A New Steganography Technique using JPEG Images

Rand A. Watheq, Fadi Almasalha, and Mahmoud H. Qutqut
Faculty of Information Technology
Applied Science Private University
Amman 11931 Jordan

***Abstract*—Steganography is a form of security technique that using ambiguity to hide a secret message within an ordinary message between senders and receivers. In this paper, we propose a new steganography technique for hiding data in Joint Photographic Experts Group (JPEG) images as it is the most known type of image compression between the lossy type compressions. Our proposed work is based on lossy compression (frequency domain) in images. This type of compression is susceptible to change even for the smallest amount of change which raises a difficulty to find a proper location to embed data. This should be done without affecting the image quality and without allowing anyone to notice the hidden message. From the senders side, first, we divide the image into 8*8 blocks, then apply a Discrete Cosine Transform (DCT), Quantization, and zigzag processes respectively. Second, the secret message is embedded at the end of each selected zigzag block array using the best method of our experimental results. Third, the rest of the code applies the Run Length Code (RLC), Different Pulse Code Modularity (DPCM) and Huffman encoder to obtain the compressed image that includes the embedded message. From the receiver's side, we will reverse the previous steps to extract the secret message using an encrypted shared key via a secure channel. Our experimental results show that the best array content size of zigzag computed coefficients are between 1 to 20. This selection allows us to utilize more than half of the image blocks to embed the secret message and the difference between the cover image that holds the secret message and the original cover image is very minimal and hard to detect.**

Keywords*—*Steganography; hide secret message; JPEG image; lossy compression; frequency domain; zigzag

I. Introduction

We live in the era of technology, which involves in growing use of the Internet; so we need a secure transfer for a secret message through the Internet. Data hiding techniques have lately become important in several application fields. The importance of data hiding techniques stems from the transmission medium is being not secure. Hence, some methods are needed to make it difficult for the unauthorized user to extract information from the images. The primary motivations to hide data are to protect personal, sensitive and private confidential data [1].

A data hiding technique is an innovative type of secret communication technologies. It is a form of embedding data into digital media (image, video, audio, text) with a minimum amount of perceivable degradation to the host signal, for the aim of identification, control access to digital media and copyright. The techniques used for data hiding differ depending on two factors; the quantity of data needs to be hidden, and the required invariability of these data to manipulation [2]. Any loophole to fill data in a host signal, either statistical or perceptual are possible targets for removal by lossy signal compression [2]. The answer to a successful data hiding is to find holes that are suitable for utilization by compression algorithms. There are different techniques of data hiding like watermarking, steganography, and cryptography.

To this end, we propose a new steganography technique for hiding data in JPEG images. Our proposed work is based on lossy compression (frequency domain) images; which raise a difficulty to find a proper location to embed data without affecting image quality and without allowing anyone to notice the hidden message. Our work focuses on how to embed and extract secret data without affecting the original image. After a comprehensive study of the JPEG encoding process, it was found that the best location to embed data is in a zigzag order based on several experiments. From the senders side, the work of this paper firstly divides the image into 8*8 blocks. Then, we apply the Discrete Cosine Transform (DCT), Quantization, and zigzag processes in order to calculate the required blocks for the secret message. After that, the process of embedding the data in the selected blocks will start based on a specific range. The rest of the code applies the Run Length Code (RLC), Differential Pulse Code Modulation (DPCM) and Huffman encoder as before to obtain the compressed image that includes the embedded message. During the previous process, the secret message is embedded at the end of each selected zigzag block array using the best method of our experimental results. The sender sends the compressed image to a communication channel, and the receiver will reverse the previous steps to extract the hidden data using a Shuffled Block Candidate Array (SBCA) array that has been sent via a secure channel. However, the receiver can use the shared private key to generate the same SCBA generated on the sender. Sharing the private keys allows both sender and receiver to exchange messages without the need to send the SCBA of each embedding process. In this paper, we choose to share the SCBA array for the sake of simplicity and to focus more on the proposed embedding technique.

The remainder of this paper is organized as follows. In Section II, we overview the background topics of steganography and image compression. Section III overviews related work of our paper. In Section IV, we introduce and describe our proposed technique and its procedures. Our experiments and results are presented in Section V. Section V describes the conclusion of the paper.

II. Background

In this section, we provide background topics of steganography and image compression.

A. Steganography

The word steganography derived from the Greek language and means ”covered writing” [1]. Steganography is a science of hiding message, file, image or other data types within another data file, in a way to prevent anyone to know if there is message hiding in the original message. Fig. 1 shows the steganography steps. In the following, we describe the different types of steganography.

1) Steganography based on Protocols: There are three main types of steganographic protocols as shown below [3].

- Pure Steganography: This type assumed the sender and receiver must have access to the process of embedding and extraction.
- Secret Key Steganography: In this type, the sender and the receiver use the same key to hide and extract the secret message in a cover media.
- Public Key Steganography: The sender and the receiver use the different key to hide and extract the secret message in a cover media.

2) Steganography based on Cover Media: There are five types of Steganography based on the carrier object that is used for hiding the secret data. These types are Audio Steganography, Text Steganography, Image Steganography, Video Steganography, and Network Steganography [4].

3) Steganography based on Domain: Steganography techniques can be divided into two domain types as described below.

- Spatial Domain: In this type, the secret message bits are embedded in a cover image by directly changing the pixel’s color values. An example of this type is the Least Significant Bit (LSB) technique.
- Frequency Domain: In this type of techniques is tries to embed the secret message bits in the frequency domain coefficient of the cover image. An example of this is the Discrete Cosine Transform (DCT) technique.

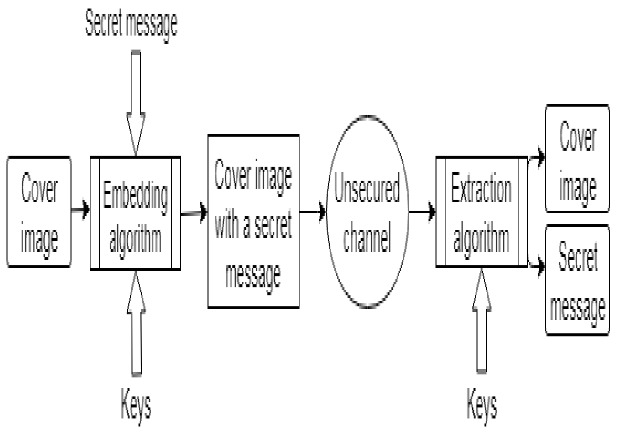


Fig. 1. Steganography steps overview (reproduced from [4]).

