Project Report on

Image Steganography

Submitted by

Om Jogani (CE050) (21CEUOD011)

&

Jay Chauhan (CE027) (21CEUBD003)

For term work of

B.Tech CE Semester: VI
Subject: (CE-621) System Design Practice

Under the guidance of

Prof. Ashish K. Gor
(Assistant Professor Department of Computer Engineering)

Faculty of Technology

Department of Computer Engineering

Dharmsinh Desai University

Department of Computer Engineering

Dharmsinh Desai University

Certificate

This is to certify that the practical / term work carried out in the subject of System Design Practice and recorded in this journal is the bonafide work of

Om Jogani (CE050) (21CEUOD011)
Jay Chauhan (CE027) (21CEUBD003)

of B.Tech semester VI in the branch of Computer Engineering during the academic year 2022-23.

Prof. Ashish K. Gor (Assistant Professor, Department of Computer Engineering)

Table of Contents

Abstract

Steganography is the art of hiding secret message in such a way that no one, apart from sender and intended recipient suspects existence of message, a form of security through obscurity. The goal of steganography is to hide messages inside other harmless data in a way that does not allow any enemy to even detect that there is data hidden.

In this project, the most famous steganographic approach **Least Significant Bit (LSB)** is used where LSB refers to the last or the right-most bit in a binary number. This approach replaces some LSBs of the cover image with the secret data bits of the hidden message. The proposed algorithm includes 3 different phases. First, the secret message is encrypted based upon the user input and the secret key provided by the user and for encryption we have used AES (Advanced Encryption Standard) algorithm to encrypt the data. Second, the encrypted data is hidden inside an image (provided by the user) for that the LSB algorithm is used. Third, in the decryption phase the user inputs the stego-image (image with the hidden message) and enters the key and as an output user will get the extracted text decrypted by the key.

1. Introduction

In today's world, people communicate over the Internet and share private information. In order to block data from intruders and hackers, this secret information should be protected through a secure technique. The secret data can be hacked for the purpose of copyright violation, for tampering it or can be illegally accessed without the knowledge of owner. Due to these reasons there is a need of hiding secret data inside different types of digital data such that owner can prove copyright ownership, identify attempts to tamper with sensitive data and to embed annotations.

Steganography is a process that involves hiding important information (message) inside other carrier (cover) data to protect the message from unauthorized users. Our Human Visual System is not able to recognize the small changes that occur in cover data after hiding the secret data. Thus, the stego data will be seen by the Human Visual System (HVS) as one piece of data. The message and the cover data can be of any format such as text, audio, image, and video. However, there are many steganalytical detectors that detect a secret message from an unsecure steganography algorithm. Hence, researchers are working on developing secure steganography algorithms that are protected from both attackers and steganalysis detectors. A good steganography system should have high embedding payload and high embedding efficiency. First, the embedding payload is defined as the amount of secret information that is going to be embedded inside the cover data. The algorithm has a high embedding payload if it has a large capacity for the secret message. The embedding efficiency includes the stego visual quality, security, and robustness against attackers. Second, both a low modification rate and good quality of the cover data lead to a high embedding efficiency.

1.1 History

To understand steganography, we should first take a look at its predecessor: cryptography. Cryptography is the art of protecting information by transforming it into an unreadable format, called cipher text. To decipher this unreadable format, a secret key is required.

- Cryptography was found as far back as 1900 BC. In an ancient Egyptian scribe.
- From 500 – 600 B.C. ATBASH, a reversed alphabet simple solution cipher, was used by Hebrew.
- From 50 - 60 B.C. a simple substitution with the normal alphabet in government communications was used by Julius Caesar.
- In today's scenario Quantum Cryptography is being used that combines physics and cryptography to produce a new cryptosystem that is very difficult to cryptanalyze without having the knowledge of the attempted and failed intrusion. Through the long history of cryptography, steganography was developed and flourished on its own.
- The use of steganography dates back to 440 BC.
    i. The Greek ruler Histaeus gave birth to steganography by: **shaving the head of a slave**, tattooing the message on the slaves scalp, waiting for the growth of hair in order to hide the secret message, and sending the slave on his way to deliver the message.

ii. And Demaratus, who sent a warning about attack to Greece by writing it directly on the wooden backing of a wax tablet before applying its beeswax surface.

- In 1600s, Sir Francis Bacon used a variation in type face to carry each bit of the encoding.
- During the American Revolutionary War **Invisible Inks** were used by both the British and American forces.
- During World War II the Germans introduced microdots. The microdots were complete documents, pictures, and plans reduced in size to the size of a period and attached to common paperwork. Null ciphers were also used to pass secret messages. Null ciphers are unencrypted messages with real messages embedded in the current text..
- During the 1980's Margareth Thatcher, then Prime Minister in UK, became so irritated about press leaks of cabinet documents, that she had the word processors programmed to encode the identity of the writer in the word spacing, thus being able to trace the disloyal ministers.

1.2 Definition of Steganography

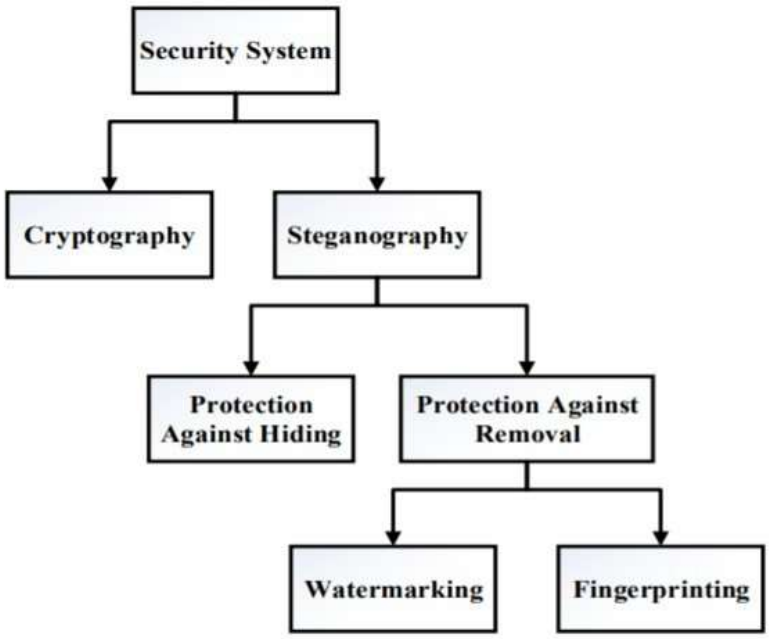
Steganography comes from the Greek steganos (covered or secret) and graphy (writing or drawing).

Or

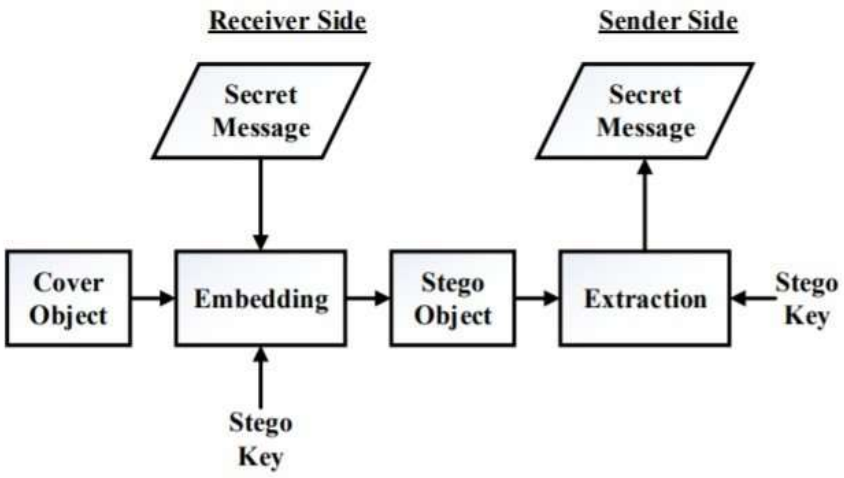
The art and science of concealing message in the form of text, image, video or file within another text, image video or file is called steganography.

1.3 Overview of Steganography

Often steganography is confused with cryptography because both have objective of protecting important information. The difference between the two is that while cryptography involves coding the message using an encryption key and sending it as cipher text, steganography involves hiding the intended message within a seemingly harmless cover. To exploit human perception is the work of steganography. Human senses are not trained to look for files that have information inside it. The most common use of steganography is to hide a file inside another file. Steganography can be done by either providing protection against detection or by providing protection against removal. Protection against removal can be carried out by either watermarking or fingerprinting.



