

Image Encryption Using Steganography

PROJECT REPORT

Submitted By:

Salonee Gupta	16BCE0194
Rishab Gupta	16BCE0757

Course Code: CSE4019

Image Processing

Professor Jaisakthi
School of Computer Science and Engineering



VIT[®]
Vellore Institute of Technology

October 2018

CERTIFICATE

This is to certify that the project work entitled “**Image Encryption Using Steganography**” that is being submitted by **Salonee Gupta** and **Rishab Gupta** for the course **CSE4019 - “Image Processing”** is a record of bonafide work done under my supervision. The contents of this project work in full or in parts, have neither been taken from any other source nor have been submitted for any other course.

Signature of Faculty

Prof. Jaisakthi S M

ACKNOWLEDGEMENT

We would like to thank our professor **Jaisakthi S M** for her precious guidance and constant support and the Dean of School of Computer Science and Engineering for their pleaded information and opportunity given to us for completion of our project.

Submitted by:

Salonee Gupta

Rishab Gupta

Table of Contents

S. No	Topic	Page #
1.	Problem Statement	5
2.	Abstract	6
3.	Introduction	7
4.	Proposed Design	10
5.	Algorithm	13
6.	Implementation	18
7.	Conclusion	23
8.	References	24

1. PROBLEM STATEMENT

Private communication is becoming difficult day by day for common user's due to a sudden upsurge in hackers. Finding a safe method to send secure messages has become a hard task for the users. We need to find a way to send secure, well encrypted messages to the receiver without carrying a risk of it being hacked.

Steganography is one of the most powerful techniques to conceal the existence of hidden secret data inside a cover object. Images are the most popular cover objects for Steganography and in this work image steganography is adopted. Embedding secret information inside images requires intensive computations, and therefore, designing Steganography in hardware speeds up Steganography.

2. ABSTRACT

Steganography is the art of hiding information within other information in such a way that it is hard or even impossible to identify the existence of any hidden information. There are many different carriers for steganography. Of which, most popular ones are digital images. Due to recent developments in steganalysis, providing security to personal contents, messages, or digital images using steganography has become difficult.

By using steganalysis, one can easily reveal existence of hidden information in carrier files. This project introduces a novel steganographic approach for covert communications between two private parties. The approach introduced in this project makes use of both steganographic as well as cryptographic techniques. The process involves converting a secret image into a text document, then encrypting the generated text into a cipher text using a key (password) based encryption algorithm, and finally embedding the cipher text on to a cover image. This embedding process is carried out using a threshold based scheme that inserts secret message bits into the cover image only in selected pixels. The security to maintain secrecy of message is achieved by making it infeasible for a third person to detect and retrieve the hidden message.

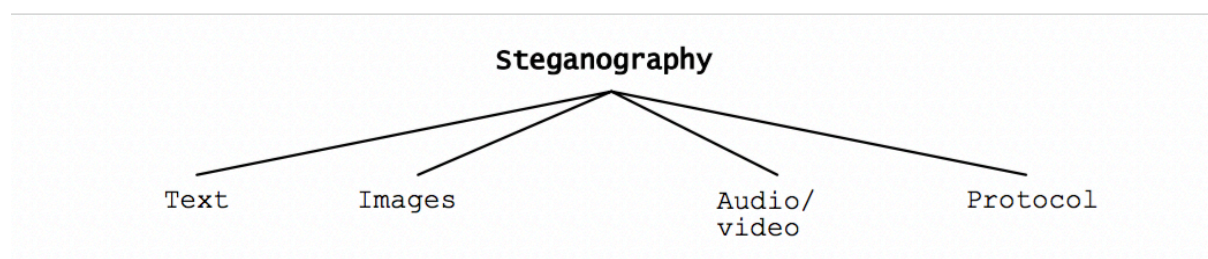
Many different carrier file formats can be used, but digital images are the most popular because of their frequency on the Internet. For hiding secret information in images, there exists a large variety of steganographic techniques some are more complex than others and all of them have respective strong and weak points. Different applications have different requirements of the steganography technique used. For example, some applications may require absolute invisibility of the secret information, while others require a larger secret message to be hidden. This paper intends to give an overview of image steganography, its uses and techniques. It also attempts to identify the requirements of a good steganographic algorithm and briefly reflects on which steganographic techniques are more suitable for which applications.

3. INTRODUCTION

Since the rise of the Internet one of the most important factors of information technology and communication has been the security of information. Cryptography was created as a technique for securing the secrecy of communication and many different methods have been developed to encrypt and decrypt data in order to keep the message secret. Unfortunately, it is sometimes not enough to keep the contents of a message secret, it may also be necessary to keep the existence of the message secret. The technique used to implement this, is called steganography.

3.1 Different kinds of steganography

Almost all digital file formats can be used for steganography, but the formats that are more suitable are those with a high degree of redundancy. Redundancy can be defined as the bits of an object that provide accuracy far greater than necessary for the object's use and display. The redundant bits of an object are those bits that can be altered without the alteration being detected easily. Image and audio files especially comply with this requirement, while research has also uncovered other file formats that can be used for information hiding. The figure shows the four main categories of file formats that can be used for steganography.



Hiding information in text is historically the most important method of steganography. An obvious method was to hide a secret message in every nth letter of every word of a text message. It is only since the beginning of the Text Images Audio/ video Protocol Internet and all the different digital file formats that it has decreased in importance. Text steganography using digital files is not used very often since text files have a very small amount of redundant data.

