

**A Project Report on
CONCEALING SECRET MESSAGE IN AN IMAGE USING NUAIS AND
SECRET SHARING ALGORITHM**

**Project report submitted in partial fulfilment for the requirement of
award of B.Tech. Degree in Information Technology
by**

UDAY KUMAR REDDY, G	114015038
KIRAN, P	114015052

**Under the Guidance of
Prof. B.Srinivasan
Assistant Professor-II, IT**



**SCHOOL OF COMPUTING
SHANMUGHA ARTS, SCIENCE, TECHNOLOGY & RESEARCH ACADEMY
(SASTRA UNIVERSITY)**

(A University Established under Section 3 of the UGC Act, 1956)

**THIRUMALAISAMUDRAM, THANJAVUR-61340
April 2014**

**SHANMUGHA ARTS, SCIENCE, TECHNOLOGY & RESEARCH ACADEMY
(SASTRA UNIVERSITY)**

**(A University Established under Section 3 of the UGC Act, 1956)
THIRUMALAISAMUDRAM, THANJAVUR-613401**



**SCHOOL OF COMPUTING
BONAFIDE CERTIFICATE**

**This is to certify that the Project entitled
CONCEALING SECRET MESSAGE IN AN IMAGE USING NUAIS AND
SECRET SHARING ALGORITHM**

is a work done by

UDAY KUMAR REDDY, G	114015038
KIRAN, P	114015052

**BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY
OF SASTRA UNIVERSITY, Thanjavur during the year 2013-14**

Internal Guide

**Associate Dean
Department of Information Technology**

Submitted for the university examination held on:

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

First and Foremost, we take pride in thanking the Almighty who gave us strength for the successful completion of this project.

We would forever remain grateful to **Prof. R.SETHURAMAN, Vice chancellor** and **Dr. G.BHALACHANDRAN, Registrar**, SASTRA UNIVERSITY, for giving us an opportunity to do this project.

We are extremely thankful to my dignified **Dean, Dr. P.SWAMINATHAN, School of Computing and Associate Dean, Dr. N.SAIRAM** for the constant motivation and support offered by them in materializing this project.

We express our gratitude to my guide **Dr. Shankar Sriram .V.S, Associate Professor**, for the valuable suggestions and constant encouragement he offered leading to the successful completion of the project.

We are also very grateful to **Prof P. RAJENDRAN, Assistant Professor-III** for being very helpful with coordinating among the students and providing useful tips.

Last but not least, we also thank all the **Teaching staff, Non-Teaching staff**, and those who have directly or indirectly helped me by extending their moral support and encouragement for completion of this project.

We gratefully acknowledge all the contributions and encouragement from our family and friends resulting in the successful completion of this project.

TABLE OF CONTENTS

S. No.	Title	Page No
1	Synopsis	1
2	System Configuration	2
3	Project Description	3
4	Diagrams	13
5	Sample source code	18
6	Sample output (Snapshots)	59
7	Conclusion	62
8	Bibliography	63

1. SYNOPSIS

Data security is the most important source of research and study currently and computer analysts and scientists are working towards a fully secured system for data. There are various methods that are used to hide important data. Hiding the data onto any other form of multimedia like image, audio, video etc., broadly referred to as the Steganography, is one of the most commonly used method. But the way in which it is done depends on the need.

Unlike traditional approaches that typically use a direct embedding of the data onto the image, we provide an efficient mechanism that first encrypts the data to be sent and the cipher text is then embedded onto an image. Instead of embedding the cipher directly into the image, the image is initially divided into various non-uniform segments. Depending on the size of the data and amount of data to be hidden in each segment, the number of segments to hide cipher data is decided. Later, the same number from remaining segments is used to hide the key value of corresponding segments. Then all these segments are combined into the original image. Thus, the same image is then used to transfer the key. The Encryption and Decryption mechanism that is used is Rivest, Shamir and Adleman Algorithm(RSA). Once the images are ready, they are sent to the intended receiver who has the appropriate segmentation key to receive the full data. Thus we combine the concepts of security at the matrix level, cryptography and steganography to efficiently hide secret data.

Thus by following the above explained process we ought to get a significant improvement in the security. This can be particularly useful in case of top secret information transfer services.

2. SYSTEM CONFIGURATION

2.1 HARDWARE CONFIGURATION

- The code was developed in a system with processor Intel Core i5-2410M CPU @ 2.30GHz.
- The Installed Memory (RAM) is 4.00 GB. The graphics support of 1.0 GB NVIDIA GEFORCE is provided in addition to INTEL Graphics card that was already provided.
- With this configuration available, it will be easy to work with MATLAB and also jdk (JAVA) can also be installed.

2.2 SOFTWARE CONFIGURATION

- The operating system is Windows 7. Its 64-bit.
- Notepad which acts as an Editor to type my Java Code
- a Java Runtime Environment(JRE) to compile and execute the java code

3. PROJECT DESCRIPTION

Anything and everything in this world has associated information. This information or data is available in great lengths that it can be used for non-productive purposes too. Hence it is imperative for us to have a security mechanism that ensures that the data does not fall into the wrong hands. There have been many security measures that have been adopted from the yester years but none can claim to be perfect. This is the problem that we are going to address in this project.

Data can be made secure by doing an encryption. By encryption we mean that the text to be sent to a receiver is being converted to some other form using some cryptographic encryption algorithms so that any intruder will be unable to know the original text as it is being replaced by some other scrambled text which we call cipher text. This cipher text is sent to the other end where the reverse process of encryption called decryption is done. The process by which a text is converted to a cipher text involves the use of a special 'key' that specifies how the conversion takes place.

Cryptographic algorithms can be broadly classified into 2 categories of symmetric and asymmetric key algorithms. The symmetric key algorithms use the same key to encrypt and decrypt and thus requires the ability to have the same key at either end to retrieve the original text. Asymmetric algorithm also known as the Public key algorithm uses 2 keys: private and public keys. Asymmetric key algorithms require the presence of different key and private key generated for the user has to be kept secretly because once the key is known for an intruder the plain text can be easily retrieved from the cipher text. Many algorithms follow this standard and RSA is one of them.

Though there is a wide range of encryption and decryption algorithms, we have used RSA since it has been accepted by the National Security Agency of the US as one of the most secure systems existing at present and this encouraged us to look it as an option and we did use it to encrypt the plain text that we wanted to send to the receiver end.

This encryption system is of a higher standard and thus is being used by many countries for their security purposes. RSA algorithm is one of the most successful algorithms for encryption and decryption that it comes with a wide of specifications and depending on the need, the correct specification is used.

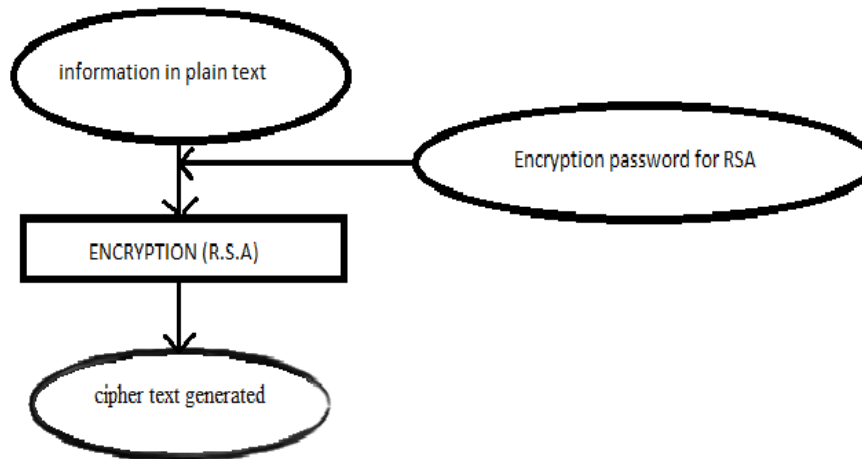
The entire project that is to be done can be carved up into separate units so that it will be easy to understand and also it will be easy to work with. First, let's have a brief discussion about the project objective, tasks involved in achieving it and then all these tasks will be explained in detail.

3.1 OBJECTIVE OF THE PROJECT

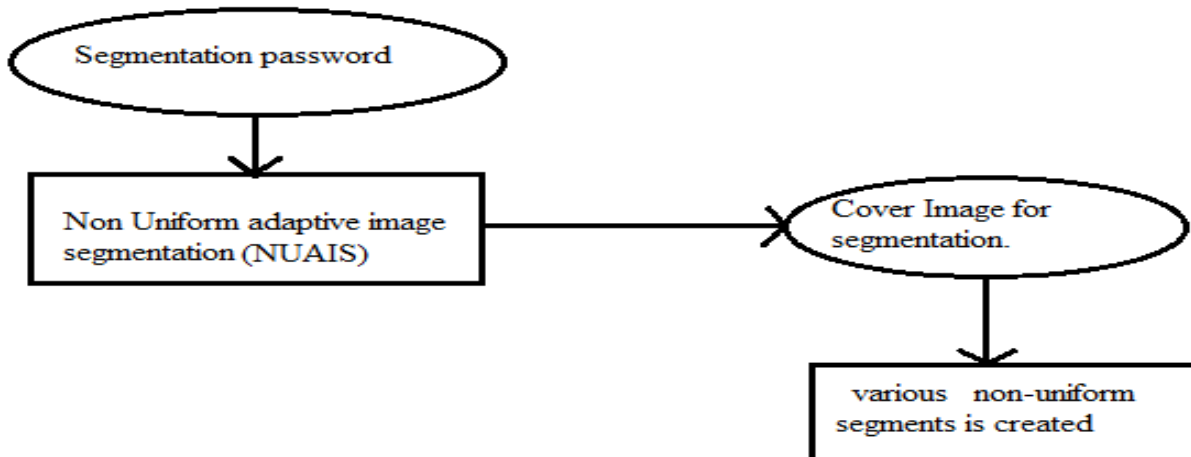
The overall objective of this project is to provide an efficient way of hiding the message. Then we start by encrypting the message to be sent using a very powerful algorithm called RSA which will produce a cipher text. This cipher text that is generated will be totally different from the plaintext provided as input, so it is impossible for anyone to break it unless he/she has the key. The public key is given and the private key is generated which is useful for the receiver to decrypt. So the Second step is that the cipher text that is generated is hidden into an image using LSB techniques so as to avoid image quality degradation. In the same way, the key generated after secret sharing algorithm is also hidden in an image. The private key generated after RSA encryption and the image, segmentation key is required to decrypt. All these are sent to the receiver where he looks into the specified image for data and key and then decodes the images. Then he decrypts using the RSA decryption algorithm to get the original message.

