ANALYSIS PERFORMANCE OF FAST IMAGE ENCRYPTION

Weny M. Rahmawati[1)], Matthew N. Dailey[2)]
[1)]Teknik Informatika, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
[2)]Computer Science, Asian Institute of Technology, Pathumthani, Thailand
e-mail: wenymistarika@gmail.com[1)]

ABSTRAK

Perkembangan teknologi mengakibatkan peningkatan kebutuhan pengiriman data melalui media internet. Banyak pengiriman data yang membutuhkan keamanan dalam pengirimannya untuk berbagai keperluan. Enkripsi data merupakan salah satu topic pengamanan yang banyak dilakukan penelitian dengan tujuan untuk mengamankan data yang dikirimkan melalui media internet. Salah satu data yang banyak digunakan adalah data citra. Citra merupakan data yang memiliki kapasitas besar dan memiliki sifat perulangan yang tinggi sehingga dibutuhkan metode tertentu untuk melakukan proses enkripsi dan dekripsi citra. Permutasi dan difusi merupakan cara yang banyak digunakan untuk melakukan enkripsi citra. Permutasi bertujuan untuk mengacak posisi citra sedangkan difusi merubah nilai citra. Permutasi dan difusi banyak dilakukan sebagai dua tahap yang berbeda sehingga dibutuhkan dua kali pembacaan citra. Sebuah algoritma untuk menggabungkan proses permutasi dan difusi sehingga hanya diperlukan satu kali pembacaan citra untuk melakukan enkripsi telah diajukan. Selain permutasi dan difusi, fungsi chaos *juga digunakan dalam algoritma tersebut karena kemampuannya untuk menghasilkan angka random yang sangat sensitif terhadap beberapa parameter. Dengan ide demikian, algoritma akan cepat untuk melakukan proses enkripsi dan dekripsi. Dalam penelitian ini dianalisis kinerja algoritma gabungan permutasi dan difusi menggunakan fungsi* chaos*. Analisis dilakukan dengan mengimplementasikan algoritma, mendapatkan waktu yang dibutuhkan untuk proses enkripsi dan dekripsi serta membandingkannnya dengan algoritma baku yang telah banyak digunakan, Advanced Encryption Standart (AES).*

***Kata Kunci**: citra, fungsi* chaos*, difusi citra, enkripsi citra, permutasi citra.*

ABSTRACT

Rapid development on computer network causes increasing data transfer needs via internet. The need of transfer security start come from business or personal application. Data encryption is one of security topics that become the concern by many researchers. Many algorithms and schemes are invented to get the better security and performance. Image as one type of data also get much concern since the necessary of transfer image data are increasing. Image data has special type such as bulky data capacity and high redundancy, causes many researchers attempt to invent different type of encryption algorithm. Permutation and diffusion are common ways to encrypt an image. Permutation's goal is to make the location of image value scramble, whereas diffusion goal is to change the value of the image pixel. Permutation and diffusion are two different processes. Many researchers want to use permutation and diffusion in encryption scheme. A research proposes an algorithm chaos-based and combines permutation and diffusion to encrypt image in one time pass. Means, permutation and diffusion is done simultaneously so only need to read the image one time. Chaos based is used because it produces pseudorandom numbers that very sensitive to some parameters. With this idea, the encryption and decryption claimed as a fast encryption scheme. In this paper analyze the performance of that particular algorithm by implementing the algorithm, find the running time and compare it with standard encryption algorithm, Advanced Encryption Standard (AES).

***Keywords**: chaos function, image diffusion, image encryption, image permutation.*

I. INTRODUCTION

RECENT rapid developments in computer network technology have resulted in increasing information transmission. One kind of information commonly transmitted on the internet is image data. Image data is used in many business and personal applications. In some cases, it need secure image data transmission over open networks possibly hosting adversaries. This requirement has led to a great deal of research proposing algorithms to encrypt image data.

Image has special characteristics such as a large number of bits, high correlation of adjacent pixels, and special storage formats. Some researcher believe that traditional block cipher based encryption schemes, such as Data Encryption Standard (DES), International Data Encryption Algorithm (IDEA), and Advanced Encryption Standard are not suitable for image encryption, and that specialized encryption techniques are needed [1]. One of techniques that are widely used is chaos-based method, which relies on ergodicity, sensitivity to control parameter, and sensitivity to initial condition. There are many chaos based techniques for image encryption. One of chaos-based technique is the chaotic map, which has several varieties such as Arnold's cat map, Baker's map, the logistic map, and the tent map. One example of chaos function for image encryption is used to permute the location the image pixels [2]. Other work is image encryption algorithm using the logistic map [3].

