Journal of Theoretical and Applied Information Technology
28th February 2021. Vol.99. No 4

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SELECTIVE IMAGE COMPRESSION-ENCRYPTION ALGORITHM USING ADAPTIVE HUFFMAN CODING AND AES

[1]RUSHDI HAMAMREH, [2]EMAD TABIB

[1]Computer Engineering Department, Al-Quds University, Palestine

[2]Computer Engineering Department, Graduate Studies, Al-Quds University, Palestine

E-mail: [1]rushdi@staff.alquds.edu, [2]imad.tabib@students.alquds.edu

ABSTRACT

The interest in data transmission is increasing in all types of multimedia such as digital images, text, audio, and video. Digital images have special characteristics such as data redundancy and strong correlation between adjacent pixels, that make it difficult for traditional ciphers like IDEA, AES, DES, RSA to deal with real-time digital image encryption as these ciphers require high computational power, to overcome all these issues in digital images transmission, and keep the model secure, we have proposed a selective image encryption algorithm that combines Adaptive Huffman coding (AHC) and Advanced Encryption Algorithm (AES) and proposed a selective bits model. As we know the digital image is represented by a 2D matrix, and each pixel in the matrix represented by 8 bits (1-Byte). To begin with, the plain-image matrix is divided into N X N blocks; where N is a multiple of 4. Then we selected the leftmost bits of the binary representation of the block pixels, then the selected bits from each byte are re-converted to decimal, the values of the generated block will be in the range of [0:1], [0:3], [0:7] for 1, 2, 3 bits selections respectively. Then the resulted blocks are compressed by AHC, then encrypted by AES algorithm. Also, we calculated the mean value of each block from the non-selected bits above. Both the encrypted selected bits matrix and the encrypted mean vector will be transmitted. The experimental result shows that the proposed algorithm provides a competitive compression ratio, high image quality (SSIM), high PSNR, in addition to the high-security; we measured the pixel sensitivity of images (NPCR and UACI) and got values close to the optimal values.

Keywords: *Image compression, lossy image compression, Selective Image Compression and Encryption, Adaptive Huffman coding, Advanced Encryption Algorithm,*

1. INTRODUCTION

Today, the Internet is used to transfer large amounts of important and valuable data. Due to the significant increase in the transmission of digital images over the internet, the goal of obtaining a model that combines both bandwidth reduction and digital image transmission in a secure manner has become more attractive to researchers around the world. To achieve this, we should use the data compression and encryption algorithms. Data compression is a technique to reduce the quantity of data without excessively reducing of its quality. The transition and storing of compressed multimedia data are much faster and more efficient than original uncompressed multimedia data [1]. Because the Internet has many attack points, it is vulnerable to many types of attacks, so this information must be protected from unauthorized access using some of the encryption methods. Data encryption is a security method in which information appears to be encrypted or unreadable by an unauthorized person or organization. and can only be decrypted by the authorized user with the correct encryption key. Many researchers approach different models for image compression and encryption [2,3,4,5,6,7] to overcome image transmission issues. The goal of digital image transmission is to get a secured and less computational system; less encryption time, to make it real-time encryption. In this thesis. We proposed a model to reduce the redundancy of the image data before using the AES encryption; to overcome the image encryption issues mentioned above, by taking the bits containing most of data in the 8-byte-pixel of the 2D image matrix. To further reduce the image data, we used Real-Time Encoding (AHC) technology to reduce redundant data storage and enhance transmission time. Advanced Encryption Standard is applied to provide the highest level of security.

The rest of the paper is organized as follows: In Section 2, we list some of the related works in the same field. In Section 3, we describe the image compression technique used in our proposed model. In Section 4, we describe the image encryption algorithm used in our proposed model. briefly introduce the methods used in this paper. The proposed model is described in section 5. In section 6, we test and analyze the compression efficiency and the security of the model, through various statistical analysis. Finally, conclusions and future works are given in section 7 and 8 respectively.

2. RELATED WORKS

In recent years, some approaches [2,3,4,5,6,7] utilizing image compression and encryption models were introduced. Bruno Carpentieri in [2] studied the combination of compression and encryption techniques on various digital data. The experimented done on 2D data with three standard compression algorithms, JPEG, Lossless JPEG, and JPEG 2000 in lossless mode, and using four standard encryption algorithms, DES, 3DES, RC4, and AES. The aim of his work is to study the cost of encryption in terms of file size after performing data compression, and how bad is performing first encryption and then compression. Per his results, the cost of security is negligible if we perform first compression and then encryption and It is not efficient at all to encrypt the file then compress it. The file size will definitely grow and in some cases the resulting output will be far larger than the original input. Tong et al. [3] studied combining lossy compression technique using LWT with lossless compression technique using SPIHT coder, followed by symmetric cryptosystem using Chaotic sequence generation. The experiment test is done using five grayscale images with a size of 512 x 512 pixels. Experimental results show that the compressed file size is about 50% of the input file size. Also, the encryption method passes many security tests, such as sensitivity test, entropy test, autocorrelation test. Samer Isayed et al. [4] proposed an algorithm using lossy compression technique with lossless compression technique using Huffman coding, followed by symmetric cryptosystem using AES. The testing results is done on 4 grayscale images with a size of 512 x 512 pixels. The proposed lossy technique is a thresholding algorithm which mainly set image intensity values between ± threshold (T), They studied the effect of changing the threshold and block sizes on the compression ratio and the quality of the image, their results show that the compression ratio increases when block value is increased, but with a poor image quality. They achieved the best compression ratio at T= 8 with a block size of 64 × 64 pixels in all test images with average 0.4, 17.6dB, 3.58 for the SSIM, PSNR and CR respectively. Pratibha Chaudharya et al. [5] proposed a Joint Image Compression and Encryption Scheme is proposed for grayscale images. The testing done on images with different dimensions 256X256, 512X512 and 1024X1024. The proposed model used Huffman and Arithmetic coding for compression and XOR cipher encryption method. The proposed model shows good compression ratio and execution time. Chao-Jen Tsai et al. [6] proposed A chaos-based joint compression and encryption (JCAE) schemes. This scheme improved the computation time, compression ratio, and estimation accuracy of three different chaos-based JCAE schemes. The fist used the auxiliary data structures to improve the existing chaos- based scheme. The second scheme solved the issues of large multidimensional lookup table overheads, and the last also enhances the accuracy of frequency distribution estimations. The results showed that the proposed scheme is faster and generate smaller files than existing JCAE schemes. T. Sudarson and R. Perumal [7] proposed a lossless compression and encryption model method, the compression using arithmetic coding technique after splitting the data into equal intervals, and encryption is performed by symmetric encryption technique using bit-wise XOR with pseudorandom bit sequence. The algorithm results showed that the model is secure and immune to chosen plaintext attack.

