Military Technical College
Kobry El-kobbah,
Cairo, Egypt

5th International Conference
on Electrical Engineering
ICEENG 2006

Improving the Performance of AES Encryption Algorithm

Alaa El Din Fahmy **Ahmed Sobhy**

ABSTRACT

A software simulation has been implemented for Advanced Encryption Algorithm AES. Moreover, a chaotic generator has been added to AES to improve its security performance via generating a multiple keys instead of using one key. The modified AES has been tested for image encryption. The encrypted images have random shape and it is absolutely un-identified and has no clue for the original image.

1. Introduction

Cryptography provides solutions to two major problems of data security, the privacy problem, preventing an opponent from extracting information from a communication channel, and the authentication problem, preventing an opponent from injecting false data into the communication channel, or altering messages. On January 1997, NIST began the process of choosing a replacement for DES. The replacement would be called the Advanced Encryption Standard (AES). A formal call for algorithms was made on September 12, 1997. It was required that the AES have a block length of 128 bits, and support key lengths of 128, 192, and 256 bits. Twenty-one cryptosystems have been submitted.

This paper focuses on increasing the performance of the AES algorithm by adding the chaotic generator to change the key before using it in the algorithm. AES candidates were evaluated for their suitability according to three main factors:

- Security.
- Cost.
- Algorithm and implementation characteristics.

In the last thirteen years, there has been a great deal of interest in the study of nonlinear dynamical systems [7, 8]. The introduction of chaos into communication systems offers several opportunities to improve the security performance. This is because of the random nature of chaotic systems. Since a chaotic dynamical system is deterministic system, it is random like behavior can be very helpful in disguising modulations as noise. Moreover, through the sensitivity dependence of chaotic systems on their initial conditions, a large number of uncorrelated, random like yet deterministic and reproducible signals can be generated. These signals are only reproducible on finite arithmetic machines. The quantization doesn’t destroy the desirable properties of the sequences, and there would still be a large pool of chaotic sequences from which to choose [9]. Mixing the AES encryption algorithm and the chaotic generator is the main concern of this paper and a comparison between the results of the standard AES and chaotic added AES in image encryption with the same seed Key (128 bit).

This paper is organized as follows: section 2; deals with AES algorithm, section 3, introduces the chaotic generator, section 4, presents the AES simulation with results, and section 5 concludes the paper.

2. AES Encryption Algorithm

The Rijndael algorithm [10] is selected by National Institute of Standards and Technology (NIST) as a new Advanced Encryption Standards (AES). The Rijndael algorithm is based on arithmetic in a finite Galois field, GF(2^8). It has 10, 12, or 14 rounds (Nr), depending on the key size. In this paper, we consider only operations using a 128-bit cipher key and 128-bit data blocks, although the algorithm scales to accommodate different key and data block sizes. In this case, the algorithm requires 10 rounds; each round operates on the state, a 4 x 4 array of bytes called the state (table 1). The dimensions of the state depend on the block size.

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$

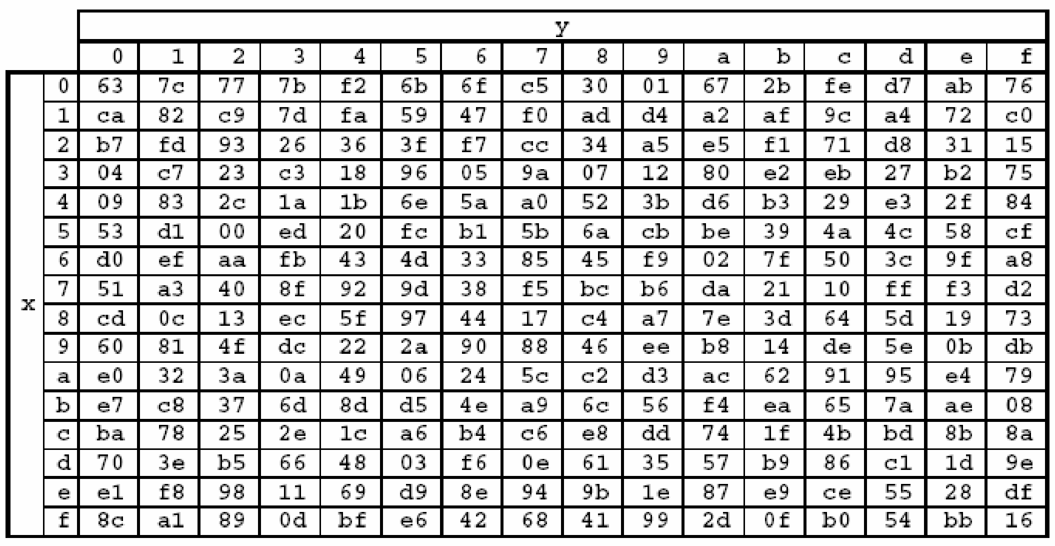
Table 1 4x4 Array of Bytes (State)

In addition, each round involves up to four basic transformations: -
(a) Substitution byte operation;
(b) Shift rows operation;
(c) Mix columns operation;
(d) Add round key operation.