3.2 STEPS INVOLVED

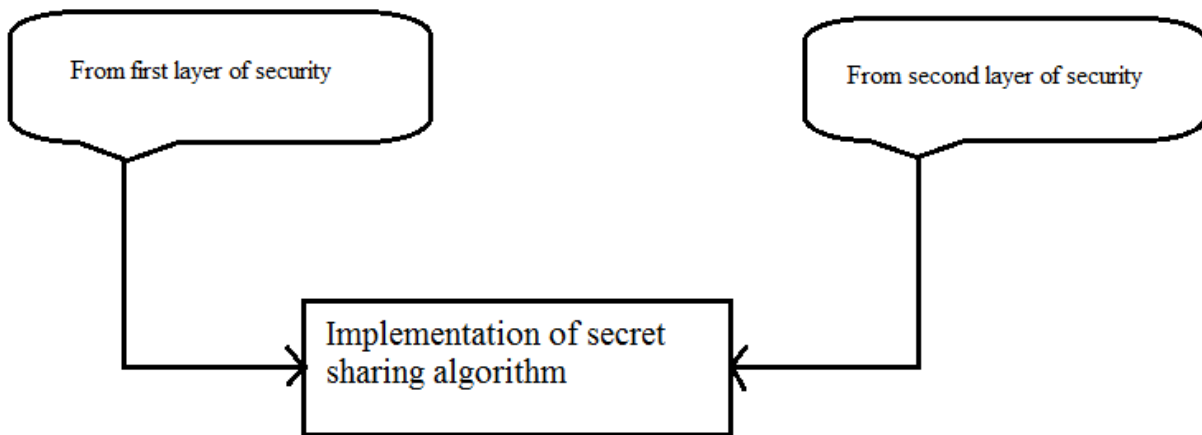
FIRST LAYER OF SECURITY



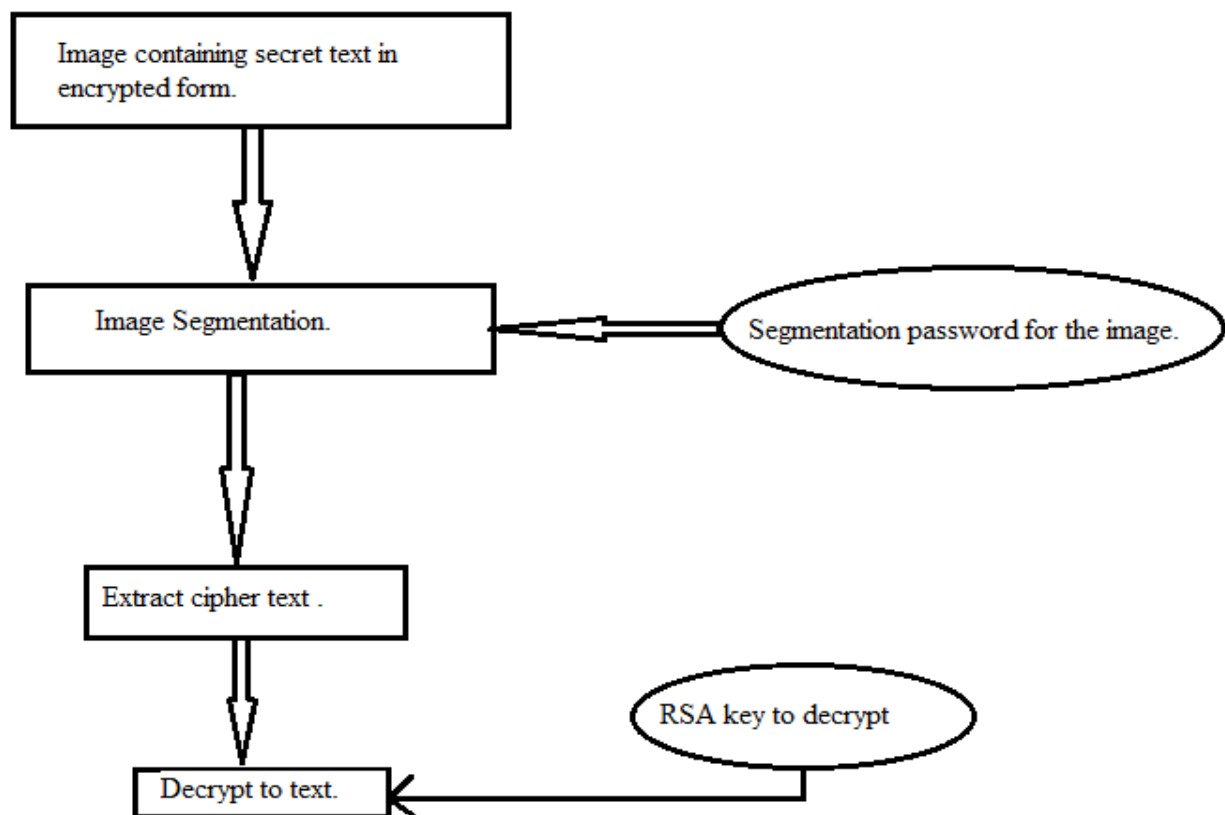
SECOND LAYER OF SECURITY



THIRD LAYER OF SECURITY



STEPS INVOLVED DURING EXTRACTION OF DATA



3.2.1 RSA ENCRYPTION AND DECRYPTION:

The RSA encryption starts by creating a text file named as “rsainput.txt”, which is given as input to code RSA written in JAVA programming language, through file reader object.

In the code the java.math.BigInteger package is imported.

Initially two random numbers are generated and they are sent as arguments to the ProbablePrime method of BigInteger class, which in turn returns a BigInteger object. In this way two BigInteger Objects are created. These two BigIntegers are multiplied and a new BigInteger object is created. Then the user is requested to give a public key which is mostly an integer.

For this public the GCD is calculated with the two created prime numbers of type BigInteger.

If the GCD is equal to ‘1’, then the numbers are said to be co-primes.

When the GCD is equal to one a public key will be created which is assigned to BigInteger object. From the generated public key private keys are created. The creation of private key is done by calling modInverse method of BigInteger class, through previously created public key BigInteger object. Thus the generated private key is stored in the file ‘keys.txt’.

Based on the key given and the key generated, by following the RSA algorithm the given text in the input file is encrypted. For each character in the text file an equivalent big integer is created and the created is stored in ‘rsaoutput.txt’ file. Thus the cipher text which is to be stored in the image is created and the encryption is done successfully, by compiling and executing the RSA.java file.

The private key generated in ‘keys.txt’ and the ‘rsaoutput.txt’ are given as FileInputStream to the decryption code ‘NRSA.java’. Based on the key and the file generated the reverse process of encryption is carried out and the original output is obtained in ‘rasoutput.txt’ which contains the same text as the input file given to RSA for encryption. Thus the RSA encryption and decryption is done successfully by writing the code in JAVA.

3.2.2 Non- Uniform Adaptive Image Segmentation (NUAIS):

To perform Non-Uniform Adaptive Image Segmentation, segmentation password and a cover image in '.bmp' format for which segmentation is to be performed is required. The segmentation of the image is carried out based on the ASCII values of the characters of the segmentation password.

Initially, the image is read using read method of ImageIO class, which returns BufferedImage object equivalent to the image read. The buffered image is initially segmented along the width in the order in which the segmentation password is given as input and the divisions are stored in the 'horizontal array', which contains the ASCII values of all the characters in the given password in the same order as it is given by user.

Later the image is segmented in the vertical direction in the reverse order of the given password and the ASCII values of the password in reverse order i.e. the segments length direction in vertical direction is stored in 'vertical array'.

For the first row the segmentation is carried out with the first character of the vertical array as the height of the all segments in that row. The width of the segments in the first row is carried accordingly to the values in the horizontal array.

For the second row the segmentation is carried out with the second character of the vertical array as the height of the all segments in that row. The width of the segments in the second row is carried accordingly to the values in the horizontal array, in reverse order.

The segmentation of the remaining part of the image is carried out based on the row value of the image. The row number gives the height of all segments in that row, the height will be equal to the value of the vertical array at that row number index. (Eg: If the value of the 10th element in vertical row is 'x', then the height of all elements in 10th row is equal to 'x').

For a particular row the length of each segment also depends on the row number of that segment. If the row number is odd(i.e. 1,3,5,7,9.....) then the width of each segment will be equal to the value of each element of horizontal array in the same order. If the row number is even(i.e. 2,4,6,8.....) then the width of each segment will be equal to the value of each element of the horizontal array taken in reverse order.

Thus the various segments of the image which are generated is taken in to a file in the same format of the input image. Thus the image segmentation is done successfully, which is shown below.

IMAGE SEGMENTATION

Let Image segmentation password be...
pqrstu

Let the ascii values of the characters be as below..

