

DATA HIDING USING IMAGE STEGANOGRAPHY

Major project report submitted in partial fulfillment of the requirement for
the degree of Bachelor of Technology

In

Computer Science and Engineering

By

Ankush Kashyap (181202), Parul Damalu (181454),

UNDER THE SUPERVISION OF

Dr. Kapil Sharma



Department of Computer Science & Engineering and Information
Technology

**Jaypee University of Information Technology, Wagnaghat,
173234, Himachal Pradesh, INDIA**

CERTIFICATE

This is to certify that the work which is being presented in the project report titled “**Data hiding using Image Steganography**” in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by “Ankush Kashyap (181202), Parul Damalu (181454)” during the period from January 2022 to May 2022 under the supervision of **Dr. Kapil Sharma**, Assistant Professor (Senior Grade), Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Ankush Kashyap
(181202)

Parul Damalu
(181454)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Supervised by:



Dr. Kapil Sharma

Assistant Professor (Senior Grade)

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology

Dated:

AKCNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the project work successfully.

I really grateful and wish my profound my indebtedness to Supervisor **Dr. Kapil Sharma, Assistant Professor (SG)**, Department of CSE Jaypee University of Information Technology, Wakhnaghat. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr. Kapil Sharma**, Department of CSE, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straight forwardly or in a roundabout way in making the project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patience of my parents.

Ankush Kashyap

Parul Damalu

TABLE OF CONTENT

| <i>Content</i> | <i>Page No.</i> |
|--|------------------------|
| Chapter 1 <i>(Introduction)</i> | 1 -5 |
| Chapter 2 <i>(Literature Survey)</i> | 6-8 |
| Chapter 3 <i>(System Development)</i> | 9-24 |
| Chapter 4 <i>(Performance Analysis)</i> | 25-45 |
| Chapter 5 <i>(Conclusions)</i> | 46-49 |
| <i>References</i> | 50-51 |

LIST OF FIGURES

| | <i>Figure name</i> | <i>Page no.</i> |
|--------|--|-----------------|
| 3.1 | Bit plane in Image Steganography | 11 |
| 3.2 | Zig Zag Scanning | 11 |
| 3.3 | LSB representation | 12 |
| 3.4 | Steganography Model | 19 |
| 3.5 | Use case Diagram of Image Steganography | 20 |
| 3.6 | Data Flow Diagram Level 0 | 21 |
| 3.7 | Data Flow Diagram Level 1 | 22 |
| 3.8 | Data Flow Diagram Level 2 | 23 |
| 4.1 | Libraries Used | 25 |
| 4.2 | Image Input and Image data | 26 |
| 4.3 | PNG format Conversion Function | 26 |
| 4.4 | Secret Message function | 27 |
| 4.5 | Encoding Function and Encoding Cryptography | 27-28 |
| 4.6 | Decoding Function and Decoding Cryptography | 29-30 |
| 4.7 | Image Steganography (Encoding, Decoding) GUI | 30 |
| 4.8 | Phase one Encoding Window | 31 |
| 4.9 | Decoding Window | 31 |
| 4.10 | Example 1 Cover image (sunflower.png) | 32 |
| 4.10.1 | Example 1 Encoding | 33-35 |
| 4.10.2 | Example 1 Decoding | 35-38 |
| 4.11 | Example 2 Cover image (juit.jpg) | 39 |
| 4.11.1 | Example 2 Encoding | 39-40 |
| 4.11.2 | Example 2 Decoding | 41-45 |
| 5.1 | Stego image vs Original cover | 40 |

ABSTRACT

Due to a vast increase in security breaches and other violations, Data hiding has become a vital measure for the protection of the integrity of an individual or a group.

Image steganography is a perfect technique that exists for the protection of data from getting altered and getting safely transmitted. Steganography serves as an excellent method to secure information as the unauthorized party barely recognizes the secret message or a text that is hidden behind the original or the cover image.

In this project we have mainly focused on the Least Significant Bit (LSB) for hiding the texts behind the images and build our idea through a GUI. LSB is quiet simple and an efficient algorithm in which the least significant bit that barely changes the quality or the build of the original image is changed and is replaced by the secret text that we want to communicate to the other users.

Image preprocessing and generation of the output stego image is done by Pillow (PIL) and OpenCV in which image binary pixels are taken as input then they are altered with the secret text and then finally sent to the desired user. Using LSB in decoding side allows the receiver to extract the secret text bits easily to get the message.

The interest of this project is to explain the technique of hiding the secret data over the images as over the past few years Data security has become a vital thing. So due the high efficiency and the simplicity, LSB algorithm was chosen for our project.

Keywords: Data Hiding; Least Significant Bit; Data Security; Image Steganography; OpenCV; Pillow;

Chapter 01

INTRODUCTION

- 1.1 Introduction**
- 1.2 Problem Statement**
- 1.3 Objectives**
- 1.4 Methodology**
- 1.5 Technical Tools and Technology**

1.1 Introduction

Every organization or an individual has to transfer the data in a network or as end to end. The main aim of the communication is that the information that anyone is transferring or sending must be accurate, efficient and the most important factor, it must be “secured”. So to secure our data from any unauthorized accesses or grants, information security has various techniques of the data hiding.

Data hiding involves hiding or securing the secret data (text, image, and video, audio) inside the digital data (text, image, and video, audio).

Suppose there is one company that has to give its Bank Account details to the respective Bank. There are very high chances that hackers or other unauthorized users may track the

details and misuse the information provided by the company. So in such cases to securing the communication, data hiding is needed.

Steganography is one example of the Data hiding technique used in Information Security. The word steganography is made by Greek words “Steganos” which means cover and “Graph” that means to construct.

The word Steganography was first used in Golden age in Greece. It is believed that during the golden age, people of Greece used this technique to input the secret messages in woods. Steganography is of 4 major types i.e. Image Steganography (Secrecy in Images), Audio Steganography (Secrecy in Audio), Video Steganography (Secrecy in video), and Text Steganography (Secrecy in Text).

This project is on Image steganography in which a cover image or the original image is altered with the secret textual message that is to be sent to the receiver side. And then after successfully sending the cover image with text, the receiver decodes the secret text message from the cover image.

Image Steganography can be implemented by various algorithms such as LSB, Bit plane Algorithm, Spiral Embedding etc.

In our Project Image steganography is done by the most efficient and simple algorithm known as the Least Significant Bit substitution (LSB) algorithm. In LSB algorithm the last bit of the image pixel is substituted by the binary bits of the Secret messages. Original image altered with the secret message makes the Stego image, which is transferred to the receiver and the receiver will just extract out the least significant bits of the stego image and combine it to get the entire secret text.

Secret message transferring is also done by Cryptography techniques in which the original message is converted into the cipher text or unreadable text. This technique of cryptography still has chances of being cracked or hacked as the unauthorized party have an idea that something is being transferred and they can apply all sort of permutations to encode the cipher text but in Steganography the hackers barely have any idea about the

hidden data as the changes are not visible in the images in which secret message is being hidden.

So the ideal way of increasing the security of the communication is to use both cryptography and steganography together to make a strong communication process. So in this project we have designed our own cryptography algorithm named “Star-Dollar Algorithm”. We will discuss about this algorithm later on.

Steganography is in demand these days as it is very accurate, efficient method and serves as the most vital method for the secured sharing of the information.



1.2 Problem Statement

The main problem of sharing data between the users is interference of the unauthorized users or parties. Main aim of the communication over a network is that the data that is shared between the users must be secured and safely transferred.

So through our project we design a Steganography Algorithm that allows safe and secured data sharing.

1.3 Objective

The main purpose of the entire project on Data Hiding using Image Steganography is building a secured and a completely safe communication and data sharing. Main aim on which we focused during the entire project is to hide the secret textual data behind the images in an efficient manner so that the changes in the original images should not be visible to the naked eyes.

This is a vital and an interesting field of information security that has various useful and important applications like modern printers for encoding serial numbers, Intelligence services (FBI, RFIS) for secret communication, copyrights protection, watermarking etc.

1.4 Methodology

In our project on “Data hiding using Image steganography”, we used a very popular substitution algorithm named “*Least Significant Bit Substitution*” (LSB).

In this algorithm the least significant bit which in our project was the last binary digit of the pixel values is replaced with the Binary 0s and 1s of the secret message. The combined pixel binary bits and binary zeros and ones of the secret texts are encoded in the original image which is termed as the “*stego*” image. This digital image is transmitted to the desired receiver and the receiver decodes the stego image and extracts the last bits from it which are the least significant bits and contains the secret information or data that is sent by the sender.

1.5 Technical Tools and Technology

Software Requirements:

- Python 3.9

- Python Libraries
 1. Pillow (PIL)
 2. Numpy
 3. OpenCV (cv2)
 4. Tkinter(GUI)
 5. OS module

- Anaconda3 (Spyder)
- Operating system: Windows 10(recommended)

Hardware Requirements:

- Core i3 or higher,(cache- 3MB or 4MB recommended)
- Memory(RAM): Minimum 2GB; Recommended 4GB or above

Chapter 02

LITERATURE SURVEY

-Arun Kumar Singh, Juhi Singh, Dr. Harsh Vikram Singh, “Steganography in Images Using LSB Technique”.

This paper establishes a Last bit substitution technique for hiding the secret data in existing data. Data hiding in lossless formats is explained through this paper. LSB technique does not affect the visibility of the original cover image. Secret messages can be hidden in any sort of digital data like image, video, audio, text etc.

As the years are progressing, hackers are also increasing so it is quiet necessary to have a strong medium for communication where our data can remain safe. So to ensure security and safety of the data the steganography technique plays a vital role nowadays. Image steganography is hiding data in an image so that attackers cannot easily track down the data that is being transferred.

The last bit in a pixel must be changed to embed the secret message. So this technique is used to achieve the data hiding in information security. A secret message is embedded into the cover or the original image, then that message and image combined makes a stego image. This digital image is given to the receiver and then is decoded by the receiver to get the least significant bits and the corresponding secret text message.

Image steganography has numerous advantages. If any cover image contains the secret data, then also it is quiet tough to detect its presence even with the tools and software available to check the hidden data. No visible changes are seen, hence it is lossless, and therefore no data is lost in it.

Image steganography is very applicable these days. It is used in digital watermarking and fingerprinting, used in military services for transferring secret information.

-V. Lokeswara Reddy, Dr. A. Subramanyam, Dr. P. Chenna Reddy, “Implementation of LSB Steganography and its Evaluation for Various File Formats”.

This research paper explains the implementation of the lossless steganography techniques i.e. Lossy and Lossless techniques. Examples of Lossless compression formats are LSB in PNG.

Image steganography uses 8 bit image size for the implementation. The eighth bit inside an image is changed to a bit of the secret message. As the colored image pixels can cause complexity so every bit substitution is done in gray scale for all the 256 possible pixels. Last bit is altered and the secret information is hid in these bits. The altered image is called as stego image.

Various Steganography algorithms are available and out of those LSB is quiet simple, fast and an accurate algorithm for the implementation. LSB in PNG has one drawback that it is used to store the secret data of short length whereas the in GIF format, greater length texts can be stored. But GIF requires highly scaled Original pictures for implementation.