3. ADAPTIVE HUFFMAN CODING

Huffman coding is a lossless data compression technique. Huffman coding is based on the frequency of occurrence of a data item i.e. pixel in images. The technique is to use a lower number of bits to encode the data in to binary codes that occurs more frequently [8]. Huffman coding suffers from the fact that the decompressor needs to have some knowledge of the probabilities of the characters in the compressed files. Not only this can add somewhat to the bits needed to encode the file, but, if this crucial piece of knowledge is unavailable, then compressing the file will require two passes, one-pass to find the frequency of each character to construct the Huffman tree and the second parameter to actually compress the file. Adaptive Huffman coding algorithms improve the compression ratio by applying to the model the statistics based on the source content sent from the immediate past. An alphabet and its frequency table are dynamically adjusted after reading each symbol during the process of compression or decompression [9]. The

algorithm working during creating the tree for "abb" text is represented in Figure 1. Table 1 shows the sequence of codes sent to the decoder for Figure.1 example.

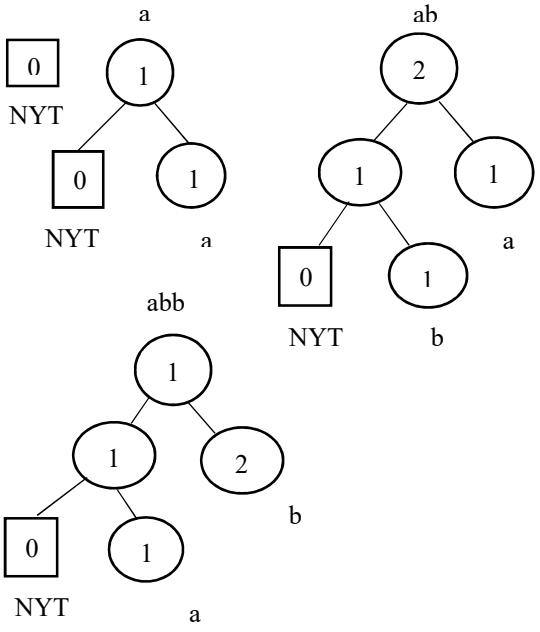

Figure 1: Encoding abb using AHC.

Table 1: The Sequence of Codes Sent to the Decoder

NYT	A	NYT	B	b
0	01100001	0	01100010	01

This method is based on the same principles as the static method with the following extensions:

a) every node in the tree has its key number, maximum value of the key number in the tree can be calculated as in formula (1):
KeyNum=2 .maxNumOfCharInAlphabet+1 (1)
- the root has the largest key number;
- an ancestor has larger number than any of its descendants;
- the right descendant should have larger key number than the left descendant.

b) After input of any character the tree is updated

c) A special leaf node called NYT (not yet transmitted) is used for both indicating the place for a new character and for signalizing that there is a new character is obtained, its key has the least value in the tree, and its weight should always equal to zero;

d) A set of nodes with equal weight values is called a block.

In case of AHC not only the codes of the characters are transmitted. Auxiliary codes such as the code of NYT Node and ASCII codes of the new-coming characters are being transmitted as well [10].

3.1 Image Compression Performance Parameters

To measure the loss in the image compression, we use some standards performance parameters as described in the formulas (2, 3, 4, 5): Compression Ratio (CR), Mean Square Error (MSE), Peak Signal to Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM) [11,12]. PSNR is the measurement of the peak error between the compressed image and original image. The higher the PSNR contains better quality of image. MSE is the cumulative difference between the compressed image and original image. Small amount of MSE reduce the error and improves image quality [13]. The SSIM index evaluates a test image X with respect to a reference image Y to quantify their visual similarity [14].

$$\mathrm{MSE} = \frac{1}{\mathrm{MN}} \sum_{i=1}^{M} \sum_{j=1}^{N} (X(i,j) - Y(i,j))^2 \quad (2)$$

$$\mathrm{PSNR} = 10 \log_{10} \frac{(2^n - 1)^2}{\sqrt{\mathrm{MSE}}} \quad (3)$$

$$\mathrm{SSIM} = \frac{(2 * \bar{x} * \bar{y} + C1)(2 * \sigma_{xy} + C2)}{(\sigma_x^2 + \sigma_y^2 + C2) * ((\bar{x})^2 + (\bar{y})^2 + C1)} \quad (4)$$