Given the proliferation of digital images, especially on the Internet, and given the

large number of redundant bits present in the digital representation of an image, images are the most popular cover objects for steganography. This paper will focus on hiding information in images in the next sections. To hide information in audio files similar techniques are used as for image files.

One different technique unique to audio steganography is masking, which exploits the properties of the human ear to hide information unnoticeably. A faint, but audible, sound becomes inaudible in the presence of another louder audible sound. This property creates a channel in which to hide information. Although nearly equal to images in steganographic potential, the larger size of meaningful audio files makes them less popular to use than images.

The term protocol steganography refers to the technique of embedding information within messages and network control protocols used in network transmission. In the layers of the OSI network model there exist covert channels where steganography can be used. An example of where information can be hidden is in the header of a TCP/IP packet in some fields that are either optional or are never used.

3.2 Objective and goal of the project

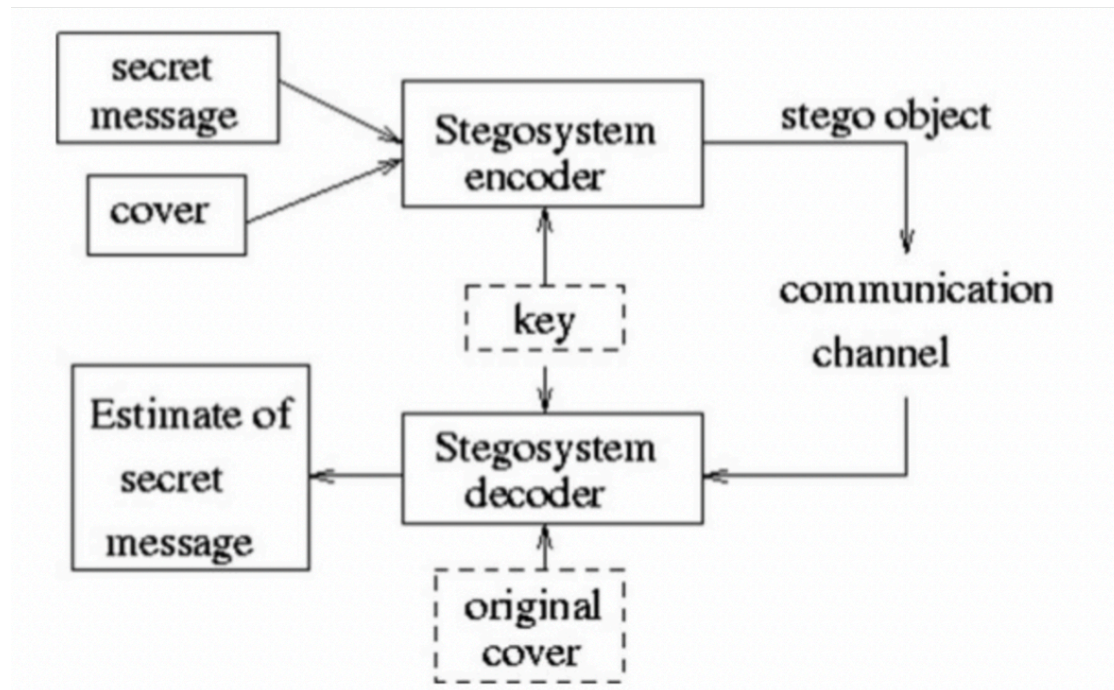
Lately, exponential growth of technology in every aspect of life is observed. Improvement of technology provides facilities to both users and hackers/intruders too. Advancement in technology that encourages hackers/intruders' activities result in lack of security to user's confidential data. The most common and popular techniques for data hiding that have been in use since long time are cryptography and steganography. Cryptography: There are many possible definitions for cryptography. One of which is, "The computerised encoding and decoding of information" to define cryptography. This is a process of converting a message from a human readable or understandable form (plaintext) to non-understandable format (cipher text) to enable secure sending and back to original format at the other receiving end. The cipher text in cryptography always reveals static information of plaintext. Many methodologies were introduced that follow their own strategy, but all the methodologies use some patterns. The underlying idea in pattern based approach is to decode the encoded message, that is, using a pattern of one's own choice or a standard pattern, a sender encodes the message and thus generates a cipher text. The receiver uses the same pattern and decodes the cipher text to generate message (plaintext). Over a period, cryptographic approaches evolved over phases.

It is suggested that a key should be used in the process of encoding and decoding a message. Based on this concept of keys, cryptography is further classified into two types, symmetric-key cryptography and public-key cryptography. In case of symmetric key cryptography, same key has to be used by both sender and the receiver while encoding and decoding respectively. In contrast, in the case of public key cryptography, the keys used by the sender and the receiver are different. Steganography: It can be defined as "The art and science of communicating in a way which hides the existence of the communication".

A steganographic model facilitates hiding or embedding of sender's secret message in a file (carrier) that does not give out a clue about the existence of secret message in it when viewed. For this, any media format or file format like .bmp, .doc, .gif, .jpeg, .mp3, .ppt, .txt and .wav is taken as a carrier that can act as cover for the sender's message, that is, a message here is hidden in a carrier and that carrier is transmitted. The underlying operation of this methodology is both logical and technical. In general, a steganography algorithm takes a secret message and a carrier as input and gives a carrier message as output (in which the message is embedded). In the process of steganography, the carrier which hides the message in it will be sent to the receiver.