-Arvind Kumar, Km. Pooja, “Steganography- A Data Hiding Technique”.

Steganography is the art of hiding information and an effort to conceal the existence of the embedded information. Steganography is different concept as compared to the cryptography. Steganography hides the secret message behind the cover whereas cryptography makes the data or information unreadable and only the desired users have the key or the secret technique to decode that original message.

Various single users or the organizations require doing the secret communication over the internet. There are very high chances that the hackers or the unintended person or a group

can try to steal that vital or the confidential information. To prevent this image steganography plays a crucial role.

In steganography the data is hidden in the lossless covers that can be any text, image, audio or video and is not visible to the unauthorized person.

This technique is beneficial as the unauthorized parties cannot easily detect the secret transmission

Every algorithm used for image steganography whether LSB or any other algorithm, there is one thing in common that is every algorithm will work on hiding the data behind the cover images.

Lossless formats like PNG are used as the original images for hiding the data and transmission.

Image steganography is an interesting field due to various security breaches server hacks and stealing confidential data. Possible uses of the image steganography techniques are peer to peer communication, hiding data if a data breach or leak occurs, posting secret information over the internet etc.

Chapter 3

SYSTEM DEVELOPMENT

3.1 Analysis of the Algorithms

3.1.1 Least Significant Bit substitution

3.1.2 Cryptography: Star Dollar Algorithm

3.2 Pseudo Code/Steps of the Implementation

3.3 Model Development

3.3.1 Use case diagram of the Project

3.3.2 Data Flow diagram (DFD) of the project

3.1 Analysis of the Algorithms:

The basic principle of the information security involves the secured and the safe data transmission and communication. Steganography is one the most efficient and the popular technique that is used for the secured, safe and confidential communication and data sharing. Data hiding in information security is the best possible way for communicating safely.

Steganography can be done in various sorts of data like image, video, text or audio. Image steganography involves hiding information or secret data behind the images.

There are numerous algorithms used to achieve the data hiding using image steganography. Some of the popular algorithms are stated below:

1. LSB substitution: This is the most popular and the simplest way to achieve the data hiding. Every pixel has some range. Mostly the range lies between 0 and 255. Every single pixel value of RGB has some values lying between the range 0 and 255. Mostly the 8 bit binary representation of these pixels is done. So the representation in 8 bit gives us the 8 possible values of 0s and 1s. The last binary bit is termed as Least Significant bit i.e. the position 2 raised to the power 0 . Any changes done in this 8 bit binary representation is done by altering the last bit (least significant bit). There will not be major changes in the image pixel values after altering the last bit. Therefore the concept of lossless decomposition is also achieved. LSB technique works perfectly for the PNG format of the images as the PNG is lossless compression format.

2. BIT plane technique: Instead of changing the Gray scale pixels, this technique focuses on the contribution of every single bit for making the entire input image. Suppose we have an 8bit pixel having 8, 1-bit planes. Initial plane 0 is LSB plane that contains all the lowest order bits and the remaining bit plane 7 are MSB. So the division of the digital image to the relative planes is crucial for determining the contribution made by the bit planes for building up the entire digital image. Planes will determine adequacy of the bits and importance order of the bit. Finally the bits having lesser contribution or the lower order bits will be replaced by the secret message bits so as to achieve the lossless decomposition.

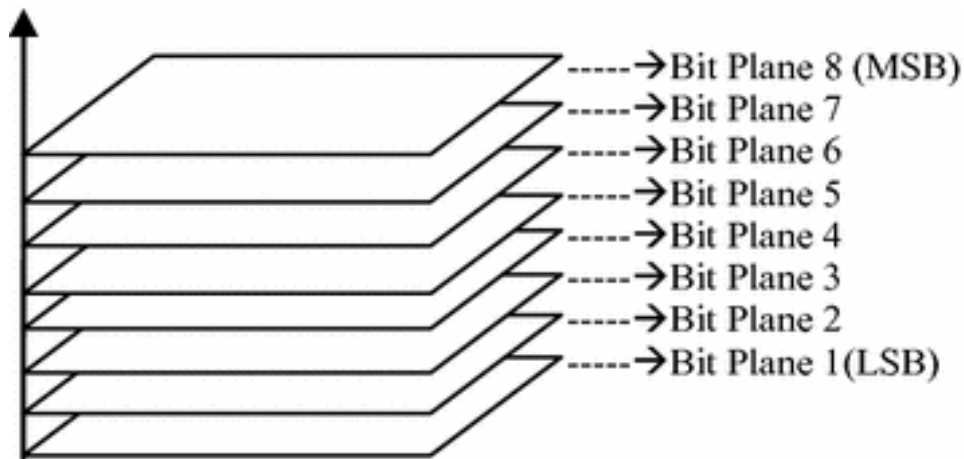


Fig 3.1 Bit plane in Image Steganography

3. Zigzag Scanning: Zig zag scanning is also a method that is used to implement the Image steganography. The secret message will be converted into the 8 binary bits of 0s and 1s. This secret message will be hidden in the image pixels in a way which will only be known to the sender and receiver. Secret message bits could be altered in any 8 bit binary spaces.

It is similar to the “Key” concept used in the cryptography. As the sender and the receiver have the information about the key for encoding decoding, similarly the zig zag pattern will only be known to the sender, receiver to prevent the leaks and hacks.

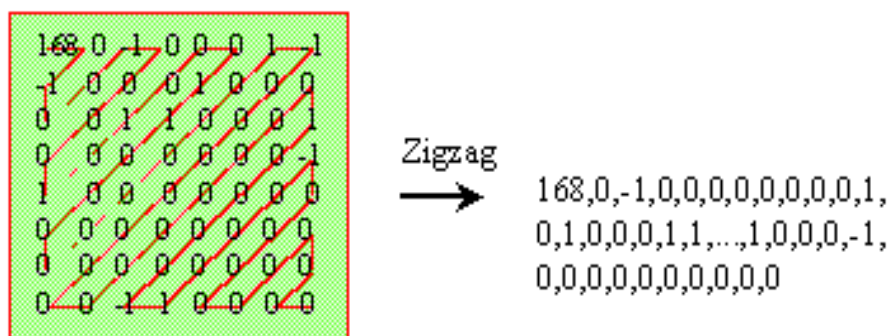


Fig3.2Zig Zag Scanning

3.1.1 Designing the LSB substitution Algorithm

Least Significant bit substitution algorithm is the most efficient and the simple substitution technique used in data hiding and steganography.

Least Significant bit (LSB) is the last bit of the Binary representation of the number. Basically 2 raised to the power 0^{th} bit will be the Least significant bit.

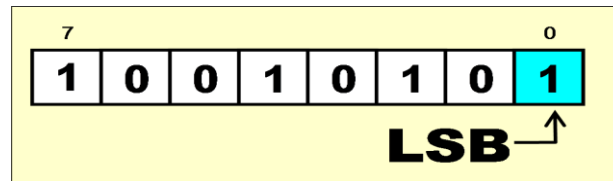


Fig3.3LSB representation

So the basic idea behind the LSB embedding is that the last bit of the binary representation will be replaced by the binary bits of the Secret message that is to be encoded in the input image.

LSB substitution has 2 stages: **ENCODING and DECODING**

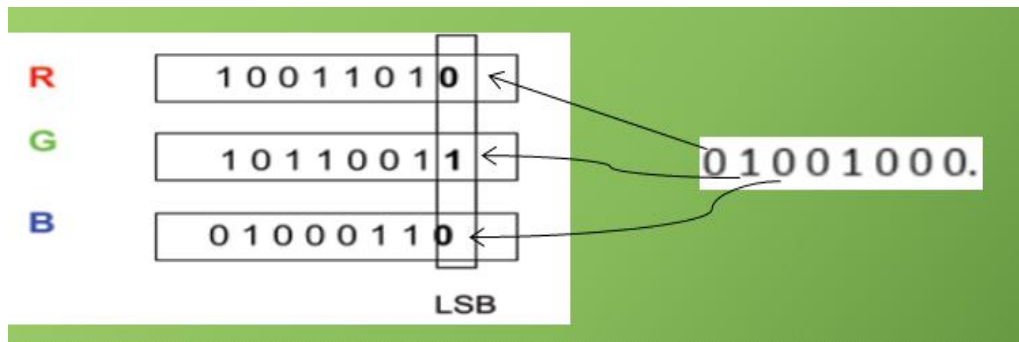
LSB ENCODING: This phase of the algorithm involves the sender's part. The main aim of the encoding is to alter the least significant bit and adding the secret message bits to it.

Let us consider an example:

- Suppose there is the secret Message "H" that is to be encoded in the random RGB pixels given below:

R = 1 0 0 1 1 0 1 1
G = 1 0 1 1 0 0 1 0
B = 0 1 0 0 0 1 1 0

- ASCII value of the secret message which here is the letter “H” is 72. The 8 bit binary representation of 72 will be **0 1 0 0 1 0 0 0**.
- So the task is to encode the 8 bit binary digits of 72 in the given RGB pixels. Below will be the basic idea of the bit alteration. And then the altered bit pixels will be sent to the receiver.



LSB DECODING: This is the algorithm that concerns the receiver. After the successful sending of the stego image containing the altered RGB pixel bits, the task of the receiver is to extract the least significant bits of the stego image pixels and combine them in the pairs of 8 and then get the desired 8 bits.

Again taking the above example:

- The receiver will be receiving the stego image pixels (Secret message + original image). Suppose the following RGB pixels will be received:

1 0 0 1 0 1 0 0, 1 0 0 1 0 1 0 1, 1 0 0 1 0 1 0 0,
1 0 0 1 0 1 0 0, 1 0 0 1 0 1 0 1, 1 0 0 1 0 1 0 0,
1 0 0 1 0 1 0 0, 1 0 0 1 0 1 0 0, 1 0 0 1 0 1 0 0

- So the red colored bits in the above picture are least significant bits of the stego image pixels.

The aim of the decoding is to take these red colored bits individually, combine them and then find out the decimal number of the that 8 bit representation. The decimal number calculated will be the ASCII code of the secret message that is embedded in the original cover image.

The red bits combined will look like:



And finding the Decimal number of the above bits will give the value 72 which is the ASCII code of the letter “H”

Finally the decoded word will be the letter H.

So the LSB substitution Image Steganography without majorly changing the image quality, successfully allows the data hiding process. It is mainly used for the Lossless Cover image format such as PNG.

3.1.2 Cryptography: Star Dollar Algorithm

Cryptography is the most crucial technique that can be used to make the security more advanced. This involves conversion of the text into the unreadable or the cipher text. Encryption involves Plain to Cipher text conversion and Decryption or decoding involves Cipher to Plain text.

Star-Dollar Cryptography Algorithm

It is our own cryptography algorithm that is mainly based on the monoalphabetic substitution where a single alphabet is replaced by dollars (\$) and star (*) and an integer 1 at the end and for the small letters 2 at the end.

