**ABSTRACT**

Steganography is the art of hiding the fact that communication is taking place, by hiding information in other information. Many different carrier file formats can be used, but digital images are the most popular because of their frequency on the internet. For hiding secret information in images, there exists a large variety of steganography techniques some are more complex than others and all of them have respective strong and weak points. Different applications may require absolute invisibility of the secret information, while others require a large secret message to be hidden. This project report intends to give an overview of image steganography, its uses and techniques. It also attempts to identify the requirements of a good steganography algorithm and briefly reflects on which steganographic techniques are more suitable for which applications .Steganography is the practice of hiding private or sensitive information within something that appears to be nothing out to the usual. Steganography is often confused with cryptology because the two are similar in the way that they both are used to protect important information. The difference between two is that steganography involves hiding information so it appears that no information is hidden at all. If a person or persons views the object that the information is hidden inside of he or she will have no idea that there is any hidden information, therefore the person will not attempt to decrypt the information.  
What steganography essentially does is exploit human perception, human senses are not trained to look for files that have information inside of them, although this software is available that can do what is called Steganography. The most common use of steganography is to hide a file inside another file.

## **CONTENTS**

**CHAPTER NO.** **TITLE** **PAGE NO.**

**ABSTRACT**  **1**

**CONTENTS** **2**

**LIST OF FIGURES** **4**

**LIST OF SYMBOLS** **5**

**1** **INTRODUCTION** 7

1.1 OBJECTIVE 7

1.2 PROJECT DESCRIPTION 7

1.3 METHODOLOGY 8

**2** **SYSTEM ANALYSIS** 10

2.1 EXISTING SYSTEM 10

2.2 PROPOSED SYSTEM 10

3  **LITERATURE SURVEY** 12

3.1.1 REVERSIBLE DATA HIDING 12

3.1.2 SPATIAL DOMAIN 13

3.1.3 DIFFERENCE EXPANSION 14

3.1.4 HISTOGRAM SHIFTING / 18

MODIFICATION

3.1.5 PIXEL VALUE ORDERING 25

4.  **SYSTEM DESIGN**

4.1 ARCHITECTURE DIAGRAM

4.2 UML DIAGRAM

4.2.1 SEQUENCE DIAGRAM

4.2.2 USECASE DIAGRAM

4.2.3 CLASS DIAGRAM 10

4.2.4 STATE DIAGRAM 11

4.2.5 DATA FLOW DIAGRAM 12

4.  **SYSTEM DESIGN 26**

4.1 ARCHITECTURE DIAGRAM 26

4.2 UML DIAGRAM 27

4.2.1 SEQUENCE DIAGRAM 27

4.2.2 USECASE DIAGRAM 28

4.2.3 CLASS DIAGRAM 29

4.2.4 ACTIVITY DIAGRAM 30

4.2.5 STATE DIAGRAM 31

5.  **PROJECT OVERVIEW 33**

5.1 PROJECT DESCRIPTION 33

5.2 REQUIREMENT SPECIFICATION 35

5.2.1 HARDWARE REQUIREMENTS 35

5.2.2 SOFTWARE REQUIREMENTS 35

5.2.2.1 PYTHON 35

5.3 FEASABILITY STUDY 36

5.4 OBJECTIVE 36

5.5 SCOPE 36

5.6 IMAGE DEFINITION 37

5.7 IMAGE COMPRESSION 37

5.8 LEAST SIGNIFICANT BIT 38

5.9 DETECTION/ATTACKS 40

5.10 BENEFITS / DRAWBACKS 40

6. **CODING AND TESTING 42**

**7. APPENDICES** 46

**8. CONCLUSION** 54

8.1 CONCLUSION 55

8.2 FUTURE ENHANCEMENT 56

8.3 REFERENCES 57

**LIST OF FIGURES**

### FIGURE NO. TITLE PAGE NO

4.1.1 System Architecture 8

* + 1. Sequence Diagram 9
    2. Use case Diagram 10
    3. Class Diagram 11
    4. Activity Diagram 12
    5. State Diagram 13
    6. Data Flow Diagram 15

**LIST OF SYMBOLS**

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **SYMBOL NAME** | **SYMBOL** |
| 1. | USECASE |  |
| 2. | ACTOR |  |
| 3. | LIST OF PROCESS |  |
| 4. | START |  |
| 5. | DECISION |  |
| 6. | UNIDIRECTIONAL |  |
| 7.    8. | ENTITY SET    STOP |  |

## 

**CHAPTER 1**

**INTRODUCTION**

* 1. **OBJECTIVE**

This project is developed for hiding information in any image file. The scope of the project is implementation of steganography tools for hiding information includes any type of information file and image files and the path where the user wants to save Image and extruded file.

* 1. **PROJECT DESCRIPTION**

In recent years of information era, multimedia content in digital form is widely used in many applications. At the same time, however, a growing array of security concerns have been revealed. For example, improving smart editing tools may also encourage misuse, illegitimate duplicating and distribution, outright fraud and misapplication, which could bring down the activities of the creator or owner of the multimedia effort. This creates a strong need for systems like copyright protection and authentication of integrity that can effectively cope up with multimedia security and privacy. In the advanced digital communication era, ensuring the privacy of individuals becomes a challenging task. Various methods like encryption and steganography have been developed for the protection of personal privacy. In encryption, one can observe that communication is happening. However, steganography or data hiding is not observable. For many years, with many communications having taken place over long distances, intervened by modern technology and alertness to interception related issues, never guarantees a secure communication means. In an expansive mean, a concealed channel for covert communication is the path for one to communicate with others.

**1.3 METHODOLOGY**

General Methodology General method consists of three phases: secret sharing phase, steganography phase and data extraction phase. The secret sharing and steganography phases are used for encoding the secret message, and the data extraction phase is used for decoding and revealing the secret image. In secret sharing phase, the secret image is distributed among the shares. In steganography, the shares are hidden in the cover images. Then, the obtained stego images are given to the participants. In the embedding process, sequences

of distance differences and binary values that are generated in the preprocessing are utilized. Each distance between two random pixel channels will embed one message bit by adjusting the distance to the closest value in the distance difference sequence whose binary value is identical to the message bit. In the deembedding phase, the stego-key is used to generate the same distance differences and binary sequences as those used in the embedding process. For each random selected pixel channels, the closest distance between them is computed to find the hidden message from the corresponding binary value. The algorithm Encoding:

1. Read image files

2. Compress secret file

3. Encrypt secret file

4. Calculate Capacity of covert file

5. If the cover capacity is not sufficient goto 1

6. else repeat

i. Generate binary sequence based on the key

ii. Generate random distances sequence based on the key

iii. Get random pixel

iv. Get 2 random channels for this pixel

v. Compute distance between those channels

vi. Match with distance with minimum error and has the same message bit.

**CHAPTER 2**

**SYSTEM ANALYSIS**

### 2.1.EXISTING SYSTEM

### The existing methods for hiding information give good results only in case of information gets hidden successfully. LSB is the most popular Steganography technique. It hides the secret message in the RGB image based on it its binary coding. LSB algorithm is used to hide the secret messages by using algorithm.

### 2.2.PROPOSED SYSTEM

### The original aims of the paper are to introduce a technique for hiding a text file, which techniques hide a secret text file inside an image file, and the modified image must be similar to the original image, in other words the changes that happen on the modified image mustn’t be visible, or the human eye would be unable to notice it The project application loads 24-bit BMP, GIF, and JPG image format, embed data into them using Sunflower system and saves the images. Encryption can be used before embedding the data to provide robustness. Finally the application can also extract data that was previously embedded. The application runs in a user friend Windows environment where the user can view the image, before and after the embedding. valued image is a common information hiding method that utilizes the characteristic of the human visions insensitivity to small changes in the image. This simple LSB embedding approach is easy for computation, and a large amount of data can be embedded without great quality loss. The more LSBs are used for embedding, the more distorted result will be produced. Not all pixels in an image can tolerate equal amounts of changes without causing notice to an observer. The largest number of LSBs whose gray values can be changed without producing a perceptible artifact in each pixel is different. Changes of the gray values of pixels in smooth areas in images are more easily noticed by human eyes. In the embedding method we propose, we simply divide the cover image into a number of non-overlapping two-pixel blocks. Each block is categorized according to the difference of the gray values of the two pixels in the block. A small difference value indicates that the block is in a smooth area and a large one indicates that it is in an edged area. The pixels in edged areas may, as mentioned previously, tolerate larger changes of pixel values than those in the smooth areas. So, in the proposed method we embed more data in edged areas than in the smooth areas. And it is in this way that we keep the changes in the resulting stego-image unnoticeable

**CHAPTER 3**

**LITERATURE SURVEY**

**3.1 GENERAL**

**3.1.1 REVERSIBLE DATA HIDING**

Ever since 1990s, the investigations on technology that is able to serve as an alternative to cryptography has attracted extensive attention from both academicians and industries. Information hiding has been widely considered as a fairly promising technology to fulfill this purpose. Information hiding works by covertly embedding a message within a host digital signal. The message to be hidden can be anything such as personal identification code, electronic patient information, a company logo or any secret information in the form of bit stream based on the application. Owing to its future applications, information hiding has become an emerging area of research in the past two decades.

