Periodicals of Engineering and Natural Sciences ISSN 2303-4521
Vol. 9, No. 2, April 2021, pp.569-580

Joint image encryption and compression schemes based on hexa-coding

Mohammed H. Rasheed[1], Omar M. Salih[2], Mohammed M. Siddeq[3]
[1,2,3]Computer Engineering Dept., Technical College of Kirkuk, Northern Technical University

ABSTRACT

This research proposes a new image compression and encryption method depend on a modified JPEG technique combined with the Hexa-Coding algorithm. The compression algorithm starts by dividing an image into 8x8 blocks, then DCT (Discrete Cosine Transform) is applied to all blocks independently followed by uniform quantization. Additionally, the size of blocks is reduced by eliminating insignificant coefficients, and then Arithmetic coding is applied to compress residual coefficients. Finally, Hexa-encoding is applied to the compressed data to further reduce compression size as well as provide encryption. The encryption is accomplished based on five different random keys. The decompression uses a searching method called FMSA (Fast Matching Search Algorithm) which is used for decoding the previously compressed data, followed by Arithmetic decoding) to retrieve residual coefficients. These residuals are padded with zeros to rebuild the original 8x8 blocks. Finally, inverse DCT is applied to reconstruct approximately the original image. The experimental results showed that our proposed image compression and decompression has achieved up to 99% compression ratio while maintaining high visual image quality compared with the JPEG technique.

Keywords: Modified JPEG Technique**,** Hexa-Coding Algorithm, Fast Matching Search Algorithm**.**

Corresponding Author:

Mohammed H. Rasheed,
Technical College of Kirkuk
Northern Technical University
Kirkuk, Iraq.
E-mail: mohammed.rasheed@ntu.edu.iq

1. Introduction

Nowadays, networks and multimedia technologies are rapidly developing and became widely used in every aspect of life. Due to the wide availability and use of high-resolution multimedia devices and services, sharing of raw size digital media files requires a worthwhile amount of storage size and network traffic [1,2]. This has led the research community to drive their attention to improve image compression techniques for large data files while maintaining the best level of quality [3]. On the other hand, security and privacy have become an important and challenging aspect to protect multimedia files from unauthorized access due to the rapid increase in hacking attacks and personal privacy demands [4]. For these reasons, data compression and encryption have become a hot research area. Therefore, Joint image compression and encryption (JICE) are required to fulfill both compression and encryption goals simultaneously [5]. The inconsistency between compression and encryption makes the joint method troublesome. For images, JICE is an effective tool to provide security and save space. In the literature, two approaches have been implemented. The conventional approach suggests dividing the operation into two separated but still related stages: encryption and compression, where the encryption and compression are independent and performed separately. The other approach is to perform the encryption within the transform during the compression step [6]. The following paragraph illustrates and reviews some of the latest and well-known JICE algorithms developed by the research community.

In [7], A novel JICE based JPEG with a controllable image quality algorithm is presented. For encryption, embedded compression is developed within the DCT by introducing sign-flips into the butterfly's structure and employing them alternatively according to a secret key. The image can be recovered even without this key.

while in [8] an improved JICE algorithm is proposed. It employed a DCT transformation dictionary for image representation and then combining it with a hyper-chaotic system to attain image compression and encryption jointly. Another JICE scheme is proposed based on curvelet transform, run-length coding, and Huffman coding [6]. For encryption, a five-dimensional hyper-chaotic based on the Rabinovich scheme is employed with the construction of a new pseudorandom plain text key-stream generator. The Authors in [9] proposed a lossless JICE scheme based on IWT (Integer Wavelet Transform) and SPIHT (Set Partitioning In Hierarchical Trees|). The algorithm used a hyper-chaotic system, nonlinear inverse process, secure hash algorithm, and plaintext key-stream for secure encryption. Then, two JICE schemes are presented [10]. The first considers compression performance by dividing the image into non-overlapped 16×16 blocks and applying different encryption operations at the quantization and transformation steps of JPEG. The second scheme considers protection performance by adding the RSV (Run Size and Value) pairs combined at JPEG's coding s after the encryption of the first scheme. Additionally, in [11] a novel proposed algorithm based on two levels of DCT and Hexa-Coding encoded with Arithmetic coding is introduced and applied on images and video streams from YouTube. The results show that this algorithm can yield a high compression ratio and better perceptual quality. Finally, a novel crypto-compression algorithm based on five levels of Discrete Wavelet Transform (DWT) followed by a novel Hexa-Coding algorithm is proposed [12]. It also introduces a new algorithm called FMSA at the decompression phase. The results show high quality reconstructed images at high compression ratios.