The carrier gives the receiver no information about the message but reveals it only after using the tool or algorithm that is used by the sender. Both cryptography and steganography have found usage in many applications. For example, transmission of attack plans by military teams to hide information about their strategies. Many other applications of data hiding techniques other than its original objective, have gained importance, which include authentication and identification, watermarking and transmitting passwords etc.

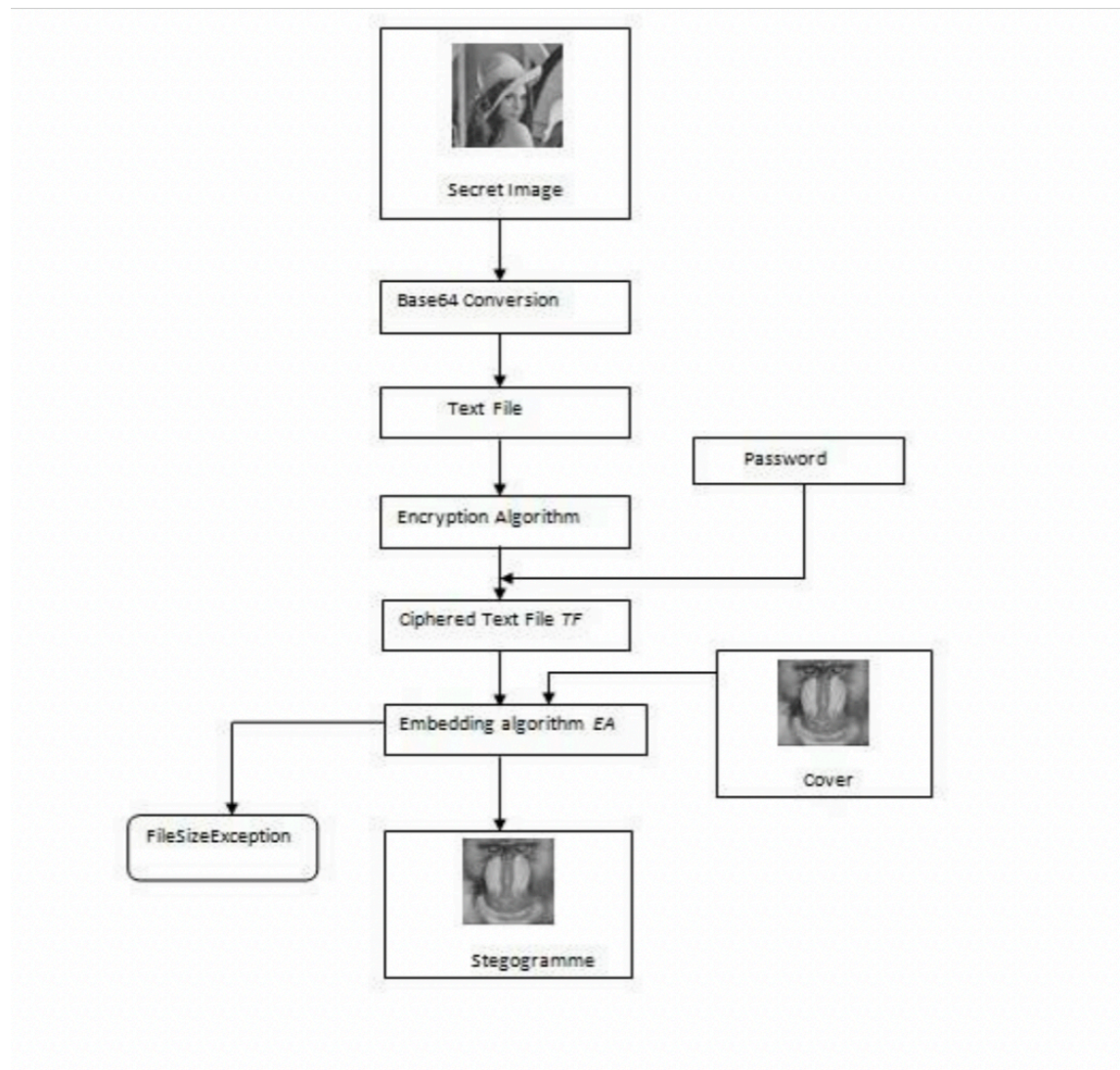
4. PROPOSED PROJECT



4.1 Flow of Execution

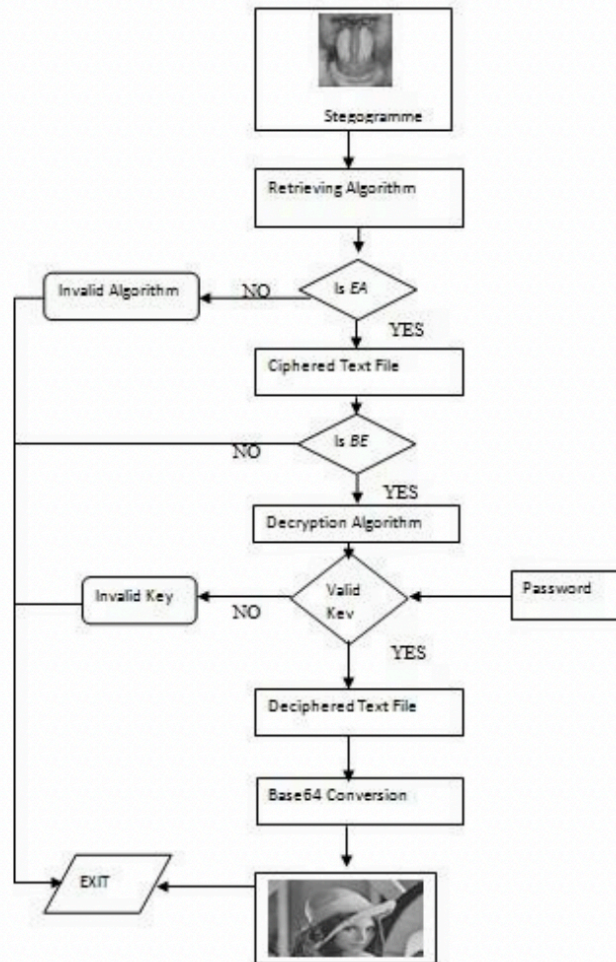
The purpose of this project is to hide an image in other image, so a secret image would be an input. At first, the secret image is converted to a text file using Base64 conversion. Then the generated text file is encrypted with a password based encryption algorithm to generate an encrypted text file called cipher text. Using a customised embedding algorithm, cipher text is embedded on to a cover image. The output is the stegogram (a cover image with a secret message embedded in it).

4.1.1 Flow of execution of Sender's Operation



4.1.2 Flow of operation of receiver's operation

To read the hidden message (secret image), the stegogramme has to be decrypted. So, the stegogramme is used as input to the retrieving algorithm. If the retrieving algorithm is not the same as the embedding algorithm, there is no way that the correct output can be obtained. The correct output from retrieving algorithm is the cipher text is used as input to the decryption algorithm. This decryption algorithm is the same as the encryption algorithm; otherwise the secret message cannot be determined. And also, the decryption algorithm takes a key (password) to generate plaintext.



5. ALGORITHM

5.1 (a) LSB Encode

```
function steg = steg_lsb_encode(carrier, message_bin)
% steg_lsb_encode Encodes a message in the least significant bits of a
% carrier image
% INPUTS
% carrier    - Image to hide message in. Should only have one colour
%              channel.
% message_bin - The message, in the form of binary data.
% OUTPUTS
% steg       - Image with encoded message.

if (length(message_bin(:)) > length(carrier(:)))
    error('secret is too large to fit within carrier');
else
    message_padded = zeros(length(carrier(:)), 1);
    message_padded(1:length(message_bin)) = message_bin;

    d = size(carrier);
    message_padded = logical(reshape(message_padded, d));

    steg = mod(message_padded, 2) + double(bitshift(carrier, -1) * 2);
end

end
```

5.2 (b) LSB Decode

```

function message_bin = steg_lsb_decode(steg)
