.

Cryptocurrencies: Cryptography is heavily used by cryptocurrencies like Bitcoin and Ethereum to safeguard transactions, thwart fraud, and maintain the network's integrity. Complex algorithms and cryptographic keys are used to safeguard transactions, making it nearly hard to tamper with or forge the transactions.

1.4 STEGANOGRAPHY

Steganography is the practice of concealing secret or sensitive information within non-secret data, such as images, audio files, videos, or text, in a way that doesn't raise suspicion. The goal of steganography is to hide the existence of the hidden information, making it difficult for unauthorized individuals to detect or extract the concealed data.

Cryptography and steganography are both methods used to hide or protect secret data. However, they differ in the respect that cryptography makes the data unreadable, or hides the meaning of the data, while steganography hides the existence of the data.

Unlike cryptography, which focuses on encrypting the content of a message to make it unreadable, steganography aims to hide the very existence of the message. By embedding the secret information within the carrier medium, steganography allows for covert communication without drawing attention.

1.5 TYPES OF STEGANOGRAPHY:

There are 4 types of steganography. They are:

- Image Steganography
- Audio Steganography
- Text Steganography
- Video Steganography
- Network Steganography (Protocol Steganography)

1.5.1 Image Steganography:

This technique involves hiding secret data within digital images. The least significant bit (LSB) manipulation is a widely used method, where the least significant bits of the pixel values are altered to embed the hidden message. Since the changes to the LSBs are often imperceptible to the human eye, the visual quality of the image remains mostly intact.

1.5.2 Audio Steganography:

Similar to image steganography, this technique hides data within audio files. By slightly modifying the amplitude or frequency of audio samples, hidden information can be concealed in the audio signal. This method can be used to hide data in various audio formats, including WAV, MP3, or even speech signals.

1.5.3 Text Steganography:

This technique involves hiding information within the text itself. Various methods can be employed, such as using invisible characters or modifying the arrangement of letters, words, or spaces in the text. Various techniques used to hide the data in the text are:

- Format Based Method
- Random and Statistical Generation
- Linguistic Method

1.5.4 Video Steganography:

Video steganography involves hiding secret data within video files. This can be achieved by modifying specific frames or by embedding the information across multiple frames of the video.

1.5.5 Network Steganography (Protocol Steganography)

It is the technique of embedding information within network control protocols used in data transmission such TCP, UDP, ICMP etc. You can use steganography in some covert channels that you can find in the OSI model. For Example, you can hide information in the header of a TCP/IP packet in some fields that are either optional.

1.6 IMAGE STEGANOGRAPHY:

Image steganography is a technique of hiding secret or confidential information within digital images without visibly altering the image's appearance. It allows for covert communication by embedding the hidden data in the image pixels.

The most common method used in image steganography is manipulating the least significant bit (LSB) of the pixel values. In digital images, each pixel is represented by a binary value composed of color channels such as red, green, and blue (RGB). The LSB is the least significant binary digit of each color channel.

1.6.1 TYPES OF IMAGE STEGANOGRAPHY TECHNIQUES:

There are several types of image steganography are there, they are

1. **Least Significant Bit (LSB) Steganography**: This method involves replacing the least significant bit of each pixel in an image with the secret data bits. Since the LSB holds the least visual impact on the image, this technique is widely used.
2. **Pixel-Value Differencing (PVD)**: PVD steganography utilizes the differences between adjacent pixel values to hide data. The differences are modified according to the secret data, and the changes are often imperceptible to the human eye.
3. **Spread Spectrum Steganography**: This method spreads the secret data across multiple pixels in the image. The spread spectrum techniques, such as frequency domain or wavelet-based methods, ensure that the hidden data is distributed across the image to avoid detection.
4. **Transform Domain Techniques**: Transform domain techniques involve transforming the image using mathematical transformations like the Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT). The secret data is embedded in the transformed coefficients, and then the inverse transform is applied to retrieve the image.
5. **Adaptive Steganography**: Adaptive steganography methods analyze the content and characteristics of the image to determine the optimal embedding strategy. These

techniques adaptively adjust the embedding parameters based on the local properties of the image, making the hidden data less noticeable.

6. **Randomized Steganography**: Randomized steganography techniques introduce randomness into the embedding process to increase security. The secret data is embedded based on a random pattern or sequence, making it difficult for an adversary to distinguish between the cover and stego images.
7. **Text-based Steganography**: Text-based steganography involves encoding secret information within the textual content of an image file. This technique utilizes the ASCII or Unicode values of characters to embed the hidden data.

1.7 TECHNIQUES USED IN IMAGE STEGANOGRAPHY:

Spatial domain techniques are broadly classified into:

- Least significant bit (LSB)
- Pixel value differencing (PVD)
- Edges based data embedding method (EBE)
- Random pixel embedding method (RPE)
- Mapping pixel to hidden data method.
- Labeling or connectivity method.
- Pixel intensity based method.
- Texture based method.

1.8 RGB PIXEL VALUES

In digital image processing, RGB (Red, Green, Blue) is a common color model used to represent colors in images. In this model, each pixel of an image is described by three numeric values representing the intensity of the red, green, and blue color channels. These values are often referred to as RGB pixel values.

The RGB pixel values typically range from 0 to 255, where 0 represents the absence of a color channel and 255 represents the maximum intensity of that color channel. The

combination of these three color channels in varying intensities allows for the creation of a wide range of colors.

For example, an RGB pixel value of (255, 0, 0) represents full intensity red, (0, 255, 0) represents full intensity green, and (0, 0, 255) represents full intensity blue. Mixing these primary colors in different proportions creates a variety of intermediate colors.

Here's how the RGB pixel values work:

- Red Channel: The value represents the intensity of the red color channel in the pixel.
- Green Channel: The value represents the intensity of the green color channel in the pixel.
- Blue Channel: The value represents the intensity of the blue color channel in the pixel.

For grayscale images, where each pixel represents a single intensity value, the RGB pixel values would be equal. For example, a grayscale pixel with an intensity of 128 would be represented as (128, 128, 128) in RGB.

CHAPTER 2:

LITRATURE SURVEY

2.1 BASE PEPER IDENTIFICATION

1.TITLE: A Novel Steganography Technique for Digital Images Using the Least Significant Bit Substitution Method

AUTHOR: Shahid Rahman, Jamal Uddin et al.

WORK DONE:

The proposed work, in this paper, is based on the Least Significant Bit (LSB) substitution method. In this paper, they proposed a novel technique in steganography within the digital images such is RGB, Gray Scale, Texture, Aerial images to achieve higher security, imperceptibility, capacity, and robustness as compared with existing methods

MERITS:

- Simplicity and ease of implementation.
- Low visual impact.
- High embedding capacity

DEMERITS:

- Sensitivity to image processing.
- Incompatibility with certain image formats.

2.TITLE : A Secure Image Steganography using LSB and Double XOR Operations.

AUTHOR: , Ali Ahmed and Abdelmotalib Ahmed

WORK DONE:

In this paper, the researchers suggest a method that involves two layers of encryption and hiding stages. First, the message is encrypted using a secret key and double XOR operations in binary form. Then, the encrypted message is hidden within the cover image

using the LSB technique. To make sure their method is reliable, they calculated well-known evaluation measures such as MSE, PSNR, Entropy, and histogram distribution.

MERITS:

- Resistance against statistical analysis.
- Improved robustness against image processing.
- Increased resistance against steganalysis techniques

DEMERITS:

- Complexity and implementation challenges.
- Vulnerability to steganalysis techniques.

3.TITLE: Image Steganography based Cryptography

AUTHOR: Jemima Dias, Dr. Ajit Danti

WORK DONE:

In this paper, the image encryption and decryption method has been proposed where the secret image gets encrypted carefully into the cover image using the NET, based on using XOR operation between two images using the weights present within the network**.**

MERITS:

- Dual protection.
- Resistance to brute-force attacks.
- Seamless integration.