Where C1 and C2 are constants, $\bar{x}, \bar{y}, \sigma_x, \sigma_y$ and σ_{xy} are given as:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \quad (4.1)$$

$$\bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i \quad (4.2)$$

$$\sigma_x^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2 \quad (4.3)$$

$$\sigma_y^2 = \frac{1}{N-1} \sum_{i=1}^{N} (y_i - \bar{y})^2 \quad (4.4)$$

$$\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})((y_i - \bar{y}) \quad (4.5)$$

$$CR = \frac{numberOfBitsInCompressedMessage}{numberOfBitsInOrigionalMessage} \quad (5)$$

4. ADVANCED ENCRYPTION STANDARD

Advanced Encryption Standard (AES) algorithm is one of the block cipher encryption algorithm that was published by National Institute of Standards and technology (NIST) in 2000. The main goal of this algorithm was to replace DES, after some vulnerable aspects had appeared. NIST has called the security experts around the world to provide an innovative block encryption algorithm to encrypt and decrypt data with a robust and complex structure. A lot of algorithms were submitted by the expert around the world. After performing various criteria and security parameters, on October 2000 NIST announced that Rijndael is the winning algorithm, and named as AES, this algorithm has its own structure for encrypting and decrypting sensitive data, it is the strongest security protocol, since it is applied in both hardware and software [15]. AES is currently computationally unbreakable and likely to remain unbreakable unless a future quantum computer can reach the required computational ability[16].

The AES encryption procedure are shown in Figure 2. There are sets of transformation operations in AES algorithm each operation is applied on a 2D array of bytes called state matrix. The state is a rectangular array of bytes and since the block size is 128bits or 16 bytes, the array is a dimension of 4 X 4 byte. The round key is similarly pictured as a 4x4 matrix. The form of state and key matrices is shown in Figure. 3.

State matrix

$$\begin{bmatrix} S_0 & S_1 & S_2 & S_3 \\ S_4 & S_5 & S_6 & S_7 \\ S_8 & S_9 & S_{10} & S_{11} \\ S_{12} & S_{13} & S_{14} & S_{15} \end{bmatrix}$$

Key Matrix

$$\begin{bmatrix} K_0 & K_1 & K_2 & K_3 \\ K_4 & K_5 & K_6 & K_7 \\ K_8 & K_9 & K_{10} & K_{11} \\ K_{12} & K_{13} & K_{14} & K_{15} \end{bmatrix}$$

Figure 3: State and Key Matrices.

As shown in Figure. 2, after an initial application of the AddRoundKey() transformation, the state is transformed by implementing a round function. The round function is executed Nr times, where Nr is number of rounds that its value depends on the key size [17]. It uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. But the final round slightly differs from the previous Nr-1 rounds, that it does not have MixColumn() transformation.

4.1 Image Encryption Test Parameters

Suppose ciphertext images before and after one-pixel change in a plaintext image are C1 and C2 respectively; the pixel value at grid (i, j) in C^1 and C^2 are represented as $C^1(i,j)$ and $C^2(i,j)$ and a bipolar array D is defined in eqn. (6). Then the Number of Pixel Change Rate (NPCR) and Unified Average Changing Intensity (UACI) can be mathematically represented by equations (7) and (8), respectively, where symbol T denotes the total

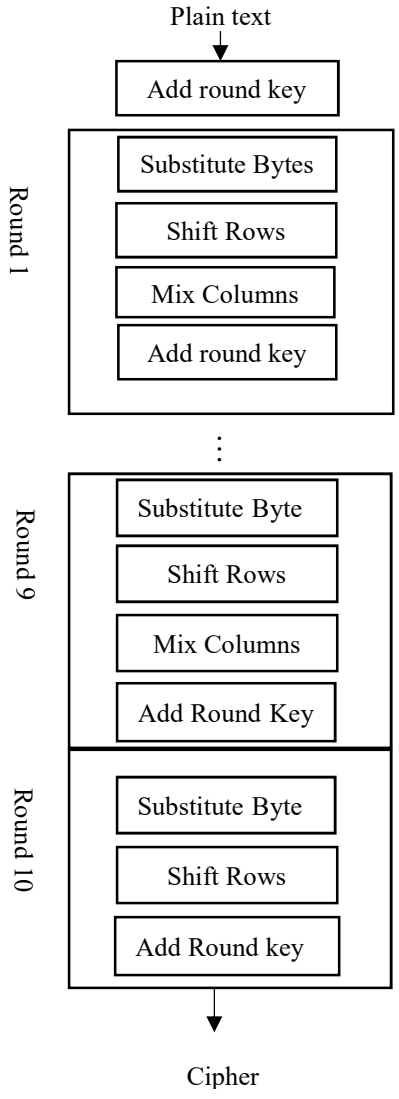

Figure 2: AES Encryption (128-bits).

number pixels in the ciphertext, symbol F denotes the largest supported pixel value compatible with the ciphertext image format. It is clear that NPCR concentrates on the absolute number of pixels that changes value in differential attacks, while the UACI focuses on the averaged difference between two

paired ciphertext images. The range of NPCR and UACI is [0, 1]. [18].

$$D(i,j) = \begin{Bmatrix} 0, \text{if } C^1(i,j) = C^2(i,j) \\ 1, \text{if } C^1(i,j) \neq C^2(i,j) \end{Bmatrix} \quad (6)$$

$$NPCR = \sum_{i,j} \frac{D(i,j)}{T} * 100\% \quad (7)$$

$$UACI = \sum_{i,j} \frac{|C^1(i,j) - C^2(i,j)|}{F \cdot T} * 100\% \quad (8)$$

5. PROPOSED MODEL

In order to reduce the redundancy of digital images to be sent over the network, we've proposed a model to selected the bits with highest data from each image pixel followed by lossless image compression and Symmetric image encryption to guarantee the highest security level. The model implemented using Matlab 2017b on different images. Figure 4 shows the proposed model architecture.