STEPS INVOLVED IN ENCODING ALGORITHM:

1. To convert the plain text to cipher text, we take help from the table that contains the respective values of keys for every alphabet. In the table each letter has either a "*" or "\$" ended with an integer 1 for the capital letters and for small letters it is ended with an integer 2 as the output for the cipher text.
2. The first letter has a single '*' and this value of the '*' is increasing for the letters after one gap terminated with the integer 1. This is similar in the case of the dollars where value of dollar is increasing after one gap terminated with the integer 2.
3. For the spaces in between the message or the words, the fix characters are used i.e. "\$*1"
4. For the numeric values in the input message, the respective cipher text will be in the order of "#3". The total "#" depends on the integer. Example for 4 it would be "####3". 3 is used for separation that will be needed while decoding.

STEPS INVOLVED IN DECODING ALGORITHM:

1. For decoding any character from the cipher text, the integers that were used for the separation in the encryption phase will be traversed.
2. Possible integers in the encryption phase were 1, 2, and 3.
3. Loop is started and whenever these integers (1,2,3) are found in the traversing loop, the string up to which the numbers are found are matched with the values present in the table for the corresponding character.

Consider an example where plain text is “ANKUSH”

So to encode this we will individually substitute the letters from the table as following:

- For the first alphabet i.e. **A** it is **single ‘*1’**
- For The next letter i.e. **N** it would be **‘\$\$\$\$\$1’**
- Similarly for the next letter i.e. **K** the key symbol encrypted would be **‘*****1’**
- After repeating for every letters of “ANKUSH”, the final encrypted cipher text will be:

1\$\$\$\$\$1**1*****1*****1\$\$\$\$\$1**

For decoding this above cipher text:

- The unreadable string will be traversed and whenever the integer 1,2 or 3 is found the loop is stopped and read from the beginning.
- In the above encrypted ANKUSH string, firstly we traverse from * and then after 1 is found the loop is stopped and is compared with the table values for the characters. “*1” will give “A”.

- Again now \$ will be traversed and after traversing 7 “\$” the integer 1 comes and loop is again stopped and corresponding value of the string is checked which will corresponds to “N”
- Above steps are repeated and finally “ANKUSH” will be returned as original text.

Table for Alphabets and their respective Symbol Key

| <i>Plain Text Letter</i> | <i>Key Symbol</i> | <i>Plain Text Letter(Small)</i> | <i>Key Symbol</i> |
|--------------------------|-------------------|---------------------------------|-------------------|
| A | *1 | a | *2 |
| B | \$1 | b | \$2 |
| C | **1 | c | **2 |
| D | \$\$1 | d | \$\$2 |
| E | ***1 | e | ***2 |
| F | \$\$\$1 | f | \$\$\$2 |
| G | ****1 | g | ****2 |
| H | \$\$\$\$1 | h | \$\$\$\$2 |
| I | *****1 | i | *****2 |
| J | \$\$\$\$\$1 | j | \$\$\$\$\$2 |
| K | *****1 | k | *****2 |
| L | \$\$\$\$\$\$1 | l | \$\$\$\$\$\$2 |
| M | *****1 | m | *****2 |
| N | \$\$\$\$\$\$\$1 | n | \$\$\$\$\$\$\$2 |
| O | *****1 | o | *****2 |
| P | \$\$\$\$\$\$\$1 | p | \$\$\$\$\$\$\$2 |
| Q | *****1 | q | *****2 |
| R | \$\$\$\$\$\$\$1 | r | \$\$\$\$\$\$\$2 |
| S | *****1 | s | *****2 |
| T | \$\$\$\$\$\$\$1 | t | \$\$\$\$\$\$\$2 |
| U | *****1 | u | *****2 |
| V | \$\$\$\$\$\$\$1 | v | \$\$\$\$\$\$\$2 |
| W | *****1 | w | *****2 |
| X | \$\$\$\$\$\$\$1 | x | \$\$\$\$\$\$\$2 |
| Y | *****1 | y | *****2 |
| Z | \$\$\$\$\$\$\$1 | z | \$\$\$\$\$\$\$2 |

3.2 Pseudo Code/Steps of the Implementation

The Implementation of the project involves 2 phases: Encoding and Decoding. The major steps involved in implementation of the LSB Image steganography are:

1. Importing the libraries (Pillow, Tkinter, numpy, OpenCV, OS).
2. Converting the input image into PNG (lossless) format.
3. Getting the input image pixels and conversion of pixels to 1D array.
4. Encoding the message bits by changing the least significant bit.
5. Creation of the Stego Image.
6. Forwarding the stego image to receiver.
7. Decoding the stego image pixels' least significant bit.
8. Combining the LSB of stego and finding the decimal number of it.
9. Decimal number will give ASCII code and decoded message will be visible to the receiver.

3.3 Model Development

Basic Image steganography model has 2 Phases i.e. encoding and the decoding phase. Encoding phase involves creation of the “stego” image that has the original cover picture pixels embedded with the secret message bits at the least significant bits.

Then there is the second phase of steganography that has decoding part. It deals with the decoding of the stego image pixels for getting the least significant bits of the stego image which will be the secret bits embedded inside it.

Basic model will look like this:

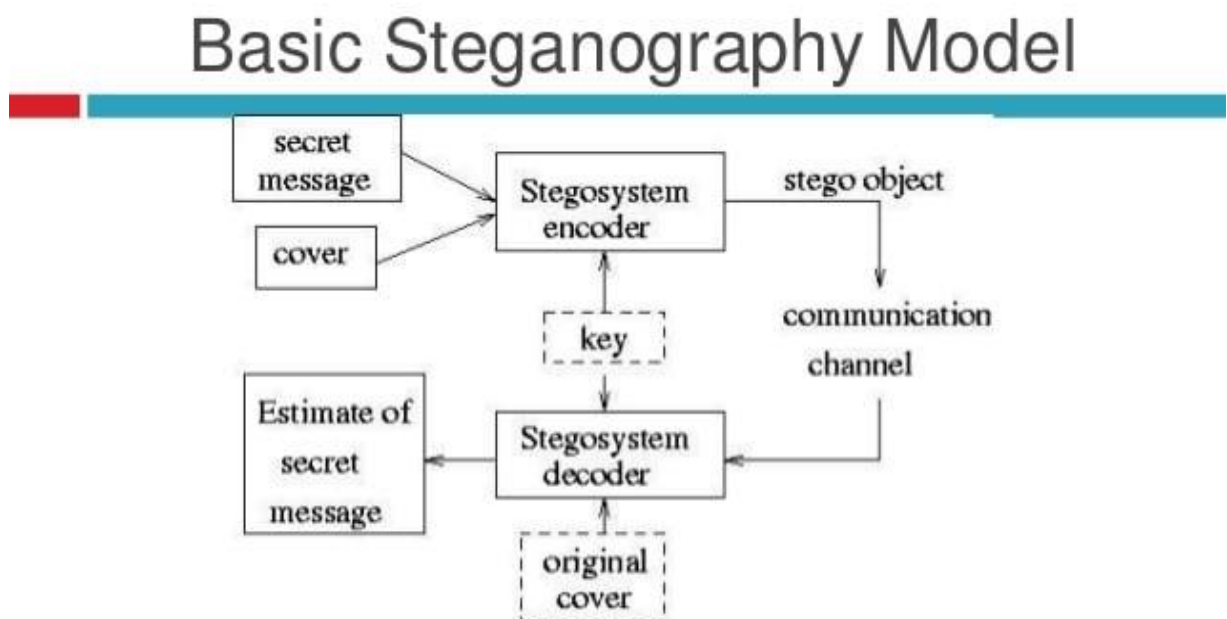


Fig3.4Steganography Model

So the phase 1 will be encoding the secret text in the image. Then that image and the secret bits will be passed on to the Stegosystem Encoder. The combined secret message and the image will give the rise to the stego object or the stego image. This stego object will then be forwarded via any network or a communication medium to the Stegosystem Decoder.

After the arrival of the stego object at the Stegosystem decoder, Phase 2 of the entire Image steganography will start.

There is an optional step involved in the encoding and decoding that is similar to the cryptography. The sender can encode the secret text in a human unreadable form or as a cipher text. And sender will also send the key attached with the stego object so that the receiver can decode the secret message as well as convert that message into human readable form through the key. So it is secured as the key will only be shared with the sender and the receiver and no outer unauthorized interventions would be possible.

3.3.1 Use Case diagram of the Project

USE CASE DIAGRAM IMAGE STEGANOGRAPHY

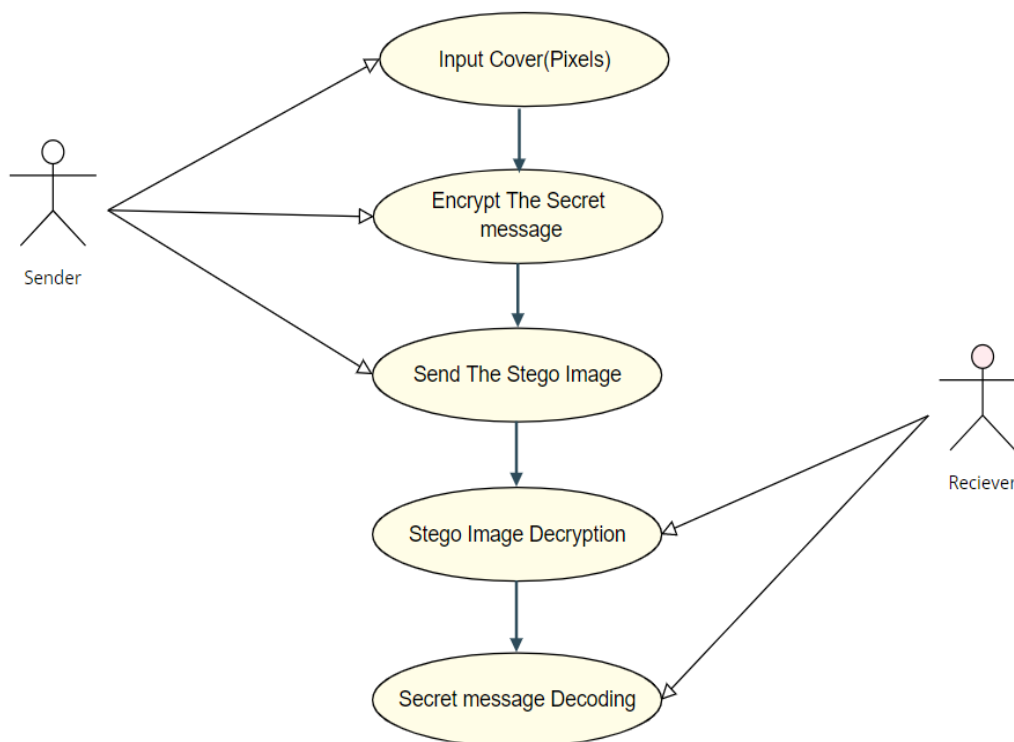


Fig3.5 Use case diagram of Image Steganography

In the above user case diagram, the user or the sender provides the input cover image (pixels) and then encrypts the secret message for sending to the receiver on the other side. After the successful encryption of the secret message and cover image, Stego image is forwarded. The Receiver will receive the stego image that has the secret message to be decoded. The receiver will decrypt the image pixels; extract out the last significant bits and then Decode the Secret message successfully.