DEMERITS:

- Sensitivity to initial conditions.
- Lack of cryptographic strength.

4.TITLE: Image steganography using lsb bit-plane substitution

AUTHOR: Tanmay sinha roy.

WORK DONE:

In this paper, we have used the BIT-PLANE substitution method to hide a Messase image into a Cover image. we have considered two uint8 digital images, one is the cover image and the other is the message image. In the LSB approach, the basic idea is to replace the Least Significant Bits (LSB) of the cover image with the Bits of the message image to be hidden without destroying the property of the cover image significantly.

MERITS:

- Simplicity and ease of implementation
- High embedding capacity
- Low computational overhead

DEMERITS:

- Low robustness against intentional attacks
- Susceptibility to lossy compression

5.TITLE: Data hiding in image using lsb with des cryptography.

AUTHOR: Krati Yadav, Swapnil Rajput.

WORK DONE:

This paper discussed a technique used on the LSB (least significant bit) and a new encryption algorithm. By matching data to an image, there is less chance of an attacker being able to use steg analysis to recover data. Before hiding the data in an image the application first encrypts it.

MERITS:

- Robust against cryptographic attacks.
- Enhanced confidentiality.
- Enhanced confidentiality.

DEMERITS:

- Dependency on the encryption algorithm.
- Performance overhead

6.TITLE: End-to-End Image Steganography Using Deep Convolutional Autoencoders

AUTHOR: Nandhini Subramanian, Ismahane cheheb.

WORKDONE:

The proposed method has been evaluated using various images including Lena, airplane, baboon and peppers and compared against other traditional image steganography methods. The experimental results have demonstrated that the proposed method has higher hiding capacity, security and robustness, and imperceptibility performances than other deep learning image steganography methods.

MERITS:

- Integration of steganography within a single network
- Robustness to steganalysis
- Preservation of image quality
- Adaptability to different types of images

DEMERITS:

- Limited embedding capacity
- Vulnerability to adversarial attacks
- Computational complexity

7.TITLE: Steganography in Digital Images Using LSB Technique

AUTHOR: Dr. Arun K Singh.

WORKDONE:

In this paper one method of data has been discussed in digital images. One image format in which each pixel is represented using 8-bit binary number whose range come from zero to 255. So gray scale image maximum having 256 colors in each pixel. LSB hiding technique uses LSB of 8-bit number representing a pixel.

MERITS:

- Simplicity and ease of implementation

- High embedding capacity
- Imperceptibility
- Compatibility

DEMERITS:

- Low robustness against intentional attacks
- Susceptibility to lossy compression

8. TITLE: A Multiple-Format Steganography Algorithm for Color Images.

AUTHOR: Arshiya S. Ansari, Mohammad S. Mohammadi.

WORKDONE:

The proposed algorithm is the first Steganography algorithm that can work for multiple cover image formats. In addition, we have utilized concepts like capacity pre-estimation, adaptive partition schemes and data spreading to embed secret data with enhanced security.

MERITS:

- Increased embedding capacity
- Enhanced security and robustness
- Improved imperceptibility

DEMERITS:

- Increased complexity.
- Reduced embedding capacity per format

9.TITLE: RC4 Algorithm and Steganography to Double Secure Messages in Digital Image.

AUTHOR: Rahmat Sulaiman, Chandra Kirana.

WORKDONE:

In this research, we use the time process of encryption and decryption as a parameter, and we also compare the Stego image based on BITMAP and JPEG format. The result of the Stego image in the BITMAP format shows us that this method is better than the Stego image in JPEG format.

MERITS:

- Encryption strength
- Dual-layer security
- Camouflage and inconspicuousness

DEMERITS:

- Vulnerability to cryptographic attacks
- Key management challenges

10.TITLE: Enhancement of Security of Diffie-Hellman Key Exchange Protocol using RSA Cryptography.

AUTHOR: Chira deep Gupta , N V Subba Reddy.

WORKDONE:

This paper proposes a model of integrating the publickey RSA cryptography system with the DH key exchange to prevent the MITM attack. The performance of the proposed work has been compared to the DH Key Exchange algorithm as well as RSA Cryptosystem to conclude for effectiveness of the proposed model.

MERITS:

- Strong security guarantees
- Authentication and non-repudiation
- Key secrecy preservation

DEMERITS:

- Increased computational overhead
- Key size considerations

2.2 DRAWBACKS OF THE EXISTING WORK:

Key Management Challenges:

With DES, both the sender and receiver must have access to the same secret key. Establishing and securely exchanging the key can be a challenge, especially when communicating over an insecure channel. If the key is compromised or intercepted, the security of the communication can be compromised as well.

Limited Embedding Capacity:

LSB steganography replaces the least significant bit of pixel values with hidden data, which limits the amount of information that can be embedded within an image. The embedding capacity depends on the size of the image and the number of bits modified per pixel. This limited capacity may not be sufficient for large amounts of data.

Susceptibility to Image Manipulation:

LSB steganography modifies the least significant bit of pixel values, which results in a minor alteration of the image. However, this alteration can be detected through visual inspection or by applying specific analysis techniques. Attackers who are aware of the steganographic method being used can potentially remove or manipulate the hidden data, compromising the integrity and confidentiality of the message.

Fragility to Lossy Compression:

If the steganographically altered image undergoes lossy compression, such as JPEG compression, the hidden data may be significantly altered or lost altogether. The compression algorithm can introduce artifacts and quantization errors, leading to the corruption or removal of the embedded information.

Low security:

LSB steganography is a weak encryption method, and DES is a relatively old and weak block cipher. This combination can be easily broken by attackers with access to powerful computing resources.

CHAPTER 3

3.1 LEAST SIGNIFICANT BIT TECHNIQUE:

The LSB (Least Significant Bit) technique is one of the simplest and most commonly used methods in image steganography. It involves replacing the least significant bit of each pixel in an image with the secret data bits. The LSB of a binary number represents the smallest and least noticeable change in its value.

The LSB technique in steganography involves replacing the LSB of the pixel values with the secret data bits. Since the LSB has the least impact on the visual appearance of the image, modifying it typically results in minimal perceptual changes. By replacing the LSB with the secret data bits, one can hide information within the image.

LSB-Steganography is a steganography technique in which we hide messages inside an image by replacing Least significant bit of image with the bits of message to be hidden

For example, consider a pixel value of 150, which in binary is represented as 10010110. If the secret data bit to be embedded is 1, the LSB of the pixel value would be modified to 1, resulting in 10010111

LSB ENCODER

The LSB (Least Significant Bit) Encoder is a tool or program used in image steganography to hide secret data within the least significant bit of pixel values in an image. It replaces the LSB of each pixel with the secret data bits, thereby embedding the data within the image.

Digital data is computed in binary format, where the rightmost digit is considered the lowest digit whereas the leftmost is considered the highest digit. In positional notation, the least significant bit is also known as the rightmost bit. It is the opposite of the most significant bit, which carries the highest value in a multiple-bit binary number as well as the number which is farthest to the right. In a multi-bit binary number, the significance of a bit decreases as it approaches the least significant bit. Since it is binary, the most significant bit can be either 1 or 0. The least significant bit is frequently employed in hash functions, checksums and pseudorandom number generators.