Steganography is known as the art of covert communication between the sender and the receiver, in which the covert message is confidentially embedded in an imperceptible manner into the cover signal. This ensures that, nobody else other than sender and receiver is aware about the existence of the hidden message. Concealing the presence of hidden data is the ultimate goal of steganography and it is least bothered about recovering the cover medium at the receiver side. More similar concept to steganography is watermarking. Watermarking refers to the process of embedding messages, which is often called as watermark into a host signal without causing noticeable degradation to human eyes. The focus of a watermarking system is achieving a high level of robustness against attacks. In other words, a watermarked system makes it highly impossible to remove the injected watermark without causing visual damage to the stego image.

The art of hiding the data and recovering the original data back losslessly after retrieving the hidden content is known as Reversible Data Hiding. Over 20 years, RDH has been one of the active fields in research. This is evident as more and more research papers are being published these days. This RDH technique embeds secret message into the cover medium withoutcausing distortion in cover medium that remains unnoticeable by human eyes. It also restores the cover medium exactly after retrieving the secret data. However, the other data hidden techniques like steganographyand digital watermarking rarely care about the recovery of the cover medium. In few sensitive domains such as medical military, forensic, law enforcement etc., permanent distortion to the cover media is strictly forbidden. To overcome these issues, RDH which is otherwise called as invertible data hiding is proposed. RDH algorithms can be broadly classified based on the operation domain namely (i) frequency domain, (ii) spatial domain and (iii) encrypted domain.

## **3.1.2 SPATIAL DOMAIN**

RDH algorithms in spatial domain are broadly classified into Difference Expansion (DE), Histogram Shifting (HS) and Prediction Error Expansion (PEE). In spatial domain techniques, manipulations are carried out in the pixel values directly. Initially in 2001, RDH schemes were designed and developed just for fragile authentication. The scheme developed by [(Honsinger et al. 2001)](#_bookmark162) involved modulo 256 operation on the cover and the secret data. Main drawback of the scheme is the boundary pixels 0 or 255 may suffer salt and pepper noise while performing modulo 256 operation. In order to overcome this issue many schemes have been developed later.

* + 1. **DIFFERENCE EXPANSION**

In Difference Expansion (DE), expandable pixel pairs are selected. Difference values of selected pairs are computed, expanded and finally the message bit is appended to the binary version of expanded difference. This is finally added with the integer average of the selected pair of pixels. A single bit of information is embedded into each of the selected pairs. To identify the embeddable pair of pixels, a LM is formed. LM is nothing but an array of bits in which bit ‘1’ indicates expandable and bit ‘0’ indicates non-expandable or the other way The LM is compressed and embedded along with the secret message.

[Tian (2003)](#_bookmark202) introduced DE-RDH scheme which calculates the difference values of neighboring pixels and in that, few values were selected to expand the difference values. The information to be hidden is embedded into the difference values.

As the magnitude of DE increases, this algorithm produced sharpening effect in the mid tone region. This approach was experimented by generating pseudo random numbers rounded to nearest integer as payload. It was tested on various grayscale standard images, JPEG 2000 compressed images and achieved 44 dB PSNR for 0.15 bpp payload capacity. The payload achieved in this approach seems to be very low.

[Alattar (2004)](#_bookmark145) presented a reversible watermarking algorithm using DE of generalized integer transform for color images. This algorithm discusses about the reversible integer transform and several conditions in order to avoid underflow and overflow. He also discussed the technique to maximize the embedding to an extent by recursively embedding across the color components and proved that spatial quad based algorithm accepts more bits to be embedded, maintaining the signal-to-noise ratio at a high level. Simulation was carried out using triplets and quads of spatial and cross-color. The author also indicated that, (T[ian 2003)](#_bookmark202) method outperforms his approach for PSNR less than 35 dB.

[Kim et al. (2008)](#_bookmark171) introduced a reversible embedding method using DE transform in which a simplified LM and new expandable pixel pairs. The authors achieved this by exploiting the quasi-Laplace distribution of the expandable pixel pair values. This resulted in new expandable pixel pairs into which more message bits can be embedded. They also achieved improved performance keeping the distortion level same compared to (T[ian 2003),](#_bookmark202) Kamstra and Heijmans.

[Thodi and Rodríguez (2007)](#_bookmark201) proposed an alternative technique, which overcomes the undesirable distortion suffered by (T[ian 2003](#_bookmark202)). Histogram technique is introduced as an alternative to location map embedding. Prediction Error Expansion Embedding method is proposed which exploits the similarity inherent between the neighbor pixels instead of performing difference expansion. This also shows an improvement in the embedding capacity maintaining the signal-to noise ratio. He also described an embedding algorithm that combines histogram shifting and difference expansion that uses highly compressible overflow map and flag bits. The embedding capacity achieved by prediction error expansion method in a single pass is 1 bpp which is comparatively more than difference expansion method. [Sachnev et al. (2009)](#_bookmark194) introduced an embedding technique which does not use location map in most cases. The local variance magnitude is calculated and a sorting technique is used for recording the prediction error. By using sorting and prediction, information is embedded into the pixels with less distortion and location map if required. They also introduced the concept of rhombus prediction to exploit sorting. They achieved a maximum embedding capacity of 1bpp with minimal LM using histogram shifting and using the double embedding process on each pixel.NH

[Li, Li, Yang and Zeng (2013)](#_bookmark176) presented the concept of adaptive embedding and pixel selection for an efficient RDH technique. Unlike uniform embedding in conventional PEE, they proposed an adaptive embedding of 1 or 2 bits within the pixels which are expandable based on local complexity. The authors exploited the property of features invariant in conventional PEE data hiding and extraction and solved the problem of identifying adaptive embedding from the decoder perspective. They figured out that the major drawback of the proposed system was finding out the adaptive embedding threshold value iteratively.

[Bouslimi et al. (2012)](#_bookmark148) presented a scheme to protect medical images by combining a substitutive watermarking algorithm along with an encryption technique and quantization index modulation. Even though the image is stored encrypted, they proved access to the image integrity outcomes. They analyzed the proposed scheme in 8 bit ultrasound images and 16 bit encoded positron emission tomography images.

[Ou et al. (2013)](#_bookmark188) proposed an efficient reversible data hiding using prediction error expansion for pixel pair. They exploited image redundancy and also considered the correlation between pixel errors which leads to superior performance. They proposed to pair up adjacent pixel prediction error and generated an error sequence. Based on the error sequence and the predicted error 2D histogram, single or two bits could be embedded in by shifting and expanding the bins of 2D error histogram. They showed that the algorithm is best suited for low embedding capacity to avoid distortion while embedding more number of bits.

[Gui et al. (2014)](#_bookmark158) described a large embedding capacity RDH scheme based on adaptive embedding and generalized pixel prediction error expansion. They predicted each pixel using Gradient Adjusted Predictor (GAP) and also computed the complexity measurement partitioned into several levels according to its context. The amount of information bits that can be embedded depends upon the complexity measurement.

They embedded more bits in smooth region and in each complexity level, the amount of bits are adaptively chosen to give best performance. Author observed that the algorithm does not perform well for images, which generated larger location map.

[Chang and Huang (2016)](#_bookmark149) presented an effective RDH algorithm by predicting the pixel values and computing the error between predicted and original pixels. They utilized a weighting factor to predict the pixel values close to the original ones. Later, they preserved the weighting factor as a secret key for enhanced security.

[Kim et al. (2017)](#_bookmark172) proposed a RDH method to supplement any kind of distribution information. In this approach 42 bytes of distribution information is embedded invertibly and updated constantly without causing any damage to original image. They secured the distribution of data by providing a mask of Bose, Chaudhri, Hocquenghem (BCH) error correction code (31, 21, 2) technique having a block length of 31 bits, parity check bits of size 21 and error correcting capability of 2 bits. They also assured reversibility by means of incorporating DE method. The authors observed that the robustness deteriorates as the hidden data increases. Another disadvantage of the scheme is, it is vulnerable to geometric attacks. However, they proved that it sustains robustness against image alteration and recompression attacks which happen more frequently in image distribution settings.

[Ou et al. (2016),](#_bookmark186) [(Ou et al. 2017),](#_bookmark187) [(Xiao et al. 2019)](#_bookmark210) described a RDH scheme which adds enhancement to conventional pairwise PEE. They estimated the geodesic path metric between the pixel and its neighbors to determine the most similar one. Window size for this search varies from 1 to 3 and is fixed based on high PSNR by experimental analysis. Another parameter called histogram entropy was used to analyze the concentration of 2D histogram. Embedding distortion is said to be low when the histogram entropy is smaller. Main drawback of their approach is computational complexity. As the search window size increases, Dijkstra’s algorithm, which is used for solving the shortest path, takes longer time. Moreover, this scheme is used for low capacity embedding with a maximum of 0.4 bpp.

[Wang and Wang (2020)](#_bookmark205) presented a novel reversible data embedding technique by means of histogram shifting. There are two steps explained in this approach. First step is generating a difference histogram, which is more effective and produces steep difference. Second step is to extend and shift some of the differences in the histogram bin to create space for embedding the information bits. Steeper histogram is obtained by computing the second order difference histogram of the prediction error.

For each window of size 2 x 2 two first order differences are computed and second order differences are computed from the first order values. Finally, the message bits are embedded by shifting and expanding the second order difference values.

