ABSTRACT

Due to the significant advancement of the Internet of Things (IOT) in the health-care sector, the security, and the integrity of the medical data became big challenges for health-care services applications. This paper proposes a hybrid security model for securing the diagnostic text data in medical images. The proposed model is developed through integrating either 2-D discrete wavelet transform 1 level(2D-DWT-1L), 2-D discrete wavelet transform 2 level (2D-DWT-2L) and 2-D discrete wavelet transform 3 level(2D-DWT-3L) steganography techniques with a proposed hybrid encryption scheme. The proposed hybrid encryption schema is built using a combination of Advanced Encryption Standard, and Rivest, Shamir, and Adleman algorithms. The proposed model starts by encrypting the secret data; then it hides the result in a cover image using 2D-DWT-1L, 2D-DWT-2L, 2D-DWT-3L. Both colour and grey-scale images are used as cover images to conceal different text sizes. Compared with the state-of-the-art methods, the proposed model proved its ability to hide the confidential patient data into a transmitted cover image with high imperceptibility, capacity, and minimal deterioration in the received stego-image.

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.No.	Abbreviations	Expansion
1	DWT	Discrete Wavelet Transform
2	HVS	Human Visual System
3	IOT	Internet Of Things
4	ASCII	American Standard Code for Information Interchange
5	2D	Two Dimensional
6	PSNR	Peak Signal-To-Noise Ratio
7	SSIM	Structural Similarity Index Measure
8	MSE	Mean Square Error

CHAPTER 1
INTRODUCTION

1. INTRODUCTION

IOT creates an integrated communication environment of interconnected devices and platforms by engaging both virtual and physical world together [1]. With the advent of remote digital healthcare based IOT systems, the transmission of medical data becomes a daily routine. Therefore, it is necessary to develop an efficient model to ensure the security and integrity of the patient's diagnostic data transmitted and received from IOT environment. This goal is carried out using steganography techniques and system encryption algorithms together to hide digital information in an image.

Cryptography is another term for data encryption. Encryption cryptography is the process of encoding messages in a way that hackers cannot read it, but that can be authorized personnel. The two main algorithms used for data encryption in this work are the Advanced Encryption Standard (AES) and the Rivest-Shamir-Adleman (RSA) algorithm. AES is a symmetric cipher where the same key is used on both sides. It has a fixed message block size of 128 bits of text (plain or cipher), and keys of length 128,192, or 256 bits. When longer messages are sent, they are divided into 128-bit blocks. Apparently, longer keys make the cipher more difficult to break, but also enforce a longer encrypt and decrypt process. On the contrary, the RSA is a public key algorithm, which widely used in business and personal communication sectors. It has the advantage of having a variable key size ranging from (2-2048) bits.

The primary research in hiding data started with steganography, which refers to the science and art of hiding information within an image. The benefit of steganography is that it can be utilized to transmit classified messages without the fact of the transmission being detected. The DWT has a tremendous spatial localization, frequency spread, and multi-resolution characteristics, which are matching with the theory of forms in the human visual system. This paper implements both 1-level and 2-level of DWT steganography techniques that operate on the frequency domain. It split up the image into high and low iteration parts. The high iteration part contains edge information, whereas the low iteration part is frequently divided into high and low iteration parts.

The purpose of the steganography is not only preventing others from knowing the hidden information, but also removing the suspicion in having hidden information. The message is a confidential document to be transmitted and camouflaged in the carrier so that it becomes difficult to detect. There are two main aspects in any steganography system, which are steganography capacity and imperceptibility. However, these two properties are confusing with each other because it is tough to increase capacity while maintaining the steganography imperceptibility of a steganography system. Furthermore, there are still limited methods of concealing information for use with data transfer communication protocols, which can be unconventional but their future is promising.

CHAPTER-2

SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

2.1 FEASIBILITY STUDY

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key consideration involved in the feasibility analysis are

- ❖ TECHNICAL FEASIBILITY
- ❖ SOCIAL FEASIBILITY
- ❖ OPERATIONAL FEASIBILITY

2.1.1 TECHNICAL FEASIBILITY

The system is developed using MATLAB which is high-performance language for technical computing integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Using MATLAB, you can solve technical computing problems faster than with traditional programming languages, such as C, C++, and Fortran.

2.1.2 SOCIAL FEASIBILITY

The aspect of study is to be checking the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

2.1.3 OPERATIONAL FEASIBILITY

Operational Feasibility deals with the study of prospects of the system to be developed. This system operationally eliminates all the tensions of the Admin and helps him in effectively tracking the project progress. This kind of automation will surely reduce the time and energy, which previously consumed in manual work. Based on the study, the system is proved to be operationally feasible.

2.2 EXISITING SYSTEM:

With the advent of remote digital healthcare based IOT systems, the transmission of medical data becomes a daily routine.

- Bairagi, A. K., et al, proposed three color image steganography approaches for protecting information in an IOT infrastructure. The first and third approaches use three (red, green, and blue) channels, while the second approach uses two (green and blue) channels for carrying information. Dynamic positioning techniques have been used for hiding information in the deeper layer of the image channels with the help a shared secret key.
- Anwar, A. S., et al, developed a technique to secure any type of images especially medical images. They aimed to maintain the integrity of electronic medical information, ensuring availability of that information, and authentication of that information to ensure that authorized people only can access the information. First, the AES encryption technique was applied on the first part. The ear print also embedded in this work, where seven values were extracted as feature vector from the ear image. The proposed technique improved the security of medical images through sending them via the internet and secured these images from being accessed via any unauthorized person.
- Abdel-Nabi, H., et al, proposed a crypto-watermarking approach based on AES standard encryption algorithm and reversible watermarking data hiding technique to secure medical images. The results proved that the proposed approach achieves both the authenticity and integrity of the images either in the spatial domain or the encrypted domain or both domains.

2.3 PROPOSED SYSTEM

The proposed model composes of four continuous processes:

1. The confidential patient's data is encrypted using a proposed hybrid encryption scheme that is developed from both AES and RSA encryption algorithms.
2. The encrypted data is being concealed in a cover image using 2D-DWT-2L and produces a stego-image.
3. The embedded data is extracted.
4. The extracted data is decrypted to retrieve the original data.

Encryption cryptography is the process of encoding messages in a way that hackers cannot read it, but that can be authorized personnel. The two main algorithms used for data encryption in this work are the Advanced Encryption Standard (AES) and the Rivest-Shamir-Adleman (RSA) algorithm. AES is a symmetric cipher where the same key is used on both sides. It has a fixed message block size of 128 bits of text (plain or cipher), and keys of length 128,192, or 256 bits. When longer messages are sent, they are divided into 128-bit blocks. Apparently, longer keys make the cipher more difficult to break, but also enforce a longer encrypt and decrypt process. On the contrary, the RSA is a public key algorithm, which widely used in business and personal communication sectors. It has the advantage of having a variable key size ranging from (2-2048) bits.

2.4 SYSTEM DESIGN

2.4.1 INPUT DESIGN

Input Design plays a vital role in the life cycle of software development, it requires very careful attention of developers. The input design is to feed data to the application as accurate as possible. So inputs are supposed to be designed effectively so that the errors occurring while feeding are minimized. According to Software Engineering Concepts, the input forms or screens are designed to provide to have a validation control over the input limit, range and other related validations.

This system has input screen only at the start of the program. All the remaining inputs are none other than the outputs of the previously executed module in the program. Error messages are developed to alert the user whenever he commits some mistakes and guides him in the right way so that invalid entries are not made. Let us see deeply about this under module design.

Input design is the process of converting the user created input into a computer-based format. The goal of the input design is to make the data entry logical and free from errors. The error is in the input are controlled by the input design. The application has been developed in user-friendly manner. The input panel have been designed in such a way that its default location is same as that of application but also can be navigated to the user desired location. The user is also provided with in an option to select an appropriate input from various alternatives related to the field in certain cases.

Validations are required for each data entered. Whenever a user enters an erroneous data, error message is displayed and the user can select the input that matches the specified requirement.

2.4.2 OUTPUT DESIGN

The output from the computer is required to mainly create an efficient model to ensure the security and integrity of the patient's diagnostic data transmitted and received from IOT environment, in other words, between the Sender and the Receiver. This system consists of several outputs that are generated at the end of each modules which the program is comprised of. These outputs are generally considered as the intermediary outputs which serve as input for the succeeding module. The final output decides whether the message has maintained its integrity throughout its journey starting from the sender until it reached the receiver.

2.5. SYSTEM BLOCK DIAGRAM

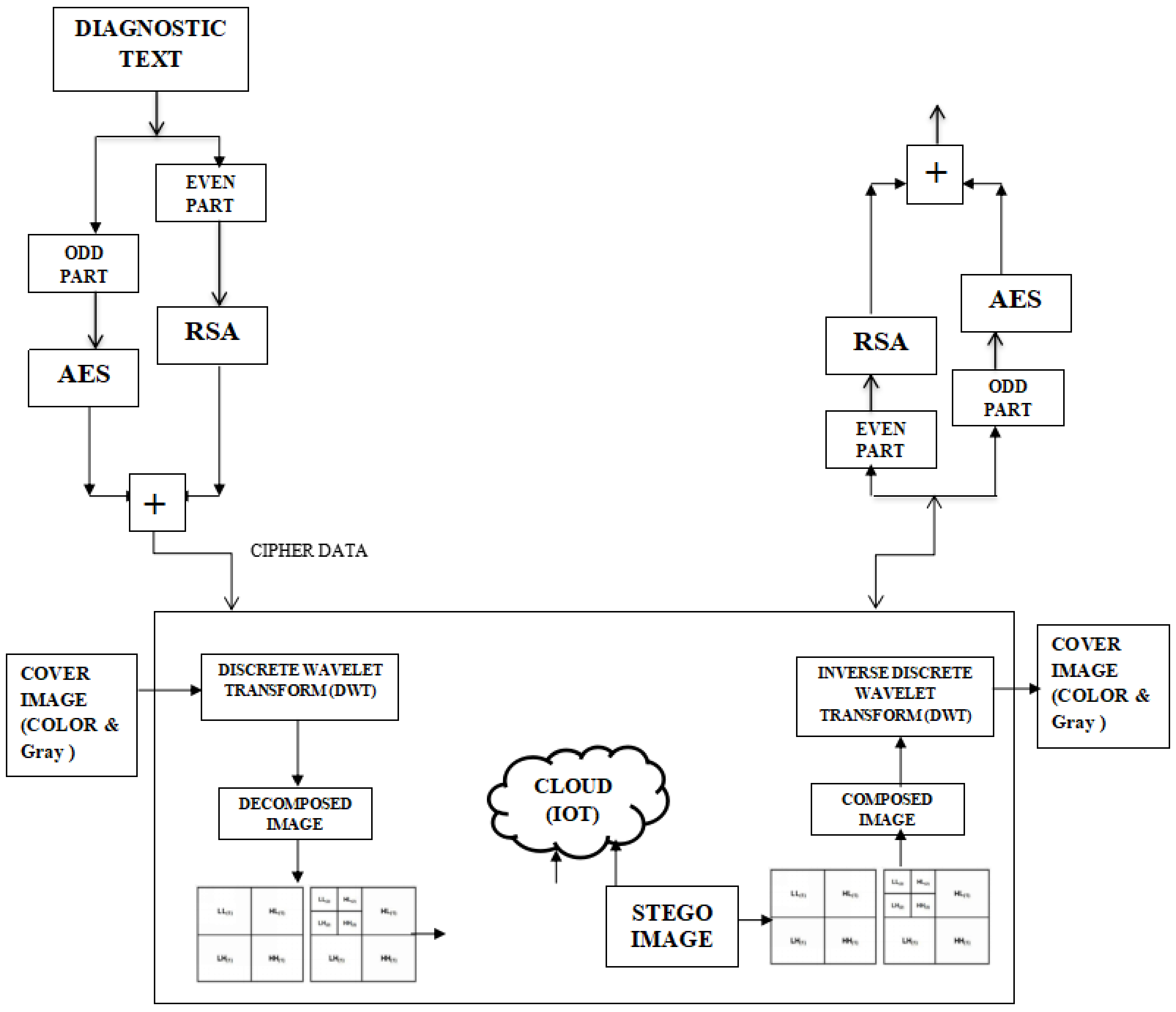

DIAGNOSTIC TEXT
EVEN PART
ODD PART
RSA
AES
+
CIPHER DATA
COVER IMAGE (COLOR & Gray)
DISCRETE WAVELET TRANSFORM (DWT)
DECOMPOSED IMAGE
CLOUD (IOT)
STEGO IMAGE
INVERSE DISCRETE WAVELET TRANSFORM (DWT)
COMPOSED IMAGE
RSA
AES
EVEN PART
ODD PART
+
COVER IMAGE (COLOR & Gray)

Figure 1: System Block Diagram

CHAPTER-3

SOFTWARE REQUIREMENT SPECIFICATIONS

3. SOFTWARE REQUIREMENT SPECIFICATIONS

3.1. FUNCTIONAL REQUIREMENTS

The functional requirements are the critical requirements which should be within the system in order to overcome or at least minimize the drawbacks of the existing system. Functional requirements comprise of what are and what type of data should be given as input to the system, what processing should be done to the inputs in order to get the required working, functionalities and output as expected by the user and in what ways or how the output must be presented to the user. In order to make this application functional, we require the following:

3.1.1. Hybrid (AES & RSA) Algorithm

Input: The sender need to provide secret plain text message.

