Video Encryption using Wavelet Transform with Shift Number Coding

Mohammed Mustafa Siddeq[1], Abdulrahman Ikram Siddiq[2]
Software Engineering Deprt. – Technical College/Kirkuk
[1]mamadmmx76@yahoo.com , [2]dr_aisiddiq@yahoo.com

***Abstract*:-** The nature of playing video streams over a network in real time requires that the transmitted frames are sent in a bounded delay. Then, the most suitable encryption for real time video transmission is that with the shortest execution time for a given security level. In this paper, a new video encryption algorithm is suggested for real time applications, by using the Discrete Wavelet Transform (DWT), the Shift Number Coding (SNC) and a secret key. The SNC converts each 6 bytes into a single floating point number. The time of execution and decrypted video quality of the proposed algorithm is evaluated for distinct frames of a video file. This performance is then compared with other encryption algorithms that use the Discreet Cosine Transform (DCT), Huffman coding, and Arithmetic Coding. The tests showed that the proposed algorithm is faster than the other tested algorithms. Therefore, it seem to be more suitable for real time video encryption.

***Keywords*:-** *Discrete Wavelet Transform, Shift Number Coding*

1. Introduction

The advent of networked multimedia systems made continuous media streams, such as real time audio and video, increasingly pervasive in computing and communications environments. It is thus important to secure networked media from potential threats such as hackers, eavesdroppers, etc [1,2]. The nature of playing video streams over a network in real time requires that the transmitted frames are sent in a bounded delay. Also, video frames need to be played at a certain rate. Therefore, the processes of sending and receiving encrypted packets must be performed within a certain amount of time limited by the admissible delay [3]. The size of the multimedia data before encryption is hence a concerning factor. That is, the smaller multimedia data size, the shorter time is required to encrypt/decrypt it, and vice versa. Generally, the encryption and decryption of a video stream can be performed In two ways:

1. Secret key encryption: a single secret key can be used to encrypt and decrypt the video stream. Only the sender and the receiver have this key.
2. Public key encryption: there are two keys, one for encryption and the other for decryption. The public key, which is known for all senders, is used for encryption. While the private key, which is owned by the receiver is used for decryption.

Public key cryptography is not applicable for securing real time video because its operations require an amount of time not suitable for this application. However, the idea of public key encryption is applicable for other multimedia security aspects such as signature and authentication [4]. On the other hand, secret key encryption can provide the encrypted data within a limited time [3,5]. Therefore, secret key encryption is usually used to secure real time video transmission. In the literature, video encryption techniques may be classified as Native, Selective, Zigzag, and pure permutation algorithms.

- Native Algorithms

This technique deals with the video stream as text data and encrypts every byte in the stream individually [6]. It guarantees high security level, but it is not applicable for large amounts of data, as the case of video streams, because its operations will result in a delay not suitable for real time video encryption.

- Selective Algorithms

Tang in [7] suggested employing different levels of encryption for selected parts of the video streams. This is based on the fact that the nature of the different components of a video file is not the same. Tang has suggested four levels of selective algorithms. These levels are: encrypting all headers, encrypting all headers and I frames, encrypting all I frames and all I blocks in P and B frames, and finally, encrypting all frames individually as in Native algorithms to guarantee the highest security. The encryption time is controllable and it depends on the number of used levels.

- Zig-zag Algorithms

The idea of zig-zag algorithm is basically encrypting the video stream before compressing it. Explicitly, rather than mapping an 8x8 block to a 1x64 vector each time in the same order, a predetermined permutation can be used. However, the concept of the encryption key does not exist in the zig-zag algorithm. Once the permutation list is known the algorithm is not secure any more [8].

- Pure Permutations

The idea of pure permutation is simply to apply a permutation technique for the I frames. Both the sender and the receiver have the correct permutation order to encrypt and decrypt the video stream. Later works [9,10,11] proved that it is not secure to use pure permutations.

In this paper, a new video encryption algorithm is suggested for real time applications, by using the Discrete Wavelet Transform (DWT), the Shift Number Coding (SNC) and a secret key, the proposed algorithm deals with the successive frames individually.

2. The Proposed Algorithm

The proposed algorithm deals with the successive frames of a video stream individually. That is, each frame is encrypted and decrypted independently of the other frames in the same video sequence. This property results in a small buffering memory requirement limited in size to the size of a single frame, which is suitable for real time operation.