% steg_lsb_decode Extracts binary data from least significant bits
% INPUTS
% steg      - Image to extract message from.
% OUTPUTS
% message_bin - Extracted message.

message_bin = mod(steg,2);
message_bin = reshape(message_bin, [], length(length(message_bin)));

end

```

5.3 (a) WDCT Encode

```

function [im, im_wavelet] = steg_wdct_encode(im, secret_msg_bin, mode,
frequency_coefficients, persistence)
%UNTITLED7 Summary of this function goes here
% Detailed explanation goes here

[ll lh hl hh] = dwt2(im, mode);
hh = steg_dct_encode(secret_msg_bin, hh, frequency_coefficients, persistence);
im_wavelet = [ll, lh; hl, hh];
im = idwt2(ll, lh, hl, hh, mode);

end

```

5.4 (b) WDCT Decode

```

function [extracted_msg_bin] = steg_wdct_decode(im, mode, frequency_coefficients)
%UNTITLED8 Summary of this function goes here
% Detailed explanation goes here

[~, ~, ~, hh] = dwt2(im, mode);

extracted_msg_bin = steg_dct_decode(hh, frequency_coefficients);

end

```

5.5 (a) Fusion Encode

```
function [im_stego, im_wavelet_original, im_wavelet_stego] = steg_fusion_encode(im,
secret_msg_bin, alpha, mode)
% steg_fusion_encode Performs steganography using fusion through wavelets
% See paper: "Data Hiding techniques Based On Wavelet-like transform and Complex
Wavelet Transforms"
% INPUTS
% im          - Carrier image. Must only have 1 colour channel.
% secret_msg_bin - Binary data to encode.
% alpha       - Preprocessing parameter (preferably 0.05).
% mode        - Wavelet mode, e.g. 'haar'.
% OUTPUTS
% im_stego     - Carrier image with message embedded.
% im_wavelet_original - For debugging. Wavelet transform before stego.
% im_wavelet_stego - For debugging. Wavelet transform after stego.

secret_msg_bin_len = length(secret_msg_bin);
uint8_max = 255;

% Normalise
im = double(im) / uint8_max;

% Preprocess
im(im < alpha) = alpha;
im(im > 1 - alpha) = 1 - alpha;

% Perform wavelet transform
[ll lh hl hh] = dwt2(im, mode);

% For debugging, take a copy of the transform before stego
im_wavelet_original = [ll, lh; hl, hh];

secret_msg_pos = 1;
[w h] = size(hh);
for x = 1:w
    for y = 1:h
        % Choose bit to encode
        if secret_msg_pos <= secret_msg_bin_len
```



```

        bit = secret_msg_bin(secret_msg_pos);
        secret_msg_pos = secret_msg_pos + 1;
    else
        bit = 0;
    end;

    % Encode the bit
    if bit == 1
        hh(x,y) = hh(x,y) + alpha;
    else
        hh(x,y) = hh(x,y) - alpha;
    end;
end;
end;

% For debug, take a copy of the transform after stego
im_wavelet_stego = [ll, lh; hl, hh];

% Inverse wavelet transform
im_stego = idwt2(ll, lh, hl, hh, mode);

% Denormalise
im_stego = uint8(im_stego * uint8_max);

end