This research paper uses the second approach where compression and encryption are performed separately. Our proposed algorithm introduces a new idea of image compression using a modified JPEG technique. This research modifies the JPEG technique by introducing a new quantization procedure that contributes to adjusting compression ratio as well as image quality. The Hexa-Coding algorithm is used then for providing encryption. The novelty of the Hexa-Coding algorithm is that it uses two groups of keys. The first group uses five randomly generated floating-point keys to minimize the amount of data without affecting quality, while the second group of keys is the probability table generated from the compressed data used as a way of encryption. One of the strengths of the proposed method is that it combines encryption and compression within its internal structure. Furthermore, the Hexa-Coding algorithm provides a second level of compression since it abbreviates every six values of the compressed data into a single value. The following sections explain the proposed algorithm in detail and organized as follows, Section 2 introduces the proposed joint compression-encryption algorithm and Section 3 describes the decryption and decompression algorithm. Sections 4 and 5 presents experimental results and conclusions, respectively.

2. The proposed joint compression-encryption algorithm

The proposed JICE algorithm illustrated as a functional block in Figure 1. The algorithm starts initially by partitioning the original image into non-overlapped blocks of size 8×8. If the image dimension is not in the form of a multiple of eight, zeros are padded to the last column and the last row of the image to compensate the missing values [13,14,15]. For simplicity, one 8×8 block is selected as an example of what is performed on all other blocks.

A pre-processing operation is performed directly on original image values to normalize the values and eliminate spurious and unwanted noise which will contribute to a higher compression ratio [16,17]. Our preprocessing methodology suggests that each original value in the block is subtracted from an integer value (r) which equals to the quotient of dividing the maximum value in the block by two and then round the result to the nearest integer.

$$round(r) = Max\ Value / 2 \tag{1}$$

The results show that careful selection of the value of (r) can contribute to removing irrelevant data and hence improve compression ratio while maintaining the overall image quality after decompression.