Processing: The plain text T is divided into odd part T-ODD and even parts T-EVEN. The AES is used to encrypt T-ODD using a secret public key s. The RSA is used to encrypt T-EVEN using a secret public key m.

Output: The output include Cipher message and key s.

3.1.2. Embedding 2D-DWT-3L Algorithm

Input: The user need to provide an image which can be either colour or grey scale image.

Processing: The processing is done in two steps

1. 2D-DWT-3L can be formulated as a consecutive transformation using low-pass and high-pass filters.
2. Read the value of the pixel, convert it to its equivalent binary form and modify the least significant bit accordingly.

Output: The output is a Steganography image.

3.2. NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements are those that do not directly contribute to the system functionality, modules or processing but are major factors contributing to the quality of the system.

3.2.1. EFFICIENCY

Efficiency is concerned with the system resources used when providing the required functionality. The amount of disk space, memory, network etc. provides a good indication of this characteristic. Every successive module is executed only after completion of preceding module given that the inputs by user are valid, otherwise rejects it. Thus the system provides efficiency in providing results.

3.2.2. PORTABILITY

Portability specifies the ease with which the software can be installed on all necessary platforms, and the platforms on which it is expected to run. We use MATLAB for developing our project. The MATLAB file can only be run on another platform if it either has MATLAB or the MATLAB Component Runtime (MCR). The MCR is free to download.

3.2.3. RELIABILITY

Reliability characteristic defines the capability of the system to maintain its service provision under defined conditions for defined periods of time. One aspect of this characteristic is fault tolerance that is the ability of a system to withstand component failure. All the details provided by user are recorded. Hence the system is extremely reliable.

3.2.4. AVAILABILITY

Availability is the ratio of time a system or component is functional to the total time it is required or expected to function. The entire project is situated on personal computers of both sender and receiver and the data can be transferred from sender to receiver using any peripheral devices or through a communication channel. Thus can be accessed by senders and receivers at any time and place.

3.3. ENVIRONMENT SPECIFICATION

3.3.1. SOFTWARE REQUIREMENTS

Operating system	:	Windows 7/8/10
Coding Language	:	MATLAB (99.4%), M (0.6%)
Software used	:	MATLAB
Software version	:	R2018a

3.3.2. HARDWARE REQUIREMENTS

CPU	:	I3 or later
RAM	:	4GB or above
INPUT DEVICE	:	Local Text files (.txt) and Image files (.jpg)
OUTPUT DEVICE	:	Monitor

CHAPTER-4

SYSTEM DEVELOPMENT ENVIRNOMENT

4. SYSTEM DEVELOPMENT ENVIRNOMENT

4.1. INTRODUCTION TO MATLAB

The name MATLAB stands for Matrix Laboratory. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects.

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. These factors make MATLAB an excellent tool for teaching and research.

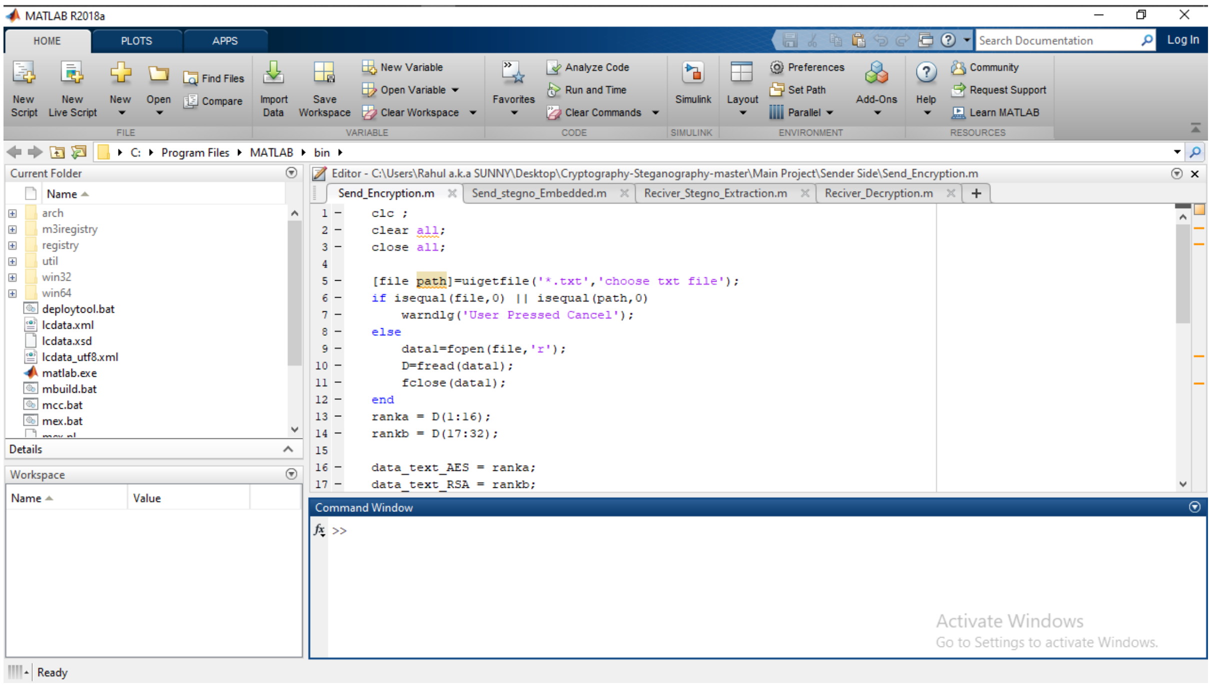
MATLAB has many advantages compared to conventional computer languages (e.g., C, FORTRAN) for solving technical problems. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide.

It has powerful built-in routines that enable a very wide variety of computations. It also has easy to use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as toolbox. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.

4.2. STARTING MATLAB

After logging into your account, you can enter MATLAB by double-clicking on the MATLAB shortcut icon on your Windows desktop. When you start MATLAB, a special window called the MATLAB desktop appears. The desktop is a window that contains other windows. The major tools within or accessible from the desktop are:

- The Command Window
- The Command History
- The Workspace
- The Current Directory
- The Help Browser
- The Start button



```
clc ;
clear all;
close all;

[file path]=uigetfile('*.txt','choose txt file');
if isequal(file,0) || isequal(path,0)
    warndlg('User Pressed Cancel');
else
    data1=fopen(file,'r');
    D=fread(data1);
    fclose(data1);
end
ranka = D(1:16);
rankb = D(17:32);

data_text_AES = ranka;
data_text_RSA = rankb;
```

```
>>
```

Figure 2: The graphical interface to the MATLAB workspace

4.3. QUITTING MATLAB

To end your MATLAB session, type quit in the Command Window, or select File → Exit MATLAB in the desktop main menu.

4.4. INTRODUCTION TO PROGRAMMING IN MATLAB

4.4.1. M-FILE SCRIPTS

A script file is an external file that contains a sequence of MATLAB statements. Script files have a filename extension .m and are often called M-files. M-files can be scripts that simply execute a series of MATLAB statements, or they can be functions that can accept arguments and can produce one or more outputs.

4.4.2. SCRIPT SIDE-EFFECTS

All variables created in a script file are added to the workspace. This may have undesirable effects, because:

- Variables already existing in the workspace may be overwritten.
- The execution of the script can be affected by the state variables in the workspace.

As a result, because scripts have some undesirable side-effects, it is better to code any complicated applications using rather function M-file.

4.4.3. M-FILE FUNCTIONS

As mentioned earlier, functions are programs (or routines) that accept input arguments and return output arguments. Each M-file function (or function or M-file for short) has its own area of workspace, separated from the MATLAB base workspace.

This simple function shows the basic parts of an M-file.

```
function f = factorial (n)                              (1)
% FACTORIAL (N) returns the factorial of N.             (2)
% Compute a factorial value.                            (3)
f = prod (1: n);                                        (4)
```

Table: Anatomy of an M-File function

Part no.	**M-file element**	**Description**
(1)	Function definition line	Define the function name, and the number and order of input and output arguments
(2)	H1 line	A one line summary description of the program, displayed when you request ‘Help’
(3)	Help text	A more detailed description of the program
(4)	Function body	Program code that performs the actual computations

Both functions and scripts can have all of these parts, except for the function definition line which applies to function only. In addition, it is important to note that function name must begin with a letter, and must be no longer than the maximum of 63 characters. Furthermore, the name of the text file that you save will consist of the function name with the extension .m. Thus, the above example file would be ‘factorial.m’.

4.5. INPUT AND OUTPUT ARGUMENTS

The input arguments are listed inside parentheses following the function name. The output arguments are listed inside the brackets on the left side. They are used to transfer the output from the function file. The general form looks like this:

```
function [outputs] = function_name (inputs)
```

Function file can have none, one, or several output arguments.

function C=FtoC(F)	One input argument and one output argument
function area=TrapArea(a,b,h)	Three inputs and one output
function [h,d]=motion(v,angle)	Two inputs and two outputs

4.5.1. INPUT TO A SCRIPT FILE

When a script file is executed, the variables that are used in the calculations within the file must have assigned values. The assignment of a value to a variable can be done in three ways:

- The variable is defined in the script file.
- The variable is defined in the command prompt.
- The variable is entered when the script is executed.

We have already seen the two first cases. Here, we will focus our attention on the third one. In this case, the variable is defined in the script file. When the file is executed, the user is prompted to assign a value to the variable in the command prompt. This is done by using the input command. Here is an example:

```
% this script file calculates the average of points
% scored in three games.
% the point from each game are assigned to a variable
% by using the ‘input’ command.
game1 = input (’Enter the points scored in the first game ’);
game2 = input (’Enter the points scored in the second game ’);
game3 = input (’Enter the points scored in the third game ’);
average = (game1+game2+game3)/3
```

The following shows the command prompt when this script file (saved as example3) is executed.

```
>> example3
>> Enter the points scored in the first game 15
>> Enter the points scored in the second game 23
>> Enter the points scored in the third game 10
average = 16
```

4.5.2. OUTPUT COMMANDS

As discussed before, MATLAB automatically generates a display when commands are executed. In addition to this automatic display, MATLAB has several commands that can be used to generate displays or outputs.

Two commands that are frequently used to generate output are: disp and fprintf. The main differences between these two commands can be summarized as follows:

Disp

- Simple to use
- Provide limited control over the appearance of output

Fprintf

- Slightly more complicated than disp.
- Provide total control over the appearance of output

4.6. INTRODUCTION TO DISCRETE WAVELET TRANSFORM (DWT)

DWT is a mathematical tool for pyramidal image decomposition. The transformation depends on a signal analysis into small waves or waves, of varying frequency and limited duration. Wavelet properties. The original reference to wavelet transformation parameters that contain location information is degraded. The original signal can be completely reconstructed by performing a reverse wavelet transformation on these coefficients.

DWT analyses a picture into sub-images or sub-ranges, three points of interest and one zoom. The groups are LL, LH, HL, and HH. LL has more frequencies within the level and vertical course. HH has tall frequencies in both the horizontal and vertical course. HL has tall frequencies within the level direction and more frequencies within the vertical course. LH has more frequencies within the even heading and tall frequencies within the vertical course. The low-frequency portion is made up of the unpleasant data of the signal whereas the high-frequency portion is made up of data approximately the edge components. The LL tape is the foremost vital tape since it contains most of the picture vitality and speaks to the picture guess. Watermarks can be included in high-frequency detail groups (LH, HL, and HH) since these ranges are less delicate to human vision. Counting these bars increments the strength of the watermark without any extra effect on the picture quality. At each level of decay, the primary DWT is executed within the vertical course, taken after by the DWT within the level course. The primary level of deterioration gives four sub-bands: LL1, LH1, HL1, and HH1. The LL sub-band from the past level is utilized as input for each sequential level of decay. This LL sub-band is partitioned into four multi resolution sub-bands for the taking after coarse wavelength coefficients. This handle is rehashed a few times depending on the application for which it is utilized.

The reason why a separate wavelet transformation is better than a transformation of Fourier transform is because DWT has a better ability to localize time and frequency. This makes image compression easier to hand.

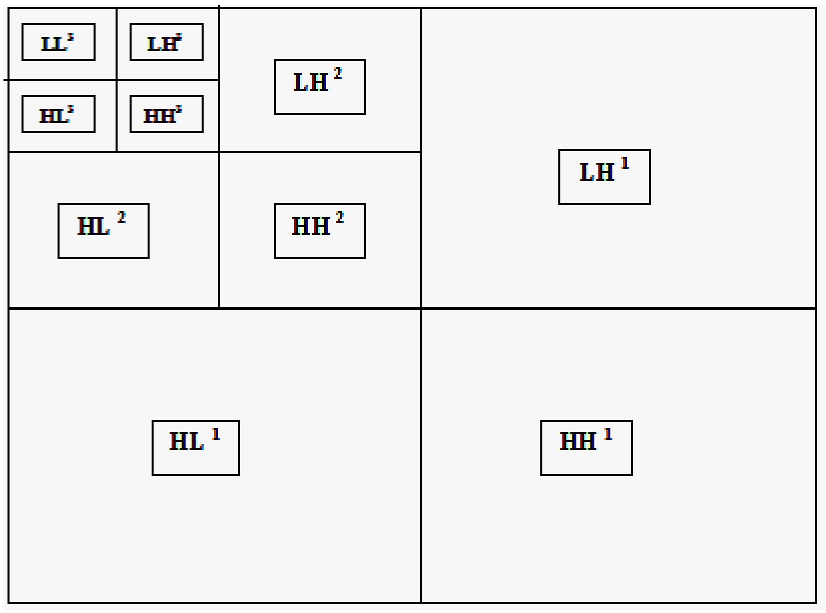


Figure 3: DWT decomposition of an image

4.6.1. CHARACTERISTICS OF DWT

The wavelet change breaks down the picture into three spatial bearings, i.e. flat, vertical and corner to corner. Consequently, wavelets reflect the anisotropic properties of HVS more accurately.