3.3.2 Data Flow diagram (DFD) of the Project

Data Flow Diagram of IMAGE STEGANOGRAPHY

DFD LEVEL 0

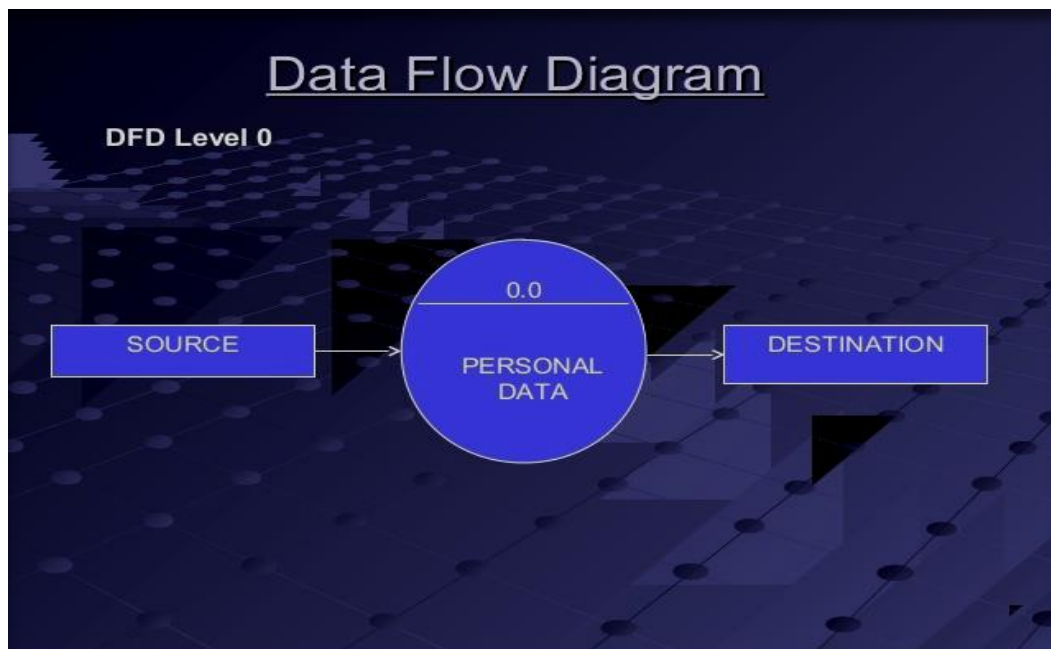


Fig3.6Data Flow Diagram Level 0

From the above DFD diagram, it is clear that the main aim is to send a secret message or information from a source to the desired destination securely.

DFD LEVEL 1

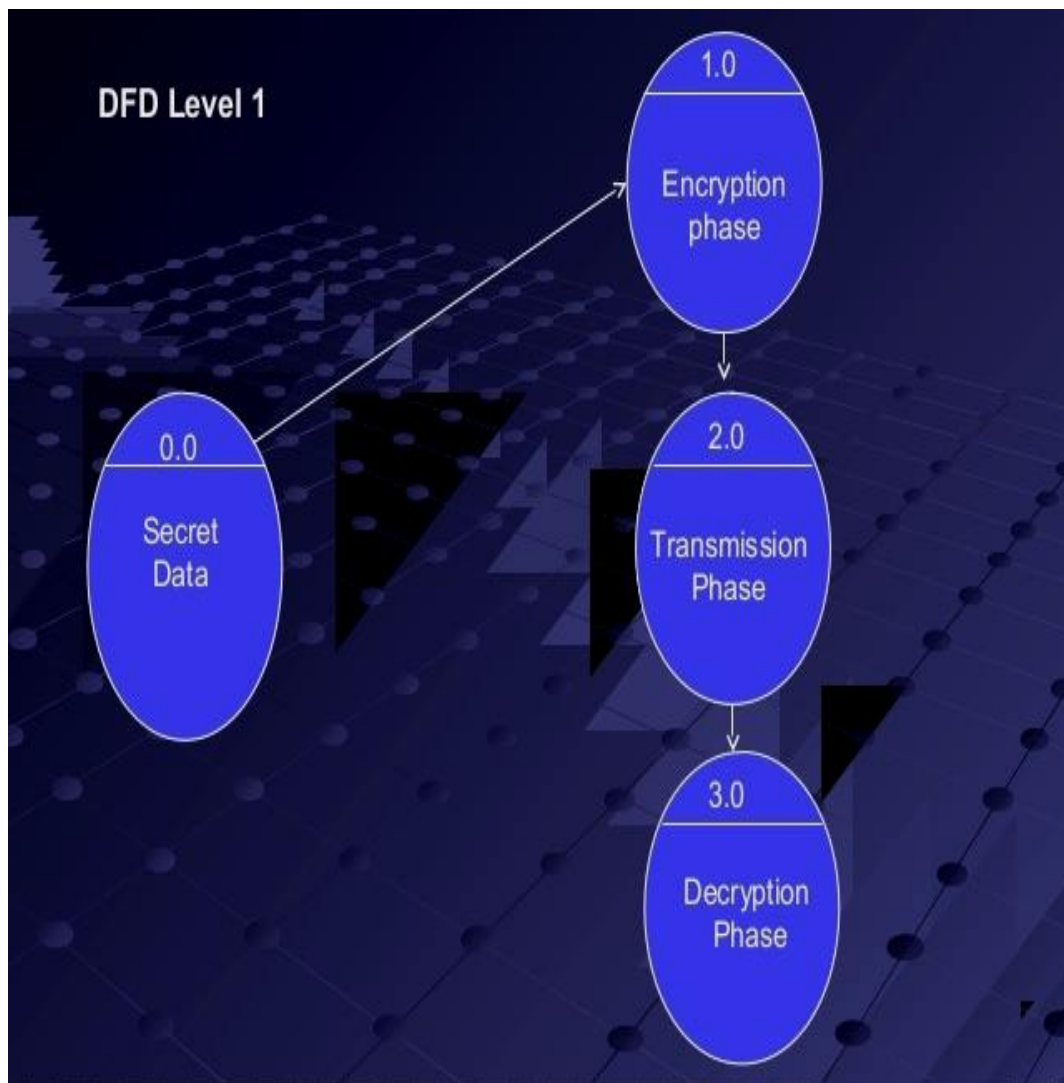


Fig3.7 Data Flow Diagram Level 1

The above DFD diagram signifies the different phases involved in the Image steganography algorithms. Mainly there are 3 phases: Encryption, Transmission and Decryption Phase.

Encryption Phase deals with the encrypting the secret message bits in the cover image pixels and building up of the stego image.

Transmission phase includes the sending or transferring of the stego image or the stego object to the receiver.

Decryption involves taking the last bits to get the text that was sent by the sender.

DFD LEVEL 2

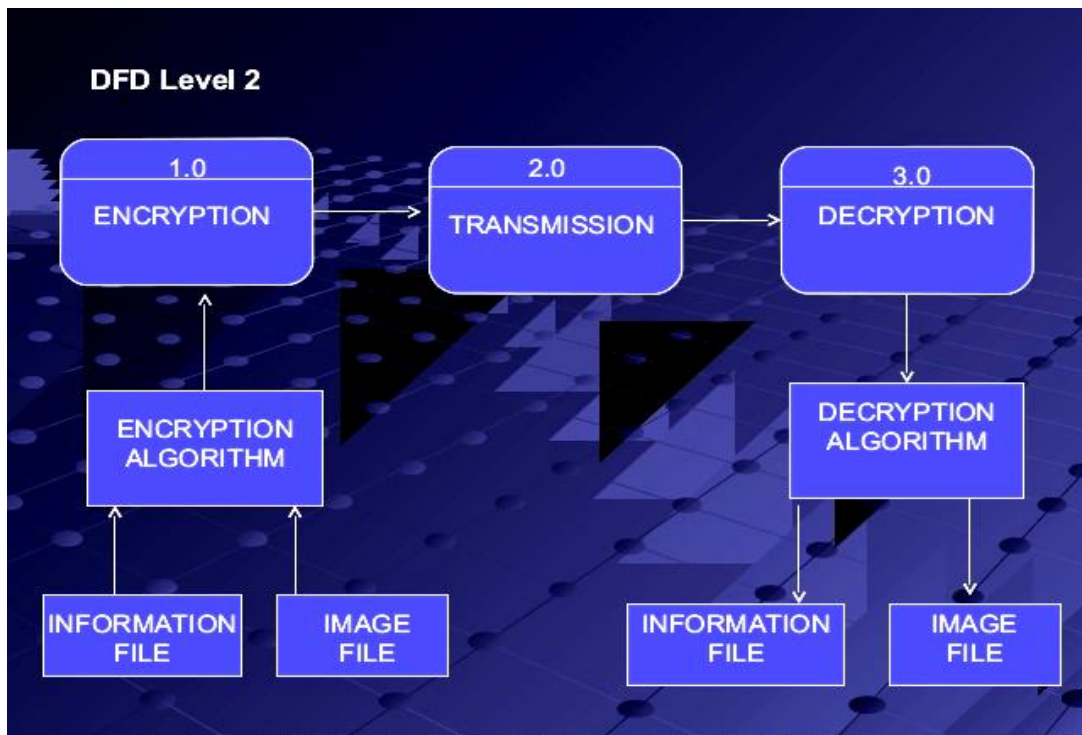


Fig3.8 Data Flow Diagram Level 2

The above DFD diagram is the main working of the project (Combination of the Level 0 and 1 DFD).

2 main phases Encoding and Decoding with the additional phase Transmission phase is involved in the Level 2 DFD of the Image steganography. In the first phase the user or the sender inputs the image file and the information or the secret message file. The Encryption algorithm works which encrypts the secret text and cover image pixel together making the stego object. Then in the Second phase there is Transmission of the Stego image to the decoder i.e. to the receiver side.

In the next third phase, the receiver decodes the secret message by extracting out the LSB of the stego image pixels.

Chapter 4

Performance Analysis

The results of Image steganography project of ours works perfectly for the process of data hiding. Any cover image if not in PNG format, the function designed to convert images to PNG format does the work. The change in the original cover and stego object is negligible as last bit is changed only. So it is lossless form of compression and embedding. So the results and transmissions are highly accurate.