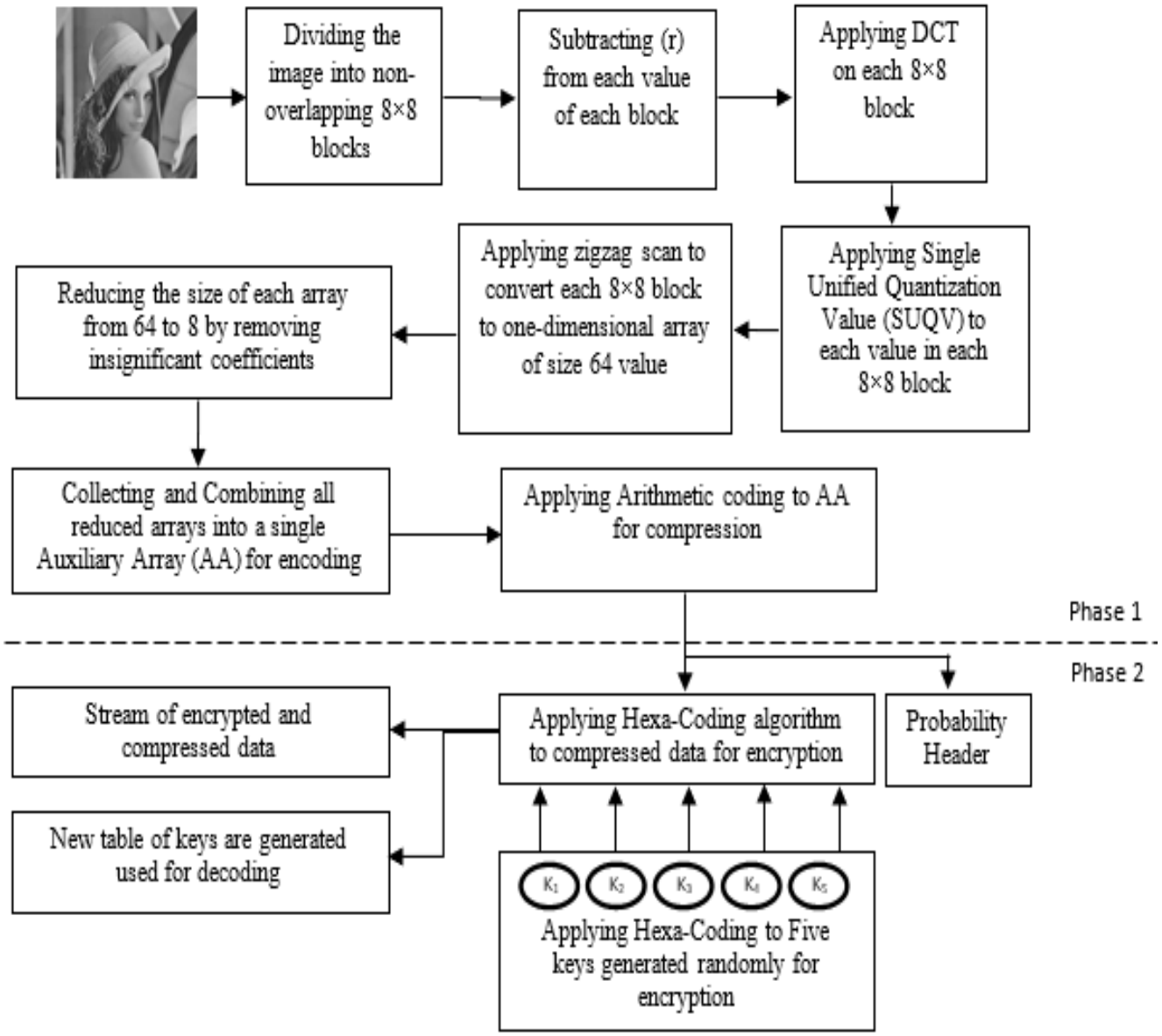


Figure 1. The proposed compression-encryption algorithm

For example, Figure 2 represents an 8×8 block containing original image values, a quick value-based search reveals that the largest value contained in this specific block is equal to 213, this value is then applied in equation (1) and the result is used to subtract all the original pixel values from as shown in Figure 2. It is important to note that resulted value from equation (1) (i.e., in our example 213/2) should be saved to be used to retrieve the data in the decompression phase therefore it will be saved in the header file.

The next step in our proposed algorithm is transforming each pre-processed 8×8 block from its current spatial domain to an equivalent frequency domain. The DCT is employed on all generated blocks to perform the transformation. Implementing DCT will help in separating differing important information in each block from the original image [17,18,19]. A DCT transformed block is shown in Figure 3, where an array consists of 64 values arranged as one single low frequency coefficient located on the top left side corner of the block, while the remaining 63 values are considered high frequency coefficients. Many of the high frequency coefficients can be ignored without causing serious deformation of the image since higher frequencies data have much small effect than low frequency data.

181	201	202	195	189	194	197	206
171	198	201	192	190	193	197	207
175	195	193	183	187	192	197	210
184	201	192	180	188	195	200	213
197	206	194	185	188	194	201	211
200	202	190	188	194	197	204	212
192	196	182	187	196	197	208	212
196	189	181	189	199	203	210	211

(a) Pixel values of an original

Y(i,j) - *r*

74	94	95	88	82	87	90	99
64	91	94	85	83	86	90	100
68	88	86	76	80	85	90	103
77	94	85	73	81	88	93	106
90	99	87	78	81	87	94	104
93	95	83	81	87	90	97	105
85	89	75	80	89	90	101	105
89	82	74	82	92	96	103	104

(b) Each value in the block 8x8 is subtracted with (213/2)

Figure 2 (a, b). a block 8×8 pre-processed by subtracting (213/2) from all values

A Single Unified Quantization Value (SUQV) is used for all data in our 8×8 block. Each numerical value is divided by the SUQV and rounded to the nearest integer. Based on our experiments and their outcomes, we have elected a single unified value equal to 64 as our SUQV. This value represents hard quantization to remove most of the unwanted coefficients. One of the reasons behind selecting a quantization value of 64 is that it represents one-fourth of the maximum value of each element in each block, which is 255. The second reason is to reduce the amount of data included in the compressed header file since quantization values must be preserved to be used in the decompression phase. Figure 3(b) shows the DCT transformed block quantized by a SUQV quantization value.

Now, it is very important to confirm the integrity of our results therefore all steps performed up to this point are inversed. Multiplying each value in the resulted 8×8 block by (64), and then inverse DCT applied to the block, finally adding (213/2) to each value in the 8×8 block. A simple comparison between the resulted 8×8 block values shown in Figure 4 with the original image values illustrated in Figure 2(a) clearly shows an affinity with the original values which proves the effectiveness of our proposed algorithm so far.

By examining the resulted two-dimensional array shown in Figure 4 and Figure 2(a), two important properties were observed. The first is that most of the values became zeros and thus can be omitted. The second is that most of the important values have been distributed among the bounded cells on the top left corner. Therefore, a zigzag scan is applied on each block to convert each quantized block from a two-dimensional array into a one-dimensional array of size (64) [13,18-21].

