

# Image Steganography and Trade-off between Secrecy of Image and their PSNR values

**Analysis By:**  
**Abhinav Dholi**

## **Table of Content**

### **1. Abstract**

### **2. Introduction**

#### **2.1 Background**

#### **2.2 Objective**

#### **2.3 Motivation**

### **3. Project Resource Requirements**

#### **3.1 Software Requirements**

#### **3.2 Hardware Requirements**

### **4. Literature Survey**

#### **4.1 Background**

#### **4.2 Literature Review**

#### **4.3 Summary**

## 5.Implementation Details and User Manuals

### 5.1 Implementation Details

### 5.2 User Manuals

## 6.Experimental Results and Analysis

### 6.1 Results

### 6.2 Tables

## 7.Conclusion and Future Work

## 8.References

## **1.Abstract**

Simply having the capacity to transfer a file from one location to another isn't enough. Businesses today face multiple security threats and a highly competitive environment. They need a secure file transfer system to protect and reliably transfer their sensitive, business-critical data.

It's basic that business need a protected document move framework that can move important information safely and effectively, regardless of file size, file transfer volume, or complexity.

While it's true that external file transfers create a security challenge, the movement of sensitive data to external end users is also a core operational process for just about every business. From a security perspective, data in transit is always data at risk since data in transmission presents an opportunity for interception. Unauthorized access can also occur when data is stored at rest for download on a file transfer server. And there's always the chance of files being delivered to an unintended recipient or mishandled by end users that receive the files. External file transfers of sensitive data thus require considerable attention— to protect digital assets.

## **2.Introduction**

## 2.1 Background

### What is steganography?

Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video.

### What is Image Steganography?

Image Steganography refers to the process of hiding data(here another image) within an image file. The image selected for this purpose is called the **cover-image** and the image obtained after steganography is called the **stego-image**.

### How is it different from cryptography?

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

The advantage of steganography over cryptography alone is that the intended secret message does not attract attention to itself as an object of scrutiny. Plainly visible encrypted messages, no matter how unbreakable they are, arouse interest and may in themselves be incriminating in countries in which encryption is illegal.

Cryptography is often used to supplement the security offered by steganography. Cryptography algorithms are used to encrypt secret data before embedding it into cover files.

### What is a digital image?

As we are performing image steganography we should know about digital images. It is described as a finite set of digital values, called **pixels**.

Pixels are the smallest individual element of an image, holding values that represent the brightness of a given colour at any specific point. So we can think of an image as a **matrix** (or a two-dimensional array) of pixels which contains a fixed number of rows and columns.

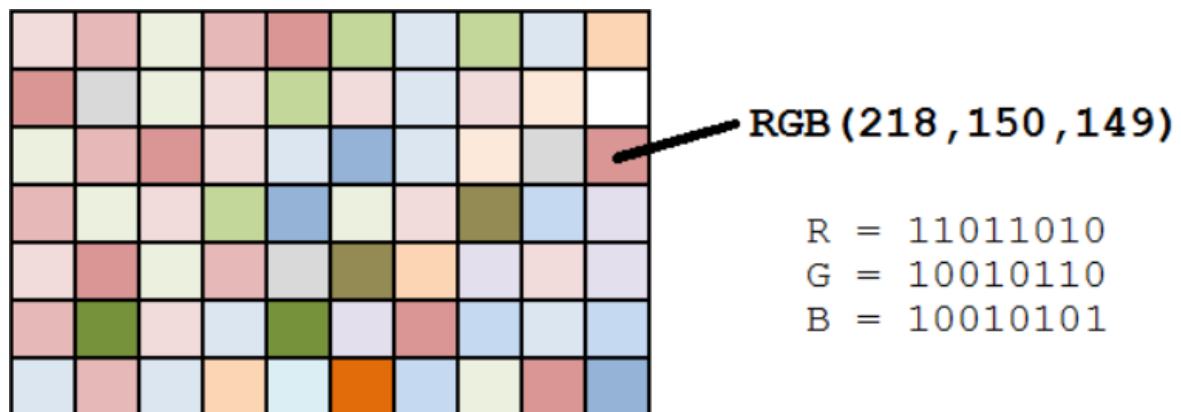
### Pixel concept and colour models

Each pixel is a sample of an original image. It means, more samples provide more accurate representations of the original. The intensity of each pixel is variable. In colour imaging systems, a colour is typically represented by three (red, green, and blue) or four component intensities (cyan, magenta, yellow, and black).

