COMMUNICATION ENGINEERING LAB 2

EC4091D

COURSE PROJECT

ENCRYPTED COMMUNICATION USING IMAGE TRANSFER

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Group No: C3

Guided By: Dr Vinay Joseph

(Assistant Professor Grade I)

S.NO	ROLL NUMBER	NAME OF THE STUDENT
1.	B201032EC	PRANITH KUMAR BOGE
2.	B201015EC	PULYAPUDI ADITYA
3.	B201037EC	MUDAVATH KUMAR
4.	B201036EC	MOHINDER CHAWAN
5.	B200929EC	RISHI DHARMESHKUMAR SHAH

OBJECTIVE:

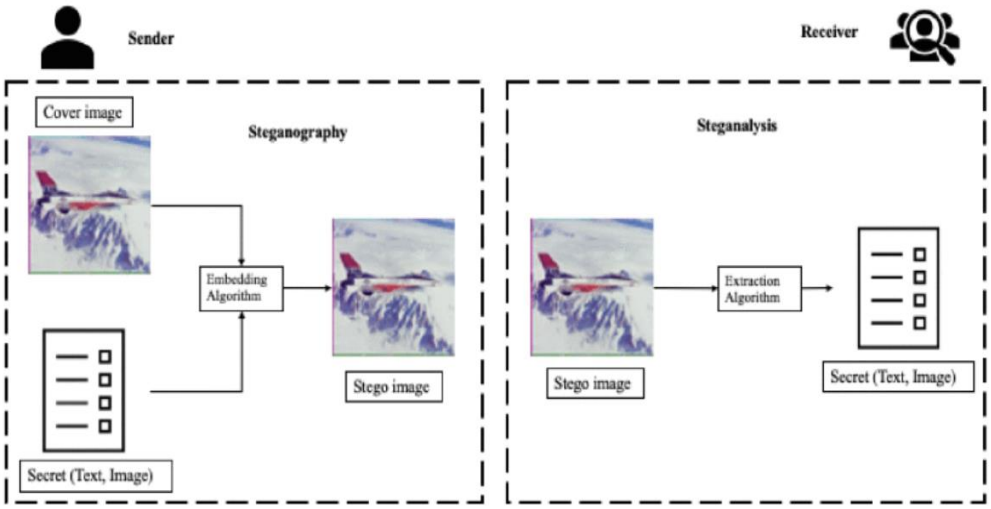
To create an End-to-End Encrypted Communication model that provides secure information transfer using LSB and DCT Image steganography techniques. Where the message is embedded into an Image and sent or the Image is Watermarked to maintain authenticity of image.

THEORETICAL BACKGROUND:

Information hiding techniques have been available for a long time but their importance has been increasing recently. The main reason is the increase in the data traffic through the internet and social media networks. Though the objectives of cryptography and steganography are similar, there is a subtle difference. Cryptography makes the data unbreakable and unreadable but the cipher text is visible to human eyes. Steganography, which is used to hide the information in plain sight, allows the use of wide variety of the secret information forms like image, text, audio, video, and files. Digital watermarking is another method where confidential information is embedded to claim ownership.

➔ IMAGE STEGANOGRAPHY:

Image Steganography is the process of hiding information which can be text, image, or video inside a cover image. *The secret information is hidden in a way that it not visible to the human eyes.* The main goal is *to embed data in such a manner (LSB or DCT) so that it remains undetectable to an observer who is not aware the message's existence*. This practice is particularly valuable for secure communication, where the sender wishes to conceal the fact that sensitive information is being exchanged. We can also use these techniques to watermark an image (using DCT). Watermarking is a technique with similarities to steganography. It has been around for centuries and is commonly used in money and stamps to assist in identifying counterfeiting. The idea behind watermarking is to create a translucent image on the paper to provide authenticity.



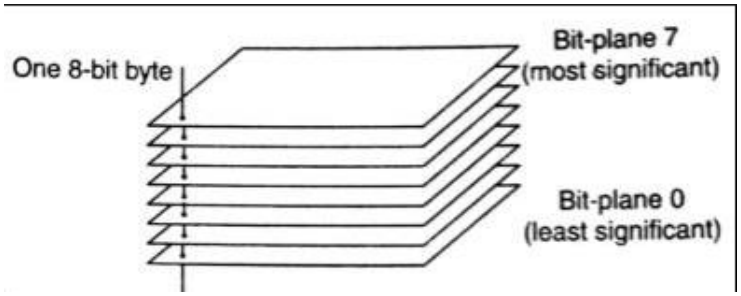
In the above figure the inputs are the cover image and the secret information and an embedding algorithm is used to generate the container stego (steganography) image. The extraction algorithm takes the stego image as input to extract the ingrained secret information. The figure shows the process of Image Stegaography.