B. Image Compression

Image compression is the science/art of efficiently encoding digital images to minimize the number of bits that are required to represent an image [5]. Image compression divided into two types as follows [6].

- Lossy Compression: The concept of this type is that when data is compressed, it loses a portion of itself. Therefore, there became a slight difference between the image before compression and the image after decompression. In this type of compression, the amount of loss can be adjusted to achieve the desired compression ratio. Examples of this type are Joint Photographic Experts Group (JPEG), Moving Picture Experts Group (MPEG), and Moving Picture Experts Group Layer-3 Audio (MP3).
- Lossless Compression: The concept of this type is that when the data is compressed, it does not lose any data. In other words, the reconstructed data after the compression is identical to original data. The most common techniques of this type are Huffman coding, RLC, and Lempel–Ziv–Welch (LZW).

An image compression process contains two phases [7]; encoder and decoder.

1) Encoder: In this phase, we take an original image and apply some steps to obtain a compressed image. It contains multi-steps as shown in Fig. 2.
2) Decoder: In this phase, we use the result from the encoder phase and apply inverse the steps (the steps that applied in the encoder phase) to obtain a decompressed image. It contains multi-steps as shown in Fig. 3.

III. Related Work

There are many research work on data hiding techniques in multimedia data. We overview the latest research work related to our paper in this section. H. Lu *et al.* propose an algorithm for binary images that can embed a watermark in DC component [8]. They combine the embedding watermarks in the DC components of DCT and employing a biased binarization threshold. The results show that the embedding algorithm provides some degree of robustness against conventional image processing. M. Kaur *et al.* [9] propose a method using two watermarks in a cover image. They embed the first watermark in the middle frequency of the blue component. The second watermark is embedded into magnitude coefficients of the Discrete Fourier transform (DFT) in the form of local peaks. They concluded that the Red Green Blue (RGB) model is more suitable in case of repeating watermark than the YCbCr, and the blue component is more suitable for embedding the watermark. The work in [10] proposes a method using a secret data and Laplacian sharpening method. The author compresses the secret data using Huffman coding. After that, she embeds it in a cover image using Laplacian sharpening method to determine the useful hiding places based on a threshold value. S. Sujatha *et al.* [11] propose a method to generate watermark from the original image. First, they used low frequency, a rescaled version with the help of Arnold transform to generate a watermark. Second, they used a high frequency of Discrete Wavelet Transform (DWT) to embed the watermark. This method provides good robust against attacks such as JPEG compression, scaling and rotation.

R. Preda proposes a method for image authentication using a semi-fragile watermark in [12]. In this method, the author used a semi-fragile watermark to detect malicious tampering in the image and embedded a bit of watermark in the coefficient by means quantization. He selected wavelet coefficient with random permutation using a secret key to provide a higher level of security. This method achieved decent image quality and tampering detection resolution with a low watermark payload. In work [13], the authors propose a method to embed a binary watermark in a compressed color image. They embedded a binary watermark in middle bands of (Y) luminance. This method provided good watermark in colored images. M. Khan *et al.* [14] propose a method that combined between DWT and DCT. They applied DWT to an original image after that applied DCT to High-High (HH) band to obtain matrix H. They converted H matrix into four quadrants using zigzag. Finally, they applied Singular Value Decomposition (SVD) in each quadrant to embed the watermark. In research work [15], the authors propose a method using DCT. They applied DCT to a middle frequency of B plane and selected DCT (4, 3), DCT (5, 2) to embed the watermark. This method provides fair robust against different types of attack. M. Mundher *et al.* [16] they propose a method using the preprocessing stage and Discrete Slantlet Transform (DST). They used a preprocessing stage to find the best channel. After that applies DST to the selected quadrant to find the best frequency sub-band to embed the watermark in the best frequency. This method employed to ensure the imperceptibility and robustness of watermarked images.

In the research work [17], the authors propose a method

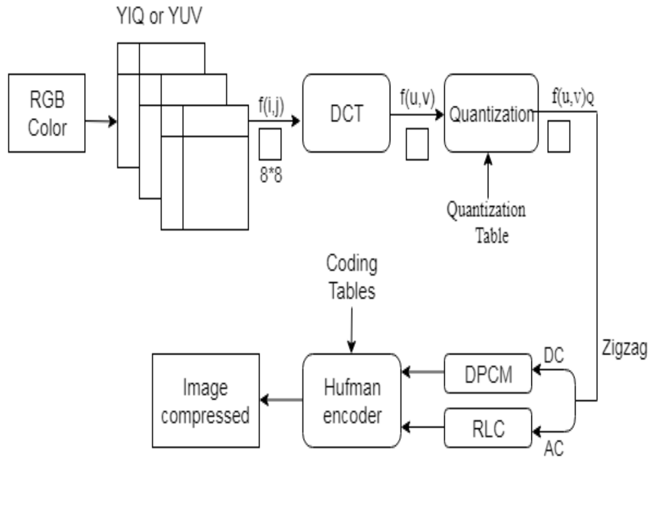


Fig. 2. JPEG Encoder Compression (reproduced from [7]).

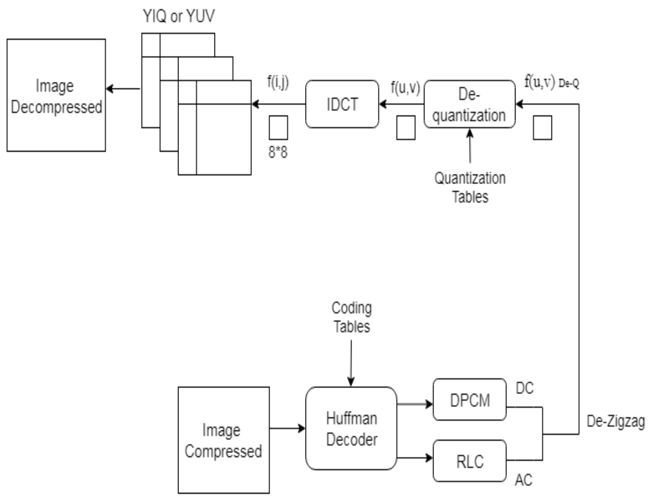


Fig. 3. JPEG Decoder Compression (reproduced from [7]).

