JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A Fast Hybrid Image Steganography Approach with High Fidelity

[1]Jyotirmoy Bharali, [2]Laxman Sahoo
[1]M.Tech Scholar, [2]Senior Professor
[1]Department of Computer Science and Engineering,
[1]GITAM University, Visakhapatnam, India.

Abstract: Steganography is the practice of encrypting coded information within a document. It is often used in the same way as encryption, which prevents confidential data access without the proper tools or techniques used by experts in steganalysis. It has applications in various media sources such as text, images, video, audio, network channels, etc. This paper focuses on the image steganography technique using Discrete Cosine Transform and Least Significant Bit Substitution. The cover image is an RGB image. It is converted into YCbCr color space before a 2-D DCT is applied, where secret information is embedded. The inverse 2-D DCT is applied, and the image is converted back to RGB to give the stego image. The quality of the image was measured using PSNR (Peak Signal to Noise Ratio), MSE (Mean Square Error) and SSIM(Structural Similarity Index). Results showed a good quality stego image and low time complexity of the embedding and extraction algorithm.

***IndexTerms* – Steganography,DCT, LSB, MSE, PSNR,RGB,YCbCr,SSIM**

I. Introduction

Steganography is the method of concealing information into a source to hide it from unwanted attention. The word Steganography comes from Greek steganographia, which adds the word ‘steganos’, which means covered or concealed, and ‘graphia’, meaning writing [1]. It is different from cryptography as cryptography involves altering the structure of data in order to hide it whereas in steganography, the information is left unaltered and hidden in another source of media. Both of these techniques are related to information security and have its own pros and cons. Cryptography fails if the attacker cracks the ciphertext to get the plain text whereas Steganography fails when the attacker comes to know the presence of hidden data in the stegano object [2]. Digital multimedia like images and videos use watermarking, steganography, etc. to protect it from piracy and copyright issues. Other types of media which can use Steganography are audio, text, and network. An image is a representation of a real image in a numerical format known as picture elements or pixels. These pixels can be manipulated by using image steganography to hide secret information. There are four important terminologies related to image steganography [3]. Cover Image where the original secret data is to be hidden, message which is the secret information, and Stego Image which is the image generated after hiding the message in the cover image. There are two types of steganographic techniques depending on the image domain, namely the spatial domain and the transform/frequency domain. In spatial domain steganography, the secret information is directly embedded into the pixel values of the image. In the transform domain, the image is converted to the frequency domain. The secret information is then embedded in the transform coefficients obtained. For the purpose of our research, we will focus on a spatial domain technique known as Least Significant Bit (LSB) substitution and a transform domain method known as Discrete Cosine Transform (DCT).

1.1 Least Significant Bit (LSB) Steganography

LSB substitution [4] forms the most commonly used technique to hide a large amount of data into the cover image. The least significant bit is the last bit of a number in binary and has little significance because changing it will only increment or decrement the number by 1. A 24-bit image contains values ranging from 0 to 255 for each of the red, green, and blue channels. Therefore, changing the values of each pixel by LSB substitution in order to hide a message will not have a drastic effect on the image. The variation in the image is impossible to detect by the Human Visual System (HVS). LSB steganography is easy to implement and can embed a large capacity depending on the dimensions of the cover image. However, it comes with the drawback of being less robust and susceptible to noise and image transformations.

1.2 Discrete Cosine Transform (DCT) Steganography

DCT is a technique used in JPEG compression algorithm to transform successive 8x8 blocks of image from spatial domain to 64 DCT coefficients in the frequency domain [5]. The result of the transformation will generate 64 coefficients which consist of 63 AC coefficients and one single DC coefficient. The DC coefficient represents zero frequency, while the 63 AC coefficients show the non-zero frequencies.

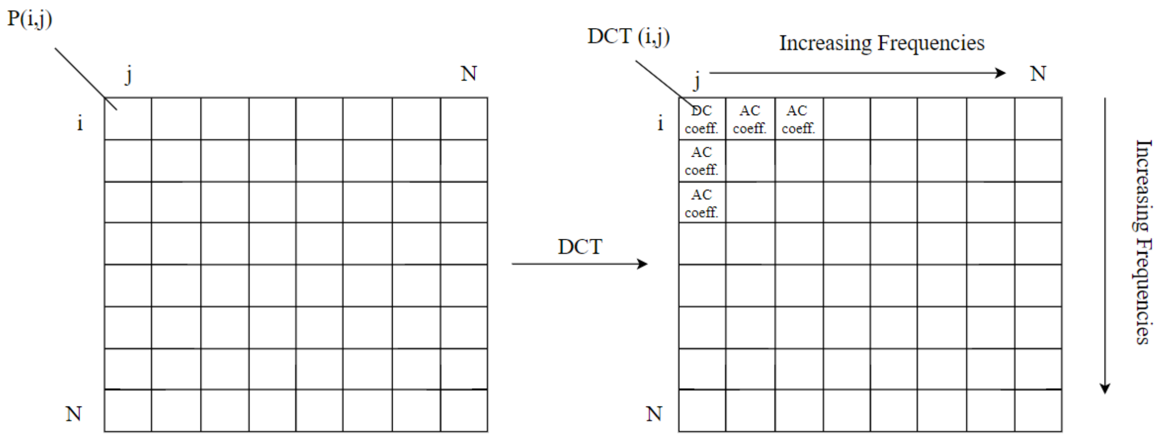


Figure 1: DCT of a NxN block

$$DCT(i,j) = \frac{1}{\sqrt{2N}} C(i)\, C(j) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} P(i,j) \cos\left[\frac{(2i+1)\pi i}{N}\right] \cos\left[\frac{(2j+1)\pi j}{N}\right] \quad (1)$$