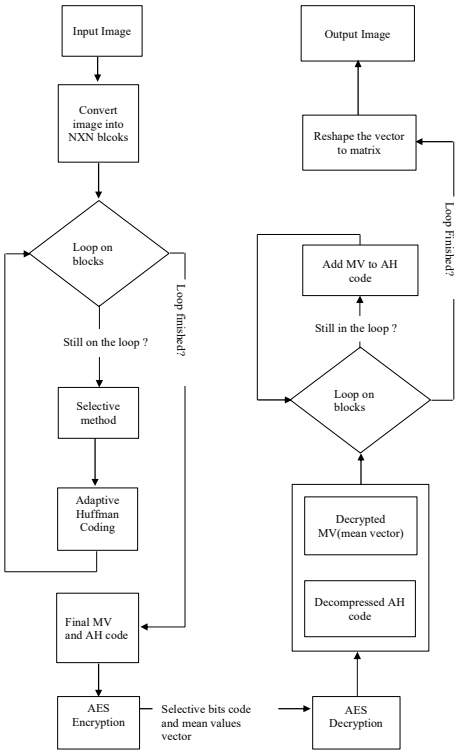

Figure 4: Proposed Model.

5.1 Compression and Encryption Process:

There are three steps at the sender side as follows:
1. Selective methodology: The plain-image matrix is divided into N X N blocks; where N is a multiple of 4. We've selected the leftmost bits of the binary representation of the block pixels, then the selected bits from each byte are re-converted to decimal, the values of the generated block will be in the range of [0:1], [0:3], [0:7] for 1,2,3 bits selections respectively. Also, we calculated the mean value of each block from the non-selected bits above, this will combine a vector of the mean values with length equal to the image blocks number. The detailed steps with and example are listed below. Figure 5. Shows our selective model.

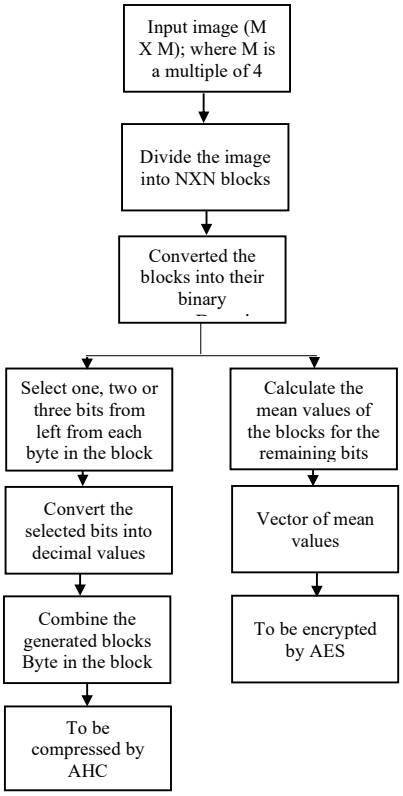

Figure 5: Bits Selective Model.

1.1. Divide the image into N × N blocks; where N is one the following values 4,8,16,32,64. Let's take a 4 X 4 image block example as shown in Table 2 to describe each step.

Table 2: 4 X 4 Image Block Example.

143	144	142	144
134	155	139	162
149	140	148	160
144	150	137	149

1.2. Convert the blocks into their binary values as shown in Table 3.

Table 3: The Binary Values for the Block.

10001111	10010000	10001110	10010000
10000110	10011011	10001011	10100010
10010101	10001100	10010100	10100000
10010000	10010110	10001001	10010101

1.3. Select the bits with the highest values, i.e. the leftmost bits; we have selected one, two or three bits in our experiments. Convert these bits into their decimal values. In this example we will select 3 bits as shown in Table 4 and Table 5.

Table 4: 3 Bits selected from the leftmost bits of each pixel

100	100	100	100
100	100	100	101
100	100	100	101
100	100	100	100

Table 5: Convert the Selected Bits to Decimal Values.

4	4	4	4
4	4	4	5
4	4	4	5
4	4	4	4

1.4. Convert the remaining bits in the blocks into their decimal values, and calculate the mean value of each block. The size of this mean vector will be small compared to the original image size and based on the block size selected. For example, here, from the 4 X 4 X 8 bits, we just sent 1 X 8 bits. The mean value for this block is 14. We added this step after we tested the effect of not sending it as we will show in the results section. Tables 6 and 7 describes this step.

Table 6: The Remaining Bits of Each Pixel After We Selected 3 Bits.

01111	10000	01110	10000
00110	11011	01011	00010
10101	01100	10100	00000
10000	10110	01001	10101

Table 7: The Decimal Values of the Remaining Bits.

15	16	14	16
6	27	11	2
21	12	20	0
16	22	9	21

2. The decimal values of selected bits in step 1.3 above will be compressed by Adaptive Huffman coding. The generated Adaptive Huffman code is: 000000100111111000000101111011111. The compression ratio for this block is (4 X 4 X 8) / (41 + 8) = 3.12. The compression ratio may be more for other blocks based on how close the block values are to each other.

3. All the compressed blocks generated from Adaptive Huffman coding will be combined and sent to be encrypted using Advanced Encryption Standard encryption algorithm, also to encrypt the vector of the mean values of the image blocks generated from step 1.4 above. The output code will have the same length as the code entered to the encryption.

5.2 Decompression and Decryption Process:

The inverse steps will be applied at the receiver side as follows:

1. The encrypted code of the selective bits block will be decrypted using AES decryption. As well as the mean vector. The output for the above example for the selected bits code is: 000000100111111000000101111011111. And the first value of the decrypted mean vector value will be 14.