<u>CHAR</u>	<u>ASCII</u>
p	P
q	Q
r	R
s	S
t	T
u	U

The image is
segmentated as
shown...

	P	Q	R	S	T	U	P	Q	R	S	
U											Segmentation for odd numbered row
T	U	T	S	R	Q	U	T	S	R	Q	Segmentation for even numbered row
S											
R											
Q											
P											
U											
T											
S											
R											
Q											
P											

3.2.3 HIDING CIPHERTEXT WITHIN IMAGE USING SECRET SHARING ALGORITHM:

The cipher text that is been generated is converted in to binary and is stored in to a file. The above obtained binary file is later given as input to another code where the binary file containing the binary form of the cipher text is converted in to various lines of binary data and is stored in another text file. Each line of this text file represents the number of bits of the binary data that is to be hidden in each segment of the cover image that is obtained after the segmentation process.

The hiding of the binary data in to the image segments follows the secret sharing algorithm, where the data is hidden based on the value of main case and sub case. Initially, the segment is read using the read method of ImageIO class which returns a BufferedImage. By using this BufferedImage object each pixel is read from the image segment. The pixel contains three color components, Red, Green and Blue. Each color component has an intensity value ranging from 0 to 255. For each pixel three values are calculated called the main case of each color, which is calculated using the formula.

$$\text{Main Case (MC)} = (\text{color intensity})/16+1;$$

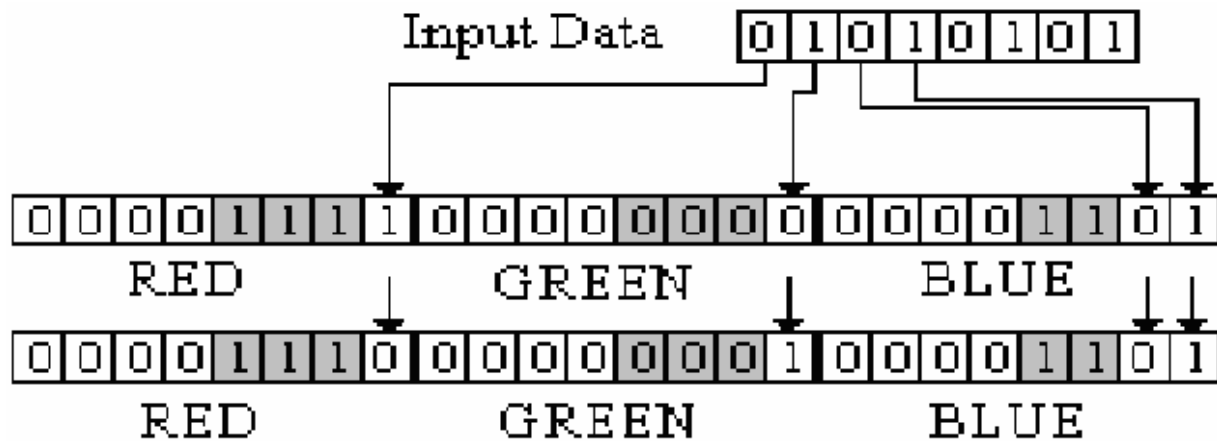
Thus for each color we get a MC value which is in the range $i < MC < 16$. Thus for a pixel we get three MC values. The least among those three values is the MC value of the entire pixel. Thus every pixel falls under a MC value from 0 to 16. The MC value determines the storage capacity of the pixel. If MC value is least the pixel can hide a large amount of data bits, without any large change in the color intensity of the pixel. The way in which the data bits should be hidden inside the pixel depends on the value of the Sub Case (SC). If MC is Main Case of a pixel and is indeed the main case value of a color in that pixel, let the main case value of the rest colors of that pixel be X and Y. Then the sub case for that pixel is determined by the following calculations. Six sub case values are determined they are obtained as follows. For each sub case the algorithm follows a specific way of hiding of the data bits into the color value of each pixel, which can be taken as a String of 24 bits, with 8 equal divisions showing the RGB bytes.

Based on the sub case, the number of data bits that should be hidden in each color component of the image is determined and the hiding of data bits is done as shown below.

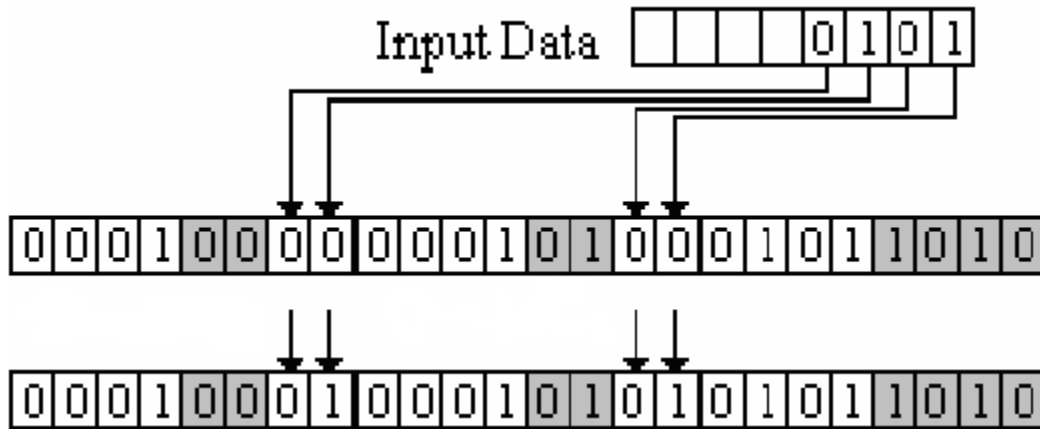
SC value	Conditions of MC, X, Y
1	$X, Y < MC$
2	$X = MC$ and $Y < MC$ or $Y = MC$ and $X < MC$
3	$X, Y = MC$
4	$X > MC$ and $Y < MC$ or $X < MC$ and $Y > MC$
5	$X > MC$ and $Y = MC$ or $Y > MC$ and $X = MC$
6	$X, Y > MC$

The below figure represents how the data bits are hidden in the pixel based on Main Case and Sub Case.

MAIN CASE GREATER THAN 1 AND SC=3



MAIN CASE GREATER THAN ONE AND SUB CASE IS 5 OR 6



After hiding the data in the pixel, the positions of pixel in which the data is stored and the number of color intensities changed is taken in a file, which is required for the user to extract the data from the image in binary form. This data is also converted into binary value and the number of binary file created depends on the number of segments in which the data is hidden.

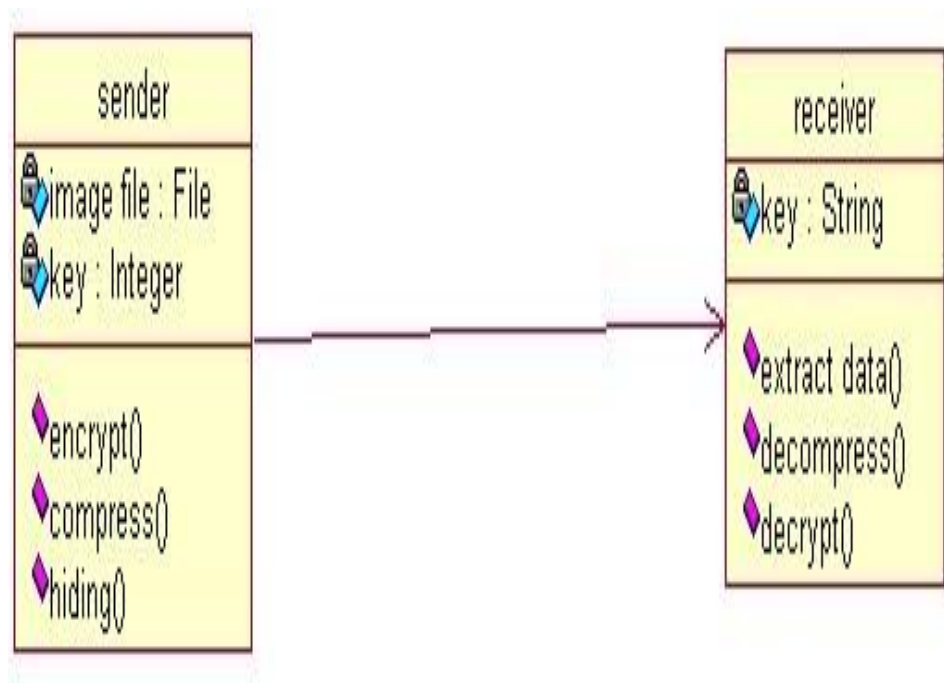
The above obtained Binary file is stored in same number of segments from the rest of the image segments, by placing six binary bits in each pixel orderly starting from the fourth pixel. The first four pixel is used to store information about the number of pixels used to hide the data in that segment of the image, and till what color the data is stored. Thus if 'n' is the number of segments used to hide the secret data, the next 'n' segments from the remaining segments is used to hide the key to extract the data from the image.

The combine algorithm is used to combine all the segments of the image into an image which resembles as the original image and this image is sent to the requested user along with the segmentation password used to extract the data from the image.