- Wavelet Change is computationally effective and can be actualized by utilizing basic channel convolution.
- With multi-resolution investigation, a picture can be spoken to at more than one determination level.
- Size of DWT coefficients is bigger within the most reduced groups (LL) at each level of decay and is littler for other groups (HH, LH, and HL).
- The bigger the greatness of wavelet coefficient, the more critical it is.
- Watermark discovery at lower resolutions is computationally successful since at each progressive determination level, less no. of recurrence groups is included.
- Tall determination subgroups offer assistance to effortlessly find edge and surfaces designs in an image

4.6.2. IMAGE WATERMARKING

Watermarking risen within the mid-90s as a tech, among the wide extent of the multidisciplinary field of information covering up, as a technique of securing computerized Image from any robbery act. It comprises of implanting a watermark (a follow) inside an advanced picture sometime recently utilizing or distributing it. The effectiveness of a watermarking strategy lies for the most part in its capacity to satisfy three prerequisites: strength, security and intangibility.

The generalized watermark demonstrate comprises of two forms: the consideration and discovery of the watermark. In the implanting handle, the watermark may be encoded within the cover picture employing a particular key. This key is utilized to encode the watermark as an extra level of assurance. The yields of the implanting handle, the watermarked picture, are at that point exchanged to the recipient. In the discovery prepare, moreover called the extraction prepare, the watermark is extricated from the attacked signal. Amid exchange in the event that the signal isn't adjusted, the watermark is still shown and can be extracted.

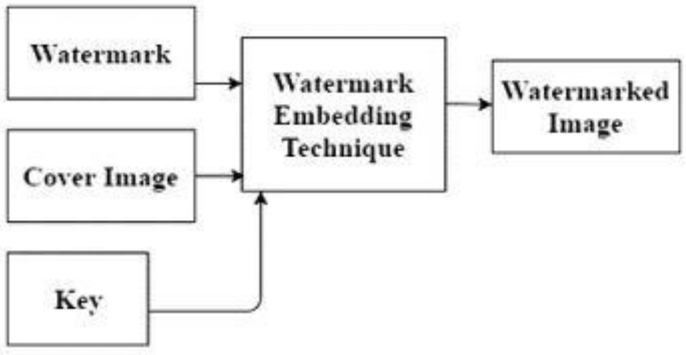


Figure 4: Watermark embedding

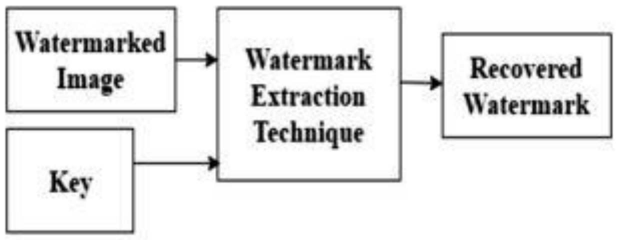


Figure 5: Recovered watermark

4.6.3. DWT FIGURES

The following are the DWT figures obtained in our program for the given input figure say for example

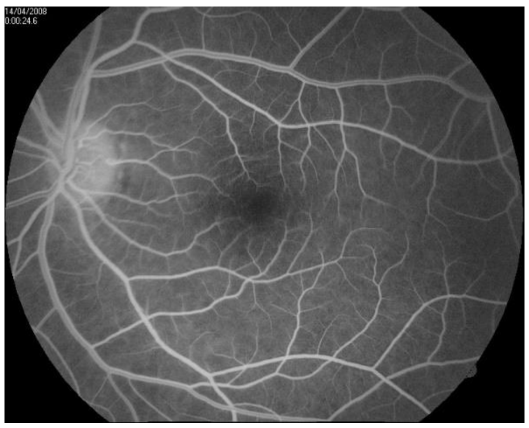


Figure 6: Sample input image

are as follows:

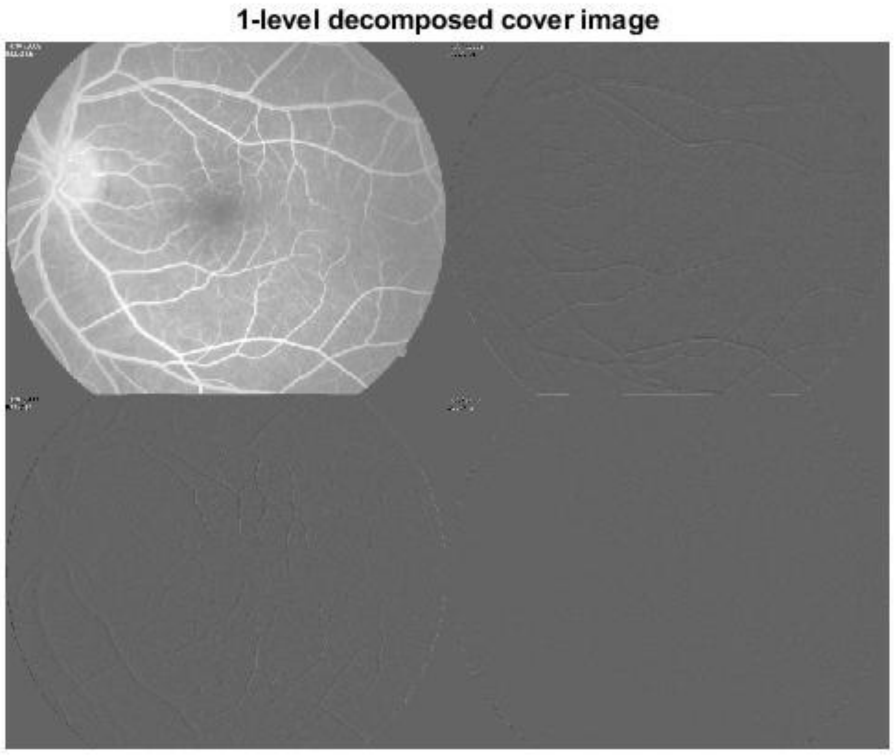


Figure 7: Level 1 Decomposition of image

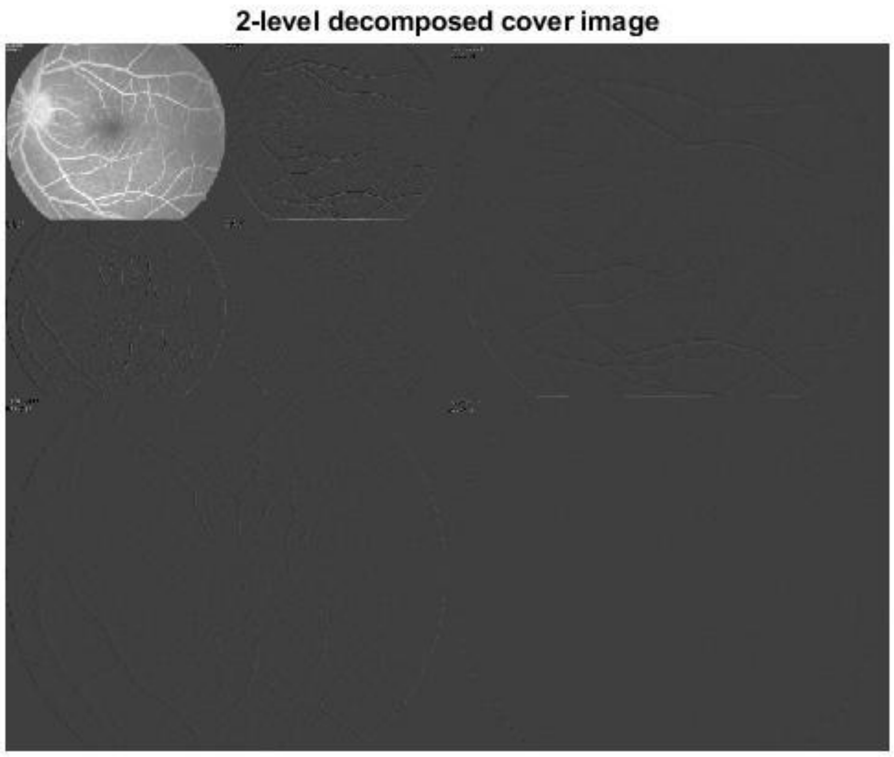


Figure 8: Level 2 Decomposition of image

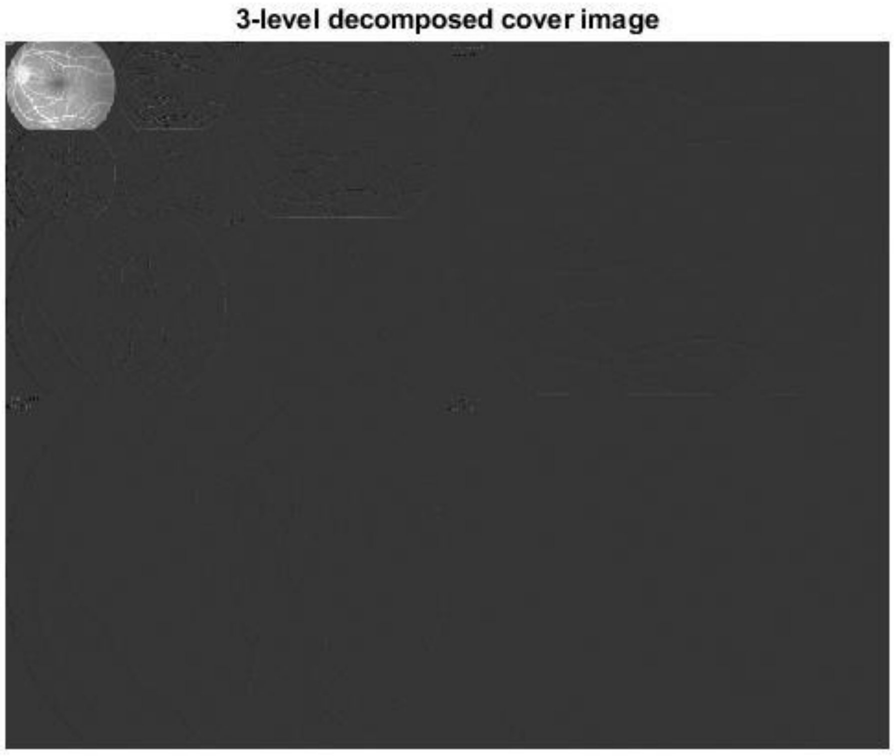


Figure 9: Level 3 Decomposition of image

CHAPTER-5
SOFTWARE DESIGN

5. SOFTWARE DESIGN

The proposed model composes of four continuous processes: (1) The confidential patient's data is encrypted using a proposed hybrid encryption scheme that is developed from both AES and RSA encryption algorithms. (2) The encrypted data is being concealed in a cover image using 2D-DWT-1L, 2D-DWT-2L, 2D-DWT-3L and produces a stego-image. (3) The embedded data is extracted. (4) The extracted data is decrypted to retrieve the original data.

5.1. DATA ENCRYPTION SCHEME

The proposed model implements the cryptographic scheme. The cryptographic scheme $\hat{C}$ = {ﬀŋ, ﬀŋ$^{-1}$, C, S, T} is composed of encryption and decryption processes. Throughout the encryption process, the plain text T is divided into odd part T_{odd} and even parts T_{even}. The AES is used to encrypt T_{odd} using a secret public key s. The RSA is used to encrypt T_{even} using a secret public key m. The private key x that used in the decryption process at the receiver side is encrypted using AES algorithm and sent to the receiver in an encrypted form to increase the security level. The encryption process can be mathematically modelled as given in the following equations below.

$C = \{E_{AES}, E_{RSA}, T_{odd}, T_{even}, \check{T}_{odd}, \check{T}_{even}, s, m, x\}$ (1)

$\check{T}_{odd} = \{E_{AES}(T_{odd}, s)\}$ (2)

$\check{T}_{even} = \{E_{RSA}(T_{even}, m)\}$ (3)

$X = \{E_{AES}(x, s)\}$ (4)

The algorithm used in the encryption procedure is described below.

Algorithm (1): Hybrid (AES & RSA) Algorithm

```
Inputs: secret plain text message

Output: main_cipher message, key s

Begin

    • Divide plain msg into two parts (Odd_Msg, Even_Msg)
    • Generate new AES key s
    • EncOdd = AES-128 (Odd_Msg, s)
    • Generate new RSA key (public = m) and (private = x)
    • EncEven = RSA (Even_Msg, m)
    • Build FullEncTxt by inserting both EncOdd and EncEven in their indices
    • EncKey = AES-128 (x, s)
    • Compress FullEncMsg by convert to hash
    • Compress EncKey by convert to hash
    • Define message empty main_cipher = ""
    • main_cipher = Concatenate (FullEncMsg, EncKey )
    • Return main_cipher and s
End
```

5.2. EMBEDDING PROCEDURE

In this process, A Haar-DWT was implemented. Throughout Haar-DWT, 2D-DWT-2L can be formulated as a consecutive transformation using low-pass and high-pass filters long the rows of the image; then the result decomposed along the columns of the image.

