International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 2, April 2023, pp. 1708~1723
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i2.pp1708-1723

Advanced approach for encryption using advanced encryption standard with chaotic map

Yahia Alemami[1], Mohamad Afendee Mohamed[1], Saleh Atiewi[2]
[1]Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Besut, Malaysia
[2]Department of Computer Science, College of Information Technology, Al Hussein Bin Talal University, Maan, Jordan

Article Info

Article history:

Received Feb 16, 2022
Revised Sep 15, 2022
Accepted Oct 13, 2022

Keywords:

Advanced encryption standard
Avalanche effect
Sine map
Throughput

ABSTRACT

At present, security is significant for individuals and organizations. All information need security to prevent theft, leakage, alteration. Security must be guaranteed by applying some or combining cryptography algorithms to the information. Encipherment is the method that changes plaintext to a secure form called cipherment. Encipherment includes diverse types, such as symmetric and asymmetric encipherment. This study proposes an improved version of the advanced encryption standard (AES) algorithm called optimized advanced encryption standard (OAES). The OAES algorithm utilizes sine map and random number to generate a new key to enhance the complexity of the generated key. Thereafter, multiplication operation was performed on the original text, thereby creating a random matrix (4×4) before the five stages of the coding cycles. A random substitution-box (S-Box) was utilized instead of a fixed S-Box. Finally, we utilized the eXclusive OR (XOR) operation with digit 255, also with the key that was generated last. This research compared the features of the AES and OAES algorithms, particularly the extent of complexity, key size, and number of rounds. The OAES algorithm can enhance complexity of encryption and decryption by using random values, random S-Box, and chaotic maps, thereby resulting in difficulty guessing the original text.

Corresponding Author:

Mohamad Afendee Mohamed
Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin
Besut Campus, 22200, Terengganu, Malaysia
Email: mafendee@unisza.edu.my

1. INTRODUCTION

Information is confronted with various challenges primarily owing to the Internet, which has transformed the world into a small village, where numerous illegal methods are used to obtain information. Accordingly, the subject of security has become extremely critical for information protection, in which many security algorithms are used to achieve safety goals, and that the various aims of secure systems include authentication, confidentiality, and data integration [1]. In this current research a new method has been developed to improve advanced encryption standard (AES) algorithm. The reminder of this paper is organized as follows: section 2 explains the used chaotic map in the proposed algorithm, section 3 presents an overview of AES, section 4 provides a detailed steps for the proposed algorithm, sections 5 and 6 represent the analysis and conclusion respectively.

Encryption is required to safeguard sensitive information from unauthorized access. Moreover, chaos has been introduced to cryptography, which is similar to confusion and diffusion in cryptography, because of its ergodicity, pseudo-randomness, and sensitivity to initial conditions and control parameters. Accordingly, chaotic systems could be used to build cryptosystems owing to these characteristics [2].

Cryptographic algorithms that are considered security algorithms involve two technicalities: symmetric and asymmetric cryptography [1]. Encipherment is a process that converts original text to unclear text, whereas decryption is the reverse of this process [3]. Figures 1 and 2 show the symmetric and asymmetric cryptography, respectively, [3].

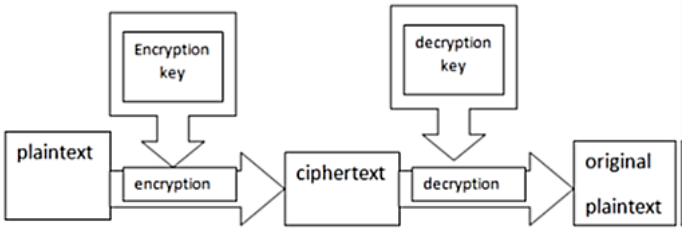


Figure 1. Symmetric cryptosystem

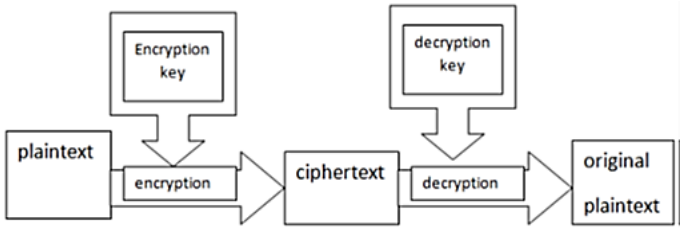


Figure 2. Asymmetric cryptosystem

The following parameters are utilized to determine the strength of encryption algorithms.

- Avalanche effect: avalanche effect is significant to cryptography. To date, numerous algorithms have been presented for the substitution-box (S-Box) generation to improve the avalanche effect [4]. The avalanche effect is calculated using (1) [3], [5]:

$$Avalanche\ Effect\ = \frac{Number\ of\ flipped\ bits\ inside\ the\ cipher\ text}{Number\ of\ bits\ in\ this\ cipher\ text} * 100\% \quad (1)$$

  An adjustment in one bit of the plaintext or one bit of the key should create a change in numerous bits of the cipher texts. This property is known as the avalanche effect [5].
- Encryption time: for encryption time, the unit of measure is millisecond (ms) and depends on the length of text and key size. The higher the encryption speed, the better the algorithm will perform.
- Decryption time: decryption time is the time span to recapture the first text from the encrypted text and is likewise estimated in ms. The higher the encryption speed, the better the algorithm will perform.
- Throughput: is the total size of the encrypted text file in bytes divided by the total time of encipherment. The performance is better, when the higher is the throughput [3].
- Central processing unit (CPU) utilization: refers to the amount of time the CPU spends processing data. It is the amount of time that the CPU is dedicated to the calculation of the process. It shows how much the CPU is being used. The load on the CPU increases as the encryption time increases [6].