2. The decrypted code of selective block will be decompressed by Adaptive Huffman decoding. The output block will be same as displayed in Table 5.

3. Rebuild the selective bits blocks as follows:

3.1. The decoded block will be converted into their binary values, the output block will be same as displayed in Table 4.

3.2. Add zeros to the right side of the binary values, the number of zeros will be (1 Byte – number of selected bits). In our example we will add 5 zeros (8-3). And convert the block values to its decimal values as displayed in Tables 8 and 9 respectively.

Table 8: Add Zeros to the Compressed Block Data.

10000000	10000000	10000000	10000000
10000000	10000000	10000000	10100000
10000000	10000000	10000000	10100000
10000000	10000000	10000000	10000000

Table 9: Covert the Compressed Block to Decimal.

128	128	128	128
128	128	128	160
128	128	128	160
128	128	128	128

3.3. Add the encrypted mean value for this block generated in step 1 to the block in the previous step. The generated block is displayed in Table 10.

Table 10: Add the Mean Value to the Decompressed Block.

142	142	142	142
142	142	142	174
142	142	142	174
142	142	142	142

6. DATA AND RESULTS

This section presents the results of the experiments done on some standard test images. We will compare the results of the original Adaptative Huffman compression and on AES image encryption without using our selective method with our new proposed selective method. As we mentioned before our ultimate goal is to reduce data volume and speed up the encryption process.

Simulation was carried out using MATLAB R2017b simulation software under Microsoft Windows 10 operating system.

6.1 Data Set:

We have used four standard test images in our experiments, in two sizes (dimensions) 512 X 512 and 1024 X 1024 and in both gray-scale and colored format. We used Lena, Boat, Barbara and Pepper images.

6.2 Results and Analysis:

We have split our results into 3 categories; in each category, we calculated the standards image compression and encryption metrics we mentioned in the previous sections; which are CR, SSIM and PSNR for data compression, NPCR and UACI for data encryption.

6.2.1 AHC followed by AES encryption without using our selective model

To compare our selective method with the AHC coding on the original image, Table 11 shows the testing results of compressing Lena image with adaptive Huffman coding only without using the selective method. As shown from the table, the compression ratio is less than 1; the code size is more than the original input matrix length, this is because the image values range is varying from 0:255 and this will generate a large Huffman tree since the size of the unique values will be almost equal to the size of the image block. From the table we can also see that SSIM is 1 and PSNR is inf; this is expected because the system is lossless

Table 11: Compression metrics result for AHC on the original image.

Block Size	CR	SSIM	PSNR
4 X 4	0.86	1	Inf
8 X 8	0.60	1	Inf
16 X 16	0.37	1	Inf
32 X 32	0.23	1	Inf
64 X 64	0.15	1	Inf

6.2.2 Using the selective method and Adaptive Huffman compression and AES encryption.

We've applied our selective model on different sizes to measure its performance on different sizes, we used 512 X 512 and 1024 X 1024. As mentioned in the proposed model chapter, our model tested on two stages as follows:

1. Compress and encrypt the selected bits only and ignoring the remaining bits to get better CR and to reduce the redundancy data.

2. Compress and encrypt the selected bits, and calculate the mean values for each block to generate a vector with size equal to image size (512 *512) / (Block size * Block size); for 512 X 512 images. This vector will reduce the CR a little as its size will be small compared to the input image size. For example, for 8 X 8 block size, the final mean vector will be (1 * 4096 * 8) bits. While the input image size is (512 * 512 * 8) bits. This vector will be encrypted with the AES to use it to generate a good image quality at the receiver side.

We will calculate and study the data compression and encryption performance for both stages in the next 2 subsections.

1. Image Compression Analysis:

Tables 12 to 15 show the testing results using our model on standard images with dimension 512 X 512, using different block sizes and different selective bits range using the two stages mentioned above.

In the first stage, we studied the results without sending the mean values vector, and we found that the image quality is slightly poor compared to the increase in CR compared to the second phase. Tables 12 and 13 show the compressed images results for CR, SSIM and PSNR for different block sizes.

In the second stage, we sent the mean values and compared it with the results from not sending this vector. Table 14 and 15 show the compressed images results for CR, SSIM and PSNR for different block sizes.

We got a good CR, PSNR and SSIM values, the high compression ratio in our method because of the block values range is reduced to 0:1 for 1-bit selection, 0:4 for 2-bits selection and 0:7 for 3-bit selections instead of 0:255 range for the original image; which made the use of Adaptive Huffman coding is more productive.

As we can see from the tables, we found that using more than 3 bits is not efficient as the compression ratio is decreased and so the execution time is increased in 3 bits selection than the 2 bits selection. However, it has better SSIM in 3 bits selection. We found that there are 2 options to select based on the compression metrics values, 8 X 8 and 16 X 16 Block sizes with 2-bits and 3-bits selections, and using the second stage where the mean vector was sent. 2-bits selection has the best values for the compression ratio and execution time, and 3-bits has the best image quality but with less compression ratio and more execution time.

We also studied our method on Colored images, Table 16 and Table 17 show the results of a colored Pepper image and Pepper gray-scale image respectively. We can see that the compression ratio appears close in both gray-scall and colored format, but SSIM is more for the colored image than gray-scall, this is because the colored image has 3-dimensional matrix, and when our blocks for each dimension are combined in the receiver side, may reduce the image loss.

Table 19 show the compression results for 1024 X 1024 images, we noticed that compression ratio, SSIM and PSNR increased when the image size increased using our model.