708	-40	31	-18	-12	-19	-7	-3
-12	6	-14	-26	-13	-6	5	-2
2	-6	-13	3	4	7	-1	-1
11	13	3	-1	1	2	-1	-2
4	4	4	3	0	1	2	-1
-3	-3	3	1	0	-2	-2	0
1	1	2	3	2	2	2	0
-2	1	2	-1	0	1	-1	0

(a) DCT transformed block

$Y_{DCT}(i,j)$ divided by *64*

11	-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

(b) The transformed block quantized by a single value of 64

<----- 8 Values ----->							
11	-1	0	0	0	0	0	0

(c) Zigzag scan applied to the block 8x8 to convert it to a one-dimensional array. Then the first 8 coefficients kept as residual, while other coefficients are ignored

Figure 3 (a-c). Three steps show DCT, uniform quantization, and reduced zigzag scan applied to the block 8×8 respectively.

It has been observed that zigzag scan is more practical to use in terms of arranging important values into close groups, rather than dispersing them between unimportant values when using column scan. After conversion completes, our proposed algorithm suggests that only the first eight values are retained from each 8×8 block consequently eliminates 87% of insignificant coefficients from each block. Considering our example above as show in Figure 3, the only retained values are: [11,-1,0,0,0,0,0,0].

All reduced arrays are collected and combined into one single array called auxiliary array (AA). In our case, AA is encoded using Arithmetic coding [17]. The implementation of Arithmetic coding on AA produces two outputs. The first output is the probability table used in the decompression algorithm. The second output is the main compressed data stream consist of integer data. By examining the values in the probability header produced from arithmetic coding, it clearly shows sequences of many unrepeated values which makes it difficult to compress, therefore the probability header is kept in the compressed file to be used as a reference in the decompression phase. The other part which represents the compressed data stream is subject to a second level of encoding called Hexa-Coding algorithm [12] as shown in Figure 1 phase two.

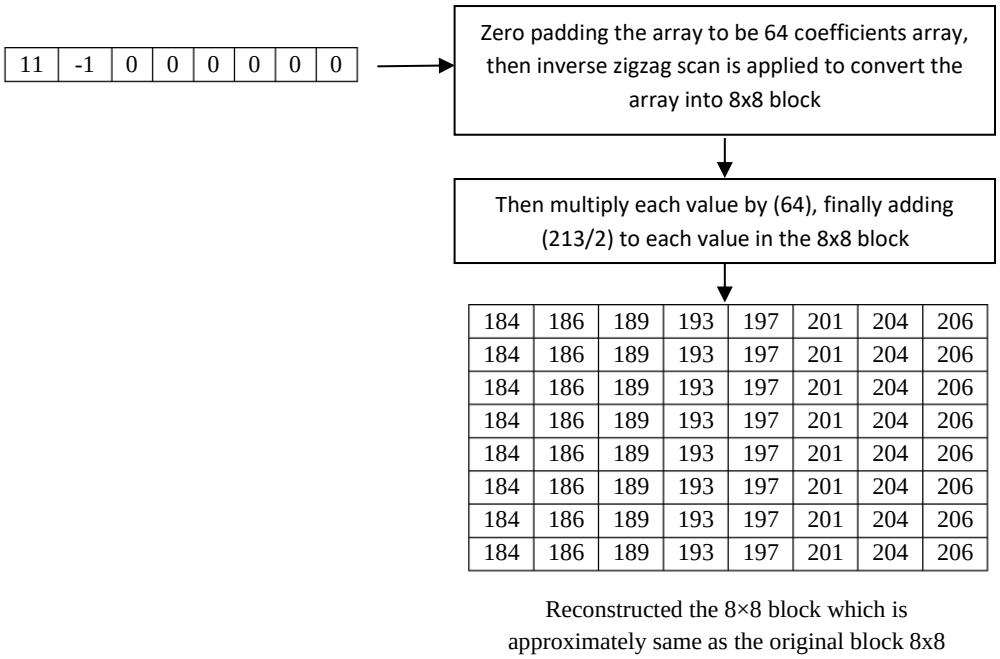


Figure 4. An 8×8 block represents inverse steps applied to the last 8 coefficients array (See Figure 3).