Here we have used RGB model.

The **RGB colour model** is an additive colour model in which red, green and blue light are added together in various ways to reproduce a broad array of colours. The name of the model comes from the initials of the three-additive primary colours, red, green, and blue. The main purpose of the RGB colour model is for the sensing, representation and display of images in electronic systems, such as televisions and computers.

Each pixel from the image is composed of 3 values (red, green, blue) which are 8-bit values (the range is 0–255 in integer format).



**Fig 2.1: representation of pixels of an image as 2-dimensional array**

In an 8-bit binary digit the **leftmost** bit is the **most significant bit**. If we change the leftmost bit it will have a large impact on the final value.

This impact factor reduces by the power of 2 while moving rightwards. The rightmost bits are less significant.

For eg:

if we change the leftmost bit from **1** to **0** (**11111111** to **01111111**) it represents **50.19%** change but when we change the leftmost bit from **1** to **0** (**11111111** to **11111110**) it will represent change less than **1%**.

So, if we change the rightmost bits it will have a small visual impact on the final image. This is the steganography key to hide an image inside another.

Hence, the idea is to change the less significant bits from an image and include the most significant bits from the other image.

PIXEL FROM IMAGE 1	PIXEL FROM IMAGE 2
R(11001010)	R(00001010)
G(00100110)	G(11000001)
B(11101110)	B(11111110)

NEW PIXEL FROM THE NEW IMAGE
R(11000000)
G(00101100)
B(11101111)

REG NO: 19BCE2423

19BCE2362

19BCE2396

19BCE2543

**Fig 2.2 : merging pixels of both images in ratio 4:4**

## 2.2 Objective

This aim of this project is to perform image steganography and calculate psnr values to analyse the perfect combination of bits of hidden and cover image by analysing trade-off between merged image quality and authenticity of retrieved hidden image at the other end.

## 2.3 Motivation

Due to the increasing number of threats to the existing data transfer protocols, we require some modern security techniques like steganography to protect important digital assets. It helps in providing privacy of information transmitted across the internet cloud. As cryptography alone is not able to enhance the robustness of information, we require techniques like steganography to prevent attacks related to image processing method.

## 3 Project Resource Requirements

### 3.1 Software Requirements

1. Python 3

2. Ubuntu OS 20.03 or any other linux based OS
3. VM Virtual Machine
4. Venv python package
5. Also other python packages used for image processing and running linux commands

### 3.2 Hardware Requirements

Device name	LAPTOP-5MEFB0GB
Processor	Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz
Installed RAM	8.00 GB (7.82 GB usable)

## 4 Literature Survey

### 4.1 Background

#### Steps used in LSB steganography:

##### a. Steps for hiding message image:

1. Read the image to be used as cover image. Noise is added to make it easier to disguise changes due to embedding the message image.
2. Read the image to be used as message image.
3. Separate the bit planes of each image. As it is known that the LSB (least significant bit) plane contains the least information associated with any image, and the MSB (most significant bit) plane contains most of the shape, colour information of an image. It is generally ideal to replace up to 4 least bitplanes of the cover image, with the upper 4 bitplanes without revealing changes in the resultant image. Lesser number of bitplanes from the message image could be used, but the retrieved image would become distorted and loses information.
4. Replace the least 4 bitplanes of cover image with the 4 most significant bitplanes from message image.
5. Get the resultant Steganographic image by recombining these bitplanes.

## b. Retrieving message image:

1. Read the Steganographic image.
2. Extract the required number of bitplanes of the image.
3. Recombining the lower four bitplanes would give the retrieved message image.

