ALGORITHM FOR IMAGE MIXING AND ENCRYPTION

Ayman M. Abdalla[1] and Abdelfatah A. Tamimi[2]

[1]Dept. of Multimedia Systems, Al-Zaytoonah University, Amman, Jordan
ayman@zuj.edu.jo
[2]Dept. of Computer Science, Al-Zaytoonah University, Amman, Jordan
drtamimi@zuj.edu.jo

ABSTRACT

This new algorithm mixes two or more images of different types and sizes by employing a shuffling procedure combined with S-box substitution to perform lossless image encryption. This combines stream cipher with block cipher, on the byte level, in mixing the images. When this algorithm was implemented, empirical analysis using test images of different types and sizes showed that it is effective and resistant to attacks.

KEYWORDS

Cryptography, Stream Cipher, Block Cipher, S-box

1. INTRODUCTION

Algorithms applying different encryption techniques with shuffling were presented in previous work [1, 2, 3, 4, 5, 6, 7]. Examples on applying the four steps of the Advanced Encryption Standard (AES) including the use of S-box substitution are available [8]. Many encryption algorithms based on AES were also developed [9, 10, 11, 12, 13]. However, AES has limitations on some multimedia specific requirements [7, 14], so other encryption algorithms need to be developed.

A new algorithm is presented, which concatenates two or more images of different types and sizes and performs lossless mixing and encryption in three steps. These steps include a shuffling step and a substitution step, combining stream cipher with block cipher. The algorithm was implemented and tested. Analysis showed effectiveness of the cipher and its resistance to attacks.

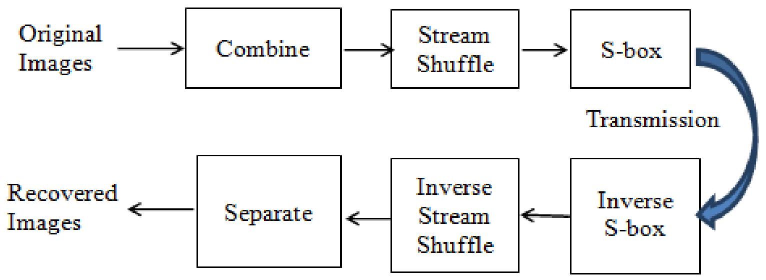


Figure 1. Diagram showing the main steps of the algorithm

2. THE NEW ALGORITHM

This algorithm takes two or more images and a private key as input, and it works as follows. It starts with concatenating the input images. Then, it performs byte shuffling of the combined result. Finally, it applies byte substitution using a lookup table called S-box. The main encryption and decryption steps of the algorithm are illustrated in Figure 1, where the decryption performs the inverse of the encryption steps in reverse order. The details of the encryption steps are outlined in Figure 2.

```
ImageV = Concatenation of input images into a single one-dimensional vector

key2 = vector initialized with each value key2[j] is the jth byte of the original key
For i = 1 to k
    keySum = the sum of the first (i) elements of key2
    fixBit = keySum MOD 8
    D = vector where D[j] is the value of bit (fixBit) of the jth byte of ImageV
    S0 = vector containing numbers of ImageV bytes (j) that have (D[j] == 0)
    S1 = vector containing numbers of ImageV bytes (j) that have (D[j] == 1)
    Shuffle = concatenation of S0 with S1
    Substitute the bytes of ImageV so that the new location of byte (j) is byte (Shuffle[j])
End For

Generate S-box table
Substitute the bytes of ImageV based on the S-box table
```

Figure 1. The Encryption Algorithm

The first step of the encryption algorithm transforms the input images into one-dimensional (1D) arrays and concatenates them. This allows combining images of different sizes and types, including the combination of color images with grayscale images.