In Figures 6 to 15, the received images for two of the used images after applying our algorithm on the standard images on all block sizes and with all bit selections.

2. Image Encryption Analysis:

We studied the security of our method using some standard metrics for differential attacks, NPCR and UACI. First, we encrypted the original image using our model, and then we changed one-pixel value from the original image and encrypted it using our model. We changed one bit from the first byte of the image, if the changed bit is from the least significant bits which have the most data of the image; the bits we selected in our algorithm, the results show very high values for NPCR and UACI for 8 X 8 Block size and with 2- and 3-bits selection. Table 18 and Table 20 shows the result of the security metrics on Lena image on both sizes, the average values for NPCR and UACI were: 99.7% and 33.3% respectively. Which are close to the optimal values mentioned in [18]. Therefore, we selected 8 X 8 block size than 16 X 16 Block size which we also obtained a good compression performance.

Comparing our model with the AES image encryption directly without our model, shows that we reduced the encryption time, data size, and improved the pixel sensitivity of images, as the results in [43] shows the NPCR and UACI for AES encryption are 0.0354 and 0.0137 respectively, but in our SICE/AHAES model we achieved values close to the optimal values; we achieved 99.6, 33.46 for NPCR and UACI respectively.

Also comparing our model with similar lossy image compression-encryption models, shows that our model has high values for the CR, SSIM and PSNR, and optimal values for security tests parameters. For example, comparing to the suggested model in [4] we got very competitive values for the compression parameters. The best values obtained in [4] were (3.22, 0.45, 18.74) for CR, SSIM and PSNR respectively, where we got (5.54, 0.6940, 25.07) and (4.61, 0.7807, 30.08) for the same image. Also, our method works for all image sizes and in both types (greyscale and colored) images.

7. CONCLUSION

In this paper, we proposed a method for image compression using Adaptive Huffman coding to reduce the data size followed by image encryption using AES encryption algorithm to ensure image security. The proposed model divide the M X M image; where M is a multiple of 4, into equivalent sub images each has N ×N blocks; where N is one

the following values 4, 8, 16, 32 or 64. Then we converted the blocks into binary values and selected one of the following values from the 8 bits of each byte: 1, 2 or 3 bits, and reconverted the selected bits into decimal values. Also, we compute the mean value from the remaining bits that were in the step above after they converted to their decimal values. After that we apply Adaptive Huffman coding algorithm on the resulting selected bits image (NXN blocks), so we can use the power of Adaptive Huffman coding when the blocks values vary from 0:1 for 1 selected bit, 0:3 for 2 selected bits, and 0:7 for 3 selected bits instead of original image values from 0:255. Then, an AES encryption has been used to encrypt the combined blocks generated from the adaptive Huffman coding. As well as, we encrypt the mean values vector to be used in rebuild the image in the receiver side to improve the image quality.

We have studied the effect of changing the number of binary bits selected, block size ranges, image width and the effect of sending/not sensing the mean vector on the compression ratio, image quality (SSIM, PSNR), execution time. we found that the best values are when the Block size is 8 X 8 with two-bits or three-bits selection. As well, the security measurements show that we achieved values close to the optimal values for NPCR and UACI for 8 X 8 block size with two-or-three bits selection. This applies for both gray-scale and colored images.

Also, we found that the compression ratio and image quality is increased when the image width increased, as our tables show in the previous section. If we interested on one metric than other, we can select other options than the one we selected. For example, if we interested in the compression ratio, we can see that the best compression is at 16 X 16 block size with 1-bit selection and it still has a good image quality.

8. FUTURE WORKS

As our proposed methodology works only on images (Grey, RGB) with width NXN; where N is a multiple of 4, the block division algorithm can be modified to work with any width. Also, we can use some of enhanced AES methods to reduce the execution time since the execution time is still can be improved and multiple methods were introduced recently to enhance the execution time. Also, it can be modified to work with audio, video transmission, since we used Adaptive Huffman coding which works well with audio and video compression.

REFRENCES:

[1] Asadollah Shahbahrami, Ramin Bahrampour, Mobin Sabbaghi Rostami, Mostafa Ayoubi Mobarhan, "Evaluation of Huffman and Arithmetic Algorithms for Multimedia Compression Standards", *International Journal of Computer Science, Engineering and Applications (IJCSEA)*, Vol. 1, No. 4, University of Guilan (Iran), August 31, 2011, pp. 8-9.

[2] Bruno Carpentieri, "Efficient Compression and Encryption for Digital Data Transmission", *Security and Communication Networks*, Vol. 2018, Article ID 9591768, University of Salerno (Italy),May 17, 2018, pp. 1-8.

[3] X.-J. Tong, P. Chen, and M. Zhang, "A Joint Image Lossless Compression and Encryption Method Based on Chaotic Map", *Multimedia Tools and Applications*, Vol. 76, No. 12, July 30, 201, pp. 1-10.

[4] Samer Isayed, Dr. Liana Tamimi and Dr. Mousa Farajalla. "A Gray-scale Image Compression-Encryption Algorithm using Huffman Coding and AES", *Thesis for: MSC*, Palestine Polytechnic University (Palestine), Sept, 2018, pp. 56-57.

[5] Pratibha Chaudharya, Ritu Guptab, Abhilasha Singhc, Pramathesh Majumderd and Ayushi Pandey, "Joint image compression and encryption using a novel column-wise scanning and optimization algorithm". *International Conference on Computational Intelligence and Data Science (ICCIDS 2019)*, January 2020, pp. 1-10.

[6] Chao-Jen Tsai, Huan-Chih Wang and Ja-Ling Wu, "Three Techniques for Enhancing Chaos-Based Joint Compression and Encryption Schemes", *Entropy 2019*, Vol. 21, No. 1, January 2019, pp. 1-14.