Where $C(x) = \frac{1}{\sqrt{2}}$ if x is 0, else 1 if x > 0

$$P(i,j) = \frac{1}{\sqrt{2N}} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} C(i)\, C(j)\; \cos\left[\frac{(2i+1)\pi i}{N}\right] \cos\left[\frac{(2j+1)\pi j}{N}\right] \quad (2)$$

Where $C(x) = \frac{1}{\sqrt{2}}$ if x is 0, else 1 if x > 0

Equation (1) shows the DCT of pixel P(i,j) is given by the following formulas for an NxN matrix. To reconstruct the image after DCT, we use its inverse function shown in equation (2). The picture representation is concentrated on the upper left coefficients of the output matrix when DCT is applied to the input data.In contrast, the lower right coefficients contain less useful data. This useless data can be eliminated in the next step known as quantization, thus compressing the input image [6]. It has its applications in JPEG compression.

II. RELATED WORK

There has been a lot of research work on the field of steganography. In this section we have discussed some significant works using LSB, DCT or a hybrid method with a combination of other techniques or both. Arun et al. [7] proposed a simple method of hiding the secret message into the cover image using a stego key. This key will serve as a password needed to extract the message. Sari & Siahaan [8] did a comparison study between 1-bit and 2-bit LSB steganography. The results showed that 1-bit is better for image quality, while 2-bit is superior for larger embedding capacity. Saxena et al. [9] proposed a method that is a combination of three methods image compression using a wavelet transform, encryption of the secret message using a symmetric key and LSB substitution for embedding the secret data. The first steganographic algorithm for JPEG images, called JSteg, was developed by Upham [10]. The algorithm applies Discrete Cosine Transform to the image blocks and embeds the information into the LSBs of the DC coefficients sequentially. The algorithm being sequential and no presence of a stego key makes the algorithm susceptible to eavesdropping. Edgar M. [11] proposed an algorithm of DCT steganography using Python libraries opencv and bitstring. They only used the luminance layer to embed the message. The stego image obtained is of good quality even though the technique has low embedding capacity. AbdelWahab et al. [12] proposed an algorithm that introduced a secret key to encrypt the message for additional security. They also compressed the image after the quantization stage, in order to lower the cost of transmission of the stego image over the internet. Saroj & Banu [13] proposed a hybrid scheme based on DWT, DCT and Singular Value Decomposition (SVD). The cover image is wavelet decomposed to two levels and cosine transformed by one level. The SVD message image is then cosine transformed and embedded with the cover image replacing the least prominent data. The generated stego image has less distortions. The important feature of this work is that the correlation coefficient of the secret image which is exactly one-quarter of the image is 1.000.

III. PROPOSED SYSTEM

The proposed steganography algorithm is composed of two steps i.e., the embedding part as well as the extraction step. The RGB image must be converted into YC_bC_r format to reduce the bandwidth. Y stands for the luminance component, and C_b and C_r are the chrominance-red and chrominance-blue components. Luminance carries information on the brightness of the channel. Chrominance is the difference between a color and a reference channel at the same brightness. Because the human brain is not sensitive to minor variations in brightness and chrominance, this conversion during the embedding process is necessary.

$$Y = 0.299R + 0.587G + 0.114B \quad (3)$$
$$C_b = 0.564(B - Y) + 128 \quad (4)$$
$$C_r = 0.713(R - Y) + 128 \quad (5)$$

Equations (3), (4) and (5) show the conversion of R, G and B channels into Y, C_b and C_r channels for an 8-bit image.
The next step is to apply the forward 2-D DCT to each 8x8 sub-block. After this we perform quantization, this will help in removal of high frequency data which is undesirable. Since the process is lossy, it will conceal the adjustments made to the frequency content even further. This step involves the division of each element in the 8x8 sub-blocks with the elements in a quantization table and rounding to the nearest integer. The quantization table is chosen according to the required quality of the image. For the purpose of our research, we have chosen a standard quantization table shown in Figure 2.

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Figure 2: Standard Quantization Table

Quantization will produce a sparse matrix which can be compressed by huffman coding or run length encoding. But in our method, we will only use the result to embed the bits of the secret message into the DC coefficient. This is done by Least Significant Bit substitution. To understand how it works, let us take a character 'a' which is 97 in ASCII(American Standard Code for Information Interchange). Therefore it's binary representation will be 01100001. We will now logical AND each bit of the message to the last bit of every DC coefficient we come across. Suppose the coefficient's for eight 8x8 blocks are 221 , 302, -215, 233, -192, 240, 200 and -202. After LSB substitution the values would become 220, 303, -215, 232, -192, 240, 200 and -201. The next step is dequantization, which is basically the multiplication of every 8x8 block with the quantization table. After this, we apply the inverse 2-D DCT or IDCT to get the YC_bC_r values of the image. These values are converted back to R, G and B format using equations (6), (7) and (8) below.