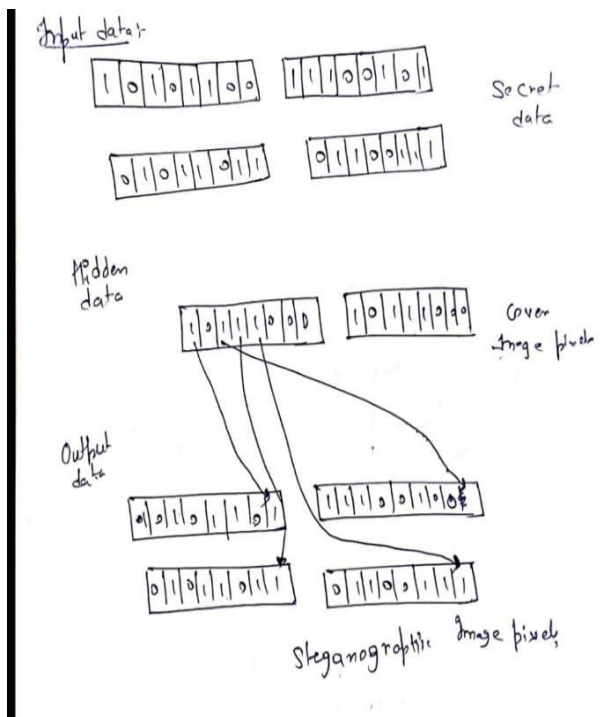


Fig. 2.3 This image portrays the traditional method of steganography

## 4.2 Literature Review

In this project, using steganography we have tried to conceal an image in another by experimenting with taking the different number of the most significant bits from the 8-bit pixels of the two images and forming a new encrypted image then comparing, which combination produces the most inconceivable image without completely ruining the value factor of the hidden image after its decryption using PSNR.

### Algorithm we are using to perform image steganography Fig. [2.1], [2.2]

1. Load the pixel maps of both the images using `image.load()` method after checking whether image to be hidden is smaller than or equal to the size of cover image.
2. Traverse through all the pixels (2D array - rows and columns) of both the images to store in respective integer tuples.



3. Convert the integer tuple to binary tuple using `__bin_to_int` method.
4. Using `_merge_rgb` method combine '4' bits (of choice) from both the tuples to be merged as single tuple containing concatenated 8-bit binary RGB valued pixels.
5. After checking pixel map position for the second image convert binary tuple back to an integer tuple using `bin_to_int` method.
6. Return stego image.
7. For retrieving the hidden image use `unmerge` method.
8. Create a new image that will be outputted using `image.new()` method according to cover image size.
9. Fill the new image with black pixels by initiating all RGB values to (0,0,0).
10. Get the RGB (as a string tuple) from the current pixel
11. Extract the last 4 bits (corresponding to the hidden image)
12. Concatenate 4 zero bits because total is 8 bits and store it in a binary tuple
13. Check valid position and crop the image based on valid pixels.
14. Return the hidden image

### **Steps to take out the PSNR values of different kinds of image steganographic techniques (Algorithm)**

Peak signal-to-noise ratio (PSNR) is the ratio between the maximum possible power of an image and the power of corrupting noise that affects the quality of its representation.

$$PSNR = 10 \log_{10} \left( \frac{(L-1)^2}{MSE} \right) = 20 \log_{10} \left( \frac{L-1}{RMSE} \right)$$

**Eq [1.1]**

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (O(i, j) - D(i, j))^2$$

**Eq [1.2]**

Where, **O** represents the matrix data of original image. **D** represents the matrix data of degraded image. **m** represents the numbers of rows of pixels and **i** represents the index of that row of the image. **n** represents the number of columns of pixels and **j** represents the index of that column of the image.

1. Read the new and the original image from its location using `cv2.imread()` function
2. Extract the values of the r, g, b from the pixels of both the image
3. Calculate the MSE for r, g, b separately (formula given above)
4. Get the final MSE by dividing all the MSE(r), MSE(g), MSE(b) by the 3\*size(height \* width) of the image and adding all of them up. **Eq [1.2]**

5. Calculate the final PSNR value by applying the formula given above and print it. Also rounding it up to 2 decimal places. **Eq [1.1]**

Use the above steps to calculate the PSNR which denotes the efficiency of every kind of steganographic technique given as follows

### 4.3 Summary

Steganography is one of the most consideration methods in the world of where secret information transmission, especially in defense field, is at stake. From encoding the hidden message (in form of text/image) to extract that secret message several steps are involved. In this paper we have discussed several techniques available for each step involved in steganography. The major loophole exists in selecting the correct method at correct time for which we have shown detailed comparative analysis between each technique. The next step involves calculating and comparing PSNR values of different encoded image samples (with difference in number of LSB bits changed). Higher the PSNR value better is the image resolution of the encoded sample or minimal exposure of secret message. All these steps are executed by various tools and steganographic algorithms. Steganography helps in building the security and lead to a secure communication between two clients. Any image can be processed into other image using steganography and to get a better security we learnt various methodologies using the research papers.