METHODS OF IMAGE STEGANOGRAPHY USED:

- LSB (LEAST SIGNIFICANT BIT) SUBSTITUTION.
- DCT (DISCRETE COSINE TRANSFORM) STEGANOGRAPHY.

➢ **LSB (LEAST SIGNIFICANT BIT) SUBSTITUTION:**

In a grayscale image, each pixel is represented by a certain number of bits (typically 8 bits per pixel in an 8-bit grayscale image). These bits can be organized into planes, with each plane representing the value of a specific bit.



For example, in an 8-bit grayscale image, you can have eight bitplanes:

- Bitplane 7: 2^7 (128)
- Bitplane 6: 2^6 (64)
- Bitplane 5: 2^5 (32)
- Bitplane 4: 2^4 (16)
- Bitplane 3: 2^3 (8)
- Bitplane 2: 2^2 (4)
- Bitplane 1: 2^1 (2)
- Bitplane 0: 2^0 (1)

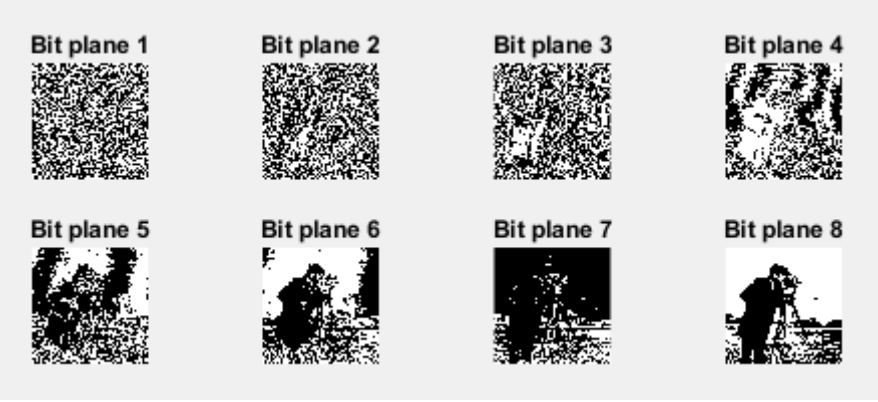


Figure shows Bit-planes of cameraman.tif image

LSB steganography is a technique where information is hidden in the least significant bit of each pixel's color value in an image. The least significant bit is the rightmost bit in a binary representation. Since it has the least impact on the overall pixel intensity, small changes to this bit are less likely to be noticeable to the human eye.

In the context *of LSB steganography, you hide information by modifying the least significant bit of each pixel in one or more bitplanes.*

This method is particularly suited for high-quality images, where not all pixels are actively used, and it operates on the premise that altering a few pixel values should not introduce noticeable changes to the overall visual appearance.

In the LSB substitution process, the secret information, whether text, another image, or ***any data, undergoes conversion into binary form.*** This binary representation is then embedded into the pixels of the cover image. The least significant bits(LSBs) of these pixel's intensity that represent are selected, and they are modified to incorporate the binary bits from the secret information.

Despite the simplicity and effectiveness of LSB substitution, the process requires caution and strategic implementation. The substitution method must strike a delicate balance between hiding sufficient information and avoiding visible changes. Overloading the cover image with too much hidden data can lead to noticeable alterations, potentially compromising the concealment of secret information. We, should note that this type of steganography usually the original image is not circulated so that people can't detect the difference between original and the stego image.