```

5.6 (b) Fusion Decode


```

function [extracted_msg_bin] = steg_fusion_decode(im_stego, im_original, mode)
% steg_fusion_decode Performs steganography using fusion through wavelets
% See paper: "Data Hiding techniques Based On Wavelet-like transform and Complex
Wavelet Transforms"
% INPUTS
% im_stego - Stego image to extract message from.
% im_original - Original image used before steganography.
% mode - Wavelet mode, e.g. 'haar'.
% OUTPUTS
% extracted_msg_bin - The extracted message in binary.

[lls lhs hls hhs] = dwt2(im_stego, mode);
[llo lho hlo hho] = dwt2(im_original, mode);

hh = hhs - hho;

[w h] = size(hh);

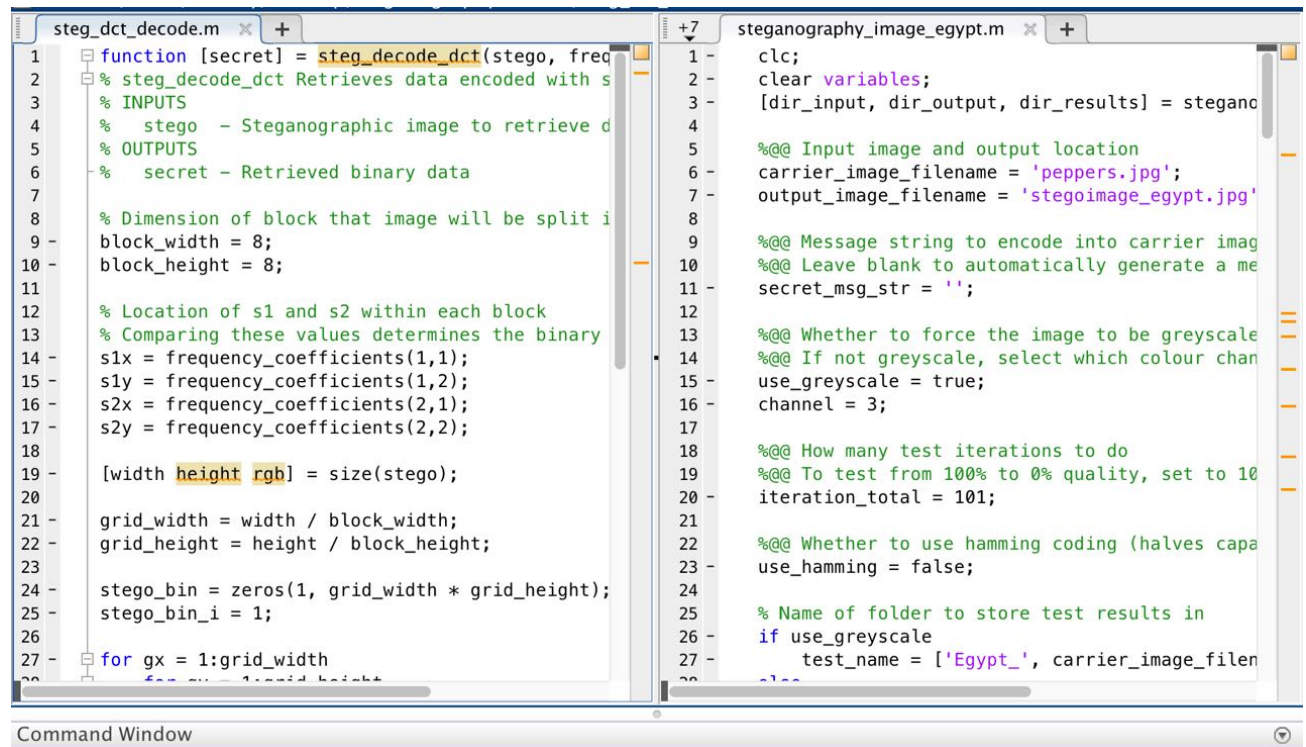
extracted_msg_bin = zeros(1, w * h);

extracted_msg_pos = 1;
for x = 1:w
    for y = 1:h
        if hh(x,y) > 0
            extracted_msg_bin(extracted_msg_pos) = 1;
        else
            extracted_msg_bin(extracted_msg_pos) = 0;
        end;
        extracted_msg_pos = extracted_msg_pos + 1;
    end;
end;

end