2.1 Substitution Byte (SubByte) Operation

The operation of SubBytes performs a substitution on each byte of state independently, using affine transformation, namely, S-box (table 2), which is a permutation of $\{0, 1\}^8$ .The bytes are represented in hexadecimal notation as l6x16 array, where the rows and columns are indexed by hexadecimal digits. In contrast to the S-boxes in DES, which are apparently random substitutions, the AES S-box can be defined algebraically. The algebraic formulation of the AES S-box includes operations in a finite field, defined by:
$\phi : (a_7\, a_6 . . . a_0) \rightarrow \sum a_i x^i \quad a_i \,\varepsilon\, GF(2)$, then:
SUBBYTES (a) = $Ø[(x^4+x^3+x^2+x+1)Ø(a)'+(x^6+x^5+x+1)\mathrm{mod}(x^8+1)]$.,
This operation can be performed using the following steps:
$z = Ø(a)$ field represented of the byte a.
$z = z^{-1}$ the inverse in GF(2^8).
$b = Ø^{-1}(z)$ map the field element z to the byte b.

Output the byte b using the following affine transformation



		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	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
x	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

Table 2 AES S-Box

Where:

$b_i = b_i \oplus b_{i+4 \bmod 8} \oplus b_{i+5 \bmod 8} \oplus b_{i+6 \bmod 8} \oplus b_{i+7 \bmod 8} \oplus c_i$ and c = (11000110)

The inverse of the SUBBYTES can be defined by:

INVSUBBYTES(a) = $Ø^{-1}$ $[((x^6+x^3+x)\,Ø(a)+(x^2+1) \bmod (x^8+1))]$

2.2 ShiftRows Operation

The operation of shift row acts on state. Each row of the *state* is cyclically shifted by *i* bytes to the left, where *i* is the row number (0, 1, 2, or 3) (table 3). The inverse operation INVShiftRows applies right shifts instead of left shifts.

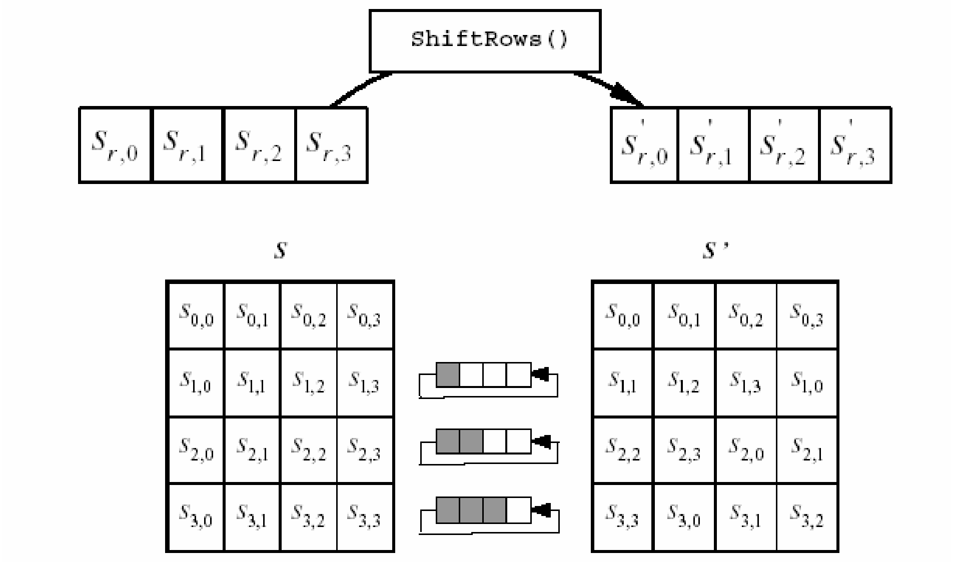
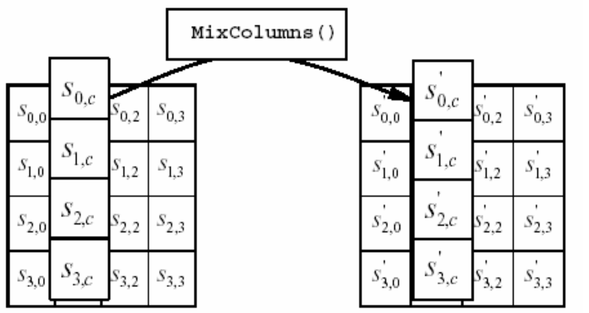