The encryption process consists of two stages as shown in Figure. 1. Firstly, it uses the DWT to decompose a frame into its low and high frequency components. The low frequency part is encoded with the efficient coding technique SNC. Then the result is XORed with the secret key. In parallel with this operation, only the nonzero pixels of the high frequency parts, which usually consist of a great number of zeros, are XORed with the same secret key. At the receiver, the reverse operations are performed to decrypt the received frames.

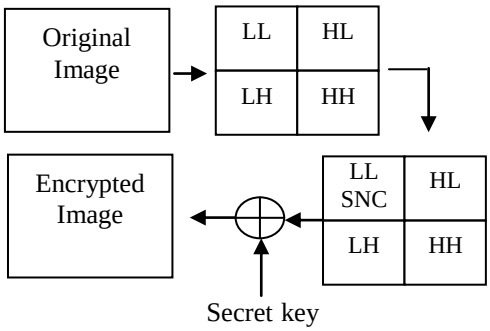
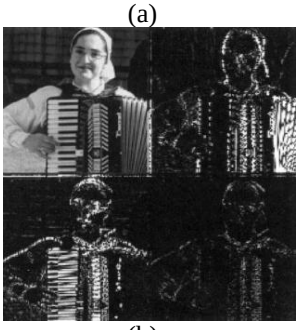


Figure – 1 Encryption Algorithm

2.1 Discrete Wavelet Transform

A single-stage wavelet transformation consists of a filtering operation that decomposes an image (frame) into four frequency bands as shown in Figure. 2. The original image is shown in Figure.2 (a), and Figure.2 (b) is the result of a single-stage wavelet transform. The top-left corner of the transformed image "LL" is the original image, low-pass filtered and sub-sampled in the horizontal and vertical dimensions. The top-right corner "HL" consists of residual vertical frequencies (i.e. the vertical component of the difference between the sub-sampled "LL" image and the original image). The bottom-left comer "LH" contains residual horizontal frequencies (the accordion keys are very visible here), whilst the bottom-right corner "HH" contains residual diagonal frequencies [11,12,13].

(a)



(b)

Figure – 2 (a) Original image (b) Single stage wavelet Transform

The reason of using the wavelet transform is to minimize the number of operation in the encryption and decryption processes. Because most transformed image regions are zeros which can be ignored when it is XORed with the secret key, as a worked example, let the 8x8 square matrix "A" represent an image, and defined as follows:

$$A=\begin{bmatrix} 136, 136, 137, 137, 139, 134, 129, 134 \\ 136, 136, 137, 137, 139, 134, 129, 134 \\ 136, 136, 137, 137, 139, 134, 129, 134 \\ 133, 140, 133, 131, 133, 128, 128, 130 \\ 128, 132, 133, 129, 138, 132, 135, 128 \\ 130, 131, 130, 126, 132, 130, 133, 131 \\ 130, 131, 130, 132, 135, 132, 128, 128 \\ 131, 133, 128, 133, 127, 122, 128, 128 \end{bmatrix}$$

The matrix A is converted to four sub-images by using the DWT. Then,

$$LL=\begin{bmatrix} 272, 274, 273\ 263 \\ 273, 269, 267, 261 \\ 261, 259, 266, 264 \\ 263\ 262, 258, 256 \end{bmatrix} \quad HL=\begin{bmatrix} 0, 0, 0, 0 \\ -1, 5, 6, 3 \\ -1, 3, 4, -1 \\ -2, 1, 9, 0 \end{bmatrix}$$

$$LH = \begin{bmatrix} 0, 0, 5, -5 \\ -3, 1, 5, -3 \\ -3, 4, 4, 4 \\ -2, -3, 4, 0 \end{bmatrix} \quad HH = \begin{bmatrix} 0, 0, 0, 0 \\ 3, -1, 0, -2 \\ -1, 0, 2, 3 \\ 0, 2, -1, 0 \end{bmatrix}$$

The matrices HL, LH, and HH are encrypted by XORing them with the secret key, while the LL matrix is first encoded by the SNC technique, and then the result is XORed with the same secret key.

2.2 Shift Number Coding

The implementation of SNC begins with reducing the number of data levels from M levels by using the following equation:

$$\text{Pixels}_{(new)} = \frac{(\text{Pixels}_{(old)} * \text{Threshold})}{\text{MAX}} \quad (1)$$

Where Threshold is the maximum value of the new pixels, and MAX is the maximum value of the old pixels.