Another common way to encrypt image is by permutation and diffusion stage. The general permutation-diffusion architecture for chaos based image encryption was employed [4-7]. The goal of permutation is to reorder the location of image pixels, whereas diffusion is to transform the pixel values. Permutation and diffusion are two different stages, so to use both of these stages it needed to scan the image twice. A research proposes combination of permutation and diffusion in one time pass and uses nearest neighboring coupled map lattice to encrypt images, claimed that the process of encryption is fast but did not show the running time of algorithm [8]. This paper purposes to implement that particular algorithm, find the running time and compare it with AES running time performance.

II. A NEW CHAOS BASED FAST IMAGE ENCRYPTION ALGORITHM

Images have large capacity. Reading the same data twice might be more expensive than doing it once, even when same operations are being performed in the two steps. To solve this problem, research [8] propose the simple idea of combining permutation and diffusion in the same step. Permutation and diffusion are done together so that only one pass through the image is required. First, the image is divided into blocks of pixels. Then, spatiotemporal chaos is used to shuffle the blocks and to change pixel values.

A. Advanced Encryption Standard (AES)

AES is a symmetric key block cipher published by National Institute of Standards and Technology (NIST) in 2001. As a block cipher, AES takes a plaintext block size of 128 bits, or 16 bytes. The length of the key used in AES can be 16, 24, or 32 bytes (128,192, or 156 bits). The algorithm can thus be referred to as AES-128, AES-192, or AES-256, depending on the length of the key.

There are some modes that can be used to operate block cipher. These modes are used as a way how to interact the encryption for each block. Cipher block chaining (CBC) is one of mode that used to encrypt blocks. In CBC mode, each block of the plaintext is first XOR-ed with the previous cipher text block before being encrypted. For the first block, the block is XOR-ed with an initial vector that must be known by both sender and receiver.

Since there are modes to operating blocks, AES also use these modes in implementation. AES can be used to encrypt such a large data by divide them into blocks. The example of large data is image data.

B. Chaotic Tent Map

The chaotic tent map is a way to map a real number to another real number based on function.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Fig. 1 S-box: substitution values for bytes xy (in hexadecimal format)

The function used for tent map chaos is shown by (1). The chaotic tent map is used because it is very sensitive to the initial condition and parameter control.

$$x_{n+1} = f(x_n) = \begin{cases} \dfrac{x_n}{p} & jika 0 < x_n < p \\ \dfrac{1-x_n}{1-p} & lainnya \end{cases} \quad (1)$$

where $i=0,1,\ldots,n$ is index sequence, x_0 is the initial value, whereas b is the control parameter. The result of tent map strongly influence by x_i and b. If the equation is repeated n times, it will make output an n-element sequence. The resulting sequence can be used to scramble another sequence with the same length.

C. Nearest-Neighboring Coupled-Map Lattice (NCML)

NCML is a general nearest-neighboring spatiotemporal chaos system, can be described by (2).

$$x_{n+1}(i) = (1-\varepsilon) f(x_n(i)) + \varepsilon f(x_n(i+1)) \quad (2)$$

where $n=1,2,\ldots$is the time index, $i=1,2,..,N$ is the lattice state index, f is a chaotic map which in this case use tent map, and $\varepsilon \in (0,1)$ is coupling constant.

D. Generate Pseudorandom Number

There are many ways to generate pseudorandom number to encrypt data. This pseudorandom number is very important so the encryption algorithm is not easy to be attacked. In research [8], generating pseudorandom number is done by following steps.

1. Iterate NCML once as Equation 2 and extract 16 bits (9^{th} to 24^{th} bits after decimal point). The resulting bits is denoted as A[0], A[1], …, A[16].
2. Generate 16 numbers, S[0],S[1],… S[15] from A[0], A[1], …, A[16] according this pseudo-code.

```
flag = 0;
FOR ( i = 0 up to 15)
 IF ( i mod 4 = 0)
        S[flag] = (((A[i] ∧ A[i+1]) ∨ ((¬A[i]) ∧ A[i+2]))+A[i+3]) mod 256
 ELSIF (i mod 4 = 1)
```