```

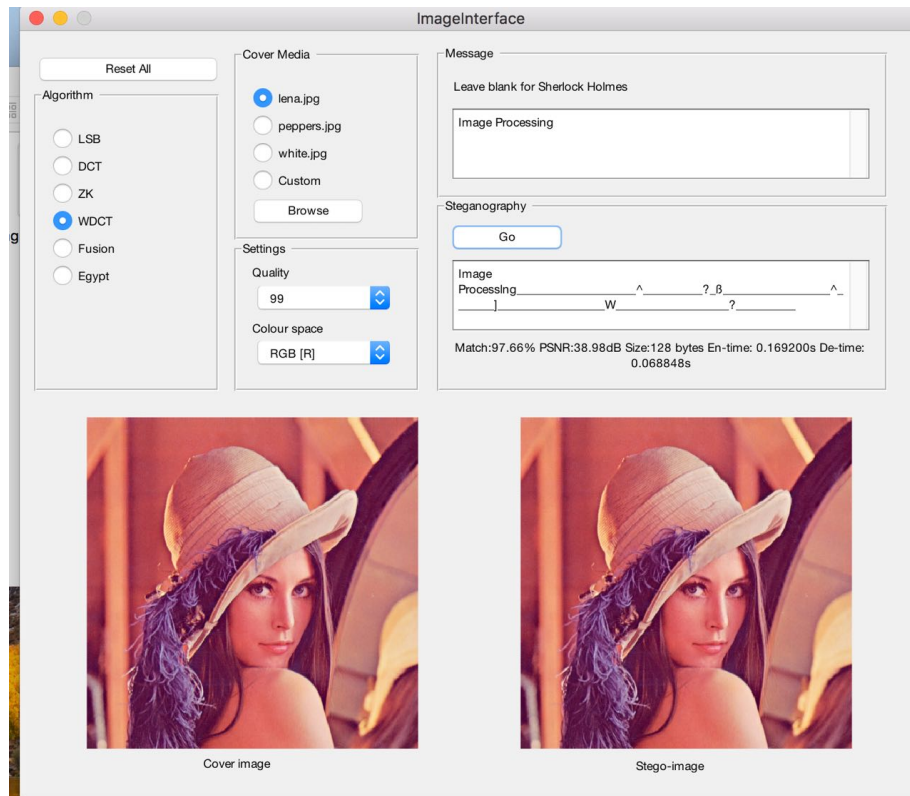
6. IMPLEMENTATION



The image shows a MATLAB code editor with two scripts open. The left script, `steg_dct_decode.m`, defines a function `steg_decode_dct` that takes a stego image and frequency coefficients as input and returns the decoded secret data. The right script, `steganography_image_egypt.m`, is a main script that sets up the environment, defines input/output file names, message string, and test parameters, and then calls the `steg_decode_dct` function.

```
1 function [secret] = steg_decode_dct(stego, freq)
2 % steg_decode_dct Retrieves data encoded with steganography
3 % INPUTS
4 %   stego - Steganographic image to retrieve data from
5 % OUTPUTS
6 %   secret - Retrieved binary data
7
8 % Dimension of block that image will be split into
9 block_width = 8;
10 block_height = 8;
11
12 % Location of s1 and s2 within each block
13 % Comparing these values determines the binary value
14 s1x = frequency_coefficients(1,1);
15 s1y = frequency_coefficients(1,2);
16 s2x = frequency_coefficients(2,1);
17 s2y = frequency_coefficients(2,2);
18
19 [width height rgb] = size(stego);
20
21 grid_width = width / block_width;
22 grid_height = height / block_height;
23
24 stego_bin = zeros(1, grid_width * grid_height);
25 stego_bin_i = 1;
26
27 for gx = 1:grid_width
28     for gy = 1:grid_height
```