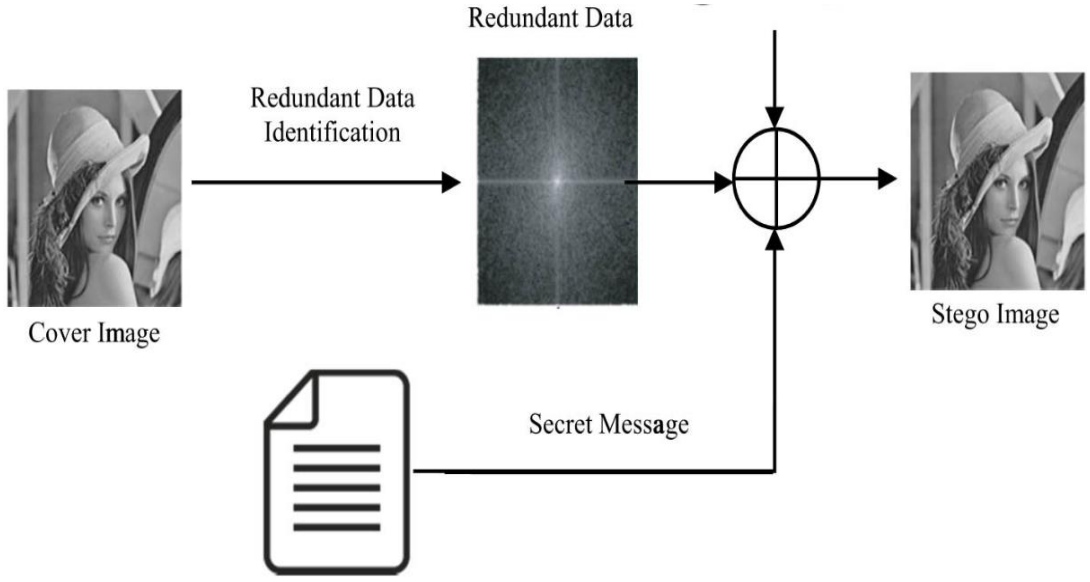


Figure shows the flow of LSB steganography (Redundant Data here is LSB of a Pixel's Intensity)

Algorithm to embed text message: -

Step 1: Read the cover image and text message which is to be hidden in the cover image.
Step 2: Convert text message in binary.
Step 3: Calculate LSB of each pixel of cover image.
Step 4: Replace LSB of cover image using XOR operation, with each bit of secret message one by one.
Step 5: Write stego image

Algorithm to retrieve text message: -

Step 1: Read the stego image.
Step 2: Calculate LSB of each pixel of stego image.
Step 3: Retrieve bits and convert each 8 bits into character.

- **DCT (DISCRETE COSINE TRANSFORM) STEGANOGRAPHY:**

In the context of steganography, DCT-based methods capitalize on the unique properties introduced by this transformation to embed hidden information with minimal visual impact.

DCT (Discrete Cosine Transform) based steganography involves embedding hidden information in the frequency domain of an image using the DCT. **Hence it effects the whole image making it difficult to recognize any difference with original image (making it better than LSB).** DCT is commonly used in image compression techniques, such as JPEG compression, and it transforms the image from spatial domain to frequency domain. This transformation allows for the manipulation of specific frequency components, making it a suitable method for steganography.

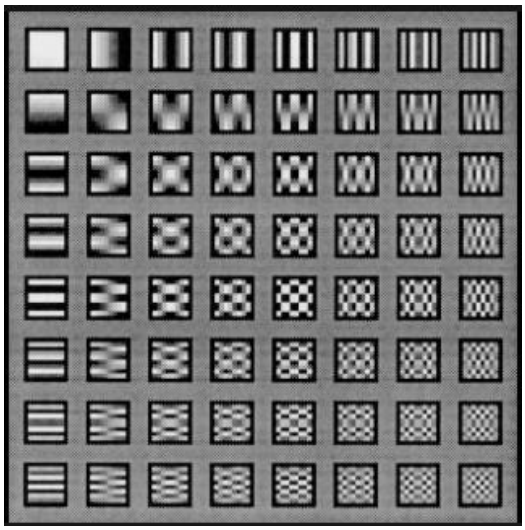


Figure shows the DCT produces a set of coefficients representing different frequency components of the image. The lower-frequency components are located in the top-left corner of the transformed block, and the higher-frequency components are towards the bottom-right. The steganographic information is embedded by modifying the DCT coefficients. Here we add the secret message to the DCT coefficients.