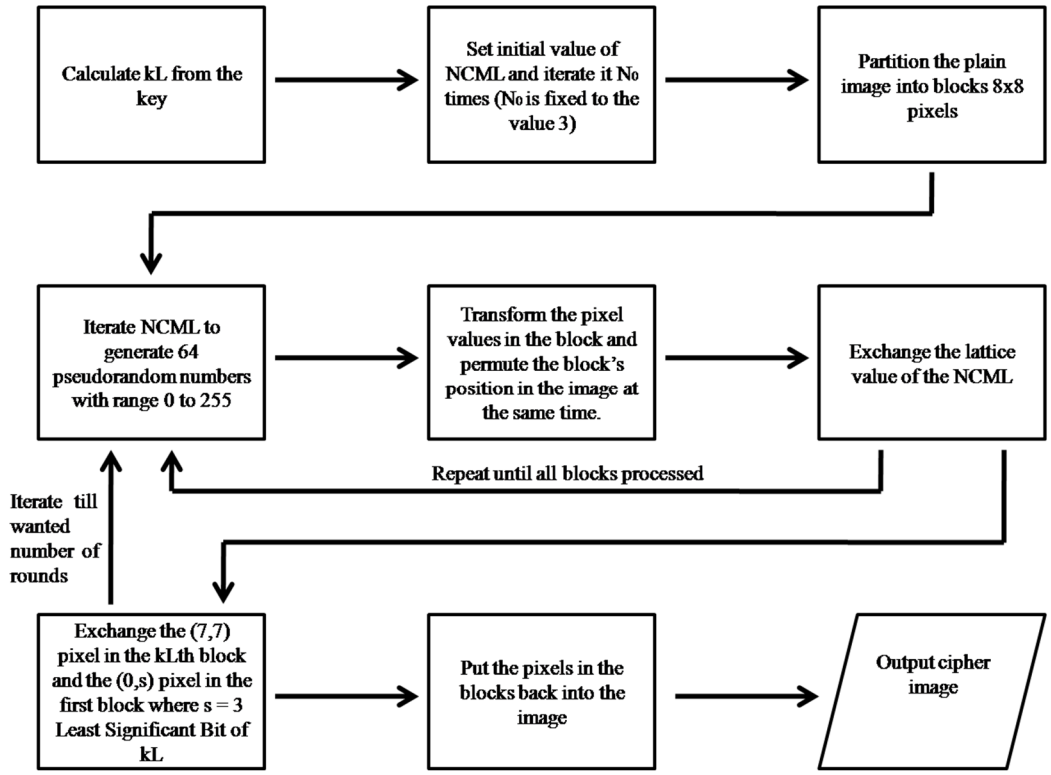

Fig. 2 Flowchart of research [8] encrytption scheme

S[flag] = (((A[i] ∧ A[i+2]) ∨ (A[i+1] ∧ (¬A[i+2]))) + A[i-1]) mod 256
ELSEIF (i mod 4=2)
S[flag]=((A[i] xor A[i+1] xor A[i-2]) + A[i-1]) mod 256
ELSIF (i mod 4=3)
S[flag]=((A[i-3] xor (A[i] ∨ (¬A[i-2]))) + A[i-1]) mod 256
ENDIF
flag=flag+1;
ENDFOR
Change the value of A.
A[0] → A[1], A[1] → A[2], ..., A[14] → A[15], A[15] → A[0]
Repeat step 2 till get 64 pseudorandom numbers.

3. Substitute the value of S from step 2 to S-box which is used in AES algorithm. S-box can be found in Fig. 1. Every first 4 bits matches as row and the 4 last bits is matches with column of S-box. The resulting of substitution are noted as S_subs.
4. The last step to generate pseudorandom number is produce by according pseudorandom.
FOR (i=0 up to 63)
Random[i]=((S_subs[i] xor S_subs[(i+1) mod 64]) + (S_subs[(i+2)mod64]) xor (S_subs[(i+3) mod 64])) mod 256
ENDFOR

The pseudorandom number generator will be used for each block of image.

III. METHODOLOGY

The methodology of implementation consist of encryption and decryption algorithm. Most of encryption and decryption stages are explained in pseudocode.