Basic Model of Steganography:



It consists of:

- **Cover Object:** It is the input image, video file, audio file or a text file in which concealment of secret data is to be performed.
- **Stego-Object:** After the concealment of secret data into the cover medium, the cover object becomes the stego-object.
- **Embedding:** Embedding is the process of making a stego-object from a cover object. Or we can define it as the process of concealment of secret message into some digital medium.
- **Extraction:** This is the reverse process of embedding. In this process, the conceded message is recovered from stego-object to read it.
- **Message:** It is the secret information that is to be embedded in the cover object for safe transmission.

Image Steganography is the art of writing hidden messages inside images, in such a way that no one apart from the sender and intended recipient realizes the existence of a hidden message. Steganography uses repeating portions of the Image files to embed the secret message. There are many techniques for hiding data within Image/Video.

Important Steganographic Measures

- Mean Squared Error(MSE)
- Peak Signal Noise Ratio(PSNR)
- Time Complexity
- Universal Image Quality Index(UIQI)
- Structural Similarity Index Metric (SSIM)
- Bit Error Rate(BER)
- Similarity Function(SF)

1.4 Taxonomy/Classification of Steganography

- **Based on carrier:** text, image, audio, video
- **Based on message format:** text, image, audio, video
- **Based on domain:** Spatial domain, Frequency domain
- **Based on methods used:** Spatial Domain Methods (LSB, Pseudorandom LSB Encoding), Frequency Domain Methods (DCT, DFT, DWT), Spread Spectrum Method, Statistical Method, Distortion Method, Visual Cryptography, Cover Generation Method

1.5 Applications of Steganography

- Confidential digital communication and secret data storing.
- Media Database systems.
- Military and intelligence agencies.
- Protection of tampering data by criminals.
- Law enforcement and counter intelligence agencies.
- Online Free Speech on the net, including anonymous remailers and Web proxies.
- Digital elections and digital cash.
- Marketers use email forgery techniques.
- Finger prints and forensic.
- Telecommunication.
- Medical images.

1.6 Problem Definition

The earlier techniques consists of linguistic or language forms of hidden writing. The later techniques, such as invisible try to hide messages physically. One disadvantage of linguistic steganography is that users must have a good knowledge of linguistry. In recent years, everything is rapidly trending towards digitization. And with the development of the internet technology, digital media can be transmitted smoothly over the network. Therefore, messages can be secretly carried through digital media by using the steganography techniques, and then be transmitted through the internet rapidly.

Many different carrier file formats can be used for steganography, but image and video are the most popular beeause of their high embedding capacity. For hiding secret information in an image or a video, there exists a large variety of steganography techniques. Some techniques are more complex than others and all of them have their respective strong and weak points.

Many of the existing steganography algorithms are designed without taking into account the security aspect of the message hidden inside the image. As a result, these algorithms are built with the lack of security, robustness, and imperceptibility. Moreover, most techniques lack intelligent processing of the cover, that is, the whole cover is equally utilized in the hiding process without following any adaptive approach for **selecting the best regions for embedding data**. Additionally, the whole cover is involved in the hiding process which affects the visual quality of the resultant stego-object and also affects the quality of the extracted data. Image steganography in **LSB** is very fast and easy to implement in comparison to other methods of image Steganography. The output image has very slight difference to the input image. Instead of embedding the message in only the LSB, we can embed the message in last two LSBs, thus embedding even large messages. This method forms the basics of many other complex algorithms Instead of embedding the message in only the LSB, we can embed the message in last two LSBs, thus embedding even large messages. Also similar system directly hides the data inside the image without providing enough security to the data, for that we have used **AES** to encrypt the data and then to hide it inside the image rather hiding the data directly into the image.

1.7 Purpose

Recent steganography algorithms are based on spatial domain which can simply be plagiarized by anyone. Moreover till now in most of the work embedding is just a simple substitution method; imperceptibility, while security and capacity is still the issue.

1.8 Scope and Objective

This project is developed for hiding information in an image file. The main goal is to design an algorithm that is more efficient and secure along with high embedding payload at a less cost. The critical task is to select an appropriate algorithm to provide the security of the data and particular positions to embed secret data on the sender side as well as successfully retrieve back the exact secret data on the receiver side in order to fulfill the goal. The proposed work supports images of .jpg and .png format as cover object. Selective portions of the image are considered for hiding different secret message. Therefore images sizes should be equivalent or large enough to hide the entire user data.

We also created basic GUI using flet framework. This GUI uses the code we developed for hiding and extracting messages to and from image. In this tool we developed user can also find out different performance measure for images.

1.9 Challenges/Requirements

- To design an easy, efficient and simple embedding_and extracting algorithm.
- To design a highly imperceptible and highly secure algorithm.
- It should have high embedding payload (capacity) as well as high embedding efficiency (good quality of stego object, robust).

2. LITERATURE REVIEW

1.1 Image Steganography

I. Spatial Domain Based

These techniques use the pixel grey levels and their color values directly for encoding the message bits. These techniques are some of the simplest schemes in terms of embedding and extraction complexity. The major drawback of these methods is amount of additive noise that creeps in the image which directly affects the Peak Signal to Noise Ratio and the statistical properties of the image. Moreover these embedding algorithms are applicable mainly to lossless image-compression schemes like TIFF images.

For lossy compression schemes like JPEG, some of the message bits get lost during the compression step.

The most common algorithm belonging to this class of techniques is the Least Significant Bit (LSB) replacement technique in which the least significant bit of the binary representation of the pixel grey levels is used to represent the message bit. This kind of embedding leads to an addition of a noise of 0:5p on average in the pixels of the image where p is the embedding rate in bits/pixel. This kind of embedding also leads to an asymmetry and a grouping in the pixel grey values (0,1);(2,3);. . . (254,255). To overcome this undesirable asymmetry, the decision of changing the least significant bit is

randomized i.e. if the message bit does not match the pixel bit, then pixel bit is either increased or decreased by 1. This technique is popularly known as LSB Matching. It can be observed that even this kind of embedding adds a noise of 0:5p on average. To further reduce the noise, have suggested the use of a binary function of two cover pixels to embed the data bits. The embedding is performed using a pair of pixels as a unit, where the LSB of the first pixel carries one bit of information, and a function of the two pixel values carries another bit of information. It has been shown that embedding in this fashion reduces the embedding noise introduced in the cover signal.

II. Frequency Domain Based

These techniques try to encode message bits in the transform domain coefficients of the image. Data embedding performed in the transform domain is widely used for robust watermarking. Similar techniques can also realize large capacity embedding for steganography. Candidate transforms include discrete cosine Transform (DCT), discrete wavelet transform

(DWT), and discrete Fourier transform (DFT). By being embedded in the transform domain, the hidden data resides in more robust areas, spread across the entire image, and provides better resistance against signal processing. For example, we can perform a block DCT and, depending on payload and robustness requirements, choose one or more components in each block to form a new data group that, in turn, is pseudo randomly scrambled and undergoes a second-layer transformation. Modification is then carried out on the double transform domain coefficients using various schemes. These

techniques have high embedding and extraction complexity. Because of the robustness properties of transform domain embedding, these techniques are generally more applicable to the "Watermarking" aspect of data hiding. Many steganographic techniques in these domain have been inspired from their watermarking counterparts.

Discrete Cosine Transform (DCT):

DCT is a well-known method which is utilized in many applications such as image and video compression. The DCT separates the signal into low, middle, and high frequency regions. The DCT is closely related to the discrete Fourier transform (DFT). It is a separable linear transformation; that is, the 2D-DCT is equivalent to a 1D-DCT performed along a single dimension followed by a 1D-DCT in the other dimension. For an input video frame, A, of resolution M x N the DCT frequency coefficients for the

transformed frame, B, and the inverse DCT coefficients of the reconstructed frame are calculated according to the following equations, respectively:

$$B_{pq} = \propto_p \propto_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N} \quad (1)$$

$$A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \propto_p \propto_q B_{pq} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N} \quad (2)$$

Where $$\propto_p = \begin{cases} \frac{1}{\sqrt{M}}, p = 0 \\ \sqrt{\frac{2}{M}}, 1 \le p \le M-1 \end{cases}$$

And

$$\propto_q = \begin{cases} \frac{1}{\sqrt{N}}, q = 0 \\ \sqrt{\frac{2}{N}}, 1 \le q \le N-1 \end{cases}$$

A (m, n) is the pixel value in row m and column y of the frame A, and B (p, q) is the coefficient in row p and column q of the 2D-DCT matrix. Each of low, middle, and high frequency coefficients were used as cover data to embed the encoded secret message.