The proposed model implements the steganographic scheme. The steganographic scheme $\hat{S} = \{\text{fŋ}, \text{fŋ}^{-1}, C, S, T\}$ is composed of embedding and extraction processes. The embedding process takes a cover image C and a secret text message T as input and generates a stego-image S. While the extraction process inversely extracts the embedded message. It can be mathematically modelled as given in the following equations below.

$\hat{S} = \{\text{fŋ}, \text{fŋ}^{-1}, C, S, T\}$ (5)

$S = \{\text{fŋ}, C, T\}$ (6)

$T = \{\text{fŋ}^{-1}(S)\}$ (7)

Throughout the embedding process, the secret text is transformed into an ASCII format and then divided into even and odd values. The odd values are concealed in vertical coefficients mentioned by LH3. The even values are concealed in diagonal coefficients specified by HH3. The algorithm that is used in the embedding procedure by evolved 2D-DWT-3L is described below.

Algorithm (2): Embedding 2D-DWT-2L Algorithm

Inputs: cover image, a secret message (main_cipher and s).

Output: stego image.

Begin

```
• Convert the secret message in ASCII Code as asciiMsg
• Divide asciiMsg to odd and even
• Scan the image row by row as img
• Compute the 2D wavelet for the first level by harr filter that generates (LL1), (HL1), (LH1), and (HH1)
• Compute the 2D wavelet for the second level by harr filter that generates (LL2), (HL2), (LH2), and (HH2)
• Compute the 2D wavelet for the third level by harr filter that generates (LL3), (HL3), (LH3), and (HH3)
• Loop
    o Hide odd values in vertical coefficient, set LH3(x, y) = odd values
    o Hide even values in vertical coefficient, set HH3(x, y) = even values
• End Loop
• Return Stego-image
End
```

5.3. EXTRACTION PROCEDURE

After incorporating the text into the cover image, the 2DDWT-3L technique is carried to extract the secret message and retrieve the cover image. The extraction algorithm is described below.

Algorithm (3): Extraction algorithm

```
Inputs: stego image
Output: Retrieved secret message and original cover image
Begin
• Scan the stego image row by row
• Compute the 2D wavelet for the first level by harr filter
• Compute the 2D wavelet for the second level by harr filter
• Compute the 2D wavelet for the third level by harr filter
• Prepare msg = ""
• Loop
```

```
        o Extract the text embedded in vertical coefficient, set odd values =LH3(x, y)
        o Extract the text embedded in vertical coefficient, set even values = HH3(x, y)
    • End Loop
    • msg = Append (odd values, even values)
    • Compute idwt2 for the constructed approximation that generates the original image
    • Return msg as a retrieved secret message and original cover image
End
```

Once the secret text message has been extracted, the cover image is synthesized from the reconstructed approximation by calling the idwt2 for the third level, second level and then for the first level.

5.4. DATA DECRYPTION SCHEME

Decryption refers to the process of converting the encrypted data back to the user in a well-known format; which is the reverse of the encryption process. The same key used by the sender has to be used over the cipher-text throughout the encryption process. The decryption process can be mathematically expressed as given in the following equations below.

$\hat{C} = \{E_{AES}^{-1}, E_{RSA}^{-1}, T_{odd}, T_{even}, \check{T}_{odd}, \check{T}_{even}, s, x\}$ (8)

$x = \{E_{AES}(X, s)\}$ (9)

$T_{even} = \{E_{RSA}^{-1}(\check{T}_{even}, x)\}$ (10)

$T_{odd} = \{E_{AES}^{-1}(\check{T}_{odd}, s)\}$ (11)

The proposed decryption algorithm is described below.

Algorithm (4): Hybrid Decryption (AES & RSA) Algorithm

```
Inputs: main_cipher (secret) message, key

Output: secret (plain, text) message

Begin
    • Divide main_cipher into two parts; HashedTxt and HashedKey
    • FullEncMsg = Decompress (HashedTxt)
    •   EncKey = Decompress (HashedKey)
    •   x = Decrypt_AES-128 (EncKey, s)
    •   EncOdd = Split (FullEncMsg, odd)
    •   EncEven = Split (FullEncMsg, even)
    •   Odd_Msg = Decrypt_ AES-128 (EncOdd, s)
    •   Even_Msg = Decrypt_ RSA (EncEven, x)
    •   Define main_plain message
    •   Loop on All Char
            o   If odd
                    ▪   Insert odd characters into odd indices within main_plain message
            o   Else
                    ▪   Insert even characters into even indices within main_plain message
    •   End of Loop
    •   Return main_plain (text) message

End
```

CHAPTER-6

MODULES

6. MODULES

In this application there are three modules:

- Sender module
- Server/IOT module
- Receiver module

6.1. SENDER MODULE:

- Input plain text message
- AES secret public key generation
- RSA public key and private key generation
- Dividing plain text into odd and even parts
- Odd part encryption using AES public key
- Even part encryption using RSA public key
- Converting cipher message into ASCII code
- Computing 2D wavelet for the third level
- Hiding odd parts in vertical coefficient(LH)
- Hiding even parts in vertical coefficient(HH)
- Sending Stego image to server or an IOT device

6.2. SERVER/IOT MODULE:

- Receiving the Stego image from sender which contains the secret message
- Passing that image without any third party intervention safely to the receiver

6.3. RECEIVER MODULE:

- Computing 2D wavelet for the third level
- Extracting odd parts from vertical coefficient(LH)
- Extracting even parts from vertical coefficient(HH)
- Odd part decryption using AES public key
- Even part decryption using RSA private key
- Inserting odd and even character into their respective places
- Returning the plain text message

CHAPTER-7

CODING

7. CODING

7.1. SEND_ENCRYPTION.m

```
clc;
clear all;
close all;
[file path]=uigetfile('*.txt','choose txt file');
if isequal(file,0) || isequal(path,0)
   warndlg('User Pressed Cancel');
else
   data1=fopen(file,'r');
   D=fread(data1);
   fclose(data1);
end
ranka = D(1:16);
rankb = D(17:32);
data_text_AES = ranka;
data_text_RSA = rankb;
[s_box, w, poly_mat] = aes_init;
plaintext_AES = data_text_AES;
ciphertext_AES = cipher (plaintext_AES, w, s_box, poly_mat, 1);
disp('Implementation of RSA Algorithm');
p = input('\nEnter the value of p: ');
q = input('\nEnter the value of q: ');
[Pk,Phi,d,e] = intialize(p,q);
[N,Phi,d,e] = intialize(p,q);
MAsg = data_text_RSA;
```

```
Length_1=length(MAsg);
for A= 1:Length_1
   cipher_RSA(A)= crypt(MAsg(A),N,e);
end
disp('Cipher_RSA Text of the entered Message:');
disp(cipher_RSA);
Cipher_Test = cat(2,ciphertext_AES, cipher_RSA);
disp('Cipher Text of the Original Message:');
disp(Cipher_Test);
disp('Cipher text generated');
fid_1=fopen('cipher.txt','w');
fprintf(fid_1,'%d ',Cipher_Test);
```

7.2. SEND_STEGNO_EMBEDDED.m

```
clc;
clear all;
close all;
fid_2 = fopen('cipher.txt','rb');
Str = fread(fid_2, [1, inf], 'char');
fclose(fid_2);
Str=(Str);
[filename, pathname] = uigetfile( ...
      {'*.jpg;*.tif;*.tiff;*.png;*.bmp', 'All image Files (*.jpg, *.tif, *.tiff, *.png, *.bmp)'}, ...
       'Pick a file');
fIle_1 = fullfile(pathname, filename);
disp('Reading Cover image');
disp('Cover Medium found');
```

```
IMAge=imread(fIle_1);
figure,imshow(IMAge);
impixelinfo;
title('Input Cover Image');
imwrite(IMAge,'original.jpeg');
[Rows_1 Col_1 Dim]= size(IMAge);
if Dim == 3
    IMAge=rgb2gray(IMAge);
    figure,imshow(IMAge);
    impixelinfo;
    title('Input Gray Image');
end
histogram(IMAge);
[ll1,hl1,lh1,hh1]=dwt2(IMAge,'haar');
DWT_1=[ll1,hl1;lh1,hh1];
figure,imshow(DWT_1,[]);
title('1-level decomposed cover image');
[ll2,hl2,lh2,hh2]=dwt2(ll1,'haar');
b=[ll2,hl2;lh2,hh2];
DWT_2=[b,hl1;lh1,hh1];
figure,imshow(DWT_2,[]);
title('2-level decomposed cover image');
[ll3,hl3,lh3,hh3]=dwt2(ll2,'haar');
c=[ll3,hl3;lh3,hh3];
cc=[c,hl2;lh2,hh2];
DWT_3=[cc,hl1;lh1,hh1];
figure,imshow(DWT_3,[]);
```

```
title('3-level decomposed cover image');
[Rows_2 Col_2]=size(hh3);
hh3_16 = (hh3);
Length_2 = numel(Str);
Length_3 = 1;
for B=1:Rows_2
    for C=1:Col_2
        if(Length_3 <= Length_2)
            EMBED_16(B,C) = Str(Length_3);
        else
            EMBED_16(B,C) = hh3_16(B,C);
        end
        Length_3 =Length_3+1;
    end
end
EMBED = (EMBED_16);
IDWT_1=idwt2(ll3,hl3,lh3,EMBED,'haar');
figure,imshow(IDWT_1,[]);
title('1-level embedded image');
IDWT_2=idwt2(IDWT_1,hl2,lh2,hh2,'haar');
figure,imshow(IDWT_2,[]);
title('2-level embedded image');
IDWT_3=idwt2(IDWT_2,hl1,lh1,hh1,'haar');
figure,imshow(IDWT_3,[]);
title('3-level embedded image');
Embed_IMAge=(IDWT_3);
save('Embed_IMAge.mat','Embed_IMAge')
```

```
Embed_IMAge=uint8(IDWT_3);

histogram(Embed_IMAge);

figure(),imshow(Embed_IMAge,[]);

impixelinfo;

title('Embedded Cover Image');

imwrite(Embed_IMAge,'Embed_IMAge.tiff');

disp('Stego-Object created');

PSNR =psnr(IMAge,Embed_IMAge);

SSIM=ssim(IMAge,Embed_IMAge);

err = immse(IMAge,Embed_IMAge);

Correlation = corr2(IMAge,Embed_IMAge);

fprintf('PSNR= %f - SSIM= %f\n MSE= %f\n',PSNR,SSIM,err);

fprintf('Correlation= %f \n',Correlation);
```

7.3. RECIVER_STEGNO_EXTRACTION.m

```
clc;

disp('STEGANOGRAPY EXTRACTION FOR THESIS | SDP2');

disp('Stego-Object found');

clear all;

close all;

Embed_IMAge=imread('Embed_IMAge.tiff');

figure,imshow(Embed_IMAge);

impixelinfo;

title('Stego Image');

load('Embed_IMAge.mat')

[LL1,HL1,LH1,HH1]=dwt2(Embed_IMAge,'haar');

dwt_1=[LL1,HL1;LH1,HH1];
```

```
figure,imshow(dwt_1,[]);

title('1-level decomposed cover image');

[LL2,HL2,LH2,HH2]=dwt2(LL1,'haar');

b_1=[LL2,HL2;LH2,HH2];

dwt_2=[b_1,HL1;LH1,HH1];

figure,imshow(dwt_2,[]);

title('2-level decomposed cover image');

[LL3,HL3,LH3,HH3]=dwt2(LL2,'haar');

c_1=[LL3,HL3;LH3,HH3];

cc_1=[c_1,HL2;LH2,HH2];

dwt_3=[cc_1,HL1;LH1,HH1];

figure,imshow(dwt_3,[]);

title('3-level decomposed cover image');

[Rows_3 Col_3]=size(HH3);

HH3_16 = (HH3);

Length_2 = 126;

Length_4 = 1;

for D=1:Rows_3

   for E=1:Col_3

       if(Length_4 <= Length_2 )

         Extract(Length_4)= HH3_16(D,E);

       end

       Length_4 =Length_4+1;

   end

end

Extract_Data = uint8(Extract);

idwt_1=idwt2(LL3,HL3,LH3,HH3,'haar');
```

```
figure,imshow(idwt_1,[]);
title('1-level embedded image');
idwt_2=idwt2(idwt_1,HL2,LH2,HH2,'haar');
figure,imshow(idwt_2,[]);
title('2-level embedded image');
idwt_3=idwt2(idwt_2,HL1,LH1,HH1,'haar');
figure,imshow(idwt_3,[]);
title('3-level embedded image');
Extract=uint8(idwt_3);
figure(),imshow(Extract,[]);
impixelinfo;
title('Extracted Cover Image');
imwrite(Extract,'Extract.tiff');
disp('Stego-Object created');
disp('Text message extracted');
fid=fopen('extraction.txt','w');
for F=1:Length_2
    fprintf(fid,'%c',Extract_Data(F));
end
```

7.4. RECIVER_DECRYPTION.m

```
clc ;
clear all;
close all;
 fileID_2 = fopen('extraction.txt','r');
Extract_MSG= fscanf(fileID_2,'%d');
fclose(fileID_2);
```

```
Length_5=length(Extract_MSG);
ranka_1 = Extract_MSG(1:16);
rankb_2 = Extract_MSG(17:32);
data_text_RSA = ranka_1;
data_text_AES = ranka_1;
[inv_s_box, w,inv_poly_mat] = aes_init;
Extracted_AES_Text = inv_cipher (data_text_AES, w, inv_s_box, inv_poly_mat, 1);
p = input('\nEnter the value of p: ');
q = input('\nEnter the value of q: ');
[Pk,Phi,d,e] = intialize(p,q);
for G= 1:Length_5
  message(G)= crypt(Extract_MSG(G),Pk,d);
end
disp('Decrypted ASCII of Message:');
disp(message(17:32));
Extracted_RSA_Text = message(17:32)
Extracted_original_msg = cat(2,Extracted_AES_Text, Extracted_RSA_Text);
disp(['Decrypted Message is: ' Extracted_original_msg]);
disp('Decrypted text file generated');
fid=fopen('readable.txt','w');
fprintf(fid,'%c',Extracted_original_msg);
```

CHAPTER-8

TESTING

8. TESTING

8.1. TESTING

Software testing is a critical element of software quality assurance and represents the ultimate service of specification design and coding. The increasing visibility of software as a system element and the attended costs associated with the software failure and motivating forces for well planned, through testing. It is not unusual for a software development to spend between 30 and 40 percent of total project effort in testing.