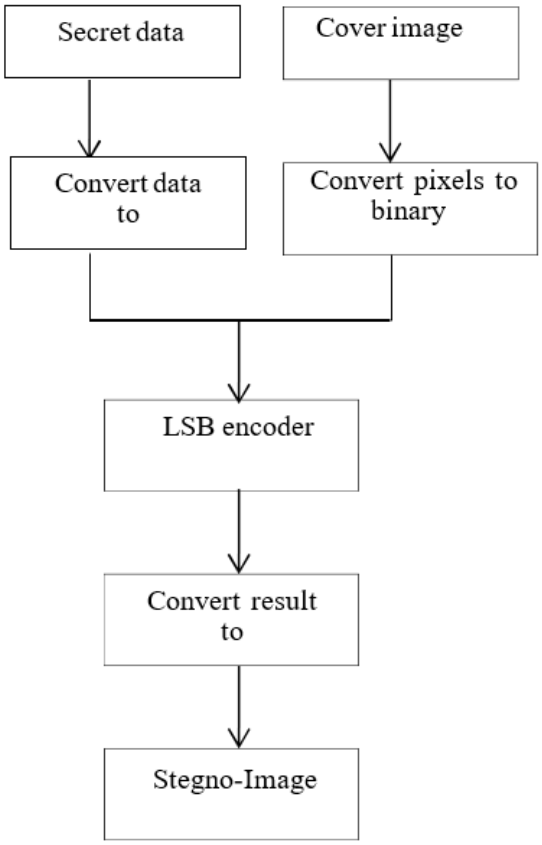

Figure:3.1

The LSB Encoder typically performs the following steps:

1. **Image Selection**: Choose the cover image or the image in which you want to hide the secret data.
2. **Secret Data Conversion**: Convert the secret data into binary format. Each character or byte of the secret data is represented as a sequence of bits.
3. **Data Embedding**: Start from the first pixel (or channel) of the image and the first bit of the secret data. Replace the LSB of the pixel value with the secret data bit. Repeat this process for each bit and pixel until all the secret data is embedded.
4. **Preservation of Image Quality**: During data embedding, the LSB Encoder ensures that the changes made to the LSB of the pixel values do not significantly affect the visual quality of the image. The modifications should be imperceptible to the human eye.

5. **Output**: The LSB Encoder generates a stego image, which is the cover image with the secret data embedded.

LSB DECODER

The LSB (Least Significant Bit) Decoder is a tool or program used in image steganography to extract secret data that has been hidden using LSB encoding. It reverses the process of LSB encoding by extracting the LSBs of pixel values from a stego image to retrieve the hidden data

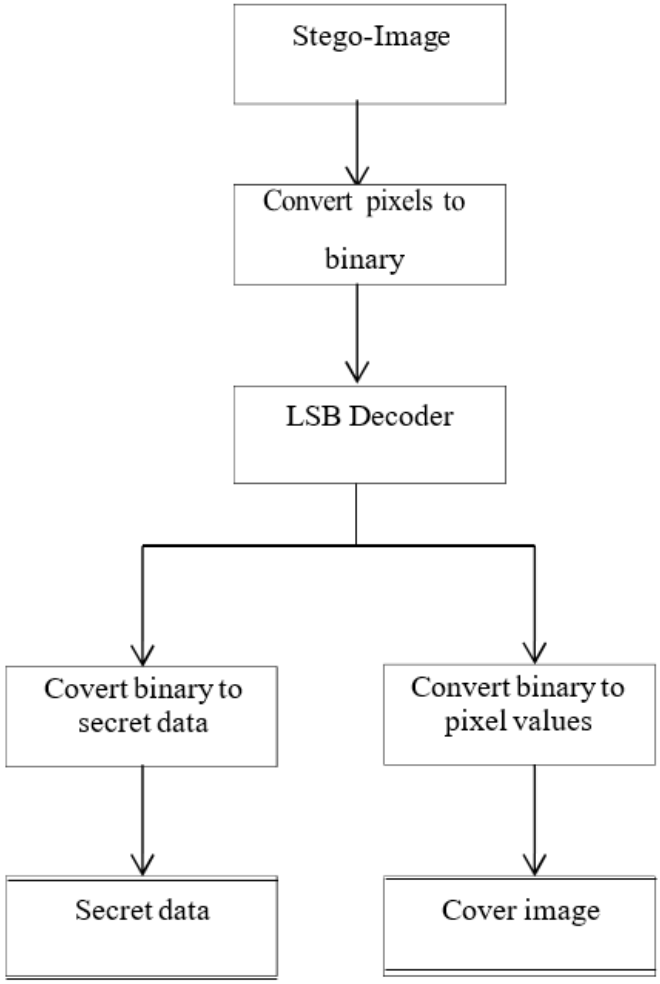

Figure:3.2

The LSB Decoder typically follows these steps:

1. **Stego Image Selection**: Choose the stego image from which you want to extract the hidden data.

2. **Data Extraction**: Start from the first pixel (or channel) of the stego image and extract the LSB of the pixel value. Collect these LSBs sequentially to reconstruct the binary representation of the hidden data.

3. **Conversion of Extracted Data**: Convert the extracted binary data back into its original form. This process depends on the format and structure of the hidden data. For example, if the hidden data was originally text, you would convert the binary data into characters or a string.

4. **Output:** The LSB Decoder generates the extracted secret data in its original form, allowing you to access and use the hidden information.

It's important to note that the LSB Decoder should be used with the same LSB encoding technique that was employed during the hiding process. If any additional encryption or compression was applied to the hidden data before embedding, the decoder should handle those processes accordingly to obtain the final, meaningful information.

LSB decoding is a basic approach in steganography, and while it can extract the hidden data, it has limitations in terms of capacity and security. Advanced steganalysis techniques may be able to detect the presence of LSB-encoded data. Therefore, more robust and sophisticated steganography methods may be necessary for higher security requirements.

3.2 ADVANCED ENCRYPTION STANDARD:

The AES (Advanced Encryption Standard) algorithm is a widely used symmetric key encryption algorithm. It is a symmetric block cipher that operates on fixed-size blocks of data and uses a secret key for both encryption and decryption. AES is designed to provide a high level of security while being computationally efficient.

- AES is a block cipher.
- The key size can be 128/192/256 bits.
- Encrypts data in blocks of 128 bits each.

That means it takes 128 bits as input and outputs 128 bits of encrypted cipher text as output. AES relies on substitution-permutation network principle which means it is performed using a series of linked operations which involves replacing and shuffling of the input data.

3.3 TYPES OF ADVANCED ENCRYPTION ALGORITHM

Here are the main types or variations associated with the AES algorithm:

- **AES-128:**

This variant of AES uses a 128-bit key size. It operates on 128-bit blocks and undergoes a fixed number of encryption rounds (10 rounds for AES-128).

- **AES-192**:

This variant uses a 192-bit key size. It operates on 128-bit blocks and undergoes a higher number of encryption rounds (12 rounds for AES-192).

- **AES-256:**

This variant uses a 256-bit key size. It operates on 128-bit blocks and undergoes the highest number of encryption rounds (14 rounds for AES-256).The key size determines the level of security provided by AES. AES-256, with its 256-bit key size, offers a higher level of security compared to AES-192 and AES-128. However, all three variants of AES are considered secure and widely used.

3.4 DIFFIE-HELLMAN KEY EXCHANGE TECHNIQUE

The Diffie-Hellman key exchange is a cryptographic technique that allows two parties to securely establish a shared secret key over an insecure communication channel.

The key exchange process work as follows:

1. **Initialization**: Both parties, let's call them Alice and Bob, agree on a common set of parameters. These parameters include a large prime number, usually denoted as "p," and a primitive root modulo p, denoted as "g." These parameters are public and can be shared openly.

2. **Private Keys**: Alice and Bob each independently choose a private key. Let's call Alice's private key "a" and Bob's private key "b." These private keys are kept secret and not shared with anyone.