Discrete Wavelet Transform

The Discrete Wavelet Transform can identify portions of cover image where secret data could be effectively hidden. DWT splits information into its high and low frequency components. The high frequency part of the signal contain details about the edge components, whereas the low frequency part contains most of the signal information of the image which is again split into higher and lower frequency parts. For each level of decomposition in two dimensional applications, first DWT is performed in the vertical direction followed by horizontal direction.

Comparison between the above two approaches:

Domain	Methods	Advantage	Disadvantage	Hiding Rate
Spatial domain	Low bit encoding [1][6]	High embedding rate, simple and easy	Noticeable to human ear, less robust to human ear	16kbps [6]
	Echo hiding [1][2][6]	Recovers easily from lossy data compression algorithms	Low capacity and low security	50bps [6]
Transform domain	Spread spectrum [1][6]	More robust	More vulnerable to time scale modifications	20bps [6]
	Discrete Wavelet Transform [1][6][7]	High embedding capacity	Inaccurate data retrieval	70kbps [6]
	Tone Insertion [1][2][6]	Imperceptibility of embedded data	Poor Transparency	250bps [6]
	Phase Coding [2][6][8]	Robust against signal distortion	Low capacity	333bps [6]

3. PROPOSED APPROACH

Image steganography can be defined as the art and science of embedding secret data in images. Data hiding in images has gained practical significance nowadays due to the huge technological advancement of multimedia systems. The greatest advantage of image is the large amount of data that can be hidden inside it. Therefore any small but otherwise noticeable distortion might go unobserved by humans because of the continuous flow of information.

3.1 Introduction to proposed Design

In the proposed system:

- Image is used as cover.
- Secret message can be in the form of text.
- Cover Image format used: .jpg or .png

The basic steps of the proposed system are divided into 3 phases:

- First, the secret message is encrypted by the **AES** algorithm with the key provided by the user.
- Second, the encrypted message is hidden in the image for that **LSB** algorithm is used.

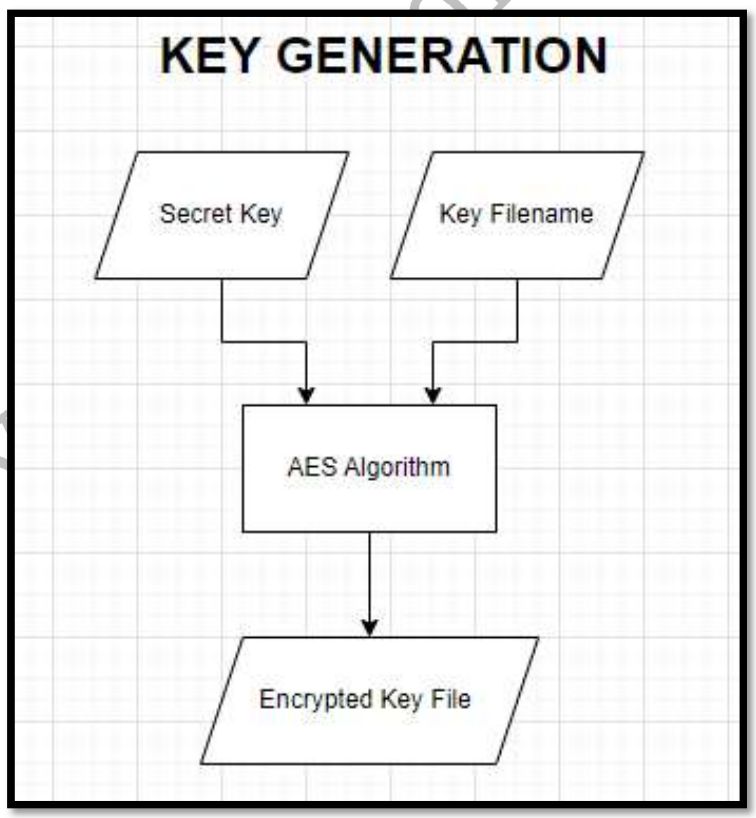
- Third, in the decryption phase the user inputs the stego-image (image with the hidden message) and enters the key and as an output user will get the extracted text decrypted by the key.

3.2 Steganographic Technique

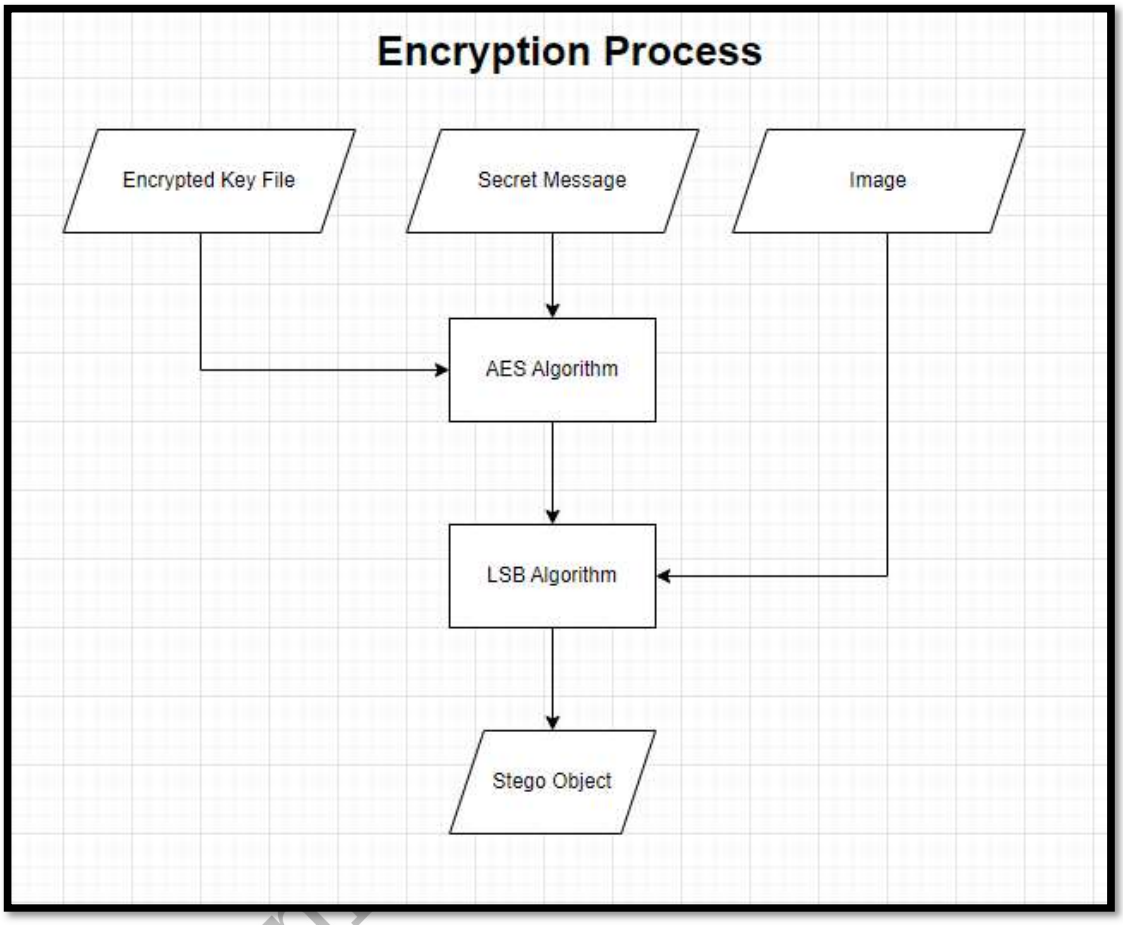
- Instead of using frequency domain DWT technique which has fine visual quality and high embedding payload, spatial domain based technique has been used in this project.
- Moreover, spatial domain based techniques make it more robust against the attacks compared to frequency domain based techniques.