System Testing Strategies for this system integrate test case design techniques into a well-planned series of steps that result in the successful construction of this software. It also provides a road map for the developer, the quality assurance organization and the customer, a roadmap that describes the steps to be conducted as path of testing, when these steps are planned and then undertaken and how much effort, time and resources will be required. The test provisions are follows.

8.1.1. TESTING OBJECTIVES

The following are the testing objectives -

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as yet undiscovered error.
- A successful test is one that uncovers as a yet undiscovered error.

The above objectives imply a dramatic change in view point. They move counter to the commonly held view that a successful test is one in which no errors are found. Our objective is to design tests that systematically different clauses of errors and do so with minimum amount of time and effort.

8.1.2. TESTING PRINCIPLES

Before applying methods to design effective test cases, a software engineer must understand the basic principles that guide software testing.

- All tests should be traceable to customer requirements.
- Tests should be planned ling before testing begins.
- Testing should begin "in the small" and progress towards testing "in the large".

- Exhaustive testing is not possible.

8.2. TEST PLAN

A test plan is a document that contains a complete set of test cases for a system, along with other information about the testing process. The test plan should be return long before the testing starts.

Test plan identifies

- A task set to be applied as testing commences ,
- The work products to be produced as each testing task is executed
- The manner, in which the results of testing are evaluated, recorded and reuse when regression testing is conducted.

In some cases the test plan is indicated with the project plan. In others the test plan is a separate document.

8.3. TESTING STRATEGIES

A strategy for software testing indicates software test case design methods in to a well-planned series of steps that results in the successful construction of software. The strategy provides a road map that describes the steps to be conducted as part of testing, when these steps are planned and then undertaken, and how much effort, time and resources will be required. It must be rigid enough to promote reasonable planning and management tracking as the project progresses.

The strategies for testing are envisioned by the following methods – A number of software testing strategies have been proposed. All provide the software developer with a template for testing and all the following generic characteristics.

- To perform effective testing, a software team should conduct effective formal technical reviews. By doing this, many errors will be eliminated before testing commences.
- Testing begins at the component level and works ―outward‖ toward the integration of entire computer based system.
- Different testing techniques are appropriate at different points in time.
- Testing is conducted by the developer of the software and an independent test group.
- Testing and debugging are different activities, but debugging must accommodate in any testing strategy.

8.3.1. TYPES OF TESTING

The primary objective of test case design is to derive a set of tests that have the highest likelihood for uncovering errors in the software. To accomplish this objective two different categories of test case design techniques are used.

8.3.1.1. WHITE-BOX TESTING

White box testing sometimes-called glass box testing is a test case design that focuses on the program control structure. Test cases are derived to ensure that

1. Guarantee that all independent paths within a module have been exercised at least once.
2. Exercise all logical design on their true and false sides
3. Executes all loops at their boundaries and within their operational boundaries.
4. Exercise internal data structure to ensure their validity.

Several methods are used in the white box testing

- **Statement Coverage** In this each statement in a program is executed at least once. This is done by checking the program in debug mode and verifying each statement.
- **Branch Coverage** In this each and every branch cases are tested whether the different branch conditions are true or false. It is a strong test criterion over statement coverage testing.
- **Condition Coverage** Condition testing is a test case design method that exercises to logical conditions contained in a program module. The purpose of condition testing is to detect not only errors in the condition of a program but also other errors in the program.
- **Path Coverage** In this testing all the linearly independent paths in a program are executed once.
- **Cyclomatic Complexity** Software metric provides a quantitative measure of a program. The Mc Cabe‘s Cyclomatic complexity of a program defines the number of independent paths in a program.

8.3.1.2. BLACK BOX TESTING

Tests can be conducted at software interface by knowing the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operational, at the same time searching for errors in called black box testing, sometimes called as behavioural testing.

Black box testing is not an alternative to white box techniques. Rather a complementary approach is likely to uncover a different class of errors than white box methods. Black box tests are designed to uncover errors in functional requirements without regard to the internal workings of a program. Black box testing techniques focus on the information domain of the software.

Black box testing attempts to find errors in the following categories –

a) Incorrect or missing functions.
b) Interface errors
c) Errors in the data structures or external database access
d) Performance errors
e) Initialization and termination errors

8.3.1.3. INTEGRATION TESTING

Integration testing is a systematic technique for constructing the program structure and to conduct tests for uncovered errors with interfacing. In this system, Top-Down integration is performed for the construction of program structures.

8.3.1.4. SYSTEM TESTING

System tests are designed to validate a fully developed system with a view to assuming that it meets its requirements. There are three kinds of system testing:

1. **Alpha Testing** Alpha testing refers to the system testing that is carried by the customer within the organization along with the developer. The alpha tests are conducted in controlled manner.
2. **Beta Testing** Beta testing is the system performed by a selected group of customers, the developer is not present at the site and the user will inform the problems that are encountered during testing. The software developer makes the necessary changes and submits to the customer.
3. **Acceptance Testing** Acceptance testing is the system testing performed by the customer to whether or not to accept the delivery of the system.

8.3.1.5. VALIDATION TESTING

This testing is performed to ensure that the system functions in a manner that can be reasonably by the users.

Here input data is validated first at interface (authentication), before sending it to the server. In this system, for example hidden message is analysed before being transmitted and after being received by the expected recipient. That is to guarantee less distortion occurs within to the original cover file after concealing the secret text.

By this testing, we can ensure that all functional requirements are satisfied, and all performance requirements are achieved.

8.4. TEST CASE DESIGN

Test case is an explicit set of instructions designed to detect a particular class of defect in a software system, by bringing about a failure. A test case can give rise to many tests. Each test is a particular run of the test case on a particular version of the system.

8.4.1. INFORMATION TO INCLUDE IN A FORMAL TEST CASE

Each test case should have the following information.

8.4.2. IDENTIFICATION AND CLASSIFICATION

Each test case should have a number, and may be given a descriptive title that indicates its purpose. The system, subsystem or module being tested should also be clearly indicated, with a reference to the related requirements and design documents.

8.4.3. INSTRUCTIONS

These tell the tester exactly what to do. The instructions must tell the tester how to put the system into the required initial state and what inputs to provide.

8.4.4. EXPECTED RESULT

This tells the tester how the system should behave in response to the instructions i.e. what it should output and what state it should then be in. The tester reports the failure if he/she does not encounter the expected result.

8.4.5. LEVELS OF IMPORTANCE OF TEST CASE

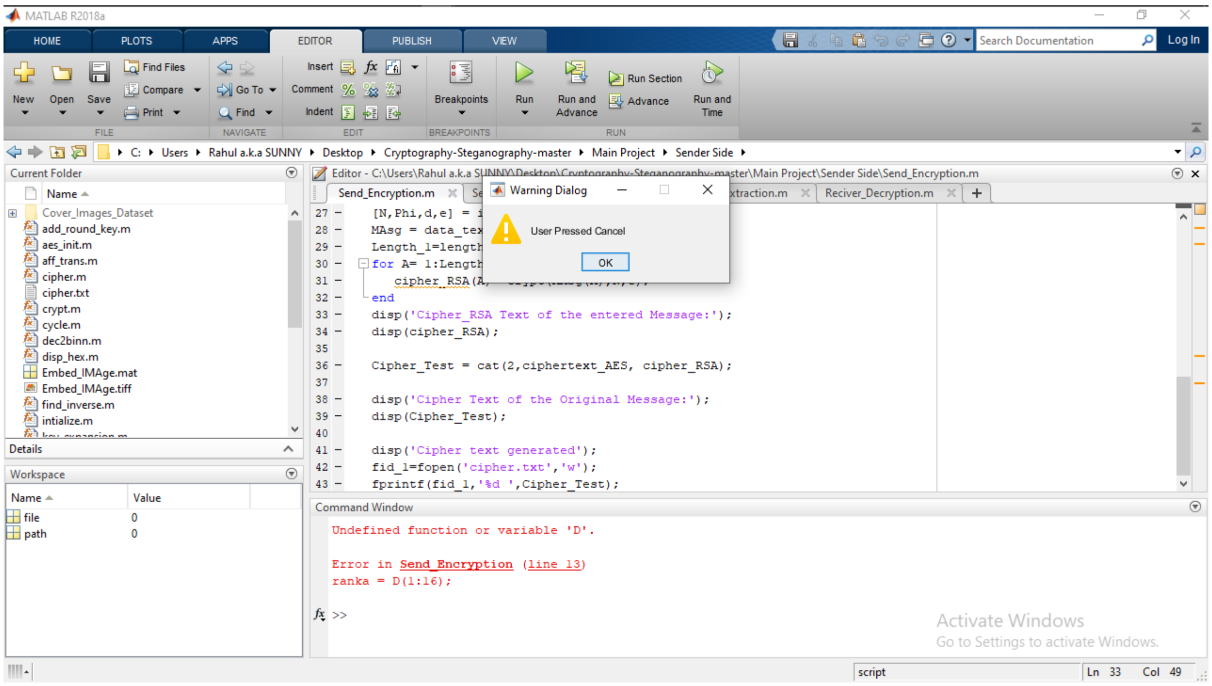
It is a good idea to classify the test cases according to their importance, or severity level. The most important test cases are executed first, and are designed to detect the most severe classes of defect. A typical scheme for levels of importance is:

- **Level 1** First, pass critical test cases- These are designed to verify that the system runs and is safe. Any level 1 failure normally means that no further testing is possible.
- **Level 2** General Test cases- These verify that the system performs its day-to-day functions correctly and is therefore a success‘. A level two failure while important to fix, may still permit testing of other aspects of the testing to continue in the meantime.
- **Level 3** Detailed Test Cases- These test requirements are of lesser importance. If desired, level 3 test cases can also be used to provide some redundancy. If there are many failures of level 3 test cases the system can probably be used, but is lacking in overall quality.

8.6 TEST CASES

The following figures are some of validations done in our project

1. The following test case showcases a situation when the sender decides to not give any input, the system is designed in a way to return a dialog box when such things happen



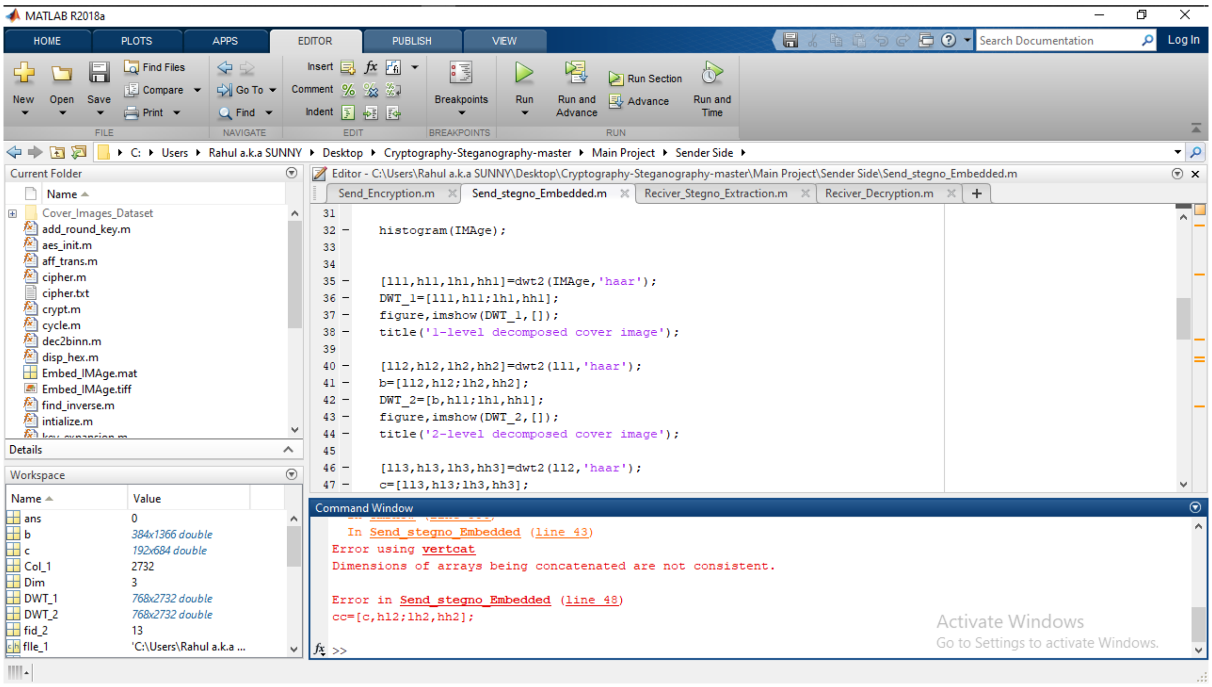
```
Editor - C:\Users\Rahul a.k.a SUNNY\Desktop\Cryptography-Steganography-master\Main Project\Sender Side\Send_Encryption.m

    [N,Phi,d,e] = i
    MAsg = data_tex
    Length_l=length
for A= 1:Length
    cipher_RSA(A,
end
disp('Cipher_RSA Text of the entered Message:');
disp(cipher_RSA);

Cipher_Test = cat(2,ciphertext_AES, cipher_RSA);

disp('Cipher Text of the Original Message:');
disp(Cipher_Test);

disp('Cipher text generated');
fid_1=fopen('cipher.txt','w');
fprintf(fid_1,'%d ',Cipher_Test);
```