Hexa-Coding Algorithm is a novel lossless compression technique proposed by Siddeq and Rodrigues [11,12]. The Hexa-Coding algorithm provides a high data compression ratio with high perceptual decoded images. The novelty of this method is that it can represent every sequenced six values in an array into only one single floating-point number. This method provides a tremendous lossless compression. The algorithm also combines two functionalities within its structure, it provides data compression and encryption at the same time. The steps below explain the Hexa-Coding algorithm applied to an array:

1. Every three sequenced data are selected and represented by a single floating-point number. This is done by generating three random keys ranging between 0.01 and 0.999. Each key is multiplied by its respective value and then the results are summed together. The equation below reflects the mathematical equation of this step.

$$e(n) = K1 \times d(i) + K2 \times d(i+1) + K3 \times d(i+2) \quad (2)$$

Where e(n) is the encoded output, and n is the encoded output index. K1, K2, and K3 are key 1, key 2, and key 3 respectively. d(i) represents the original data stream, and i is the original data stream index.

2. Every two sequenced floating-point values resulted from step (1) are applied to a second level of encoding, where a second level of randomly generating keys named (K4 and K5) are generated. These keys are multiplied by the two floating-point values respectively and summed together to produce one single floating-point value. The equation below reflects step (2) mathematically and Figure 5 demonstrates the Hexa-Coding algorithm.

$$c(j) = K4 \times e(n) + K5 \times e(n+1) \quad (3)$$

Where y(j) represents the 2nd level encoded output, and j is the 2nd level encoded output index. K4 and K5 are key 4 and key 5 respectively. c(n) is the1st level encoded output, and n is the 1st level encoded output index.

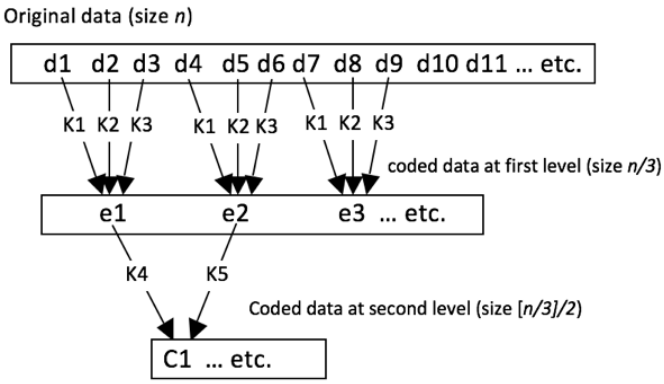


Figure 5. The Hexa-Coding algorithm layout [19]

As an example, assume we have 30 values resulted from Arithmetic coding as A =[11, 255, 34, 11, 34, 12, 10, 40, 9, 0, 90, 23, 0, 0, 20, 44, 1, 67, 11, 255, 34, 11, 34, 12, 0, 0, 20, 44, 1, 67], we apply Hexa-Coding algorithm using five randomly generated keys as: K1= 0.8147, K2= 0.9058, K3= 0.1270, K4= 0.9134, and K5= 0.9575. By using equations (2) and (3), every 6 data in A is converted to a single floating-point number. Therefore, the result for this example is represents an Encoded Array = [262.6342, 122.4339, 45.658, 262.6342, 45.658]. Up to this point, it is necessary to compute the probability of every 6 items within the array to be used later in the decompression phase. Table 1 shows the probability table based on the values of the example above. It is important to note that the probability table represents the compression-encryption keys. Therefore, it must be saved as a stream of integer keys beside the five floating-point keys to recover the image during the decompression-decryption phase.

Table 1. Probability of 30 values computed of every 6 values

Probability for every 6 values	Number of iterations of every 6 values
0, 0, 20, 44, 1, 67	2
10, 40, 9, 0, 90, 23	1
11, 255, 34, 11, 34, 12	2

The example above shows clearly the ability of the Hexa-Coding algorithm to abbreviate 30 data values into only five encrypted data. Additionally, the Hexa-coding algorithm produced a probability table represented as

a stream of integer keys. This feature in Hexa-Coding makes the algorithm more secure since in the event of losing one of these probability values the data cannot be recoverable.

3. The decryption and decompression algorithm

The proposed decryption-decompression steps are considerably straightforward and very fast compared with our proposed encryption-compression algorithm. The decryption-decompression algorithm starts by decoding the Hexa-Coding encrypted data. The decryption algorithm uses the Fast-Matching Search algorithm to search the probability table created in the encryption phase [1,11,12].