The study [7] assessed the presentation of two encipherment techniques: AES and data encryption standard (DES). The outcomes in Table 1 are based on processing time, CPU usage, and transfer rate of the cipher in two operating systems (i.e., Windows and Mac platform) for various text sizes. Meanwhile, Tables 2 and 3 show the time comparison based on entropy for two different tests.

Table 1. Encryption time, throughput, and CPU utilization for diverse data size in Windows/Mac OS

Algorithm types	Average encryption time in sec for diverse data size (MAC OS)	Average encryption time in sec for diverse data size (Windows OS)	Throughput for windows	Throughput for MAC	CPU utilization in windows	CPU utilization in MAC
AES	7.11	8.105	27.76	31.65	4.2%	2%
DES	21.83	22.195	10.13	10.31	2%	1.9%

The research [8] compared AES and DES and determined that AES is safer than DES. Tables 2 and 3 show the execution times of AES and DES, respectively, by using entropy. The result showed that both algorithms consumed diverse times in diverse computers. Diverse computers take diverse times for the same algorithm over the same data packet, which shows that all algorithms have diverse speeds. The research [9] compared Rivest-Shamir-Adleman (RSA) and AES in terms of coding time and memory usage. The comparison was made on different types of data with such extensions as *.jpeg, .txt, .doc*, and *.pdf*. Table 4 presents the outcomes for text file.

Table 2. Test 1-comparison depending on entropy

File sizes	AES times (s)	DES times (s)
512 Kb	49.03	66.09
1 Mb	44.84	67.94
2.0 Mb	51.14	66.29
2.5 Mb	51.28	59.42

Table 3. Test 2-comparison depending on entropy

File sizes	AES times (s)	DES times (s)
512 Kb	36.81	35.28
1 Mb	32.81	45.45
2.0 Mb	31.15	37.7
2.5 Mb	45.53	37.89

Table 4. Comparison analysis between RSA and AES for text file

Data sizes (bytes)	Algorithms	Encryption times (ms)
408	RSA	6
408	AES	26
42317	RSA	11
42317	AES	62
42317	AES	44

The research [10] utilized a new methodology on AES to produce dynamic encipherment by giving a character (salt) to the original text. Salt was shaped from the after effect of a linear congruential generator (LCG) calculation. At this point, a blend of salt and plain content will merge with the algorithm of the onetime pad (OTP). Testing the dynamic original text demonstrated an effective test, thereby demonstrating the integrity of the consequences of the encipherment process.

In [11] an improvements were made to the AES algorithm. First, a skew tent map was utilized to create the key sequences. Second, the eXclusive OR (XOR) operation was performed between matrix D and the sequence key. The plaintext matrix was encoded from left to right, while the ciphertext matrix was decoded from top to bottom and from right to left. After the XOR process, the key and plaintext were joined. Third, the mix columns array tasks were modified, and addition and subtraction were adopted to strengthen the relationship between pixels. Tests showed that the improved algorithm is suitable for encipherment in a cloud computing environment. A hybrid system was proposed, in which AES-ECC is used, which combines the advantages of the AES algorithm in accelerating data encipherment with ECC to secure symmetric key session exchange. Accordingly, strength in information security was proven [12].

Some random sequences were used in the process of modifying the AES algorithm. In particular, the pseudorandom number was utilized to avoid the derivation of the sequences, thereby preventing hackers from knowing the secret key. The outcomes indicated that the proposed algorithm is superior to the standard algorithm in terms of low and secure key generation [13]. The study [14] amended S-Box in the AES algorithm by utilizing the Fibonacci series and prime factor. The results showed that the proposed S-Box has good encipherment properties compared with the original S-Box. The results are shown in Table 5.

Table 5. Comparison between the original and improved S-Box

Parameters	[14]	Standard AES S-Box
Size	8 MB	8 MB
Absolute indicator	32	32
Sum of the square indicator	133120	133120
Algebraic immunity	4	4
Algebraic degree	7	8
Transparency order	7.860	7.859
Signal to noise ratio (SNR)	9.600	9.887

The number of rounds was increased to 16 rounds for the AES algorithm to increase security. Experiments showed that it provides high speed, and the encryption key was created by utilizing the Polybius square [15]. The study [16] proposes a secured modified advanced encryption standard algorithm that reduces the number of rounds in the AES to 14 in order to reduce encryption and decryption process time

while boosting data security. The findings show that the proposed technique has a higher efficiency in terms of encryption and decryption process time, while increasing security as evidenced by the avalanche effect test. Table 6 shows execution time for various files during encryption and decryption.

Table 6. Execution time of encryption for various files

Input file size (KB)	Execution time for encryption in seconds		Execution time for decryption in seconds	
	AES	[16]	AES	[16]
2	24	15	24	15
3	72	40	72	40
4	143	125	143	125
5	291	240	291	240
6	481	425	471	425
7	692	598	692	598

According to [16] the average amount of avalanche effect of all rounds utilizing the suggested technique is more than 50%, proving that the proposed system is secure, according to the results of the avalanche effect test, where the highest result reached to 58.59%. The research [17] suggested a new technique to improve AES with shift row and S-Box alteration for the mix column transformation, in which Sub byte (S-Box) is moved to join the mix column. The outcomes showed that the enhancement rates of encipherment and decipherment reached 86.143% and 13.085%, respectively.