3.3 Flow Chart Representation:

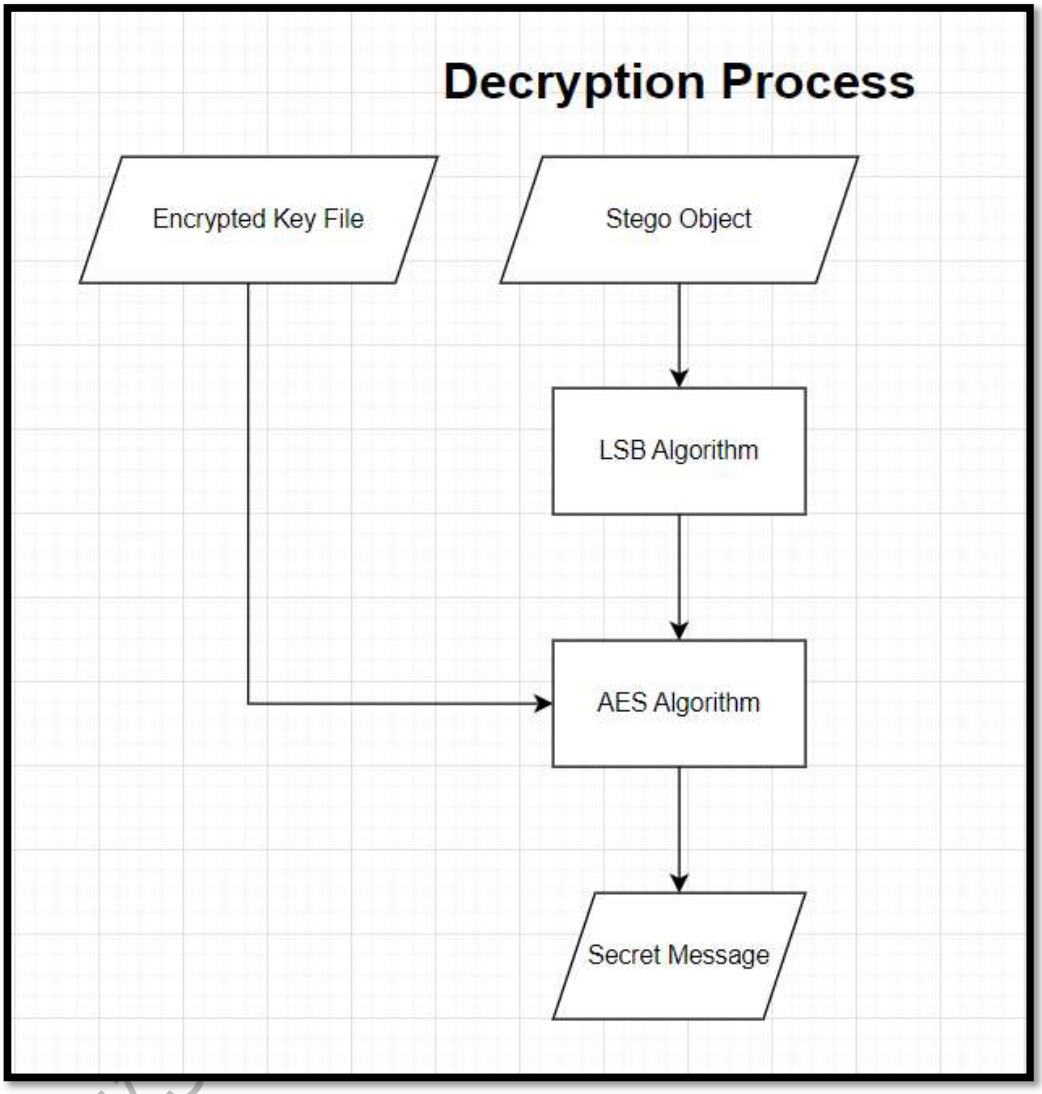
- **Key Generation:**



- **Encryption:**



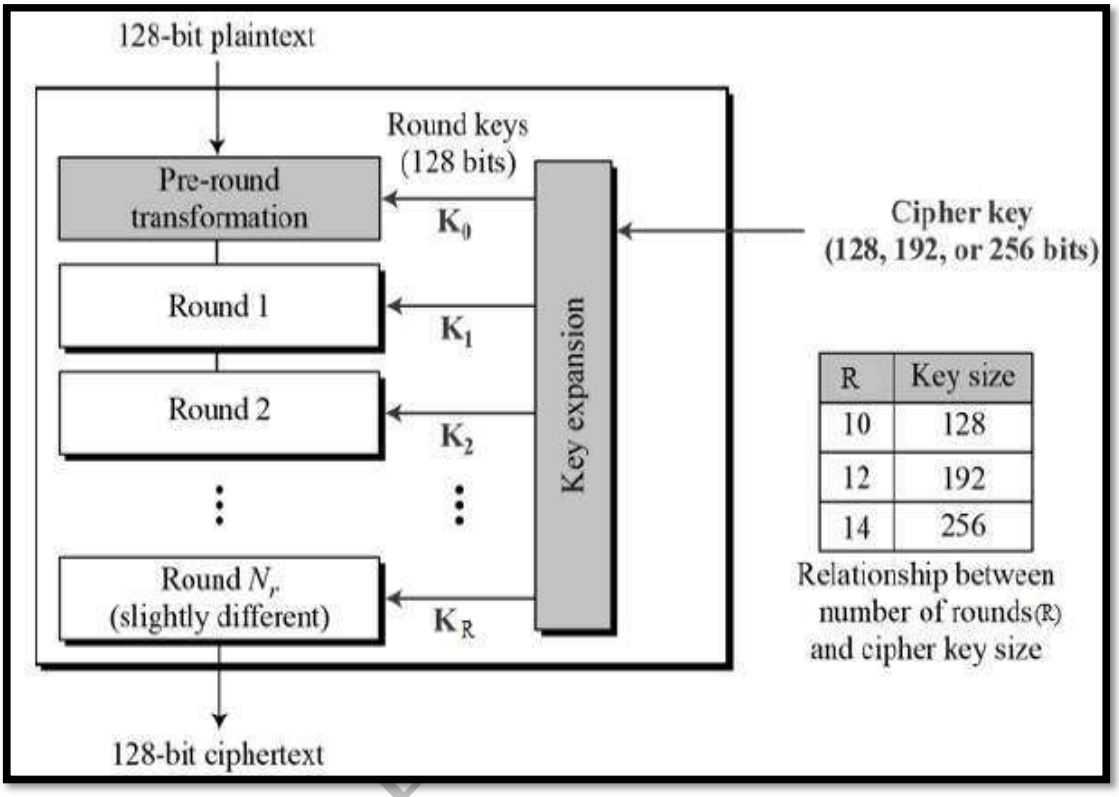
- **Decryption:**



3.4 Pre-processing Secret Data

In the proposed work the secret data is text and it is pre-processed before embedding phase.

In pre-processing secret data is converted into an encrypted text which is not human readable by using **AES** algorithm whose working is depicted below.



R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

3.5 Performance metrics

Table 4. 1 Way of checking performance metrics

Performance Metrics	How to check
Visual Quality	**PSNR:** It is measured by calculating PSNR and MSE. PSNR is a nonperceptual objective metric measuring the difference between the original and distorted images. $PSNR = 10.\log_{10}(\frac{MAX_I{}^2}{MSE})$ Where, MAXI represents maximum value of pixel of the image. In the images with pixel having 8 bits per sample, its value is 255. The MSE stands for cumulative squared error between the stego image and the original image. $MSE = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{h}[C(i,j,k) - S(i,j,k)]^2}{m \times n \times h}$ C and S refer to the cover image and stego image respectively. m and n defines as image resolutions and h indicated the R, G and B color channels. (k=1,2 & 3)
Time Complexity	It can be calculated by noting down the time taken for embedding process and extraction process.