Reduction of pixel levels leads to a reduction in the pixels size. The Threshold decides new pixels range. For example, when Threshold=60, the new pixels range becomes [0 - 60]. The same threshold value is used for all of the frames in the given video stream. Next, the new pixel values are normalized to there maximum value (Threshold) to fall within the new range [0 - 1]. Then, the interval [0 - 1] is divided into 60 parts, which means that the interval between any two pixel values is 0.016, as shown in Table 1.

Table 1 Value for each pixel

Pixels	Values	Pixels	Values
0	0.0	….	….
1	0.016	51	0.816
2	0.032	52	0.832
3	0.048	53	0.848
4	0.064	54	0.864
5	0.08	55	0.88
6	0.096	56	0.896
7	0.112	57	0.912
8	0.128	58	0.928
9	0.144	59	0.944
10	0.160	60	1.0

The SNC algorithm encodes each 6 bytes of the new pixels into a single floating number by shifting and summing pixel values. This technique does not need any statistical computations on the image file, and also does not need to store any extra information for the decoding process.

$$\text{Coded Value} = \sum_{i}^{n} \text{Shift}(i) * \text{Value}(i) \quad (2)$$

For $i = 1,2 \ldots 6$

The coded value is a single floating point number resulted from combining 6 bytes as described above. The values of Shift(i) are given in Table 2 for each index value i.

Table 2 Shift function

index	Shift(index)
1	1
2	0.01
3	0.0001
4	0.000001
5	0.00000001
6	0.0000000001

The SNC algorithm for each 6-Byte data is shown below:

Set Coded_Value to 0.0;
Set index to 1;
While (index $\leq$ 6) **Do**
  Read (symbol)
  Value = **interval** (Symbol);
  Coded_Value=Coded_Value+Shift(index)* Value;
  Index = index + 1;
End *While*

The SNC of the LL matrix resulted from the DWT decomposition of matrix A, defined before, is achieved first by changing the levels of the pixels to the new range using Threshold=60. The result is as follows

$$LL = \begin{bmatrix} 272, 274, 273, 263 \\ 273, 269, 267, 261 \\ 261, 259, 266, 264 \\ 263, 262, 258, 256 \end{bmatrix} \quad LL_{(new)} = \begin{bmatrix} 60, 60, 60, 58 \\ 60, 59, 59, 57 \\ 57, 57, 58, 58 \\ 58, 58, 57, 56 \end{bmatrix}$$

Next, when equation 2 is applied on $LL_{(new)}$ every 6 bytes (6 new pixels) are transformed to a single floating point number as shown in Table 3.

Table 3 Applied SNC

Data	Value	Shift(index)	Coded_Value
60	0.96	1	0.96
60	0.96	0.01	0.9696
60	0.96	0.0001	0.969696
58	0.928	0.000001	0.969696928
60	0.96	0.00000001	0.9696969376
59	0.944	0.0000000001	0.969696937694
Data	**Value**	**Shift(index)**	**Coded _Value**
59	0.944	1	0.944
57	0.912	0.01	0.95312
57	0.912	0.0001	0.0532112
57	0.912	0.000001	0.953212112
58	0.928	0.00000001	0.953212121279
58	0.928	0.0000000001	0.953212121372
Data	**Value**	**Shift(index)**	**Coded _Value**
58	0.928	1	928.
58	0.928	0.01	0.93728
57	0.912	0.0001	0.9373712
56	0.996	0.000001	0.937372096

Therefore the original sub-matrix $LL_{(new)}$ is converted to:

$$LL_{SNC} = \begin{bmatrix} 0.969696937694 \\ 0.953212121372 \\ 0.937372096000 \end{bmatrix}$$

The final step in the encryption process it is XOR matrices: LL_{SNC}, HL, LH, and HH with the secret key, the secret key is a 16-byte integer number. It is XORed with each 16-byte from matrices: LL_{SNC}, HL, LH, and HH. If a pixel value is zero, XOR operation ignores this pixel. For example: the matrix HL contains zeros, XOR ignored zeros, and other bytes from secret key XORed with other elements from the matrix. Also the matrix LL_{SNC} is XORed with secret key, by converting each floating point number to integer numbers.

For example: in matrix LLSNC the floating point number is [0.969696937694] convert to integer numbers as: [96, 96, 96, 93, 76,94].

The decryption algorithm is reverse for encryption, the operation start with XOR with secret key to return original matrices: LL,HL, LH, and HH. Finally the Inverse SNC applied on matrix LL to get original pixels. The Inverse SNC illustrated in the following steps, and Table 4.