ShiftRows()
$S_{r,0}$ $S_{r,1}$ $S_{r,2}$ $S_{r,3}$
$S'_{r,0}$ $S'_{r,1}$ $S'_{r,2}$ $S'_{r,3}$
S
S'
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$

Table 3 Shift Row Operation

2.3 MixColoums Operation

The operation MixColumns is carried out on each of the four columns of state. Each column of the state is replaced by a new column which is formed by multiplying that column by a fixed polynomial, c(x) ='03'x3 $\oplus$'01'x2 $\oplus$'01'x$\oplus$'02', modulo x^4+1.The multiplication is over $GF(2^8)$. That is the MixColumns can be described as a linear transformation applied to each column of state, multiplying each 4-element column vector by a 4x4 matrix with the coefficients in F_2 (Fig.1). The MixColoums Transformation Equation for encryption is shown below.


MixColumns()
$s_{0,0}$ $s_{0,c}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,c}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,c}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,c}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,c}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,c}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,c}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,c}$ $s'_{3,2}$ $s'_{3,3}$

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Fig.1 MixColumns Operation

2.4 AddRoundKey and the Key Schedule

In AddRoundKey, each column of state is bitwise XORed with one word of the round key. The round key is an extension of the cipher key unique to each round. For 128- bit Rijndael, there are 10 rounds plus one preliminary application of AddRounKey. Therefore, the key schedule must produce 11 round keys, each consisting of four 4-byte words, from the 128-bit key. Key expansion produces an expanded key consisting of the required 44 words. The key $K=(k_0,k_1,k_2,k_3)$, where the k_1 are 4-byte words, and the expanded key is denoted by the word vector $(w_0,w_1,w_2 \ldots ,w_{43})$. The key schedule for the 10-round version of AES, which uses a 128-bit key are similar to key schedules for 12, and 14 round versions of AES, but there are minor differences in the key-scheduling algorithm.

The key schedule has a much slower diffusion structure than the cipher and contains relatively few non-linear elements. It was designed with the requirement that knowledge of a part of the cipher key or round key bits shall not allow to compute many other round key bits. There is an interesting property is that the key can be split into two halves. The two topmost rows interact with the two bottommost rows through only 14-byte (in case of 128-bit block size and 256 bit key). The key can be splitted by rows or by column (at least for a few cycles).

The Rijndael algorithm proceeds as follows:

1. Initialize state with the plaintext M: Where M consists of the 16 bytes M_0, M1...... M_{15} .
2. Perform ADDRoundKey, which XOR's the first RoundKey with state.
3. For each of the first N_{r-1} rounds:
    - Perform SUBBYTES on state;
    - Perform SHIFTROWS on state;
    - Perform MIXCOLUMNS on state;
    - Perform ADDROUNDKEY.

4. For the last round:
    - Perform SUBBYTES on state;
    - Perform SJEIIFTROWS on state;
    - Perform ADDROUNDKEY.
5. Define the ciphertext to be state.

2.5 Decryption of Rijndael algorithm

To decrypt, perform the cipher in reverse order, and the reverse key schedule as follows:

1. ADDROUNDKEY with round key Nr.
2. For the round N_r-1 to 1:
    - InvShiftRows.
    - InvSUBBYTEs.
    - ADDROUNDKEY.
    - InvMixColumns.
3. For round l:
    - InvShiftRows.
    - InvSUBBYTES.
    - ADDROUNDKEY using rounds key 1.