## 5. Implementation Details and User Manuals

### 5.1 Implementation Details

#### CODE for image steganography:

```
# -*- coding: utf-8 -*-
```

```
''''
```

```
Created on Sat Oct 10 17:49:37 2020
```

```
@author: user
```

```
''''
```

```
import click
```

```
from PIL import Image
```

```
class Steganography(object):
```

```
    @staticmethod
```

```
    # __int_to_bin method to convert a decimal value to a binary value
```

```
    #parameter rgb takes integer tuple as input eg:(220, 110, 96) of form (R, G, B)
```

```
    #returns a string tuple eg:("00101010", "11101011", "00010110")
```

```
    def __int_to_bin(rgb):
```

```
        r, g, b = rgb
```

```
        return ('{0:08b}'.format(r),
```

```
                '{0:08b}'.format(g),
```

```
                '{0:08b}'.format(b))
```

```
    #the same above return value when used as input
```

```
    #__bin_to_int method to convert a binary value to a decimal value
```

```
    #parameter rgb takes string tuple as input eg:("00101010", "11101011",  
    "00010110")
```

```
    #returns an integer tuple eg:(220, 110, 96)
```

```
    @staticmethod
```

```
    def __bin_to_int(rgb):
```

```
        r, g, b = rgb
```

```
        return (int(r, 2),
```

```
                int(g, 2),
```

```
                int(b, 2))
```

```
    @staticmethod
```

```
    #bits exchanged(1+7)
```

```
    #takes two rgb values and returns integer tuple with the two RGB values merged.
```

```
#eg: ("00101010", "11101011", "00010110") + ("00101010", "11101011",  
"00010110")
```

```
# in ratio 1 : 7
```

```
#merge result : ("[0]0101010", "[1]1101011", "[0]0010110")
```

```
def __merge_rgb(rgb1, rgb2):
```

```
    r1, g1, b1 = rgb1
```

```
    r2, g2, b2 = rgb2
```

```
    rgb = (r1[:1] + r2[:7],
```

```
           g1[:1] + g2[:7],
```

```
           b1[:1] + b2[:7])
```

```
    return rgb
```

```
@staticmethod
```

```
def merge(img1, img2):
```

```
#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.
```

```
    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!')
```

```
# Image.load() Load the pixel map of both images
```

```
    pixel_map1 = img1.load()
```

```
    pixel_map2 = img2.load()
```

```
#Image.new() creates a black image and takes input as mode(RGB/RGBA) and size  
of img1
```

```
    new_image = Image.new(img1.mode, img1.size)
```

```
    pixels_new = new_image.load()
```

```
#two loops to go through all rows and columns (each pixel/2-D array) from the  
images.
```

```
# size[0][1] -> width/row and height/column
```

```
    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)
#convert the new binary value to a decimal value

            pixels_new[i, j] = Steganography.__bin_to_int(rgb)

    return new_image

# to extract the hidden image from merged image
    @staticmethod
    def unmerge(img):
# Load the pixel map of merged image
        pixel_map = img.load()
# Create the new image and load the pixel map according to merged image...this
time
        new_image = Image.new(img.mode, img.size)
        pixels_new = new_image.load()
# Tuple (width,height) used to store the image original size
#unnecessary just to initialize
        original_size = img.size

        for i in range(img.size[0]):

```

```

        for j in range(img.size[1]):

            r, g, b = Steganography.__int_to_bin(pixel_map[i, j])

            # Extract the last 7 bits (as we used 7 MSB of the hidden image and concatenated
            # after 1 bit of cover image)

            # Concatenate 1 zero bit because of total 8 bit values

            rgb = (r[1:] + '0',
                  g[1:] + '0',
                  b[1:] + '0')

            #convert to int tuple again

            pixels_new[i, j] = Steganography.__bin_to_int(rgb)

            # If this is a 'valid' position, store it

            # as the last valid position(or update size tuple)


            #final size should be -> size of image 1 - extra black border

            #(the hidden image was smaller than the image which is hiding it)

            if pixels_new[i, j] != (0, 0, 0):

                original_size = (i + 1, j + 1)


            #image.crop takes 4-tuple as parameter (left, upper, right-> width, and lower->
            #height pixel coordinate)

            # and returns image output

            # to remove the black borders

            # 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

@click.command()
#merge --img1=/home/debalay/Desktop/osproject/img1.png
#--img2=/home/debalay/Desktop/osproject/img2.png
#--output=/home/debalay/Desktop/osproject/merge/outputstthreebit.png
#address is string type
@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)

@click.command()
#unmerge --img=/home/debalay/Desktop/osproject/merge/outputstthreebit.png
#--output=/home/debalay/Desktop/osproject/unmerge/unmergethreebit.png

@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()