3. **Public Keys**: Using the chosen parameters and their respective private keys, Alice and Bob generate their public keys. Alice computes her public key, denoted as "A,"

by taking g raised to the power of her private key (A = g^a mod p). Bob similarly computes his public key, denoted as "B," by taking g raised to the power of his private key (B = g^b mod p). The public keys are then exchanged between the two parties over the insecure channel.

4. **Shared Secret**: After receiving the other party's public key, Alice and Bob independently calculate the shared secret key. Alice computes the shared secret key, denoted as "S," by taking Bob's public key raised to the power of her private key (S = B^a mod p). Bob similarly computes the shared secret key, denoted as "S," by taking Alice's public key raised to the power of his private key (S = A^b mod p).
5. **Shared Key**: Alice and Bob now have a shared secret key, which is the same for both of them. This shared key can be used for symmetric encryption or other cryptographic operations to secure their communication.

CHAPTER 4

4.PROPOSED WORK:

4.1. LSB TECHNIQUE COMBINED WITH AES-256 ALGORITHM:

Our approach is to provide security for the secret information hidden inside the cover image performed through LSB technique with the help of AES algorithm. The first step is to read the secret data and the cover image. The next step is to hide the data into the cover image which is done with the help of LSB encoder. To protect the data from the intruder the resultant stego-image is given to AES encryption where each of the pixels get scrambled .To get back the stego-image AES decryption is done and the output of this is provided to LSB decoder where the hidden data is retrieved finally.

General Block Diagram:

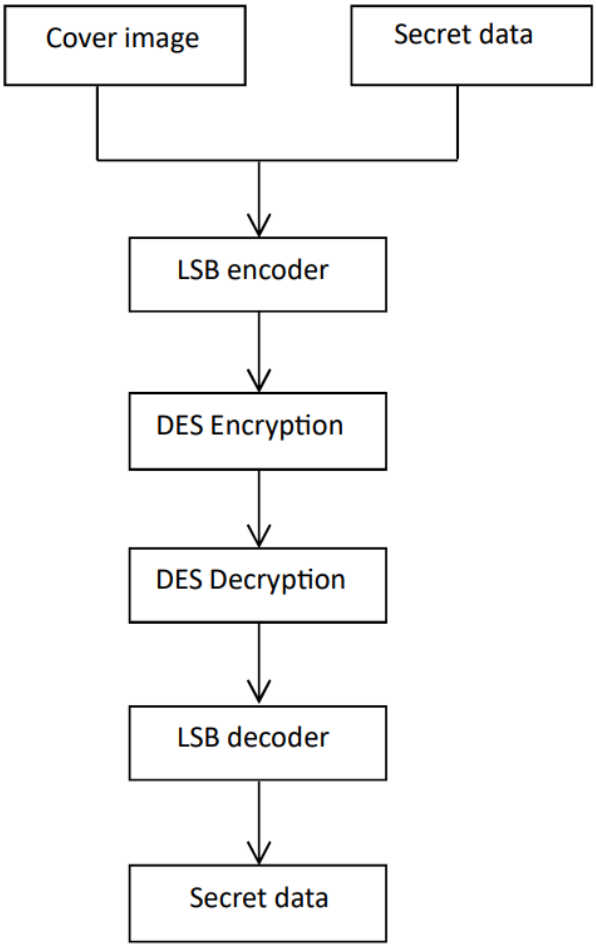

Figure:4.1

STEPS TO FOLLOW FOR HIDING:

1. Read the input cover image of size [256 256].

2. Read the secret data i.e. text or the image.

3. Apply LSB encoder where the bits of the secret data are hidden into the least significant bit of the pixel value of the cover image.

4. The resultant stego-image is given to AES encryption in order to provide security for the hidden data.

5. The function of the AES decryption is to get back the stego image in order to retrieve back the data.

6. The LSB decoder finds each of the data which was embedded in the cover image and the resultant is the secret data.

4.2 ADVANTAGES

The advantage of steganography is as follows −

- The advantage of steganography is that messages do not send consideration to themselves. Clearly detectable encrypted message no matter how tough will stimulate suspicion, and may in themselves be compromising in countries where encryption is illegitimate.
- In steganography, cryptography secures the contents of a message, steganography can be said to secure both messages and connecting parties.
- This approach featured security, capacity, and robustness, the three needed element of steganography that creates it beneficial in hidden exchange of data through text files and creating secret communication.
- There are some important files carrying confidential data can be in the server in and encrypted form and No intruder can receive some beneficial information from the initial file during transmit.
- With the need of Steganography Corporation government and law enforcement agencies can connect privately.
- The major objective of steganography is to connect privately in a completely imperceptible aspect and to prevent drawing uncertainty to the transmission

of a hidden information. It is not to maintain others from understanding the hidden data, but it is to maintain others from thinking that the data even exists. If a steganography approach generates someone to suspect the carrier medium, thus the method has unsuccessful.

- The advantage of steganography is that it can be generally used to secretly send messages without the case of the transmission being found. By using encryption, it can recognize the sender and the receiver.
- Steganography has a double component of protection such as first, the file itself is secret and second, the data in it is encoded.

4.3 BLOCK DIAGRAM:

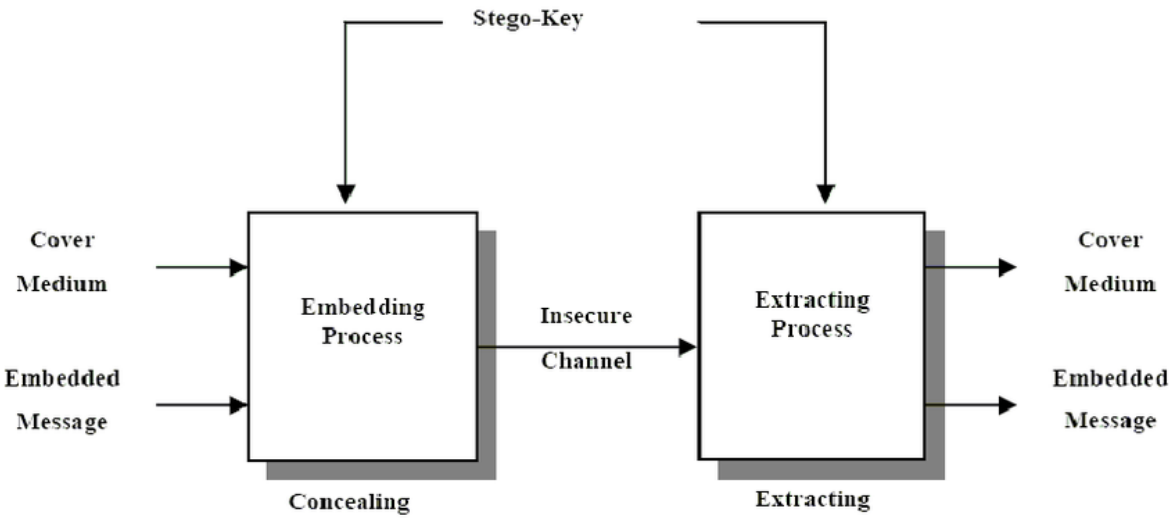

Fig 4.3

4.4 REQUIREMENTS

4.4.1 SOFTWARE REQUIREMENTS

Language: Python 3.1,

IDE: Visual Studio Code or PyCharm.

LANGUAGE SPECIFICATION: This project is implemented in PYTHON.

INTRODUCTION TO PYTHON:

Python is a very popular general-purpose interpreted, interactive, object oriented, and high-level programming language. Python is dynamically typed and garbage-collected programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Python supports multiple programming paradigms, including Procedural, Object Oriented and Functional programming language. Python design philosophy emphasizes code readability with the use of significant indentation.

PYTHON FEATURES:

Python's features include –

➢ Easy-to-learn – ▪ Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

➢ Easy-to-read – ▪ Python code is more clearly defined and visible to the eyes. ➢ Easy-to-maintain – ▪ Python's source code is fairly easy-to-maintain.