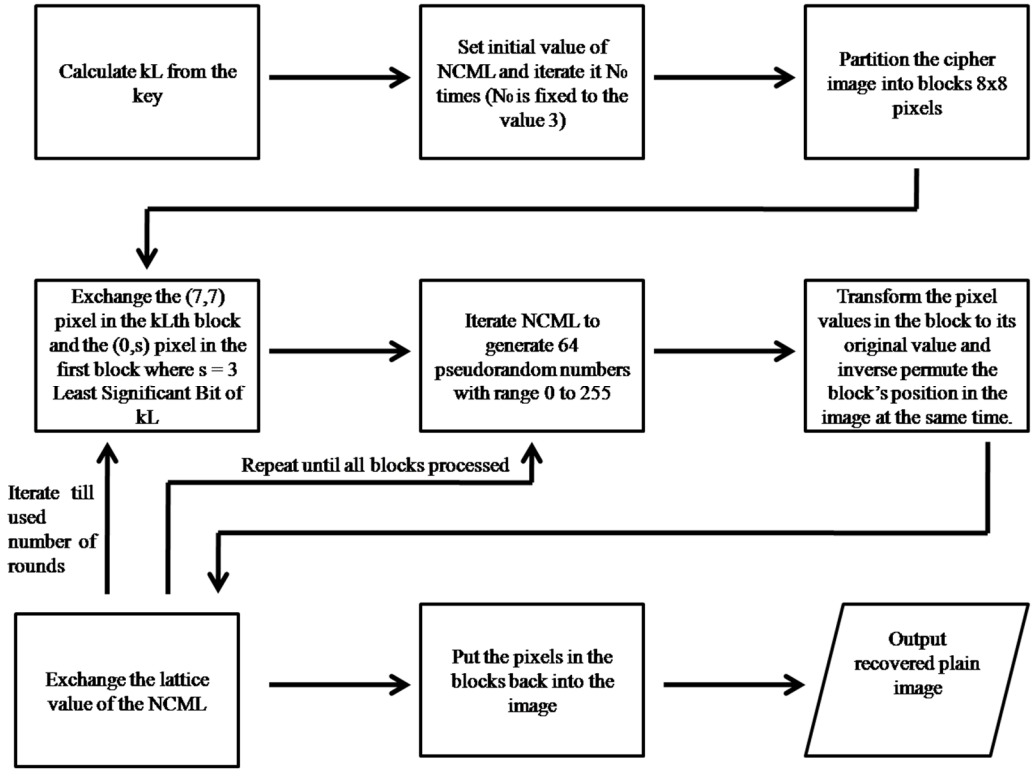

Fig. 3 Flowchart of research [8] decryption scheme

A. Encryption Image

In the encryption scheme, the input of the algorithm is the plain image and used key. The key must be known by both sender and receiver. The key consist of 128 bit which divide into K[0], K[1],…, K[15]. The flowchart of whole encryption scheme is shown in Fig.2. Detail algorithm of encryption is explain as below.

1. Calculate the value of kL and r from the key.
   kL= floor (K[0] xor K[1] xor ... xor K[15]) * num/256)
   r=(K[8] + K[9] + ... + K[15]) mod (H*W)
   where H and W are heigh and width of plain image, respectively. Whereas num = (WxH)/64 is the number of blocks that produced by plain image.
2. Get initial values of NCML process.
   FOR (i=0 up to 7)
   x0[i]=(K[i]+0.1)/256
   ENDFOR
   FOR (i=0 up to N)
   (ncmlresult[],A)=NCML(x0)
   x0=ncmlresult
   ENDFOR
   Where x0[i] is the initial value of the i^{th} lattice. Iterate NCML for N times to get rid of the transient effect.
3. Image pixels are reshape to one row sequence I={I_0,I_1,I_2, ..., I_{WxH-1}} and reorder to new sequence I'={I_0, I_r, I_{2r}, ..., I_1, I_{r+1}, I_{2r+1},...}. The first 64 value of I' is packed as first block 8x8, second 64 value as second block, etc. The resulting is set of block B ={B_0, B_1,..., B_{num-1}}.
4. For each block in set B, perform diffusion and permutation simultaneously.

i. Get the 64 pseudorandom number as explained in Section 2, which are expressed as Φ[8][8] and the NCML result are saved as x0[].

ii. Change the value of image pixel and reorder in the same time.

```
knew= floor (x0[0]*num)
mark=0
WHILE (mark = 0)
    IF (flagknew[knew]=0 && knew != kL)
        flagknew[knew]=1
        mark=1
    ELSE
        knew=knew+1
    ENDIF
ENDWHILE
row=knew/32*8
flag=0
FOR (i=0 up to 7)
    col=(knew  mod 32)*8
    FOR (j=0 up to 7)
        IF (numblck=0 && i=0)
            newblock[i][j]=(block[flag] xor Φ[i][j] + K[j]) << 3 LSB of K[j+8-1 xor
Φ[i][j]
        ELSEIF (numblck !=0 && i=0)
            newblock[i][j]=(block[flag] xor Φ[i][j] + lastCk[j]) << 3 LSB of lastCk[j-1]
xor Φ[i][j]
        ELSE
            newblock[i][j]=(block[flag] xor Φ[i][j] + newblock[i-1][j]) << 3 LSB of
                        newblock[i-1][j-1] xor Φ[i][j]
        ENDIF
        IF (i = 7)
            lastCk[j]=newblock[7][j]
        ENDIF
        encryptimage [row] [col] = newblock[i] [j]
        flag=flag+1
        col=col+1
    ENDFOR
    row=row+1
ENDFOR
```