using DWT in Hue, Saturation, and Intensity (HSI) color space. First, selecting I plane from the original image (in HSI form). Then, they apply DWT to it and select low-frequency. Second, dividing the watermark image into 8*8 blocks. Finally, comparing both images based on entropy values, then multiply the scaling factor. This method provides fair, robust watermark to noise attacks. In [18], the authors propose a method using 3D chaotic cat map, DWT and lifting scheme. The irregular output of the chaotic cat map is used to embed the secret message in a cover image. Then, they embedded the secret message in the mean coefficient of DWT. Finally, they applied lifted scheme to guarantee lossless extraction of hidden information. The method provides good performance and imperceptibility according to two measures (Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM)). The work in [5] proposes a method called Compressed Encrypted and Embedded Technique (CEET). In this method, they compressed a secret image using JPEG compression to reduce the size of a secret image. After that, they selected key randomly and applied encryption to the secret image to increase security. Finally, they embedded a secret image in a cover image using LSB by inserting at least 2 bits. This method collected different techniques to reach the goal of steganography which is embedding highest possible rate while remaining undetectable to steganalysis. J. Mazumder *et al.* [19] propose a method using DWT and optimized message dispersing. They used a high frequency of all color components (R, G, B) to embed a secret message. They start from the last column of each of the components from top to bottom based on the length of the message. Finally, they used Mean Squared Error (MSE) and PSNR to measure the imperceptibility of the method that shows it is acceptable compared to other methods.

G. Swain in [20] proposed a method using the LSB but in new method Group of Bits Substitution (GBS). Firstly, they appends length of message in the beginning of the binary message, after that embedding one bit of the secret message in the cover image if size of cover image $\geq$ size of secret message or embedding two bits of secret message in one byte of the cover image if size of cover image size of

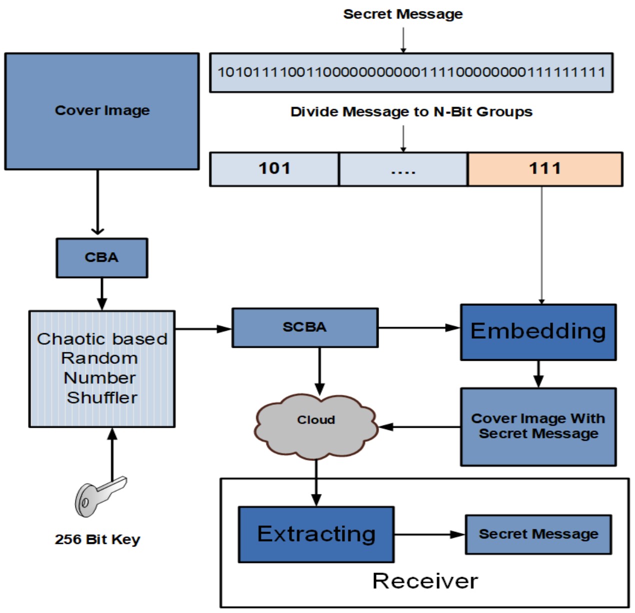


Fig. 4. Overview of Our Proposed Technique.

secret message divided by 2. Finally, this method provides the security to a higher level. In [21], the authors propose a method using ”LSB substitution” and ”Arnold transform”. They used ”Arnold transform” to encrypt the secret images using different keys. Then, embedding the first Most Significant Bit (MSB) (in secret image 1) in last three LSB once using red pixel of the cover image, again using green pixel and a blue pixel. Results reveal that the method successfully secures the high capacity data keeping the visual quality of transmitted image satisfactory. S. Kumar *et al.* [22] propose a method using index based on a chaotic mapping. They used a chaotic map to generate pseudo-random numbers (index). The index is used as a position to embed a message in a cover image, and they used LSB substitution to embed bit pixels in the cover image. In [23], the authors propose a system using three methods Huffman, zigzag and Pptimal Pixel Adjustment Process (OPAP). They used Huffman to compress the secret message to reduce the size of the secret message and to provide high embedding capacity then, used Zigzag scanning to select the pixels that the secret message be hidden in them and used OPAP to enhance the quality of the Stego-images to keep minimizing embedding error.

We test the effect of modification in the zigzag order using three methods then selected the best method that was adopted in this paper to hide data; the selected method provides better security with high embedding capacity and a better Stego-image quality than the existing system.

IV. Our Proposed Technique

The visual quality of Stego-image (imperceptibility), the security level (robustness) and embedding capacity are three basic principles that are used to evaluate the performance of the steganographic scheme. This section presents our proposed solution illustrated in Fig. 4, which provides a new method for embedding and extracting secret messages in zigzag order using JPEG image format. Our proposed technique consists

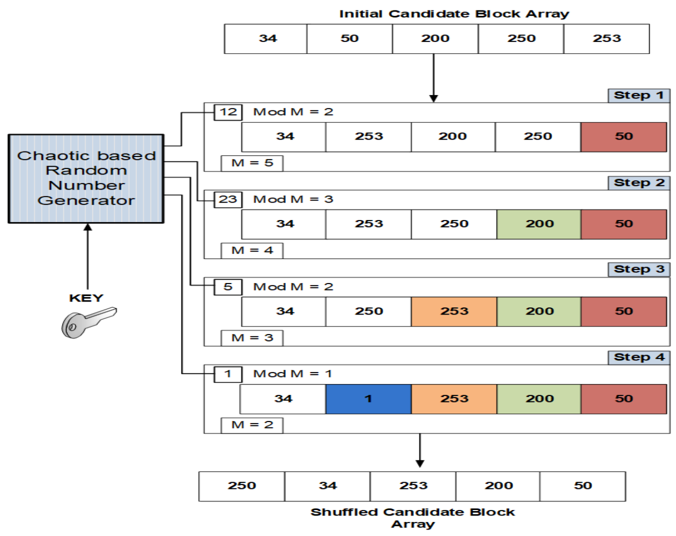


Fig. 5. Random number generation phase (reproduced from [24]).