➤ **Screenshots of the various stages of the project**

```
1  from tkinter import *
2
3  from PIL import Image
4  import numpy as np
5  from cv2 import cv2
6  import os
```

Fig4.1 Libraries used

```

23 def imgData():
24
25     a = d_path
26     global dataSize, data_1d
27     image = Image.open(a)
28     image = np.array(image)
29
30     print("Original list of pixels", image)
31     dataSize = image.shape
32
33     data_1d = image.ravel()
34     print("\nPixels in Input image:", data_1d)
35

```

Fig4.2 Image input and Image data

```

13 def img_ConvToPNG():
14     global s_path, d_path
15     s_path = str(l.get())
16     d_path = str(m.get()) + c.get()
17     print(s_path)
18     print(d_path)
19
20     im = Image.open(s_path)
21     im.save(d_path)

```

Fig4.3 PNG format conversion Function


```

37 def message():
38     inp = inputtxt.get("1.0", "end-1c")
39     mytext = inp
40     global binarytxt
41
42     length = str(bin(len(mytext)))
43     print("Messasge length=",length)
44
45     length = length[2:].zfill(10)
46     print("\n10 bit message length=",length)
47
48     binarytxt = str(bin(int.from_bytes(mytext.encode(), 'big')))
49     print("\nBinary coversion of message from ascii =",binarytxt)
50
51     binarytxt = length + binarytxt[2:]
52     print("\nConactenation 10bit msg + Ascii's Binary=",binarytxt)
53
54     binarytxt = [int(x) for x in list(binarytxt)]
55     print(binarytxt)
56

```

Fig4.4 Secret Message Function

```

67
68 def encode():
69
70     for i in range(
71         len(binarytxt)):
72         val = data_1d[i] % 2
73         if binarytxt[i] == 1:
74             if val == 1:
75                 pass
76             else:
77                 data_1d[i] = oddConvrt(data_1d[i])
78         else:
79             if val == 0:
80                 pass
81             else:
82                 data_1d[i] = evenConvrt(data_1d[i])
83
84     global stegoimg
85     stegoimg = data_1d.reshape(dataSize)
86     print("\n3d array of Msg+ Pixels",stegoimg)
87

```

```

59 def ch2ciph(c):
60     if c == ' ':
61         return "$1"
62     elif c.isdigit():
63         val=ord(c)-ord('0')+1
64         x='#'*val+'3'
65         return x
66     elif c.isupper():
67         val=ord(c)-ord('A')
68         x=''
69         if val%2==0:
70             x='*'*((val+2)//2)
71         else:
72             x='$'*((val+1)//2)
73         x=x+'1'
74         return x
75     else:
76         val=ord(c)-ord('a')
77         x=''
78         if val%2==0:
79             x='*'*((val+2)//2)
80         else:
81             x='$'*((val+1)//2)
82         x=x+'2'
83         return x
84
85 def mytxt2ciph(text):
86     cipher=''
87     for c in text:
88         cipher=cipher+ch2ciph(c)
89     return cipher
90

```

Fig4.5 Encoding Function and Encoding Cryptography

```

90 def decode():
91     global dec_path
92     dec_path = str(ds.get())
93     image = Image.open(dec_path)
94     image = np.array(image)
95
96     data_1d = image.ravel()
97     print("\n1d array of stego 3d list:",data_1d)
98
99     strData = ""
100    length = ""
101
102    for i in range(10):
103        val = data_1d[i] % 2
104        length += str(val)
105    print("\n10 bit stego msg length:",length)
106
107    length = '0b' + length
108    print("\nBinary 10 bit: ",length)
109
110    length = int(length, 2)
111    print("\nDecimal value of binary 10 bit length:",length)
112
113    length_1 = length * 8 + 9
114    print ("\nTotal pixel length used(Msg bits+ reserved msg length bits):", length_1)
115
116    for i in range(10, length_1):
117        val = data_1d[i] % 2
118        strData += str(val)
119    print("\nBinary of Ascii msg bits:",strData)
120
121    strData = '0b' + strData
122    print("\nBinary identifier **0b**:", strData)

```

```

91 def ciph2ch(ciph,n):
92     x=len(ciph)
93     if ciph=='*%':
94         return ' '
95     elif n=='1':
96         if ciph[0]=='*':
97             return chr(x*2-2+65)
98         else:
99             return chr(x*2-1+65)
100    elif n=='2':
101        if ciph[0]=='*':
102            return chr(x*2-2+97)
103        else:
104            return chr(x*2-1+97)
105    else:
106        return chr(x-1+48)
107
108 def ciph2txt(cipher):
109     ciph=''
110     text=''
111     i=0
112     while i < len(cipher):
113         while cipher[i]!='1' and cipher[i]!='2' and cipher[i]!='3':
114             ciph=ciph+cipher[i]
115             i=i+1
116         text=text+ciph2ch(ciph,cipher[i])
117         i=i+1
118         ciph=''
119     return text

```

Fig4.6 Decoding Function and Decoding Cryptography

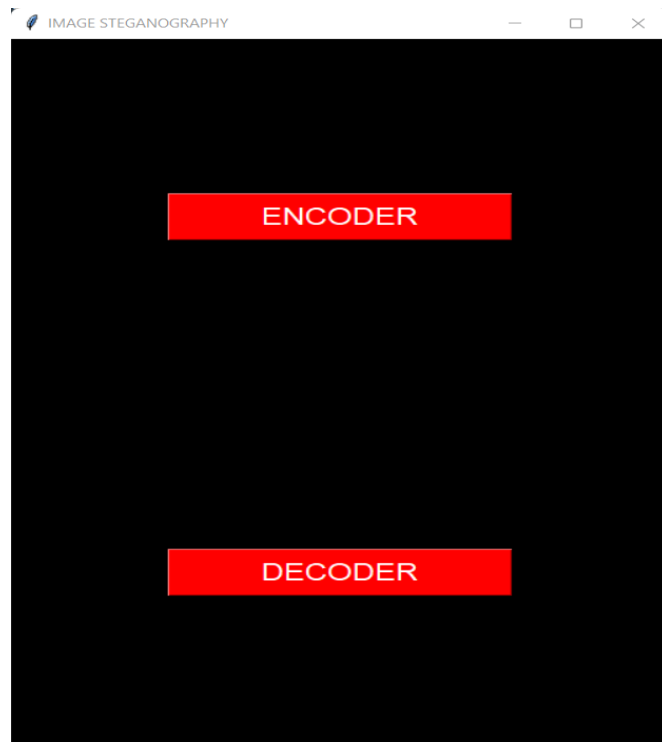


Fig4.7 Image Steganography (Encoding, Decoding) GUI

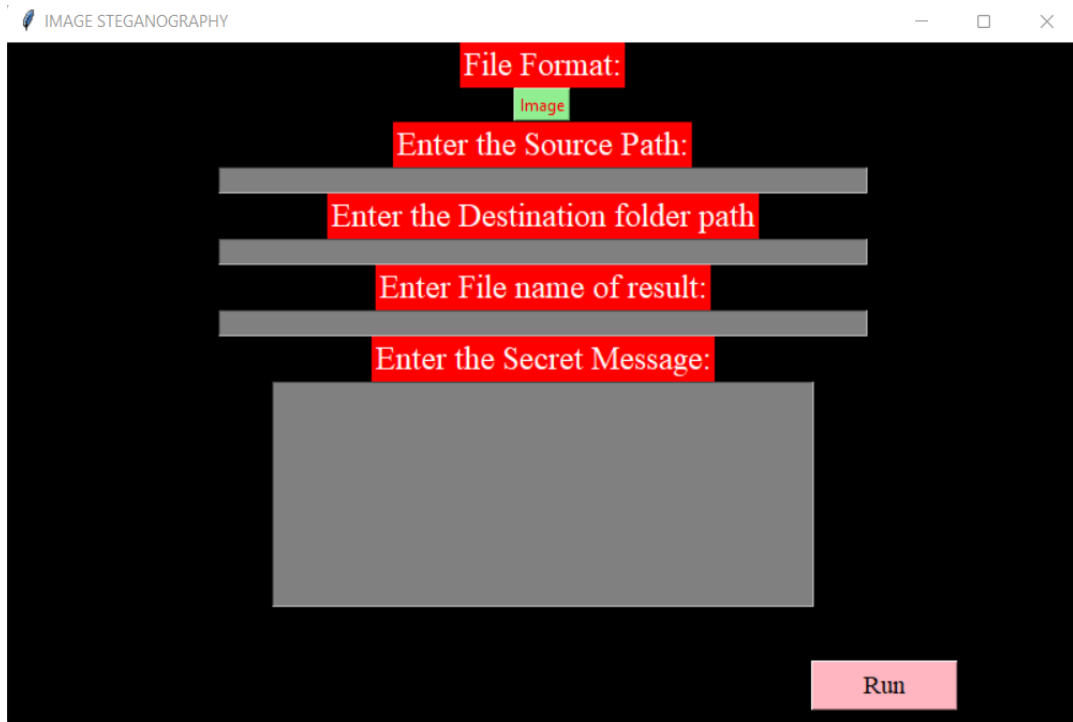


Fig4.8 Phase one Encoding Window



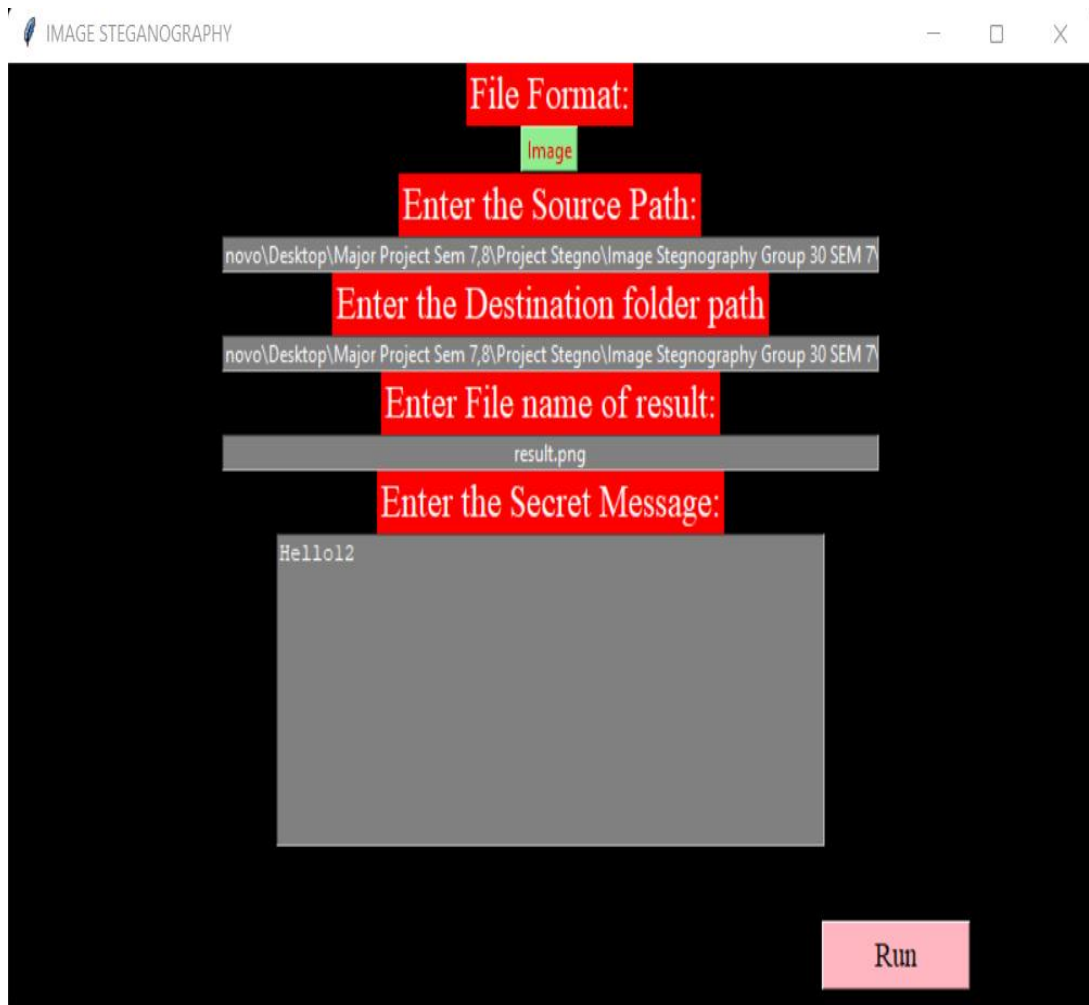
Fig4.9 Decoding Window

- Some examples with output:

Example 1:



Fig4.10 Example 1 Cover image (sunflower.png)



Encoding Window

```

1d array of stego 3d list: [128 178 232 ... 128 129 98]

10 bit stego msg length: 0000100111

Binary 10 bit: 0b0000100111

Decimal value of binary 10 bit length: 39

Total pixel length used(Msg bits+ reserved msg length bits): 321

Binary of Ascii msg bits:
0100100001001000010010000100100001100010010101000101010001010100011001000100100001001
0000100100001001000010010000100100001100100010010000100100001001000010010000100100001
0010000110010001010100010101000101010001010100010101000101010001010100010101000110010
00100011001000110011001100100011001000110010001100110011

Binary identifier **0b**:
0b01001000010010000100100001001000011000100101010001010100010101000110010001001000010
0100001001000010010000100100001001000011001000100100001001000010010000100100001001000
0100100001100100010101000101010001010100010101000101010001010100010101000101010001100
1000100011001000110011001100100011001000110010001100110011

Decimal conversion of entire message bits:
1177934027261096298379632843147147257576882102189893138968418966598374885855785685668
080067379
310

***So the Hidden message in photo is*** : $$$1***2$$$$$2$$$$$2*****2##3###3

```



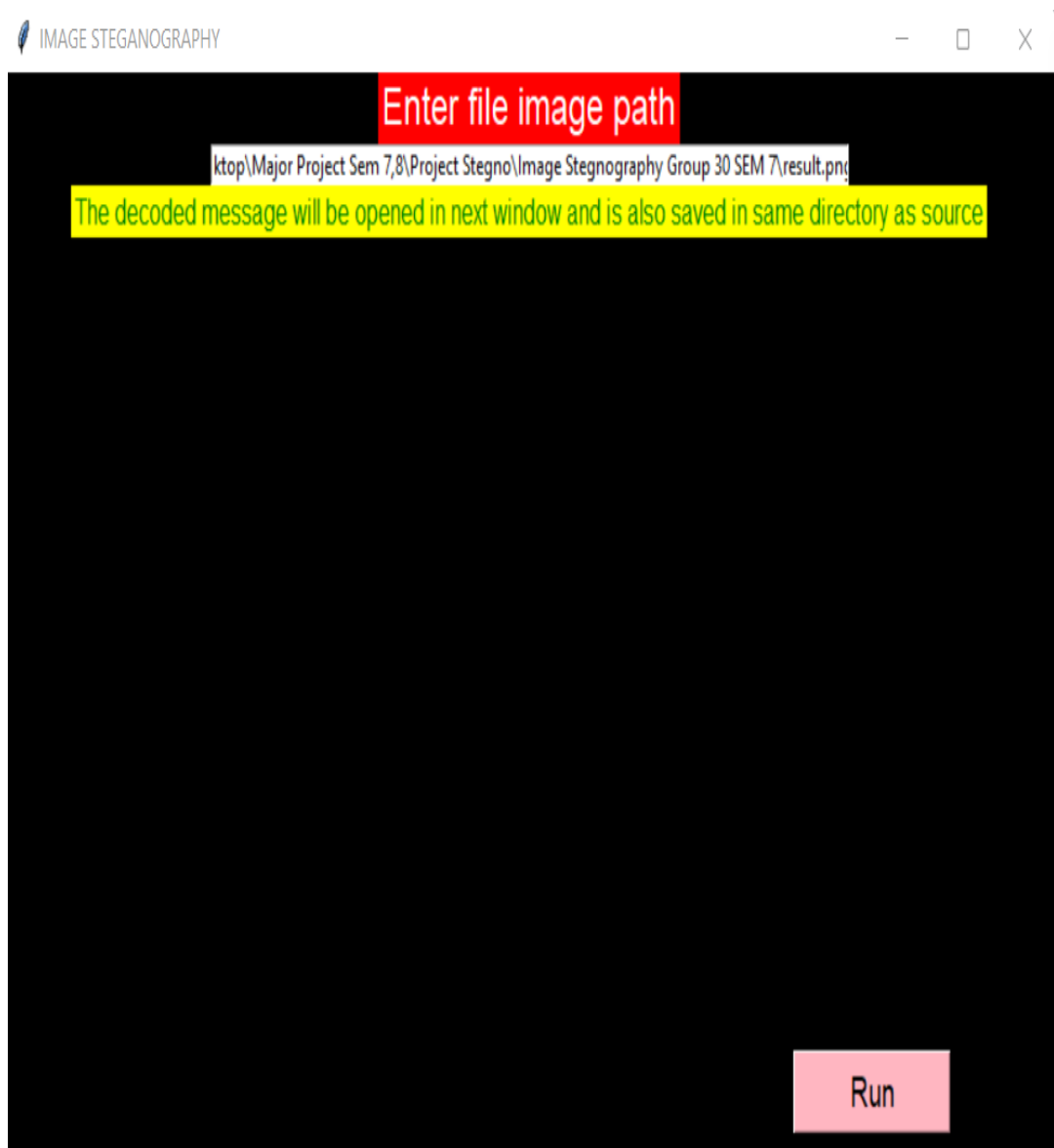
| | | | |
|---|---------------------|----------|----------|
|  Result.png | 11/29/2021 10:10 PM | PNG File | 1,901 KB |
|  sunflower.png | 7/10/2021 11:51 AM | PNG File | 1,901 KB |

Fig4.10.1 Example 1 Encoding



Decoding Window

The message

\$\$\$1**2\$\$\$\$\$2\$\$\$\$\$2*****2##3##3

(Secret Cipher Text)

The message

Hello12

(Decoded Cipher text i.e. Original Text)

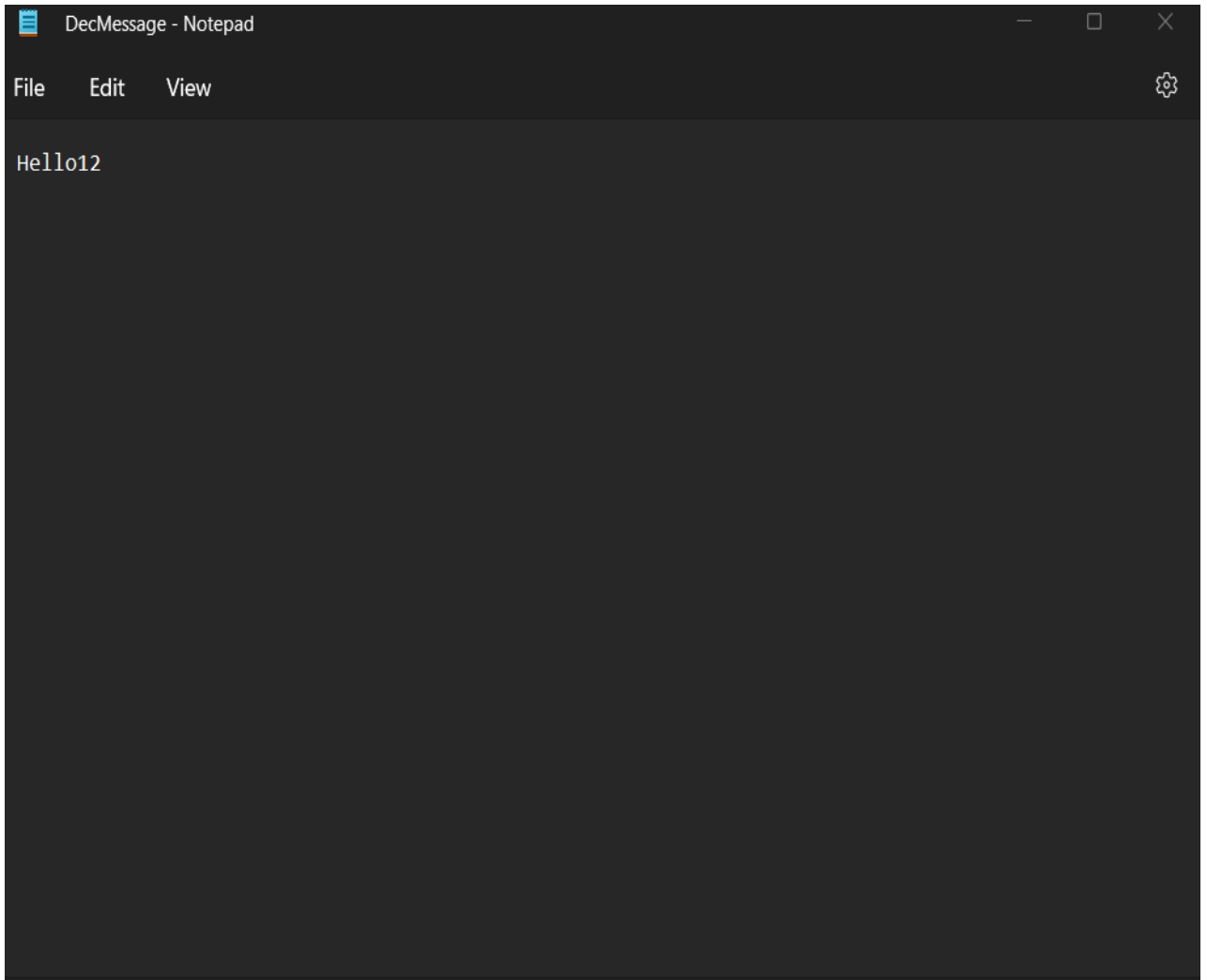
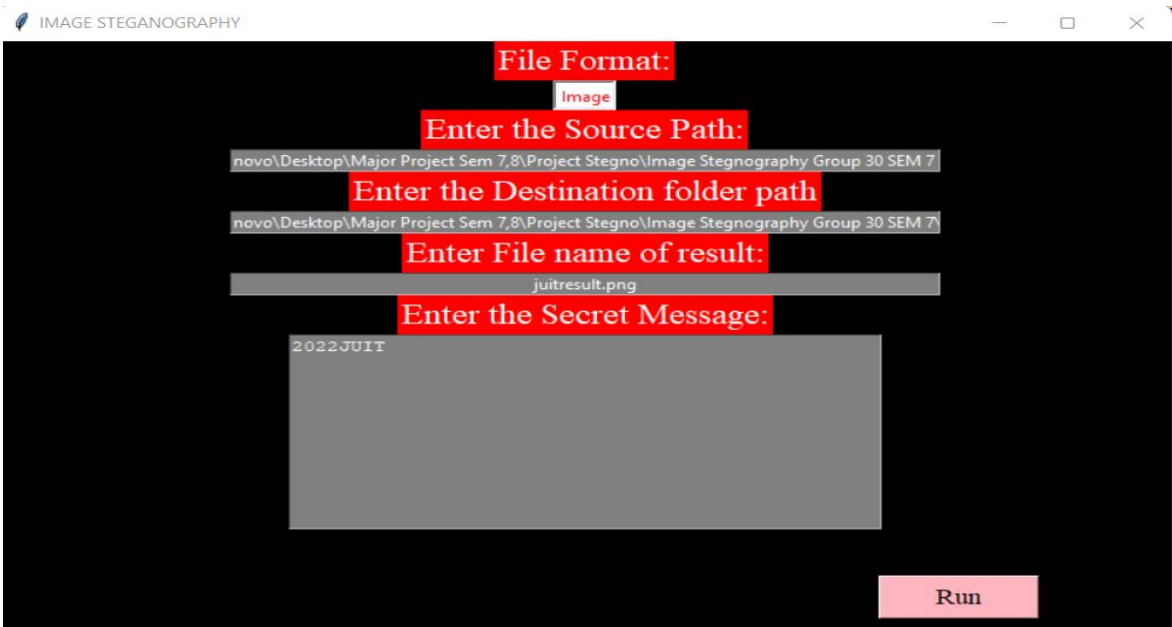


