PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

UE22EC342AC1 - DIP

Aug - Dec 2024

Report on

STEGANOGRAPHY

Submitted by :

PRAKASH (PES1UG22EC202)

KIRAN N NESVI (PES1UG22EC124)

MANOJKUMAR (PES1UG22EC154)

Course Instructor: Dr. Shikha Tripathi

Department of Electronics and Communication Engineering

CONTENTS:

ABSTRACT:

This project explores steganography using MATLAB, focusing on concealing messages within digital images through the Least Significant Bit (LSB) method. By embedding data in the LSBs of image pixels, this technique allows hidden communication without noticeable changes to the image. The project involves implementing algorithms to encode and decode hidden messages, ensuring data integrity and minimal visual distortion. This method has applications in secure communication and data privacy, demonstrating MATLAB's capability for digital image processing.

INTRODUCTION:

Steganography is the practice of concealing a message, image, or file within another message, image or file in such a way that it is hidden from plain view. In the context of image processing steganography involves hiding data within digital images. Unlike cryptography, which conceals the content, steganography conceals the existence of the message itself. This project utilizes image-based steganography to embed messages in images, leveraging MATLAB for the implementation. The objective is to achieve a secure method for embedding and retrieving messages while keeping the image visually unchanged

THEORY AND ALGORITHM:

Theory of Steganography

1. Carrier Medium (Cover): The digital file (image, audio, or video) in which the secret message is hidden. This carrier should look innocuous and unchanged to avoid suspicion.
2. Payload: The actual information or message intended to be concealed within the carrier. The payload is typically in binary form and encoded into the carrier's data.
3. Stego-Medium (Steganographic File): The final product, which is the modified carrier containing the embedded payload. An effective stego-medium should look visually or audibly identical to the original carrier to avoid detection.

Types of Steganographic Techniques

1. Spatial Domain Methods: These modify pixel values directly in the spatial domain of images, typically using methods like Least Significant Bit (LSB) substitution. LSB works by replacing the least significant bit(s) of each pixel's value with the payload's bits, causing negligible changes to the visual appearance.
2. Transform Domain Methods: These hide data within the frequency components of the media, using techniques like Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT), which make the hidden message more resistant to compression and filtering.
3. Adaptive Steganography: This approach selects regions of the carrier where modifications will be less noticeable, such as high-contrast areas, and adjusts embedding based on statistical features. Adaptive methods are more resilient to detection by steganalysis.

Algorithm for LSB Steganography

LSB (Least Significant Bit) insertion is one of the simplest and most commonly used algorithms in image-based steganography. Below is a step-by-step explanation of how it works:

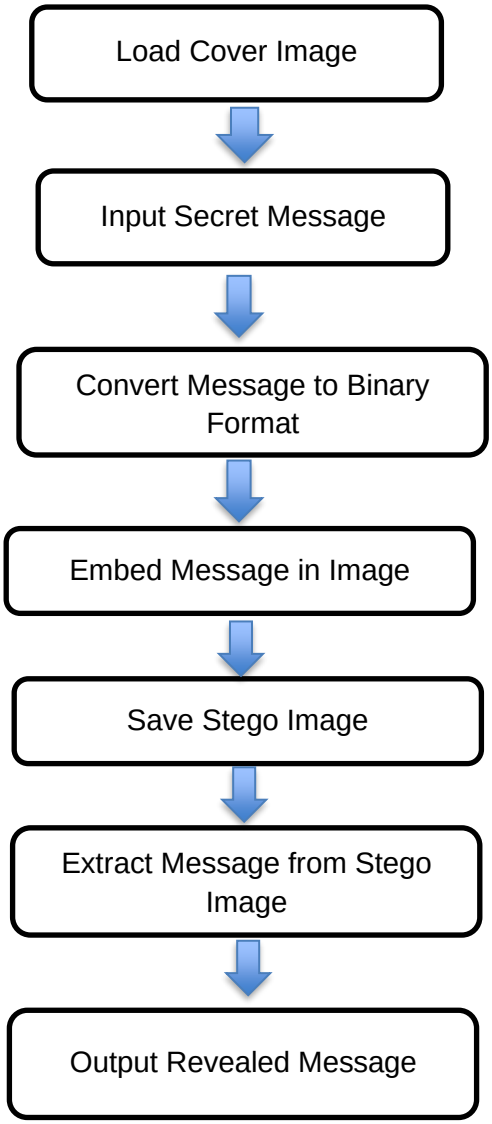
Algorithm Steps for Embedding Data

1. Convert Message to Binary:
    - Start by converting the secret message into binary form. Each character is represented as a binary sequence (usually 8 bits per character in ASCII encoding).
2. Select Image Pixels for Embedding:
    - Choose the pixels in the cover image where the binary data will be embedded. For a simple implementation, you may select pixels sequentially.
3. Replace LSBs of Pixels:
    - For each pixel selected, replace the least significant bit of the pixel's color channels (typically the RGB channels) with one bit from the message's binary sequence.
    - Example:
        - Original pixel RGB: (10110011, 11101001, 11011100)
        - Binary of message bit to embed: 1
        - Modified RGB after embedding: (10110011, 11101000, 11011101)
    - This replacement makes minimal visible changes to the image as only the LSB is altered.
4. Repeat Until Complete:
    - Continue replacing the LSB of each pixel with message bits until the entire message has been embedded.
5. Save the Stego-Image:
    - Save the modified image as the stego-image, which now contains the hidden message.

Algorithm Steps for Extracting Data

1. Load the Stego-Image:
    - Load the image that contains the hidden message.
2. Retrieve LSBs from Pixels:
    - For each pixel used in the embedding process, read the least significant bit from the chosen color channels. Extract each LSB in sequence until all bits of the hidden message are retrieved.
3. Reconstruct the Binary Data:
    - Convert the extracted bits back into bytes and then to characters to reconstruct the original message.
4. Display the Message:
    - The extracted message should now be readable and match the original payload.

FLOWCHART:



CODE:

```
% Steganography grayscale image
clc
close all
clear all
original = imread('coins.png');
original = double(original);
[row col] = size(original);

secret = imread('Amitabh_Bachchan.jpg');
```

```
secret = double(secret);
for x = 1:1:row
   for y = 1:1:col
      temp1 = dec2bin(original(x,y),8);
      temp2 = dec2bin(secret(x,y),8);
      temp1(8) = temp2(1); % Replacing the LSB with the MSB
      temp1(7) = temp2(2); % If we replace more than 1 bit
      temp1(6) = temp2(3); % we start getting the watermark
      %temp1(5) = temp2(4);
      %temp1(4) = temp2(5);
      %temp1(3)= temp2(6);
      %temp1(2) = temp2(7);
      %temp1(1)= temp2(8);
      stego(x,y) = bin2dec(temp1);
   end
end
stego = uint8(stego);
imwrite(stego,'newimage.png');
% above line will create image in our existing folder
subplot(2,2,1)
imshow(uint8(original))
title('Carrier Image');
subplot(2,2,2)
imshow(uint8(secret))
title('Secret Image');
subplot(2,2,3)
imshow(uint8(stego))
title('Stego Image');
%%% Retrieval of Secret Data

image = imread('newimage.png');
image = double(image);
[row col] = size(image);
for x = 1:1:row
   for y = 1:1:col
      temp = dec2bin(image(x,y),8);% This gives a character value
      if (temp(8)=='0')
         temp='00000000';
         secret_data(x,y)=bin2dec(temp);
      else
         temp='11111111'
```

```
            secret_data(x,y) = bin2dec(temp);
        end
    end
end
subplot(2,2,4)
imshow(secret_data)
title('Retrieved Image');
```

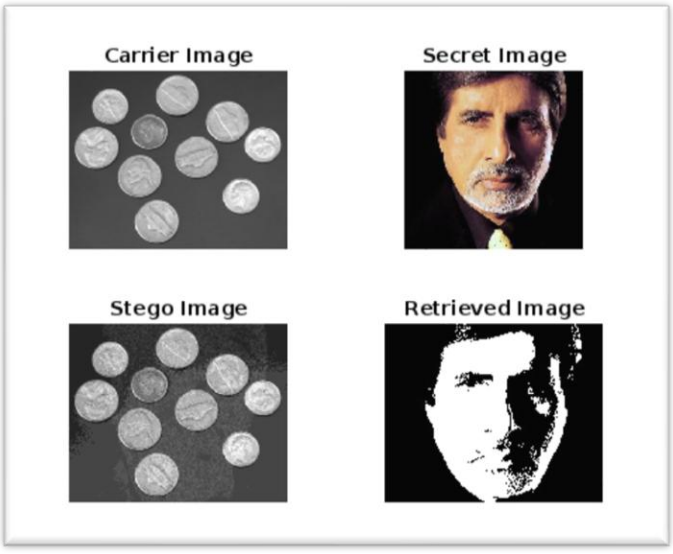
RESULTS AND OUTPUT SCREENSHOTS:

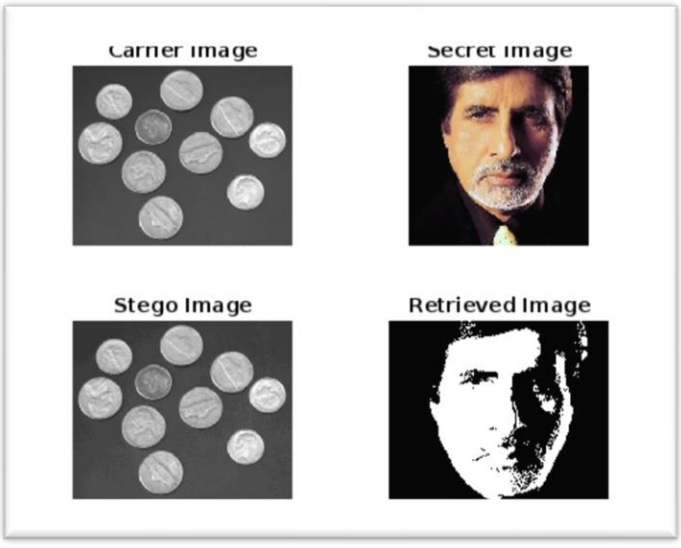
The results of this project demonstrate successful steganography using a grayscale image, with a hidden "secret" image embedded within a "carrier" image. By replacing the least significant bits (LSBs) of the carrier image with the most significant bits (MSBs) of the secret image, we achieve a visually undetectable embedding in the stego-image.

The resulting stego-image retains the appearance of the original carrier image, showing minimal visual difference, which is ideal for concealing the hidden data. However, as we increase the number of bits replaced, the watermark effect from the secret image becomes slightly more visible in the stego-image, showing a trade-off between data embedding capacity and image quality.

Upon retrieval, the hidden secret data is extracted from the stego-image. The retrieval process successfully distinguishes the hidden bits and reconstructs the retrieved secret image, which appears in binary form (with pixel values either black or white) due to the binary comparison of the extracted LSBs. This confirms the successful embedding and retrieval of hidden data.

This project validates that basic LSB steganography effectively hides and extracts data in grayscale images with minimal visible impact. However, the approach is sensitive to embedding capacity, as using multiple bits for embedding may increase the visibility of the secret image within the stego-image.





We tried to remove the watermark in the second image.

OBSERVATIONS AND CONCLUSIONS:

Observations

1. Stego Image Quality: The stego image, created by embedding the secret image within the carrier image, shows minimal visible change from the original carrier image. This demonstrates that LSB-based steganography is effective at concealing data without significantly altering the appearance of the cover image.
2. Trade-Off Between Capacity and Quality: As more bits in the carrier image are replaced with bits from the secret image, the embedded data becomes more noticeable. Using only 1-3 bits for embedding provides better visual quality, while using more bits introduces a watermark effect of the secret image, highlighting the balance between data capacity and image quality.
3. Successful Retrieval of Secret Image: The secret image was accurately retrieved from the stego image by examining the modified bits. The binary output correctly indicates areas where data was embedded, confirming the effectiveness of the retrieval algorithm.
4. Binary Appearance of Retrieved Image: The retrieved image appears in binary (black and white), reflecting the extracted bits. This outcome is a result of interpreting the embedded bits in their simplest form and shows the limitations of embedding more detailed data with basic LSB techniques.
5. Sensitivity to Image Modifications: This method is sensitive to image modifications like compression or resizing, which may alter the LSBs and compromise the integrity of the hidden message. This highlights a limitation of basic LSB steganography for robustness.

Conclusions

1. Feasibility of LSB Steganography: The project successfully demonstrates the feasibility of using LSB steganography to embed and retrieve data within grayscale images. The LSB method proves effective for simple, small-capacity embedding tasks.
2. Visual Integrity and Data Concealment: LSB steganography provides a good balance between data concealment and visual integrity, making it difficult for an observer to detect hidden information when minimal bits are replaced.
3. Optimal Use for Small Data Payloads: LSB-based embedding works best for small, non-complex messages, as it allows for high image quality. For embedding larger or more complex data, other methods or adaptive steganography techniques may be more suitable.
4. Future Enhancements: To improve robustness, future implementations could incorporate encryption or more advanced embedding techniques, which would make the method less susceptible to image modifications and enhance data security.

NOVELTY AND INDIVIDUAL CONTRIBUTION:

Novelty: This project explores using MATLAB for implementing the steganography algorithm efficiently. Potential enhancements include multi-bit LSB modifications or adding encryption for added security.

CONTRIBUTION:

Prakash (research about the project)
Kiran (Matlab code)
Manoj (PPT and report)

REFERENCES:

R. S. Mohit, "Image Steganography: A Review of the Recent Advances," IEEE Journals & Magazine, vol. XX, no. XX, pp. XX-XX, 2021.