➢ A broad standard library – ▪ Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

➢ Portable – ▪ Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

➢ Extendable – ▪ You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

➢ Databases – ▪ Python provides interfaces to all major commercial databases.

➢ GUI Programming – ▪ Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

➢ Scalable – ▪ Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

➢ It supports functional and structured programming methods as well as OOP.

➢ It can be used as a scripting language or can be compiled to bytecode for building large applications.

➢ It provides very high-level dynamic data types and supports dynamic type checking.

➢ It supports automatic garbage collection.

➢ It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

APPLICATIONS OF PYTHON:

These are some real-world Python applications:

- Web and Internet Development
- Desktop GUI Applications
- Science and Numeric
- Software Development
- Education
- Database Access
- Network Programming
- Games and 3D Graphics

OpenCV:

OpenCV is a cross-platform library using which we can develop realtime computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

Tkinter:

Tkinter is a popular Python library used for creating graphical user interfaces (GUIs). It provides a set of modules and classes that allow developers to build interactive and visually appealing applications with ease.

Tkinter is included with the standard Python distribution, so there is no need for any additional installations.

NumPy:

NumPy (Numerical Python) is a powerful Python library for scientific computing and numerical operations. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

4.5 METRICS

4.5.1 MSE VALUE:

Mean squared error (MSE) is a measure of the average squared difference between the actual and predicted values. It is a common metric used to evaluate the performance of regression models. The lower the MSE, the better the model is at predicting the actual values

MSE is calculated as follows:

$$\mathbf{MSE = \Sigma(y - y\hat{})^2 / n}$$

where:

- y is the actual value,
- y^ is the predicted value,
- n is the number of observations.

The following table shows the typical MSE values for different image quality levels.

Image Quality	MSE
Very poor	100
Poor	50
Fair	25
Good	12.5
Excellent	6.25

Table:3.1 MSE Value Image Quality

4.5.2 PSNR value:

Peak signal-to-noise ratio (PSNR) is a measure of the quality of a reconstructed image. It is calculated as the ratio of the maximum possible signal power to the power of the noise in the reconstructed image. A higher PSNR indicates a better quality image.

PSNR is calculated using the following formula:

PSNR = 10 * log10(MAX_VAL^2 /MSE)

where:

- MAX_VAL is the maximum possible value for a pixel in the image.
- MSE is the mean squared error between the original and reconstructed images.

The following table shows the typical PSNR values for different image quality levels.

Image Quality	PSNR (dB)
Very poor	20
Poor	30
Fair	40
Good	50
Excellent	60

Table:3.2 PSNR Value Image quality

4.5.3 RGB PIXEL VALUE:

The RGB value of a pixel represents the intensity of the red, green, and blue color channels that make up the pixel's color. Each color channel is typically represented by an 8-bit value ranging from 0 to 255, where 0 represents no intensity (no color) and 255 represents full intensity (maximum color). In an RGB image, each pixel is composed of three values:

the red (R), green (G), and blue (B) color channels. These values define the contribution of each color channel to the overall color of the pixel. For example, a pixel with an RGB value of (255, 0, 0) represents full intensity red, as the red channel is at its maximum value (255).

Fig 4.5.1: Pixel with RGB value of (255, 0, 0)

Fig 4.5.2: Pixel with RGB value of (0,255,0)

Fig 4.5.3: Pixel with RGB value of (0,0,255)

CHAPTER 5

5.CODING AND RESULT:

5.1 CODING:

Steg.py

```
import cv2

import tkinter as tk

import numpy as np

from tkinter.filedialog import askopenfilename, asksaveasfilename

from tkinter import messagebox

from PIL import Image, ImageTk

from lsb import LSB

from aes import AESCipher

root = tk. Tk()

#create a scrollbar and set it's Orientation to vertical

Scrollbar = tk. Scrollbar (root, orient="vertical")

root.title('Image')

root.geometry("300x350")

# Activity handle all user interaction like:

# 1. preview image

# 2. handle button click interaction

# 3. program lifecycle from start and exit

# 4. how User Interface looks like

class Activity:
```

```
# root window object
master = tk.Tk()
master.geometry("600x350")
# store image on cv2 object to be able to image manipulation
image = None
# store image on Imagetk object to be able to preview on window
imgPanel = None
keyInput = None
messageInput = None
path = "./dst.png"
def __init__(self):
  self.master.title('Image Steganography with Encryption')
  # use blank image when program started
  self.image = np.zeros(shape=[500, 500, 3], dtype=np.uint8)
  self.updateImage()
  # configure open button
  openBtn = tk.Button(self.master, text = 'Select Image', command = self.openImage)
  openBtn.pack()
  btnFrame = tk.Frame(self.master)
  btnFrame.pack()
  # configure encode button
  encodeBtn = tk.Button(btnFrame, text = 'Encode', command = self.encode)
  encodeBtn.pack(side = tk.LEFT)
  # configure decode button
```

```
decodeBtn = tk.Button(btnFrame, text = 'Decode', command = self.decode)
decodeBtn.pack(side = tk.LEFT)
# Add button to save key
saveKeyBtn = tk.Button(btnFrame, text = 'Save Key', command = self.saveKey)
saveKeyBtn.pack(side = tk.LEFT)
savebtnFrame = tk.Frame(self.master)
savebtnFrame.pack()
# configure save button
saveBtn = tk.Button(savebtnFrame, text = 'Save Image', command = self.saveImage)
saveBtn.pack(side = tk.LEFT)
# configure save value button
saveValueBtn = tk.Button(savebtnFrame, text = 'Save Value', command = self.saveValue)
saveValueBtn.pack(side = tk.LEFT)
# configure input box for key
tk.Label(self.master, text='Key').pack()
self.keyInput = tk.Entry(self.master)
self.keyInput.pack()
# configure input box for secret message
tk.Label(self.master, text='Secret Message').pack()
self.messageInput = tk.Text(self.master, height=10, width=60)
self.messageInput.pack()
# add button to clear input boxes
clearBtn = tk.Button(btnFrame, text = 'Clear', command = self.clearInputBoxes)
clearBtn.pack(side = tk.LEFT)
```

```
# clearInputBoxes function
def clearInputBoxes(self):
  self.keyInput.delete(0, tk.END)
  self.messageInput.delete(1.0, tk.END)
  # Clear the image
  self.image = np.zeros(shape=[200, 200, 3], dtype=np.uint8)
  self.updateImage()
# saveValue export int value for every color channel (RGB)
# on csv format
def saveValue(self):
  path = asksaveasfilename(title = "Select file")
  if path == '':
    return
  np.savetxt(path+'_blue.csv', self.image[:, :, 0], delimiter=',', fmt='%d')
  np.savetxt(path+'_green.csv', self.image[:, :, 1], delimiter=',', fmt='%d')
  np.savetxt(path+'_red.csv', self.image[:, :, 2], delimiter=',', fmt='%d')
  messagebox.showinfo("Info", "Saved")
# updateImage read image from cv2 object and preview on image window
def updateImage(self):
  image = cv2.cvtColor(self.image, cv2.COLOR_BGR2RGB)
  image = Image.fromarray(image)
  image = ImageTk.PhotoImage(image)
  if self.imgPanel == None:
    self.imgPanel = tk.Label(image=image)
```

```
    self.imgPanel.image = image
    self.imgPanel.pack(side="top", padx=10, pady=10)
  else:
    self.imgPanel.configure(image = image)
    self.imgPanel.image = image
def saveKey(self):
  path = asksaveasfilename(title = "Select file")
  if path == '':
    return
  with open(path, 'w') as f:
    f.write(self.keyInput.get())
  messagebox.showinfo("Info", "Saved")
# cipher create AESCipher object to encode message with inputed key as secret key
def cipher(self):
  key = self.keyInput.get()
  # key length must 16 character
  if len(key) != 16:
    messagebox.showwarning("Warning","Key must be 16 character")
    return
  return AESCipher(self.keyInput.get())
# encode encode message using AESCipher and embed cipher text to image
def encode(self):
  message = self.messageInput.get("1.0",'end-1c')
  # will convert only if the image is selected
```

```
  if self.image is None:
   messagebox.showwarning("Image not selected", "Please select an image.")
   return
  # message length will forced to be multiple of 16 by adding extra white space at the end
  if len(message)%16 != 0:
    message += (" " * (16-len(message)%16))
  cipher = self.cipher()
  if cipher == None:
    return
  cipherText = cipher.encrypt(message)
  obj = LSB(self.image)
  obj.embed(cipherText)
  self.messageInput.delete(1.0, tk.END)
  self.image = obj.image
  # preview image after cipher text is embedded
  self.updateImage()
  messagebox.showinfo("Info", "Encoded")
# decode extract cipher text from image and try decode it using provided secret key
def decode(self):
  cipher = self.cipher()
  if cipher == None:
    return
  obj = LSB(self.image)
  cipherText = obj.extract()
```

```
        msg = cipher.decrypt(cipherText
        # show decoded secret message to message input box
        self.messageInput.delete(1.0, tk.END)
        self.messageInput.insert(tk.INSERT, msg)
    # openImage ask user to select image
    def openImage(self):
        path = askopenfilename()
        if not isinstance(path, str):
            return
        self.image = cv2.imread(path)
        self.updateImage()
    # saveImage save image on png format
    def saveImage(self):
        path = asksaveasfilename(title = "Select file",filetypes=[("png files", "*.png")])
        if path == '':
            return
        if ".png" not in path:
            path = path + ".png"
        obj = LSB(self.image)
        obj.save(path)
        messagebox.showinfo("Info", "Saved")
    def startLoop(self):
        self.master.mainloop()
if __name__ == "__main__":
```

```
  app = Activity()
  app.startLoop()
```