$$R = Y + 1.403(C_r - 128) \quad (6)$$
$$G = Y - 0.714(C_r - 128) - 0.344(C_b - 128) \quad (7)$$
$$B = Y + 1.773(C_b - 128) \quad (8)$$

We have now obtained the resultant stego image from the embedding algortihm. Now, in order to retrieve the original message we will have to follow all the steps we used for our embedding algorithm up to the quantization stage. We check the DC coefficients of each 8x8 sub-blocks. The values are logically ANDed with the number 1. This will give the bits of the message. After every eight blocks, we convert the binary information into ascii to get the characters of the secret message. Once there is no more message left, we can display the ouptut to the user. Figures 3 and 4 show the embedding and extraction algorithm.

3.1 Embedding

1. Read the input color cover image.
2. If the dimensions of image is not a multiple of 8, make it multiples of 8 and pad the image.
3. Convert the image matrix to Y C_b C_r color space.
4. Create a new array and have its shape equal to the cover image.
5. Read the secret message and prepend its length to it. So, for example: 'abcd' would become '4abcd'. Convert the message to binary.
6. Convert the padded image matrix to 8x8 blocks.
7. Apply 2-D DCT on each block.
8. Each block is quantized using the standard JPEG Quantization Table.
9. Embed each bit of the message to the LSB of each DC coefficient.
10. Dequantize each sub-block using the quantization table.
11. Store the obtained matrix to each channel of the new array in step 3.
12. Go back to step 7 and repeat the process for all three channels.
13. Convert the array to RGB.
14. This array is the new stego Image.

3.2 Extraction

1. Read the stego image
2. Convert to Y C_b C_r and store in a new array.
3. Divide the image matrix into 8x8 sub-blocks.
4. Apply 2-D DCT and quantization on each sub-block.
5. Read the DC coefficients of each sub-block. Check if the bits form numeric values. If so, store and append the value obtained into a new variable called L.
6. Once there are no numeric values, break out of the loop. L will be the length of the message.
7. Now, check the DC coefficients of rest of the sub-blocks for the message storing and appending to another new variable M. This will store the actual message.
8. Once the length of M equals L. Break out of the loop and display the message to the user.

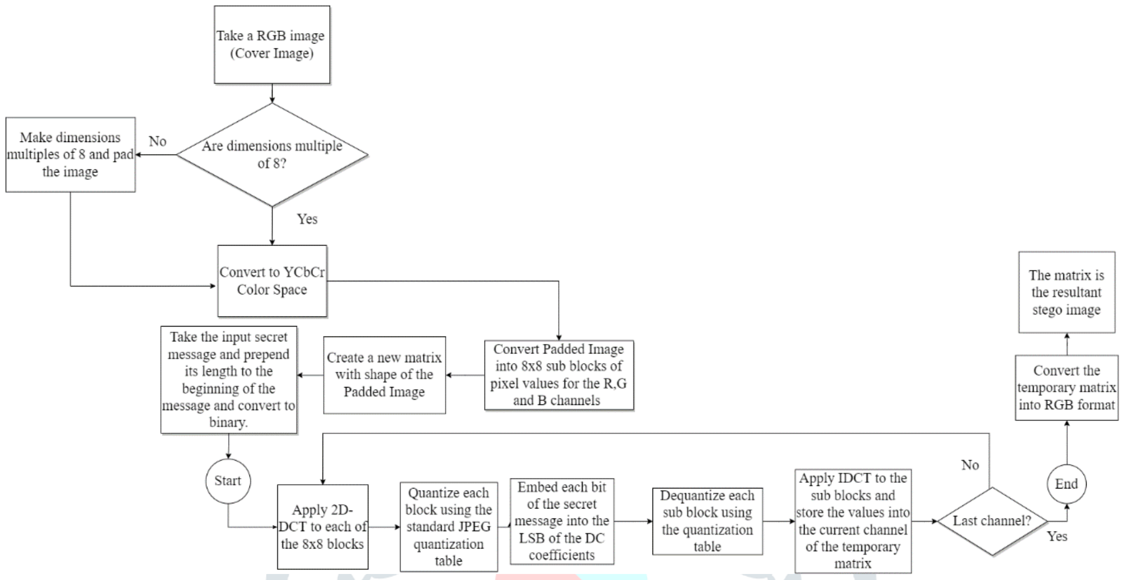

Take a RGB image (Cover Image)
Are dimensions multiple of 8?
No
Make dimensions multiples of 8 and pad the image
Yes
Convert to YCbCr Color Space
Convert Padded Image into 8x8 sub blocks of pixel values for the R,G and B channels
Create a new matrix with shape of the Padded Image
Take the input secret message and prepend its length to the beginning of the message and convert to binary.
Start
Apply 2D-DCT to each of the 8x8 blocks
Quantize each block using the standard JPEG quantization table
Embed each bit of the secret message into the LSB of the DC coefficients
Dequantize each sub block using the quantization table
Apply IDCT to the sub blocks and store the values into the current channel of the temporary matrix
Last channel?
No
Yes
End
Convert the temporary matrix into RGB format
The matrix is the resultant stego image