```

### **CODE for calculating PSNR:**

```
import math

import cv2

import csv


def read_png(image_name):

    return cv2.imread(image_name+'.png')


def calculate_psnr(n):

    #To read merged image

    s = read_png('/home/debalay/Desktop/osproject/source/s'+n)

    #To read cover image

    r = read_png('/home/debalay/Desktop/osproject/recovery/rs'+n)


    height, width, channel = s.shape

    size = height*width


    sb,sg,sr = cv2.split(s) #for merged image

    rb,rg,rr = cv2.split(r) #for cover image


    mseb = ((sb-rb)**2).sum()

    mseg = ((sg-rg)**2).sum()

    mser = ((sr-rr)**2).sum()


    MSE = (mseb+mseg+mser)/(3*size)

    psnr = 10*math.log10(255**2/MSE)

    return round(psnr,2)
```



```
def write_csv(n,data):

    with open('/home/debalay/Desktop/osproject/target/psnrvalues'+n+'.csv', 'w',
newline='') as myfile:

        wr = csv.writer(myfile, quoting=csv.QUOTE_ALL)

        wr.writerow(data)

for i in range(4):

    print("Creating CSV of PSNR-result",i+1,"...",sep="")

    write_csv(str(i+1),[calculate_psnr(str(i+1))])
```

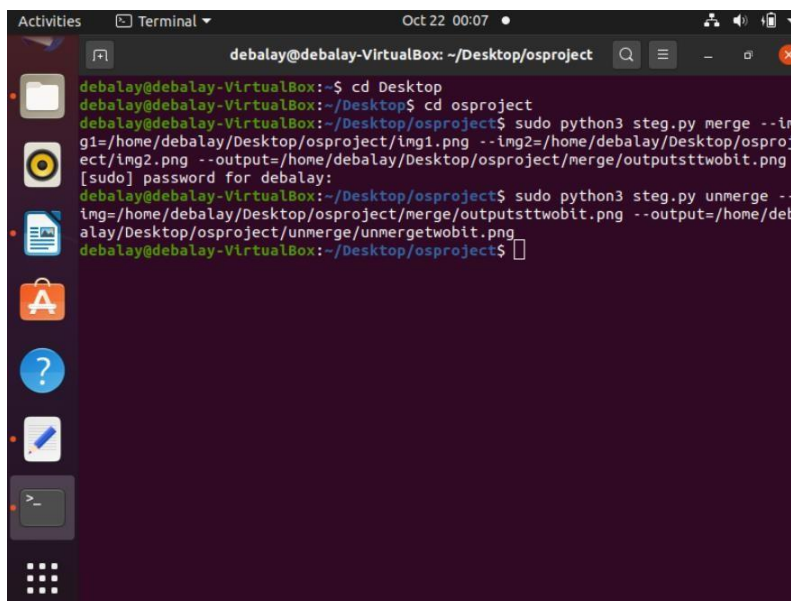
## Linux Commands used to run the code through terminal

### For merging the images

```
sudo python3 steg.py merge --img1=/home/debalay/Desktop/osproject/img1.png
--img2=/home/debalay/Desktop/osproject/img2.png --
output=/home/debalay/Desktop/osproject/merge/outputstthreebit.png
```

### For unmerging the images

```
sudo python3 steg.py unmerge --
img=/home/debalay/Desktop/osproject/merge/outputstthreebit.png --
output=/home/debalay/Desktop/osproject/unmerge/unmergethreebit.png
```



The screenshot shows a terminal window titled "Terminal" with the date "Oct 22 00:07". The user is logged in as "debalay" on a "debalay-VirtualBox" machine. The terminal shows the following commands and output:

```
debalay@debalay-VirtualBox: ~/Desktop/osproject
debalay@debalay-VirtualBox:~$ cd Desktop
debalay@debalay-VirtualBox:~/Desktop$ cd osproject
debalay@debalay-VirtualBox:~/Desktop/osproject$ sudo python3 steg.py merge --im
g1=/home/debalay/Desktop/osproject/img1.png --img2=/home/debalay/Desktop/osproj
ect/img2.png --output=/home/debalay/Desktop/osproject/merge/outputsttwobit.png
[sudo] password for debalay:
debalay@debalay-VirtualBox:~/Desktop/osproject$ sudo python3 steg.py unmerge --
img=/home/debalay/Desktop/osproject/merge/outputsttwobit.png --output=/home/deb
alay/Desktop/osproject/unmerge/unmergetwobit.png
debalay@debalay-VirtualBox:~/Desktop/osproject$
```