The research [18] modified the S-Box through two S-Boxes, the first of which is the same as the Rijndael S-Box and the second was built through XOR and affine transformation, and will replace the MixColumns operation within the internal rounds in the coding process. The results of the AES algorithm after modification were 108.369% more efficient than the original AES. Various polynomials were used to generate novel S-Boxes. In the alteration, x8+x6+x5+x+1 was utilized instead of x8+x4+x3+x+1, which was used in the original AES. Results indicated that the modified AES is superior to the standard AES in terms hamming distance, balanced output, and avalanche effect [19].

The study [20] suggests a change to the AES algorithm. The bit permutation transformation was utilized instead of the mix columns transformation since the utilization of bit change in an encryption calculation accomplishes productivity by giving least encryption time and reduce memory utilization. The improved AES algorithm performed 18.47% quicker encryption and 18.77% faster decryption for text files, according to the study's findings. In comparison to the standard AES method, the modified AES algorithm produced a 16.53 percent larger avalanche effect compared to standard AES, thus lead to improving security.

The encryption method that combines the AES and RSA algorithms in this research takes full advantage of both symmetric key and asymmetric key. The file's session key is encrypted with RSA, while the data file's encryption is done with AES. The encryption processing efficiency of the system is great. 128-bit encryption key was created, which is then encrypted using the RSA asymmetric technique [21]. The results were as follows, as shown in Table 7.

Table 7. Simulation test

Algorithm	Data size (MB)	Encryption time (s)	Decryption time (s)	Memory used (KB)
AES	1	0.153	0.305	14.7
	10	3.813	15.253	
	20	19.067	305.064	
[21]	1	0.183	0.405	17.756
	10	4.213	16.153	
	20	19.867	315.064	

A symmetric encryption technique was proposed in [22]. The modification is based on AES and adds a second or additional key. Another change is made at the SubBytes stage, where the transportation operation is added to the original SubBytes operation. Table 8 shows the results of the tests. The study [23] utilized a one-dimensional S-Box, In any event, expelling a complete byte necessitates replacing the S-Boxes twice. Four bits of state byte were replaced, while another four bits were superseded by the original S-Box. A reducible polynomial (i.e., x8+x4+x3+1) was used. Furthermore, a particular byte was selected for the static fragment outline 0×63. The results showed that productivity reached 18.35%.

The work in [24] proposes a new approach for generating dynamic S-Box from cipher key to improve the encryption process of the advanced encryption standard algorithm. This approach will construct more secure block ciphers, solve the problem of fixed structure S-Boxes, and improve the AES block cipher

system's security level. The main advantage of this technique is that it can generate many S-Boxes simply by altering the cipher key. Table 9 shows the avalanche effect results of both original AES and proposed AES.

The inclusion of the Dynamicity properties in the AES S-Box for greater resilience against brute force attack and biclique attack is the goal of this study in reference [25]. Hashing techniques are paired with the AES algorithm to produce variance in security using MD4, SHA3, or SHA5. To improve the avalanche effect of the AES algorithm, a novel key dispersion mechanism is presented. Table 10 shows the avalanche effect.

In the study [26], AES was fortified by consolidating symmetric and asymmetric types. Diffie-Hellman was utilized for the AES key creation, while private number creation was used to generate dynamic S-Boxes. The proposed approach is ideal for rapid calculation and can resist cryptographic assaults, such as linear and differential cryptanalysis assaults.

A chaotic dynamic S-Boxes was constructed utilizing the Chebyshev polynomial theory of the first type. This approach was used to create 80 random 8×8 S-Boxes, and has acceptable safety and low computational cost compared with 21 modern approaches [27]. In the study [28], it is shown that changing the polynomial does not result in a major change in the avalanche.

Table 8. Execution time in milliseconds for different files (encryption/decryption)

File size	Encryption time in ms		Decryption time in ms	
	AES	[22]	AES	[22]
1 KB	1.371	0.970	1.917	1.744
2 KB	2.321	1.811	3.281	3.164
5 KB	5.418	4.151	9.024	8.178
10 KB	9.377	7.526	15.915	15.421
50 KB	44.568	32.273	77.595	77.240
100 KB	103.011	65.927	186.433	126.285
1 MB	867.852	717.192	1540.999	1540.755

Table 9. The avalanche test findings

Algorithm	Plaintext	Avalanche effect
AES	11111111111111111111111111111111	50.78%
[24]	11111111111111111111111111111111	54.68%

Table 10. Avalanche effect comparison

Algorithm	Avalanche effect
Original AES	46
[25]	67.18

2. CHAOTIC MAP

An encipherment system based on chaotic maps is a highly effective system in the coding process. The reason is that chaotic theories are characterized by their sensitivity to input factors and initial conditions. This section presents the types of chaotic maps used in the proposed system. Sine map: Sine map is considered one of the discrete chaotic techniques and expressed in (2) [29].

$$Y(m+1) = a.sin(pi.Y(m)) \quad (2)$$

where a is the control parameter, a ϵ [0-1], pi=3.14, and m index of encryption key. As shown in Figure 3 the sine map diagram represent.

