Data Hiding using Image Encryption in MATLAB

Information and Web Security

Assignment 3 Project Report

submitted by

Aman Chopra 140911358

Ashray Dimri 140911038

Riju Khatri 140911088

Rishabh Drolia 140911316

Suvimal Yashraj 140911394

Department of

Information and Comunication Technology,

MIT, Manipal

MANIPAL
INSTITUTE OF TECHNOLOGY
A Constituent Institute of Manipal University, Manipal

August 2017

ABSTRACT

Data hiding is a process of hidden embedding data into data sources, and encryption is a method of providing privacy protection such as confidentiality, safety of integrity and authentication of data.

In our project well be hiding data ideas for encrypted images which will include, encryption of image, embedding of data and extraction of data. This will consist of two phases, the encryption phase and the decryption phase. In the encryption phase, the encrypted image will have an encryption key associated with the data to be embedded, with the data hiding key. In decryption phase the embedded image is extracted and decrypted with encryption key. The encrypted data is extracted and decryption is made using the data hiding key.

The project objective is to provide an efficient data hiding technique and Image Encryption in which data and image can be retrieved independently. We have incorporated two tasks one of which is encypting text in image and decrypting it and the other being encrypting images and images without deteriorating the quality of image and then decrypting it using th algorithm **Inverted Data Hiding**. [1]

Keywords: Data hiding, Image Encryption, Inverse Data Hiding, Matlab.

Contents

List of Figures

ABBREVIATIONS

PLIP	:	Parameterized Logarithmic Image Processing
XOR	:	Exclusive OR
LSB	:	Least Significant Bit
PSNR	:	Peak Signal to Noise Ratio

Chapter 1

Introduction

1.1 Data Hiding

Cryptography is the science of writing a hidden code or the study of mathematical techniques related to aspects of data security such as confidentiality, data integrity, entity authentication and data origin authentication. The basic cryptographic ideas used are Symmetric or Hidden key cryptography and Asymmetric or Public key cryptography as shown in figure 1.1. It employs complex computational algorithms for encryption and decryption. In order to reduce the complexity, Cryptography ideas can be used.

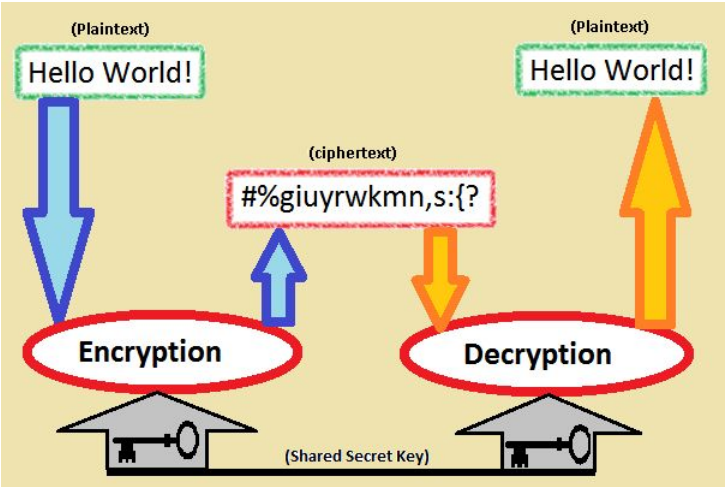

Figure 1.1: Cryptography

The main need of cryptography is to hide the data with the hidden key without knowing to the user. The fast progression of data exchange in electronic way, data security is becoming important in data storage and transmission. Because of widely using images in industrial process, it is important to protect the confidential image data from unauthorized access. Security is an important issue in storage communication and communication of images, and encryption is one of the ways to ensure security. Images are different from text.