Algorithm 1 Generating Shuffled Candidate Block Array

Input:
Cover Image M.
Private key K.
Zigzag array threshold value ZT

```
1:  Create empty Candidate Block Array (CBA).
2:  Create empty array Shuffled Candidate Block Array (SCBA).
3:  Get No of Micro Blocks N from cover Image M.
4:  for each B of N Blocks do
5:      Retrieve Zigzag Array Z of B Block.
6:      if Index of last Coefficient value of Z >= ZT then
7:          Save the location of this block into CBA.
8:      else
9:          Skip it
10:     end if
11: end for
12: SCBA = PRNG_Shuffle(CBA) [24].
```

Output: Shuffled Candidate Block Array (SCBA)

three phases; namely, **1)** Random Number Generator phase, **2)** Generating the Shuffling Array phase, **3)** Embedding or Extracting Secret message in cover image phase. In the following subsections, we will describe our proposed technique phases in details.

Phase 1: Random Number Generator

A Random Number Generator (RNG) is a computational or physical device or a piece of software code which is designed to generate a sequence of numbers or symbols that cannot be reasonably predicted better than by a random chance. The input range for generating every random number depends on the bit length of the keys and nature of data and operation to be performed on data. We use a Chaotic based random number generator proposed by [24] to shuffle the secret message location within the cover image. We use a 256 key to increase the security level.

Phase 2: *Shuffle Array Generation*

In this phase, we generate an array to represents the secret message in order to build a corresponding location inside the cover image. Firstly, the secret message is converted to a sequence of ASCII codes. Then, we store the ASCII code into a binary representation. The size of the latter array is divided by 3 bits to calculate the required Number of image micro-Blocks (NOB). Second, the cover image is decoded to find the locations of the micro-blocks that passes the validity condition which will be discussed in later sections. Third, we initiate the Candidate Block Array (CBA) array (where its size equals to NOB) that contains the micro-blocks locations obtained from the previous step. Finally, we pass the CBA array into the Pseudo-Random Number Generator (PRNG) to obtain the Shuffled Candidate Block Array (SCBA). Fig. 5 illustrates the process of shuffling the CBA.

Algorithm 1 represents the generation of the SCBA. The algorithm receives the cover image, secret key and a threshold value of the maximum number of non zero coefficients in the zigzag array. Steps 1 to 3 initialize the arrays CBA and SCBA. Steps 4 to 11 iterate through each block to examine the threshold value condition. After step 11, the CBA will contain the sequence location of each zigzag array that passes the threshold condition which selected to hold the secret message. In step 12, the CBA array is passed to the proposed shuffler in order to shuffle the location. Step 12 will randomize the location of the data which makes the possibility of assembling the secret message very hard without the availability of the SCBA.

Example: Generating SCBA for the secret message ”Computer” using the image of Fig. 6.

- Cover image: Lena
- Secret massage: Computer
- Data array: 01000011 01101111 01101101 01110000 01110101 01110100 01100101 01110010 00100000
- Total blocks in image: (512*3*512)/64=12288 blocks.
- Using blocks that contain values between 1 to 20
- NOB = 24
- Initialize CBA
- CBA = {1,2,4,7,10,13,16,19,21,22,23,28,29,31,34, 37,40,43,46,49,52,55,58,61}
- SCBA = {13,21,55,52,31,22,2,43,49,46,61,4,19, 16,7,34,23,37,40,29,1,10,28,58}

Phase 3: *Embedding secret message in JPEG Image*

This phase works to embed the secret message in the cover image; without affecting the compressed image. This phase is delicate as JPEG image format (lossy compression), and the frequency domain is complicated and hard to find a location to embed the secret message without affecting the compression process. We present in this paper two different proposed methods to be evaluated based on the following two targets before giving the final algorithm.

Fig. 6. Lena Cover Image

1) We need to find the maximal number of bits to be embedded at the end of each selected blocks in the cover image. Our experiments include 2 bits, 3 bits, and 4 bits chunks.
2) We need to find how to add the data chunks to the last value in the selected block in a way that guarantee minimal effects on the encoding process.

In order to achieve our goals, we proposed a set of experiments using two different methods of adding the data chunks at the end of the zigzag array. First, it worthy to explain the structure of the zigzag order to better understand the proposed methods. As we see in Fig. 7, the 2d array of the calculated coefficients is converted into a 1D array in a way that accumulates the zero value coefficient at the end of the array. This structure allows us to add our data chunks after the last non-zero coefficient value without affecting the encoding or decoding process. Now, based on the structure of the zigzag order we proposed the following two methods in order to find the best way to add our data chunks without affecting the final compressed image and obtaining the most available space to hide the data in.

1) First Method: The first method puts the secret data after the last non-zero coefficient using three data chunk sizes as follows.

1) We divide the binary form of the secret message into a group of 2 bits and represent each group with a decimal value of -1, 0, 1, and 2. The corresponding

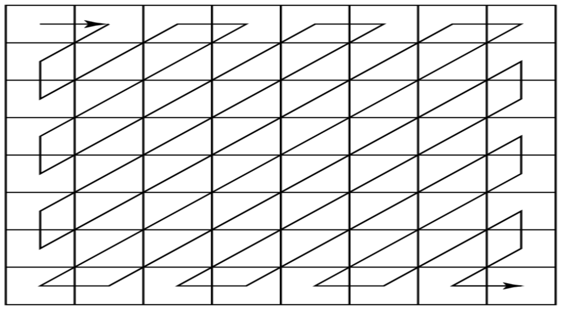


Fig. 7. Zigzag scan process in JPEG [7].

decimal values are then embedded after the last value of the non-zero coefficient.

2) We divide the binary form of the secret message into a group of 3 bits and represent each group with a decimal value from -4 to 4 excluding the zero. The corresponding decimal values are then embedded after the last value of the non-zero coefficient.
3) We divide the binary form of the secret message into a group of 4 bits and represent each group with a decimal value from -8 to 8 excluding the zero. The corresponding decimal values are then embedded after the last value of the non-zero coefficient.

2) Second Method: The second method adds the value of the data chunk to the last non-zero coefficient value also using the same previous three data chunk sizes as follows.

1) We divide the binary form of the secret message into a group of 2 bits and represent each group with a decimal value of -1, 0, 1, and 2. The corresponding decimal values are then added to the value of the last value of the non-zero coefficient.
2) We divide the binary form of the secret message into a group of 3 bits and represent each group with a decimal value from -4 to 4 excluding the zero. The corresponding decimal values are then added to the value of the non-zero coefficient.
3) We divide the binary form of the secret message into a group of 4 bits and represent each group with a decimal value from -8 to 8 excluding the zero. The corresponding decimal values are then added to the value of the non-zero coefficient.

After the evaluation process explained in Section V, we found that the second case in the first method is the best approach in term of minimal effect on the final compressed image and the larger space to hide data.