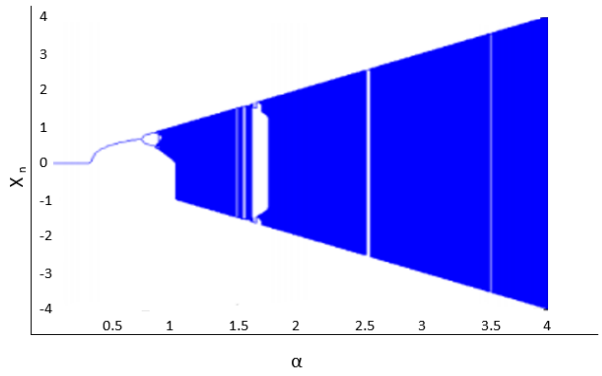


Figure 3. Sine map where α=0.9 [7]

3. ADVANCED ENCRYPTION STANDARD

AES is a block cipher with block size of 128 bits. The key can be 128, 192, or 256 bits. The number of inner rounds of the cipher is shown in Table 1. The algorithm is called AES-128, AES-192 or AES-256, depending on the key size [3], [7]. Figure 4 shows the steps for every round. Table 11 shows the key length and number of rounds of the AES algorithm [30].

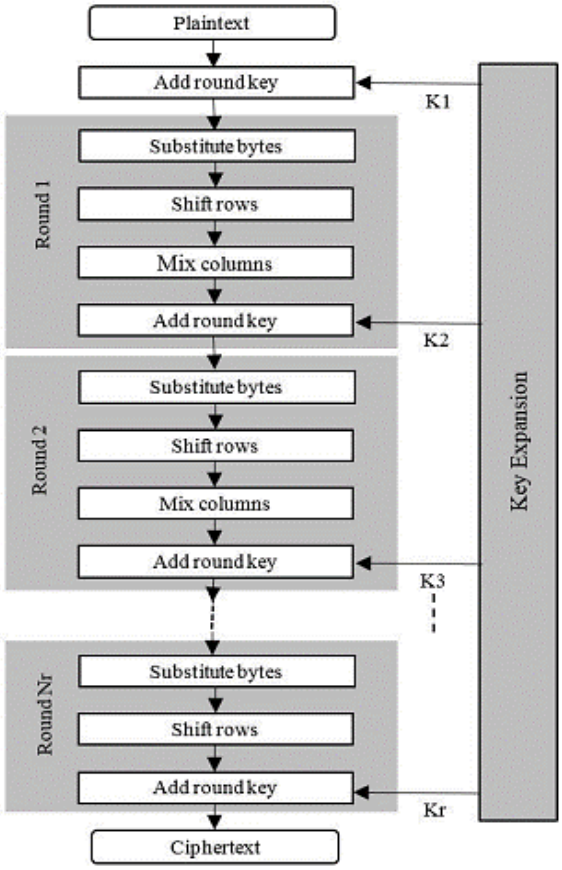


Figure 4. AES structure [3], [31]

Table 11. Key lengths and number of rounds of the AES algorithm

Key lengths	Number of rounds
128 bits	10
192 bits	12
256 bits	14

4. PROPOSED ALGORITHM

Figure 5 illustrates the structure of the optimized advanced encryption standard (OAES) encryption, in which new steps were added to the original AES, including the derivation of a new key from the original key (128) bits by using sine map, thereby leading to camouflage and complexity in obtaining the original text due to changing the entered key, then using a random matrix (4×4) to perform multiplication with the entered text, also random S-Box generation, and the last step is performing an XOR operation with the last key that was generated (the new key).

OAES contains the following new steps.

1) User enters the key (128) bit to encrypt the text.
2) A change is made to the key so that the range becomes [0 to 8], with a size of (128 bit).
3) Using the sine map algorithm, a new key is derived from the original key. This step leads to increased complexity for hackers in finding the new key, in which the sine map is considered one of the chaotic map algorithms.