**3.1.4 HISTOGRAM SHIFTING/MODIFICATIION**

Histogram modification is a technique in RDH to embed the bits by altering the histogram of the cover image. Shifting the gray level immediate to peak of the histogram leaves a vacated position for embedding the data. Multiple peak points are identified to embed more information bits. Histogram of prediction error bins of cover image can also be modified to embed data into the cover media.

[Lin et al. (2008)](#_bookmark179) presented a multilevel histogram modification reversible data hiding technique considering the difference images. They applied the peak point of a histogram into a difference image and generated an inverse transformation in spatial domain in order to create free space. They also observed the characteristics of an image and found that probability of adjacent pixels being similar is very high. Therefore, they used the characteristics of difference histogram being derived from difference image.They combined the multi-level strategy and the peak point of difference image concept to embed a large amount of information with reversibility.

[Lin et al. (2008)](#_bookmark179) presented a difference histogram based RDH technique for embedding the information. Since, the embedding rate in conventional histogram is very limited, they adopted this difference histogram. Understanding the image characteristics that the adjacent pixels are highly correlated, the difference between the neighboring pixels is computed and generated histogram for the same. Histogram shifting has been performed and space for embedding is created. Later the shifted histogram has been modified to embed 1/0.

[Hong et al. (2008)](#_bookmark161) presented a Histogram Shifting of Prediction Error (HSPE) based reversible data hiding technique which has a two-stage structure. The prediction stage, in which each pixel value is predicted and the error value is calculated. In the error modification stage, histogram is constructed and is modified to create vacant position for embedding the secret message. They also proved that the PSNR obtained by HSPE is always above 48dB for an embedding rate of 0.2 bpp.

[Tai Wei—liang (2009)](#_bookmark199) introduced a RDH scheme based on histogram shifting by having a binary tree structure to solve multiple peak point communication. The algorithm has ‘L’ levels of binary tree and 2L number of peak points for embedding the message bits.

Number of tree levels increases as the embedding rate increases. The pixels of the image are predicted and prediction errors are computed. Each prediction error that satisfies certain conditions discussed in the algorithm embeds a single bit within it by visiting the right/left child node depending on the bit to be embedded is 0/1. Common drawback of all histogram techniques, that is providing an additional communication channel for peak and valley points, is overcome by having a binary tree structure. This binary tree structure predetermines more than one peak point that can be used to embed information.

[Yang and Tsai (2010)](#_bookmark215) introduced a RDH algorithm based on histogram modification using interleaving predictions. They carried out this approach on greyscale images and drawbacks in other histogram methods are overcome by interleaving prediction method. This method offers predictive values as many as that of the pixel values and finally these values are transformed into histogram. Histogram is then modified to embed data within it. They proved that the results guaranteed above 48dB.

(T[ai et al. 2009)](#_bookmark198) introduced a reversible data hiding scheme based on histogram modification. They described a binary tree structure to communicate with peak points pairs. They achieved large data hiding capacity with low distortion level. They also adopted histogram shifting technique to prevent underflow and overflow of pixel values. The tree level L is shared between sender and receiver which is used to determine the different peak points used to embed the secret information. ([Yang and Tsai 2010)](#_bookmark215) presented a histogram shifting based RDH by means of interleaving predictions. This approach enhances the embedding rate by having predictive values more or less signal to the number of pixels in an image. Later, all the predictive values are transformed to image histogram to create a high peak to enhance the embedding rate. They have adopted a pre-processing technique to avoid underflow/overflow. Before each layer embedding, a flag bit is used for recording the pixels that might underflow/overflow and altered a gray level before or after. The authors discussed about two interleaving prediction methods which can strengthen the histogram based RDH technique.

[Zhao et al. (2011)](#_bookmark225) introduced a new reversible data hiding technique called histogram modification in multiple levels and sequential recovery of data. They observed the similarity of the neighboring pixels and constructed histogram out of it. They adopted inverse ‘S’ order scan to generate pixel difference. More number of peak points and valley points are used to get a multilevel embedding using histogram modification. Moreover, while extracting the data, embedding levels are used instead of peak and valley points. They also proved that the stego image quality is better than the previously existing techniques.

[(Wu and Huang 2012)](#_bookmark207) described an invertible data hiding mechanism utilizing the histogram modification methodology. In this, they have explained how efficiently we can modify a pair of histogram bins for embedding. Multiple pairs of bins were chosen to embed data before which the pre-processing mechanism helps in presenting underflow/overflow. They suggested four prediction modes and later combined all to get the best performance. They also incorporated the location map and the overhead information along with the payload.

[Tsai et al. (2013)](#_bookmark203) and [(Wang et al. 2013)](#_bookmark206) proposed a RDH scheme for grayscale image. They constructed a histogram using the neighboring difference between pixel pair. Then they modified the constructed histogram to incorporate secret information into pixels whose pixel difference was within the histogram peak values. They achieved PSNR of 30 dB after performing multiple layers of data embedding using Neighbor Pixel Difference.

[Wang et al.](#_bookmark206) ([2013)](#_bookmark206) described a novel RDH method using histogram modification. This approach utilized the peak points of image segments based on intensity rather than utilizing the peakpoint in conventional image histogram. The authors modified the pixel values within each segment and limited it only with the segment. So, the stego image quality is dependent only on the segmentation size. A location map is created to register the location of the segment peak into which data is hidden. The authors proved that this scheme is very much suitable for multiple embedding without causing any loss in fidelity of image, as the segment size determines the degree at which pixels were modified.

[Li, Li, Li and Yang (2013)](#_bookmark177) presented a general framework for constructing a histogram based RDH algorithm. They designed functions called, shifting and embedding functions and proved that most of the histogram shifting algorithms are special cases of these functions. They also incorporated the general framework with pixel selection and PEE techniques. They also discussed how to embed auxiliary information and the required amount of information bits.

[Lo et al. (2014)](#_bookmark180) and [(Kim et al. 2018*a*](#_bookmark169)) proposed a RDH scheme for Block Truncation Coding (BTC) compressed images using histogram shifting. There are three steps involved in this scheme. First is to generate secret data, second is the embedding for the generated secret data, finally the secret data extraction procedure. Secret data has been generated using the Pseudo Random Number Generator (PRNG) of predefined seed. They embedded the data losslessly into the quantization levels of block truncation codes of compressed codes.

[Chang et al. (2015)](#_bookmark150) identified a high capacity RDH scheme for block truncation coding based on residual histogram shifting. The approachwas carried out iteratively in compressed images using block truncation coding. The residual values are generated by processing the quantization levels in each round. They used the residual values to embed the secret data. They also performed multiple levels of embedding to improve the hiding capacity of their scheme. Even after the data embedding procedure is carried out, the embedded compressed codes still follow the standard BTC format.

[Lu et al. (2016)](#_bookmark182) proposed a method based on asymmetric histogram shifting. They analyzed the sensitivity of edges using a methodology adopted by [(Lukac et al. 2004).](#_bookmark183) This approach reduces the prediction errors and also linked the asymmetric histogram shifting introduced by [(Chen et al. 2013)](#_bookmark154) which was used to restore the error value closer to the original pixel value during second shifting. The authors also showed that complementary mechanism of pixel gave better image quality even in multi-layered embedding. Embedding capacity differs for different characteristics of image.

[Jia et al. (2019), (Jia et al. 2019)](#_bookmark165) described a histogram shifting based RDH approach which emphases on reducing the distortion caused due to embedding. They divided the image into two sub-images with the help of checkerboard pattern and the fluctuation values caused in each sub images were computed. Smoother regions in image caused smaller fluctuations and un-smoother region resulted in more fluctuations. This is because of invalid shifting of histogram while embedding the data. The authors identified and avoided the invalid shifting. Invalid shifting will be more in un-smooth regions than the smoother regions.

[Hou et al. (2018)](#_bookmark163) presented a reversible embedding scheme which was completely varying from the existing schemes. They had analysed this scheme in color images with gray scale invariance. This was done by preserving the gray scale version of the color images. The unchanged gray version of the color images was used for embedding and

extraction of the hidden data. Embedding was performed by adjusting the red and blue channel of the color images and the remaining green channel was modified adaptively to remove the offset caused in embedding in red and blue channel of the color image. The algorithm involves generating of sharp histogram and adaptive casual predictor was also used. The authors had also adopted recursive embedding algorithm to achieve a higher embedding capacity. They had also applied and presented the various applications in which the gray version of the color images can be used.

[Tang et al. (2020)](#_bookmark200) introduced a reversible hiding scheme in color images which has emerged as an important topic of discussion these days. This was accomplished by means of performing double layer embedding in the cover images. They had also used the concept of HS and interpolation. Interpolation was used to generate the prediction error matrix for HS. Higher embedding capacity was achieved by incorporating inter channel correlation between the two embedding layers .The authors had also proved using open standard data sets.

## **3.1.5 PIXEL VALUE ORDERING**

[He et al. (2018)](#_bookmark159) and [Kumar and Jung (2020)](#_bookmark173) introduced PVO based RDH scheme for moderate embedding capacity. This scheme offers high fidelity stego image quality by segregating the cover image into smooth and complex region based on the rhombus mean prediction of each non-overlapping block. The authors had explained the embedding concept by introducing several pass based on the category of the block. They also showed significant improvement in the performance parameters.

**CHAPTER 4**

**SYSTEM DESIGN**

### 4.1 ARCHITECTURE DIAGRAM

The Steganography application will allow the user to load an image into the system for steganography encoding, supply the message file, encrypt the file content using a specified key and compress the file if required. Once the user has supplied all this parameters, the stego-Encoder takes over during the process of sending it via the communication channel (internet).