Algorithm 2 shows the process of embedding the secret message in a cover image while Algorithm 3 shows the steps of extracting the secret message.

At this point, an example is always good to illustrate the process of our proposed algorithms. The following example is divided into two parts. The first part will illustrate the process of embedding a secret message. The second part illustrates the process of extracting the same message from the cover image.

Algorithm 2 Embedding secret message in a cover image.

```
Input:
Cover Image M.
Hidden data size in each block N
Secret Message SM.
Shuffled Candidate Block Array(SCBA)
1: Convert SM into Binary Representation B
2: Initialize counter to Zero
3: for each D of size N from B do
4:     Location = SCBA[counter].
5:     Retrieve Zigzag Array Z of M[Location].
6:     Insert the D value after the last value in Z.
7: end for
Output: Steganographic Image
```

*A) Embedding the secret data in the 8*8 block pointed with a rectangle in Fig. 8:* .

The secret message in this example is ”OLA”. The compression algorithm will generate the quantized DCT coefficients shown in Fig. 9 A of the selected blocks. The later 2D array is then converted into a 1D array using the zigzag order as shown in Fig. 9 B.

The binary representation of our secret message is:
O = (01101111)
L = (01101100)
A = (01100001)

The groups of binary representation using the 3 bits data size chunks will be as follow:
Group 1 = 000
Group 2 = 001
Group 3 = 010
Group 4 = 011
Group 5 = 100
Group 6 = 101
Group 7 = 110
Group 8 = 111

The previous group is mapped to each decimal values as follows:
Group 1 = -4
Group 2 = -3

Algorithm 3 Extracting secret message in a cover image.

```
Input:
Steganographic Image M
Shuffled Candidate Block Array(SCBA)

1: Initialize B array
2: Initialize counter to Zero
3: for each I from SCBA do
4:     Retrieve Zigzag Array Z of M[I].
5:     Extract the D value from the last value in Z.
6:     Insert D to B
7: end for
8: Convert B from Binary to ASCII representation SM
Output: Secret Message (SM)
```

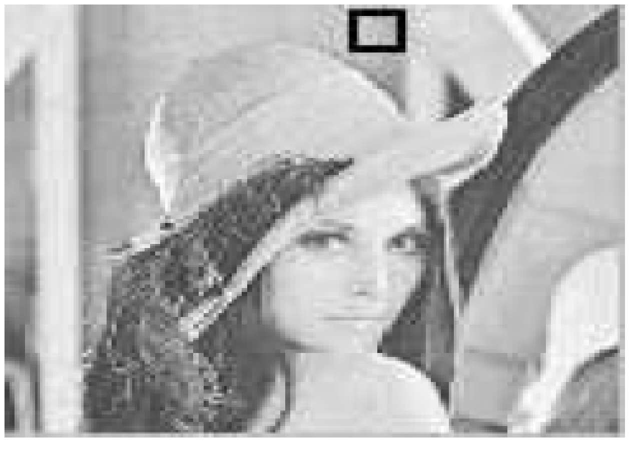
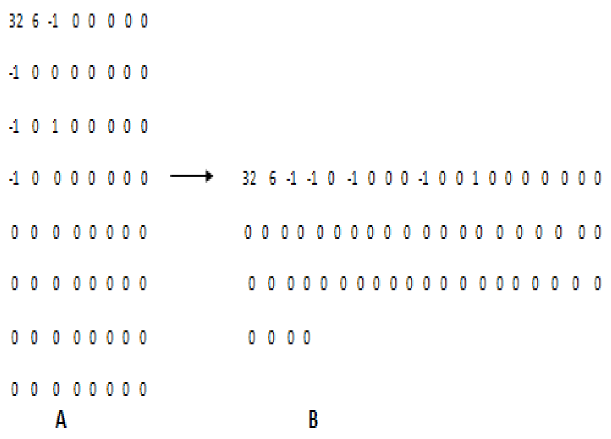


Fig. 8. Lena’s image in Y sub-band



```
32 6 -1 0 0 0 0 0
-1 0 0 0 0 0 0 0
-1 0 1 0 0 0 0 0
-1 0 0 0 0 0 0 0  ⟶  32 6 -1 -1 0 -1 0 0 0 -1 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0      0 0 0 0
 0 0 0 0 0 0 0 0
        A                        B
```

Fig. 9. A) Quantization of DCT coefficients, B) zigzag order.

Group 3 = -2
Group 4 = -1
Group 5 = 1
Group 6 = 2
Group 7 = 3
Group 8 = 4

Now, using the first three bits of the letter O. We place the corresponding decimal valuec (-1) after the last non-zero value within the zigzag array showed in Fig. 9 B). The resulted Zigzag array will be:
32 6 -1 -1 0 -1 0 0 0 -1 0 0 1 -1 0 0 0 . . . 0 0 0

Finally, the processed of encoding will continue as usual by applying RLC on DC component and DPCM on AC value then to Huffman encoding.

B) *Extracting the secret message from the cover image:* At the receiver side, the last value in the Zigzag array of the retrieved block based on the SCBA will be extracted and mapped back to the original bit representation. The extracted value will be added to a buffer that accumulates the binary representation of the completed secret message. The receiver will obtain the SCBA array through a secure channel.

V. Performance Evaluation and Results

In this section, we show and discuss the results of our experiments to illustrate the impact of hiding the secret message based on our proposed technique. We first provide a brief description of the experiment setup followed by the conditions and results of the experiment.

A. Implementations

Our proposed algorithm is implemented entirely using Java Eclipse. We calculate the difference between the cover image and cover image after embedding the secret message using MATLAB to measure the effect of the embedding method on the compression process. We also developed a Graphical User Interface (GUI) that allows the user to input a secret message to be embedded in a cover image using the proposed technique.

B. Experiments

In order to choose the best zigzag array of the cover image blocks, we need to answer the following questions.

1) Does the number of elements in the zigzag array before the last non-zero coefficient tolerate the inserted data chucks differently?
2) Does the size of the inserted chucks affects the compression process differently?
3) Which proposed method of adding the data chunk affect the compression process and what is the best threshold to use?

The first information we need is to find how many blocks are available in each image of our test data set and for each block how many are the coefficient values in the zigzag array. Our data set includes four popular images that many researchers use in order to compare the results. In TABLE I, it shows the maximum number of blocks in each range using 17 different intervals based on the number of coefficient values in each zigzag array before the last non-zero values of each image block. The results in TABLE I show the following information.