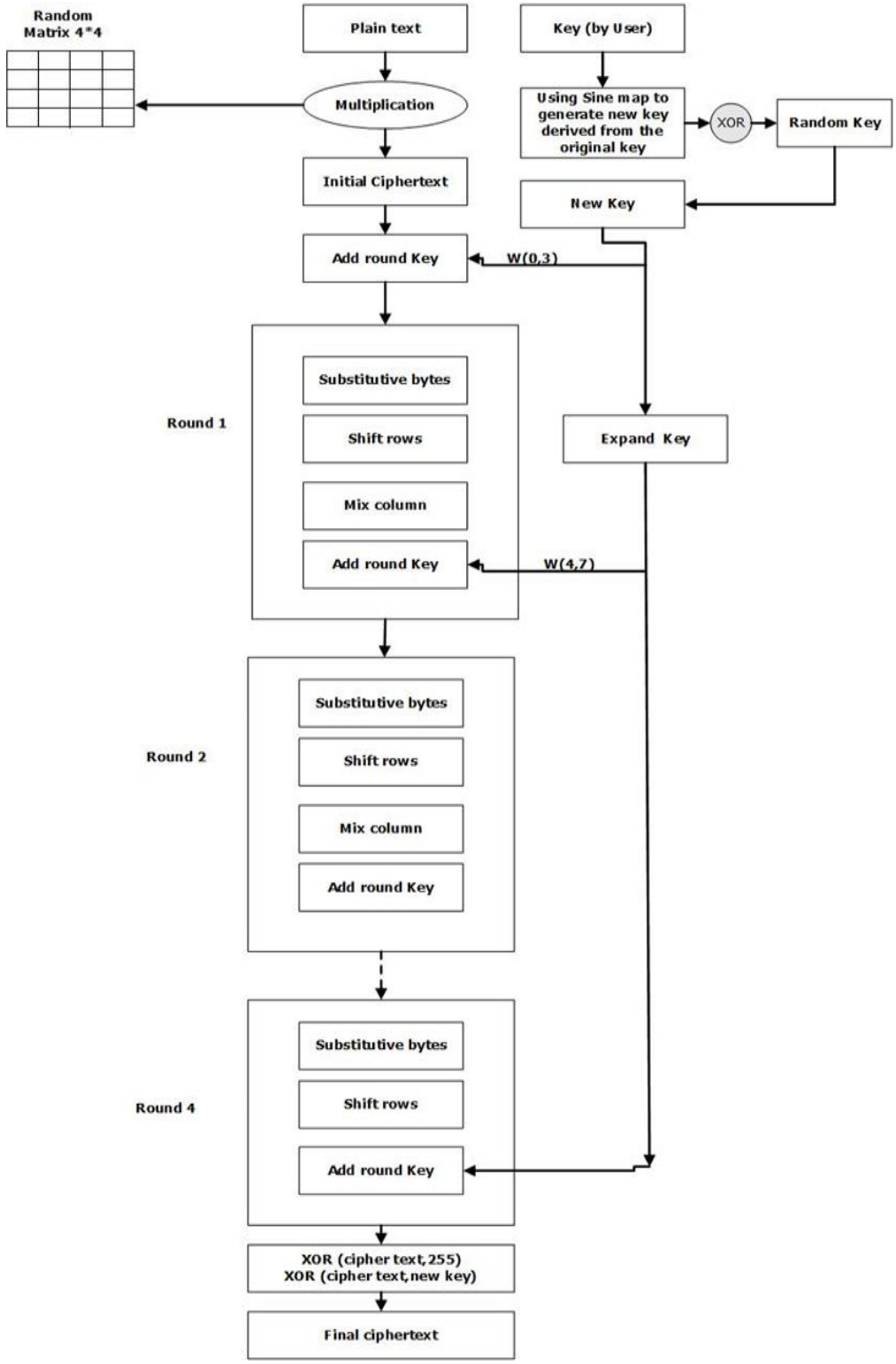


Figure 5. OAES algorithm

4) A random number (128 bit) is generated. XOR is implemented between the key generated by the sine map and the random number. The first party enters a key (e.g., "1234567891234567") after performing steps 2, 3, and 4, the entered key will be converted to a new one. Figure 6 shows the entered user's key and the new key after the conversion process using the MATLAB programming language. When using the same key in another encryption process (e.g., "1234567891234567"), this key will be converted to another key

(e.g., 50DF421B01F3D533). Hence, every encryption step will arbitrarily generate a new key. Figure 7 shows the prior same key entered by the user and the new key.

```
enter the key=1234567891234567
key = 1234567891234567
new_key = FACFE667D10D9F65
```

Figure 6. User's key and new key

```
enter the key=1234567891234567
key = 1234567891234567
new_key = 60EF454B1FAE3871
```

Figure 7. Prior same key and new key

5) Plaintext will be multiplied by the random matrix (4×4) that contains odd numbers. Each encipherment stage will generate an odd random matrix, thereby further complicating the coding process. Figure 8 shows two matrices generated in two various encipherment stages.

```
array =

   253   119    61   175
    71    49   245    85
    33    27    59    19
   199    81   255   247

array =

    77    55   141   233
    47   109    21    71
   251    39   131   233
    71    73   207   243
```

Figure 8. Two arrays generated in two various encryption processes

The initial encoded text will be obtained after multiplying the plaintext by the random array and divided the remainder by 256.

6) The succeeding steps will be processed using standard four steps in AES:

a. Substitute Bytes: The initial stage, sub bytes are a non-linear byte substitution. All bytes of the initial ciphertext (state) are replacement from the 16-bit S-Box that works autonomously. This study modified the S-Box, in which the standard S-Box uses constant vector, but the present research uses random vector.

Standard steps in S-BOX, this affine transformation can be determined by the following steps: i) store the multiplicative inverse of the input number in two 8-bit unsigned variables: s and x; ii) shift the value of s in the 1-bit to the left; iii) implement the XOR operation (the value of x with the value of s), placing the result in x; iv) for three additional cycles, stages 2 and 3 are rehashed and aggregated four times; and v) after obtaining the value of "x," XOR will be implemented with a constant value equal to 99 in hex 0x63. Figure 9 shows the affine transformation using the AES algorithm.

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Figure 9. Affine transformation

In this study, the fixed value was randomly changed (proposed S-BOX) between [20 and 255]. This camouflage leads to increased complexity in the decipherment process by a third party, in which a new S-Box is created in every new crypto operation owing to the random value. The reason is that the static box is

a weakness as reported in previous studies. Figure 10 shows the new affine transformation by XOR operation with random numbers.