### kumar09.kanagala - Steganography

***Figure 4.1.1 System architecture***

### 4.2 UML DIAGRAMS

**4.2.1 SEQUENCE DIAGRAM**

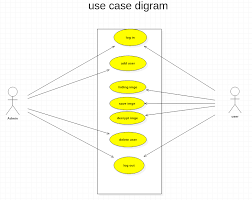
Actor in sequence diagram is user who gives command to select stego video and input. Then using extract,it takes encrypted text or image or video is being decrypted. Sequence diagram of steganography encryption steg text image message encryption image takeimage() selectmessage() taketext() selectimage() encrypt() ..

### Sequence Diagram for user to extract secret message | Download Scientific Diagram

### Fig 4.2.1 SEQUENCE DIAGRAM

* + 1. **USECASE DIAGRAM**

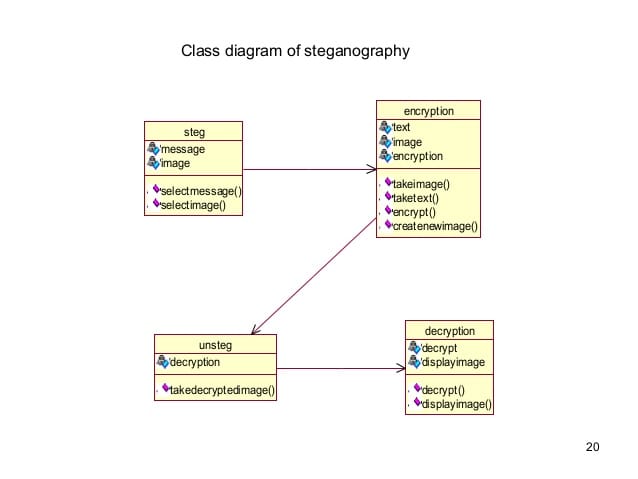
The use case diagram is used to identify the primary elements and processes that form the system. The primary elements are termed as "actors" and the processes are called "use cases." The Use case diagram shows which actors interact with each use case. A use case diagram captures the functional aspects of a system. Fig. 3.7 Use case diagram representing the sender end of System Fig 3.7 represents all actions taken by a user (sender) when he tries to send a hidden message to another user (receiver) using the steganography system. Within the steganography application, the user must specify a message file and enter an encryption key and an image and submit it to the steganography encoder for processing.



### Fig No.4.2.2 USECASE DIAGRAM

**4.2.3 CLASS DIAGRAM**

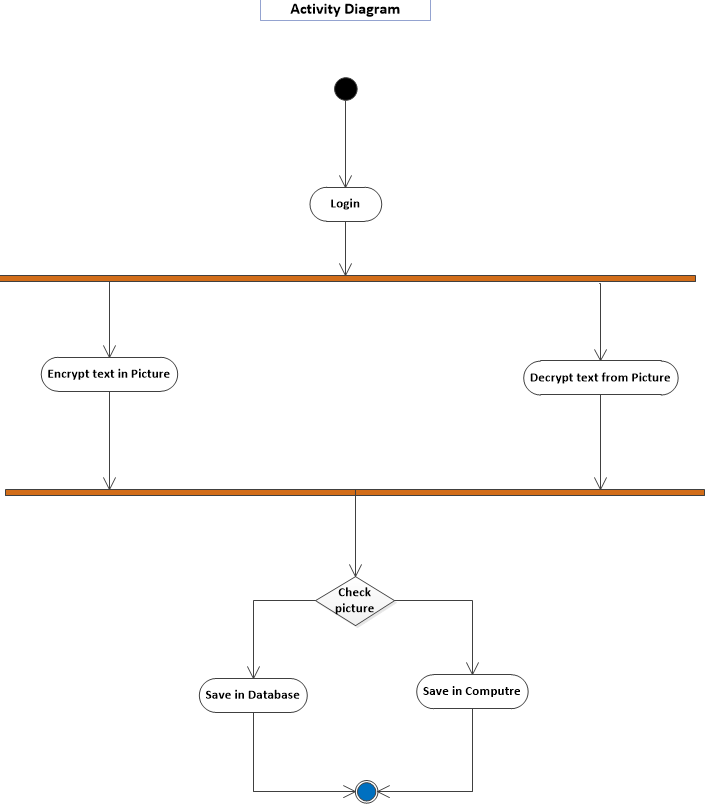
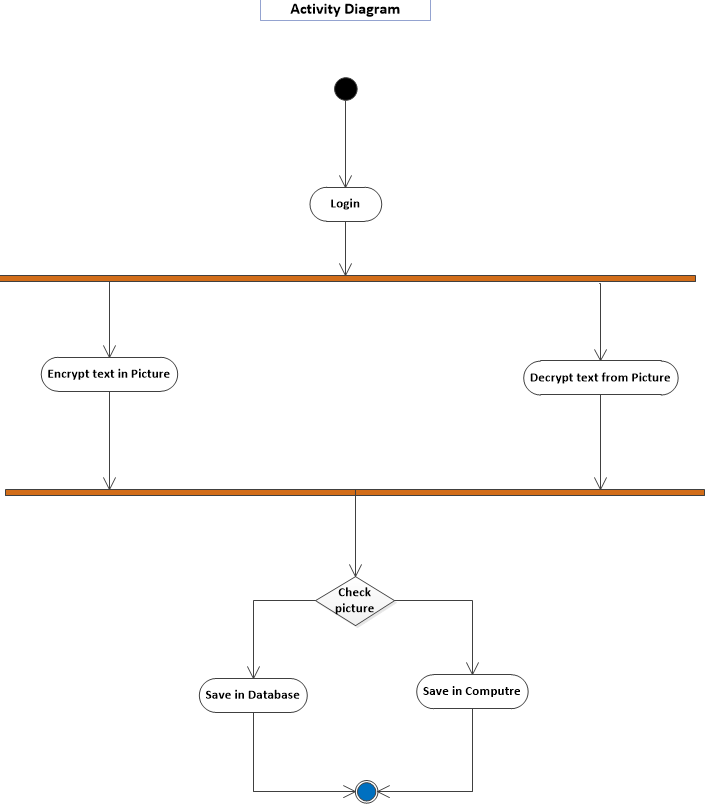
A class diagram is a diagram showing a collection of classes and interfaces, along with the collaborations and relationships



**Fig 4.2.3 CLASS DIAGRAM**

* + 1. **ACTIVITY DIAGRAM**

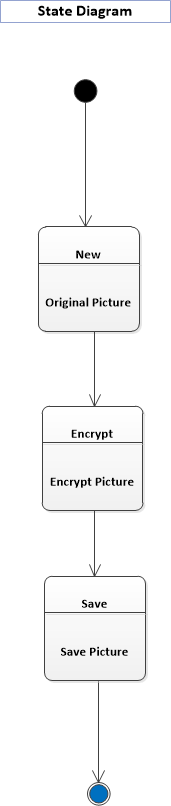
Image steganography is about hiding the secret message into the image . It is a technique uses to secure the transmission of secret information or hide their existence. It also may provide confidentiality to secret message if the message is encrypted.



**FIG 4.2.3 ACTIVITY DIAGRAM**

* + 1. **STATE DIAGRAM**

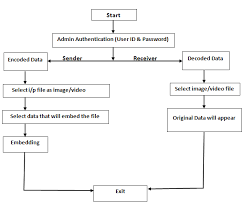
A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction.



**FIG 4.2.4 STATE DIAGRAM**

* + 1. **DATA FLOW DIAGRAM**

Privacy and security over communication channels are of primary concerns. Due to their complexity and diversity, there is a need for continuous improvements of the adopted solutions. In this study, we consider two of the adopted ones, namely, steganogrphy and cryptography and propose a new information hiding system. A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.  Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one.



**Fig 4.2.3 DATA FLOW DIAGRAM**

**chapter 5**

**PROJECT OVERVIEW**

**5.1) PROJECT DESCRIPTION**

There are two trends at the time to implement steganographic algorithms: the methods that work in the **spatial domain** (altering the desired characteristics on the file itself) and the methods that work in the **transform domain** (performing a series of changes to the cover image before hiding information. To select the best areas the Discrete Cosine Transform DCT, Wavelet Transform, etc. are used).

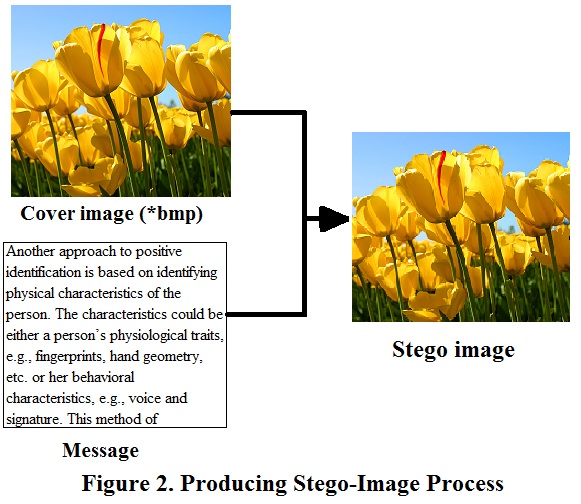
While the algorithms that work in the transform domain are more robust, that is, more resistant to attacks, the algorithms that work in the spatial domain are simpler and faster.