[7] T. Sudarson, R. Perumal. "Simultaneous compression and encryption using arithmetic coding with randomized bits", *Computer Science (2012)*, pp. 1-4.

[8] S.L. Bawa, "Compression Using Huffman Coding", *IJCSNS International Journal of Computer Science and Network Security*, Vol. 10, No. 5, May 2010, pp. 133-139.

[9] Ida Pu, "Fundamental Data Compression", December 2005, pp. 91-92.

[10] Mikhail Tokovarov, "Modification of Adaptive Huffman Coding for use in encoding large alphabets", *International Conference of Computational Methods in Engineering Science (CMES'17),* Vol. 15, No. 4, January 2017, pp. 2-3.

[11] Alain Hore and Djemel Ziou, "Image quality metrics: PSNR vs. SSIM", *20th International Conference on Pattern Recognition*, Turkey, August 23-26, 2010, pp. 2366-2367.

[12] C.Sasi varnan, A.Jagan, Jaspreet Kaur, Divya Jyoti, Dr.D.S.Rao, "Image Quality Assessment Techniques pn Spatial Domain", *International Journal of Computer Science and Technology*, Vol. 2, No. 3, September 2011, pp. 178-181.

[13] Mr.Chandresh K Parmar, Prof.Kruti Pancholi, "A review on image compression techniques", *Journal of information, knowledge and research in electrical engineering*, Vol. 2, No. 2, Nov 12 - Oct 13, 2012, pp. 282.

[14] Gabriel Prieto Renieblas, Agustín Turrero, Alberto Muñoz González, Eduardo Guibelalde. "Structural similarity index family for image quality assessment in radiological images", *Journal of Medical Imaging,* Vol. 4, No. 3, July-Sept, 2017, pp. 035501-2.

[15] William Stallings, "Cryptography and Network Security" 4th Ed, 2005, pp. 58-309.

[16] Ahmad-Loay Sousi, Dalia Yehya, Mohamad Joudi, "AES Encryption: Study & Evaluation", *Cryptography & Network Security Papers*, Rafik Hariri University, November 2020, pp. 3-22.

[17] Dessalegn Atnafu Ayalneh, Dr. V.N.V. Manoj, Dr. Mike Venter, "Optimizing AES implementation for High-speed Embedded Application", *Thesis for: MSC*, March 2008, pp. 21-34.

[18] Yue Wu, Joseph P. Noonan, Sos Agaian, "NPCR and UACI Randomness Tests for Image Encryption" *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT).* April 2011, pp. 32.

[19] Farhad Maleki, Ali Mohades, S Mehdi Hashemi, and Mohammad Ebrahim Shiri. "An image encryption system by cellular automata with memory", *The Third International Conference on Availability, Reliability and Security*, March 4-7, 2008, pp. 1266–1271.

[20] Omar Farook Mohammad, Mohd Shafry Mohd Rahim, Subhi R. M. Zeebaree, Falah.Y.H Ahmed. "A Survey and Analysis of the Image Encryption Methods". *International Journal of Applied Engineering Research.* Vol. 12, No. 23. December 2017, pp.13-14.

Table 12: Lena image (512 X 512) compression results (without the mean vector).

Block Size	1 Bit Selected			2 Bits Selected			3 Bits Selected		
	CR	SSIM	PSNR	CR	SSIM	PSNR	CR	SSIM	PSNR
4 X 4	4.87	0.4503	11.6	4.43	0.6649	17.34	3.84	0.7734	23.07
8 X 8	6.28	0.4503	11.6	5.60	0.6649	17.34	4.65	0.7734	23.07
16 X 16	6.52	0.4503	11.6	5.66	0.6649	17.34	4.42	0.7734	23.07
32 X 32	6.22	0.4503	11.6	5.16	0.6649	17.34	3.77	0.7734	23.07
64 X 64	5.53	0.4503	11.6	4.36	0.6649	17.34	2.89	0.7734	23.07

Table 13: Barbara image (512 X 512) compression results (without the mean vector).

Block Size	1 Bit Selected			2 Bits Selected			3 Bits Selected		
	CR	SSIM	PSNR	CR	SSIM	PSNR	CR	SSIM	PSNR
4 X 4	4.54	0.3038	11.20	3.91	0.5743	16.83	3.03	0.7880	22.99
8 X 8	5.92	0.3038	11.20	4.95	0.5743	16.83	3.68	0.7880	22.99
16 X 16	6.17	0.3038	11.20	4.89	0.5743	16.83	3.44	0.7880	22.99
32 X 32	5.93	0.3038	11.20	4.32	0.5743	16.83	2.89	0.7880	22.99
64 X 64	5.68	0.3038	11.20	3.87	0.5743	16.83	2.44	0.7880	22.99

Table 14: Lena image (512 X 512) compression results (with the mean vector).

Block Size	1 Bit Selected			2 Bits Selected			3 Bits Selected		
	CR	SSIM	PSNR	CR	SSIM	PSNR	CR	SSIM	PSNR
4 X 4	4.69	0.7246	22.96	4.30	0.7293	26.02	3.73	0.7972	30.75
8 X 8	6.20	0.6617	21.63	5.54	0.6940	25.07	4.61	0.7807	30.08
16 X 16	6.50	0.6394	20.61	5.64	0.6826	24.35	4.41	0.7770	29.59
32 X 32	6.22	0.6362	19.57	5.15	0.6864	23.71	3.76	0.7790	29.17
64 X 64	5.53	0.6495	18.63	4.36	0.6927	23.30	2.89	0.7803	28.93

Table 15: Barbara image (512 X 512) compression results (with the mean vector).