1) The number of blocks in each image.
2) The average number of the block in each range.
3) The ratio between the number of blocks in each interval, and the number of blocks in each image.

Based on TABLE I, we can observe that the number of blocks is higher on the 1-30 and 1-20 intervals. All the other intervals have less than 30% average which limits the size of the secret message to be embedded. Now, the question of which interval is better to use in term of less impact on the compression process. To answer the latter question, we generated 1000 different secret messages that occupy at least 50% of the selected blocks and measured the difference of pixel values before the embedding and after the embedding process. The results lead that using the interval 1-20 of non-zero coefficient values before the last non-zero coefficient value always resulted in less difference. Thus, we continue our experiments using the interval 1-20; although it has 7% fewer blocks than the 1-30 interval. This results answered the first question of our experiments cause.

Now to answer question two and three, we used the previous interval as our threshold value to choose which zigzag array to embed our data in. In TABLE II we show the experiments of each of our proposed methods in order to find the best method and the best size of the inserted data chunk. TABLE II shows the following information:

1) The used method and case.
2) The number of pixels changed during the embedding process with a value greater or equal to 50.
3) The average of difference using all values pixel.

The reason for showing the number of pixels changed with values equal or larger to 50 is to emphasize what human can detect. Usually, human eyes cannot detect changes on pixels with values less than 50. Thus, the less no of changed pixels means that human eyes will not notice any changes on the picture after the embedding process. This test is not enough, as most of the detection tools are computerized. In this case,

TABLE I. Number of Blocks based on the Number of Coefficients Until the Last Non-Zero Value

Number of blocks							
Total blocks	12288	6500	108000	108000	17856	30720	
Zigzag array contents interval	gg (1024*633)	r (510*737)	Flower (1920*1200)	Girl (1920*1200)	Animal (512*512)	Lena (512*512)	Average
1 - 30	(29%) 8783	(46%) 8253	(53%) 56968	(52%) 55854	(48%) 3142	(47%) 5783	23130
1 - 20	(25%) 7806	(42%) 7459	(48%) 51945	(44%) 48013	(43%) 2809	(40%) 4886	20486
1 - 15	(21%) 6349	(26%) 4721	(38%) 40747	(26%) 27808	(36%) 2348	(33%) 4061	14339
1 - 10	(20%) 6222	(25%) 4469	(36%) 38714	(22%) 23703	(34%) 2248	(32%) 3886	13207
10 - 30	(8%) 2561	(21%) 3784	(17%) 18254	(30%) 32151	(14%) 894	(15%) 1897	9923
15 - 30	(8%) 2434	(20%) 3532	(15%) 16221	(26%) 28046	(12%) 794	(14%) 1722	8791
10 - 20	(5%) 1584	(17%) 2990	(12%) 13231	(23%) 24310	(9%) 561	(8%) 1000	7279
30 - 60	(6%) 1778	(27%) 4771	(12%) 2359	(14%) 14725	(14%) 922	(9%) 1081	4772
20 - 40	(5%) 1567	(9%) 1607	(6%) 6868	(14%) 15010	(11%) 716	(14%) 1703	4578
30 - 50	(5%) 1635	(25%) 4435	(2%) 2348	(13%) 14308	(12%) 771	(9%) 1081	4096
20 - 30	(3%) 977	(4%) 794	(5%) 5023	(7%) 7841	(5%) 333	(7%) 897	2644
40 - 60	(4%) 1188	(22%) 3958	(0.48%) 514	(7%) 7556	(8%) 539	(2%) 275	2338
40 - 50	(3%) 1045	(20%) 3622	(0.47%) 503	(7%) 7139	(6%) 388	(2%) 275	2162
30 - 40	(2%) 590	(5%) 813	(2%) 1845	(7%) 7169	(6%) 383	(7%) 806	1934
30 - 45	(4%) 1213	(17%) 3079	(2%) 2314	(11%) 12314	(10%) 691	(9%) 1068	1394
45 -60	(2%) 565	(9%) 1692	(0.04%) 45	(2%) 2411	(4%) 231	(0.11%) 13	826
50 - 60	(0.004%) 143	(2%) 336	(0.01%) 11	(0.39%) 417	(2%) 151	(0%) 0	176

the number of the pixel value has no effect of the detection process. Thus, the second information which is the average difference is introduced to fulfill this test. As we see in TABLE II, using the first method and 2-bits data chunk size has the less average of difference. However, the first method using the 3-bits data chunk size also have the same average difference. In this case, using larger data chunk will provide more space to hide the secret message. Thus, we can conclude that using Method 1 is always better than Method 2. Also, using case 2 is better for the sake of space availability for hiding the data.

TABLE II. Results of All Cases of Range from 1 to 20 Zigzag Value

Number of non-black points ($\geq$ 50)								
No. of method	Case No.	Animal	Flower	gg	Girl	Lena	r	Avg.
Method 1	Case 1 (2 bits)	1	0	1	0	0	0	0.33
	Case 2 (3 bits)	1	0	1	0	0	0	0.33
	Case 3 (4 bits)	1	0	1	0	4	1	1.16
Method 2	Case 1 (2 bits)	5	0	76	3	0	3	14.5
	Case 2 (3 bits)	462	34	0	43	30	22	98.5
	Case 3 (4 bits)	454	28	0	49	22	28	96.83

C. Results

In this section, we present the results based on the experiments that we discovered in the previous subsection. The following examples show the cover images in JPEG format as input, the cover image after embedding a secret message and the difference image between the cover image and the cover image after embedding the secret message. We use the following values for the following examples:

1) The specific range is from 1 to 20.
2) The secret message embedded in the selected block by using the second case of the first method (3 bits from the secret message).
3) The pixel value is greater than or equal to 50.

Example One: Using the animal image in Fig. 10 (a) as a cover image, the number of non-black points that represent the difference between Fig. 10 (a) and Fig. 10 (b) is illustrated in Fig. 10 (c) and is equal to 1. The MSE is 9.88, and the PSNR is 38.22.

Example Two: Using the flower image Fig. 11 (a) as a cover image, the number of non-black points that represent the difference between Fig. 11 (a) and Fig. 11 (b) is illustrated in Fig. 11 (c) and is equal to 0. The MSE is 1.89, and the PSNR is 45.41.