Structural Similarity Index Metric (SSIM)	SSIM is an objective image quality metric and is superior to traditional measures such as MSE and PSNR. PSNR estimates the perceived errors, whereas SSIM considers image degradation as perceived change in structural information. Structural information is the idea that the pixels have strong interdependencies especially when they are spatially close. These dependencies carry important information about the structure of the objects in the visual scene. The SSIM Index quality assessment index is based on the computation of three terms, namely the luminance term, the contrast term and the structural term. The overall index is a multiplicative combination of the three terms. $SSIM(x,y)=[l(x,y)]^{\alpha}[c(x,y)]^{\beta}[s(x,y)]^{\gamma}$ where, $$l(x,y)=\frac{2\mu_x\mu_y+C_1}{\mu_x^2+\mu_y^2+C_1},$$ $$c(x,y)=\frac{2\sigma_x\sigma_y+C_2}{\sigma_x^2+\sigma_y^2+C_2},$$ $$s(x,y)=\frac{\sigma_{xy}+C_3}{\sigma_x\sigma_y+C_3}$$ where μx, μy, σx,σy, and σxy are the local means, standard deviations, and cross-covariance for images *x, y*. If α = β = γ = 1 and C3 = C2/2 the index simplifies to: $$SSIM(x,y)=\frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)}$$

4. IMPLEMENTATION DETAILS

4.1 Functions Implemented

4.1.1 Key Generation and Comparison:

```
def createkeyfile(self,keyfile,keyvalue):
      f = open(keyfile+".txt", "w+")
      f.write(keyvalue)
      f.close()
      self.encrypt_file(keyfilename+".txt")
      return "Key Generated Successfully."

def matchkey(self,keyfile,key):
  self.decrypt_file(keyfile+".txt.enc")
  with open(keyfile+".txt", 'rb') as fo:
        decryptedkey = fo.read()
  enc.encrypt_file(decryptkeyfile+".txt")
  if(decryptedkey.decode() != key):
    return 0
  return 1
```

4.1.2 Encryption:

```
def encrypt(self, message, key, key_size=256):
    message = self.pad(message)
    iv = Random.new().read(AES.block_size)
    cipher = AES.new(key, AES.MODE_CBC, iv)
    return iv + cipher.encrypt(message)

def encrypt_file(self, file_name):
    with open(file_name, 'rb') as fo:
        plaintext = fo.read()
    enc = self.encrypt(plaintext, self.key)
    with open(file_name + ".enc", 'wb') as fo:
        fo.write(enc)
    os.remove(file_name)
```

4.1.3 Encoding the secret message into the image (Creating a Stego object):

```
def Encode(src, message, dest):
    img = Image.open(src, 'r')
    width, height = img.size
    array = np.array(list(img.getdata()))

    if img.mode == 'RGB':
        n = 3
        total_pixels = array.size//n
    elif img.mode == 'RGBA':
        n = 4
        total_pixels = array.size//n
    message += "$t3g0"
    b_message = ''.join([format(ord(i), "08b") for i in message])
    req_pixels = len(b_message)

    if req_pixels > total_pixels:
        print("ERROR: Need larger file size")

    else:
        index=0
        for p in range(total_pixels):
            for q in range(0, 3):
                if index < req_pixels:
                    array[p][q] = int(bin(array[p][q])[2:9] + b_message[index], 2)
                    index += 1

        array=array.reshape(height, width, n)
        enc_img = Image.fromarray(array.astype('uint8'), img.mode)
        enc_img.save(dest)
        print("Image Encoded Successfully")
```

4.1.4 Key Comparison:

```
def matchkey(self,keyfile,key):
    self.decrypt_file(keyfile+".txt.enc")
    with open(keyfile+".txt", 'rb') as fo:
            decryptedkey = fo.read()
    enc.encrypt_file(decryptkeyfile+".txt")
    if(decryptedkey.decode() != key):
      return 0
    return 1
```

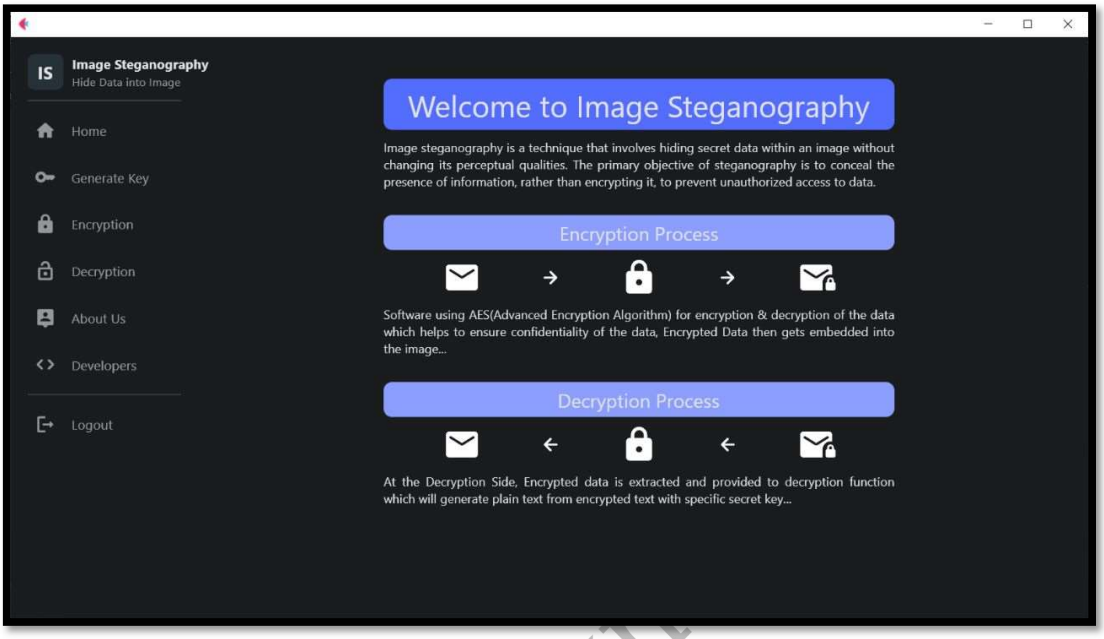
4.1.5 Decoding the Stego object to extract the secret message:

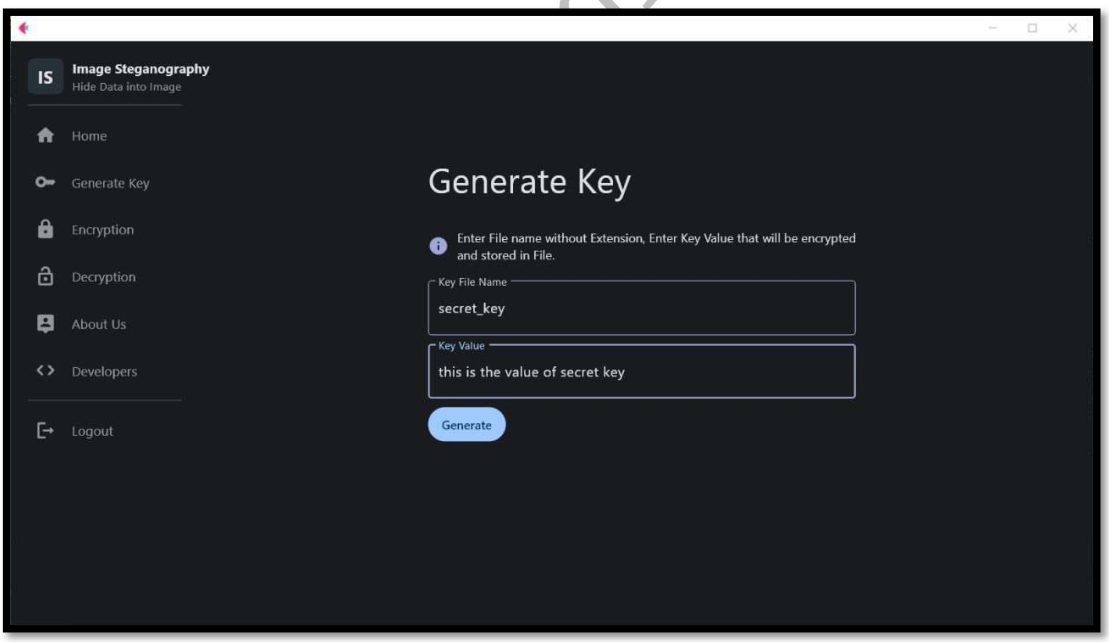
```
def Decode(src):

    img = Image.open(src, 'r')
    array = np.array(list(img.getdata()))

    if img.mode == 'RGB':
        n = 3
    elif img.mode == 'RGBA':
        n = 4
        total_pixels = array.size//n

    hidden_bits = ""
    for p in range(total_pixels):
        for q in range(0, 3):
            hidden_bits += (bin(array[p][q])[2:][-1])

    hidden_bits = [hidden_bits[i:i+8] for i in range(0, len(hidden_bits), 8)]

    message = ""
    for i in range(len(hidden_bits)):
        if message[-5:] == "$t3g0":
            break
        else:
            message += chr(int(hidden_bits[i], 2))
    if "$t3g0" in message:
        print("Hidden Message:", message[:-5])
    else:
        print("No Hidden Message Found")
```