Although the traditional cryptosystems to encrypt images directly, it is not a good idea for two reasons. One is that the image size is almost always much greater than that of text. Therefore, the traditional cryptosystems need much time to directly encrypt the image data. The other problem is that the decrypted text must be equal to the original text. In order to transmit hidden images to other people, a variety of image encryption ideas have been proposed. The security of message transmission is very important in the modern computerized and interconnected world. Security problems, such as modification, forgery, duplication, and others, on the Internet have been focused on inevitable. [2]

It has proposed an encryption method using image steganography concept and PLIP model. These image encryption ideas have the advantage of the lossless recovery of the original message. However, due to the disturbances in the transmission media, it makes harder to execute the decryption algorithm for the encrypted data with noise. In this project, we are following a new encryption method to recover the original image without distortion. In this proposed system, three methods are implemented for image encryption. First method is inverse data hiding method in which the image is encrypted by doing XOR operation bit -by-bit. The second method is Huffman coding method in which the image is scrambled using periodicity.

1.2 Steganography

Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video as shown in figure 1.2. 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. [3]

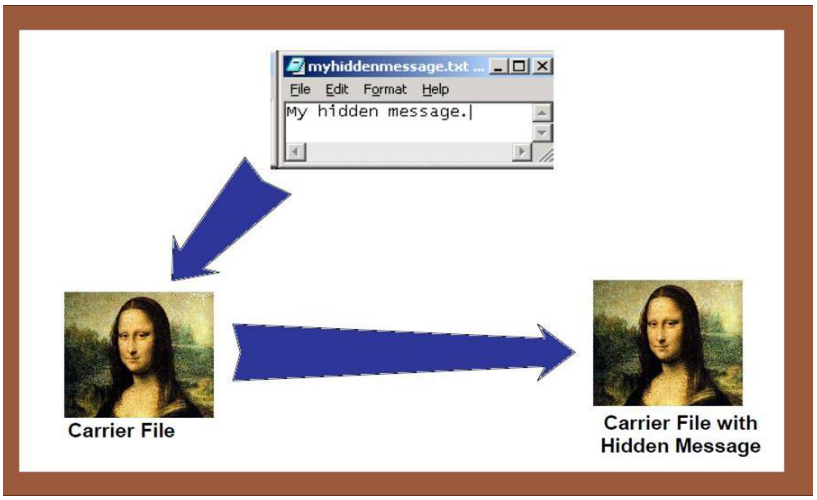

Figure 1.2: Steganography

Whereas cryptography is the practice of protecting the contents of a message alone, steganography is concerned with concealing the fact that a secret message is being sent as well as concealing the contents of the message.

Steganography includes the concealment of information within computer files. In digital steganography, electronic communications may include steganographic coding inside of a transport layer, such as a document file, image file, program or protocol. Media files are ideal for steganographic transmission because of their large size.

Chapter 2

Methodology

The program can Encode Text into Image, Decode Text from Image, Encrypt image into another image and decrypt image from the encrypted image.

2.1 Algorithm

1. Hiding an image inside the other.
2. Only two bit in pixel of the image 1 is affected.
3. Image2 is split store in two diagonally opposite quadrants.
4. The first image is resize to double its original size.
5. Logically the image is divided to four partitions
6. The bits of the image to be hid is stored in the first image as shown in figure 2.1

2.2 Encrypting Text

The first part of the project is hiding text using Image Encryption. Here, the text to be hidden is written in a text file. The maximum number of characters

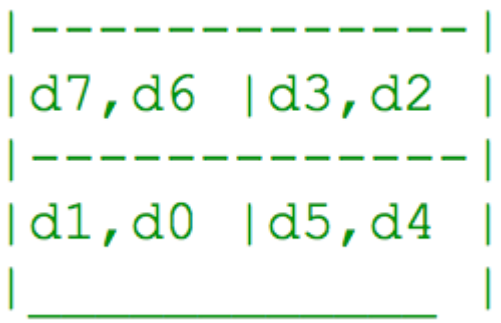

Figure 2.1: Arrangement of bits in quadrants

that can be hidden in the image is equal to the product of width and height of the image, in pixels. The image in first converted into binary values and stored into a matrix. Then the text in the file is also converted to binary values and stored in a matrix. After that the corresponding binary values of the two matrices are added, and this is how data encryption has been performed. [1]

2.3 Decrypting Text

The second part of the project is decrypting the text hidden in an image. For decryption, the encrypted image and the original image are compared, and the corresponding binary values are stored in a matrix, and then converted to text and written in another file.