Figure 3 : Flowchart of Embedding Algorithm

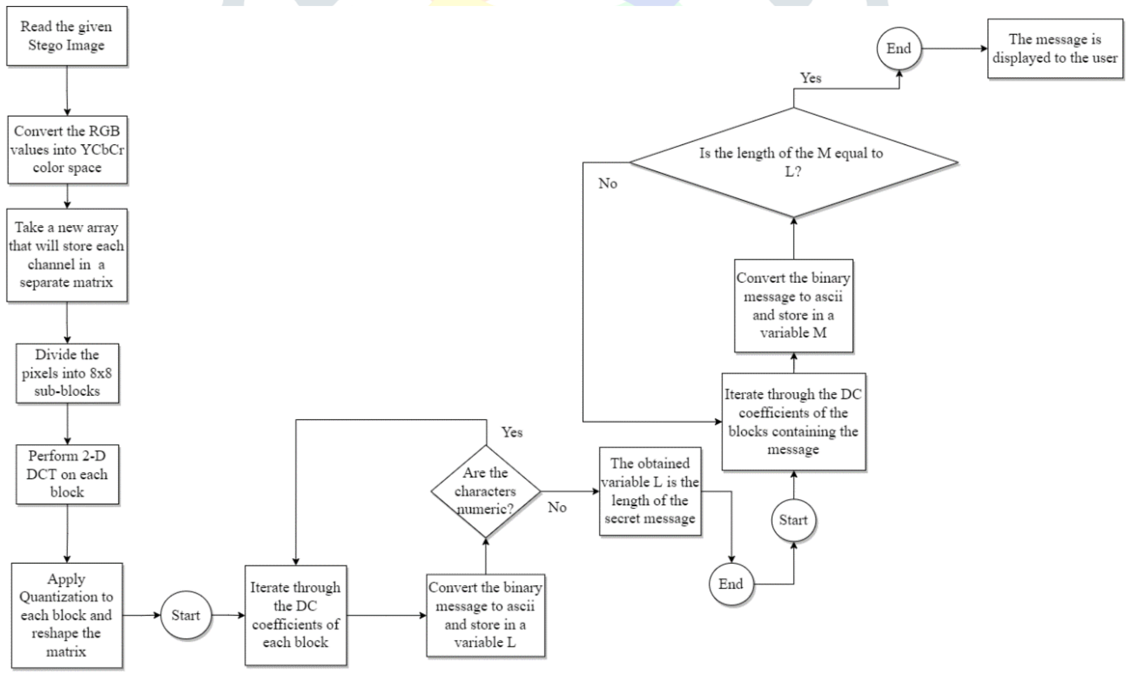
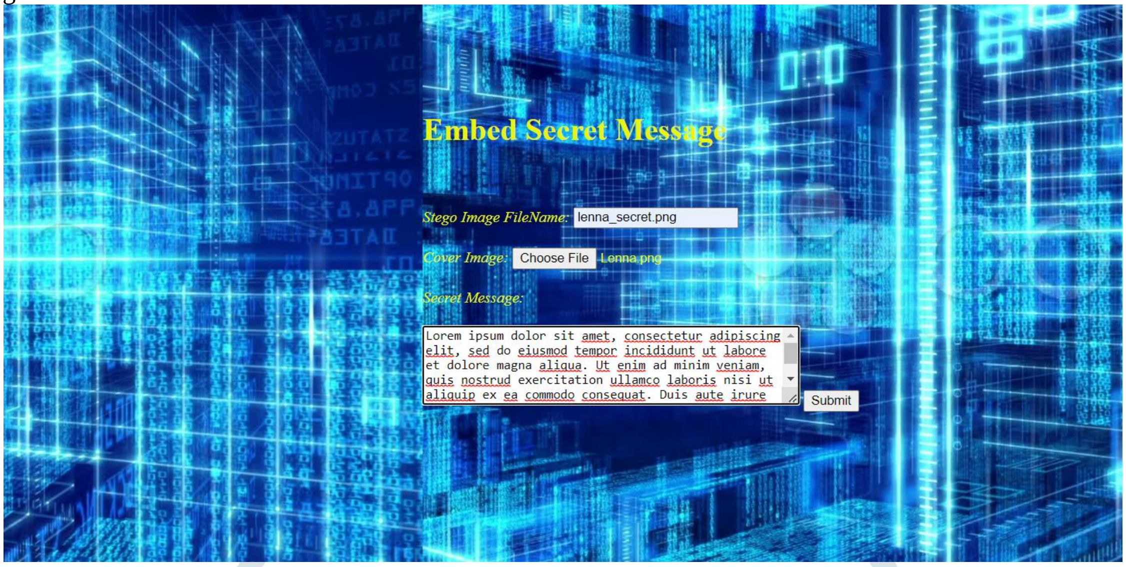

Read the given Stego Image
Convert the RGB values into YCbCr color space
Take a new array that will store each channel in a separate matrix
Divide the pixels into 8x8 sub-blocks
Perform 2-D DCT on each block
Apply Quantization to each block and reshape the matrix
Start
Iterate through the DC coefficients of each block
Convert the binary message to ascii and store in a variable L
Are the characters numeric?
Yes
No
The obtained variable L is the length of the secret message
End
Start
Iterate through the DC coefficients of the blocks containing the message
Convert the binary message to ascii and store in a variable M
Is the length of the M equal to L?
No
Yes
End
The message is displayed to the user

Figure 4 : Flowchart of Extraction Algorithm

IV. EXPERIMENTS

The embedding and extraction script were written using Python v3.10.0 and using the opencv library for image processing [14]. Additionally, we have created a web page using JavaScript, HTML and CSS to demonstrate the embedding and extraction algorithm on a test image as shown below.