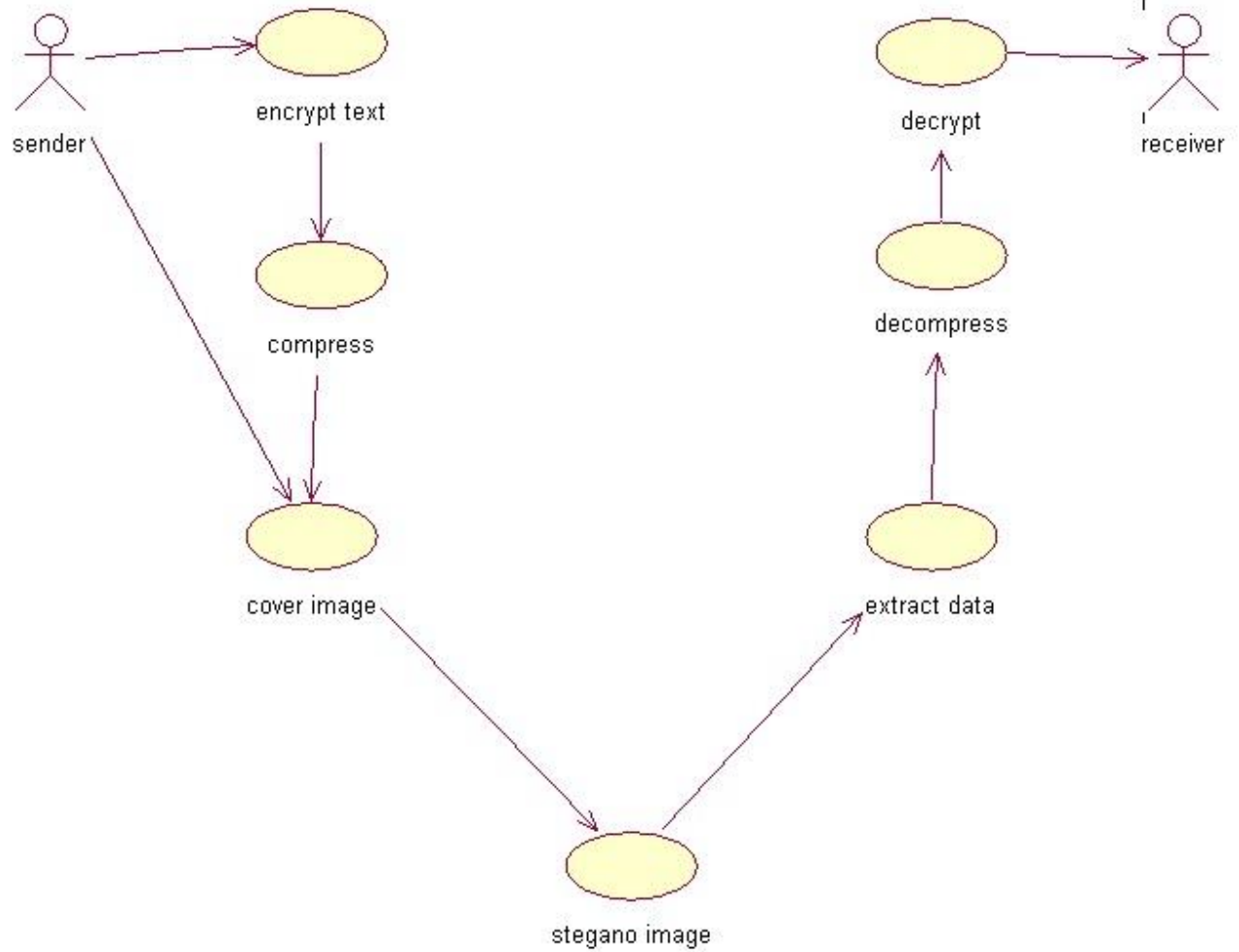
4. DIAGRAMS

To improve the readability of the project and also for better understanding of the project we go for representation using various diagrams. We will be using Class Diagram, Use Case Diagram and Sequence Diagram.