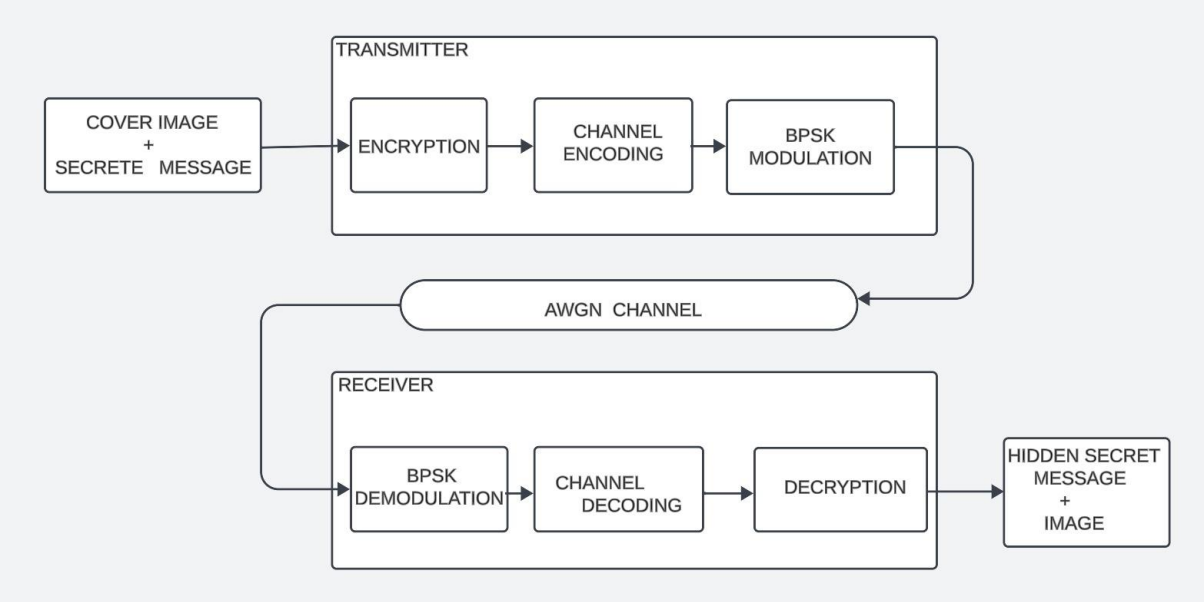
Algorithm to embed text message: -

Step 1: Read cover image.
Step 2: Read secret image.
Step 3: The cover image is broken into 8×8 block of pixels.
Step 4: Working from left to right, top to bottom subtract 128 in each block of pixels.
Step 5: DCT is applied to each block.
Step 6: Each block is compressed through quantization table.
(From step 3 to 6 we used inbuilt function B = dct2(A))
Step 7: Change DC coefficient by adding the watermark and replace with each bit of secret message.
Step 8: Convert the frequency domain back to spatial domain
(Here we used inbuilt function B = idct2(A))
Step 9: Write stego image.

Algorithm to retrieve text message: -

Step 1: Read stego image.
Step 2: Stego image is broken into 8×8 block of pixels.
Step 3: Working from left to right, top to bottom subtract 128 in each block of pixels.
Step 4: DCT is applied to each block.
Step 5: Each block is compressed through quantization table.
(From step 3 to 5 we used inbuilt function B = dct2(A))
Step 6: Subtract the original image dct with the stego dct to get secret message.
Step 7: Subtract the stego dct with saved secret m to get De-watermarked-image.

BLOCK DIAGRAM:



TRANSMITTER:

➔ **COVER IMAGE+ SECRET MESSAGE**:

It is important to choose a cover image that is large enough to hold the message without being visibly degraded. Images with more pixels, such as high-resolution images, can hold more data without being visibly affected. Also choose a cover image that is not too complex, as complex images can be more difficult to embed data into without being detected.

➔ **ENCRYPTION (STEGANOGRAPHY):**

The message is hidden into the cover image. There are many different algorithms that can be used for image steganography. Here we are using LSB and DCT steganography techniques. Each has its advantages and limitations, and their effectiveness depends on the specific application and the goals of the user.

➔ **CHANNEL ENCODING**:

The encrypted message is encoded to make it more robust to transmission errors as data is transmitted over a noisy channel (AWGN Additive White Gaussian Noise).

Channel encoding technique that we used is a simple Repetition 3 encoding. Repetition encoding is an error-correcting technique that involves repeating each bit or group of bits multiple times. The idea is that if an error occurs in one of the repetitions, the redundant information from the other repetitions can be used to correct or detect the error.

Let's say you have a sequence of bits: 0101.

In Repetition 3 encoding, you might repeat each bit three times: 000111000111.

➔ **BPSK MODULATION:**

The encoded message is modulated using Binary Phase-Shift Keying (BPSK). This is a simple modulation scheme that is well-suited for image steganography.

BPSK modulation works by encoding each bit of the message as a phase shift in a carrier signal. The two possible phase shifts represent the two possible phase states are usually 0 degrees and 180 degrees i.e. two binary values, 0 and 1.

Finally, 0 maps to -1 and 1 maps to 1

AWGN CHANNEL:

The modulated message is transmitted over an Additive White Gaussian Noise (AWGN) channel. This is a model for a noisy channel, such as the internet. AWGN noise is a type of noise that is evenly distributed across all frequencies. It can be caused by a variety of factors, such as thermal noise and interference from other electronic devices.

Mathematically, if x(t) is the transmitted signal, n(t) is the AWGN, and y(t) is the received signal, the relationship can be expressed as: y(t)=x(t)+n(t)

n(t) = sqrt(1/(2* SNR in (V/V));

SNR(V/V) = 10 .^ (SNR(db)/10);

RECEIVER:

➔ **BPSK DEMODULATION:**

The demodulated message is demodulated using BPSK. This process reverses the BPSK modulation that was performed at the transmitter. We use comparison with 0 i.e., bit > 0 then 1 else 0.

➔ **CHANNEL DECODING:**

The demodulated message is decoded to recover the encoded message. This process involves deciding about the value of each bit in the message based on the phase of the corresponding carrier signal.