To demonstrate the decryption process, Its applied to the data generated from the example presented in Section 2. The final encrypted data (i.e. Encoded Array) was equal to [262.6342, 122.4339, 45.658, 262.6342, 45.658]. These values must be converted back (decrypted) to their exact values before Hexa-Coding encryption. The idea in the decryption is to use the probability table beside the FMSA to recover the original integer data, this can be summarized by the following steps:

1. Re-compute the output for every 6 data in the probability table using equations (2) and (3) as shown in Figure 7.
2. Sort the probability table in ascending order according to the **output** of step 1.
3. Select the ***Target*** that represents the data needed to be decoded from the compressed file and passed to the fast-matching search algorithm to lookup the probability table.
4. When matching is found between the ***T*arget** and a specific **output** from the probability table, the relevant 6 data represent the decoded values, as shown in Figure 7. These decoded values are placed in a new decoded array in the exact location of ***Target***.

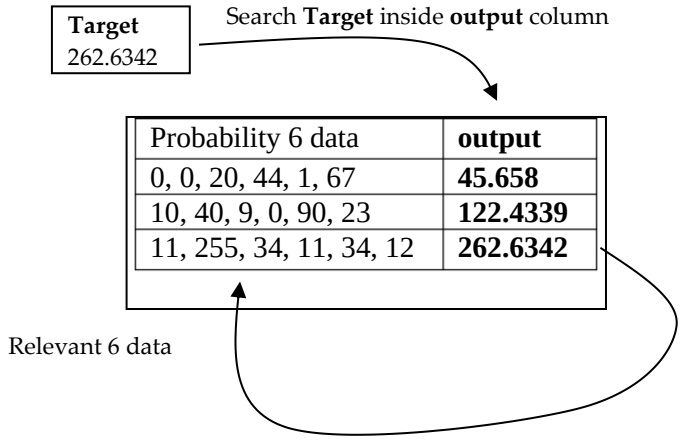


Figure 7. The decompression-decryption process

In the decryption process, there is no possibility of any mismatch since the probability table contains all possible values of every 6 values in the probability table (see Table 1). The **output** column is sorted in ascending order to facilitate and accelerate the search process. The type of search used is the Binary Search Algorithm which is a very well-known search algorithm [11,12].

After these steps, the final Hexa-Decoded data are reconstructed and decompressed using the Arithmetic decoding algorithm to obtain the values of the original coefficients. Each 8 decoded residual coefficients (presented as 8 coefficients in each block) are padded with zeros to be 64 coefficients. Thereafter, these coefficients return into 8×8 block again by using inverse zigzag scan, followed by inverse DCT to reconstruct the approximately original image (See Figure 4).

4. Experimental results and comparison with JPEG technique

Our proposed compression-encryption algorithm has been implemented using MATLAB Language in a computer using a Core-i7 processor with 2.7MHz clock frequency and 6MB of Cache Memory. The algorithm is tested using four different greyscale images along with different dimensions. At first, the gray level of the selected image is reduced using equation (1). Secondly, the quantization process whereas the maximum value is computed according to the following steps:

A) **QF** = input( ); % select a value for Quantization Factor between 0.01 to 0.9.
B) **M** = max(original image); % Find the maximum value in the image.
C) **SUQV** = round(**M** x **QF**); % SUQV ready to be used as quantization value.

Each image has its own Quantization Factor that contributes to its level of quality. The reason behind using this type of quantization is that it can increase or decrease the image quality, thus determining the compression ratio. If the Quantization Factor = 0.01 this means most of the coefficients remain approximately the same and lead to increase image quality and decrease compression ratio accordingly. Otherwise, if Quantization Factor=0.9, this represents hard quantization which leads to increase compression ratio and decrease image quality.

Table 2 shows the compressed image size for each image. Each original image was compressed three times according to three different Quantization Factors. It also shows the PSNR (Peak Signal to Noise Ratio) for each decompressed image. The PSNR is one of the popular quality measures, which is computed very easily and very widely for comparing decompressed images against original images [18].

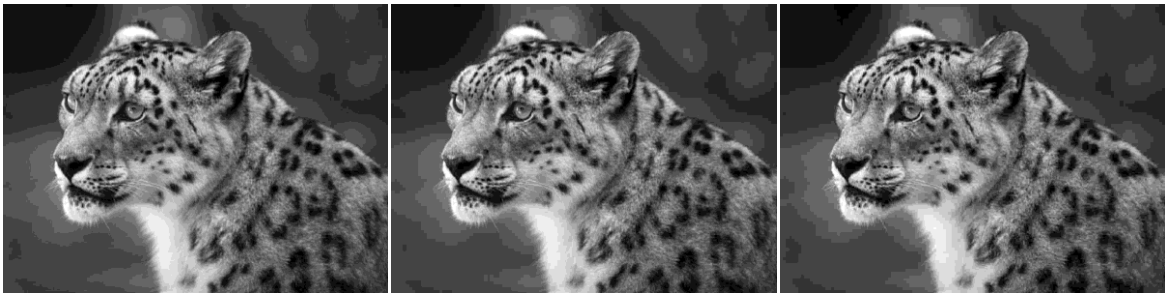
Figure 8 shows the decompressed images according to their compressed size. It illustrates the capability of the proposed algorithm for compressing images at a higher compression ratio that reaches up to 99% of the original size.