Warning Dialog: User Pressed Cancel

```
Undefined function or variable 'D'.

Error in Send_Encryption (line 13)
ranka = D(1:16);
```

Figure 10: Warning Dialog

2. The following test case showcases a situation when the sender gives an input that does not meet the specified requirements causing error in the execution of the program

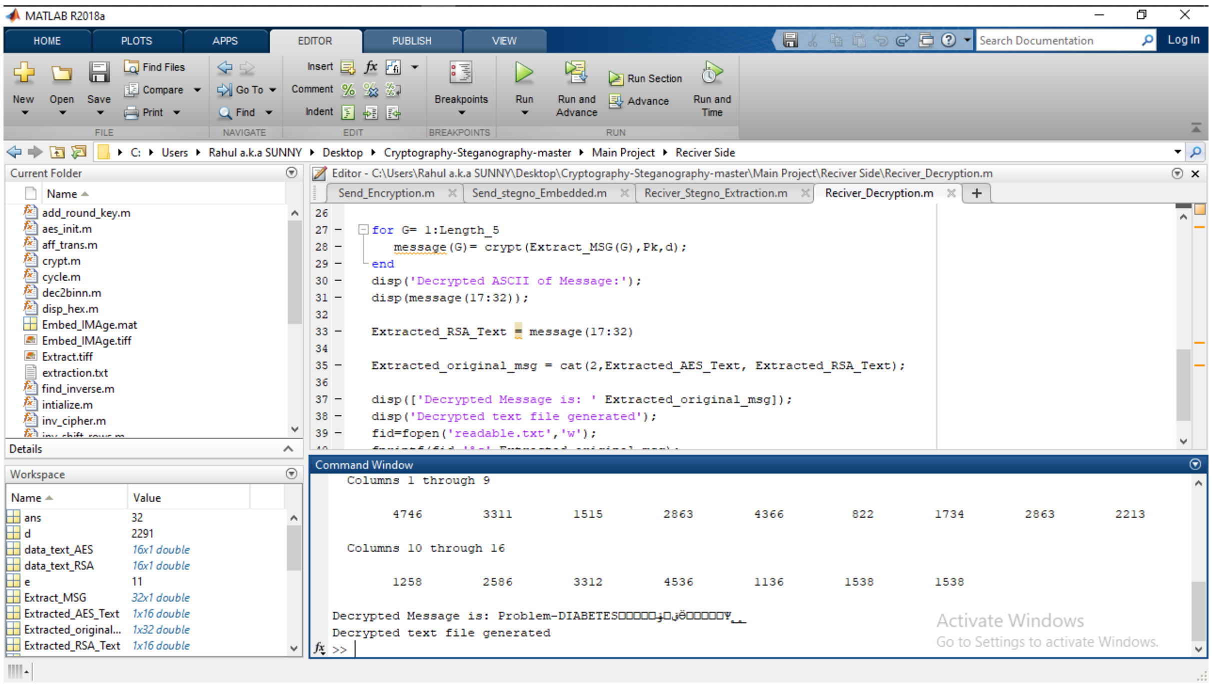


```
Editor - C:\Users\Rahul a.k.a SUNNY\Desktop\Cryptography-Steganography-master\Main Project\Sender Side\Send_stegno_Embedded.m

histogram(IMAge);


[ll1,hl1,lh1,hh1]=dwt2(IMAge,'haar');
DWT_1=[ll1,hl1;lh1,hh1];
figure,imshow(DWT_1,[]);
title('1-level decomposed cover image');

[ll2,hl2,lh2,hh2]=dwt2(ll1,'haar');
b=[ll2,hl2;lh2,hh2];
DWT_2=[b,hl1;lh1,hh1];
figure,imshow(DWT_2,[]);
title('2-level decomposed cover image');

[ll3,hl3,lh3,hh3]=dwt2(ll2,'haar');
c=[ll3,hl3;lh3,hh3];
```

```
  In Send_stegno_Embedded (line 43)
Error using vertcat
Dimensions of arrays being concatenated are not consistent.

Error in Send_stegno_Embedded (line 48)
cc=[c,hl2;lh2,hh2];
```

Figure 11: Improper Input

3. The following test case showcases a situation where the stego image that is sent by the sender to the receiver got corrupted somewhere along the transmission journey resulting in an altered message which is different from the one sent by the sender



Editor - C:\Users\Rahul a.k.a SUNNY\Desktop\Cryptography-Steganography-master\Main Project\Reciver Side\Reciver_Decryption.m

```
for G= 1:Length_5
   message(G)= crypt(Extract_MSG(G),Pk,d);
end
disp('Decrypted ASCII of Message:');
disp(message(17:32));

Extracted_RSA_Text = message(17:32)

Extracted_original_msg = cat(2,Extracted_AES_Text, Extracted_RSA_Text);

disp(['Decrypted Message is: ' Extracted_original_msg]);
disp('Decrypted text file generated');
fid=fopen('readable.txt','w');
```

Command Window

```
  Columns 1 through 9

        4746        3311        1515        2863        4366         822        1734        2863        2213

  Columns 10 through 16

        1258        2586        3312        4536        1136        1538        1538

Decrypted Message is: Problem-DIABETES□□□□□ﻗ□ﯚ□□□□□¥ـ‿
Decrypted text file generated
>> 
```

Figure 12: Improper Output

CHAPTER-9

OUTPUT SCREENS

9. OUTPUT SCREENS

9.1. Input Window:

Following output screen is when the system requests the user to input a plain text file which contains the messeage that needs to be delivered to the recepient.

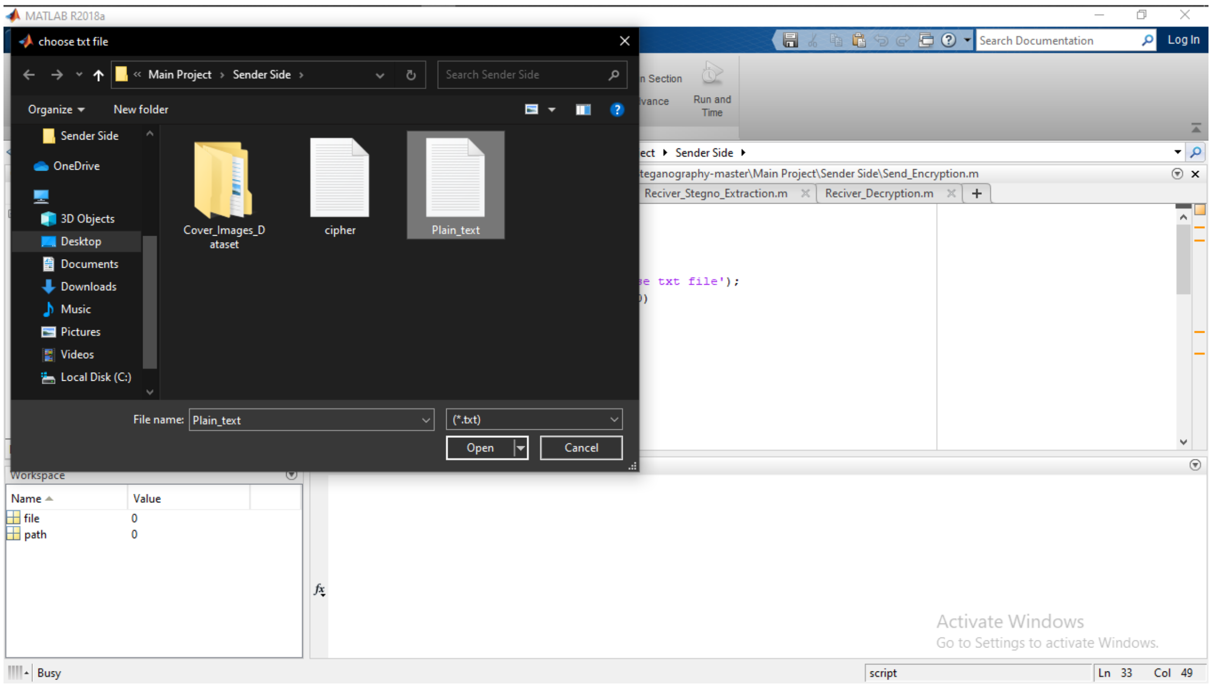
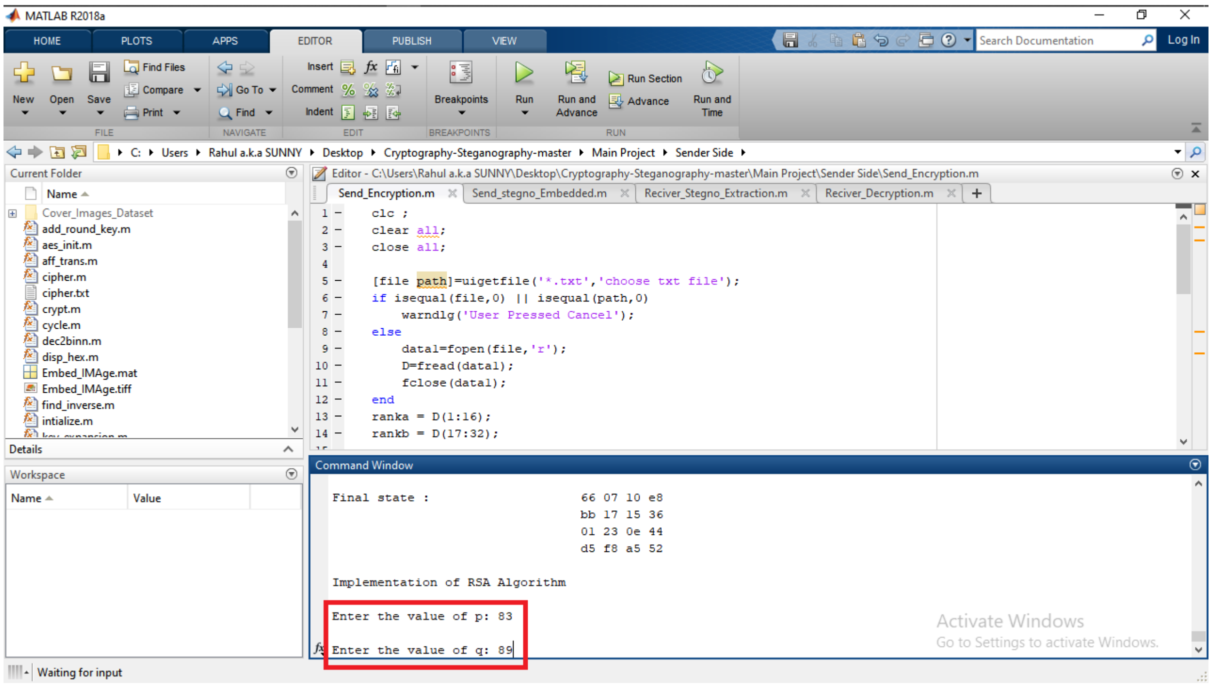


Figure 13: Input .txt file

Following output screen is when the system requests user to enter two large prime numbers for generation of public and private keys through RSA algorithm.