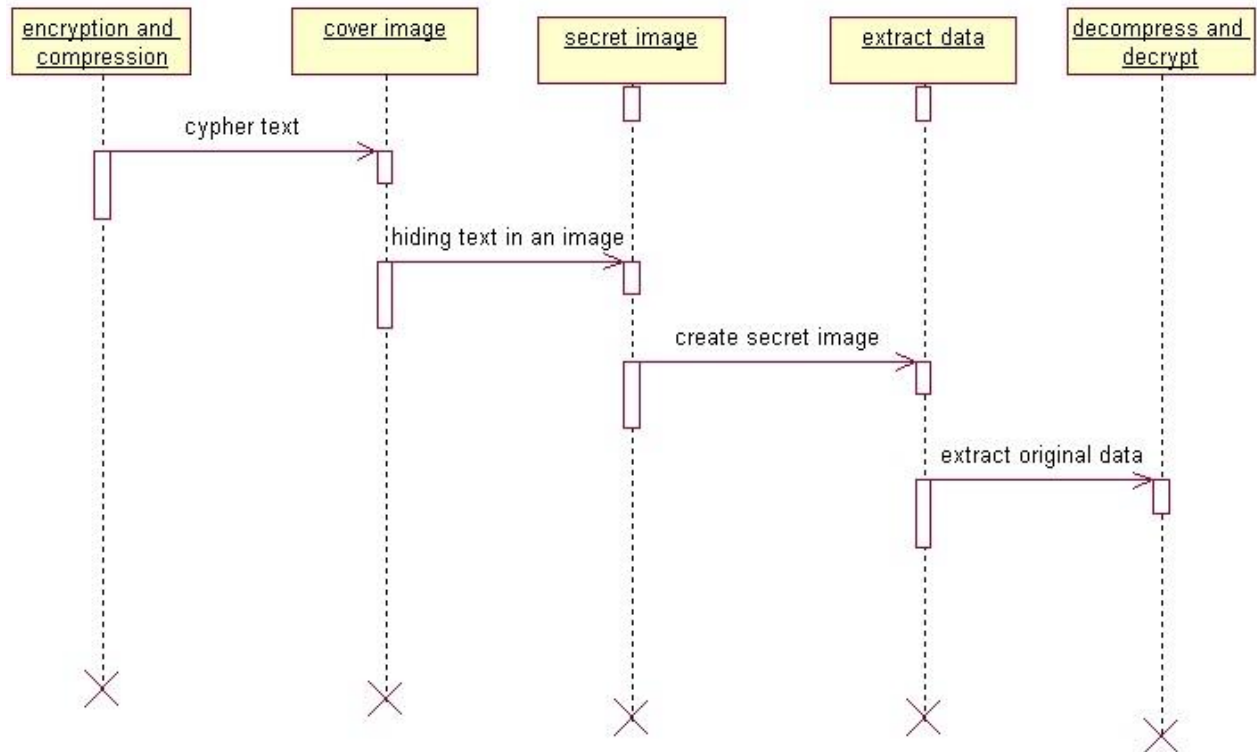
4.1 CLASS DIAGRAM



4.2 USE CASE DIAGRAM



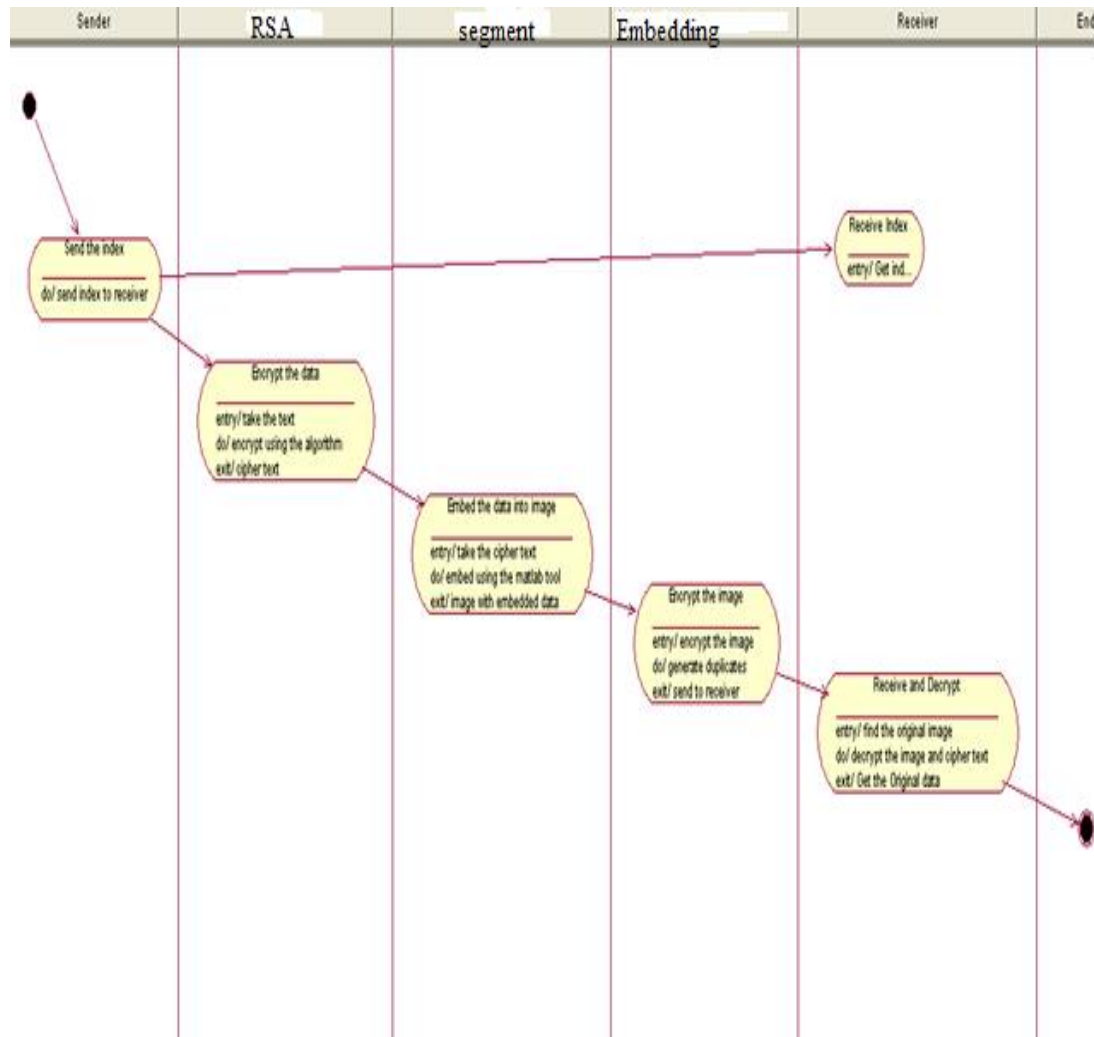
4.3 SEQUENCE DIAGRAM



4.4 COLLABORATION DIAGRAM



4.5 ACTIVITYDIAGRAM



5. SAMPLE SOURCE CODE:

5.1. FIRST LEVEL OF SECURITY:

5.1.1. RSA ENCRYPTION:

```
import java.io.*;

import java.util.*;

import java.math.BigInteger;

class RSA

{

    public static void main(String s[]) throws IOException

    {

        DataInputStream ds=new DataInputStream(System.in);

        Random rand1=new Random();

        Random rand2=new Random();

        BigInteger p=BigInteger.probablePrime(16,rand1);

        BigInteger q=BigInteger.probablePrime(16,rand2);

        BigInteger r=p.multiply(q);

        BigInteger p1=p.subtract(new BigInteger("1"));

        BigInteger q1=q.subtract(new BigInteger("1"));

        BigInteger r1=p1.multiply(q1);

        System.out.println("Enter some sample public key value:");

        int pkey=Integer.parseInt(ds.readLine());
```

```

while(true)

{

    BigInteger numgcd=r1.gcd(new BigInteger(""+pkey));
    if(numgcd.equals(BigInteger.ONE))
        break;
    pkey++;
}

    BigInteger pubkey=new BigInteger(""+pkey);
    BigInteger prvkey=pubkey.modInverse(r1);

    try
    {

        FileWriter fw=new FileWriter("keys.txt");

        FileWriter fw1=new FileWriter("rsaoutput.txt");

        String m=String.valueOf(prvkey);

        fw.write(m);

        fw.write("::");

        String n=String.valueOf(r);

        fw.write(n);

        fw.close();

        FileInputStream fin=new FileInputStream("rsainput.txt");
        int c;

```

```

while((c=fin.read())!=-1)
{
    BigInteger val=new BigInteger(""+c);
    BigInteger cival=val.modPow(pubkey,r);
    String s1=String.valueOf(cival);

    fw1.write(s1);
    fw1.write("::");

}

    fw1.close();

    System.out.println("\n\nENCRYPTION IS SUCCESSFULL\n\n");
}
catch(Exception e)
{

}

}

}

```

5.1.2: CONVERSION OF CIPHER TEXT TO BINARY TEXT:

```

import java.io.*;
public class SecretBinary
{
    public static void main(String arg[])throws IOException
    {

```



```

try
{
    BufferedReader b=new BufferedReader(new FileReader("E://no. of
seg.txt"));

    int maxsec=Integer.parseInt(b.readLine());
    for(int sec=1;sec<=maxsec;sec++)
    {
        FileInputStream fis=new
FileInputStream("E://secrettext//secret"+sec+".txt");
        FileWriter fw=new
FileWriter("E://binarysecrettext//bin"+sec+".txt");

        int c,p;
        String s1=new String();
        while((c=fis.read())!=-1)
        {
            p=(int)c;
            s1=toBina(p);
            System.out.println(p);
            fw.write(s1);
        }
        fw.close();
        System.out.println("conversion to binary is succesfull for "+sec+"
files");
    }
}
catch(Exception e){ }
}

public static String toBina(int x)
{
    int numbb=x;
    StringBuilder buf1=new StringBuilder();
    StringBuilder buf2=new StringBuilder();

```

```

    while (numbb != 0){
        int digit = numbb % 2;
        buf1.append(digit);
        numbb = numbb/2;
    }
    String binary=buf1.reverse().toString();
    int length=binary.length();
    if(length<8){
        while (8-length>0){
            buf2.append("0");
            length++;
        }
    }
    String k=buf2.toString()+binary;
    return(k);
}
}

```

5.2. HIDING DATA IN THE IMAGE:

5.2.1. DIVIDING DATA FOR EACH SEGMENT:

```

import java.io.*;
class FromBinaryFile
{
    public static void main(String arg[])throws IOException
    {
        try
        {
            FileReader frb=new FileReader(new File("E://binary text.txt"));
            DataInputStream dis=new DataInputStream(System.in);
            int kbi,rbi,i=1;
            char[] a=new char[1000];
            FileWriter fw=new FileWriter("E://secret1.txt");

```

```

PrintWriter pw=new PrintWriter(fw);
    while(frb.ready())
    {
        rbi=20;
        kbi=frb.read(a,0,rbi);
        String s=new String(a,0,kbi);

        if(s.contains("\n"))
        {
            s=s.replace("\n","").replace("\r","");
            System.out.println(s.length());
            kbi=s.length();
        }

        if(s.charAt(kbi-1)=='?')
        {
            System.out.println(i+"th image can hide only"+(kbi)+"bits");
            String s1=new String(a,0,kbi-1);
            fw.write(s1);
            pw.println();
            break;
        }

        fw.write(s);
        pw.println();
        System.out.println(s);

        i++;
    }
    fw.close();
    System.out.println("binary data is divided,stored in secret1.txt");
}
catch(Exception e){ }

```

```
}  
}
```

5.2.2 HIDING DATA IN EACH IMAGE SEGMENT BY FOLLOWING SHARING ALGORITHM:

```
import java.awt.image.BufferedImage;  
  
import java.io.*;  
  
import javax.imageio.ImageIO;  
  
import javax.imageio.stream.FileImageOutputStream;  
  
import java.awt.Color;  
  
public class Image1  
{  
  
    public static String inputdata;  
  
    public static int pos=0;  
  
  
    public static void main(String arg[])throws IOException  
    {  
  
        Image1 img=new Image1();  
  
        try{  
  
            BufferedReader br1=new BufferedReader(new  
FileReader("E:/divisions.txt"));  
  
            int rows=0,cols=0;  
  
            String line;
```

```

line=br1.readLine();

rows=Integer.parseInt(line);


line=br1.readLine();

cols=Integer.parseInt(line);


int seg=1,maxseg=rows*cols;

BufferedReader br=new BufferedReader(new FileReader("E://secret1.txt"));

FileWriter fw1=new FileWriter("E://no. of seg.txt");


while((inputdata=br.readLine())!=null && seg<=maxseg)

{

    BufferedImage img1=ImageIO.read(new File("E://file/"+seg+".bmp")); //image
segment in which data will be hidden

    pos=0;

    System.out.println(inputdata);
        //input binary data from binary text file

    FileWriter fw=new FileWriter("E://secrettext//secret"+seg+".txt"); //pixels
in which data is hidden -file

    PrintWriter pw=new PrintWriter(fw);

    if(img1.getHeight()>300|| img1.getWidth()>300){System.out.println("too large input
image");}

    int[][] maincase=new int[17][img1.getWidth()*img1.getHeight()];

```

```

int[] s2=new int[img1.getWidth()*img1.getHeight()];

int
i,j,min=0,p=0,c1,x,y,mc=0,k=0,xr1=0,yr1=0,xr,yr,mc1=0,c,np=0,k1,r,g,b,aa=0,bb=0,sbrv=0,s
bgv=0,sbbv=0;

int[] maincaseindex=new int[18];

int[] sc=new int[7];

int[] index=new int[3];

int[] coll=new int[3];

String[][] whichcolor=new String[17][img1.getWidth()*img1.getHeight()];

String [][]position=new String[17][img1.getWidth()*img1.getHeight()];

String sec="",fina="";

System.out.println("inputdata length"+inputdata.length());

for(i=0;i<18;i++)

    maincaseindex[i]=0;

System.out.println("image width="+img1.getWidth()+"**image
height="+img1.getHeight());

for(i=0;i<img1.getWidth();i++)

{

    for(j=0;j<img1.getHeight();j++)

    {

        Color col=new Color(img1.getRGB(i,j),true);

        s2[k]=img1.getRGB(i,j);

        Integer I=new Integer(i);

```

```

Integer J=new Integer(j);

p=col.getRed();

r=(int)(p/16)+1;

p=col.getGreen();

g=(int)(p/16)+1;

p=col.getBlue();

b=(int)(p/16)+1;

if(r<=g)

{

    if(r<=g)

        min=r;

    else if(b<=g)

        min=b;

}

else if(g<=b)

    min=g;

else

    min=b;

    maincaseindex[min]+=1;

    maincase[min][maincaseindex[min]]=s2[k];

    position[min][maincaseindex[min]]=I.toString()+"*"+J.toString();

if(min==r)

```

```

whichcolor[min][maincaseindex[min]]="100";

if(min==g)

whichcolor[min][maincaseindex[min]]="010";

if(min==b)

whichcolor[min][maincaseindex[min]]="001";

k++;

    }

}

for(i=1;i<=16;i++)

{

    System.out.println("maincase number="+i+"**"+maincaseindex[i]);

    for(j=1;j<=maincaseindex[i];j++)

    {

        if(pos>=inputdata.length())

        {

            System.out.println("data is hidden till "+(j-1)+"th pixel of "+i+"th maincase");

            break;

        }

        if(maincase[i][j]!='\0' && whichcolor[i][j]!=null && j!=maincaseindex[i] &&
        position[i][j]!=null )

        {