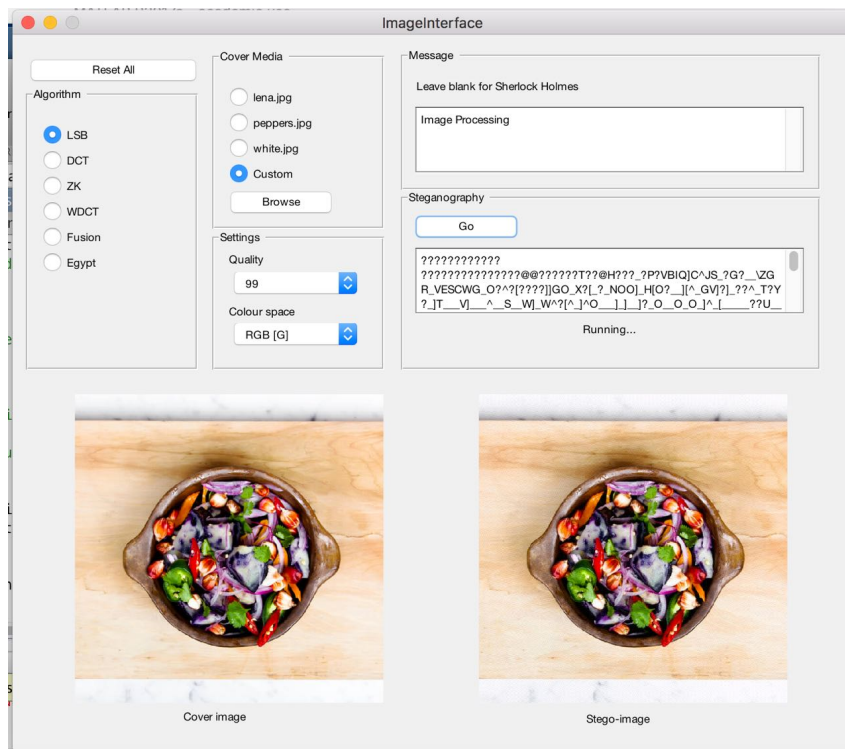
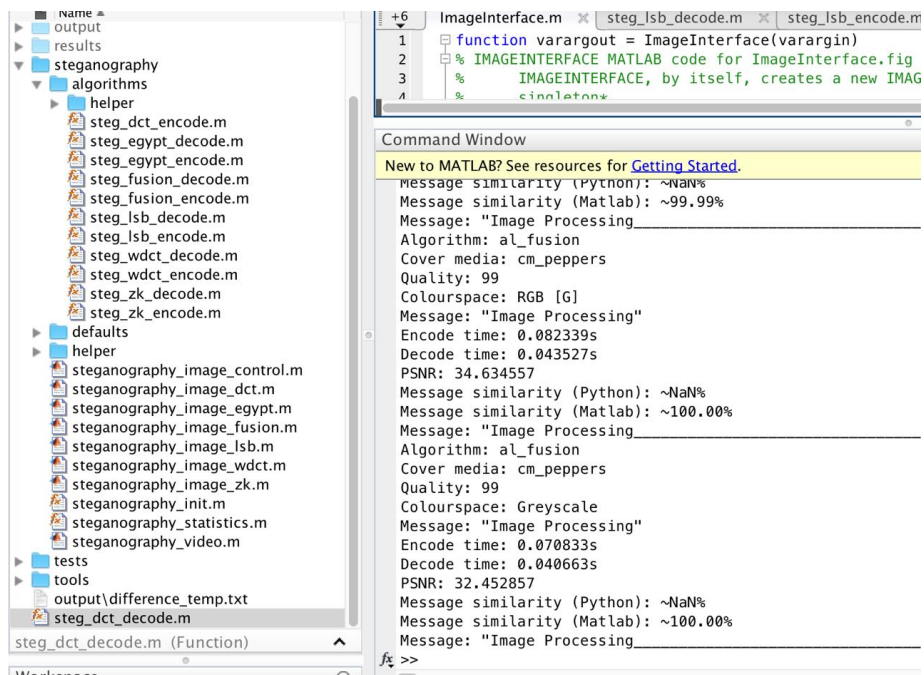
```
1 clc;
2 clear variables;
3 [dir_input, dir_output, dir_results] = steganography_image_egypt;
4
5 %@@ Input image and output location
6 carrier_image_filename = 'peppers.jpg';
7 output_image_filename = 'stegoimage_egypt.jpg';
8
9 %@@ Message string to encode into carrier image
10 %@@ Leave blank to automatically generate a message
11 secret_msg_str = '';
12
13 %@@ Whether to force the image to be greyscale
14 %@@ If not greyscale, select which colour channel to use
15 use_greyscale = true;
16 channel = 3;
17
18 %@@ How many test iterations to do
19 %@@ To test from 100% to 0% quality, set to 100
20 iteration_total = 101;
21
22 %@@ Whether to use hamming coding (halves capacity)
23 use_hamming = false;
24
25 % Name of folder to store test results in
26 if use_greyscale
27     test_name = ['Egypt_', carrier_image_filename];
28 else
```