The best known steganographic method that works in the spatial domain is the LSB (Least Significant Bit), which replaces the least significant bits of pixels selected to hide the information. This method has several implementation versions that improve the algorithm in certain aspects.

We have chosen to implement LSB Substitution in our project because of its ubiquity among carrier formats and message types. With LSB Substitution, we could easily change from Image Steganography to Audio Steganography and hide a zip archive instead of a text message. LSB Substitution lends itself to become a very powerful Steganographic method with few limitations. LSB Substitution works by iterating through the pixels of an image and extracting the ARGB values. It then separates the color channels and gets the least significant bit. Meanwhile, it also iterates through the characters of the message setting the bit to its corresponding binary value.

In our project, we have made an APPLICATION on steganography to be used by the Indian Defence Services in which we provide a user friendly interface to encrypt a data file in a bmp image format. We have used a variation of LSB technique where we are encoding the last 4 LSBs in direct proportion to the first 4 MSBs, by which I mean that, if in out image, the 8 bits are: 11110101, and we need to encode 1101 in our image bits, we will encode all the last four bits and the output will be, 11111101.

Similarly, if the first four bits of our image are 11100110, we will encode the last 3 bits and so on…



The *cover-image* will be combined with the message. This will produce the output called *stego-image*. Figure 2 is illustrating the process. The *Stego-image* seems identical to the *cover-image*. However, there are hidden message that imperceptible. This process simply embedded the message into the *cover-image* without supplied any password or *stego-key*. At this stage, we decided to do so because we have to understand the ways of LSB insert the message bit into the image and extract the message from the *stego-image* produced.

**5.2) REQUIREMENT SPECIFICATION**

**5.2.1) Hardware Requirements:**

Minimum 1GB RAM, Minimum 10 GB Hard Disk Space

**Operating System supported:** Windows XP, Windows Vista, Windows 7

**5.2.2) Software Requirements:**

Visual Studio 2010, .NET Framework 4.0 and Adobe Photoshop CS5 installed on the system.

**SOFTWARE DESCRIPTION**

**5.2.2.1) python**

Create a virtualenv and install the requirements:

virtualenv venv

source venv/bin/activate

pip install -r requirements.txt

Then, merge and unmerge your files with:

python steganography.py merge --img1=res/img1.jpg --img2=res/img2.jpg --output=res/output.png

python steganography.py unmerge --img=res/output.png --output=res/output2.png

To use the **Steganography** class in your **Python** code, you will need to use the **Image** module from the **Pillow** library, for example:

from PIL import Image

merged\_image = Steganography.merge(Image.open(img1), Image.open(img2))

merged\_image.save(output)

**5.3) FEASILBILITY STUDY**

**Technical Feasibility**

Image Steganography is a defence application with a back-end coding done in C#.NET that allows a user to hide data (documents or text files) in an image. The user friendly interface requires no technical skills and is easy to operate on. Visual Studio 2010 using .NET 4.0 framework is used for the design and coding purposes. Adobe Photoshop CS5 has also been used to design the header of the application with an animated sequence of images.

**Economic Feasibility**

The project is economic and highly beneficial project as far as the cost of development is considered. No extra costs were incurred apart from the software used.

**Operational Feasibility**

The project is operationally very feasible as it is user-friendly, the user doesn’t need any kind of knowledge about the software used in the project. The project is also really helpful as the user can use it to send encrypted data at any moment of time using the internet or the LAN system.

**5.4) OBJECTIVE**

This project comprehends the following objectives:

* To produce security tool based on steganographic techniques.
* To explore LSB techniques of hiding data using steganography.

**5.5) SCOPE**

The scope of the project as follows:

* Implementation of a variation of LSB technique for hiding information i.e. text in image files.

**5.6) IMAGE DEFINITION**

To a computer, an image is a collection of numbers that constitute different light intensities in different areas of the image. This numeric representation forms a grid and the individual points are referred to as pixels. Most images on the Internet consists of a rectangular map of the image’s pixels (represented as bits) where each pixel is located and its color. These pixels are displayed horizontally row by row. The number of bits in a color scheme, called the bit depth, refers to the number of bits used for each pixel.

The smallest bit depth in current color schemes is 8, meaning that there are 8 bits used to describe the color of each pixel. Monochrome and greyscale images use 8 bits for each pixel and are able to display 256 different colors or shades of grey. Digital color images are typically stored in 24-bit files and use the RGB color model, also known as true color. All color variations for the pixels of a 24-bit image are derived from three primary colors: red, green and blue, and each primary color is represented by 8 bits. Thus in one given pixel, there can be 256 different quantities of red, green and blue, adding up to more than 16-million combinations, resulting in more than 16-million colors. Not surprisingly the larger amount of colors that can be displayed, the larger the file size. For this project, we are considering 8-bit images.

**5.7) IMAGE COMPRESSION**

When working with larger images of greater bit depth, the images tend to become too large to transmit over a standard Internet connection. In order to display an image in a reasonable amount of time, techniques must be incorporated to reduce the image’s file size. These techniques make use of mathematical formulas to analyse and condense image data, resulting in smaller file sizes. This process is called compression. In images there are two types of compression: lossy and lossless. Both methods save storage space, but the procedures that they implement differ. Lossy compression creates smaller files by discarding excess image data from the original image. It removes details that are too small for the human eye to differentiate, resulting in close approximations of the original image, although not an exact duplicate. An example of an image format that uses this compression technique is JPEG (Joint Photographic Experts Group).

Lossless compression, on the other hand, never removes any information from the original image, but instead represents data in mathematical formulas. The original image’s integrity is maintained and the decompressed image output is bit-by-bit identical to the original image input. The most popular image formats that use lossless compression is GIF (Graphical Interchange Format) and 8-bit BMP (a Microsoft

Windows bitmap file).

Compression plays a very important role in choosing which steganographic algorithm to use. Lossy compression techniques result in smaller image file sizes, but it increases the possibility that the embedded message may be partly lost due to the fact that excess image data will be removed. Lossless compression though, keeps the original digital image intact without the chance of lost, although is does not compress the image to such a small file size. Different steganographic algorithms have been developed for both of these compression types and will be explained in the following sections.

**5.8) LEAST SIGNIFICANT BIT**

Least significant bit (LSB) insertion is a common, simple approach to embedding information in a cover image. The least significant bit (in other words, the 8th bit) of some or all of the bytes inside an image is changed to a bit of the secret message. When using a 24-bit image, a bit of each of the red, green and blue color components can be used, since they are each represented by a byte. In other words, one can store 3 bits in each pixel. An 800 × 600 pixel image, can thus store a total amount of 1,440,000 bits or 180,000 bytes of embedded data. For example a grid for 3 pixels of a 24-bit image can be as follows:

(00101101 00011100 11011100)

(10100110 11000100 00001100)

(11010010 10101101 01100011)

When the number 200, which binary representation is 11001000, is embedded into the least significant bits of this part of the image, the resulting grid is as follows:

(0010110**1** 0001110**1** 1101110**0**)

(1010011**0** 1100010**1** 0000110**0**)

(1101001**0** 1010110**0** 01100011)

Although the number was embedded into the first 8 bytes of the grid, only the 3 underlined bits needed to be changed according to the embedded message. On average, only half of the bits in an image will need to be modified to hide a secret message using the maximum cover size. Since there are 256 possible intensities of each primary color, changing the LSB of a pixel results in small changes in the intensity of the colors. These changes cannot be perceived by the human eye - thus the message is successfully hidden. With a well-chosen image, one can even hide the message in the least as well as second to least significant bit and still not see the difference.

In the above example, consecutive bytes of the image data – from the first byte to the end of the message – are used to embed the information. This approach is very easy to detect. A slightly more secure system is for the sender and receiver to share a secret key that specifies only certain pixels to be changed. Should an adversary suspect that LSB steganography has been used, he has no way of knowing which pixels to target without the secret key.

In its simplest form, LSB makes use of BMP images, since they use lossless compression. Unfortunately to be able to hide a secret message inside a BMP file, one would require a very large cover image. Nowadays, BMP images of 800 × 600 pixels are not often used on the Internet and might arouse suspicion. For this reason, LSB steganography has also been developed for use with other image file formats.

**5.9) DETECTION/ ATTACKS**

While the purpose of Steganography is to hide messages, it may not be very effective at doing so. There are several attacks that one may execute to test for Steganographed images. They are:

* Visual Attacks
* Enhanced LSB Attacks
* Chi-Square Analysis, and
* Other statistical analyses.

In performing a **visual attack** you must have the original “virgin” image to compare it the Steganographed image and visually compare the two for artifacts.

In the **Enhanced LSB Attack**, you process the image for the least significant bits and if the LSB is equal to one, multiply it by 255 so that it becomes its maximum value.

**Chi-Square Analysis** calculates the average LSB and constructs a table of frequencies and Pair of Values; it takes the data from these two tables and performs a chi-square test. It measures the theoretical vs. calculated population difference. The Chi-Square Analysis calculates the chi-square for every 128 bytes of the image. As it iterates through, the chi-square value it calculates becomes more and more accurate until too large of a dataset has been produced. Because this attack relies on statistical analysis it cannot detect patterns or Steganography on very complex images with lots of noise than one can detect through visualization of the Enhanced LSB’s.