2.6 Strengths and weakness of Rijndael algorithm

2.6.1 Strength of Rijndael

The algorithm is secure against all known attacks, and the non-linearity resides in SubBYTES. SHIFTROWS and MIXCOLUS ensure that after a few rounds, all output bits depend on all input bits. In addition, knowledge of part of the cipher key or round key does not enable calculation of many others round key bits. Each key bit affects many round key bits (Key avalanche effect). The algorithm need very low memory requirements, and very fast both in hardware and software. The use of the finite field inversion operation in the construction of the S-box yields linear approximation and difference distribution tables in which the entries are close to uniform. This provides security against differential and linear attacks. Rijndael has a low ROM requirement and very low RAM requirement. Both encryption and decryption are at least twice as fast as any other finalist. There are apparently no known attacks on AES that are faster than exhaustive search.

2.6.2 Weakness

The decryption is slower than encryption. In addition, the decryption algorithm is different from encryption. A drawback is that ROM requirements will increase if both encryption and decryption are implemented simultaneously, although it appears to remain suitable for these environments. Another drawback is observed if a block of data repeated and encrypted with the same key the result will be the same.

3. Chaotic Generators:

In this paper, we concerned with systems when they are operating in the chaotic state to generate random outputs which will be used later in generating multiple random keys for the AES algorithm. A discrete time dynamical system is:

$$X_{k+1} = f (X_k) , 0 < X_k < 1 , k= 0,1,2, 3\ldots..$$

Where $X_k \in R_n$ is called the state and f maps the state Xk to the next state X_k+1 starting with initial condition X_0. Chaotic maps don't have to be very complicated, widely studied dynamical systems like:

1- Logistic map
2- Cubic map
3- Tent map

The experiments were carded out with discrete-time dynamic systems:

1- Logistic map as a discrete chaos generator [5]:

$$X_{k+1} = r * X_k * (1-X_k) \quad \text{Where } 3.2 < r < 4$$

2- Cubic map [6]:

$$X_{k+1} = r * X_k * (X_k^2 -1) + X_k . \quad \text{Where } 3.2 < r < 4$$

Where $3.2 \le r \le 4$, and r is called the bifurcation parameter. Depending on the value of r, the dynamics of this system can change dramatically, exhibiting periodicity or chaos. For $3.75\ldots \le r \le 4$ the sequences are, for all practical purposes, non-periodic and non-converging. The initial condition was set to be ($X_0 = 0.1$), Then we but a threshold to generate the sequence of ones and zeros.

3- Tent map [3]:

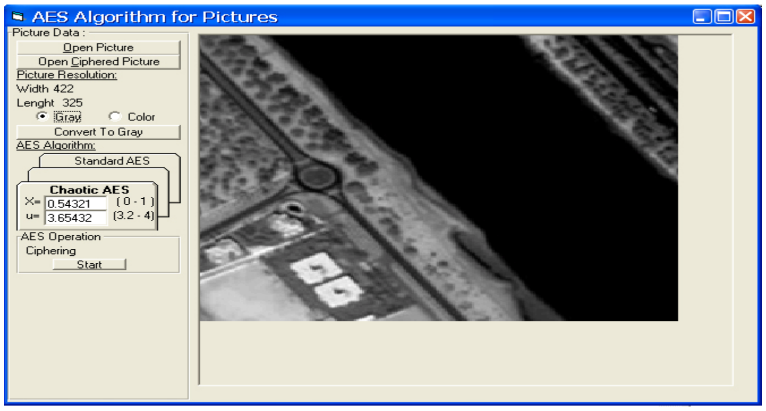
$$X_{k+1} = \begin{bmatrix} 0.5 - 2 * X_k , X_k \ge 0 \\ 0.5 + 1.8 * X_k , X_k \le 0 \end{bmatrix}$$

The initial condition was set to be (X0 = 0.1).Then we but a threshold to generate the sequence of ones and zeros. Chaotic system has a very sensitive dependence on their initial conditions. This sensitivity dependence can be demonstrated by giving two very close initial points to the iterative map. After little iteration, the two resulting sequences will look completely uncorrelated hence, any slight deference in the component of the circuit (analog one) will cause transmitter receiver mismatching [4].

4. AES standard encryption algorithm software

The software is designed to simulate the AES encryption and decryption algorithms on images (displaying and storing) as the probability of redundancy (more than one pixel with the same value) is higher than ordinary files and the performance can be clearly measured visually. The drawbacks of AES encryption algorithm in encrypting a large number of blocks (plain data) with the same value lead to the same results because of the constantly of the data and the seed key so we use a Chaotic algorithm to generate new seed key for each block of data.