Embed Secret Message

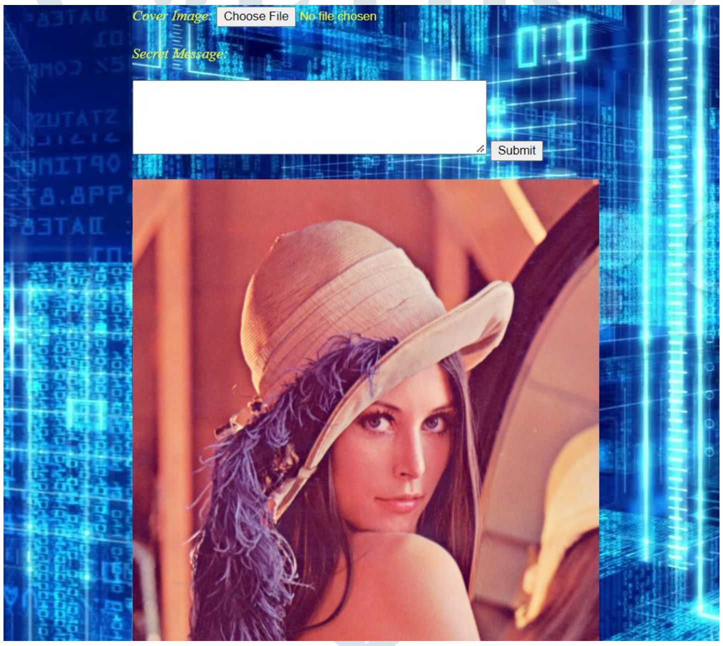
Stego Image FileName: lenna_secret.png

Cover Image: Lenna.png

Secret Message:

```
Lorem ipsum dolor sit amet, consectetur adipiscing
elit, sed do eiusmod tempor incididunt ut labore
et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo consequat. Duis aute irure
```

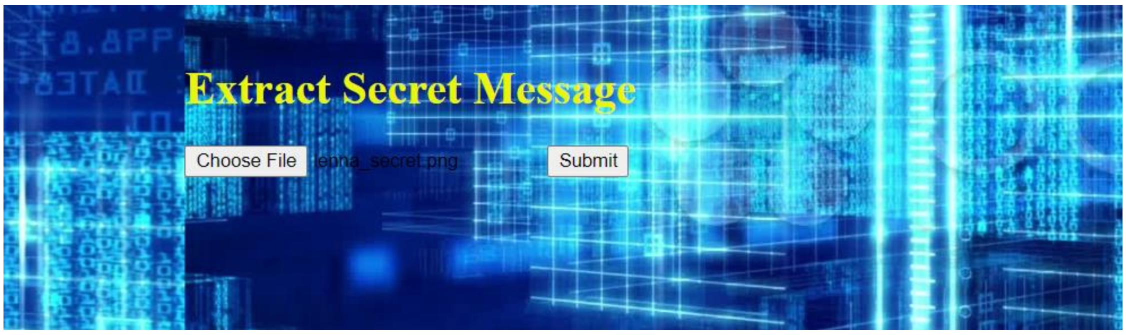
Figure 5: Embedding a secret message into the cover image



Cover Image: No file chosen

Secret Message:

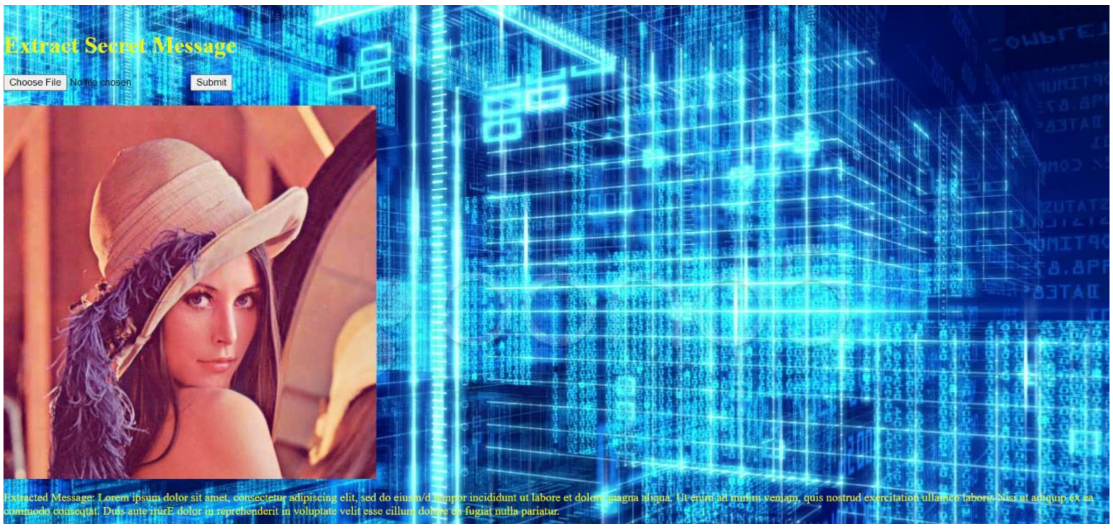
Figure 6: Cover Image



Extract Secret Message

lenna_secret.png

Figure 7 : Selection of Stego Image



Extracted Message: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusm/d tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris Nisi ut adiquip ex ea commodo conseqtat. Duis aute irurE dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Figure 8: Extraction of message from Stego Image