Table 2. Our proposed approach results

Image name	Image dimension	Original size	(QF)	Compressed size	BPP (bit/pixel)	PSNR
Lena	1024×1024	1 MB	0.05	56.5 KB	0.0551	42.8
			0.15	27.6 KB	0.0270	41.6
			0.3	17 KB	0.0166	40.24
Girl	1943×1534	2.84 MB	0.05	185 KB	0.0636	46.2
			0.15	35.9 KB	0.0123	43.8
			0.3	23 KB	0.0079	42.07
Tiger	1943×1534	769 KB	0.05	45 KB	0.0507	39.2
			0.15	27.9 KB	0.0363	38.9
			0.3	18.5 KB	0.0241	38.3
Woman	512×512	257 KB	0.01	30.4 KB	0.1183	35.95
			0.1	13.8 KB	0.0537	35.9
			0.3	7.3 KB	0.0284	35.6

Compressed size: 56.5 KB Compressed size: 27.6 KB Compressed size: 17 KB

Compressed size: 185 KB Compressed size: 35.9 KB Compressed size: 23 KB



Compressed size: 45 KB Compressed size: 27.9 KB Compressed size: 18.5 KB

Compressed size: 30.4 KB Compressed size: 13.8 KB Compressed size: 7.3 KB

Figure 8. Compressed images by our proposed algorithm

The proposed algorithm compared with the JPEG technique because it is the most useable compression technique in many applications. Moreover, the JPEG algorithm still in development, to achieve maximum compression ratio and high-quality images. Table 3 and Figure 9 show the comparison of the two methods, the comparison is based on image quality at a higher compression ratio.

Table 3. Comparison with the JPEG algorithm

Image name	dimension	Original size	Compressed size by our proposed algorithm	PSNR	Compressed size by JPEG algorithm	PSNR
Lena	1024x1024	1 MBytes	17 Kbytes	40.24	19 Kbytes	37.25
Girl	1943x1534	2.84 MBytes	23 Kbytes	42.07	39.7 Kbytes	37.96
Tiger	1943x1534	769 KBytes	18.5 Kbytes	38.3	21.6 Kbytes	37.82
Woman	512x512	257 KBytes	7.3 Kbytes	35.6	10.1 Kbytes	35.3

Compressed size: 19 KB Compressed size: 39.7 KB Compressed size: 18.5 KB

Compressed size: 10.1 KB

Figure 9. Compressed images by JPEG technique

Conclusion

We can summarize our proposed image encryption-compression algorithm in the following points:

1. The algorithm divides the image into 8×8 non-overlapped blocks, every block is transformed by two-dimensional DCT.
2. Single Unified Quantization Value is used in this algorithm as a new way for the quantization process. This uniform value is applied to each block independently. The performance of this type of quantization worked

successfully for balancing between image quality and compression ratio, as showed in the experimental results.

3. Each block size of 8×8 coefficients is converted to a one-dimensional array of 64 coefficients through a zigzag scan. Then 64 coefficients are reduced into 8 coefficients by keeping the first 8 coefficients which represent the most important coefficients. The main reasoning behind this reduction for increasing the compression ratio and keep the image quantity as much as possible.
4. All reduced 8 coefficients are collected, then addressed to Arithmetic coding for lossless compression.
5. Additionally, the Hexa-Coding algorithm is used to encrypt the compressed data. The feature of this algorithm is to reduce compressed data size and encrypt it at the same time with five different keys. Also, the Hexa-Coding algorithm helped our proposed method to increase image compression performance. However, the Hexa-decoding algorithm is based on a searching method that consumes time to decode data compared to JPEG technology that does not use any type of searching mechanism for decoding. Besides that, if the five keys are lost or damaged the image will be un-decodable.
6. Finally, based on our proposed image encryption-compression algorithm the experimental results show a high compression ratio of up to 99% compared to a traditional JPEG algorithm besides maintaining visual image details.