To present the difference between the AES algorithm on image encryption and the chaotic added AES we implement software as a simulator to the AES in encryption and decryption operations and the following results is generated through it. The interface of this software is shown in Fig.2, the software can be used in both encryption and decryption processes. It also can be used in the simulation of the standard AES and the AES with the chaotic generator. The main problem appears when encrypting an image with many redundant pixels for example if the image shown in table 4 is encrypted with the AES algorithm then due to the constant seed key then the encrypted image will be the one in table 4 and it is clear that the same block of data will lead to the same encrypted block so the cryptanalyst can estimate that this area has the same stream.



Picture Data :

Picture Resolution:

Width 422

Lenght 325

[✓ Gray] Color

AES Algorithm:

Chaotic AES

X= 0.54321 (0 - 1)

u= 3.65432 (3.2 - 4)

AES Operation

Ciphering

Fig.2 Software Interface

The block diagrams in Fig.3 shows the architecture of the AES simulation software.

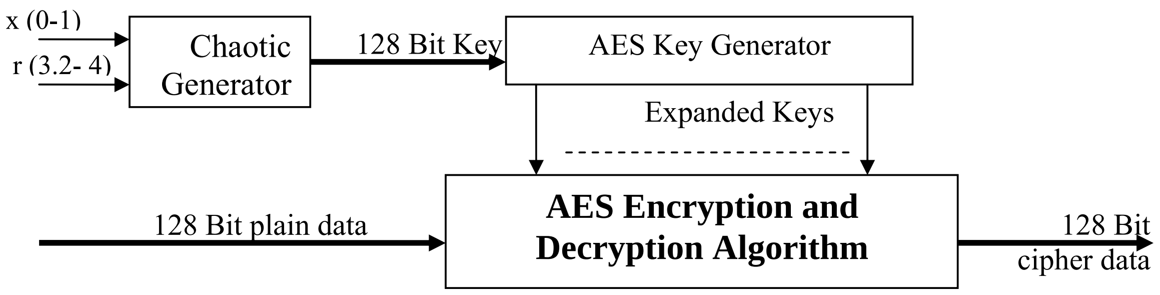


Fig.3 Architecture of Chaotic Added AES Software

In the Chaotic Generator Block the proposed chaotic system is Logistic map

$$X_{k+1} = r * X_k * (1-X_k) \quad \text{Where } 3.2 < r < 4$$

4.1 Results

In this section, we will introduce the results of the AES algorithm for encryption of an image, the basic problem was when encrypting an image with many pixels with the same color by the same seed key the encrypted image contains an area with partial known part as shown in the images tables (4, 5). It is clear the difference in the performance in encrypting the image with AES algorithm and the chaotic added AES from tables (4, 5).

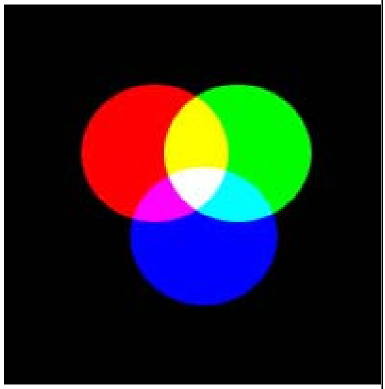
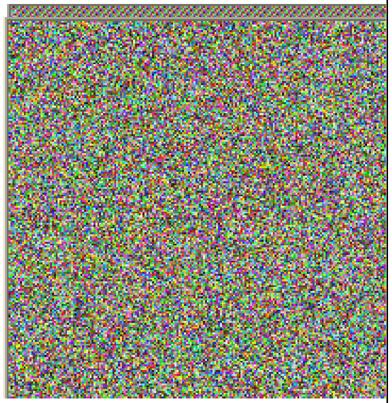
Original image	**Encrypted image With AES algorithm**
	