Figures 5 and 6 represent the web page where the embedding of secret message takes place. We choose a color cover image and upload it into the website. When the form is submitted, a stego image with the message is generated. Figures 7 and 8 represent the extraction web page. After choosing the stego image and submission of the form, the secret message is extracted from the image.

V. Results and Discussions

To test the quality of our proposed method we have used performance evaluation parameters like Mean Square Error (MSE), Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index (SSIM). Our system was experimented using MATLAB (version 2018a). The specifications of the machine used is a 2.11 GHz Intel Core i5 processor with 8 GB RAM. Additionally, we have done a comparative analysis with another algorithm (Edgar M., 2020) to test the time complexity.

5.1 Mean Square Error(MSE)

In simple terms, the mean square error (MSE) is the average of the square of the error. It is the cumulative square error between the cover and stego image. Let the dimensions of the image be M and N respectively. I and I' represent the cover image and stego image respectively. The formula for mean square error is given in equation (9).

$$\text{MSE} = \frac{1}{MN}\sum_0^{M-1}\sum_0^{M-1}(I'[N,M] - I[N,M])^2 \qquad (9)$$

5.2 Peak Signal to Noise Ratio (PSNR)

The peak-signal-to-noise ratio (PSNR) [15] is an expression for the ratio of between maximum possible value of a signal and the power of distorting noise that affects the quality of its representation. For a 24-bit color image the maximum value of a pixel is 255. Equation (10) show the calculation of PSNR.

$$\text{PSNR} = 10\log_{10}\left(\frac{255^2}{\text{MSE}}\right) \qquad (10)$$

5.3 Structural Similarity Index (SSIM)

The structural similarity index is used to measure the similarity between two images. While MSE and PSNR measure absolute errors, The premise behind structural information is that pixels have substantial interdependencies, especially when they are close together in space [16]. The SSIM formula is given by equation (11).

$$\text{SSIM(A,B)} = \frac{(2\mu_A\mu_B+c_1)(2\sigma_{AB}+c_2)}{(\mu_A{}^2+\mu_B{}^2+c_1)(\sigma_A{}^2+\sigma_B{}^2+c_2)} \qquad (11)$$

Where, μ_A and μ_B are the average of A and B, σ_A and σ_B is the variance of A and B, σ_{AB} is the covariance of A and B and c1 and c2, two variables to stabilize division with weak denominator. Figure 9 represents all the test images which were used for the experiment.