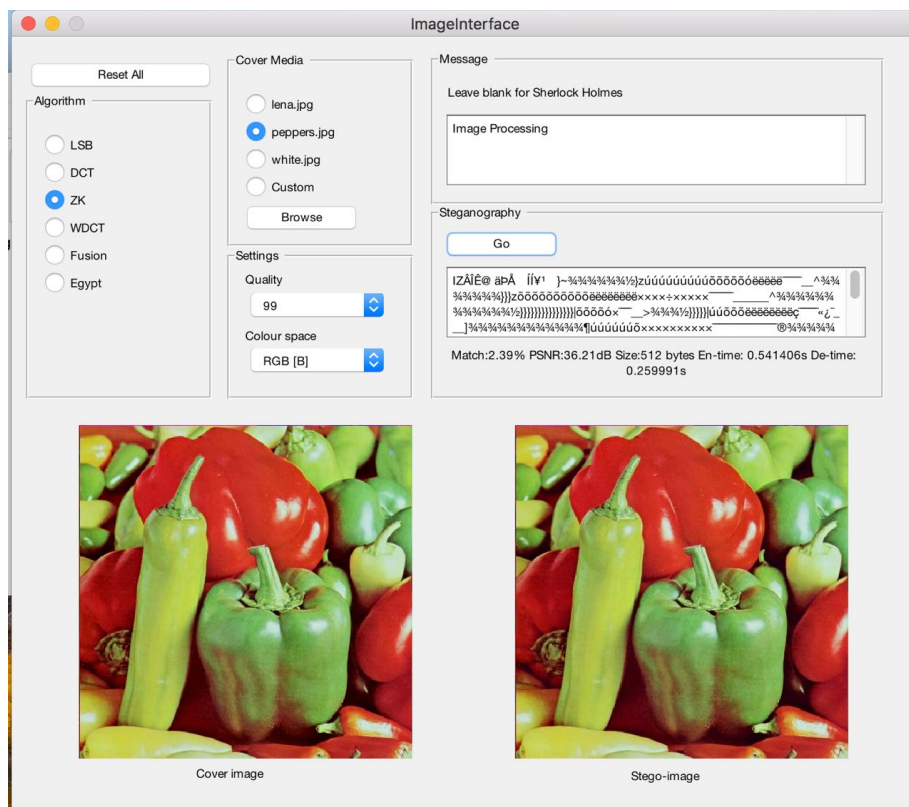
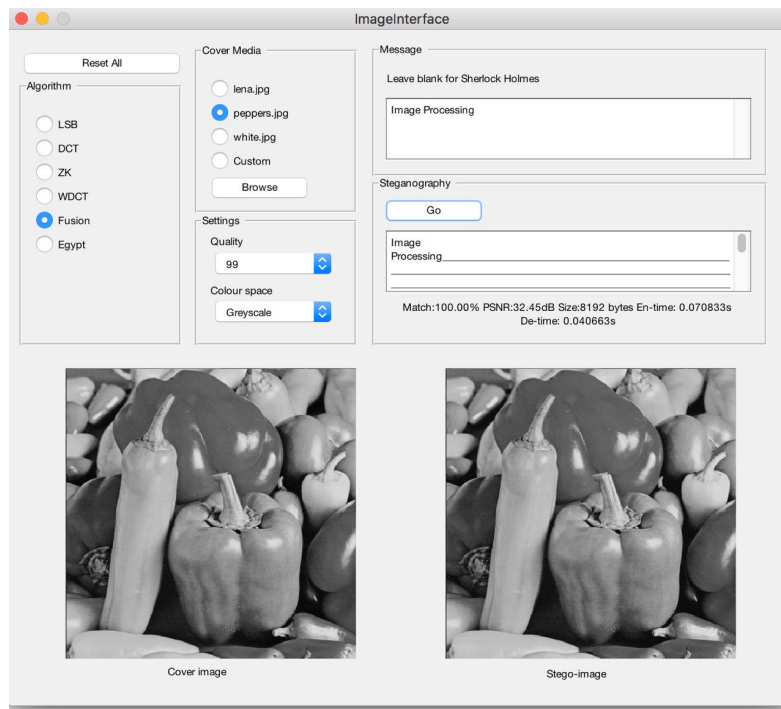


```

steg_lsb_encode.m
1 function steg = steg_lsb_encode(carrier, message_bin)
2 % steg_lsb_encode Encodes a message in the least significant bits of a
3 % carrier image
4 % INPUTS
5 % carrier - Image to hide message in. Should only have one colour
6 % channel.
7 % message_bin - The message, in the form of binary data.
8 % OUTPUTS
9 % steg - Image with encoded message.
10
11 if (length(message_bin(:)) > length(carrier(:)))
12 error('secret is too large to fit within carrier');
13 else
14
15 steg_lsb_decode.m
16 function message_bin = steg_lsb_decode(steg)
17 % steg_lsb_decode Extracts binary data from least significant bits
18 % INPUTS
19 % steg - Image to extract message from.
20 % OUTPUTS
21 % message_bin - Extracted message.
22
23 message_bin = mod(steg,2);
24 message_bin = reshape(message_bin, [], length(length(message_bin)))';
25
26 end

```







7. CONCLUSION

The enhanced LSB technique described in this project helps to successfully hide the secret data into the cover object without any distortion. MATLAB function is an easy to use, user interface function that guides a user through the process of either encoding & decoding a message into or from the image respectively. Since LSB doesn't contain any information there is no loss of information and secret image recovering back become undistorted.

Although only some of the main image steganographic techniques were discussed in this paper, one can see that there exists a large selection of approaches to hiding information in images. All the major image file formats have different methods of hiding messages, with different strong and weak points respectively. Where one technique lacks in payload capacity, the other lacks in robustness. For example, the patchwork approach has a very high level of robustness against most type of attacks, but can hide only a very small amount of information. Least significant bit (LSB) in both BMP and GIF makes up for this, but both approaches result in suspicious files that increase the probability of detection when in the presence of a warden. Thus, for an agent to decide on which steganographic algorithm to use, he would have to decide on the type of application he want to use the algorithm for and if he is willing to compromise on some features to ensure the security of others.

8. REFERENCES

- [1] Mrs. Kavitha, Kavita Kadam, Ashwini Koshti, Priya Dunghav, “Steganography Using Least Significant Bit Algorithm”, International Journal of Engineering Research and applications, vol.2, issue 3, pp. 338-341 May-June 2012.
- [2] Vijay kumar Sharma, Vishal Shrivastava, “A Steganography algorithm for hiding image in image by improved LSB substitution by minimize technique”, Journal of Theoretical and Applied Information Technology, Vol. 36 No.1, 15th February 2012.
- [3] T.Morgan, J.H.P Eloff, “AN OVERVIEW OF IMAGE STEGANOGRAPHY” Information and Computer Security Architecture (ICSA) Research Group, Department of Computer Science University of Pretoria, 0002, Pretoria, South Africa
- [4] <https://ctfs.github.io/resources/topics/steganography/invisible-text/README.html>
- [5] Alaar. A, Shahrin Bin Sahib, Mazdak Zamani, “An Introduction to Image Steganography Techniques, Advanced Computer Science Applications and Technologies (ACSAT), 2012.