```



```

aa=Integer.parseInt(position[i][j].substring(0,(position[i][j].indexOf('*'))));

bb=Integer.parseInt(position[i][j].substring((position[i][j].indexOf('*'))+1,(position[i][j].length()-1)));

Integer AA=aa;

Integer BB=bb;

System.out.println("initial integer value of "+aa+"*"+bb+" pixel is"+img1.getRGB(aa,bb));

//converting into 8 bit binary

Color colr=new Color(img1.getRGB(aa,bb),true);

coll[0]=colr.getRed();

coll[1]=colr.getGreen();

coll[2]=colr.getBlue();

String[] bin=new String[3];

for(int ii=0;ii<3;ii++)

{

int numbb=coll[ii];

StringBuilder buf1=new StringBuilder();

StringBuilder buf2=new StringBuilder();

while (numbb != 0){

int digit = numbb % 2;

buf1.append(digit);

numbb = numbb/2;

```

```

}

String binary=buf1.reverse().toString();

int length=binary.length();

if(length<8){

    while (8-length>0){

        buf2.append("0");

        length++;

    }

}

bin[ii]=buf2.toString()+binary;

}


StringBuffer sbr=new StringBuffer(bin[0]);

StringBuffer sbg=new StringBuffer(bin[1]);

StringBuffer sbb=new StringBuffer(bin[2]);

System.out.println("inital pixel value in binary
is"+sbr.toString()+sbg.toString()+sbb.toString());

//finding mc,x,y

index[1]=whichcolor[i][j].indexOf('1');

index[0]=whichcolor[i][j].indexOf('0');

index[2]=whichcolor[i][j].lastIndexOf('0');

```

```

for(int ii=0;ii<3;ii++)

{

    switch(index[ii]){

        case 0:

            switch(ii){

                case 0:p=colr.getRed();

                break;

                case 1:p=colr.getGreen();

                break;

                case 2:p=colr.getBlue();

                break;}

            xr1=(int)(p/16)+1;

        break;

        case 1:

            switch(ii){

                case 0:p=colr.getRed();

                break;

                case 1:p=colr.getGreen();

                break;

                case 2:p=colr.getBlue();

                break;}

            mc1=(int)(p/16)+1;

```

```

        break;

    case 2:

        switch(ii){

            case 0:p=colr.getRed();

            break;

            case 1:p=colr.getGreen();

            break;

            case 2:p=colr.getBlue();

            break;}

        yr1=(int)(p/16)+1;

        break;

    }

}

//determine subcase

if(xr1<mc1 && yr1<mc1)

    sc[1]=1;


if((xr1==mc1 && yr1<mc1) || (xr1<mc1 && yr1==mc1))

    sc[2]=1;

if(xr1==mc1 && yr1==mc1)

    sc[3]=1;

if((xr1>mc1 && yr1<mc1) || (xr1<mc1 && yr1>mc1))

```

```

    sc[4]=1;

    if((xr1>mc1 && yr1==mc1) || (xr1==mc1 && yr1>mc1))

        sc[5]=1;

    if(xr1>mc1 && yr1>mc1)

        sc[6]=1;

    //find mc,xr,yr

    if(mc1<=xr1)

    {

        if(mc1<=yr1)

            mc=mc1;

        else if(yr1<=xr1)

            mc=yr1;

    }

    else

        mc=xr1;

    if(xr1<=yr1)

    {

        xr=xr1;

        yr=yr1;

    }

    else

    {

```

```

        xr=yr1;

        yr=xr1;

    }

    //start hiding after determining sub case

    if(mc>=2 && sc[1]==1)continue;

    if(mc>=1 && mc<=16 && sc[3]==1)

    {

        System.out.println("in sc3");

        if(pos<=inputdata.length()-1)

        {

            sbr=img.hide3(sbr,1);

            sec=AA.toString()+","+BB.toString()+":red"+":1";

            fw.write(sec);

            pw.println();

        }

        }//sc3 blue close

    if(pos<=inputdata.length()-1)

    {

        sbg=img.hide3(sbg,1);

        sec=AA.toString()+","+BB.toString()+":Green"+":1";

        fw.write(sec);

        pw.println();

    }

    }//sc3 green close

```

```
if(pos<=inputdata.length()-1)

{

    sbb=img.hide3(sbb,2);

    if(pos==inputdata.length()-1)

    {

        sec=AA.toString()+","+BB.toString()+":blue"+":1";

    }

    if(pos<inputdata.length()-1)

    {

        sec=AA.toString()+","+BB.toString()+":blue"+":2";

    }

    fw.write(sec);

    pw.println();

} //sc3 blue close

fina=sbr.toString()+sbg.toString()+sbb.toString();

k=new Color(sbrv,sbgv,sbbv,255).getRGB();

img1.setRGB(aa,bb,k);

} //sc3 close

if(sc[2]==1 || sc[4]==1)

{

    continue;

}
```

```

if(mc>=1 && mc<=16 && (sc[5]==1 || sc[6]==1))

{

if(sc[5]==1)

    {

        System.out.println("in sc5");

        if(pos<=inputdata.length()-1)

            {

                if(pos==inputdata.length()-1)

                    {

                        sec=AA.toString()+","+BB.toString()+":red+":1";

                        sbr=img.hide56(sbr,1);

                    }

                if(pos<inputdata.length()-1)

                    {

                        sec=AA.toString()+","+BB.toString()+":red+":2";

                        sbr=img.hide56(sbr,2);

                    }

                fw.write(sec);

                pw.println();

                sbrv=Integer.parseInt(sbr.toString(),2);

            }//sc5 red close

```



```

if(xr>=mc && pos<=inputdata.length()-1)
{
    if(pos==inputdata.length()-1)
    {
        sec=AA.toString()+"," +BB.toString()+":green"+":1";

        sbg=img.hide56(sbg,1);
    }

    if(pos<inputdata.length()-1)
    {
        sec=AA.toString()+"," +BB.toString()+":green"+":2";

        sbg=img.hide56(sbg,2);
    }

    fw.write(sec);

    pw.println();

    sbgv=Integer.parseInt(sbg.toString(),2);

} //sc5 green close

if(yr>=mc && pos<=inputdata.length()-1)
{
    if(pos==inputdata.length()-1)
    {

```

```

        sec=AA.toString()+","+BB.toString()+":blue+":1";

        sbb=img.hide56(sbb,1);

    }

    if(pos<inputdata.length()-1)

    {

        sec=AA.toString()+","+BB.toString()+":blue+":2";

        sbb=img.hide56(sbb,2);

    }


    fw.write(sec);

    pw.println();

    sbbv=Integer.parseInt(sbb.toString(),2);

    }//sc5 blue close


    fina=sbr.toString()+sbg.toString()+sbb.toString();

    k=new Color(sbrv,sbgv,sbbv,255).getRGB();

    img1.setRGB(aa,bb,k);

}

else

{

    System.out.println("in sc6");

    if( pos<=inputdata.length()-1)

```

```
{
```

```
    if(pos==inputdata.length()-1)
```

```
    {
```

```
        sec=AA.toString()+","+BB.toString()+":red+":1";
```

```
        sbr=img.hide56(sbr,1);
```

```
    }
```

```
    if(pos<inputdata.length()-1)
```

```
    {
```

```
        sec=AA.toString()+","+BB.toString()+":red+":2";
```

```
        sbr=img.hide56(sbr,2);
```

```
    }
```

```
    fw.write(sec);
```

```
    pw.println();
```

```
    sbrv=Integer.parseInt(sbr.toString(),2);
```

```
}//sc6 red close
```

```
if( pos<=inputdata.length()-1)
```

```
{
```

```
if(pos==inputdata.length()-1)
```

```
{
```

```

        sec=AA.toString()+","+BB.toString()+":green+":1";

        sbg=img.hide56(sbg,1);

    }

    if(pos<inputdata.length()-1)

    {

        sec=AA.toString()+","+BB.toString()+":green+":2";

        sbg=img.hide56(sbg,2);

    }

    fw.write(sec);

    pw.println();

    sbgv=Integer.parseInt(sbg.toString(),2);

    }//sc6 green close

    if( pos<=inputdata.length()-1)

    {

    if(pos==inputdata.length()-1)

    {

        sec=AA.toString()+","+BB.toString()+":blue+":1";

        sbb=img.hide56(sbb,1);

    }

    if(pos<inputdata.length()-1)

```

```

{

    sec=AA.toString()+","+BB.toString()+":blue"+":2";

    sbb=img.hide56(sbb,2);

}

    fw.write(sec);

    pw.println();

    sbbv=Integer.parseInt(sbb.toString(),2);

} //sc6 blue close


    fina=sbr.toString()+sbg.toString()+sbb.toString();

    k=new Color(sbrv,sbgv,sbbv,255).getRGB();

    img1.setRGB(aa,bb,k);


} //sc6 close

} //sc5 & sc6 close


} //if close


System.out.println(i+"**"+j+"**"+k+"**"+pos);

System.out.println("final binary value of"+aa+"*"+bb+"pixel is"+fina);

System.out.println("final integer value of"+aa+"*"+bb+"pixel is"+img1.getRGB(aa,bb));

```

```

} //j close

} //i close

System.out.println(i);

FileImageOutputStream fo1=new FileImageOutputStream(new
File("E://file//"+seg+".bmp"));

ImageIO.write(img1,"bmp",fo1);

        fw.close();

        seg++;

} //while close

fw1.write(String.valueOf(seg-1));

fw1.close();

System.out.println("data is hidden till "+(seg-1)+" segments and secret text files created in
secret text folder");

} //try close

        catch(Exception e)

        {}

    }

StringBuffer hide3(StringBuffer s,int x)

{

    StringBuffer color=new StringBuffer(s);

        if(x==1 && pos<=inputdata.length()-1)