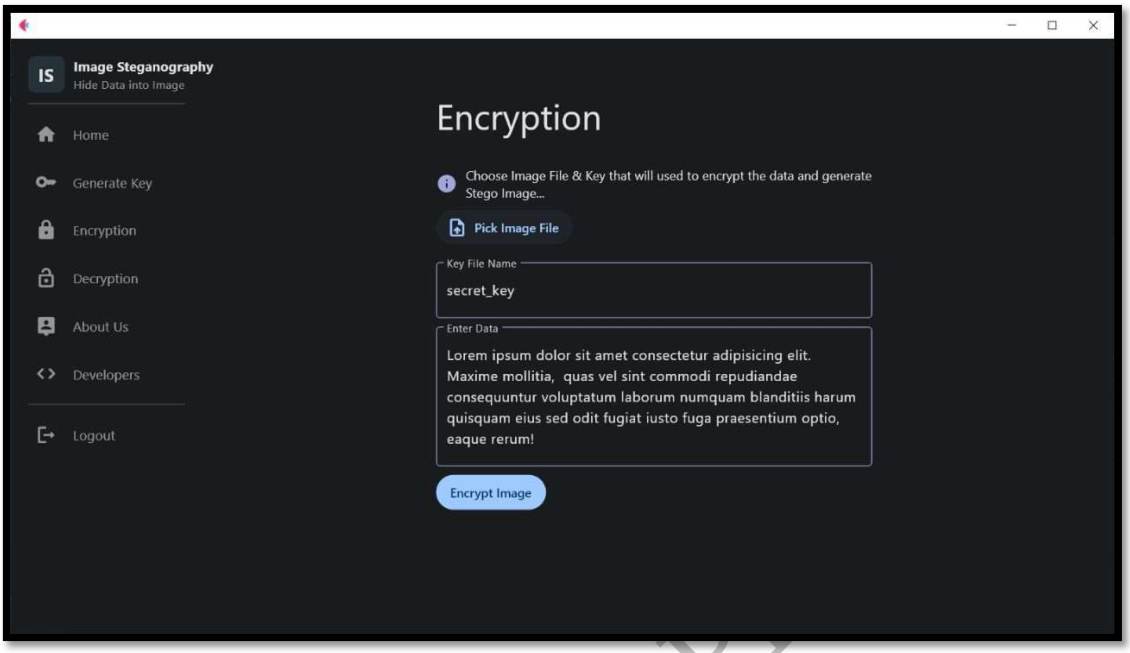
4.1.6 Decryption:

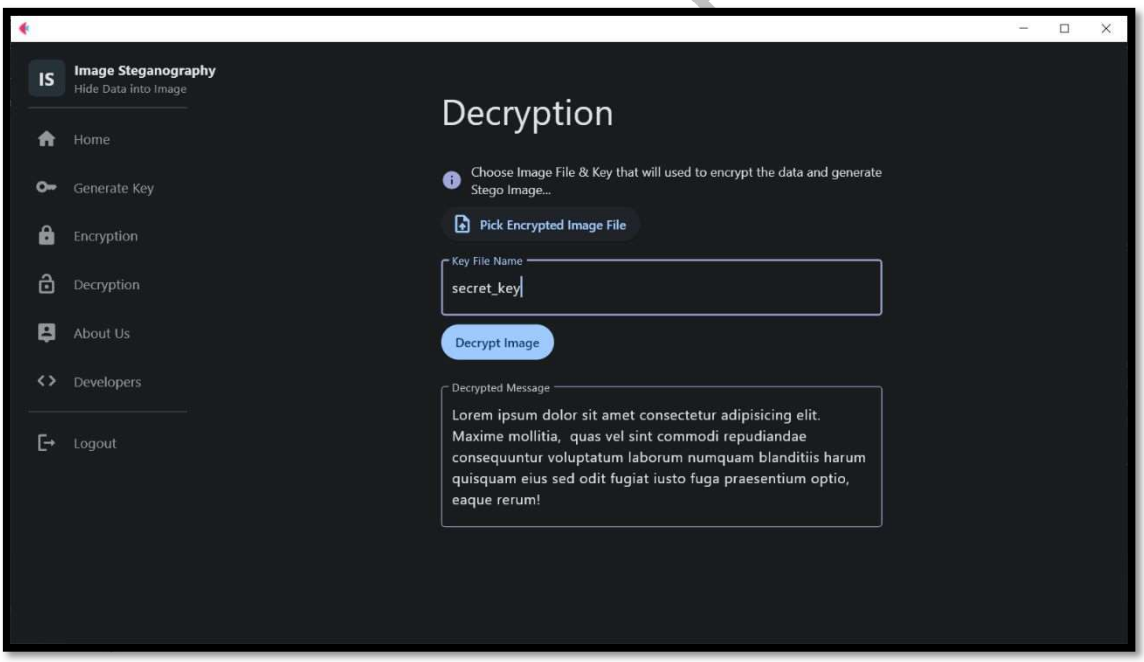
```
def decrypt(self, ciphertext, key):
    iv = ciphertext[:AES.block_size]
    cipher = AES.new(key, AES.MODE_CBC, iv)
    plaintext = cipher.decrypt(ciphertext[AES.block_size:])
    return plaintext.rstrip(b"\0")

def decrypt_file(self, file_name):
    with open(file_name, 'rb') as fo:
        ciphertext = fo.read()
    dec = self.decrypt(ciphertext, self.key)
    with open(file_name[:-4], 'wb') as fo:
        fo.write(dec)
    os.remove(file_name)
```

4.2.1 Output:


IS
Image Steganography
Hide Data into Image
Home
Generate Key
Encryption
Decryption
About Us
Developers
Logout
Welcome to Image Steganography
Image steganography is a technique that involves hiding secret data within an image without changing its perceptual qualities. The primary objective of steganography is to conceal the presence of information, rather than encrypting it, to prevent unauthorized access to data.
Encryption Process
Software using AES(Advanced Encryption Algorithm) for encryption & decryption of the data which helps to ensure confidentiality of the data, Encrypted Data then gets embedded into the image...
Decryption Process
At the Decryption Side, Encrypted data is extracted and provided to decryption function which will generate plain text from encrypted text with specific secret key...


IS
Image Steganography
Hide Data into Image
Home
Generate Key
Encryption
Decryption
About Us
Developers
Logout
Generate Key
Enter File name without Extension, Enter Key Value that will be encrypted and stored in File.
Key File Name
secret_key
Key Value
this is the value of secret key
Generate


IS
Image Steganography
Hide Data into Image
Home
Generate Key
Encryption
Decryption
About Us
Developers
Logout
Encryption
Choose Image File & Key that will used to encrypt the data and generate Stego Image...
Pick Image File
Key File Name
secret_key
Enter Data
Lorem ipsum dolor sit amet consectetur adipisicing elit. Maxime mollitia, quas vel sint commodi repudiandae consequuntur voluptatum laborum numquam blanditiis harum quisquam eius sed odit fugiat iusto fuga praesentium optio, eaque rerum!
Encrypt Image


IS
Image Steganography
Hide Data into Image
Home
Generate Key
Encryption
Decryption
About Us
Developers
Logout
Decryption
Choose Image File & Key that will used to encrypt the data and generate Stego Image...
Pick Encrypted Image File
Key File Name
secret_key
Decrypt Image
Decrypted Message
Lorem ipsum dolor sit amet consectetur adipisicing elit. Maxime mollitia, quas vel sint commodi repudiandae consequuntur voluptatum laborum numquam blanditiis harum quisquam eius sed odit fugiat iusto fuga praesentium optio, eaque rerum!

5. DATASETS, TOOLS and TECHNOLOGY

5.1 Implementation Platform Details

Proposed approach is Implemented and Tested on platform given below.

5.1.1 Hardware Specification

Processor: Intel Core i5-10265U CPU 3.6GHz

RAM: 8 GB

Google Collab: 8GB Ram (Used for Testing of Some Images)

5.1.2 Software Specification

OS: Windows 10

System Type: 64 bit OS

Front End: Python

5.2 Tools and Technology

5.2.1 Technologies

1. Python
2. OpenCV
3. Scikit-Learn

4. Flet
5. Numpy

5.2.2 Tools

1. Google Colab
2. Pycharm

5.3 Dataset Design

5.3.1 Secret Message Digest

Secret message: These messages are in the text form of various sizes. Demonstration for various input secret message and their cipher text along with its file size is listed below.