While (**Coded_Value**> 0) **Do**
  For I =1 to Threshold **Do**
    IF Pixel_Value(**I**)≤ **Coded_Value**<
      Pixel_Value(**I+1**) **THEN**
        Let K= Pixel_Value(**I**);
        Pixel=Get_Pixel (**I**);
    End *IF*
  End *For*
 Coded_Value=**Coded_Value** -K;
Coded_Value=**Coded_Value***100;
End *While*

Table 4 Inverse SNC

Data Range	Value	Selected Data
[0.96 - 0.976]	0.969696937694	60
[0.96 - 0.976]	0.9696937694	60
[0.96 - 0.976]	0.96937694	60
[0.92 – 0.94]	0.937694	58
[0.96 - 0.97]	0.9694	60
[0.94-0.96]	0.94	59
Data Range	**Value**	**Selected Data**
[0.94 – 0.96]	0.953212121372	59
[0.91 – 0.92]	0.9212121372	57
[0.91 – 0.92]	0.92121372	57
[0.91 – 0.92]	0.921372	57
[0.92 – 0.94]	0.9372	58
[0.92 – 0.94]	0.92	58
Data Range	**Value**	**Selected Data**
[0.92 – 0.94]	0.937372096	58
[0.92 – 0.94]	0.9372096	58
[0.912-0.928]	0.92096	57
[0.86-0.91]	0.896	56

The LL_{SNC} matrix converts approximately to original matrix by using equation (1), the following steps illustrates conversion:

$$LL_{SNC}=\begin{bmatrix}60, 60, 60, 58\\60, 59, 59, 57\\57, 57, 58, 58\\58, 58, 57, 56\end{bmatrix}$$

→ Apply Inverse Equation (1)

$$LL_{(new)}=\begin{bmatrix}273, 273, 273, 264\\273, 268, 268, 259\\259, 259, 264, 264\\264, 264, 259, 255\end{bmatrix}$$

Finally the matrices LL, HL, LH and HH are used by Inverse DWT to compose into one matrix (Decryption Matrix), as shown below:

$$A=\begin{bmatrix}137, 137, 137, 137, 139, 134, 130, 135\\137, 137, 137,1 37, 139, 134, 130, 135\\135, 138, 137, 136, 140, 135, 130, 133\\136, 139, 132, 131, 134, 129, 127, 130\\128, 131, 133, 129, 136, 132, 134, 130\\129, 132, 130, 126, 132, 128, 135, 131\\130, 132, 131, 134, 136, 132, 128, 128\\132, 134, 130, 133, 127, 123, 128, 128\end{bmatrix}$$

Decryption matrix "A" is approximately same as original matrix, the Encryption and Decryption algorithm is lossly algorithm. Because at encryption algorithm all data are floating point numbers, and our algorithm operates on integer numbers. The conversion between integers and floating point's numbers affects on decryption algorithm, and this leads to change at image quality (See Section 3).

3. Computer Simulation

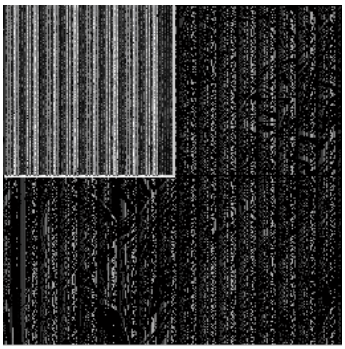
Our algorithm applied on (MATLAB Language), by using "**Pentium4 - 1.73MHz**, 1MB Cache Memory". First test for our algorithm on "Lena" image (256 x 256), at first the gray level for "Lena" image are reduced by equation(1), where MAX=256, and Threshold=55. The encrypted image shown on Figure.-3, used DWT and SNC, finally, make XORed with each sub-image, where secret key = {48, 65, 129, 51, 151, 34, 34, 167, 178, 100, 128, 9, 1, 45, 56, 48}. The decryption algorithm start with using secret key with each encrypted sub-image, then using Inverse SNC and Inverse DWT to obtain approximately original image.