In the second step of the algorithm, the concatenated images are regarded as one stream of bytes, and the encryption performed is both key dependent and data dependent. A single bit, call it *fixBit*, is chosen by a function based on the key. In this paper, this function adds some digits of the key and takes the remainder of dividing this sum by 8. A shuffle vector is constructed by listing the numbers of bytes which have the value of bit number *fixBit* equal to zero, followed by the numbers of bytes which have the value of bit number *fixBit* equal to one. This vector gives a mapping that specifies the new location of each byte in the array. This step is repeated for several iterations. Each iteration uses a different *fixBit* and applies the same steps to the new array that resulted from the preceding iteration. The number of iterations, *k*, is a small number chosen by a key-dependent function.

The third and final step uses a substitution table, known as S-box, constructed to perform two transformations: multiplicative inverse and affine transformation. This nonlinear key-dependent substitution was presented as a step in each iteration of the AES algorithm [8]. In the new algorithm presented here, however, this substitution is performed only once.

Consider this simple example that shows how the encryption is applied. First, let the input be the two vectors (245, 45) and (163, 140). The binary representation of the combined input is: *ImageV* = (11110101 00101101 10100011 10001100). The shuffle step is applied as follows. Let the number of iterations be *k*. In the first iteration, let *fixBit* = 3 and this bit is underlined in the binary representation above. Based on the values of this *fixBit*, *S*0 = (1, 3) and *S*1 = (2, 4), which make the shuffle vector (1, 3, 2, 4). Then, the input after the shuffle substitution will be *ImageV* = (11110101 10100011 00101101 10001100). This process is repeated for all *k* iterations. After that, the S-box is generated to perform two transformations: multiplicative inverse and affine transformation. S-box substitution is finally applied where each of the byte values of the last *ImageV* vector is replaced with its lookup value indicated by the S-box table.

```
Generate Inverse S-box table
Substitute the bytes of ImageV based on the Inverse S-box table

key2 = vector initialized with each value key2[j] is the j^th byte of the original key
For i = k to 1 step -1
    keySum = the sum of the first (i) elements of key2
    fixBit = keySum MOD 8
    D = vector where D[j] is the value of bit (fixBit) of the j^th byte of ImageV
    S0 = vector containing numbers of ImageV bytes (j) that have (D[j] == 0)
    S1 = vector containing numbers of ImageV bytes (j) that have (D[j] == 1)
    Shuffle = concatenation of S0 with S1
    Substitute the bytes of ImageV so that the new location of byte (Shuffle[j]) is byte (j)
End For

Separate ImageV into the original images and change them back into proper dimensions
```

Figure 2. The Decryption Algorithm

As seen in Figure 3, the decryption algorithm is similar to the encryption algorithm, where each of the above steps can be easily inverted. This decryption restores the original images without any loss.

The algorithm is intended to be used for encrypting two or more images, but it can be also used for encrypting files other than images. Therefore, information about the encrypted images, such as their dimensions, may be stored in a file and encrypted with the images. The size of this information file may be fixed as a constant, or it may be of variable size where its size is appended to the original private key.