Key used to encrypt all these message: <u>sdpprojectsubmission</u>

Msg no.	Secret Message	Cipher Text	Size
1	"I have a dream that one day every valley shall be exalted, every hill and mountain shall be made low." - Martin Luther King Jr.	铆 譶 뾜□뷀 燚 뾜□뷀□뷀 羽 뾜□뷀뾜ᆟᅻ□뷀뾜□뷀뾜□뷀뾜□뷀 腬 뾜□⁝ □뷀□뷀□뷀 燮 뾜 纽 뾜 纽 뾜□뷀□뷀뾜□뷀뾜 柰 뾜ⴱ 瓔 춣□뷀욇□뷀뾜□뷀ҳ⇟ 쫥糒젯纹뾜□뷀뾜□뷀뾜 ₛ커□뷀□뷀□뷀□릛 蓺뾜偰뾜□뷀뾜嫣ㄟ庰::뾜□뷀□뷀 Ĭc□뷀쩡悠뾜ﻻ□뷀□뷀□뷀□뷀 滈 □뷀□뷀뾜 鰔□뾜□뷀□뷀뾜□뷀뾜ˁ□뷀뾜 撴	286 bytes
2	This file is created to hide text. Tomorrow is Republic day for india. How ARE 12345 you. password1: ` 1 2 3 4 5 6 7 8 9 0 - = password2: ~!@ # $ % ^ & * () _ + password3: ABCDEFGHIJKLMNOPQRSTUV WXYZ.	♁踜榜□□□□▯ ᆱ□·찍緷멯潡擫鮫掕葫ᅩ 猠輸굢□媩婉□뉘瞮寸錳KB□遱▼倖縿 đ芎 □ 吻 燐 世 墒 곘□ 暘 殂 □ · 픎 躉 聞 □ؿ坳 쬐□□땂觥□땞딞萬꾋쳇□糖寱言햪□涬鷘 □嵘□렐ൈ衋뎅撜□릥媲搟閅22맥磏□琁顣 鑣긦昆舯홭뎁箯龀熣砎虉ｹ図鄘기骀□浪 □ؚ놁綮浝V穌肚ᴷ蓛缅훿딘搢搱玪Ȧ□¡□馀	320 bytes

	password4: abcdefghijklmnopqrstuvwxyz. password5: aVrt@56Hm Please change the size of the cover image.	[illegible]	
3	Alan Turing, in full Alan Mathison Turing, (born June 23, 1912, London, England-died June 7, 1954, Wilmslow, Cheshire), British mathematician and logician who made major contributions to mathematics, cryptanalysis, logic, philosophy, and mathematical biology and also to the new areas later named computer science, cognitive science, artificial intelligence, and artificial life. The son of a civil servant, Turing was educated at a top private school. He entered the University of Cambridge to study mathematics in 1931. After graduating in 1934, he was elected to a fellowship at King's College (his college since 1931) in recognition of his research in probability theory. In 1936 Turing's seminal paper On Computable Numbers, with an Application to the Entscheidungsproblem [Decision Problem] was recommended for publication by the American mathematical logician Alonzo Church, who had himself just published	[illegible]	3.68 KB

	a paper that reached the same conclusion as Turing's, although by a different method. Turing's method (but not so much Church's) had profound significance for the emerging science of computing. Later that year Turing moved to Princeton University to study for a Ph.D. in mathematical logic under Church's direction (completed in 1938). What mathematicians called an effective' method for solving a problem was simply one that could be carried by a human mathematical clerk working by rote. In Turing's time, those rote-workers were in fact called Computers, and human computers carried out some aspects of the work later done by electronic computers. The Entscheidungsproblem sought an effective method for solving the fundamental mathematical problem of determining exactly which mathematical statements are provable within a given formal mathematical system and which are not. A method for determining this is called a decision method. In 1936 Turing and Church independently showed that, in general, the EntscheidungsproblemResults/ problem has	[illegible]	

no resolution, proving that no consistent formal system of arithmetic has an effective decision method. In fact, Turing and Church showed that even some purely logical systems, considerably weaker than

arithmetic, have no effective decision method. This result and others- notably mathematician-logician Kurt Godel's incompleteness results-

dashed the hopes, held by some mathematicians, of discovering a formal system that would reduce the whole of mathematics to methods that (human) computers could carry out. It was in the course of his work on the Entscheidungsproblem that Turing invented the universal Turing machine, an abstract computing machine that encapsulates the fundamental logical principles of the digital computer.

An important step in Turing's argument about the Entscheidungsproblem was the claim, now called the Church-Turing thesis, that everything humanly computable can also be computed by the universal Turing machine. The claim is important because it marks out the limits of human computation. Church in his work used instead

	the thesis that all human-computable functions are identical to what he called lambda-definable functions (functions on the positive integers whose values can be calculated by a process of repeated substitution). Turing showed in 1936 that Church's thesis was equivalent to his own, by proving that every lambda-definable function is computable by the universal Turing machine and vice versa. In a review of Turing's work, Church acknowledged the superiority of Turing's formulation of the thesis over his own (which made no reference to computing machinery), saying that the concept of computability by a Turing machine 'has the advantage of making the identification with effectiveness evident immediately.'	□л 罷 ၈ 셥□∋彬□跡샅甬응簽赵萄仫ჱ料 拆¯¯칤□휅↞朋∀ 纁⑸鸽□畬盞鎧쌉憶큯 ꌸ妑ᇜ엤□몄ᄋᄅ뒸ᐩ잿낟憎夎□□ ϛ鹎媃 □娷贻 ĩ□峻죙砼□楇駣煸 Ά□□闫□覺褉㊤ ꔤ 鬓 ꩠ∫潌 □ ꔤ 秖 □□ 甕 훙읆ᄈᄈ□ 軏 칡□□□▲茾딸餕좊꾻걸瓸넓핃鉗□ρ□□芥 犭滚伽삼퀌ꆞ嗷↔樀敎ᄆ딕□肯鏈▪廹 霞昃헐痒岗膝伋괎쿒□」□□貅日□↙선輔 鳗腸僴ꈰ洳□"Oǎ□□잘렜紌□檝찓×앪퇴讓 闥ı□·评洍沄鮀玎렌□을霸裩孰□樀베縻燹 □젊鳴离突쩽 6 뉙□활惬め ᆗ냈□젰□慦 □□獮痲뾈릚辺傳怛精蚍 ƙ 숦ᄋ凰 η 킷词 □ 객랏셍眨□□埒蒕鏅⊳ 됀닏ᄀ姉妞濕巘 ᄊ≥傑뜻 d 黛□	

5.3.2 Cover Image Dataset:

Comparison between original image and stego image:

Normal Image	Stego Image
	

Size: 6.09 KB	**Size: 6.80 KB**
Hidden Data: "I have a dream that one day every valley shall be exalted, every hill and mountain shall be made low." - Martin Luther King Jr.	
PSNR value is 41.12227560374623 unit MSE value is 5.021723546828736 unit SSIM value is 0.9865740886675972 unit	
	
Size: 80.6 KB	**Size: 80.6 KB**
Hidden Data: This file is created to hide text. Tomorrow is Republic day for india. How ARE 12345 you. password1: ` 1 2 3 4 5 6 7 8 9 0 - = password2: ~!@ # $ % ^ & * () _ + password3: ABCDEFGHIJKLMNOPQRSTUVWXYZ. password4: abcdefghijklmnopqrstuvwxyz. password5: aVrt@56Hm Please change the size of the cover image.	
PSNR value is 38.29010742879392 unit MSE value is 9.639835390946502 unit SSIM value is 0.9812002670331539 unit	