In particular, [x0, x1, …., x7] is a random number between [20 and 255]. After executing the program, a random number equal to 136 (binary: 10001000) was generated, as shown in Figure 11. The resulting S-Box and inverse S-Box are shown in Figures 12 and 13, respectively. Figures 14 and 15 show another new S-Box, inverse S-Box after creating a new random number, in which the random number=194(11000010).

$$\begin{vmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{vmatrix} \begin{vmatrix} b0 \\ b1 \\ b2 \\ b3 \\ b4 \\ b5 \\ b6 \\ b7 \end{vmatrix} \text{XOR} \begin{vmatrix} x0 \\ x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \end{vmatrix}$$

Figure 10. Affine transformation by XOR random number

$$\begin{vmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{vmatrix} \begin{vmatrix} b0 \\ b1 \\ b2 \\ b3 \\ b4 \\ b5 \\ b6 \\ b7 \end{vmatrix} \text{XOR} \begin{vmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{vmatrix}$$

Figure 11. New affine transformation

****
* SBOX *
****

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

****

Figure 12. New S-Box

****
* INVERSE-SBOX *
****

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

Figure 13. New inverse S-Box

****
* SBOX *
****

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

Figure 14. S-Box, where r=194

****
* INVERSE-SBOX *
****

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

Figure 15. New inverse S-Box

b. Shift rows: This is called shift rows operation, which is a nonlinear operation. The value of the rotations relies on the column number (0, 1, 2 or 3) of the state array. Line 0 bytes are not rotated and lines 1, 2, and 3 are rotated left at 1, 2, and 3 bytes, respectively. Figure 16 shows the shift rows process.

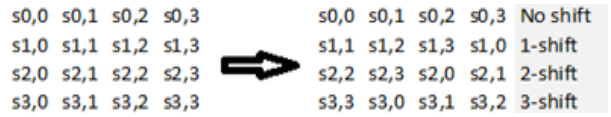


Figure 16. Shift rows

c. Mix column: In this step, each stage transformation works at the column level. It changes every column of the state to another column, in which the state columns are multiplied by a fix square array.
d. Add round key: In this step, XOR is performed for the combined column output and key, in which the total rounds are 5.

7) After these steps, XOR operation with key that eventually generated, also with the number 255 will be used to obtain the final ciphertext, thereby obtaining a high degree of complexity. As shown in Figure 17, the S-box values are produced via an affine transform on the multiplicative inverse of the Galois finite field GF (2^8).

		Y														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	00	01	8D	F6	CB	52	7B	D1	E8	4F	29	C0	B0	E1	E5	C7
	1	74	B4	AA	4B	99	2B	60	5F	58	3F	FD	CC	FF	40	EE	B2
	2	3A	6E	5A	F1	55	4D	A8	C9	C1	0A	98	15	30	44	A2	C2
	3	2C	45	92	6C	F3	39	66	42	F2	35	20	6F	77	BB	59	19
	4	1D	FE	37	67	2D	31	F5	69	A7	64	AB	13	54	25	E9	09
	5	ED	5C	05	CA	4C	24	87	BF	18	3E	22	F0	51	EC	61	17
	6	16	5E	AF	D3	49	A6	36	43	F4	47	91	DF	33	93	21	3B
	7	79	B7	97	85	10	B5	BA	3C	B6	70	D0	06	A1	FA	81	82
X	8	83	7E	7F	80	96	73	BE	56	9B	9E	95	D9	F7	02	B9	A4
	9	DE	6A	32	6D	D8	8A	84	72	2A	14	9F	88	F9	DC	89	9A
	A	FB	7C	2E	C3	8F	B8	65	48	26	C8	12	4A	CE	E7	D2	62
	B	0C	E0	1F	EF	11	75	78	71	A5	8E	76	3D	BD	BC	86	57
	C	0B	28	2F	A3	DA	D4	E4	0F	A9	27	53	04	1B	FC	AC	E6
	D	7A	07	AE	63	C5	DB	E2	EA	94	8B	C4	D5	9D	F8	90	6B
	E	B1	0D	D6	EB	C6	0E	CF	AD	08	4E	D7	E3	5D	50	1E	B3
	F	5B	23	38	34	68	46	03	8C	DD	9C	7D	A0	CD	1A	41	1C

Figure 17. Multiplicative inverse table in GF (2^8) used within the AES S-Box

The decryption process will use the following steps (inverse of the encryption process): i) enter the new password created by the sine map XOR random key; ii) read the encrypted file; iii) perform XOR operation with key that generated by sine map and 255; iv) add the round key; v) decryption round one includes inverse shift row, inverse SubByte (inverse of the new S-Box), inverse mix column, and add round key; vi) repeat steps 4 for 4 times but without inverse mix column in the final stage; and vii) cipher text is multiplied by the inverted array (4×4) (modular multiplicative inverses, such that ($a*a^{-1}$) mod 256=1). Figure 18 shows the random and inverted arrays.