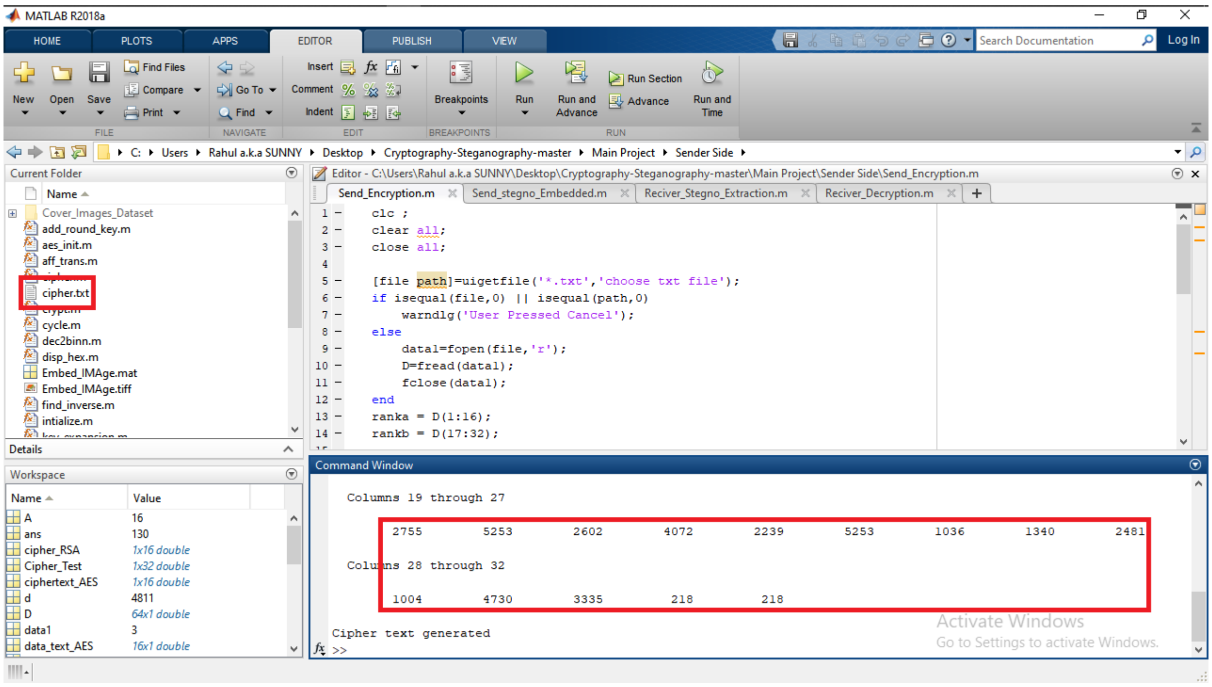


```
clc ;
clear all;
close all;

[file path]=uigetfile('*.txt','choose txt file');
if isequal(file,0) || isequal(path,0)
    warndlg('User Pressed Cancel');
else
    data1=fopen(file,'r');
    D=fread(data1);
    fclose(data1);
end
ranka = D(1:16);
rankb = D(17:32);
```

```
Final state :                   66 07 10 e8
                                bb 17 15 36
                                01 23 0e 44
                                d5 f8 a5 52

Implementation of RSA Algorithm

Enter the value of p: 83

Enter the value of q: 89
```

Figure 14: RSA Implementation

Following output screen showcases Cipher Text which is generated for a given plain text and is stored within the same location as that of the program



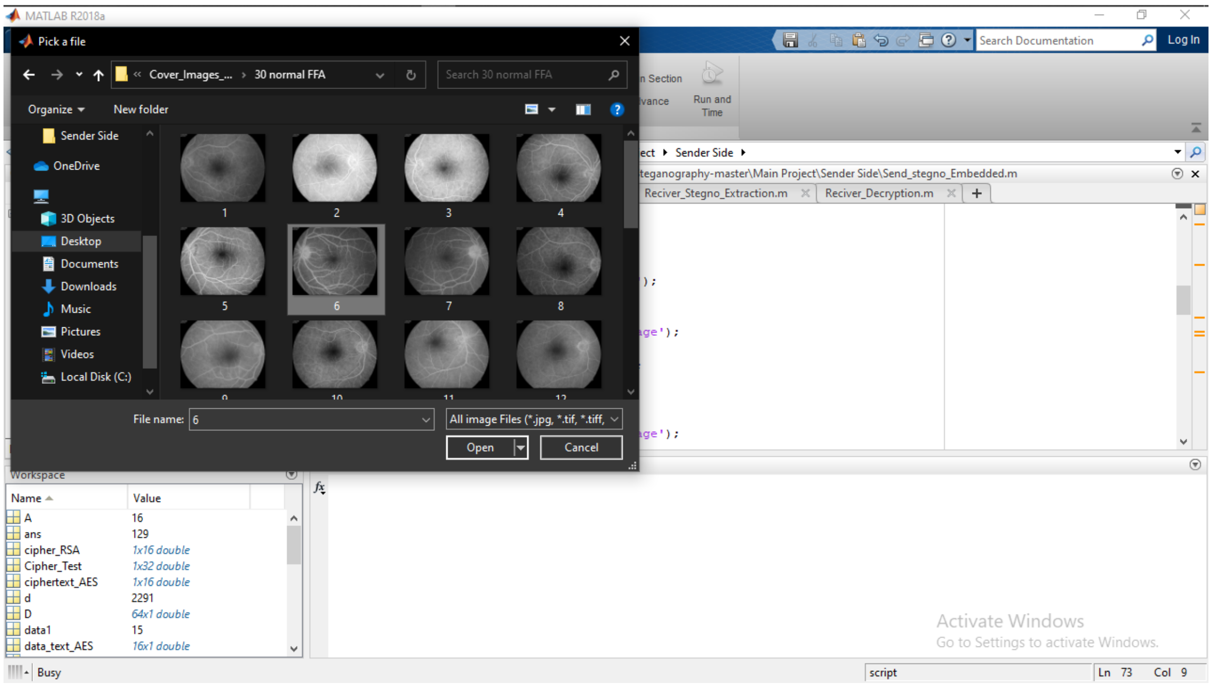
```
clc ;
clear all;
close all;

[file path]=uigetfile('*.txt','choose txt file');
if isequal(file,0) || isequal(path,0)
    warndlg('User Pressed Cancel');
else
    data1=fopen(file,'r');
    D=fread(data1);
    fclose(data1);
end
ranka = D(1:16);
rankb = D(17:32);
```

Name	Value
A	16
ans	130
cipher_RSA	1x16 double
Cipher_Test	1x32 double
ciphertext_AES	1x16 double
d	4811
D	64x1 double
data1	3
data_text_AES	16x1 double

```
  Columns 19 through 27

        2755        5253        2602        4072        2239        5253        1036        1340        2481

  Columns 28 through 32

        1004        4730        3335         218         218

Cipher text generated
>>
```

Figure 15: Cipher text

Following output screen is when the system requests the user to input an image which is used to conceal the cipher message in turn returning a stego-image that is sent to the recepient.

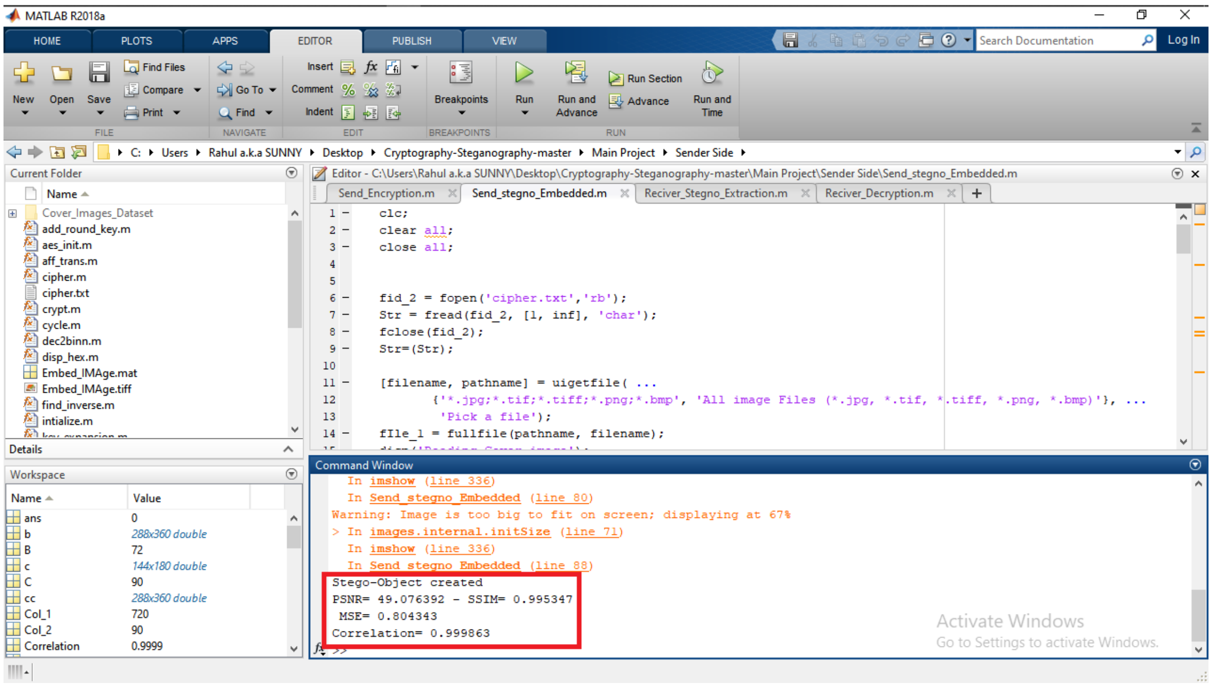


Name	Value
A	16
ans	129
cipher_RSA	1x16 double
Cipher_Test	1x32 double
ciphertext_AES	1x16 double
d	2291
D	64x1 double
data1	15
data_text_AES	16x1 double

Figure 16: Input image file

9.2. Output Window:

Following output screen showcases the statistical parameters that are generated each time whenever a stego-image is created which are later used for the analysis of the system.



Editor - C:\Users\Rahul a.k.a SUNNY\Desktop\Cryptography-Steganography-master\Main Project\Sender Side\Send_stegno_Embedded.m

```
clc;
clear all;
close all;


fid_2 = fopen('cipher.txt','rb');
Str = fread(fid_2, [1, inf], 'char');
fclose(fid_2);
Str=(Str);

[filename, pathname] = uigetfile( ...
        {'*.jpg;*.tif;*.tiff;*.png;*.bmp', 'All image Files (*.jpg, *.tif, *.tiff, *.png, *.bmp)'}, ...
         'Pick a file');
fIle_1 = fullfile(pathname, filename);
```

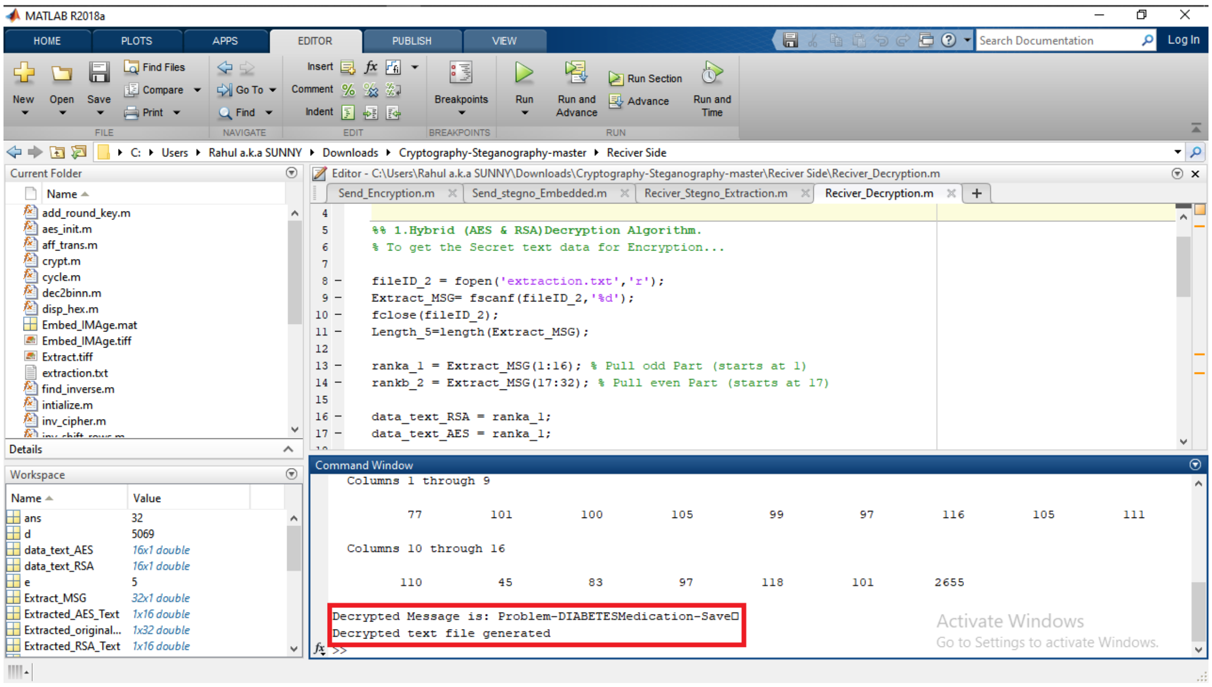
Command Window

```
  In imshow (line 336)
  In Send_stegno_Embedded (line 80)
Warning: Image is too big to fit on screen; displaying at 67%
> In images.internal.initSize (line 71)
  In imshow (line 336)
  In Send_stegno_Embedded (line 88)
Stego-Object created
PSNR= 49.076392 - SSIM= 0.995347
 MSE= 0.804343
Correlation= 0.999863
>>
```

Name	Value
ans	0
b	288x360 double
B	72
c	144x180 double
C	90
cc	288x360 double
Col_1	720
Col_2	90
Correlation	0.9999

Figure 17: Statistical Parameters

Following output screen is the final result after successful transmission of message from sender to receiver



Name	Value
ans	32
d	5069
data_text_AES	16x1 double
data_text_RSA	16x1 double
e	5
Extract_MSG	32x1 double
Extracted_AES_Text	1x16 double
Extracted_original...	1x32 double
Extracted_RSA_Text	1x16 double

```

%% 1.Hybrid (AES & RSA)Decryption Algorithm.
% To get the Secret text data for Encryption...

fileID_2 = fopen('extraction.txt','r');
Extract_MSG= fscanf(fileID_2,'%d');
fclose(fileID_2);
Length_5=length(Extract_MSG);

ranka_1 = Extract_MSG(1:16); % Pull odd Part (starts at 1)
rankb_2 = Extract_MSG(17:32); % Pull even Part (starts at 17)

data_text_RSA = ranka_1;
data_text_AES = ranka_1;
```

```
  Columns 1 through 9

        77         101         100         105          99          97         116         105         111

  Columns 10 through 16

       110          45          83          97         118         101        2655

Decrypted Message is: Problem-DIABETESMedication-Save□
Decrypted text file generated
>>
```

Figure 18: Decrypted Message

9.3. Intermediary Images:

Following are the set of images that are generated during the intermediate stages of execution of the system. They are as follows.