lsb.py

```
import cv2
class AppError(BaseException):
  pass
def i2bin(i, l):
  actual = bin(i)[2:]
  if len(actual) > l:
    raise AppError("bit size is larger than expected.")
  while len(actual) < l:
    actual = "0"+actual
  return actual
def char2bin(c):
  return i2bin(ord(c), 8)
# LSB used to do image manipulation especially embedding secret message
# using LSB method
class LSB():
  # before embedding secret message on image, we need
  # to know which cell is used or will be used to store
  # secret message, to achive that, we will use 16 first cell
  # to store length, this value will be converted to binary
  # and no more that 16 bit which means max length of message is
  # 2^16 = 65536
```

```
MAX_BIT_LENGTH = 16
def __init__(self, img):
  self.size_x, self.size_y, self.size_channel = img.shape
  self.image = img
  # pointer used to refer which cell on image will be read or write
  self.cur_x = 0
  self.cur_y = 0
  self.cur_channel = 0
# move pointer to next cell
def next(self):
  if self.cur_channel != self.size_channel-1:
    self.cur_channel += 1
  else:
    self.cur_channel = 0
    if self.cur_y != self.size_y-1:
      self.cur_y += 1
    else:
      self.cur_y = 0
      if self.cur_x != self.size_x-1:
        self.cur_x += 1
      else:
        raise AppError("need larger image")
# replace last bit from value of cell referred by pointer
# and move pointer to next cell
def put_bit(self, bit):
```

```
  v = self.image[self.cur_x, self.cur_y][self.cur_channel]
  binaryV = bin(v)[2:]
  # replace last bit if different
  if binaryV[-1] != bit:
    binaryV = binaryV[:-1]+bit
  self.image[self.cur_x, self.cur_y][self.cur_channel] = int(binaryV,2)
  self.next()
# put_bits put array of bit to designated cell respectively
def put_bits(self, bits):
  for bit in bits:
    self.put_bit(bit)
# read_bit read last bit from value of cell referred by pointer
# return bit as result
def read_bit(self):
  v = self.image[self.cur_x, self.cur_y][self.cur_channel]
  return bin(v)[-1]
# read_bits read last bit for every cell referred by pointer until length
# return array of bit as result
def read_bits(self, length):
  bits = ""
  for _ in range(0, length):
    bits += self.read_bit()
    self.next()
  return bits
# embed embed text to image
```

```
def embed(self, text):
  # calculate text length and convert it to binary with length 16 bit
  text_length = i2bin(len(text), self.MAX_BIT_LENGTH)
  # put length to first 16 cell
  self.put_bits(text_length)
  # put every character on text to image
  for c in text:
    # convert character into binary with 8 length
    bits = char2bin(c)
    # put every bit to cell respectively
    self.put_bits(bits)
# extract extract text from image
def extract(self):
  # read 16 first cell as length of text that contained on image
  length = int(self.read_bits(self.MAX_BIT_LENGTH), 2)
  text = ""
  for _ in range(0, length):
    # read every 8 bit as a character
    c = int(self.read_bits(8), 2)
    # convert binary as a character
    text += chr(c)
  return text
# save save image to dstPath
def save(self, dstPath):
  cv2.imwrite(dstPath, self.image)
```

```
if __name__ == "__main__":
  # obj = LSB(cv2.imread('src.jpg'))
  # obj.embed("I'm sure that one day everything will happen, you will love me and will never let me go I want to be with you, I want to love your flaws, always willing to make you happy no matter what happens, I promise I am...")
  obj = LSB(cv2.imread('dst.png'))
  text = obj.extract()
  print(text)
```

aes.py

```
from Crypto.Cipher import AES
# AESCipher used to do text manipulation/cryptography
# key length : 16 character
# message length : multiple of 16
class AESCipher:
  def __init__(self, key):
    self.key = str.encode(key)
  # encrypt encript message in msg using key
  # and return ciphertext result
  def encrypt(self, msg):
    cipher = AES.new(self.key, AES.MODE_ECB)
    cipherText = cipher.encrypt(str.encode(msg))
    return cipherText.hex()
  # decrypt try decrypt cipher text in cipherText using key
```

```
  # and return secret message as result
  def decrypt(self, cipherText):
    decipher = AES.new(self.key, AES.MODE_ECB)
    msg = decipher.decrypt(bytes.fromhex(cipherText))
    return msg
if __name__ == "__main__":
  c = AESCipher("abcdefghijklmnop")
  secret = "SepertiYangBiasa"
  print(secret)
  cipherText = c.encrypt(secret)
  print(cipherText)
  secret = c.decrypt(cipherText)
  print(secret)
```

psnrmse.py

```
import cv2
import numpy as np
import math
def calculate_mse(image1, image2):
    squared_diff = np.square(image1 - image2)
    mse = np.mean(squared_diff)
    return mse
def calculate_psnr(image1, image2):
    mse = calculate_mse(image1, image2)
```

```
    max_pixel_value = 255.0  # Assuming 8-bit images
    psnr = 20 * math.log10(max_pixel_value / math.sqrt(mse))
    return psnr

# Load the images
image1 = cv2.imread("princess.jpg", cv2.IMREAD_GRAYSCALE)
image2 = cv2.imread("encoded princess.png", cv2.IMREAD_GRAYSCALE)

# Calculate MSE and PSNR
mse_value = calculate_mse(image1, image2)
psnr_value = calculate_psnr(image1, image2)
print("MSE:", mse_value)
print("PSNR:", psnr_value)
```

5.2 TESTING

Software testing can be stated as the process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases.

The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

Software testing can be divided into two steps:

1. **Verification**: it refers to the set of tasks that ensure that the software correctly implements a specific function.

2. **Validation**: it refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.