2.4 Encrypting Image

The third part is encrypting an image into another image. This can be done by increasing the size of image 1(image used to hide image 2) then encoding pixel details of image 2 into image 1. By doing this the quality of image 2 will not be affected. Also only 2 bits in every pixel of image 1 is used for encoding this is because the encoded image does not show any patches of image 2. The bits in every pixel of the image 2 is split into four and placed in four different

locations in image 1. So it provides some encryption. So it will be hard for others to decode the hidden image. The details of the methods used are given below:

2.4.1 Inverse Data Hiding

The inverse data hiding scheme figure 2.2 is used for encrypting the images. It is mainly used to embed additional message into some distortion, with an inverse manner so that the original content can be perfectly restored after extraction of the hidden messages. The proposed scheme is the content owner encrypts the original uncompressed image using an encryption key to produce an encrypted image. [1] The data embedding algorithm hides data into the encrypted image using LSB method with hidden key. According to the data-hiding key, he can further extract the embedded data and recover the original image from the decrypted version. Within an encrypted image containing additional data, a receiver must first decrypt it using the encryption key and the decrypted version is similar to the original image.

The detailed instructions are as follows:

1. Determine an image, which is determined as a cover image.
2. Convert the cover image to binary.
3. Random bits are generated which act as the encryption key.
4. XOR operation is performed between the random bits generated and cover image.
5. Convert the above binary image to grayscale image. Thus, encrypted image is obtained.

Figure 2.2: (a) Original image, (b) Encrypted image (c) Encrypted image with Encrypted message and (d) decrypted image.

2.4.2 LSB Method

This method is used to hide the data in encrypted images. In 8-bit gray scale images are selected as the cover media. Cover-images with the hidden messages embedded in them are called stego- images. For data hiding methods, the image quality refers to the quality of the stego-images. [4] One of the common techniques is based on manipulating the least-significant-bit (LSB) planes by directly replacing the LSBs of the cover image with the message bits. LSB methods typically achieve high capacity. This allows a person to hide data in the cover image and make sure that no human could detect the change in the cover image. The LSB method usually does not increase the file size, but depending on the size of the data that is to be hidden inside the file, the file can become noticeably distorted.

The detailed procedure:

1. Determine an encrypted image (cover image).
2. Determine the message or data that should be hidden into the cover image.
3. Convert all the pixel values of the encrypted image from grayscale to 8-bit binary values.
4. Embed the message or data into the cover image by hiding the data into the LSB bit of cover image.
5. The original image along with the data is recovered with less distortion and the PSNR value is calculated.

2.5 Decrypting Image

For decrypting the image, the encrypted images resolution is decreased and made the same as the original images resolution. [5] Then the matrix that had

been formed while encrypting is used as a key to decrypt the image.