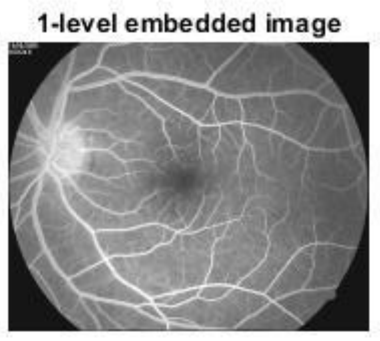


Figure 19: 2D-DWT-1L

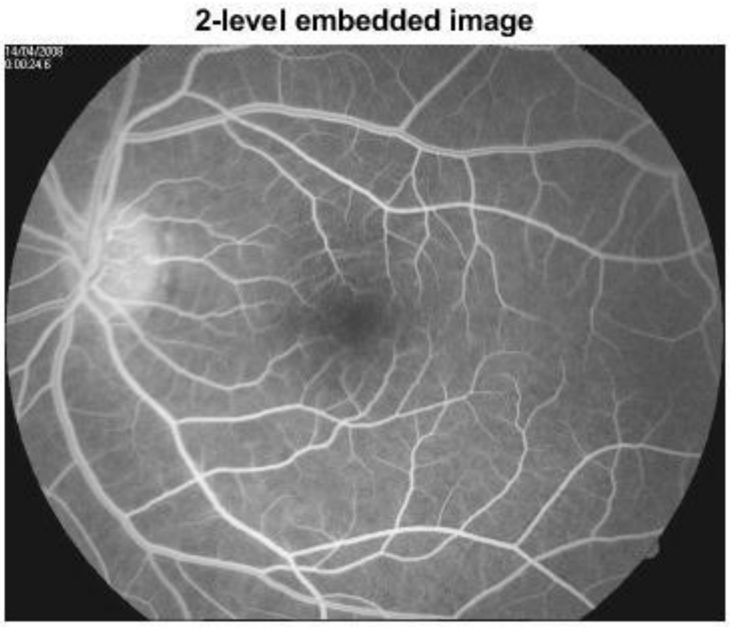
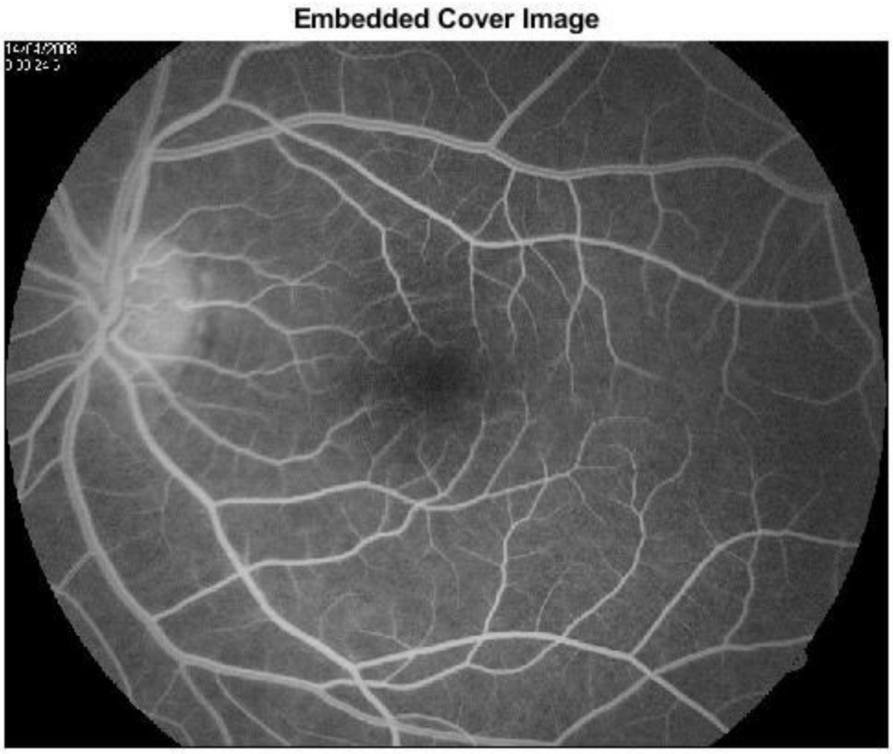


Figure 20: 2D-DWT-2L



Pixel info: (X, Y) Intensity

Figure 21: Stego-image

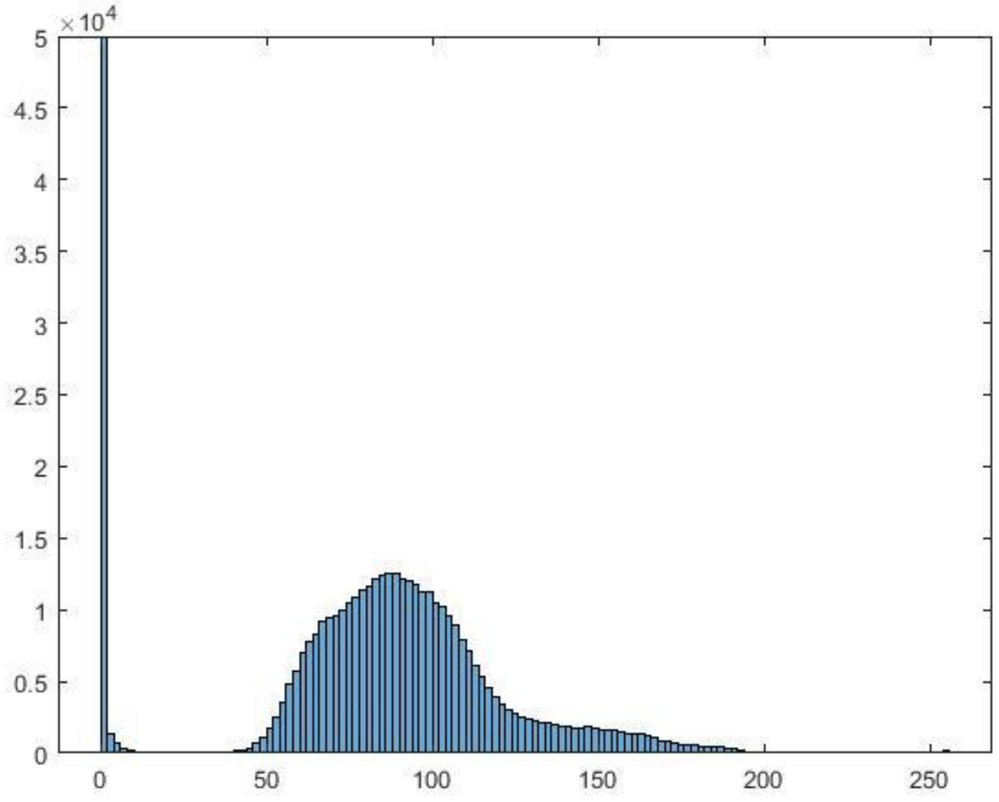


Figure 22: Histogram of Input image

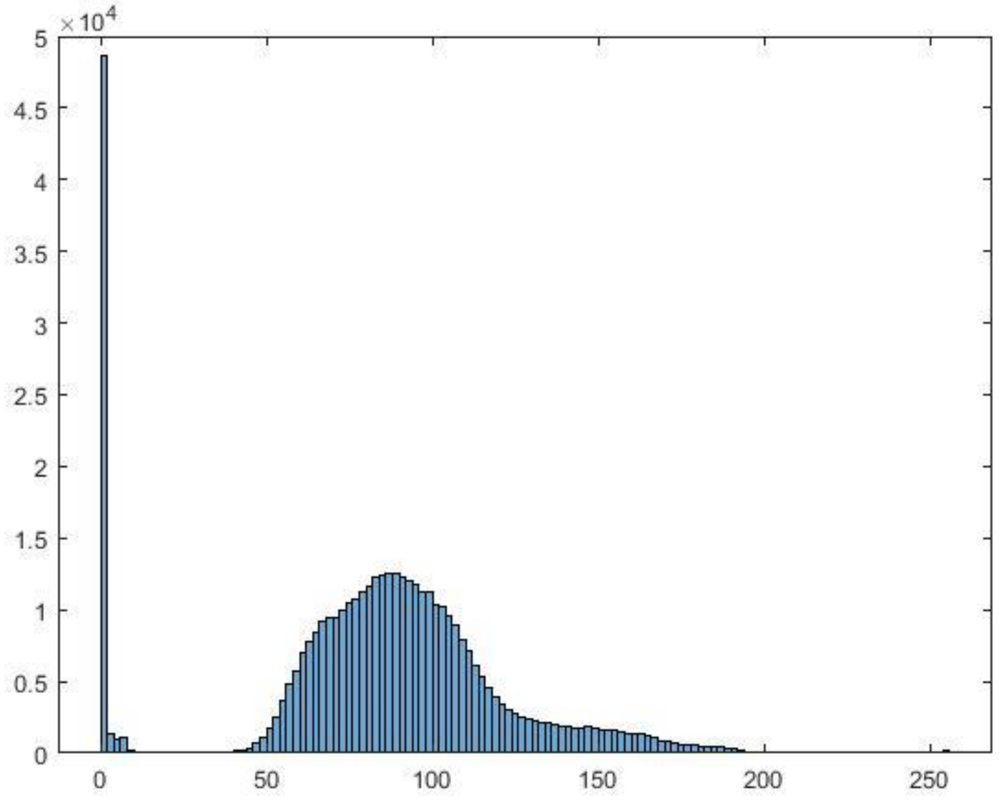


Figure 23: Histogram of Stego-image

CHAPTER 10

CONCLUSION

10. CONCLUSION

A secure patient's diagnostic data transmission model using both colour and grey-scale images as a cover carrier for healthcare based IOT environment has been proposed. The proposed model engaged 2D-DWT-1L, 2D-DWT-2L, 2D-DWT-3L steganography and hybrid blending AES and RSA cryptographic techniques. The experimental results were evaluated on both colour and grey-scale images with different text sizes. The performance was assessed based on the six statistical parameters (PSNR, MSE, BER, SSIM, SC, and correlation). Compared to the state-of-the-art methods, the proposed model proved its ability to hide the confidential patient's data into a transmitted cover image with high imperceptibility, capacity, and minimal deterioration in the received stego-image.

CHAPTER 11
REFERENCES

11.REFERENCES

[1] Ashraf Darwish, Aboul Ella Hassanien, Mohamed Elhoseny, Arun Kumar Sangaiah, Khan Muhammad, The Impact of the Hybrid Platform of Internet of Things and Cloud Computing on Healthcare Systems: Opportunities, Challenges, and Open Problems, Journal of Ambient Intelligence and Humanized Computing, 2017 (https://doi.org/10.1007/s12652-017-0659-1)

[2]Abdulaziz Shehab, Mohamed Elhoseny, Khan Muhammad, Arun Kumar Sangaiah, Po Yang, Haojun Huang, Guolin Hou; Secure and Robust Fragile Watermarking Scheme for Medical Images, IEEE Access, Volume: PP, Issue: 99, (DOI: 10.1109/ACCESS.2018.2799240).

[3] Bairagi, A. K., Khondoker, R., & Islam, R. (2016). An efficient steganographic approach for protecting communication in the Internet of Things (IOT) critical infrastructures. Information Security Journal: A Global Perspective, 25(4-6), 197-212.

[4] Anwar, A. S., Ghany, K. K. A., & Mahdy, H. E. (2015). Improving the security of images transmission. International Journal, 3(4).

[5] Ahmed Abdelaziza, Mohamed Elhoseny, Ahmed S. Salama, A.M. Riad,"A Machine Learning Model for Improving Healthcare services on Cloud Computing Environment", Measurement, Volume 119, April 2018, Pages 117-128, 2018 (https://doi.org/10.1016/j.measurement.2018.01.022).

[6] Paschou, M., Sakkopoulos, E., Sourla, E., & Tsakalidis, A. (2013). Health Internet of Things: Metrics and methods for efficient data transfer. Simulation Modelling Practice and Theory, 34, 186-199.

[7] Muhammad Sajjad, Mansoor Nasir, Khan Muhammad, Siraj Khan, Zahoor Jan, Arun Kumar Sangaiah, Mohamed Elhoseny, Sung Wook Baik, "Raspberry Pi assisted face recognition framework for enhanced law-enforcement services in smart cities", Future Generation Computer Systems, Elsevier, 2018 (DOI: https://doi.org/10.1016/j.future.2017.11.013)

[8] Kumar, P., & Lee, H. J. (2011). Security issues in healthcare applications using wireless medical sensor networks: A survey. Sensors, 12(1), 55-91.

[9] Razzaq, M. A., Sheikh, R. A., Baig, A., & Ahmad, A. (2017). Digital image security: Fusion of encryption, steganography and watermarking. International Journal of Advanced Computer Science and Applications (IJACSA), 8(5).

[10] Dey, N., & Santhi, V. (Eds.). (2017). Intelligent Techniques in Signal Processing for Multimedia Security. Springer International Publishing.