Algorithm 1. Optimized AES

```
    Input: Key by user
1   Key1=mod(key,9) % convert the key between the range (0-8), (128 bit)
2     a=0.9
       pi=3.14
          for i=1:15
              key1(i+1)=a*sin(pi*key1(i));
       end
3           key2=round (key1. *100);
            key2=mod(abs(key2),256);
```

```
4   random_key=randi (6, 1, 16) %128 bit random key; values of random numbers in the
    interval [1:6]
5     Key3=XOR (Random_key, key2)
6     Input: Plaintext
7     State=mod (plaintext. *Sbox_new,256) % Perform multiplication with a new random odd
    matrix (4*4).
8     conducts first round (Add round key)
9     for k=2:3%conducts follow-on round
10              state=SubBytes %random S-BOX
11              state=ShiftRows
12              state=MixColumns
13              state=AddRoundKey
14      end
15    state=SubBytes(state);
16    state=ShiftRows(state);
17    state=AddRoundKey                          % Last round
18    For k=1:length (state)
          State=XOR (state,255)
          State=XOR (state, key3) % Cipher text is done
      end
```

Sbox_New =

129	107	247	83
211	255	67	237
255	237	255	145
53	185	55	107

Sbox_inv_new =

129	67	199	219
91	255	107	229
255	229	255	113
29	137	135	67

Figure 18. Random and inverted arrays

5. RESULTS AND EVALUATION

The following results were obtained using MATLAB on a computer with the following specifications: Intel (R) Core (TM) i7-8550U CPU @ 1.80 GHz, 2.00 GHz, and 8.00 GB RAM. The plaintext size is 16 characters (128 bit). Tables 12, 13, and 14 represents the results. All tests in Table 12 used the plain text "yehea@unisza.edu" and password "1234567891234567".

Table 12. Test show new password that derived from user's password and ciphertext

Test #	New password derived from the original password	Cipher texts in hexadecimal
1	591E5469CAB60275C8BD8BF8851D0983	D2647694A7C9EB2AD4D1AF891DCFF82
2	B0A40D1A5F32C95D20640BB1E04EFF95	A369E1244D1C89EF09413C112645A
3	6A28C7881CC91748D5464C5161A822A8	DC90789FB348A27693ADCA7696A049
4	681AF2782A27A48410E4E39A76C68228	58132573447268353DE62A61B045D7

The avalanche effect fluctuates continuously when the algorithm is executed continuously, as shown in Tables 14-16. Test result obtained after flipping single byte in the plain text (yehea@unisza.edu » yehea@Unisza.edu), as shown in Table 14. While the encrypted results for the modified AES are shown in Table 15. Each number is represented: i) encryption time (sec), ii) decryption time (sec), iii) CPU time, and iv) Avalanche.

The experiment was conducted several times and the results were recorded. Of all the results, the value (73.438) was the best for the avalanche. The text used was "11111111111111111111111111111111" in hexadecimal, which is equal in decimal to "17171717171717171717 17 17 17 17 1717", where the plaintext bit was changed to "01" instead of "11" as shown in Table 16. The results were compared with the original AES and other previous research papers as shown in Table 17.

Throughput for OAES has been shown in Table 18. The comparisons of results are shown in Figure 19. Figure 20 shows encrypted and plaintext after execute MATLAB programming, while Figure 21 shows the experiment result obtained with an avalanche effect ratio of 72.656. Table 19 shows a comparison of files of different sizes in terms of encryption and decryption execution time. Figures 22 and 23 represents graphs for encryption and decryption time, respectively.

Table 13. Histogram analysis

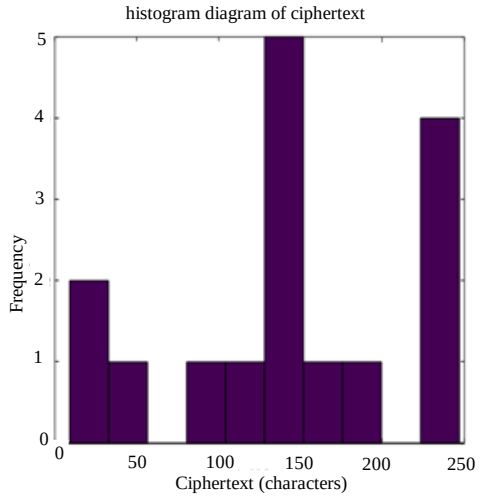
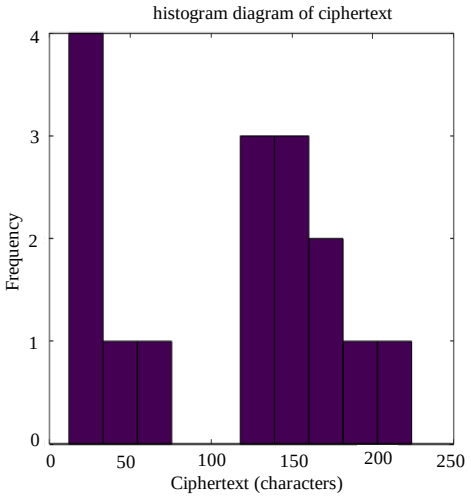
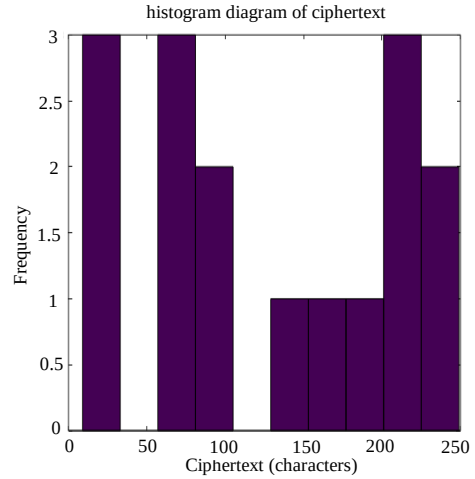
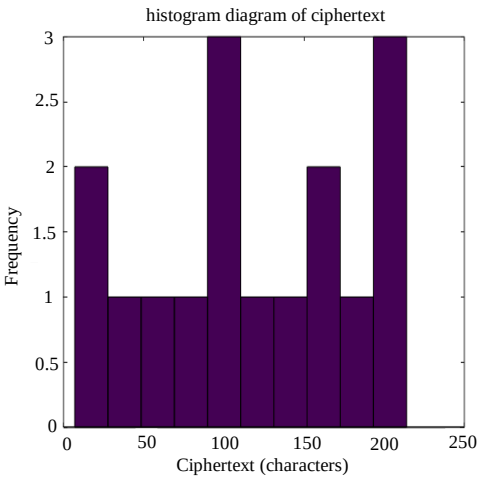
Test #	Histogram analysis	Test #	Histogram analysis
1		2	
3		4	