The detailed system architecture followed in this project is shown in figure 2.3

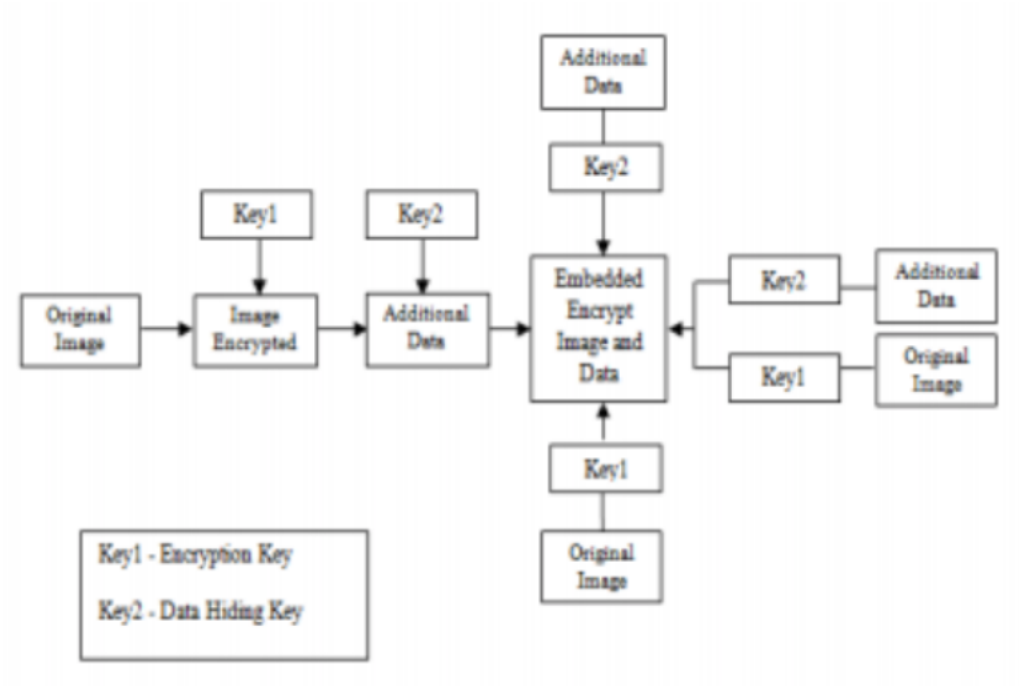

Figure 2.3: System Architecture

Chapter 3

Conclusion and Future Works

In this project, we discussed and demonstrated that the image encryption algorithm is efficient and highly secure. All parts of the proposed encryption system were simulated using MATLAB. The scheme can resist most known attacks, such as statistical analysis and brute-force attacks. [6] By comparing these entire algorithms the scalable image encryption gives the best image quality and has high level of security with less computation. It is highly robust towards cryptanalysis.This work can be extended to implement image encryption using neural networks and for many securities, authentication purpose it can be used. It can be also used in many emerging fields. Because of the rapid development of information technology, it is becoming increasingly difficult to maintain a satisfactory level of information security. New information security theory and advanced technologies are required urgently. Most encryption methods are designed for text information, and some encryptions are easily to be attacked. [7]

This project shows that the proposed encryption algorithm has high level of security with less computation, it shows how to safe guard the sensitive data on network and it helps in securing the important information inside an image or picture making sure that the quality of image is not compromised.

References

[1] Inverse data hiding. [Online]. Available: www.researchgate.net/publication/304625729_Inverse_Data_Hiding_in_Classical_Image_by_using_Scalable_Image_Encryption

[2] Image encryption. [Online]. Available: www.mathworks.com/matlabcentral/fileexchange/27698-image-encryption

[3] Steganography. [Online]. Available: wikipedia.org/wiki/Steganography

[4] [Online]. Available: www.instructables.com

[5] [Online]. Available: www.arpnjournals.com/jeas/research_papers/rp_2014/jeas_0514_1083.pdf

[6] Image encryption. [Online]. Available: www.slideshare.net/ramamurthyaashish/image-encryption-and-decryption