As it was repetition 3 channel encoding. Hence, if

sum of the 1x3 arrays elements is > 2 then 1 , else 0.(incase of hard decision decoding)

Incase , of soft decision decoding we use distances

➔ **DECRYPTION:**

The encrypted message is decrypted. This process reverses the encryption that was performed at the transmitter.

➔ **HIDDEN SECRET MESSAGE + IMAGE :**

The hidden message is extracted from the cover image. This is the result of End-to-End Encrypted Communication model.

DEMONSTRATION OF RESULTS:

LSB Steganography

Cover Image into which Secret Message is going to be Embedded

Embedded Image(Stego Image) with Secret message hidden inside

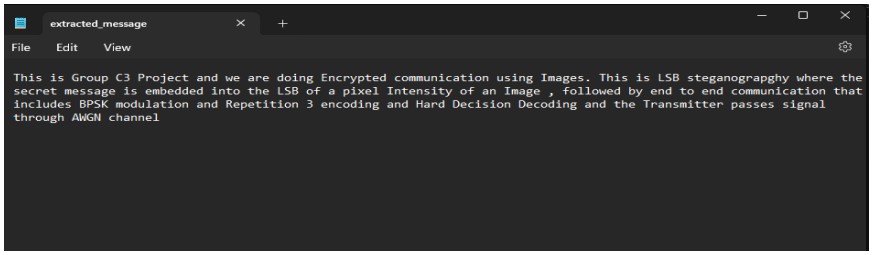
Differece Between Cover Image and Stego Image

Received Stego Image at Receiver with Secret message hidden inside

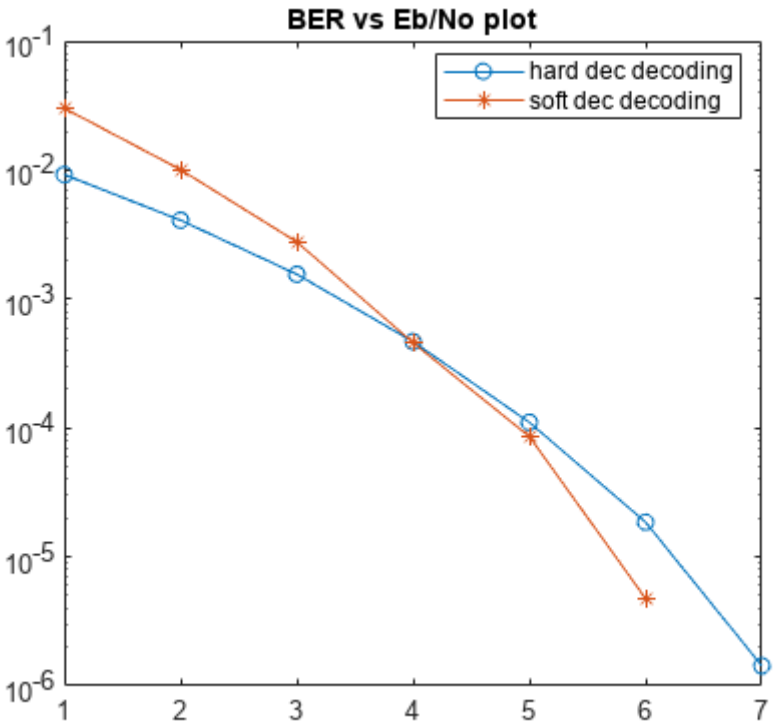
Code

```
xlim([1 162])
ylim([1 162])
```

Extracted message that is saved



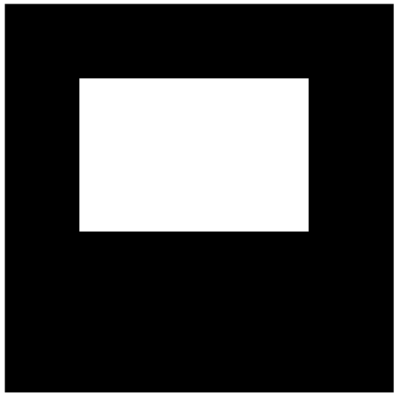
LSB Performance Analysis



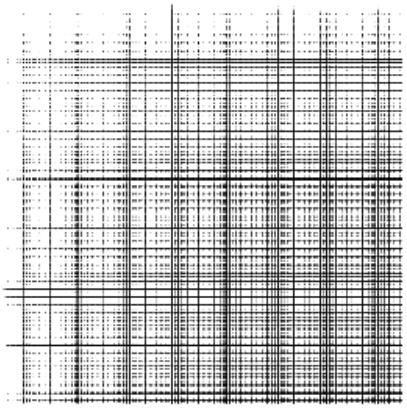
DCT Steganography

Cover Image into which Secret Message is going to be Embedded

Secret Message that is going to be Embedded



Differece Between Cover Image and Stego Image



Embedded Image(Stego Image) with Secret message hidden inside

Final De watermarked Image after End-to-End Communication

CONCLUSION:

We implemented Encrypted End to End communication using Image Steganography techniques like LSB and DCT to hide a secret message into the cover image. We applied BPSK modulation through an AWGN channel , we used Repetition 3 channel encoding and Hard and Soft Decision Decoding to tackle the errors formed due to AWGN channel.