**5.10) BENEFITS/ DRAWBACKS**

The **advantages** of LSB are its simplicity to embed the bits of the message directly into the LSB plane of cover-image and many techniques use these methods. Modulating the LSB does not result in a human-perceptible difference because the amplitude of the change is small. Therefore, to the human eye, the resulting stego-image will look identical to the cover-image. This allows high perceptual transparency of LSB.

However, there are a few **weaknesses** of using LSB. It is very sensitive to any kind of filtering or manipulation of the stego-image. Scaling, rotation, cropping, addition of noise, or lossy compression to the stego-image will destroy the message.

On the other hand, for the hiding capacity, the size of information to be hidden relatively depends to the size of the cover-image. The message size must be smaller than the image. A large capacity allows the use of the smaller cover-image for the message of fixed size, and thus decreases the bandwidth required to transmit the stego-image.

Another weakness is an attacker can easily destruct the message by removing or zeroing the entire LSB plane with very little change in the perceptual quality of the modified stego-image. Therefore, if this method causes someone to suspect something hidden in the stego-image, then the method is not success.

**CHAPTER 6**

**CODING AND TESTING**

**Hiding an Image:**

1.To hide an image inside another, the image which will be hidden needs to have at most the same size of the image which will hide it.

2. We must create two loops to go through all rows and columns (actually each pixel) from the images.

3. So, we get the RGB from the image 1 and image 2 as binary values:

We can use the **\_\_int\_to\_bin** method to convert a decimal value to a binary value:

4. We merge the most significant bits from the image 1 with the most significant bits from the image 2:

Using the **\_\_merge\_rgb** method:

Note that the **\_\_merge\_rgb** function is using the 4 most significant bits from each image, but it could be changed. Keep in mind that using fewer bits from the hidden image will result in low quality of the recovery image.

5. Finally, we convert the new binary value to a decimal value:

Using the**\_bin\_to\_int** method:

And set it to a new pixel position from the resulted image.

Now we have an image hidden inside another image.

The entire **merge** method can be found.

**Revealing an Image**:

1.To reveal an image, we must know how many bits were used to hide the image. In this case, we are using a fixed number of 4 bits.

2.First of all, we need to create two loops to go through all pixels from the image:

3. So, we extract each RGB channel as a binary value from the current pixel:

Using the **\_\_int\_to\_bin** method:

4. Then, we create a new RGB value by concatenating only the 4 rightmost bits from the current pixel with zero values (to create a new 8-bit value):

5. Finally, we convert the binary value to a decimal value and set it to the current pixel in the new image:

6. The developed algorithm has only one more last step to remove the black borders when the hidden image was smaller than the image which is hiding it.

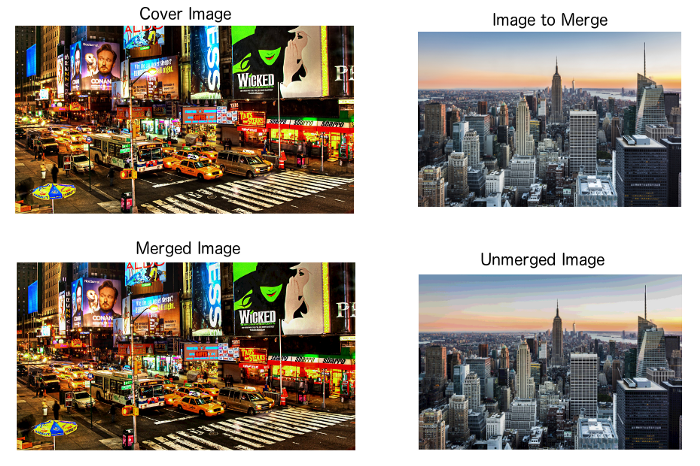
And with this simple code, we can extract an image from another.

The entire **unmerge** method can be found here.

**You can check out the result in the following image**:

The left upper image is the image that will hide the right upper image. The left lower image is the two images merged and the right lower image is the extracted (unmerged) image.





## 

## 

## **CHAPTER 7**

## **APPENDICES**

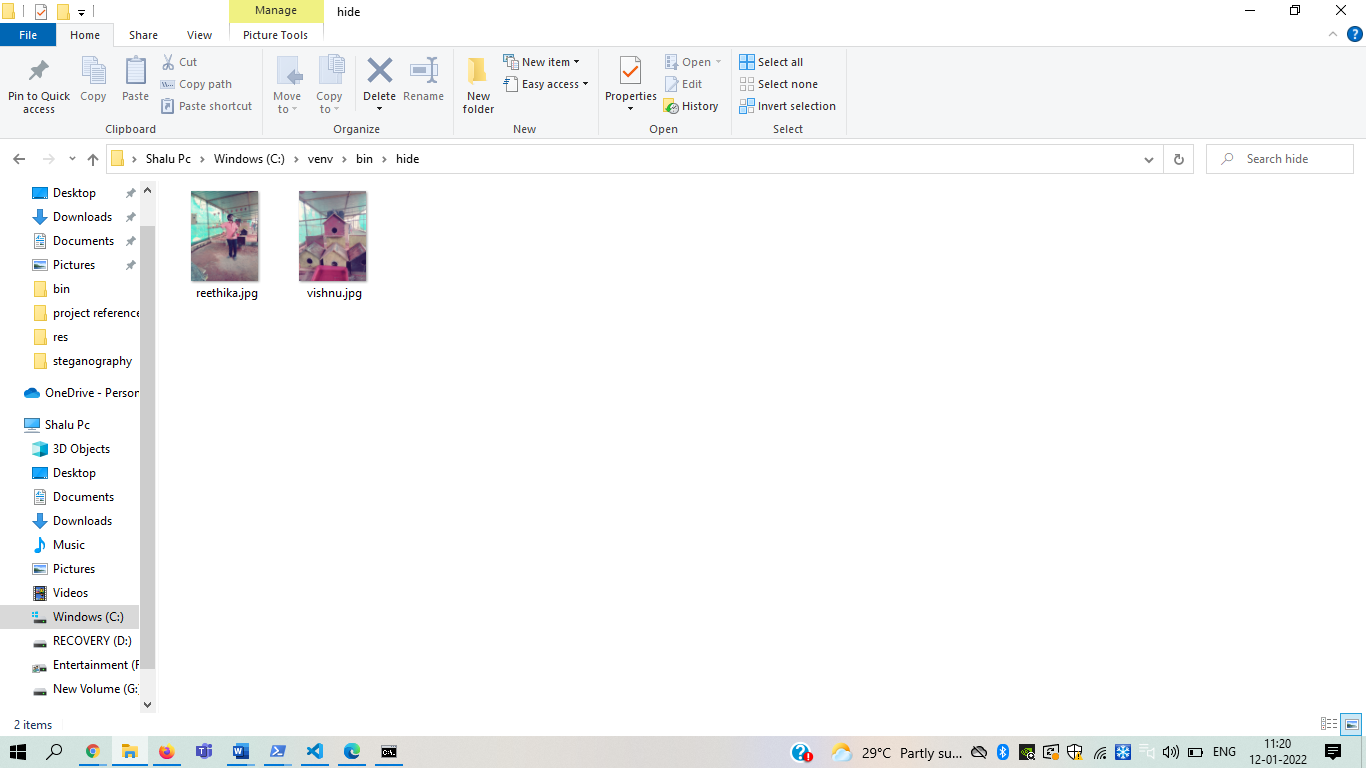
## 

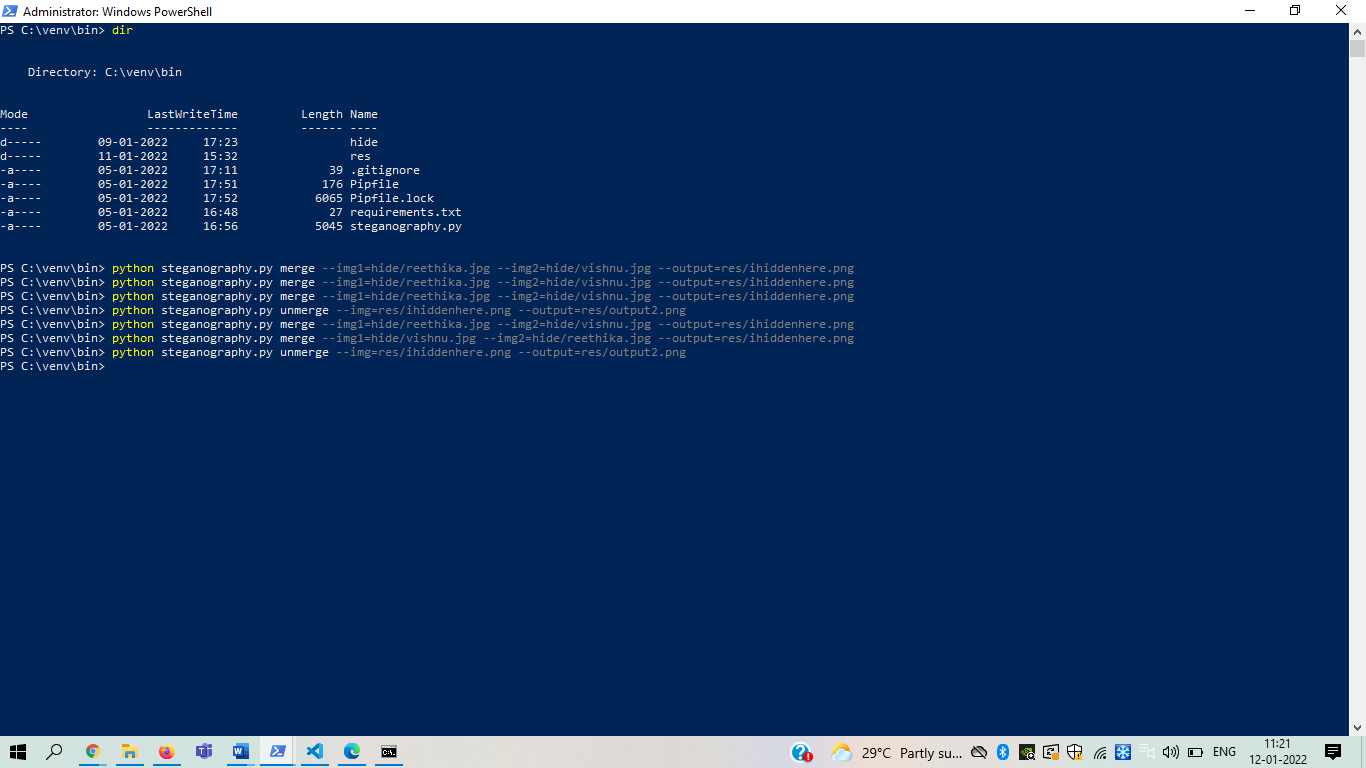
**SAMPLE CODE**:

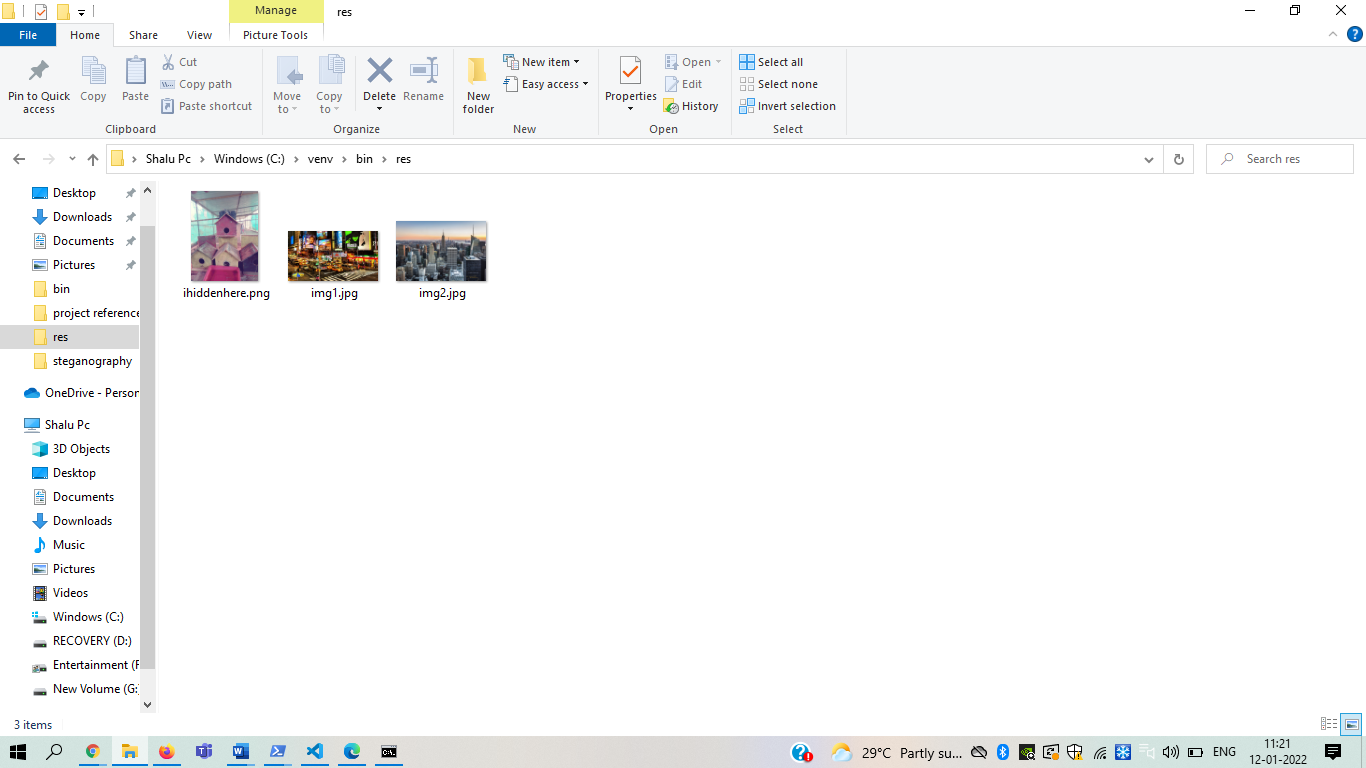
Steganography.py

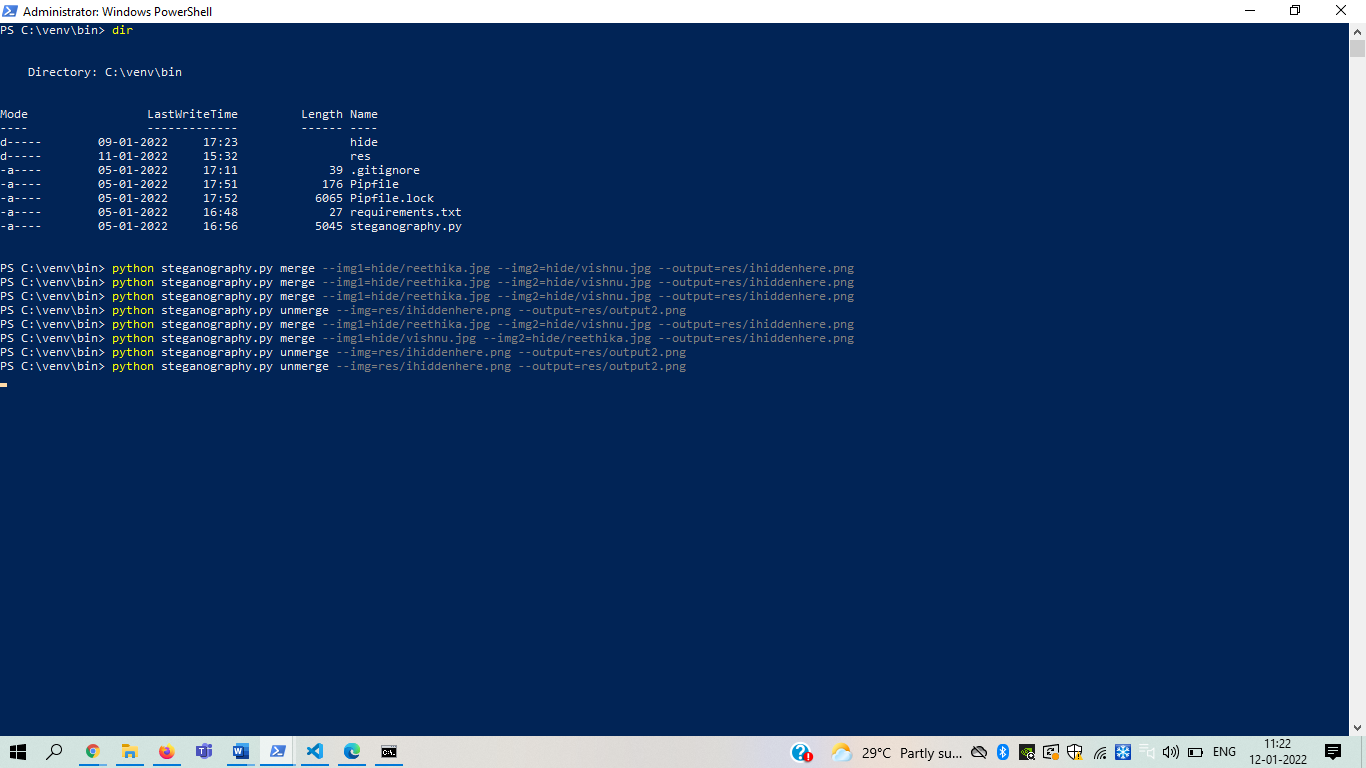
|  |
| --- |
| #!/usr/bin/env python |
|  |  |
|  | import click |
|  | from PIL import Image |
|  |  |
|  |  |
|  | class Steganography: |
|  |  |
|  | @staticmethod |
|  | def \_\_int\_to\_bin(rgb): |
|  | """Convert an integer tuple to a binary (string) tuple. |
|  |  |
|  | :param rgb: An integer tuple (e.g. (220, 110, 96)) |
|  | :return: A string tuple (e.g. ("00101010", "11101011", "00010110")) |
|  | """ |
|  | r, g, b = rgb |
|  | return (f'{r:08b}', |
|  | f'{g:08b}', |
|  | f'{b:08b}') |
|  |  |
|  | @staticmethod |
|  | def \_\_bin\_to\_int(rgb): |
|  | """Convert a binary (string) tuple to an integer tuple. |
|  |  |
|  | :param rgb: A string tuple (e.g. ("00101010", "11101011", "00010110")) |
|  | :return: Return an int tuple (e.g. (220, 110, 96)) |
|  | """ |
|  | r, g, b = rgb |
|  | return (int(r, 2), |
|  | int(g, 2), |
|  | int(b, 2)) |
|  |  |
|  | @staticmethod |
|  | def \_\_merge\_rgb(rgb1, rgb2): |
|  | """Merge two RGB tuples. |
|  |  |
|  | :param rgb1: A string tuple (e.g. ("00101010", "11101011", "00010110")) |
|  | :param rgb2: Another string tuple |
|  | (e.g. ("00101010", "11101011", "00010110")) |
|  | :return: An integer tuple with the two RGB values merged. |
|  | """ |
|  | r1, g1, b1 = rgb1 |
|  | r2, g2, b2 = rgb2 |
|  | rgb = (r1[:4] + r2[:4], |
|  | g1[:4] + g2[:4], |
|  | b1[:4] + b2[:4]) |
|  | return rgb |
|  |  |
|  | @staticmethod |
|  | def merge(img1, img2): |
|  | """Merge two images. The second one will be merged into the first one. |
|  |  |
|  | :param img1: First image |
|  | :param img2: Second image |
|  | :return: A new merged image. |
|  | """ |
|  |  |
|  | # Check the images dimensions |
|  | if img2.size[0] > img1.size[0] or img2.size[1] > img1.size[1]: |
|  | raise ValueError('Image 2 should not be larger than Image 1!') |
|  |  |
|  | # Get the pixel map of the two images |
|  | pixel\_map1 = img1.load() |
|  | pixel\_map2 = img2.load() |
|  |  |
|  | # Create a new image that will be outputted |
|  | new\_image = Image.new(img1.mode, img1.size) |
|  | pixels\_new = new\_image.load() |
|  |  |
|  | for i in range(img1.size[0]): |
|  | for j in range(img1.size[1]): |
|  | rgb1 = Steganography.\_\_int\_to\_bin(pixel\_map1[i, j]) |
|  |  |
|  | # Use a black pixel as default |
|  | rgb2 = Steganography.\_\_int\_to\_bin((0, 0, 0)) |
|  |  |
|  | # Check if the pixel map position is valid for the second image |
|  | if i < img2.size[0] and j < img2.size[1]: |
|  | rgb2 = Steganography.\_\_int\_to\_bin(pixel\_map2[i, j]) |
|  |  |
|  | # Merge the two pixels and convert it to a integer tuple |
|  | rgb = Steganography.\_\_merge\_rgb(rgb1, rgb2) |
|  |  |
|  | pixels\_new[i, j] = Steganography.\_\_bin\_to\_int(rgb) |
|  |  |
|  | return new\_image |
|  |  |
|  | @staticmethod |
|  | def unmerge(img): |
|  | """Unmerge an image. |
|  |  |
|  | :param img: The input image. |
|  | :return: The unmerged/extracted image. |
|  | """ |
|  |  |
|  | # Load the pixel map |
|  | pixel\_map = img.load() |
|  |  |
|  | # Create the new image and load the pixel map |
|  | new\_image = Image.new(img.mode, img.size) |
|  | pixels\_new = new\_image.load() |
|  |  |
|  | # Tuple used to store the image original size |
|  | original\_size = img.size |
|  |  |
|  | for i in range(img.size[0]): |
|  | for j in range(img.size[1]): |
|  | # Get the RGB (as a string tuple) from the current pixel |
|  | r, g, b = Steganography.\_\_int\_to\_bin(pixel\_map[i, j]) |
|  |  |
|  | # Extract the last 4 bits (corresponding to the hidden image) |
|  | # Concatenate 4 zero bits because we are working with 8 bit |
|  | rgb = (r[4:] + '0000', |
|  | g[4:] + '0000', |
|  | b[4:] + '0000') |
|  |  |
|  | # Convert it to an integer tuple |
|  | pixels\_new[i, j] = Steganography.\_\_bin\_to\_int(rgb) |
|  |  |
|  | # If this is a 'valid' position, store it |
|  | # as the last valid position |
|  | if pixels\_new[i, j] != (0, 0, 0): |
|  | original\_size = (i + 1, j + 1) |
|  |  |
|  | # Crop the image based on the 'valid' pixels |
|  | new\_image = new\_image.crop((0, 0, original\_size[0], original\_size[1])) |
|  |  |
|  | return new\_image |
|  |  |
|  |  |
|  | @click.group() |
|  | def cli(): |
|  | pass |
|  |  |
|  |  |
|  | @cli.command() |
|  | @click.option('--img1', required=True, type=str, help='Image that will hide another image') |
|  | @click.option('--img2', required=True, type=str, help='Image that will be hidden') |
|  | @click.option('--output', required=True, type=str, help='Output image') |
|  | def merge(img1, img2, output): |
|  | merged\_image = Steganography.merge(Image.open(img1), Image.open(img2)) |
|  | merged\_image.save(output) |
|  |  |
|  |  |
|  | @cli.command() |
|  | @click.option('--img', required=True, type=str, help='Image that will be hidden') |
|  | @click.option('--output', required=True, type=str, help='Output image') |
|  | def unmerge(img, output): |