	Encrypted with chaotic generator added to AES

Table 4 AES Results on colored image

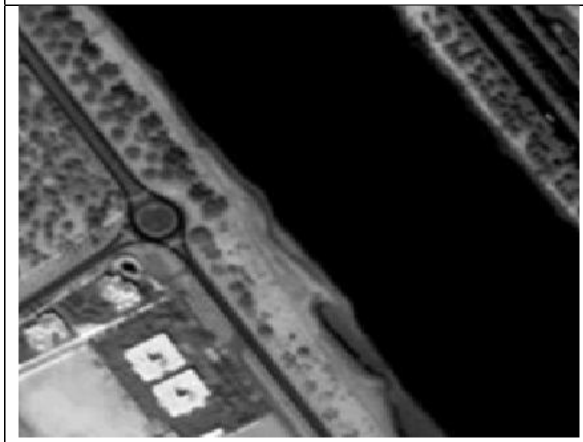
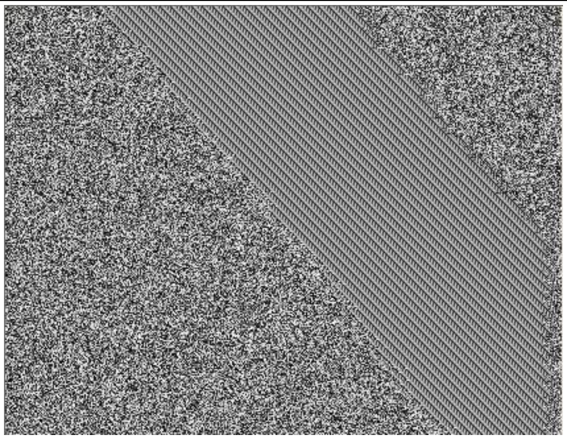
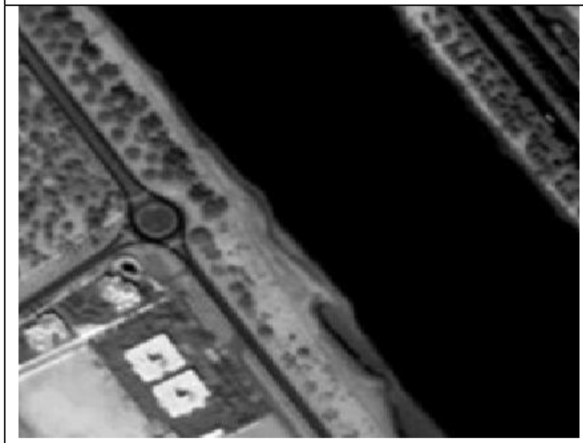
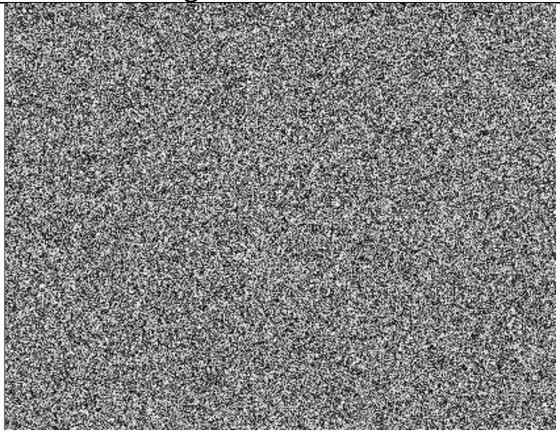
Original image	**Encrypted image** **With AES algorithm**
	
Original image	**Encrypted image** **With chaotic generator added to the AES**
	

Table 5 AES Results on a Grey Map Image

5. Conclusion

This paper deals with improving the security performance of AES algorithm especially for image encryption. This goal can be achieved by adding a chaotic generator to generate multiple 128 bit key. The software simulation has been tested for image encryption and it is clear that the pixels of encrypted images have been distributed in a random manner, despite of the use of AES only.

REFERENCES

[1] R.M. May, "Special mathematical models with very complicated dynamics," Nature, June 1976.
[2]Proc. IEEE, Special Issue on chaotic systems, 1987.

[3] S.O. Strakov, S.V.Yemetez," Digital communication systems using chaos" IEEE pp.207-210, 1997.
[4] M P Kennedy, Riccardo Rovatti and Gianluca Setti,"Chaotic Electronics in Telecommunication", CRC Press, 2000.
[5] H.G.Schuster, Deterministic chaos ,An introduction ,D-6940Weinheim Federal republic of Germany , PhysicK – Verlog, GmnH, 1984
[6] K. Umeno and K. Kitayama "Spreading sequences using periodic orbits of chaos for CDMA", Electronics Letters, Vol.35, No.7, pp.545-546, 1999.
[7] http://csrc.nist.gov/encryption/aes/
[8] C. Burwick, D. Coppersmith, E. D' Avignon, et al, "MARS-a candidate cipher for AES," 1St AES Conference, Ventura, CA, August 20-22, 1998.
[9] R. L. Rivest, M. J. B. Robshaw, R. Sidney and Y. L. Lin, "The RC6 Chiper." l AES Conference, Ventura, CA, August 20-22, 1998.
[10] "The Rijndael Block Cipher: AES Proposal", Joan Daemen, Vincent Rijmen. First AES Candidate Conference (AES 1), August 1998.