REFERENCES:

- https://betterprogramming.pub/hide-data-within-an-image-507f571aab89
- https://in.mathworks.com/matlabcentral/answers/114513-dwt-dct-steganography-code-problem
- https://www.geeksforgeeks.org/lsb-based-image-steganography-using-matlab/
- T. Bhuiyan, A. H. Sarower, R. Karim and M. Hassan, "An Image Steganography Algorithm using LSB Replacement through XOR Substitution," *2019 International Conference on Information and Communications Technology (ICOIACT)*, Yogyakarta, Indonesia, 2019, pp. 44-49, doi: 10.1109/ICOIACT46704.2019.8938486.
- X. Li, X. Wang, A. Chen and L. Xiao, "A Simplified and Robust DCT-based Watermarking Algorithm," *2017 2nd International Conference on Multimedia and Image Processing (ICMIP)*, Wuhan, China, 2017, pp. 167-171, doi: 10.1109/ICMIP.2017.18.

MATLAB CODE:

LSB Steganography

```
clear all;
clc;

% Cover Image into which Secret Message is Embedded
input = imread('cameraman.tif');
input=imresize(input, [512 512]);
figure,
imshow(input)
title("Cover Image into which Secret Message is going to be Embedded")

% Secret Message to be embedded
message='This is Group C3 Project and we are doing Encrypted communication using
Images. This is LSB steganograpghy where the secret message is embedded into the
LSB of a pixel Intensity of an Image , followed by end to end communication that
includes BPSK modulation and Repetition 3 encoding and Hard Decision Decoding and
the Transmitter passes signal through AWGN channel';

% LSB IMAGE STEGANOGRAPHY WHERE THE IMAGE PIXELS
% LEAST SIGNITFIACT BIT OF 8 BITS USED TO REPRESENT
% PIXELS INTENSITY

% Length of the message where each character is 8 bits
len = length(message) * 8;

% Convert to ASCII values
ascii_value = uint8(message);

% Convert the decimal values to binary
bin_message = transpose(dec2bin(ascii_value, 8));

% Get all the binary digits in separate row
bin_message = bin_message(:);

% Length of the binary message
N = length(bin_message);

% Converting the char array to numeric array
bin_num_message= str2num(bin_message);

% Initialize output
output = input;

% Get height and width for traversing through the image
height = size(input, 1);
width = size(input, 2);

% Counter for number of embedded bits
embed_counter = 1;

% Traverse through the image
for i = 1 : height
    for j = 1 : width

        % If more bits are remaining to embed
        if(embed_counter <= len)
```

```
            % Finding the Least Significant Bit of the current pixel
            LSB = mod(double(input(i, j)), 2);

            % Find whether the bit is same or needs to change using XOR
            % operation
            temp = double(xor(LSB, bin_num_message(embed_counter)));

            % Updating the output to input + temp
            output(i, j) = input(i, j)+temp;

            % Increment the embed counter
            embed_counter = embed_counter+1;
        end

    end
end
figure,
imshow(output)
title("Embedded Image(Stego Image) with Secret message hidden inside")

figure,
imshow((output-input)*10000)
title("Differece Between Cover Image and Stego Image")

% Convert Image into Bitsream for Communication
s = size(output);
B = dec2bin(output(:), 8)';
C = B(:)' - '0';

% Apply BPSK modulation, repetition encoding, and hard decision decoding
Eb_Nodb = 10;
Eb_No = 10 .^ (Eb_Nodb/10);
n = 3;
k = 1;
R = k/n;
sigma_coded = sqrt(1/(2*Eb_No));
% Initialize the Transmitter's Output
comm_out = zeros(1, n);

disp(length(C))

% Transmitter and Receiver
for j = 1:length(C)
    % Take a single value of bitstream
    msg = C(j);
% TRANSMITTER
    % Repetition 3 encoding
    msg_encoded = [msg msg msg];
    % BPSK modulation
    mod_out = -1 + 2 * msg_encoded;
% CHANNEL
    % Add Gussian noise
    channel_out = mod_out + sigma_coded * randn(1, n);
% RECEIVER
    % demodulating
    demod_out = channel_out > 0;
    % Hard decision decoding and output of end to end communication
    comm_out(j) = sum(demod_out) >= 2;
end
```

```
% Convert  Bitsream into Image for Decryption
bitstream = num2str(comm_out); bitstream = bitstream(bitstream ~= ' ');
disp(size(bitstream));
X = uint8(bin2dec(reshape(num2str(bitstream), 8, [])'));
I = reshape(X, s);
figure,
imshow(I);
title("Received Stego Image at Receiver with Secret message hidden inside")

% Initialize a variable to store the extracted binary message
extracted_bin_message = '';

% Initialize a counter to keep track of the number of extracted bits
extract_counter = 1;

% Traverse through the image
for i = 1 : height
    for j = 1 : width
        % If more bits can be extracted
        if (extract_counter <= len)
            % Extract the LSB of the current pixel
            LSB = mod(double(I(i, j)), 2);

            % Append the extracted bit to the binary message
            extracted_bin_message = [extracted_bin_message, num2str(LSB)];

            % Increment the extract counter
            extract_counter = extract_counter + 1;
        else
            break;  % All bits have been extracted, exit the loop
        end
    end
    if (extract_counter > len)
        break;  % All bits have been extracted, exit the loop
    end
end

% Convert the binary secret message back to text
extracted_ascii_values = reshape(extracted_bin_message, 8, []).';
extracted_decimal_values = bin2dec(extracted_ascii_values);

% Concatenate the extracted decimal values into a single string
extracted_message = char(extracted_decimal_values)';

% Display the extracted message
disp(extracted_message);

% Specify the file name to save the extracted message
filename = 'extracted_message.txt';

% Write the extracted message to the file
dlmwrite(filename, extracted_message, '');

disp(['Extracted message saved to file: ', filename]);
```