3. Manual Testing: Manual testing includes testing software manually, i.e., without using any automation tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing.

Testers use test plans, test cases, or test scenarios to test software to ensure the completeness of testing. Manual testing also includes exploratory testing, as testers explore the software to identify errors in it.

4. Automation Testing: Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves the automation of a manual process. Automation Testing is used to re-run the test scenarios quickly and repeatedly, that were performed manually in manual testing.

Apart from regression testing, automation testing is also used to test the application from a load, performance, and stress point of view. It increases the test coverage, improves accuracy, and saves time and money when compared to manual testing.

TYPES OF TESTING

- Unit Testing
- Integration Testing
- System Testing
- Functional Testing
- Acceptance Testing
- Smoke Testing
- Regression Testing
- Performance Testing
- Security Testing
- User Acceptance Testing

1.UNIT TESTING

Unit testing is a type of software testing that focuses on individual units or components of a software system. The purpose of unit testing is to validate that each unit of the software works as intended and meets the requirements. Unit testing is typically performed by developers, and it is performed early in the development process before the code is integrated and tested as a whole system.

Unit tests are automated and are run each time the code is changed to ensure that new code does not break existing functionality. Unit tests are designed to validate the smallest possible unit of code, such as a function or a method, and test it in isolation from the rest of the system. This allows developers to quickly identify and fix any issues early in the development process, improving the overall quality of the software and reducing the time required for later testing.

Unit Testing is a software testing technique by means of which individual units of software i.e. group of computer program modules, usage procedures, and operating procedures are tested to determine whether they are suitable for use or not. It is a testing method using which every independent module is tested to determine if there is an issue by the developer himself. It is correlated with the functional correctness of the independent modules. Unit Testing is defined as a type of software testing where individual components of a software are tested. Unit Testing of the software product is carried out during the development of an application. An individual component may be either an individual function or a procedure. Unit Testing is typically performed by the developer. In SDLC or V Model, Unit testing is the first level of testing done before integration testing. Unit testing is such a type of testing technique that is usually performed by developers. Although due to the reluctance of developers to test, quality assurance engineers also do unit testing.

2.INTEGRATION TESTING

Integration testing is the process of testing the interface between two software units or modules. It focuses on determining the correctness of the interface. The purpose of integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit tested, integration testing is performed.

Integration testing is a software testing technique that focuses on verifying the interactions and data exchange between different components or modules of a software application. The goal of integration testing is to identify any problems or bugs that arise when different components are combined and interact with each other. Integration testing is typically performed after unit testing and before system testing. It helps to identify and resolve integration issues early in the development cycle, reducing the risk of more severe and costly problems later on.

Integration testing can be done by picking module by module. This can be done so that there should be a proper sequence to be followed. And also if you don't want to miss out on any integration scenarios then you have to follow the proper sequence. Exposing the defects is the major focus of the integration testing and the time of interaction between the integrated units.

3.SYSTEM TESTING

System testing is a type of software testing that evaluates the overall functionality and performance of a complete and fully integrated software solution. It tests if the system meets the specified requirements and if it is suitable for delivery to the end-users. This type of testing is performed after the integration testing and before the acceptance testing.

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system.

The result of system testing is the observed behavior of a component or a system when it is tested. System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behavior of the system and also the expectations of the customer.

It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS). System Testing is basically performed by a testing team

that is independent of the development team that helps to test the quality of the system impartial. It has both functional and non-functional testing. System Testing is a black-box testing. System Testing is performed after the integration testing and before the acceptance testing.

4.FUNCTIONAL TESTING

Functional testing is basically defined as a type of testing that verifies that each function of the software application works in conformance with the requirement and specification. This testing is not concerned with the source code of the application. Each functionality of the software application is tested by providing appropriate test input, expecting the output, and comparing the actual output with the expected output.

This testing focuses on checking the user interface, APIs, database, security, client or server application, and functionality of the Application Under Test. Functional testing can be manual or automated.

5.ACCEPTANCE TESTING

Acceptance Testing is a method of software testing where a system is tested for acceptability. The major aim of this test is to evaluate the compliance of the system with the business requirements and assess whether it is acceptable for delivery or not. Standard

It is a formal testing according to user needs, requirements and business processes conducted to determine whether a system satisfies the acceptance criteria or not and to enable the users, customers or other authorized entities to determine whether to accept the system or not.

Acceptance Testing is the last phase of software testing performed after System Testing and before making the system available for actual use.

6.SMOKE TESTING

Smoke testing, also known as “Build Verification Testing” or “Build Acceptance Testing,” is a type of software testing that is typically performed at the beginning of the development process to ensure that the most critical functions of a software application are working correctly. It is used to quickly identify and fix any major issues with the software before more detailed testing is performed. The goal of smoke testing is to determine whether the build is stable enough to proceed with further testing.

Smoke Testing is a software testing method that determines whether the employed build is stable or not. It acts as a confirmation of whether the quality assurance team can proceed with further testing. Smoke tests are a minimum set of tests run on each build. Smoke testing is a process where the software build is deployed to a quality assurance environment and is verified to ensure the stability of the application. Smoke Testing is also known as Confidence Testing or Build Verification Testing.

In other words, we verify whether the important features are working and there are no showstoppers in the build that is under testing. It is a mini and quick regression test of major functionality. Smoke testing shows that the product is ready for testing. This helps in determining if the build is flawed so as to make any further testing a waste of time and resources.

7.REGRESSION TESTING

Regression Testing is the process of testing the modified parts of the code and the parts that might get affected due to the modifications to ensure that no new errors have been introduced in the software after the modifications have been made. Regression means return of something and in the software field, it refers to the return of a bug.

8.PERFOMANCE TESTING

Performance Testing is a type of software testing that ensures software applications to perform properly under their expected workload. It is a testing technique carried out to determine system performance in terms of sensitivity, reactivity and stability under a particular workload.

Performance testing is a type of software testing that focuses on evaluating the performance and scalability of a system or application. The goal of performance testing is to identify bottlenecks, measure system performance under various loads and conditions, and ensure that the system can handle the expected number of users or transactions.

9.SECURITY TESTING

Security Testing is a type of Software Testing that uncovers vulnerabilities of the system and determines that the data and resources of the system are protected from possible intruders. It

ensures that the software system and application are free from any threats or risks that can cause a loss.

Security testing of any system is focused on finding all possible loopholes and weaknesses of the system which might result in the loss of information or repute of the organization. Security testing is a type of software testing that focuses on evaluating the security of a system or application.

The goal of security testing is to identify vulnerabilities and potential threats, and to ensure that the system is protected against unauthorized access, data breaches, and other security-related issues.

10.USER ACCEPTANCE TESTING

User acceptance testing, a testing methodology where the clients/end users involved in testing the product to validate the product against their requirements. It is performed at client location at developer's site.

For industry such as medicine or aviation industry, contract and regulatory compliance testing and operational acceptance testing is also carried out as part of user acceptance testing.

UAT is context dependent and the UAT plans are prepared based on the requirements and NOT mandatory to execute all kinds of user acceptance tests and even coordinated and contributed by testing team.