	
Size: 240 KB	Size: 130 KB
Hidden Data: Alan Turing, in full Alan Mathison Turing, (born June 23, 1912, London, England-died June 7, 1954, Wilmslow, Cheshire), British mathematician and logician who made major contributions to mathematics, cryptanalysis, logic, philosophy, and mathematical biology and also to the new areas later named computer science, cognitive science, artificial intelligence, and artificial life. The son of a civil servant, Turing was educated at a top private school. He entered the University of Cambridge to study mathematics in 1931. After graduating in 1934, he was elected to a fellowship at King's College (his college since 1931) in recognition of his research in probability theory. In 1936 Turing's seminal paper On Computable Numbers, with an Application to the Entscheidungsproblem [Decision Problem] was recommended for publication by the American mathematical logician Alonzo Church, who had himself just published a paper that reached the same conclusion as Turing's, although by a different method. Turing's method (but not so much Church's) had profound significance for the emerging science of computing. Later that year Turing moved to Princeton University to study for a Ph.D. in mathematical logic under Church's direction (completed in 1938). What mathematicians called an effective' method for solving a problem was simply one that could be carried by a human mathematical clerk working by rote. In Turing's time, those rote-workers were in fact called Computers, and human computers carried out some aspects of the work later done by electronic computers. The Entscheidungsproblem sought an effective method for solving the	

fundamental mathematical problem of determining exactly which mathematical statements are provable within a given formal mathematical system and which are not. A method for determining this is called a decision method. In 1936 Turing and Church independently showed that, in general, the EntscheidungsproblemResults/ problem has no resolution, proving that no consistent formal system of arithmetic has an effective decision method. In fact, Turing and Church showed that even some purely logical systems, considerably weaker than

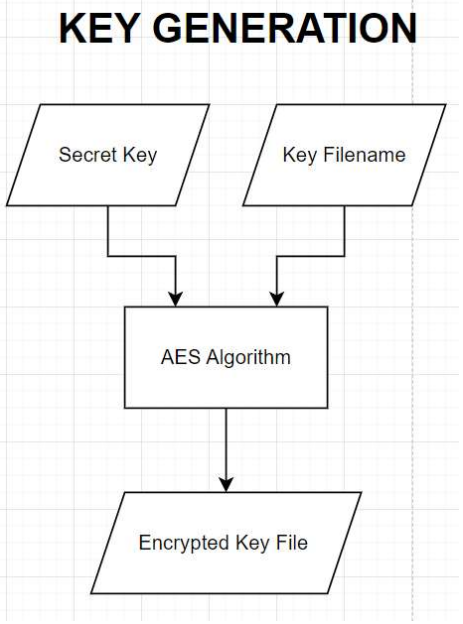
arithmetic, have no effective decision method. This result and others- notably mathematician-logician Kurt Godel's incompleteness results-

dashed the hopes, held by some mathematicians, of discovering a formal system that would reduce the whole of mathematics to methods that (human) computers could carry out. It was in the course of his work on the Entscheidungsproblem that Turing invented the universal Turing machine, an abstract computing machine that encapsulates the fundamental logical principles of the digital computer.
An important step in Turing's argument about the Entscheidungsproblem was the claim, now called the Church-Turing thesis, that everything humanly computable can also be computed by the universal Turing machine. The claim is important because it marks out the limits of human computation. Church in his work used instead the thesis that all human-computable functions are identical to what he called lambda-definable functions (functions on the positive integers whose values can be calculated by a process of repeated substitution). Turing showed in 1936 that Church's thesis was equivalent to his own, by proving that every lambda-definable function is computable by the universal Turing machine and vice versa. In a review of Turing's work, Church acknowledged the superiority of Turing's formulation of the thesis over his own (which made no reference to computing machinery), saying that the concept of computability by a Turing machine 'has the advantage of making the identification with effectiveness evident immediately.'

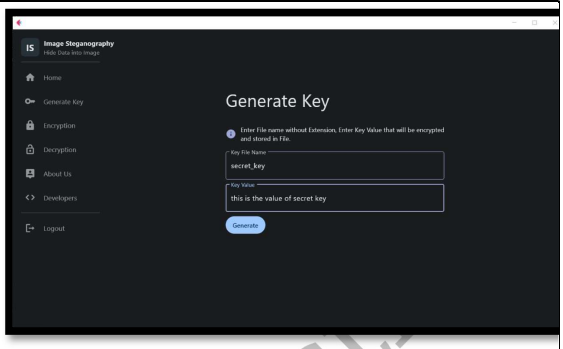
PSNR value is 34.6949220832601 unit

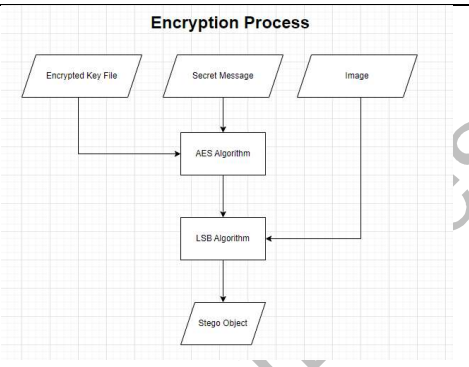
MSE value is 22.059118441358024 unit SSIM value is 0.9443287898005377 unit	
 Size: 3.61KB	 Size: 4.43 KB
PSNR value is 45.672582137542975 unit MSE value is 1.7612546272340086 unit SSIM value is 0.9980686635519616 unit	
 Size:22.2 KB	 Size: 19.2 KB
PSNR value is 65.672582137542975 unit MSE value is 1.712546272340086 unit SSIM value is 0.9980686635519616 unit	

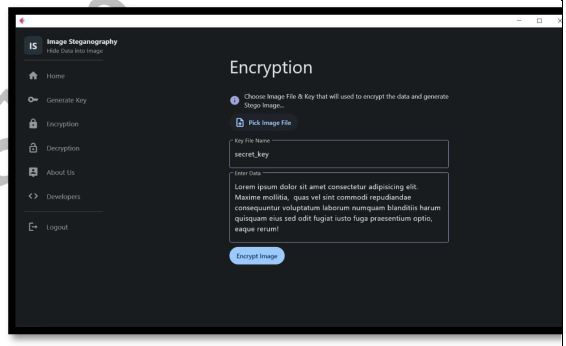
6. TESTING RESULTS

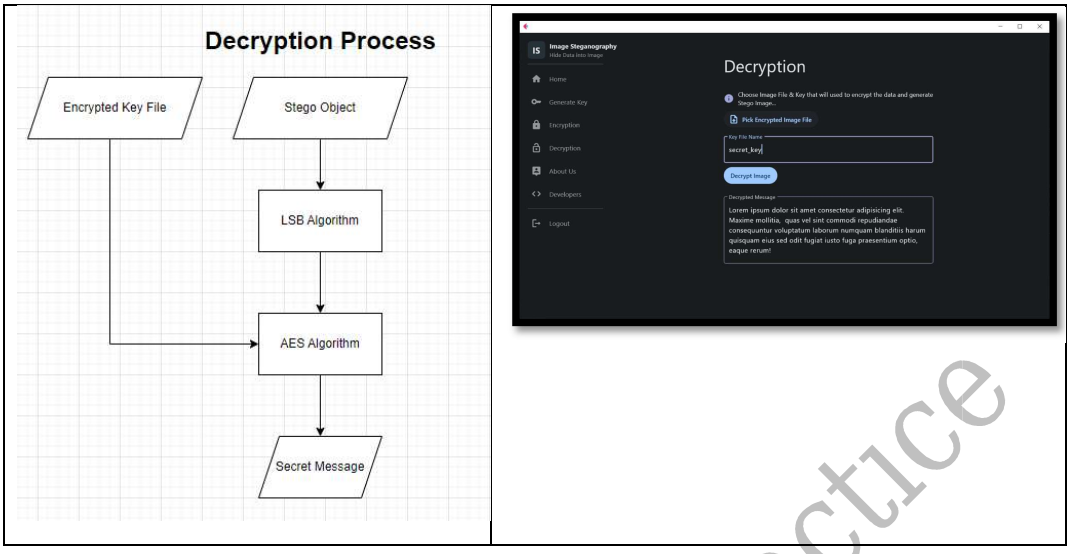


Output Screen









7. CONCLUSION AND FUTURE WORK

7.1 Conclusion

A new and efficient and secure steganographic method for embedding secret messages into images without producing any major changes has been proposed.

Security - The proposed approach uses AES algorithm to convert the readable secret message into non-human readable format

8. BIBLIOGRAPHY

[1] B. Forouzan, “Cryptography & Network Security”, Tata McGrawHill Publication, Special Indian Edition 2007.

[2] https://flet.dev/docs/

[3] https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[4] https://ieeexplore.ieee.org/document/9335027

[5] https://www.mygreatlearning.com/blog/image-steganography-explained/