|  | unmerged\_image = Steganography.unmerge(Image.open(img)) |
|  | unmerged\_image.save(output) |
|  |  |
|  |  |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  | cli() |

### 7.1 EXPERIMENTAL RESULT









# 

# CHAPTER 8

# CONCLUSION

# 8.1.CONCLUSION

Reversible data hiding has materialized as a foremost research area because of the phenomenal growth in internet and digital media technology. It involves communication of private or confidential data in a stealthy mode within the cover media so that the changes caused due to embedding remains imperceptible. Unlike steganography, this characteristic of RDH finds appropriateness in applications like medical image processing, authentication of digital media, military image processing and law enforcement applications where a slight distortion in cover media is strictly forbidden. Even a slight perceptible distortion in the cover media can cause suspicion on the transmitted data or may lead to misinterpretation of data in case of medical image processing. Hence restoration of the cover media in an undistorted manner plays a major role in such sensitive applications. A number of algorithms developed in past two decades have been discussed in the literature before. Each method differs from the other by the way the cover image points are manipulated for embedding the data. This research work is intended to carry out investigations on various methods of data hiding in spatial and frequency domain. Since the characteristics of reversible data hiding are interlinked to one another, each scheme tries to improvise one of those without causing much degradation to the other. Hence, the suggested works have proven a stable trade-off among the three extremities such as embedding capacity, imperceptibility and security for an optimal reversible data hiding in the natural and medical images.

### 8.2.FUTURE ENHANCEMENTS

### In the near future, the most important use of steganographic techniques will probably be lying in the field of digital watermarking. Content providers are eager to protect their copyrighted works against illegal distribution and digital watermarks provide a way of tracking the owners of these materials. Steganography might also become limited under laws, since governments already claimed that criminals use these techniques to communicate.

### The possible use of steganography technique is as following:

### i. Hiding data on the network in case of a breach.

### ii. Peer-to-peer private communications.

### iii. Posting secret communications on the Web to avoid transmission.

### iv. Embedding corrective audio or image data in case corrosion occurs from a poor connection or transmission.

### 8.3REFERENCES

[1] R. Roy and S. Changder, “Image realization steganography with LCS based mapping,” in *Proceedings of the 2014 7th International Conference on Contemporary Computing (IC3)*, pp. 218–223, Noida, India, 2014.

[2] S. Samima, R. Roy, and S. Changder, “Secure key based image realization steganography,” in *Proceedings of the 2013 IEEE 2nd International Conference on Image Information Processing (ICIIP)*, pp. 377–382, Shimla, India, 2013.

[3] D.-C. Wu and W.-H. Tsai, “A steganographic method for images by pixel-value differencing,” *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1613–1626, 2003.

[4] M. Y. Valandar, P. Ayubi, and M. J. Barani, “A new transform domain steganography based on modified logistic chaotic map for color images,” Journal of Information Security and Applications, vol. 34, pp. 142–151, 2017.

[5] R. Bhardwaj and V. Sharma, “Image steganography based on complemented message and inverted bit LSB substitution,” *Procedia Computer Science*, vol. 93, pp. 832–838, 2016.

[6] M.Y Valandar, M. J. Barani, P. Ayubi, and M. Aghazadeh, “An integer wavelet transform image steganography method based on 3D sine chaotic map,” Multimedia Tools and Applications, vol. 78, no. 8, pp. 9971–9989, 2019.

[7] A. Sahu, G. Swain, and G. Swain, “Dual stegnography imaging based reversible data hiding using improved LSB matching,” International Journal of Intelligent Engineering and Systems, vol. 12, no. 5, pp. 63–73, 2019.

[8] R. Kumar, S. Singh, and K.-H. Jung, “Human visual system based enhanced for color image compression using interpolation,” in *Proceedings of the 2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)*, pp. 903–907, Noida, India, 2019.

[9] C.-K. Chan and L. M. Cheng, “Hiding data in images by simple LSB substitution,” Pattern Recognition, vol. 37, no. 3, pp. 469–474, 2004.

[10] I. Hafi, M. Noman, M. Gohar et al., “An adaptive hybrid fuzzy-wavelet approach for image steganography using bit reduction and pixel adjustment,” *Soft Computing*, vol. 22, no. 5, pp. 1555–1567, 2018.

### [11] K. J. Devi, “A secure image steganography using lsb technique and pseudo random encoding technique,” Department of Computer Science and Engineering, National Institute of Technology, Rourkela, India, 2013, Ph.D. thesis