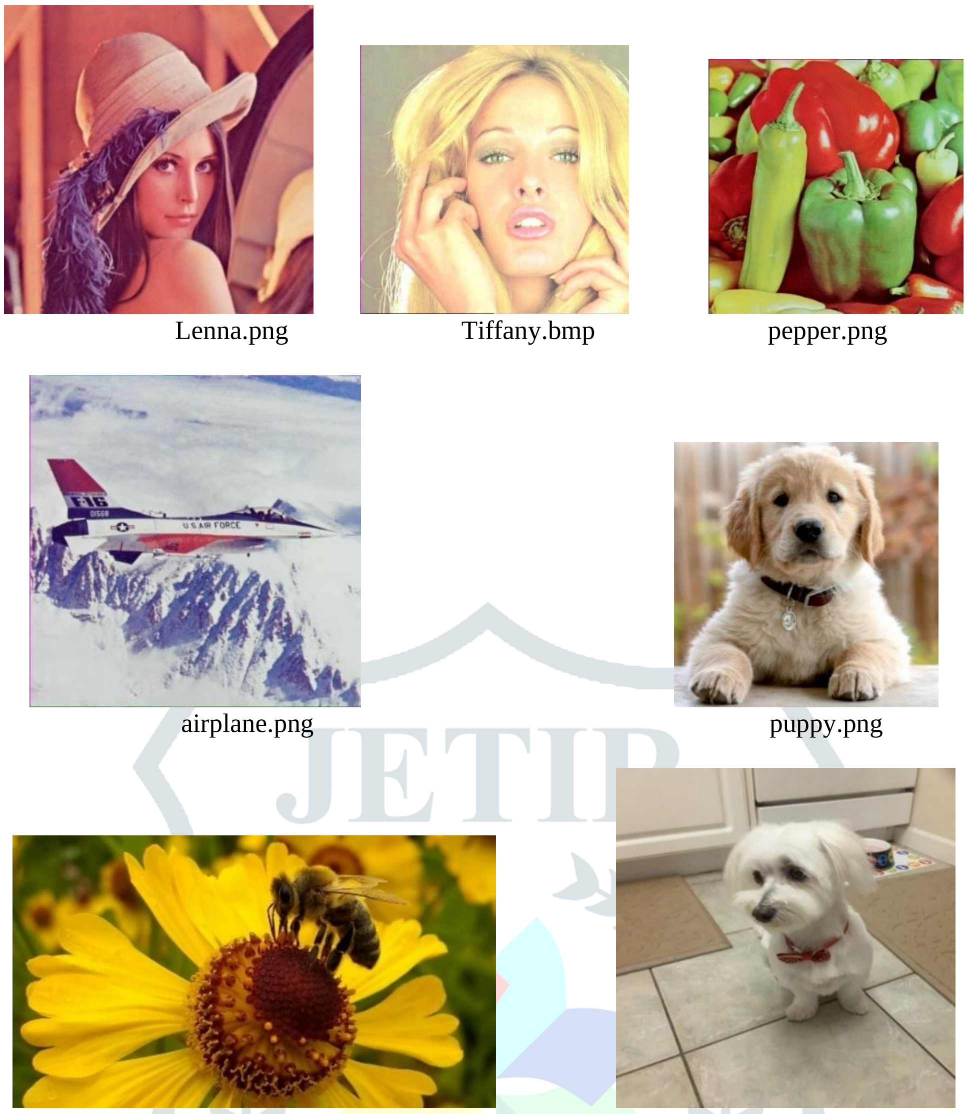


Figure 9: Test Images

Table 1 : PSNR and MSE values of stego images of given dimension

Stego Image	Image Dimensions (px)	MSE	PSNR(dB)	SSIM
Lenna.png	512x512	30.2947	33.3236	0.9871
Tifanny.bmp	512x512	31.1097	33.2018	0.9816
pepper.png	512x512	39.5768	32.1564	0.9841
airplane.png	512x512	23.5241	34.4157	0.94
puppy.png	1000x1000	14.3508	36.5620	0.9773
bee.png	1920x1080	6.0161	40.3377	0.9979
Maltese.bmp	2238x2246	6.0782	40.2931	0.9856

The average PSNR from table 1 of the 512x512 px images is 31.13 dB. We can observe that as the size of the image increases the MSE value decreases for the same payload capacity, which in turn increases the PSNR and is desirable. The mean SSIM is 0.979 which is very close to unity which confirms good quality of the stego images.
We have also generated the image histogram for two images with different dimensions, Lenna (512x512 pixels) and puppy (1000x1000 pixels). Histograms can be used during steganalysis for pixel-by-pixel comparison of the cover and stego images.

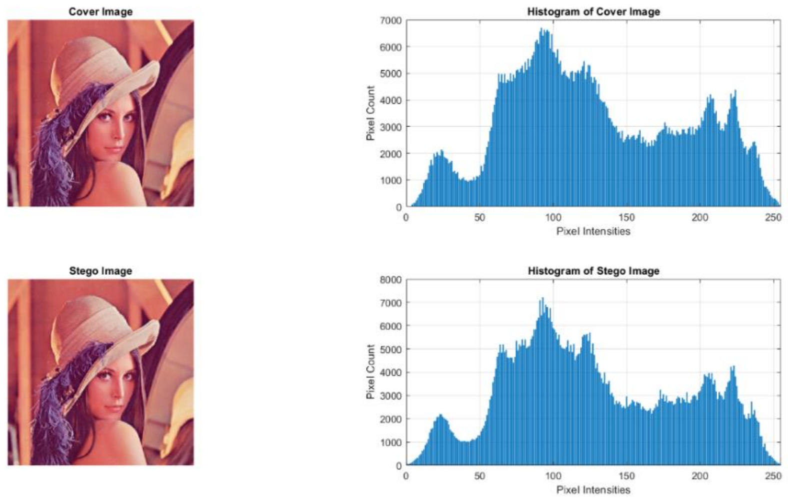


Figure 10: Image Histogram of Lenna.png

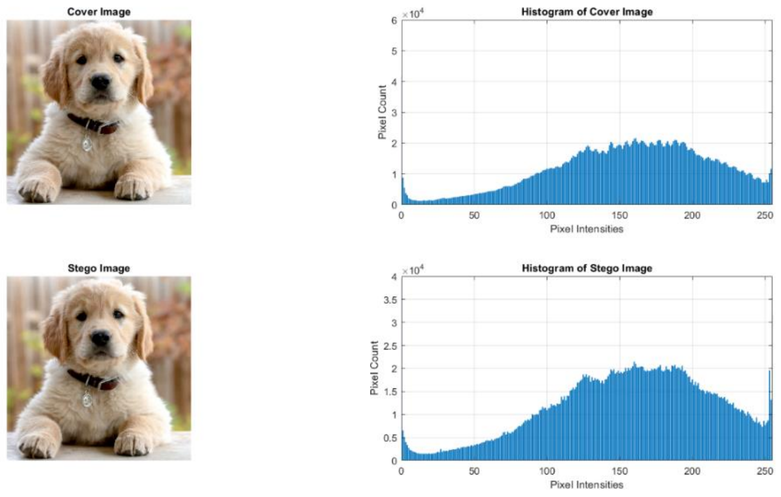


Figure 11: Image Histogram of puppy.png

In Figure 10, we can see some spikes in the pixel intensity around the range 50-150 with the peak pixel count a little above 7000 for the stego image. For Figure 11, the intensity compared to the cover image mostly remains the same but there is a sharp spike towards intensities beyond 250.