LSB Performance Analysis

```
clear all;
```

```
clc;


% Cover Image into which Secret Message is Embedded
input = imread('cameraman.tif');
input=imresize(input, [512 512]);
imshow(input)
title("Cover Image into which Secret Message is going to be Embedded")


% Secret Message to be embedded
message='This is Group C3 Project and we are doing Encrypted communication
using Images. This is LSB steganograpghy where the secret message is embedded
into the LSB of a pixel Intensity of an Image , followed by end to end
communication that includes BPSK modulation and Repetition 3 encoding and Hard
Decision Decoding and the Transmitter passes signal through AWGN channel';



% LSB IMAGE STEGANOGRAPHY WHERE THE IMAGE PIXELS
% LEAST SIGNITFIACT BIT OF 8 BITS USED TO REPRESENT
% PIXELS INTENSITY


% Length of the message where each character is 8 bits
len = length(message) * 8;

% Convert to ASCII values
ascii_value = uint8(message);

% Convert the decimal values to binary
bin_message = transpose(dec2bin(ascii_value, 8));

% Get all the binary digits in separate row
bin_message = bin_message(:);

% Length of the binary message
N = length(bin_message);

% Converting the char array to numeric array
bin_num_message= str2num(bin_message);

% Initialize output
output = input;

% Get height and width for traversing through the image
height = size(input, 1);
width = size(input, 2);

% Counter for number of embedded bits
embed_counter = 1;

% Traverse through the image
```

```
for i = 1 : height
    for j = 1 : width

        % If more bits are remaining to embed
        if(embed_counter <= len)

            % Finding the Least Significant Bit of the current pixel
            LSB = mod(double(input(i, j)), 2);

            % Find whether the bit is same or needs to change using XOR
            % operation
            temp = double(xor(LSB, bin_num_message(embed_counter)));

            % Updating the output to input + temp
            output(i, j) = input(i, j)+temp;

            % Increment the embed counter
            embed_counter = embed_counter+1;
        end

    end
end


imshow(output)
title("Embedded Image(Stego Image) with Secret message hidden inside")
```

```
imshow((output-input)*10000)
title("Differece Between Cover Image and Stego Image")
```

```
% Convert Image into Bitsream for Communication
s = size(output);
B = dec2bin(output(:), 8)';
C = B(:)' - '0';
```

```
% Apply BPSK modulation, repetition encoding, and hard decision decoding
Eb_Nodb = 1:1:10 ;
Eb_No = 10 .^ (Eb_Nodb/10);
n = 3;
k = 1;
R = k/n;
```

```
%repetition channel coding
BER_hardcode = zeros(1,length(Eb_No));
```

```
for i= 1 : 10
    sigma_coded = sqrt(1/(2*Eb_No(i)));
    %initialization
    hard_num_errors = 0;
    soft_decod_out = 0;
    soft_num_errors = 0;
    dmin1=0;
    dmin2=0;
    % Initialize the Transmitter's Output
    comm_out = zeros(1, n);
    % Transmitter and Receiver
    for j = 1:length(C)
    % Take a single value of bitstream
    msg = C(j);
    % TRANSMITTER
    % Repetition 3 encoding
    msg_encoded = [msg msg msg];
    % BPSK modulation
    mod_out = -1 + 2 * msg_encoded;
    % CHANNEL
    % Add Gussian noise
    channel_out = mod_out + sigma_coded * randn(1, n);
    % RECEIVER
    % demodulating
    demod_out = channel_out > 0;
    % Hard decision decoding and output of end to end communication
    comm_out(j) = sum(demod_out) >= 2;
    hard_num_errors = hard_num_errors + abs(comm_out(j) - msg);
    % soft decision decoding
    dmin1 = sum((channel_out-[1 1 1]).^2);
    dmin2 = sum((channel_out-[-1 -1 -1]).^2);
        if(dmin2>=dmin1)
            soft_decod_out = 1;
        else
            soft_decod_out = 0;
        end
    soft_num_errors = soft_num_errors + abs(soft_decod_out - msg);
    end
    BER_hardcode(i) = hard_num_errors/2097152;
    BER_softcode(i) = soft_num_errors/209715;
end
```

```
semilogy(Eb_Nodb,BER_hardcode,'o-','DisplayName','hard dec decoding');
legend('Location',"best");
hold on
semilogy(Eb_Nodb,BER_softcode, '*-',"DisplayName",'soft dec decoding');
hold off
title ("BER vs Eb/No plot");
```