Where lastCk is used to save the last row of executed block and will be used to process the next block. The block that already changed its value is saved in newblock. Number of block that being processed is noted as numblck, knew represent the new location of block, flagknew as a mark is the knew already taken or not. LSB is Least Significant Bit in a value. A<<B means circular rotate of A to the left b times. The result is saved as encryptimage.

iii. Change the value of x0 from step i.

```
d ← 3 LSB of newblock[0]
FOR (i=0 up to 3)
    IF (newblock[i]>newblock[(i+d)mod8])
        Exchange the value of x0[i] with the value of x0[(i+d) mod 8]
    ENDIF
ENDFOR
```

5. Repeat step 4 till all blocks are being processed. Last result of step 4 for all blocks are the full encryptimage as cipher image. The last step is exchange the value of encryptimage [kL/32*8+7][(kL mod 32)*8+7] with the value of encryptimage[0][3 LSB of kL].

The cipher image is get from relocated encrptimage to the original image. Repeat step 4 and 5 as much as needed to get more secure result. The more rounds are processed, the higher security the encryption will have, but the expanse of computational effort and time delay.

B. Decryption Image

The goal of decryption algorithm is to recover the plain image. To process the decryption, the receiver must have the key beside the cipher image. The key which is used in decryption algorithm is the same key which is used in encryption algorithm. Steps of decryption are the reverse of steps in encryption. Flowchart of the whole decryption scheme is shown in Fig. 3. For detail of decryption is explained in pseudocode.

1. In this step, do the same steps as step 1 and step 2 in encryption algorithm. Next exchange the value of encryptimage [kL/32*8+7][(kL mod 32)*8+7] and the value of encryptimage[0][3 LSB of kL].
2. For each block in encryptimage, do the following substeps.
   i. Do the same as substep i in step 4 of encryption algorithm. Do substep ii in step 4 of encryption algorithm to get the value of knew only, then find the location of block that will be processed from encryptimage as the following.

```
flag3=0
row=knew/32*8
FOR (i=0 up to 7)
    col=(knew mod 32)*8
    FOR (j=0 up to 7)
        temp_block[i][j] = encryptimage [row] [col]
        flag3=flag3+1
        col=col+1
    ENDFOR
    row=row+1
ENDFOR
```

   Block of encryptimage that will be processed is saved to temp_block.

   ii. Inverse the diffusion step.

```
flag=0
FOR (i=0 up to 7)
    FOR (j=0 up to 7)
        IF (numblck=0 ∧ i=0)
            newblock[i][j]=((Φ[i][j] xor (temp_block[i][j] << 3 LSB of K[j-1] xor
        Φ[i][j])-K[j]) + 256 mod 256
        ELSE IF (numblck !=0 && i=0)
            newblock[i][j]=((Φ[i][j] xor (temp_block[i][j] << 3 LSB of lastCk[j-1] xor
                        Φ[i][j]) - lastCk[j]) +256 mod 256
        ELSE
            newblock[i][j]=((Φ[i][j] xor (temp_block[i][j] << 3 LSB of temp_block[i-
                        1][j-1] xor Φ[i][j])-temp_block[i-1][j]) + 256 mod 256
        ENDIF
        Ibar[flaglong]=newblock[i][j]
        flaglong=flaglong+1
    ENDFOR
ENDFOR
```

   iii. Get the value of new x0 as substep iii in step 4 of encryption process.
3. Repeat step 2 till all blocks in encryptimage are being processed. Next, reshape the blocks to one row sequence and verse step 3 in encryption algorithm.

```
flag1=0
FOR (i=0 up to ceil(W*H/r))
    flag=i
```

```
    FOR ( j=0 up to  r )
        I[flag1]=Ibar[flag]
        flag1=flag1+1
        flag=flag+ ceil(W*H/r)
    ENDFOR
ENDFOR
```

The last step is reshape the pixel to original size. If the encryption algorithm is done by x times round, then the decryption also need to be done in x times round. Result of decryption process if done correctly with the correct key will produce the recovered plain image.