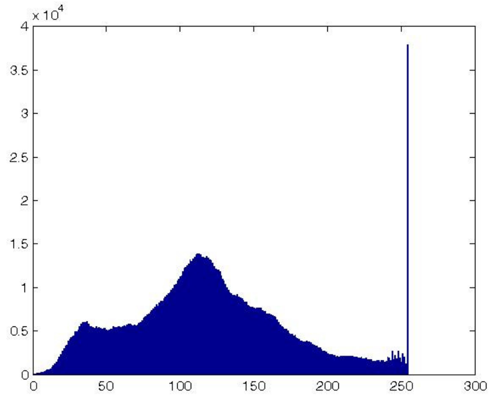
3. Implementation and Analysis

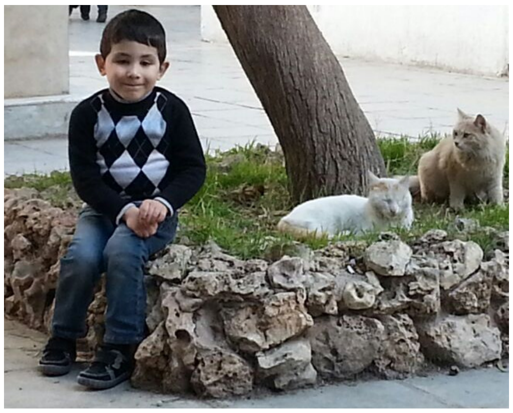
The security of the new algorithm comes from mixing any number of images of different types and sizes, on the byte level, using the shuffle operation and the S-box byte substitution. If one or more bits in the key are changed, a different shuffle bit is chosen in the Stream Shuffle step and the substitution is changed. For the Stream Shuffle step, there are $k \times 2^b$ different possible shuffle vectors for an input of size b bytes encrypted in k iterations. In addition, let an S-box of size 16×16 bytes be used in the S-box substitution step. This S-box has 2,048 different entries where each of these entries consists of 8 bits. This makes the total number of permutations for this step equal to 2^{11}. Consequently, for an input with a size of ten or more kilobytes, a brute-force attack is impossible.

The algorithm was applied to all 780 possible combination pairs of 40 different images of various types, with sizes ranging from 10 to 2000 kilobytes (kB). When different keys were used with the same image pair, they produced different encrypted images. In addition, analysis using histograms, correlation and peak signal to noise ratio (PSNR) showed properties of the algorithm that strongly resist statistical attacks. These statistics were computed by considering the original concatenated pair of images as one original image, and the encrypted mixed image as the result. After decryption, all original images were recovered without any loss.

The dimensions of each image 2D-matrix are given by its length and width (in pixels), where color images use a third dimension to let each primary color have its own 2D-matrix. The combined image pair is a 1D-matrix which includes all pixel data of both images. After Stream Shuffle is applied to this original combined image, the resulting stream is treated as a 2D-matrix during S-box substitution. The final encrypted result is one 2D-matrix.

(a) Original Zaid image


(b) Histogram of original Zaid image


(c) Original Ghaith image

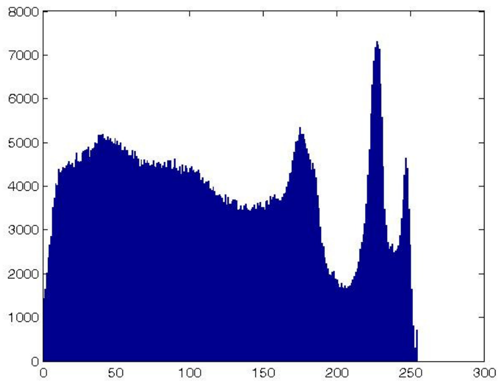

(d) Histogram of original Ghaith image

Figure 3. Original sample images and their histograms

The algorithm was tested with the sample image pair shown in Figure 4 (a) and (c) to demonstrate its results visually. The histograms of these two images are shown in Figure 4 (b) and (d), where the histogram of their combined image (before encryption) is shown in Figure 5. These two input images have different dimensions and sizes. The Zaid image is 652×752 pixels with a size of 1,437 kB. The Ghaith image is 648×518 pixels with a size of 984 kB. These two images were combined, mixed and encrypted together. The image resulting from the encryption is shown with its histogram in Figure 6. As seen in the figure, the encrypted image appears as simple noise and has no recognizable parts. In addition, the histogram of the encrypted image appears relatively uniform and has no resemblance to the histograms of the original images shown in Figure 4 (b) and (d) nor to their combined histogram shown in Figure 5.