## 5.2 User Manuals

### Other Functions:

#### **@staticmethod:**

It is a function which returns a static method (member of an object but can be directly called by the constructor)

### Linux Commands/Methods:

#### **@cli.command()**

This command is basically invoking the null operation in the code and running it on the terminal

#### **@click.group():**

This command defines the function as the “**def cli()**; in our case” main super command in the code written and the rest as sub-commands

#### **@click.option():**

option() decorator is used to add options to the commands defined under “[@cli.command\(\)](#)”

### Libraries used:

#### **Pillow**

Commonly called as pillow or pil library adds image processing capabilities to the python interpreter.

#### **Click**

Click is a Python package for creating beautiful command line interfaces in a composable way with as little code as necessary. It’s the “Command Line Interface Creation Kit”. It’s highly configurable but comes with sensible defaults out of the box.

#### **OpenCV**

It is a python library used to perform machine learning operations to the python interpreter.

#### **math**

This module provides access to the mathematical functions defined by C standard.

## **CSV**

It implements classes to read and write modules in csv format.

## **6.Experimental Results and Analysis**

### **6.1 Results**

In 4 Bit steganographic technique, it was obtained experimentally that the value of PSNR is the highest which clearly means that it is the most efficient way of hiding and retrieving the secret image from the cover image.

Below are the few inferences we noticed in change of the size of the secret image after its extraction from the hidden one

## 6.2 Tables

### STEGANOGRAPHY SIZE CHART

**Image 1 -**



Original Size: 297.4 KB

**Image 2 -**

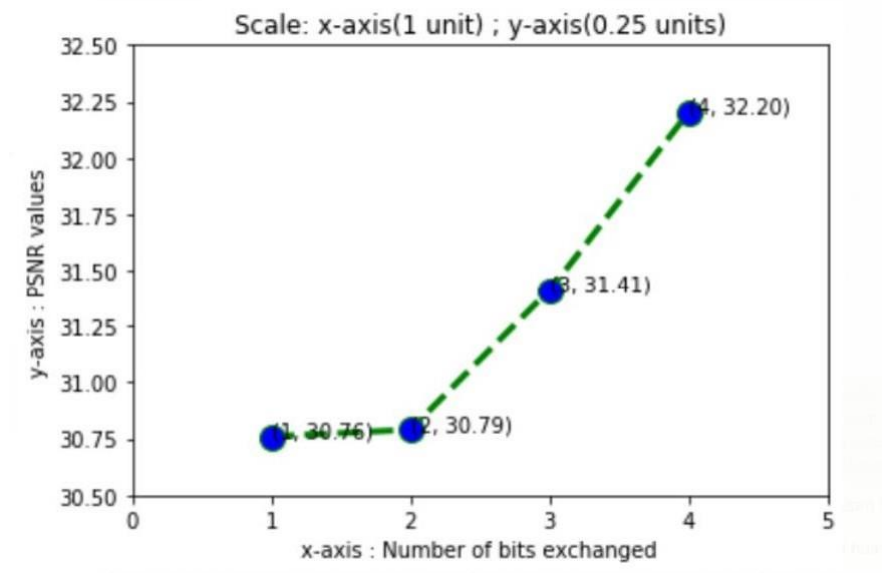


Original Size: 287.8 KB

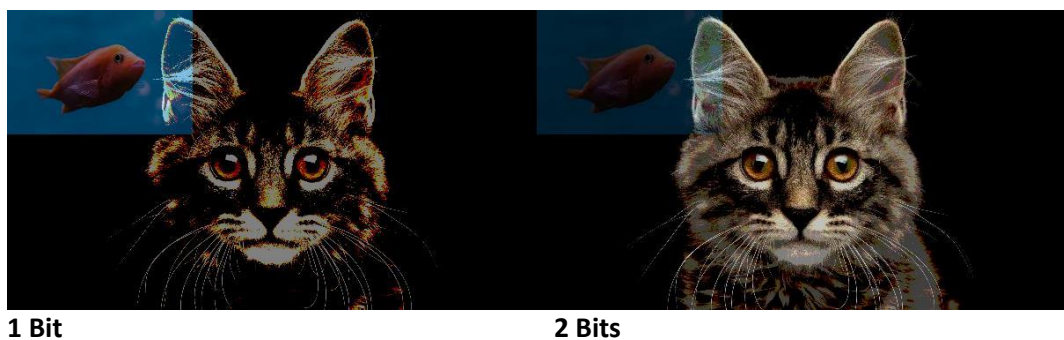
BITS EXCHANGED	MERGE (in KB)	UNMERGE size of Secret Image (in KB)
1 BIT	637.7	400.6
2 BITS	734.8	287.8
3 BITS	837.2	184.9
4 BITS	400.6	124.2