References

[1] M.M. Siddeq, M.A Rodrigues, "A novel high-frequency encoding algorithm for image compression," *EURASIP J. Adv. Signal Process, (Springer),* (2017): 26, 2017.

[2] O. M. Salih, M. H. Rasheed, M. M. Siddeq, and M. A. Rodrigues, "Image compression for quality 3D reconstruction," Journal of King Saud University - Computer and Information Sciences, Aug. 2020.

[3] M. H. Rasheed, O. M. Salih, M. M. Siddeq, and M. A. Rodrigues, "Image compression based on 2D Discrete Fourier Transform and matrix minimization algorithm," Array, vol. 6, p. 100024, Jul. 2020.

[4] S. Deb, B. Biswas, and B. Bhuyan, "Secure image encryption scheme using high efficiency word-oriented feedback shift register over finite field," *Multimedia Tools Applications*, Vol. 78, 34901–34925, 2019.

[5] P. Chaudhary, R. Gupta, A. Singh, P. Majumder, and A. Pandey "Joint image compression and encryption using a novel column-wise scanning and optimization algorithm," *Procedia Computer Science* Vol. 167, pp. 244-253, 2020.

[6] M. Zhang and X. Tong "Joint image encryption and compression scheme based on a new hyperchaotic system and curvelet transform," *Journal of Electronic Imaging* 26(4), 043008, 2017.

[7] P. Li and K. Lo, "Joint image compression and encryption based on alternating transforms with quality control," *2015 Visual Communications and Image Processing (VCIP),* Singapore, pp. 1-4, doi: 10.1109/VCIP.2015.7457867, 2015.

[8] X.Tong, M. Zhang, and Z. Wang "A joint color image encryption and compression scheme based on hyper-chaotic system," *Nonlinear Dynamics (Springer)* 84, 2333–2356, 2016.

[9] M. Zhang, and X. Tong "Joint image encryption and compression scheme based on IWT and SPIHT," *Optics and Lasers in Engineering (Elsevier),* Vol. 90, 254–274, 2017.

[10] P. Li and K. Lo "Joint image encryption and compression schemes based on 16 × 16 DCT," *Journal of Visual Communication and Image Representation (Elsevier),* 58, 12-24, 2019.

[11] M .M Siddeq, M.A Rodrigues "A Novel Hexa data Encoding Method for 2D Image Crypto-Compression," *Multimedia Tools and Applications (Springer),* 79, pp. 6045–6059, 2020.

[12] M.M. Siddeq, , M.A. Rodrigues "A Novel Method for Image and Video Compression Basedon Two-Level DCT with Hexa data Coding," *Sensing and Imaging(Springer),* 21, 36, 2020.

[13] G. K. Wallace, "The JPEG still picture compression standard," in *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii-xxxiv,1992.

[14] K.R. Rao, P. Yip, Discrete Cosine Transform. Academic Press, ISBN: 978-0-12-580203, 1990.

[15] Biju Bajracharya, and David Hua, "A Preprocessing Method for Improved Compression of Digital Images," *Journal of Computer Sciences and Applications*, vol. 6, no.1, 2018.

[16] M. Milanova, R. Kountchev, V. Todorov, R. Kountcheva, "Pre- And Post-Processing for Enhancement Of Image Compression Based On Spectrum Pyramid," *In: Sobh T., Elleithy K., Mahmood A., Karim M. (eds) Innovative Algorithms and Techniques in Automation, Industrial Electronics and Telecommunications. Springer, Dordrecht*, pp. 269-274. 2007.

[17] K. Sayood, Lossless Compression Handbook. Academic Press. ISBN: 9780126208610, 2003

[18] Abdullah Hussain, Ghadah AL-Khafaji and M. Siddeq, "Developed JPEG Algorithm applied in image compression," *IOP 2nd Conference scientific of Al-Ayen University (ISCAU-2020)*, 928(032006), 2020.

[19] M. Siddeq "JPEG and sequential search algorithm applied on low- frequency sub-band for image compression (JSS)," Journal of Information and Computing Science, 5 (3), 163-172, 2010.

[20] A. S. Abdullah, M. A. Abed, and I. Al Barazanchi, "Improving face recognition by elman neural network using curvelet transform and HSI color space," Period. Eng. Nat. Sci., vol. 7, no. 2, pp. 430–437, 2019.

[21] I. Al Barazanchi and H. R. Abdulshaheed, "Adaptive Illumination Normalization Framework based on Decrease Light Effect for Face Recognition," Jour Adv Res. Dyn. Control Syst., vol. 11, no. 01, pp. 1741–1747, 2019.