5.3 RESULT

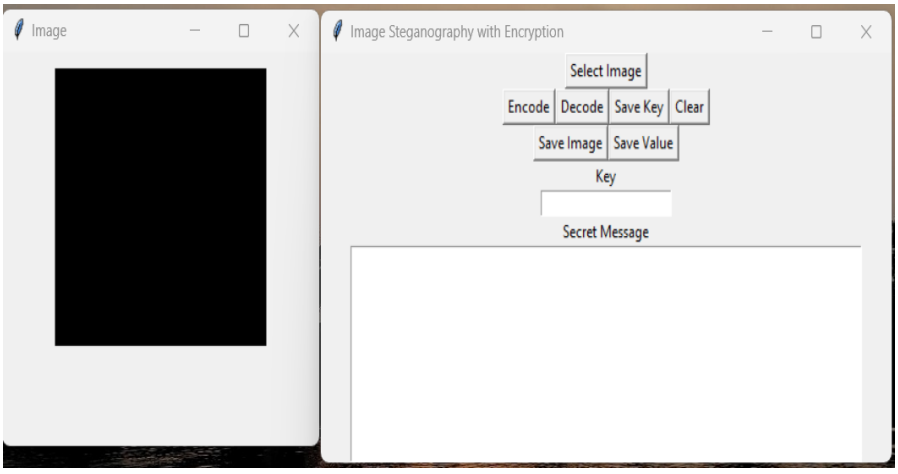

Fig: 5.1 User interface of the application

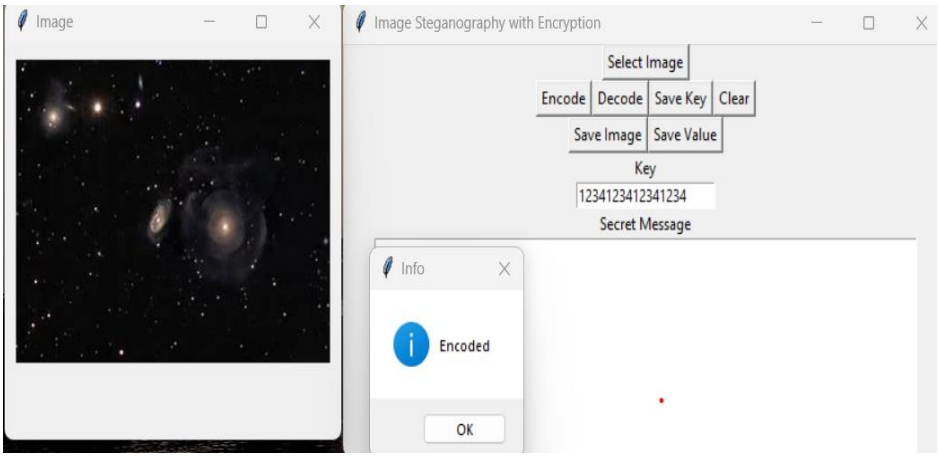

Fig:5.2 After the message is encoded in the cover image

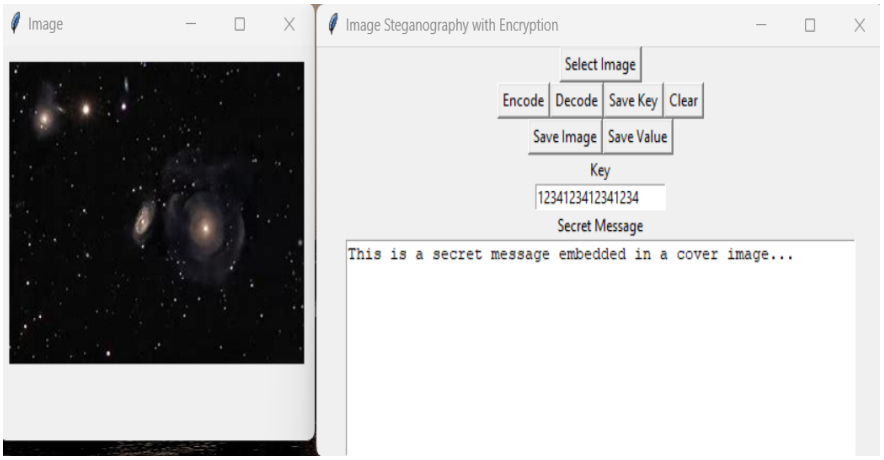

Fig:5.3 After the message is decoded

By comparing the various images with the same text length and also with varied text length and also same in case of secret image. The MSE values and PSNR values are measured to identify the quality if the image after steganolysis.

The following table shows the MSE and PSNR value of the various images.

Image(256x256)	PSNR RATIO(dB)	MSE
Bird	115.592967	27.50
Dog	40.3428344	32.07
Butterfly	47.7234370	31.34
Moon	89.6499633	28.60
Sky	1.67201232	45.89

Table:5.1 Text inside Image for 256-bit text length.

CHAPTER 6

6.1 CONCLUSION

In this proposed work, a secret data or message is embedded in an cover image with the help of LSB steganography technique and AES cryptography algorithm, which is an NIST standard symmetric algorithm with key length of 16 character that is 256 bit to provide more security for the secret data or information. Based on our implementation of this algorithm using Python in Visual Studio Code environment the secret data embedded in an cover image is successfully retrieved using an AES encryption key and the RGB pixel value of the cover image and the secret image is successfully extracted and saved.

6.2 FUTURE SCOPE

For further study one additional encryption algorithm like RSA can be added for additional security and the AES algorithm can be added with some other steganography techniques like audio or video steganography.

REFERENCES:

1. Shahid Rahman, Jamal Uddin, Habib Ullah Khan, Hameed Hussain, Ayaz Ali Khan, and Muhhamad Zakarya "A Novel Steganography Technique for Digital Images Using the Least Significant Bit Substitution Method", 2022.

2. Chiradeep Gupta and N V Subba Reddy "Enhancement of Security of Diffie-Hellman Key Exchange Protocol using RSA Cryptography", 2022.

3. U. A. Md. Ehsan Ali, Emran Ali, Md. Sohrawordi and Md. Nahid Sultan "A LSB Based Image Steganography Using Random Pixel and Bit Selection for High Payload", 2021.

4. Lalit Kumar Gupta, Aniket Singh, Abhishek Kushwaha, and Ashish Vishwakarma, "Analysis of Image Steganography Techniques for Different Image Format", 2021.

5. Arshiya S. Ansari, Mohammad S.Mohammadi, and Mohammad Tanvir Parvez" A Multiple - Format Steganography Algorithm for Color Images,"2020.

6. Jemima Dias, Dr. Ajit Danti "Image Steganography based Cryptography",2020.

7. Ali Ahmed and Abdelmotalib Ahmed "A Secure Image Steganography using LSB and Double XOR Operations" IJCSNS International Journal of Computer Science and Network Security, VOL.20 No.5, May 2020.

8. Arun Kumar Singh, "Steganography in Digital Images Using LSB Technique", Journal of Xidian University 2020.

9. Mr. Jawwad A R. Kazi, Mr. Gunjan N. Kiratkar, Ms. Sonali S. Ghogale, Prof. Atiya R. Kazi "A novel approach to Steganography using pixel -based algorithm in image hiding", 2020 International Conference on Computer Communication and Informatics,2020.

10. Sakshi Audhi, Maruska Mascarenhas "Secure Mechanism for Communication Using Image Steganography" 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT).

11. NIST, "Advanced Encryption Standard," http://csrc.nist.gov/ publications/fips/fips197/fips - 197.pdf, 2001 (accessed March 24, 2017).

12. Z. Y. Al -Omari and A. T. Al - Taani, "Secure LSB steganography for coloured images using character -colour mapping," 2017 8th International Conference on Information and Communication Systems (ICICS), 2017, pp. 104 -110.

13. Beenish Siddiqui, Sudhir Goswami “A Survey on Image Steganography Using LSB Substitution Technique” 2017 International Research Journal of Engineering and Technology (IRJET).

14. Aman Arora, Manish Pratap Singh, Prateek Thakral, Naveen Jarwal proposed a paper by name “Image steganography using Enhanced LSB substitution technique”,2016 fourth international conference on Parallel, Distributed and Grid Computing.

15. Arun Kumar Singh, Juhi Singh, Dr. Harsh Vikram Singh “Steganography in Images Using LSB Technique”, 2015 International Journal of Latest Trends in Engineering and Technology (IJLTET).