Fig4.10.2 Example 1 Decoding

Example 2:



Fig4.11 Example 2 Cover image (juit.jpg)



```

Pixels in Input image: [32 53 82 ... 16 14 19]
2612400
###3###3###3###3$$$$$1*****1*****1$$$$$$$$1
Messasge length= 0b110001

10 bit message length= 0000110001

Binary coversion of message from ascii =
0b10001100100011001000110011001100100011001100110010001100100011001000110011001100100
0110010001100100011001100110010010000100100001001000010010000100100001100010010101000
1010100010101000101010001010100010101000101010001010100010101000101010001010100011000
10010101000101010001010100010101000110001001001000010010000100100001001000010
0100001001000010010000100100001001000010010000110001

Conactenation 10bit msg + Ascii's Binary=
0000110001010001100100011001000110011001100100011001100110010001100100011001000110011
00110010001100100011001000110011001100100100001001000010010000100100001001000011000010
0101010001010100010101000101010001010100010101000101010001010100010101000101010001010
1000110001001010100010101000101010001010100010101000110001001001000010010000100100001
001000010010000100100001001000010010000100100001001000010010000110001

```

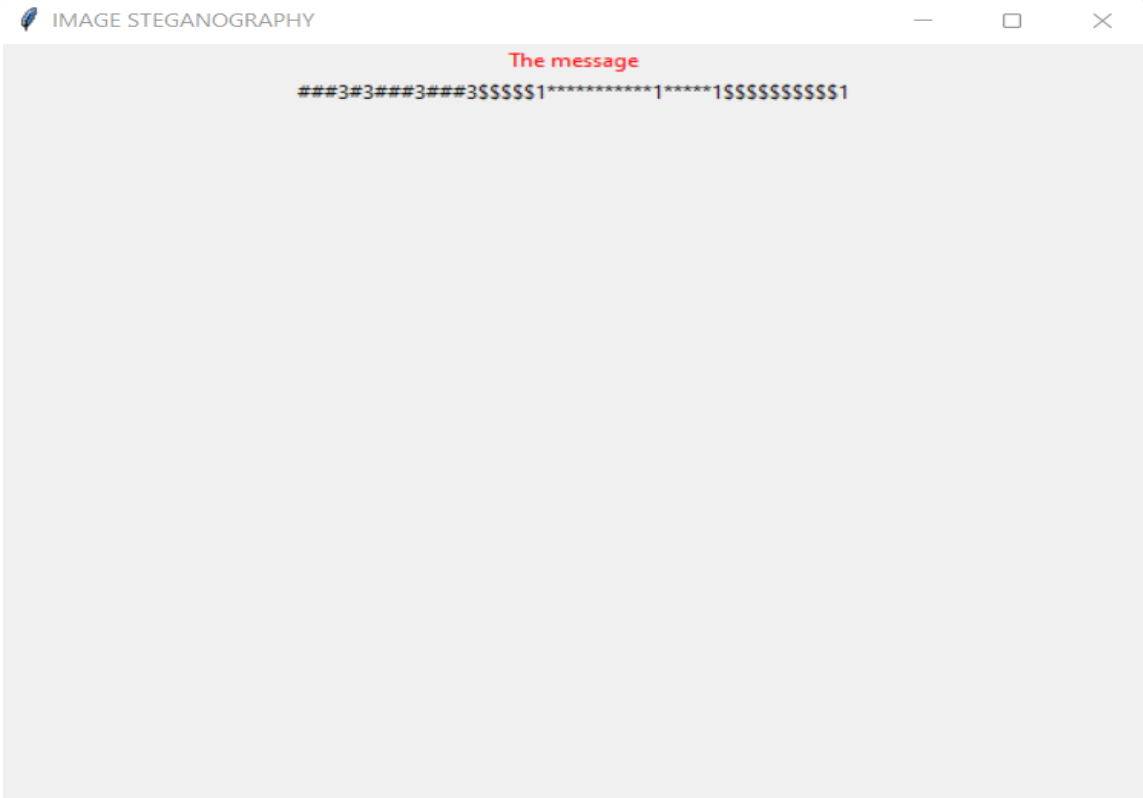
Fig4.11.1 Example 2 Encoding

Enter file image path

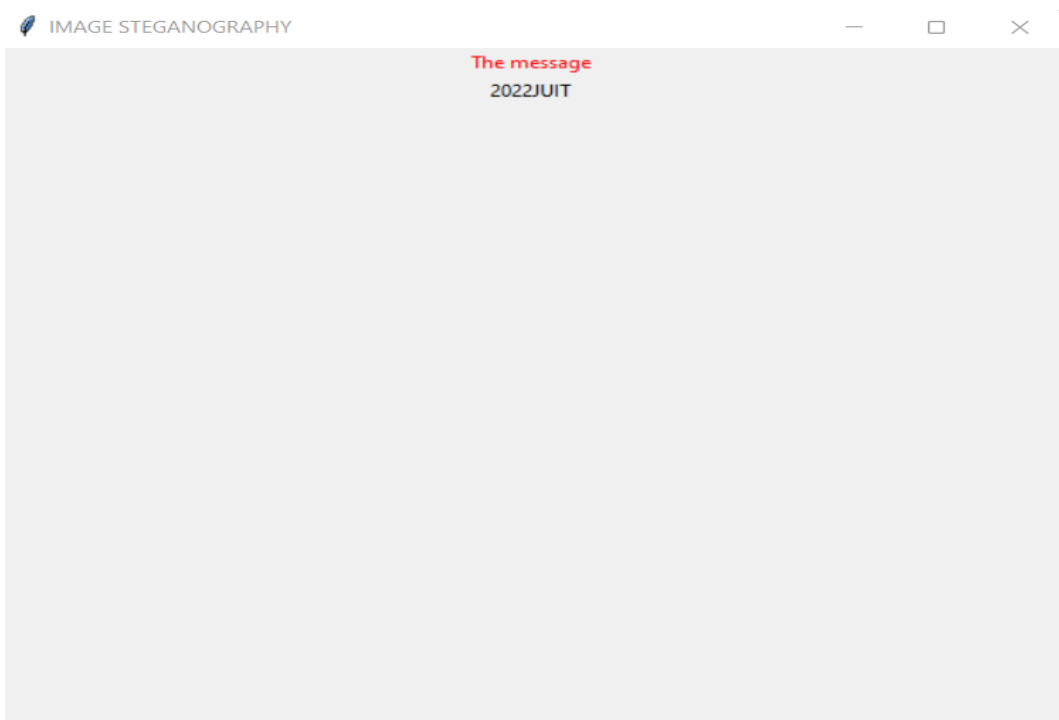
p:\Major Project Sem 7\Project Stegno\Image Steganography Group 30 SEM 7\juitresult.png

The decoded message will be opened in next window and is also saved in same directory as source

Run



(Secret Cipher Text)



(Decoded Cipher text i.e. Original Text)

```
10 bit stego msg length: 0000110001
```

```
Binary 10 bit: 0b0000110001
```

```
Decimal value of binary 10 bit length: 49
```

```
Total pixel length used(Msg bits+ reserved msg length bits): 401
```

```
Binary of Ascii msg bits:
```

```
0100011001000110010001100110011001100100011001000110010001100110011001000  
1100100011001000110011001100100100001001000010010000100100001001000011000100101010001  
01010001010100010101000101010001010100010101000101010001010100010101000110001  
0010101000101010001010100010101000101010001100010010010000100100001001000010010000100  
100001001000010010000100100001001000010010000110001
```

```
Binary identifier **0b**:
```

```
0b01000110010001100100011001100110010001100110011001000110010001100100011001100110010  
0011001000110010001100110011001001000010010000100100001001000010010000110001001010100  
0101010001010100010101000101010001010100010101000101010001010100010101000101010001100  
0100101010001010100010101000101010001010100011000100100100001001000010010000100100001  
00100001001000010010000100100001001000010010000110001
```

```
Decimal conversion of entire message bits:
```

```
1384478372948605811749267366425983223381460343060126203479925290319533306851558610173  
388771148685900172070667061306417  
390
```

```
***So the Hidden message in photo is*** : 2022JUIT
```

Binary 10 bit: 0b0000110001

Decimal value of binary 10 bit length: 49

Total pixel length used(Msg bits+ reserved msg length bits): 401

Binary of Ascii msg bits:

```
010001100100011001000110011001100100011001100110010001100100011001000110010001100110011001000
11001000110010001100110011001001000010010000010010000010010000010010000011000100101010001
010100010101000101010001010100010101000101010001010100010101000101010001010100010101000110001
0010101000101010001010100010101000101010001100010010010000100100000100100000100100000100
10000100100000100100000100100000100100000100100000110001
```

Binary identifier **0b**:

```
0b0100011001000110010001100110011001000110011001100100011001000110010001100100011001100110010
001100100011001000110011001100100100001001000001001000001001000001001000001100010010101000
010101000101010001010100010101000101010001010100010101000101010001010100010101000101010001100
0100101010001010100010101000101010001010100011000100100100001001000001001000001001000001
00100000100100000100100000100100000100100000100100000110001
```

Decimal conversion of entire message bits:

```
1384478372948605811749267366425983223381460343060126203479925290319533306851558610173
388771148685900172070667061306417
390
```

So the Hidden message in photo is : ###3#3###3###3\$\$\$\$1*****1*****1\$\$\$
\$\$\$\$\$\$1