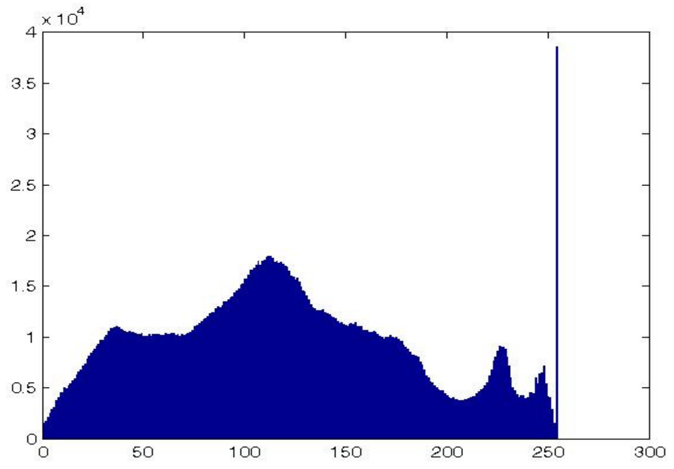
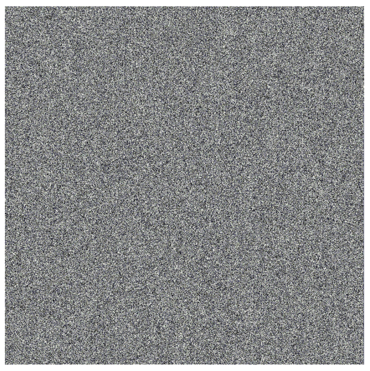
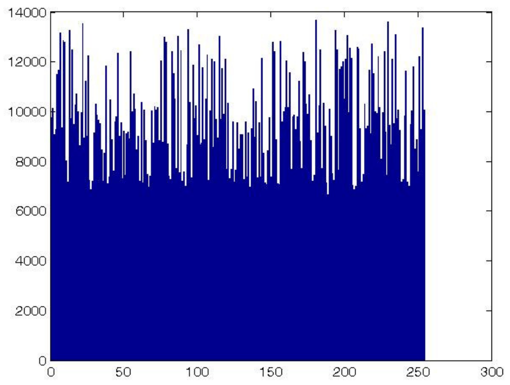


Figure 5. Histogram of image combined from original Zaid and Ghaith images



(a) Combined image after encryption



(b) Histogram of encrypted combined image

Figure 4. Encrypted combined image and its histogram

The histograms of the 780 combinations of images encrypted with the new algorithm were uniform and visibly different from the histograms of the original individual images and combined original image pairs. They gave no indication that may help statistical attacks.

The mean squared error for two images, stored in matrices A and B, is computed as follows:

$$MSE = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}(A[i,j]-B[i,j])^2 \qquad (1)$$

PSNR is computed as:

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

where MAX is the maximum pixel value of the image; usually 255. The PSNR for the encrypted sample pair (Zaid and Ghaith) was 8.422. The average PSNR computed for all encrypted pairs of images was 7.331. This low value is desired for encrypted images since it indicates more noise and, therefore, more resistance to attacks.

The correlation, r, between two images stored in matrices A and B is computed as follows, where $\overline{A}$ and $\overline{B}$ are mean values for matrices A and B, respectively:

$$r = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}(A[i,j]-\overline{A})(B[i,j]-\overline{B})}{\sqrt{\left(\sum_{i=1}^{m}\sum_{j=1}^{n}(A[i,j]-\overline{A})^2\right)\left(\sum_{i=1}^{m}\sum_{j=1}^{n}(B[i,j]-\overline{B})^2\right)}} \qquad (3)$$

The correlation between the combined sample pair (Zaid and Ghaith) and its encrypted image was 0.004. The average correlation value, taken for the absolute values of correlation for the sample image pairs, was 0.007. This low correlation value between the original images and their encryption indicates less resemblance between them, which provides more resistance to attacks.

4. CONCLUSIONS

A new encryption algorithm was presented. The new algorithm combines two or more images of different types and sizes, and it performs encryption using a shuffling procedure and an S-box substitution. These encryption procedures combine stream cipher with block cipher, and they are both private-key dependent and data dependent. Statistical analysis using histograms, PSNR and correlation showed the algorithm is not vulnerable to statistical attacks. When the algorithm was implemented and tested, the PSNR values of encrypted images and the correlations between images and their encryption were low, which indicates more noise and less resemblance between the images and their encryptions. In addition, the huge number of possible keys and possible permutations from shuffling and substitution make a brute-force attack on the algorithm impossible.