Example Three: Using the Lena image Fig. 12 (a) as a cover image, the number of non-black points that represent the difference between Fig. 12 (a) and Fig. 12 (b) is illustrated in

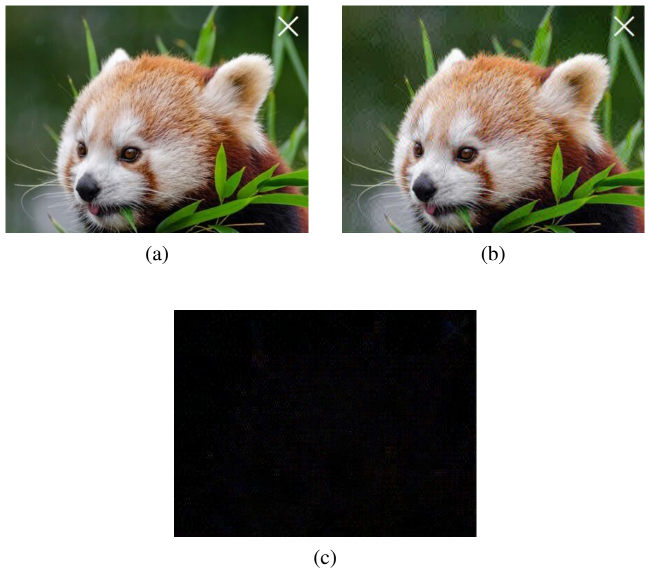


Fig. 10. A) Animal cover image, B) Animal cover image after embedding the secret message, C) The difference image between A and B.

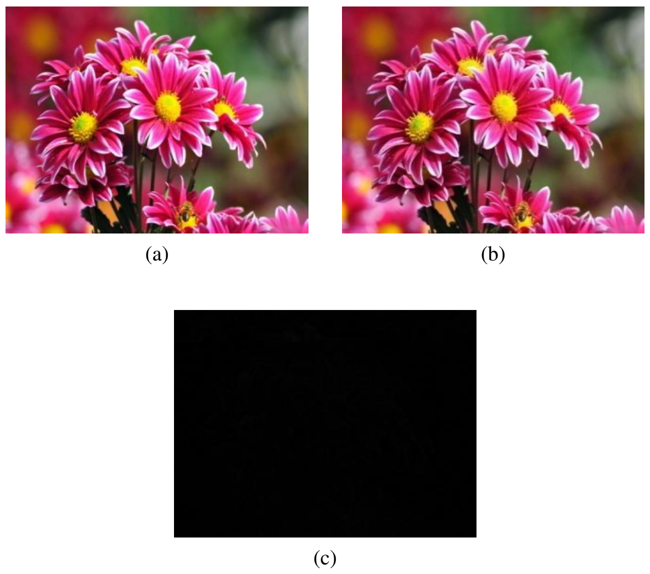


Fig. 11. A) Flower cover image, B) Flower cover image after embedding the secret message, C) The difference image between A and B.

Fig. 12 (c) and is equal to 0. The MSE is 4.86, and the PSNR is 41.36.

Example Four: Using the the girl image Fig. 13 (a) as a cover image, the number of non-black points that represent the difference between Fig. 13 (a) and Fig. 13 (b) is illustrated in Fig. 13 (c) and is equal to 0. The MSE is 6.45 and the PSNR is 40.07.

The last four examples show that the difference between the cover image and the cover image containing the secret message is hard to be detected by human eyes. However, in our proposed algorithm we assume that the attacker cannot obtain the original image. Thus, finding the locations of the secret message cannot be detected without the use of SCBA or the secret message.

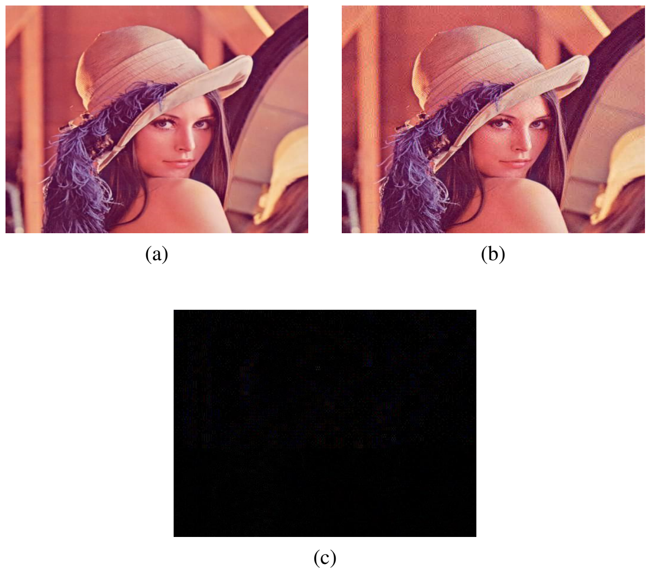


Fig. 12. A) Lena cover image, B) Lena cover image after embedding the secret message, C) The difference image between A and B.

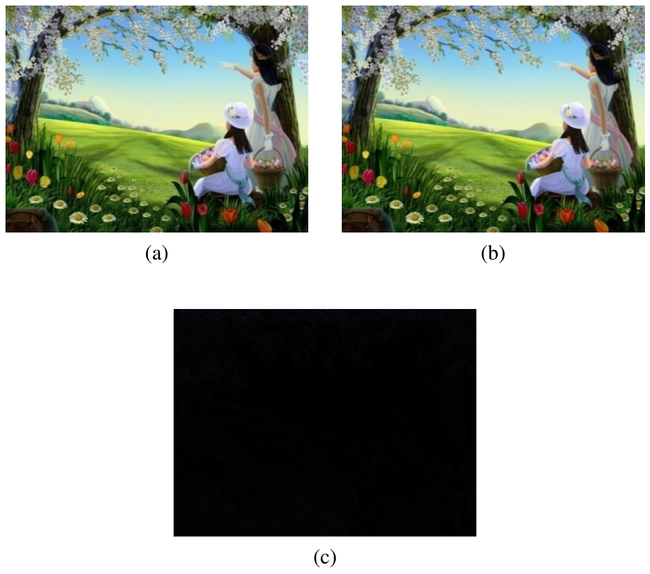


Fig. 13. A) Lena cover image, B) Lena cover image after embedding the secret message, C) The difference image between A and B.