DCT Steganography

```
clear all
close all
clc


cover_image = double(imread('cameraman.tif'));
figure, imshow(cover_image / 255);
title("Cover Image into which Secret Message is going to be Embedded")


y = cover_image;
secret_image = zeros(256, 256);
secret_image(50:150, 50:200) = 1;
figure, imshow(secret_image)
title("Secret Message that is going to be Embedded")


save m.dat secret_image -ascii
```

```
% Watermarking
% DCT of RGB of cover image
ci_r = cover_image(:,:,1);



dx1 = dct2(ci_r);  dx11 = dx1 ;



load m.dat
% binary_mask_for_watermarking
g = 10; % Coefficient of watermark's strength
[rm, cm] = size(m);
% Adding the secret Image rows and columns into cover Image's All DCT
coefficients
dx(1:rm, 1:cm) = dx1(1:rm, 1:cm) + g * m;



% Inverse DCT for the Embedded Image creation
y1 = idct2(dx);


y(:,:,1) = y1;
```

```
figure, imshow(y/255)
title("Embedded Image(Stego Image) with Secret message hidden inside")
figure, imshow(abs(y-cover_image)*100)
title("Differece Between Cover Image and Stego Image")
```

```
% Convert to uint8
A = uint8(y);
imshow(A)
```

```
% Convert Image into Bitsream for Communication
s = size(A);
B = dec2bin(A(:), 8)';
C = B(:)' - '0';
```

```
% Apply BPSK modulation, repetition encoding, and hard decision decoding
Eb_Nodb = 10;
Eb_No = 10 .^ (Eb_Nodb/10);
n = 3;
k = 1;
R = k/n;
sigma_coded = sqrt(1/(2*Eb_No));
% Initialize the Transmitter's Output
comm_out = zeros(1, n);
```

```
disp(length(C))
```

```
% Transmitter and Receiver
for j = 1:length(C)
    % Take a single value of bitstream
    msg = C(j);
% TRANSMITTER
    % Repetition 3 encoding
    msg_encoded = [msg msg msg];
    % BPSK modulation
    mod_out = -1 + 2 * msg_encoded;
% CHANNEL
    % Add Gussian noise
    channel_out = mod_out + sigma_coded * randn(1, n);
% RECEIVER
    % demodulating
    demod_out = channel_out > 0;
    % Hard decision decoding and output of end to end communication
```

```
    comm_out(j) = sum(demod_out) >= 2;
end
```

```
% Convert  Bitsream into Image for Decryption
bitstream = num2str(comm_out); bitstream = bitstream(bitstream ~= ' ');
disp(size(bitstream));
X = uint8(bin2dec(reshape(num2str(bitstream), 8, [])'));
I = reshape(X, s);
imshow(I);
title("Received Stego Image at Receiver with Secret message hidden inside")
```

```
% Convert back to the original data type of y
com = cast(I, class(y));
```

```
z = com;
[r,c,s] = size(z) ;


% De-watermarking
% Clean image (known mask)
y = z;
dy1 = dct2(y(:,:,1));



dy(1:rm, 1:cm) = dy1(1:rm, 1:cm) - g * m;
comm = dy1(1:rm, 1:cm) - dx1(1:rm, 1:cm);
comm = uint8(comm)


save comm.dat secret_image -ascii


y11 = idct2(dy);


yy(:,:,1) = y11;


figure, imshow(yy/255)
title('Final De watermarked Image after End-to-End Communication')
```