Peak signal to noise ratio (PSNR) can be calculated very easily and is therefore a very popular quality measure. It is widely used as a method of comparing the "Quality" of Original and Decrypted video images [6,13]. PSNR calculated using equation (3) it is measured on a logarithmic scale and is based on the mean squared error (MSE) between an original image and decrypted image, relative to $(255)^2$ (i.e. the square of the highest possible signal value in the image).

$$PSNR = 10\log_{10}\frac{255^2}{MSE} \quad (3)$$

(a) Original Image

(b) Wavelet Transform



(c) SNC and XORed with Secret Key

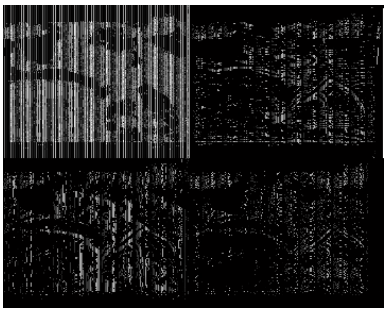
(d) Decrypted Image *PSNR=40dB*

Figure. – 3 Encryption and Decryption Algorithm applied on "Lena" image

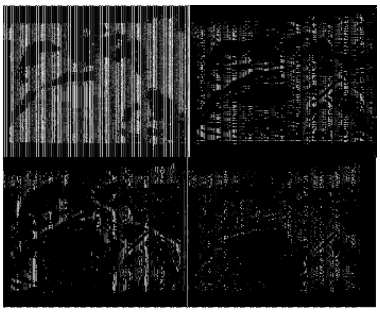
Second test for our approach on two frame of football match, frame's size (288 x 352), these frames encrypted by using same secret key, and Threshold=50, as shown on Figure - 4. Table 5 and Table 6 has shown Time execution and PSNR respectively.

(a) Original Frame 1

(b) Original Frame 2



(c) Encrypted Frame 1



(d) Encrypted Frame 2

(e) Decrypted Frame1 *PSNR=37.9dB*

(f) Decrypted Frame 2
PSNR=39.5dB

Figure. – 4 Encryption and Decryption Algorithm applied on "Frame1 and Fram2

Table 5 Time execution for our algorithm

Image Name	Encryption Time	Decryption Time	Total Time
Lena	0.178 **Sec**.	0.384**Sec.**	0.562**Sec.**
Frame 1	0.306**Sec.**	0.662**Sec.**	0.968**Sec.**
Frame 2	0.310**Sec.**	0.64**Sec.**	0.95**Sec.**

Table 6 PSNR for our algorithm

Image Name	Image Size	PSNR
Lena	64-KByte (256 x 256)	40dB
Frame 1	100-KByte (288 x 352)	37.9dB
Frame 2	100-KByte (288 x 352)	39.5dB

4. Comparison Methods

Our approach compared with (DCT), Arithmetic coding, and Huffman coding which these methods are used at most in image compression. Table 7 and Table 8 shown the comparison between our approach and DCT combined with (Arithmetic coding) and Huffman coding. Also our approach is compared with DWT combined with Arithmetic coding and Huffman coding [7,10,12,13].

Table 7 Comparison Methods with time execution

Algorithm	Image Name	Encryption Time (Sec.)	Decryption Time (Sec.)
Our Approach	Lena	0.178	0.384
	Frame1	0.306	0.662
	Frame2	0.310	0.64
DCT & Arithmetic Coding	Lena	2.71	1.138
	Frame1	3.062	0.90
	Frame2	3.0	0.92
DCT & Huffman Coding	Lena	31.7	408.53
	Frame1	52.21	508.2
	Frame2	56.21	505.2
DWT & Arithmetic Coding	Lena	1.12	0.625
	Frame1	3.45	1.0
	Frame2	3.40	1.0
DWT & Huffman Coding	Lena	29.8	212.14
	Frame1	21.9	382.9
	Frame2	20	383.0

Table 8 Comparison Methods with PSNR

Algorithm	Image Name	Total Time (Sec.)	PSNR (dB)
Our Approach	Lena	0.562	40dB
	Frame1	0.968	37.9dB
	Frame2	0.95	39.5dB
DCT & Arithmetic Coding	Lena	3.84	58.8dB
	Frame1	4.28	58.8dB
	Frame2	4.20	58.8dB
DCT & Huffman Coding	Lena	440.23	58.8dB
	Frame1	560.41	58.8dB
	Frame2	561.41	58.8dB
DWT & Arithmetic Coding	Lena	1.74	54.4dB
	Frame1	4.45	58.2dB
	Frame2	4.4	62.4dB
DWT & Huffman Coding	Lena	241.94	54.4dB
	Frame1	404.8	58.2dB
	Frame2	403	62.4dB