REFERENCES

[1] Arroyo, D., C. Li, S. Li, G. Alvarez & W.A. Halang, (2009) "Cryptanalysis of an image encryption scheme based on a new total shuffling algorithm"; Chaos, Solitons & Fractals, Vol. 41, No. 5, pp2613–2616. DOI: 10.1016/j.chaos.2008.09.051

[2] Bani Younes, M.A. & A. Jantan, (2008) "An image encryption approach using a combination of permutation technique followed by encryption"; Int. J. Comp. Sci. Net. Sec., Vol. 8, No. 4, pp191–197.

[3] Gao, T. & Z. Chen, (2008) "A new image encryption algorithm based on hyper-chaos"; Physics Letters A, Vol. 372, pp394-400.

[4] Sasidharan, S. & D.S. Philip, (2011) "A fast partial image encryption scheme with wavelet transform and RC4"; Int. J. Advances Eng. & Tech, Vol. 1, No. 4, pp322-331.

[5] Yahya, A. & A. Abdalla, (2008) "A shuffle encryption algorithm using S-box"; J. Comp. Sci. (Science Publications), Vol. 4, No. 12, pp999-1002.

[6] Yahya, A. & A. Abdalla, (2009) "An AES-based encryption algorithm with shuffling"; Proc. 2009 Int. Conf. Security & Management (SAM '09), Las Vegas, NV, USA. 13-16 July.

[7] Yoon, J.W. & H. Kim, (2010) "An image encryption scheme with a pseudorandom permutation based on chaotic maps"; Commun. Nonlinear Sci. Numer. Simulat. DOI: 10.1016/j.cnsns.2010.01.041

[8] Federal Information Processing Standards (FIPS 197). The Advanced Encryption Standard, 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[9] Benabdellah, M., M.M. Himmi, N. Zahid, F. Regragui & E.H. Bouyakhf, (2007) "Encryption-compression of images based on FMT and AES algorithm"; Appl. Math. Sci. (Hikari Ltd.), Vol. 1, No. 45, pp2203–2219.

[10] Duc, D.A., T.M. Triet & L.H. Co, (2002) "The extended Rijndael-like block ciphers"; Proc. Int. Conf. Info. Tech.: Coding and Computing, pp183–188. DOI: 10.1109/ITCC.2002.1000384

[11] El-Fishawy, N. & O.M. Abu Zaid, (2007) "Quality of encryption measurement of bitmap images with RC6, MRC6, and Rijndael block cipher algorithms"; Int. J. Net. Sec. (Femto Technique Co.), Vol. 5, No. 3, pp241–251.

[12] Zeghid, M., M. Machhout, L. Khriji, A. Baganne & R. Tourki, (2007) "A modified AES based algorithm for image encryption"; Int. J. Comp. Sci. & Eng. (World Academy of Science, Engineering & Technology), Vol. 1, No. 1, pp70-75.

[13] Zeghid, M., M. Machhout, L. Khriji, A. Baganne & R. Tourki, (2007) "A modified AES based algorithm for image encryption"; Enformatika (World Enformatika Society), Vol. 21, pp206-211.

[14] Socek D., S. Magliveras, D. C'ulibrk, O. Marques, H. Kalva & B. Furht, (2007) "Digital video encryption algorithms based on correlation-preserving permutations"; EURASIP J Inform. Security.

Authors

Dr. Ayman M. Abdalla has been a member of the Faculty of Science and Information Technology at Al-Zaytoonah University since 2001, where he held different positions including the Chair of the Department of Multimedia Systems. He received his Ph.D. in computer science from the University of Central Florida, FL, USA; and his Master's and Bachelor's degrees in computer science from Montclair State University, NJ, USA. He has experience in research and teaching in the United States and Jordan in addition to working in software development in a company in the United States.

Dr. Abdelfatah A. Tamimi has been a member of the Faculty of Science and Information Technology at Al-Zaytoonah University since 1996, where he held different positions including the Dean of the Faculty and the Chair of the Department of Computer Science. He received his Ph.D. in computer science from the City University of New York, NY, USA; his Master's degree in computer science from Montclair State University, NJ, USA; and his Bachelor's degree in mathematics from Jordan University, Amman, Jordan. In addition to his research and teaching experience, he has a 13 year experience in information technology design, development and implementation in United States companies.