### PSNR Values obtained

Bits Exchanged	1 Bit	2 Bit	3 Bit	4 Bit
PSNR	30.76	30.79	31.41	32.2

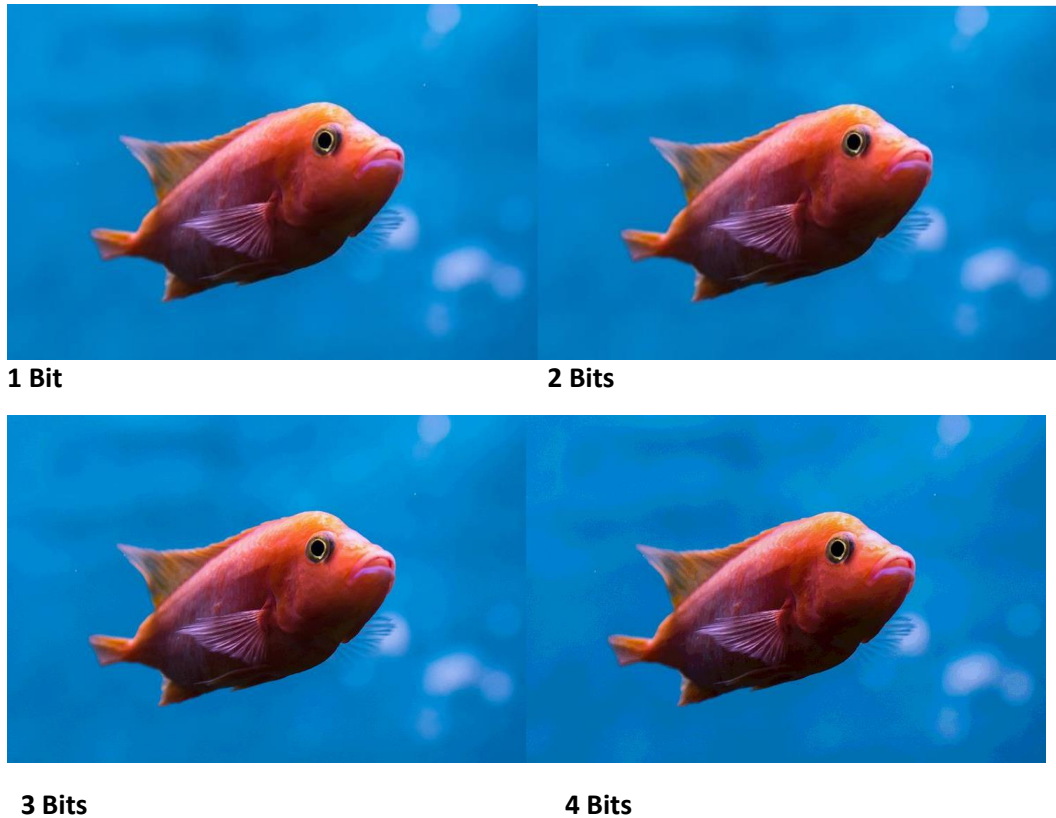


### Merged Images





### Unmerged Secret Image



## 7.Conclusion and Future Work

The project draws conclusion that combining the four MSB bits of the hidden image and cover image respectively gives the best quality of merged/stego image and on retrieving the hidden message(image) it is of acceptable quality.



Individually cryptography and steganography provide confidentiality to the data but they have some vulnerability. So, we are aiming for a combination of cryptography and steganography. Another challenge remains of performing image steganography where cover image has less size than hidden image (I.e., by lossless compression). As we are dealing with image-based steganography so reduction of image quality still remains a threat to data transfer security.