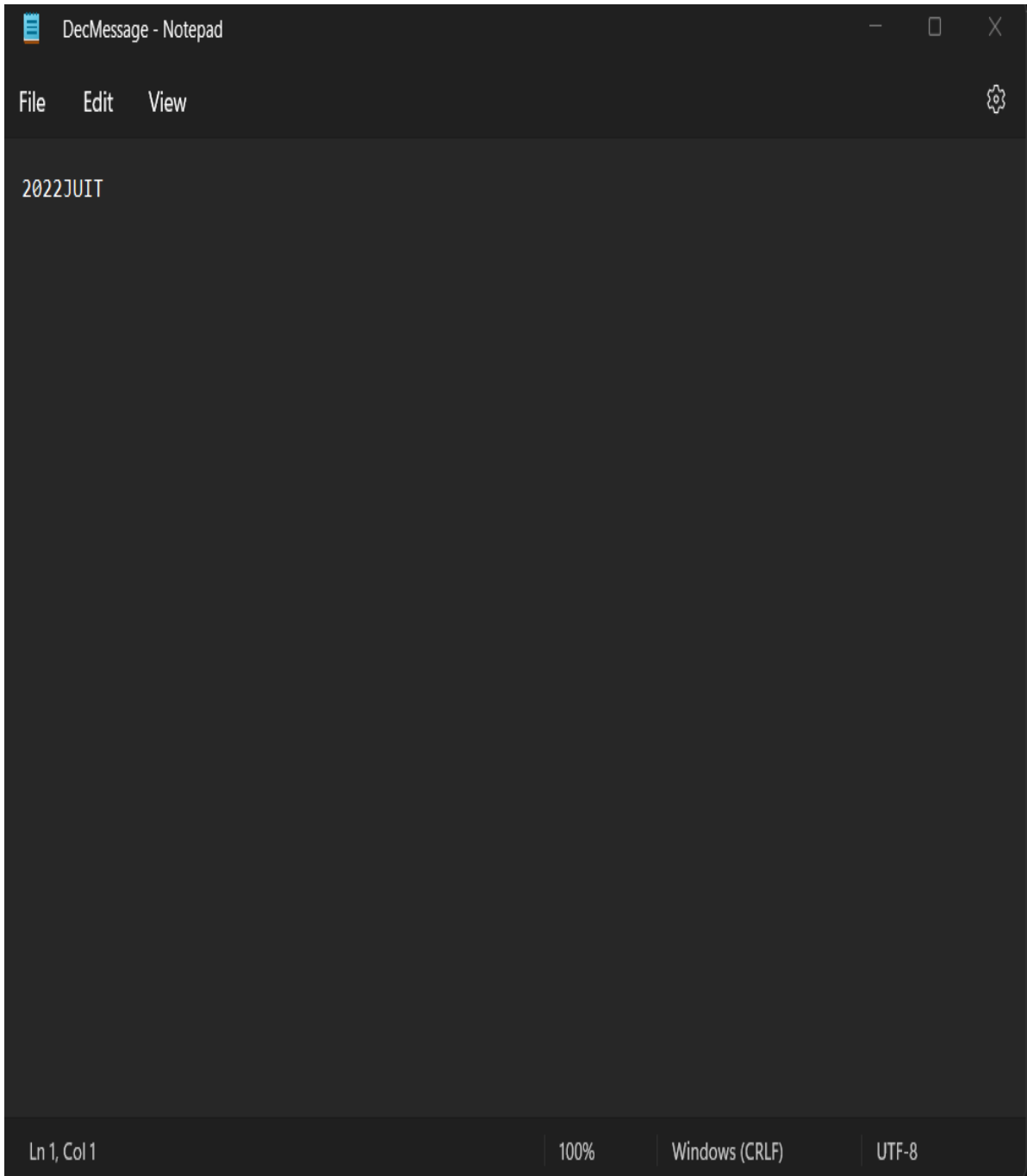


Fig4.11.2 Example 2 Decoding

Chapter 5

CONCLUSIONS

5.1 Conclusion

5.2 Applications of the Project

5.3 Future Scope of the Project

5.1 Conclusion

Image steganography is a perfect solution for the information leaks and data insecurity.

Image steganography allows secretly passing on the information between the users and the alterations and the other changes are not visible. It allows lossless compression of the cover image. The size, the quality of the original cover image and the secret text embedded image (stego image) remains unchanged. There are the alterations in the pixels but not visible to the human eye. That is the biggest reason that this technique is one of the best techniques used in the information security for the data hiding and secured communication.

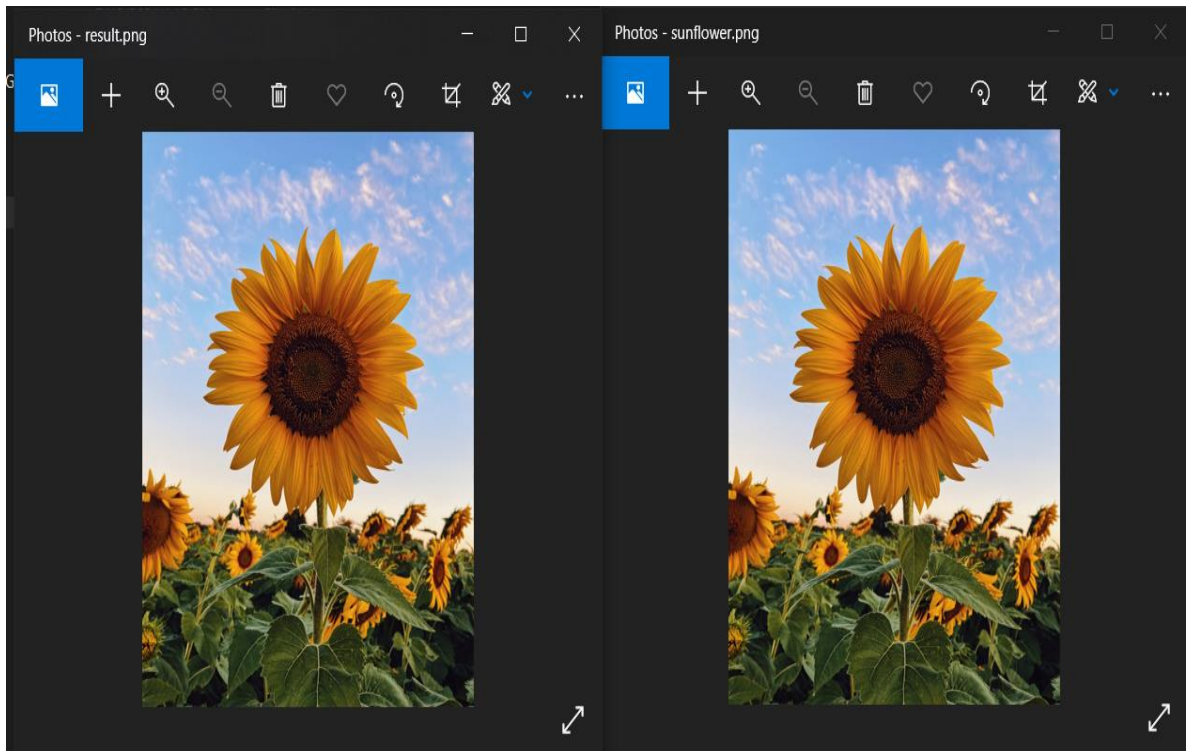


Fig 5.1 STEGO IMAGE vs ORIGINAL COVER

As you can see in the above comparison of the stego image and original cover, no change is there that can be seen through the eyes. Size and the no of pixels of the images are also the same.

Nowadays when cryptographic methods are used for communication, the developers and software engineers are looking up for the steganography techniques to make the communication highly confidential, secured and safe.

LSB steganography is in demand as it has been the most popular encoding algorithm due to its simplicity and high accuracy.

According to the various digital forensics organizations, many of the today's algorithms are unable to transfer the information in complete secured way but the Steganography combined with Cryptography has made the unauthorized parties almost impossible to track the secret codes hidden in the cover files. Therefore steganography in the coming years for sure will be a big thing for the Information Security purposes.

5.2 Applications of the Project

- 1. Securing data while Breaches:** Image steganography is an important technique used in digital forensics area during the occurrences of the data breaches or the data leak incidents. Basically whenever a data breach occurs, crucial data can be hidden using the steganography technique.
- 2. End-to-End Private communication:** If a client and a server want to share some peer to peer confidential information over a network, then they can use the data hiding and steganography technique to avoid any outer unauthorized interference.
- 3. Protection of Intellectual property Rights and Ownership:** Data hiding through image steganography can be used to hide the owner's designations or signatures secretly in any original cover image, audio, video or text so that the ownership rights remains with the owner and others cannot steal the owner's work.
- 4. Multimedia Fingerprinting:** This is similar to the ownership rights. The owner of the media can embed the secret copyrights on the covers so that the originality of the owner's work remains unaltered.
- 5. Smart Identity Cards:** Various Intelligence organizations make use of the image steganography to hide the details of their employees in their ID card images.

5.2 Future Scope of the Project

Though our project is able to reach highly accepted results and is able to hide data very efficiently, it can definitely be improved in some areas.

Our project works on hiding the textual data behind the cover images but in future, algorithms can be implemented to hide other format of data in it as well like audios, videos etc.

To make the communication more secured and impossible to hack, various existing ciphers can be implemented with the steganography algorithms. Our algorithms works on the lossless format of images or converts the lossy formats to lossless, but various techniques could be used in future for decreasing the amount of compression and the data loss in lossy formats, so that Image steganography by LSB can be implemented for the lossless form of cover data.

References

- 1) Mohammed A. Saleh, “Image Steganography Techniques”. International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE) (2018)
- 2) Arun Kumar Singh, Juhi Singh, Dr. Harsh Vikram Singh, “Steganography in Images Using LSB Technique (2015)”. International Journal of Latest Trends in Engineering and Technology (IJLTET)(2015)
- 3) Johnson, Neil F/ Doric, Zoran / Jajodia, Information Hiding: Steganography and Watermarking Attacks and Countermeasures (Advances in Information Security, Volume 1)
- 4) Tayana Morkel, “IMAGE STEGANOGRAPHY APPLICATIONS FOR SECURE COMMUNICATION”. University of Pretoria, Pretoria May 2012.
- 5) V. Lokeswara Reddy, Dr. A. Subramanyam, Dr. P. Chenna Reddy, “Implementation of LSB Steganography and its Evaluation for Various File Formats”. Int. J. Advanced Networking and Applications Volume: 02, Issue: 05, Pages: 868-872 (2011)
- 6) Arvind Kumar, Km. Pooja, “Steganography- A Data Hiding Technique”. International Journal of Computer Applications (0975 – 8887) Volume 9– No.7, November 2010

7) Kamran Ahsan., and Deepa Kundur., “Practical Internet Steganography: Data Hiding in IP”

https://www.comm.utoronto.ca/~dkundur/pub_pdfs/KunAhsTXSecWrkshp03.pdf

8) Eiji Kawaguchi, et al: A Model of Anonymous Covert Mailing System Using Steganographic Scheme, in INFORMATION MODELLING AND KNOWLEDGE BASES XIV, H. Yaakkola et al (Eds), IOS Press, pp.81-85, 2003. “Applications of Steganography”

<http://datahide.org/BPCSe/applications-e.html>.