5. Conclusion

This research introduces a new idea for video encryption, consist from two parts: 1) DWT, used to compose an image into four sub-images. 2) SNC used to convert each 6-byte data into single floating point. Finally make XOR with secret key. Also our approach compared with DCT, Arithmetic coding and Huffman coding. The main reason for using DCT compared with DWT, because DCT and DWT used Widely in image compression [3,6,13], in this paper we use them in encryption and compare them by time execution and PSNR (See Table 7,8). The main difference between DWT and DCT, the DWT decompose one image into four sub-images, and most of sub-images regions containing on zeros [12,13]. While in DCT contains negative numbers in most regions [1,13]. In our approach the DWT take less time than DCT. Because in this paper not need make XOR with zero elements. Also compared SNC with Arithmetic and Huffman coding, the SNC not need to compute the probability for an image, while arithmetic and Huffman coding using probability computations for an image, for this reason SNC more speedily than Arithmetic coding and Huffman coding (See Table 7,8).

RERERENCES

[1] S. Li, G. Chen, A. Cheung, B. Bhargava, K.-T. Lo, "On the design of perceptual MPEG-video encryption algorithms", *IEEE Transactions on Circuits and Systems for Video Technology* 17 (2) 214–223, (2007).

[2] Chengqing Li, Guanrong Chen, "On the security of a class of image encryption schemes," in Proc. *IEEE International Symposium on Circuits and Systems 2008 (ISCAS08)*, pp. 3290-3293, 2008.

[3] Zhenyu Yang, Yi Cui, Zahid Anwar, Robert Bocchino, Nadir Kiyanclar, Klara Nahrstedt, Roy H. Campbell, William Yurcik, "Real-Time 3D Video Compression for Tele-Immersive Environments", *in Proc. of SPIE/ACM Multimedia Computing and Networking (MMCN'06),* San Jose, CA, 2006.

[4] C. Li, S. Li, G. Chen, G. Chen, L. Hu, "Cryptanalysis of a new signal security system for multimedia data transmission", *EURASIP Journal on Applied Signal Processing* 1277–1288, 2007.

[5] Klara Nahrstedt, Lintian Qiao, "Multimedia Secure Gateway", *in Proc. of Multimedia Security Workshop, IEEE International Conference on Multimedia Computing and Systems (ICMCS'98),* pp. 1-8, Austin, TX, June, 1998.

[6] K. Sayood, *Introduction to Data Compression*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, 2nd dition, 2000.

[7] Witten, Ian H., Neal, Radford M., and Cleary, John G. , "Arithmetic Coding for Data Compression", *Communications of the ACM,* pp 520-540, June, 1987.

[8] Lintian Qiao, Klara Nahrstedt, Ivan Tam, "Is MPEG Encryption Using Random Lists instead of Zig Zag Order Secure?", in Proc. of IEEE International Symposium on Consumer Electronics '97, Singapore, December, 1997.

[9] G. Alvarez, S. Li, L. Hernandez, "Analysis of security problems in a medical image encryption system", *Computers in Biology and Medicine* 37 (3) 172 424–427, 2007.

[10] J. Zhou, Z. Liang, Y. Chen, A. O. C., "Security analysis of multimedia encryption schemes based on multiple Huffman table", *IEEE Signal Processing Letters 14* (3)166 (2007) 201–204.

[11] N. Saito and J.-F. Remy, ."A new local sine transform without overlaps: A combination of computational harmonic analysis and PDE", in *Wavelets: Applications in Signal and Image Processing X*, M. A. Unser, A. Aldroubi, and A. F. LaineEds, vol. Proc. SPIE 5207, pp. 495.506, 2003.

[12] S. G. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet thresholding for image denoising and compression",. *IEEE Trans. Image Process.*, vol. 9, no. 9, pp. 1532.1546, 2000.

[13] Rafael C. Gonzalez, Richard E. Woods *Digital Image Processing*, Addison Wesley publishing company – 2001.

First Author: He is works on image processing lab. and programming lab. at software Engineering Department in Technical College – Kirkuk –Iraq. He is finished his (B.Sc.) degree at Alnahhrain University –Baghdad-Iraq, and he is finished his (M.Sc.) degree at university of Technology at same country. His fields; on Digital Image Processing, Cryptography, Neural Network, Genetic Algorithms and Algorithm Design.

Second Author: He is head of department of software Engineering Department in Technical College – Kirkuk – Iraq. He is finished his (B.Sc.) degree at Alnahhrain University –Baghdad- Iraq, and he is finished his (M.Sc.) and (PhD) degree at same university. His fields; on Digital Image Processing, Cryptography, FPGA and DSP.