## 8. References

1. Nada Abdul Aziz Mustafa, "Text hiding in text using invisible character", College of Languages, University of Baghdad, Iraq .
2. Ashwak ALabaichi<sup>1</sup> , Maisa'a Abid Ali K. Al-Dabbas<sup>2</sup> , Adnan Salih<sup>3</sup>, "Image steganography using least significant bit and secret map techniques", Department of Biomedical Engineering, Engineering College, University of Kerbala, Iraq <sup>2</sup>Department of Computer Science, University of Technology, Iraq <sup>3</sup>Department of Computer Science, Science College, University of Kirkuk, Iraq
3. Zena Ahmed Alwan<sup>1</sup> , Hamid Mohammed Farhan<sup>2</sup> , Siraj Qays Mahdi<sup>3</sup>, "Color image steganography in YCbCr space", Department of Medical Instrumentation Techniques Engineering, Electrical Engineering Technical College, Middle Technical University, Iraq <sup>2,3</sup>Department of Computer Engineering Techniques, Electrical Engineering Technical College, Middle Technical University, Iraq
4. Youssef Taouil and El Bachir Ameer, "Steganographic Scheme Based on Message-Cover matching", Department of Computer Sciences, Faculty of Sciences, University Ibn Tofail, Morocco
5. Yusuf Perwej, Firoj Parwej, Asif Perwej, " [An Adaptive Watermarking Technique for the copyright of digital images and Digital Image Protection](#)", The International Journal of Multimedia & Its Applications (IJMA) Vol.4, No.2, April 2012
6. Pinki, Rajesh Mehra "Estimation of the Image Quality under Different Distortions" International Journal Of Engineering And Computer Science ISSN: 2319-7242 Volume 5 Issues 7 July 2016, Page No. 17291-17296.
7. Nik Shahidah Afifi Bt Md Taujuddin, Rosziati Bt Ibrahim, Suhaila Bt Sari, "A Comparative Analysis of PSNR value for Images using Wavelet Based Thresholding Methods" published by Universiti Tun Hussein Onn Malaysia (UTHM), Research, Innovation, Commercialization and Consultancy Management (ORICC) office and Malaysian Ministry of Education.
8. Yusra A. Y. Al-Najjar, Dr. Der Chen Soong "Comparison of Image Quality Assessment: PSNR, HVS, SSIM, UIQI" International Journal of Scientific & Engineering Research, Volume 3, Issue 8, August-2012 ISSN 2229-5518
9. Raushan Kumar, Gunjan Sharma, Varun Sanduja "A Real Time Approach to Compare PSNR and MSE Value of Different Original Images and Noise (Salt and Pepper, Speckle, Gaussian) Added Images" International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS) Volume VII, Issue I, January 2018 | ISSN 2278-2540
10. Parminder Kaur, Jagroop Singh "A Study on the Effect of Gaussian Noise on PSNR Value for Digital Images" International Journal of Computer and Electrical Engineering, Vol. 3, No. 2, April, 2011 1793-8163
11. Image Steganography Faculty: Dr. Arnab Shaw, Savitha Bhallamudi Wright State University
12. Image Steganography Techniques Mohammed A. Saleh College of Sciences and Arts in Ar Rass, Qassim University, Kingdom of Saudi Arabia IJARCCCE
13. Image Steganography Using Mid Position Value Technique, Author: Srilekha Mukherjeea, , Subhajit Roy, Goutam Sanyal Computer Science & Engineering Department National Institute of Technology Durgapur Durgapur, India PROCEDIA computer science
14. An Introduction to Image Steganography Techniques Alaa A. Jabbar Altaay, Shahrin bin Sahib Research Gate
15. International Journal of Advanced Research in Computer Science and Software Engineering Ashadeep Kaur\*, 2Rakesh Kumar, 3Kamaljeet Kainth ijarcsse
16. Amandeep Singh Sidhu et al, International Journal of Computer Science and Mobile Computing, Vol.3 Issue.12, December- 2014, Research Paper on Text Data Compression Algorithm using Hybrid Approach
17. International Journal of Scientific & Engineering Research Volume 3, Issue 3, March -2012 1 ISSN 2229-5518, An Improved Data Compression Method for General Data by Salauddin Mahmud
18. (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 3, No.8, 2012, A New Algorithm for Data Compression Optimization I Made Agus Dwi Suarjaya Information Technology Department Udayana University Bali, Indonesia
19. Engineering Science and Technology, an International Journal, A high capacity text steganography scheme based on LZW compression and color coding Aruna Malik, Geeta

Sikka, Harsh K. Verma Department of Computer Science & Engineering, National Institute of Technology, Jalandhar, India

20. IJIRST –International Journal for Innovative Research in Science & Technology| Volume 4 | Issue 1 | June 2017 ISSN (online): 2349-6010, A Research Paper on Lossless Data Compression Techniques by Prof. Dipti Mathpal, Prof. Mittal Darji, Prof. Sarangi Mehta