[7] [Online]. Available: www.ijircce.com/upload/2014/march/32_Inverse.pdf

Appendices

Appendix A

Matlab Code

Listing A.1: Source Code

```
%Program for hiding an image inside the other
%*****************************************************
% ---Only two bit in pixel of the image 1 is affected
% ---image2 is split & store in two diagonally opposite quadrants
% Algorithm
% the first image is resize to double its original size
% logically the image is divided to four partitions
% the bits of the image to be hid is stored in the first image as
% shown below

% |-------------|
% |d7,d6 |d3,d2 |
% |-------------|
% |d1,d0 |d5,d4 |
% |_____________ |

% ****************program*******************
clc; % clear the command window
```

```
clear; % clear the workspace
disp(' ');
disp(' ***** IMAGE HIDER 2.0 *****');
disp('___Program for hidimg one image inside the other image___');
disp(' ');
disp('_____________________________________________________');
task = input('---Encode Text into Image :- 1 \n---Decode Text from
    Image :- 2\n---Encrypt image into another image :-3\n---Decrypt
    image from the encrypted image :-4\n Enter your task:');
% select task
if isempty(task)
task=1;
end
if task == 1
FID = fopen('myfile.txt', 'rb'); %opening text file, integer file
    identifier obtained in fid
Str = fread(FID, [1,inf], 'char');
  %reading text file, reads binary data from the specified file and
      writes it into matrix Str
  %size[1,inf] - read elements to fill an 1-by-inf matrix, in column
      order inf read to the end of the file.
  %reads the file according to the data format specified

fclose(FID);              %closing the file
Str=uint16(Str);          %converting to unsigned 16 bit integer for
    proper calculation.

            %Any element outside limit gets rounded off to nearest
                endpoint
```

```
x=imread('original.png'); %reading the image file into x matrix
    which contains binary values

x=uint16(x);              %conversion to 16 bit
[x_row,x_col]=size(x);    %returns the dimensions of the matrix x

c=numel(Str);             %counting characters (numel=number of
    elements)
a=1;

%encrypting loop
for i=1:x_row
    for j=1:x_col
        if(a<=c)
     if(x(i,j)+Str(a)>255)        %if greater than 255 ie 8 bits
         then putting it in 8 bit form.

                temp=x(i,j)+Str(a)-256;
            else
                temp=x(i,j)+Str(a);
            end
            z(i,j)=uint8(temp);          %converting back to default
        else
            z(i,j)=uint8(x(i,j));        %putting original image bits
                for return of encrypted image

        end
        a=a+1;              %incrementing count of characters in text
            file.
    end
```

```
end


imwrite(z,'encrypted.png') %writing the encrypted data as pixels in
    image

imshow(z)

end

if task ==2
    x=imread('encrypted.png'); %reading encrypted image
y=imread('original.png'); %reading non-encrypted image

x=uint16(x);  %16 bit conversion
y=uint16(y);  %16 bit conversion

[x_row, x_col]=size(x);

b=0;k=1;
%decrypting loop
for i=1:x_row
    for j=1:x_col
        if(x(i,j)>=y(i,j))
            a=x(i,j)-y(i,j);
        else
            a=256+x(i,j)-y(i,j);
        end

        if(a~=0)
```

```
            z(k)=uint8(a);
            k=k+1;
        else
            b=1;
            break;
        end
    end
    if(b==1)
        break;
    end
end

fid=fopen('decrypted.txt','w'); %creating text file to write
    decrypted data
for i=1:k-1
    fprintf(fid,'%c',z(i)); %writing to file
end
disp('The image has been decrypted and the text in written in
    decrypt.txt')
end

    if task == 3

% reads two image files
x = imread(input(' Welcome to Encoder\n Enter the first image file
    name: ','s'));