VI. Conclusion

In this paper, we proposed a new technique to hide data in JPEG images using zigzag coefficients array to embed secret messages. The aim of using this method is to insert a secret message within a JPEG image without affecting the image quality and compression ratio. Therefore, nobody can detect or reveal the secret message hidden in the image. In this paper, we proposed two methods to embed the secret data using three sizes of data chunks that will be embedded within the zigzag coefficients array. Our experiments show that our proposed method one utilizing 3-bits data size outperform the other proposed second method in term of secret message space availability and the minimal difference after embedding the secret message. Based on our proposed selected method, we can utilize the use of almost 40% to 50% out of the total number of blocks to hide the secret message. The sender and the receiver can either share the SCBA or the private key to embed or reveal the secret message securely. We implemented the concept of shuffling the location of the embedded data using a chaotic based random number generator for each embedding process to increase the security level in our proposed algorithm. The algorithm was implemented entirely using JAVA libraries to produce an application that is used to run our experiments and to evaluate the proposed algorithm performance.

Acknowledgment

This work made possible by financial support from Applied Science Private University in Amman, Jordan.

References

[1] R. Gupta, S. Gupta, A. Singhal, Importance and techniques of information hiding: a review, *International Journal of Computer Trends and Technology (IJCTT)*, vol. 33, pp. 260–265, Mar 2014.

[2] W. Bender, D. Gruhl, N. Morimoto, A. Lu, Techniques for data hiding. *IBM System Journal*, vol. 35, pp. 313–336, Feb 1996.

[3] Z. AL-Ani, A. Zaidan, B. Zaidan, H. Alanazi, ”Overview: Main Fundamentals for Steganography”, *Journal of Computing*, Vol. 2, pp. 158-165, Mar 2010.

[4] M. Jain, S. Lenka, ”A Review of Digital Image Steganography using LSB and LSB Array”, *International Journal of Applied Engineering Research*, Vol. 11, pp. 1820-1824, 2016.

[5] P. Mahajan, A. Koul, ”CEET: A Compressed Encrypted and Embedded Technique for Digital Image Steganography”, *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 16, pp. 44-52, Apr 2014.

[6] U. Shankar, ”Image Compression Techniques, *International Journal of Information Technology and Knowledge Management*, Vol. 2, pp. 265-269, Dec 2010.

[7] A. Raid, W. Khedr, M. El-dosuky, W. Ahmed, ”Jpeg Image Compression Using Discrete Cosine Transform-A Survey”, *International Journal of Computer Science and Engineering Survey (IJCSES)*, Vol. 5, pp. 39-47, Apr 2014.

[8] H. Lu, X. Shy, Y. Shit, *et al.*, Watermark embedding in DC components of DCT for binary images, *In Proc. of IEEE Workshop on Multimedia Signal Processing*, St.Thomas, VI, USA, Dec 2002, pp. 300-303.

[9] M. Kaur, Robust watermarking into the Color Models based on the Synchronization Template, *In Proc. of International Conference on Information and Multimedia Technology*, Jeju Island, South Korea, Dec 2009, pp. 296-300.

[10] M. Lafta, ”Image Watermarking based on Huffman Coding and Laplace Sharpening”, *Journal of the college of education for women*, vol. 22, pp. 173-184, 2011.

[11] S. Sujatha, M. Sathik, ”A Novel DWT Based Blind Watermarking for Image Authentication”, *International Journal of Network Security*, vol. 14, pp. 223-228, July 2012.

[12] R. Preda, ”Semi-fragile watermark for image authentication with sensitive tamper localization in the wavelet domain”, *Journal of the international Measurement Confederation (IMEKO)*, vol. 46, pp. 367-373, 2013.

[13] M. Yesilyurt, Y. Yalman, A. Ozcerit, ”A new DCT based watermarking method using luminance component”, *Elektronika ir Elektrotechnika*, vol. 19, pp. 47-52, 2013.

[14] M. Khan, M. Rahman, M. Sarker, ”Digital Watermarking for Image Authentication Based on Combined DCT, DWT and SVD Transformation”, *International Journal of Computer Science Issues*, vol. 10, pp. 223-230, May 2013.

[15] J. Jeswani, T. Sarode, ”Improved Blind Color Image Watermarking using DCT in RGB Color Space”, *International Journal of Computer Applications (IJCA)*, vol. 92, pp. 50-56, Apr 2014.

[16] M. Mundher, D. Muhamad, A. Rehman, *et al.*, ”Digital watermarking for images security using discrete slantlet transform”, *International Journal of Applied Mathematics and Information Sciences*, vol. 8, pp. 2823-2830, Jan 2014.

[17] M. Haribabu, H. Bindu, V. Swamy, ”A Secure and Invisible Image Watermarking Scheme Based on Wavelet Transform in HSI Color Space”, *Computer Science Journal*, vol. 93, pp. 462-468, 2016.

[18] M. Ghebleh, A. Kanso, ”A robust chaotic algorithm for digital image steganography”, *Communications in Nonlinear Science and Numerical Simulation*, vol. 19, pp. 1898-1907, June 2014.

[19] J. Mazumder, K. Hemachandran, ”Color Image Steganography Using Discrete Wavelet Transformation and Optimized Message Distribution Method”, *International Journal of Computer Sciences and Engineering*, vol. 2, pp. 90-100, July 2014.

[20] G. Swain, ”Digital image steganography using variable length group of bits substitution”, *Procedia Computer Science*, vol. 85, pp. 31-38, 2016.

[21] P. Das, S. Kushwaha, M. Chakraborty, ”Multiple embedding secret key image steganography using LSB substitution and Arnold Transform”, *in Proc. 2nd International Conference on Electronics and Communication Systems (ICECS)*, 2015, pp. 845-849.

[22] S. Kumar, S. Kumari, S. Patro, *et al.*, ”Image Steganography using Index based Chaotic Mapping”, *IJCA Proc. on International Conference on Distributed Computing and Internet Technology (ICDCIT)*, Jan 2015, pp. 1-4.

[23] K. Bhaskar, M. Bakale, P. Chaure, P. Shirke, ”Image Steganography for data hiding Using Huffman code, Zigzag and OPAP”, *International Journal of Emerging Trends and Technology in Computer Science (IJETTCS)*, Vol. 4, pp. 91-93, Dec 2015.

[24] F. Almasalha, R. Hasimoto-Beltran, A. Khokhar, ”Encryption of Entropy-Coded Video Compression Using Coupled Chaotic Maps”, *Entropy*, Vol. 16, pp. 5575-5600, Oct 2014.