```

```

{

    color.setCharAt(color.length()-1,inputdata.charAt(pos));

    pos++;

}

if(x==2 && pos<=inputdata.length()-1)

{

    if(pos==inputdata.length()-1)

    {

        color.setCharAt(color.length()-1,inputdata.charAt(pos));

        pos++;

        return(color);

    }

if(pos<=inputdata.length()-1)

    {

        color.setCharAt(color.length()-2,inputdata.charAt(pos));

        pos++;

if(pos<=inputdata.length()-1)

    {

        color.setCharAt(color.length()-1,inputdata.charAt(pos));

        pos++;

    }

}

```

```

    }

    }

        System.out.println(pos);

        return(color);

    }

StringBuffer hide56(StringBuffer s,int x)

{

    StringBuffer color=new StringBuffer(s);

    if(x==2)

    {

        if(pos<=inputdata.length()-1)

        {

            color.setCharAt(color.length()-2,inputdata.charAt(pos));

            pos++;

        }

        if(pos<=inputdata.length()-1)

        {

            color.setCharAt(color.length()-1,inputdata.charAt(pos));

            pos++;

        }

    }

}

```



```

if(x==1)

{

    if(pos<=inputdata.length()-1)

        {

            color.setCharAt(color.length()-1,inputdata.charAt(pos));

            pos++;

        }

    }

    System.out.println(pos);

    return(color);

}

}

```

5.3. COMBINING THE IMAGE SEGMENTS:

```

import java.awt.image.BufferedImage;
import java.io.*;
import java.io.File;
import javax.imageio.ImageIO;

class Combine
{
    public static void main(String arg[])throws IOException
    {

```

```

try
{
    BufferedReader br=new BufferedReader(new
FileReader("E://divisions.txt"));

    int rows=0,cols=0;
    String line;
    line=br.readLine();

    rows=Integer.parseInt(line);
    System.out.println(rows);
    line=br.readLine();
    cols=Integer.parseInt(line);
    System.out.println(cols);

    int segms = rows * cols;
    int fs1=0,s1=0,s2=0;
    int segmWidth[]=new int[segms+1];
        int segmHeight[]=new int[segms+1];
        segmWidth[0]=0;
        segmHeight[0]=0;
    int type[]=new int[segms+1];

    File[] imgFiles = new File[segms+1];
        for (int i = 1; i <= segms; i++)
        {

            imgFiles[i] = new File("E://file/" + i + ".bmp");
        }

    BufferedImage[] buffImages = new BufferedImage[segms+1];
    System.out.println(segms);

```

```

for (int i = 1; i <= segms; i++)
{
    buffImages[i] = ImageIO.read(imgFiles[i]);
    type[i] = buffImages[i].getType();
    segmWidth[i] = buffImages[i].getWidth();
    s1+=segmWidth[i];
    segmHeight[i] = buffImages[i].getHeight();

    if(i%rows==0)
    {
        if(i==rows)
        { fs1=s1;}
        s2+=segmHeight[i];
    }
}

```

```

System.out.println(fs1+"**"+s2);

```

```

BufferedImage finalImg = new BufferedImage(fs1,s2,type[1]);

```

```

int num = 1,l=1;
    int p=segmWidth[0];
    int q=segmHeight[0];

```

```

for (int i = 1; i <=cols; i++) {
    p=segmWidth[0];
    for (int j = 1; j <=rows; j++) {
        finalImg.createGraphics().drawImage(buffImages[num],
        p,q,segmWidth[l],segmHeight[i], null);
        num++;
        p+=segmWidth[l];
        l++;
    }
}

```

```

}

        q+=segmHeight[i+rows];

    }
    System.out.println("Image concatenated....."+(num-1));
    ImageIO.write(finalImg, "bmp", new File("E://finalImg.bmp"));

}

catch(Exception e){ }
}
}

```

5.5 EXTRACT INFORMATION ABOUT SECRET DATA:

```

import java.io.*;

import javax.imageio.ImageIO;

import java.awt.Color;

import java.awt.image.BufferedImage;

import java.awt.color.*;

public class ExtractSecretHiding

{

    public static String toBina(int x)

    {

        int numbb=x;

        StringBuilder buf1=new StringBuilder();

        StringBuilder buf2=new StringBuilder();

        while (numbb != 0){

```

```

        int digit = numbb % 2;

        buf1.append(digit);

        numbb = numbb/2;

    }

    String binary=buf1.reverse().toString();

    int length=binary.length();

    if(length<8){

        while (8-length>0){

            buf2.append("0");

            length++;

        }

    }

    String k=buf2.toString()+binary;

    return(k);

}

public static void main(String arg[])throws IOException

{

    BufferedReader br=new BufferedReader(new FileReader("E://no. of
seg.txt"));

    int xw=Integer.parseInt(br.readLine());

    int seg=xw,j,p,i,index=0,x,y,n,txt=1,q=0,fin=0;

```

try

```
    {  
  
        for(txt=1;txt<=seg;txt++)  
  
            {  
  
                BufferedImage img=ImageIO.read(new  
File("E://secretimage//" + (txt+seg) + ".bmp"));  
  
                FileWriter fw=new  
FileWriter("E://binarysecrettext1//" + txt + ".txt");  
  
                char a1[]=new char[6];  
  
                char a[]=new char[24];  
  
                index=0;  
  
                for(i=0;i<=3;i++)  
  
                    {  
  
                        Color coll=new Color(img.getRGB(0,i),true);  
  
                        String[] s=new String[3];  
  
                        s[0]=new String(toBina(coll.getRed()));  
  
                        s[1]=new String(toBina(coll.getGreen()));  
  
                        s[2]=new String(toBina(coll.getBlue()));
```

```

        for(j=0;j<3;j++)
        {
            a[index]=s[j].charAt(s[j].length()-2);

            index++;

            a[index]=s[j].charAt(s[j].length()-1);

            index++;

        }

    }

    String sf=new String(a);

    x=Integer.parseInt(sf.substring(0,8),2);

    y=Integer.parseInt(sf.substring(9,16),2);

    n=Integer.parseInt(sf.substring(17,24),2);

    System.out.println(x+"*" +y+"*" +n);

    int

    min=img.getWidth()<img.getHeight()?img.getWidth():img.getHeight();

    fin=x*(min)+y-4;

    q=0;

    for(i=0;i<=x;i++)

    {

```

```

for(j=0;j<min && q<=fin;j++)

{

    if(i==0 && j==0 || i==0 && j==1 || i==0 && j==2 || i==0 && j==3){ continue;}


    index=0;

    Color coll=new Color(img.getRGB(i,j),true);


    String[] s=new String[3];

    s[0]=new String(toBina(coll.getRed()));

    s[1]=new String(toBina(coll.getGreen()));

    s[2]=new String(toBina(coll.getBlue()));


    for(p=0;p<3;p++)

    {

        if(q==fin && index>n)break;

        a1[index]=s[p].charAt(s[p].length()-2);

        index++;


        if(q==fin && index>n)break;

        a1[index]=s[p].charAt(s[p].length()-1);

        index++;

    }

```



```

        String sf1=new String(String.valueOf(a1));

        if(q==fin && index>n)

        {

                sf1=new
String(String.valueOf(a1).substring(0,n));

        }

        fw.write(sf1);

        q=q+1;

        System.out.println(j+"*"+i+"*"+sf1);

    }

}

        fw.close();

    }

        System.out.println("secret data extracted in binary form and stored in
binary secrettext1 folder");

    }

    catch(Exception e){}

```

```
}
```

```
}
```

5.6 EXTRACT FROM EACH SEGMENTS:

```
import java.awt.image.BufferedImage;
```

```
import java.io.*;
```

```
import java.awt.Color;
```

```
import javax.imageio.ImageIO;
```

```
public class Extract {
```

```
    public static void main(String arg[])throws IOException
```

```
    {
```

```
        try
```

```
        {
```

```
            BufferedReader br1 = new BufferedReader(new FileReader("E://no. of  
seg.txt"));
```

```
            int i;
```

```
            int s=Integer.parseInt(br1.readLine());
```

```
            FileWriter fw=new FileWriter("E://output.txt");
```

```
            for(i=1;i<=s;i++)
```

```
            {
```

```
                BufferedReader br = new BufferedReader(new  
FileReader("E://secrettext1//"+i+".txt"));
```

```
                BufferedImage img=ImageIO.read(new  
File("E://file//"+i+".bmp"));
```

```
                String line;
```

```
                int x,y,d,red,green,blue;
```

```
                String colo,reds="",greens="",blues="";
```

```
                while ((line = br.readLine()) != null)
```

```
{
```

```
    x=Integer.parseInt(line.substring(0,line.indexOf(',')));
```

```
    y=Integer.parseInt(line.substring(line.indexOf(',')+1,line.indexOf(':')));
```

```
    colo=line.substring(line.indexOf(':')+1,line.lastIndexOf(':'));
```

```
    d=Integer.parseInt(line.substring(line.lastIndexOf(':')+1,line.length()));
```

```
    System.out.println(x+"*"+y+"*"+colo+"*"+d);
```

```
    Color col=new Color(img.getRGB(x,y),true);
```

```
    if(colo.equals("red"))
```

```
    {
```

```
        red=col.getRed();
```

```
        reds=toBina(red);
```

```
        System.out.println(reds);
```

```
        if(d==2)
```

```
        {
```

```
            fw.write(reds.charAt(reds.length()-2));
```

```
            fw.write(reds.charAt(reds.length()-1));
```

```
        }
```

```
        if(d==1)
```

```
        {
```

```
            fw.write(reds.charAt(reds.length()-1));
```

```
        }
```

```
    }
```

```

if(colo.equals("green"))
{
    green=col.getGreen();
    greens=toBina(green);
    System.out.println(greens);
    if(d==2)
    {
        fw.write(greens.charAt(greens.length()-2));
        fw.write(greens.charAt(greens.length()-1));
    }
    if(d==1)
    {
        fw.write(greens.charAt(greens.length()-1));
    }
}