y = imread(input(' Enter the image to be encrypted : ','s'));
% check compatibility
sx = size(x);
sy = size(y);
```

```
x=imresize(x,[2*sy(1),2*sy(2)]); %x is 2 times the size of y

% clearing Ist files last two lsb bits & moving IInd files msb bits
   to lsb bits
x1 = bitand(x,252);  %bitwise 'and' clearing the LSB bits
y1 = bitshift(y,-4); %shifting to the right, or dividing by
   2^ABS(4) and truncating to an integer y1
% we get 4 bits of the msb of image 2
y1_= bitand(y1,12);  %and with 1100 so we get d7 and d6

y1_= bitshift(y1_,-2); %2 MSB1 bits d7 and d6

y1 = bitand(y1,3);   % and with 0011 so we get MSB2 d5 and d4

% clearing II image's msb bits
y_lsb1 = bitshift(bitand(y,12),-2);
y_lsb2 = bitand(y,3);
% inserting IInd to Ist file
z=x1;
for j=1:sy(2) % y variation
for i=1:sy(1) % x variation
for k=1:3
%we enter the bits to each quadrant for further encryption.

% IInd quadrant
z(i,j,k) = bitor(x1(i,j,k), y1_(i,j,k));
% IV th quadrant
z(i+sy(1),j+sy(2),k) = bitor(x1(i+sy(1),j+sy(2),k), y1(i,j,k));
% I st quadrant
```

```
z(i+sy(1),j,k) = bitor(x1(i+sy(1),j,k), y_lsb1(i,j,k));
% IIIrd quadrant
z(i,j+sy(2),k) = bitor(x1(i,j+sy(2),k), y_lsb2(i,j,k));
end
end
end
% display the first image
figure(1)
image(x);
xlabel(' Ist Image ');
% display IInd image
figure(2);
image(y);
xlabel(' IInd Image ');
% display encoded image
figure(3);
image(z);
xlabel(' Encoded Image ');
% saving file
sav=input('Do you want to save the file y/n [y] ','s');
if isempty(sav)
sav='y';
end
if sav == 'y'
name=input('Enter a name for the encoded image: ','s');
if isempty(sav)
name='encoded_temp';
end
name=[name,'.bmp']; % concatination
imwrite(z,name,'bmp');
```

```
end


if task==4
% Decoding encoded image
clear;
z=imread(input(' Welcome to Decoder\n Enter the image file to be
    decoded:','s'));
sy = size(z)/2; % take the size of input file
% xo is fist file- obtained by clearing lsb bits, yo is IInd file
    right
% shifting z by 4 bits
xo=bitand(z,252);
xo=imresize(xo,[sy(1),sy(2)]); % reduce the resolution to half so
%that it becoms the original image's resolution
for j=1:sy(2) % y variation
for i=1:sy(1) % x variation
for k=1:3
zout1(i,j,k) = bitshift(bitand(z(i,j,k),3),2);
zout2(i,j,k) = bitand(z(i+sy(1),j+sy(2),k), 3);
zout3(i,j,k) = bitshift(bitand(z(i+sy(1),j,k),3),2);
zout4(i,j,k) = bitand(z(i,j+sy(2),k),3);
end
end
end
zout = bitshift((zout1+zout2),4)+zout3+zout4;
yo = zout;
% display Ist & IInd image from encoded image
figure(4);
image(xo);
xlabel('Ist Decoded Image ');
```

```
figure(5);
image(yo);
xlabel('IInd Decoded Image');
% saving file
sav=input('Do you want to save the file y/n [y] ','s');
if isempty(sav)
sav='y';
end
if sav == 'y'
name1=input('Enter a name for the first image: ','s');
name2=input('Enter a name for the second image: ','s');
if isempty(name1)
name1 = 'Ist_temp';
end
if isempty(name2)
name2 = 'IInd_temp';
end
name1 = [name1,'.bmp'];
name2 = [name2,'.bmp'];
imwrite(xo,name1,'bmp');
imwrite(yo,name2,'bmp');
end
end
end
```

Appendix B

Plagiarism report.

Data Hiding Using Image Encryption in Matlab

ORIGINALITY REPORT

12%

SIMILARITY INDEX

PRIMARY SOURCES

1	wikipedia.org/wiki/Steganography Internet	90 words — 4%
2	www.mathworks.com/matlabcentral/ fileexchange/ 27698-image-encryption Crossref	68 words — 3%
3	www.instructables.com Internet	46 words — 2%
4	www.researchgate.net/publication/ 304625729_Inverse_Data_Hiding_in_Classical _Image_by_using_Scalable_Image_Encryption Publications	25 words — 1%
5	www.arpnjournals.com/jeas/research_papers/ rp_2014/jeas_0514_1083.pdf Publications	23 words — 1%
6	www.slideshare.net/ramamurthyaashish/ image-encryption-and-decryption Internet	16 words — < 1%
7	www.ijircce.com/upload/2014/march/ 32_Inverse.pdf Publications	8 words — < 1%

EXCLUDE QUOTES ON
EXCLUDE BIBLIOGRAPHY OFF
EXCLUDE MATCHES OFF