Table 2: Time Complexity of Embedding and Extraction for Proposed Algorithm

Stego Image	Image Dimensions (px)	Embedding (ms)	Extraction (ms)
Lenna.png	512x512	254.1	60.347
Tifanny.bmp	512x512	213.15	53.307
puppy.png	1000x1000	984.5	212.6
bee.png	1920x1080	1963.4	412.485
Maltese.bmp	2240x2240	5318.4	1776.9

Table 3: Time Complexity of Embedding and Extraction for Edgar's Algorithm

Stego Image	Image Dimensions (px)	Embedding (ms)	Extraction (ms)
Lenna.png	512x512	1064.4	761
Tifanny.bmp	512x512	976.7	788
puppy.png	1000x1000	4126	2313.4
bee.png	1920x1080	8458	5379
Maltese.bmp	2240x2240	22765	12910

Table 2 shows the embedding and extraction speed of our proposed method. The average embedding and extraction speed of 512x512 px images is 233.625 milliseconds and 56.827 milliseconds. As the size of the image increases, the time complexity also increases. The extraction rate is much faster as compared to the embedding rate. Table 3 shows time complexity of Edgar's algorithm for the same payload capacity. The embedding rate mean is 1020.55 milliseconds and the extraction rate mean is 774.5 milliseconds for images with 512x512 px dimensions. The rate suffers greatly with larger dimensions of the images. Our method is a vast improvement to Edgar's method with a lower time complexity.

5.4 Drawbacks of the proposed algorithm

There are a few limitations to our proposed method. One glaring problem is that this technique is not applicable with JPEG images as they have their own DCT, quantization process which will destroy the embedded message. The difference between the image histograms of the cover and stego image is quite significant, which can be used to detect the message during steganalysis. Only DC coefficients are being used to embed the message bits, which reduce the payload capacity by quite a bit. On a few test cases with images with a large number of repeating frequencies (pixels), the extraction failed to retrieve the message even though the embedding algorithm worked and a stego image was generated. Upon closer examination, the reason for failure was due to erroneous DC coefficients obtained during the quantization stage of the extraction algorithm.

VI. References

[1] Hussain, M., & Hussain, M. (2013). A Survey of Image Steganography Techniques.
[2] Rahmani, M. K., Goyal, K. A., & Mugdal, M. (2015, August 8). Study of Cryptography and Steganography System. International Journal Of Engineering And Computer Science, 4(8), 13685-13687.
[3] Deekshita, B., Gnanamanjari S., & Chaithanya, S. (2017). SURVEY ON DIFFERENT METHODS OF IMAGE STEGANOGRAPHY. International Journal of Advance Research and Innovative Ideas in Education, 2(5), 234-238.
[4] Morkel, T., Eloff, J., & Oliver, M. (2005). An overview of Image Steganography. Proceedings of Information Security South Africa (ISSA) Conference.
[5] Bhattacharya, S., Khan, A., & Sanyal, G. (2014, February). DCT Difference Modulation(DCTDM) Image Steganography. International Journal of Information & Network Security, 3(1), 40-63.
[6] Lossy Data Compression JPEG. (n.d.). Retrieved 2022, from
https://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossy/jpeg/dct.htm
[7] Singh, A. K., Singh, J., & Singh, V. H. (2015, January). Steganography in Images Using LSB Technique. International Journal of Latest Trends in Engineering and Technology, 5(1), 426-430.
[8] Sari, R. D., & Siahaan, A. P. (2018, October). Least Significant Bit Comparison between 1-bit and 2-bit Insertion. International Journal For Innovative Research in MultiDisciplinary Field, 4(10), 110-113.
[9] Saxena, K. A., Sinha, S., & Shukla, P. (2018, April). Design and Development of Image Security Technique by Using Cryptography and Steganography: A Combine Approach. International Journal of Image, Graphics and Signal Processing, 10(4), 13-21.
[10] Upham, D. (1993). Steganographic algorithm JSteg. Retrieved from https://zooid.org/~paul/crypto/jsteg/
[11] Edgar, M. (2020). Python Implementation of Image Steganography. Retrieved from Github:
https://github.com/MasonEdgar/DCT-Image-Steganography/blob/main/Image_Steganography_Report.pdf
[12] AbdelWahab, O. F., Hussein, I. A., Hamed, F. H., Kelash, M. H., Khalaf, A. A., & Ali, M. H. (2019, June). Hiding data in images using steganography techniques with compression algorithms. TELKOMNIKA (Telecommunication Computing Electronics And Control), 17(3), 1168-1175.
[13] Saroj, A., & Banu, S. S. (2018, April). An Improved Technique for Hiding Secret Image on Colour Images. International Research Journal of Engineering and Technology, 5(4), 1120-1124.
[14] Home - OpenCV. (2022). Retrieved from https://opencv.org/
[15] Peak Signal-to-Noise Ratio as an Image Quality Metric. (2016). Retrieved from https://www.ni.com/en-in/innovations/white-papers/11/peak-signal-to-noise-ratio-as-an-image-quality-metric.html
[16] Structural Similarity. (2022). Retrieved from WikiPedia: https://en.wikipedia.org/wiki/Structural_similarity