```

```

if(colo.equals("blue"))
{
    blue=col.getBlue();
    blues=toBina(blue);
    System.out.println(blues);
    if(d==2)
    {
        fw.write(blues.charAt(blues.length()-2));
        fw.write(blues.charAt(blues.length()-1));
    }
    if(d==1)
    {
        fw.write(blues.charAt(blues.length()-1));
    }
}

```

```

} //while close

        } //for close
        fw.close();
    } //try close

    catch (Exception e) { }

    }
}

```

5.7 RSA DECRYPTION:

```

import java.io.*;

import java.util.*;

import java.math.BigInteger;

class NRSA

{

    public static void main(String args[]) throws IOException

    {

        int c;

        String s1="";

        String s2="";

        String s3="";

        DataInputStream ds=new DataInputStream(System.in);

        FileInputStream fin=new FileInputStream("keys.txt");

        FileInputStream fin2=new FileInputStream("rsaoutput.txt");

        FileOutputStream fout=new FileOutputStream("originaloutput.txt");
    }
}

```

```

while((c=fin.read())!=-1)

{

    if((char)c==':')

    {

        break;

    }

    else

        s1+=(char)c;

}

c=fin.read();

while((c=fin.read())!=-1)

{

    s2+=(char)c;

}

BigInteger prvkey=new BigInteger(s1);

BigInteger r=new BigInteger(s2);

while((c=fin2.read())!=-1)

{

    if((char)c!=':')

    {

        s3+=(char)c;

```

```

    }

    else if(s3!="")

        {

            BigInteger cival=new BigInteger(""+s3);

            BigInteger plval=cival.modPow(prvkey,r);

            int d=plval.intValue();

            fout.write(d);

            s3="";

        }

    }

    System.out.println("\n\nDECRYPTED SUCCESSFULLY\n\n");

    System.out.println("ORIGINAL FILE RECEIVED:\n\n");

    }

}

```

6. SAMPLE OUTPUT

6.1 RSA ENCRYPTION OUTPUT:



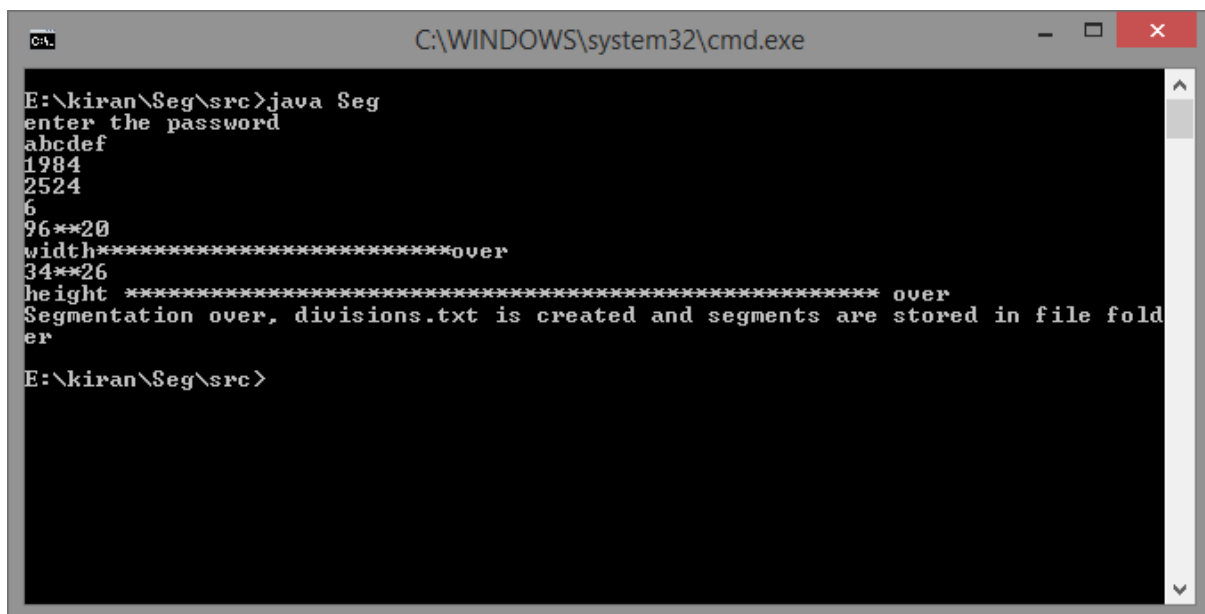
```
C:\WINDOWS\system32\cmd.exe

E:\vamsi\vam>java RSA
Enter some sample public key value:
7

ENCRYPTION IS SUCCESSFULL

E:\vamsi\vam>
```

6.2 SEGMENTATION OUTPUT:



```
C:\WINDOWS\system32\cmd.exe

E:\kiran\Seg\src>java Seg
enter the password
abcdef
1984
2524
6
96**20
width*****over
34**26
height ***** over
Segmentation over, divisions.txt is created and segments are stored in file folder
er

E:\kiran\Seg\src>
```


6.3. HIDING OF DATA IN IMAGE SEGMENTS:

```
C:\WINDOWS\system32\cmd.exe
initial pixel value in binary is010001001010011011101111
initial integer value of 98*72 pixrl is-12212240
initial pixel value in binary is01000101101001111110000
initial integer value of 98*73 pixrl is-12146447
initial pixel value in binary is010001101010100011110001
initial integer value of 98*74 pixrl is-12343058
initial pixel value in binary is010000111010100011101110
initial integer value of 98*75 pixrl is-12540439
initial pixel value in binary is010000001010010111101001
initial integer value of 98*88 pixrl is-12408590
initial pixel value in binary is010000101010100011110010
initial integer value of 98*89 pixrl is-12014861
initial pixel value in binary is010010001010101011110011
initial integer value of 98*90 pixrl is-12278809
initial pixel value in binary is010001001010001111100111
initial integer value of 98*91 pixrl is-12081691
initial pixel value in binary is010001111010010111100101
initial integer value of 98*92 pixrl is-11950105
initial pixel value in binary is010010011010011111100111
initial integer value of 98*93 pixrl is-12015641
initial pixel value in binary is010010001010011111100111
initial integer value of 98*94 pixrl is-12080662
initial pixel value in binary is010001111010100111101010
initial integer value of 98*95 pixrl is-12146198
initial pixel value in binary is010001101010100111101010
5**6130**9999**0
final binary value of98*95pixel is
final integer value of98*95pixel is-12146198
maincase number=6**0
initial integer value of 7*8 pixrl is-11423237
initial pixel value in binary is010100011011000111111011
initial integer value of 10*6 pixrl is-11489548
initial pixel value in binary is010100001010111011110100
initial integer value of 10*7 pixrl is-11489546
initial pixel value in binary is010100001010111011110110
initial integer value of 60*0 pixrl is-11423502
initial pixel value in binary is010100011011000011110010
initial integer value of 88*11 pixrl is-11489814
initial pixel value in binary is0101000010101110111101010
6**6**9999**0
final binary value of88*11pixel is
final integer value of88*11pixel is-11489814
maincase number=7**0
maincase number=8**0
maincase number=9**0
maincase number=10**0
maincase number=11**0
maincase number=12**0
maincase number=13**0
maincase number=14**0
maincase number=15**0
maincase number=16**0
17
data is hidden till 26 segments and secret text files created in secret text fol
der
E:\kiran\Hiding\src>
```

6.4. EXTRACTION FROM SEGMENTS:

```
CaL C:\WINDOWS\system32\cmd.exe
10101101
0*1*blue*2
11110101
0*7*red*2
00111000
0*7*green*2
10101011
0*7*blue*2
11110000
0*8*red*2
00111011
0*29*red*2
00111111
0*29*green*2
10101000
0*29*blue*2
11101011
0*30*red*2
00111100
0*30*green*2
10101011
0*30*blue*2
11101001
0*31*red*2
01000000
0*31*green*2
10101001
0*31*blue*2
11101011
0*58*red*2
00111101
0*6*red*2
00111000
0*6*green*2
10101100
0*6*blue*2
11101011
0*7*red*2
00111010
0*7*green*2
10101110
0*7*blue*2
11101000
0*8*red*2
00111111
0*8*green*2
10101110
0*8*blue*2
11101000
0*9*red*2
01000100
0*8*red*2
00111101
0*8*green*2
10101010
E:\kiran\Extract\src>
```

7. CONCLUSION

There are various ways in which a message that has been sent to the intended receiver by concealing it in various ways, the project shows the way of hiding it in a cover image. The message can be extracted easily unless it is provided with certain measures to conceal it. For which, various Encryption Techniques have been found. With the fast improve of technology; though we use the most secure algorithm, people can still decode it. When it comes to Steganography, there are experts who can decode the message hidden in the image. So to avoid this from happening, the best solution possible is the combination of these two messages hiding techniques and thus providing dual layer of security. In addition to this, segmentation of the image account for another layer of security. The information about the positions of the image where the data is hidden is also embedded in the same image. This process provides an easy, safe and secured way of hiding. This mechanism can be particularly useful in the cases where we need to send top secret information. It does not involve high processing overhead and so can be implemented in any platform.

8. BIBLIOGRAPHY

1. Shabir A. Parah, Javaid A. Sheikh, Abdul M. Hafiz, G.M. Bhat, Data hiding in scrambled images: A new double layer security data hiding technique, Computers and Electrical Engineering, 1 – 13, 2013
2. Stallings, William, “Cryptography and Network Security-Principles and Practice” , 5th Edition, Prentice Hall, 2011
3. Schildt Herbert, “The Complete Reference – Java2 “, 8th Edition, Tata McGraw-Hill,2011