Block Size	1 Bit Selected			2 Bits Selected			3 Bits Selected		
	CR	SSIM	PSNR	CR	SSIM	PSNR	CR	SSIM	PSNR
4 X 4	4.38	0.6570	20.73	3.79	0.7614	25.34	2.96	0.8307	29.90
8 X 8	5.85	0.5915	19.93	4.90	0.7314	24.69	3.66	0.8168	29.37
16 X 16	6.16	0.5632	19.29	4.87	0.7253	24.25	3.43	0.8143	29.05
32 X 32	5.93	0.5611	18.68	4.32	0.7267	23.90	2.88	0.8149	28.80
64 X 64	5.68	0.5651	17.98	3.87	0.7295	23.67	2.44	0.8152	28.65

Table 16: Colored Pepper image (512 X 512) compression results (with the mean vector).

Block Size	1 Bit Selected			2 Bits Selected			3 Bits Selected		
	CR	SSIM	PSNR	CR	SSIM	PSNR	CR	SSIM	PSNR
4 X 4	4.73	0.9067	23.28	4.20	0.9393	25.86	3.58	0.9761	30.39
8 X 8	6.30	0.8826	22.13	5.34	0.9286	25.05	4.36	0.9732	29.91
16 X 16	6.63	0.8520	21.05	5.35	0.9167	24.33	4.04	0.9710	29.54
32 X 32	6.44	0.8196	20.12	4.92	0.9042	23.68	3.42	0.9674	29.25
64 X 64	6.02	0.7695	19.14	4.14	0.8919	23.16	2.62	0.9646	29.02

Table 17: Gray-scale Pepper image (512 X 512) compression results (with the mean vector).

Block Size	1 Bit Selected			2 Bits Selected			3 Bits Selected		
	CR	SSIM	PSNR	CR	SSIM	PSNR	CR	SSIM	PSNR
4 X 4	4.95	0.7547	23.92	4.53	0.7671	27.25	3.84	0.7973	30.91
8 X 8	6.34	0.6716	22.36	5.58	0.7199	26.02	4.45	0.7763	30.08
16 X 16	6.53	0.6333	21.08	5.44	0.7045	25.01	4.02	0.7718	29.52
32 X 32	6.19	0.6314	19.86	4.86	0.7079	24.31	3.32	0.7732	29.15
64 X 64	5.45	0.6380	19.01	3.99	0.7123	23.74	2.46	0.7741	28.93

Table 18: Lena image (512 X 512) Encryption metrics test results.

Block Size	1 Bit Selected		2 Bit Selected		3 Bit Selected	
	NPCR	UACI	NPCR	UACI	NPCR	UACI
4 X 4	99.84	33.98	99.547	33.85	99.61	34.47
8 X 8	99.68	33.27	99.71	33.27	99.60	33.29
16 X 16	80.69	26.97	88.82	29.75	96.64	32.31
32 X 32	75.48	24.86	87.57	29.12	96.96	32.37
64 X 64	92.61	30.52	95.78	31.82	99.151	33.28

Table 19: Lena image (1024 X 1024) compression results (with the mean vector).

Block Size	1 Bit Selected			2 Bits Selected			3 Bits Selected		
	CR	SSIM	PSNR	CR	SSIM	PSNR	CR	SSIM	PSNR
4 X 4	4.87	0.7744	24.84	4.59	0.8050	27.54	4.21	0.8187	31.97
8 X 8	6.57	0.7300	22.98	5.99	0.7395	26.10	5.26	0.7935	30.83
16 X 16	6.88	0.7185	21.66	6.20	0.7200	25.15	5.17	0.7895	30.17
32 X 32	6.68	0.7195	20.64	5.83	0.7230	24.41	4.53	0.7910	29.65
64 X 64	6.29	0.7228	19.58	5.23	0.7286	23.75	3.79	0.7934	29.20

Table 20: Lena image (1024 X 1024) Encryption metrics test results.

Block Size	1 Bit Selected		2 Bit Selected		3 Bit Selected	
	NPCR	UACI	NPCR	UACI	NPCR	UACI
4 X 4	99.80	33.73	99.65	33.52	99.62	33.49
8 X 8	99.65	33.39	99.62	33.34	99.62	33.43
16 X 16	79.55	26.69	90.23	30.15	97.06	32.54
32 X 32	78.32	25.64	89.50	29.79	96.96	32.37
64 X 64	82.02	27.08	94.82	31.75	99.56	34.10

Figure 6: Barbara image (512 X 512) - 4 X 4 Block (1,2,3) bits selected respectively

Figure 7: Barbara image (512 X 512) - 8 X 8 Block (1,2,3) bits selected respectively

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 8: Barbara image (512 X 512) - 16 X 16 Block (1,2,3) bits selected respectively

Figure 9: Barbara image (512 X 512) - 32 X 32 Block (1,2,3) bits selected respectively

Figure 10: Barbara image (512 X 512) - 64 X 64 Block (1,2,3) bits selected respectively

Figure 11: Colored Pepper image (512 X 512) - 4 X 4 Block (1,2,3) bits selected respectively

Figure 12: Colored Pepper image (512 X 512) - 8 X 8 Block (1,2,3) bits selected respectively

Figure 13: Colored Pepper image (512 X 512) - 16 X 16 Block (1,2,3) bits selected respectively

Figure 14: Colored Pepper image (512 X 512) – 32 X 32 Block (1,2,3) bits selected respectively

Figure 15: Colored Pepper image (512 X 512) – 64 X 64 Block (1,2,3) bits selected respectively