IV. Implementation Result

The implementation of encryption and decryption scheme based on research [8] is built the software using Microsoft Visual Studio 2010 with the OpenCV library. Dataset uses 10 images with size 256x256 in grayscale from http://sipi.usc.edu/database/database.php?volume=misc128. On the 10 images of dataset, decryption algorithm always perfectly recovers the plain image. Fig. 4 shows the plain image as input of encryption algorithm, whereas Fig. 5 shows the cipher image as output of encryption algorithm.

From the implementation, the running time performance is found. To compare this running time, implementation of standard algorithm, AES is performed. The implementation of AES with CBC mode is built using Microsoft Visual Studio 2010 with the cryptopp library. Table 1 shows the needed time by the encryption and decryption processes of research [8] and also the running time of AES encryption and decryption as comparison.

From table 1 can be analyzed that the time needed for both encryption and decryption by research [8] are still not faster than AES as standard encryption algorithm. So can be said that AES as standard encryption with CBC mode still better to be used as encryption for image from the running time performance.

Fig. 4 Plain image

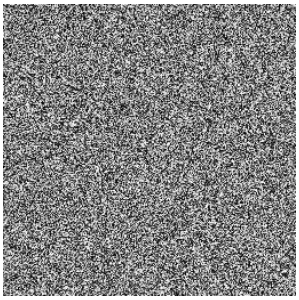

Fig. 5 Cipher image

TABLE I
RUNNING TIME PERFORMANCE

Image	Research [8] encryption time (s)	Research [8] decryption time (s)	AES encryption time (s)	AES decryption time (s)
Aerial	1. 363	1.291	0.007	0.006
Alarm	1.396	1.292	0.007	0007
Baboon	1.402	1.312	0.005	0.008
Barb	1.501	1.332	0.005	0.004
Bird	1.338	1.319	0.006	0.004
Bridge	1.387	1.238	0.007	0.006
Cameraman	1.437	1.255	0.008	0.006
Goldhill1	1.282	1.287	0.009	0.006
Lena	1.285	1.324	0.007	0.006
Peppers	1.4	1.289	0.012	0.012

V. CONCLUSION

Many researchers concern to develop algorithm to encrypt and decrypt image. One of algorithm for image encryption using permutation and diffusion with chaos-based is proposed [8]. Permutation and diffusion are combined so both can be done in only one scan image. With this way, encryption process will cost cheaper for the running time performance. In other way, AES as standard algorithm is also used by many applications. AES can work well in many kind of data, no exception for image data. In fact, the usage of AES has many modes, so data with many redundancies like image data can be encrypted well using AES algorithm. After implementing research [8], get the running time of the algorithm, and after compare it with CBC mode AES running time, the result says that CBC mode AES running time is much faster than algorithm of research [8]. From this result, CBC mode of AES is still better to be used as encryption algorithm than algorithm of combination permutation and diffusion with chaos-based by research [8] based on running time performance.

REFERENCES

[1] Zhang, Q., Guo, L., & Xiaopeng, W. (2012). A novel image fusion encryption algorithm based on DNA sequence operation and hyper-chaotic system. Science Direct on Optik, pp.3596-3600.
[2] Ye, G. (2009). Image scrambling encryption algorithm of pixel bit based on chaos map. Pattern Recognition Letters, pp.347–354.
[3] Wang, B., Wei, X., & Zhang, Q. (2012). Cryptanalysis of an image cryptosystem based on logistic map. Optik, pp.1773–1776.
[4] Chen, G.,Mao, Y., & Chui, C.K. (2004) A symmetric image encryption scheme based on 3D chaotic cat maps. Chaos Solitons & Fractals, pp.749-761.
[5] Guan, Z., Huang, F., & Guan, W. (2005) Chaos-based image encryption algorithm. Physics Letters, pp. 153-157.
[6] Lian, S., Sun. J., & Wang. Z. (2005) A block cipher based on a suitable use of the chaotic standard map. Chaos Solitons & Fractals, pp. 117-129.
[7] Wong, K.W., Kwok, B. S., & Law, W. S. (2008) A fast image encryption scheme based on chaotic standard map. Physics Letter, pp. 2645-2652.
[8] Wang, Y., Wong, K.-W., Liao, X., & Chen, G. (2011). A new chaos-based fast image encryption algorithm. Applied Soft Computing, pp. 514-522.