Table 14. Test of various encryption parameters

Test #	1	2	3	4
T1	0.47	0.967204	1.0835	49.219
T2	0.47	0.963212	1.0858	50.781
T3	0.567	1.07465	1.0989	53.906
T4	0.565	0.919029	1.0211	56.250
T5	0.431	0.948088	1.0742	65.625

Table 15. Modified AES ciphertext results

Test #	Encrypted Plaintexts	Encrypted Plaintexts with One Bit Flipped
T1	49 C4 86 79 F1 61 7A 1D 56 CD AE 66 13 5A 37 5B	3C D1 3A 2B A1 79 C1 F0 75 7C 9F A4 AF 8C D0 72
T2	57 6A 5A 42 C0 F1 9E F9 B0 A4 83 44 CB 46 11 05	40 1B BE 77 B0 DE 2C 9C 41 6F 7C B3 BE 5B D8 63
T3	97 16 20 06 61 8D ED BD 60 BD E4 8C B4 BF AB 8F	18 B3 E5 F9 CD 91 BE 2B 6C FC 6B 00 3B E3 73 3A

Table 16. Avalanche test results after changing one bit in the plain text (from (11) to (01)

Test #	Input data	Number of flipped bits	Avalanche effect %
1	11,11,11,11,11,11,11,11, 11,11,11,11,11,11,11,11	62	48.438
2		66	51.562
3		73	57.031
4	01,11,11,11,11,11,11,11, 11,11,11,11,11,11,11,11	58	45.312
5		77	60.156
6		92	71.875

Table 17. The avalanche test comparison

Algorithm	Avalanche effect %
OPTIMIZED AES	72.656
ORIGINAL AES	50.78
[24]	54.68
[16]	57.03

Table 18. Throughput value during ciphering for files of different sizes

Text file size (KB)	Encryption time (second)	
	AES [16]	PROPOSED
2	24	14.9863
3	72	25.4973
4	143	32.1341
5	291	42.7479
6	481	48.8171
7	692	58.0047
AVERAGE TIME	283.8	222.1874

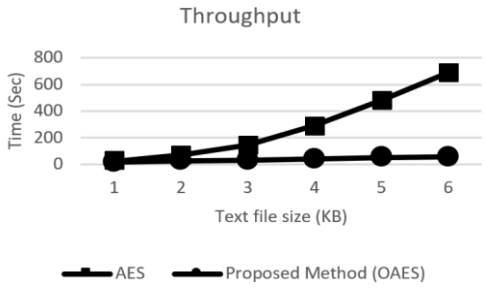


Figure 19. Results comparisons

```
>> Cipher

Reading file [C:\Users\GTS\Desktop\to deny random\OAES BYTE\Optimized AES-chang byt
contents
In is:
yehea@unisza.edu

 Ciphering..yl = XBKثب~TTJW"
Done
Elapsed time is 0.374795 seconds.
ciphered Contents SAVED to file [cC:\Users\GTS\Desktop\to deny random\OAES BYTE\Opt
ng byte\yeheal.txt].
CPU Usage
 1.2857
********************
>> InvCipher

Reading file [C:\Users\GTS\Desktop\to deny random\OAES BYTE\Optimized AES-chang byt
 contents.
Deciphering..In2 = yehea@unisza.edu
Done
Elapsed time is 0.570465 seconds.
Deciphered (Original) file contents restored into [C:\Users\GTS\Desktop\to deny ran
Optimized AES-chang byte\dcyeheal.txt] file.
```

Figure 20. Plaintext and cipher text by optimized AES

```
In =

  17   17   17   17   17   17   17   17   17   17   17   17   17   17   17   17
   1   17   17   17   17   17   17   17   17   17   17   17   17   17   17   17

Ciphering...Done
Elapsed time is 0.335825 seconds.
ciphered Contents SAVED to file [ctest1111.txt].
CPU Usage
 1.0184
Avallanch =  72.656
//////////////////////////\n
avg =  72.656
```

Figure 21. Avalanche effect result

Table 19. Execution time for encryption and decryption for various files

File size (KB)	Encryption execution time (s)			Decryption execution time (s)		
	AES	[16]	Optimized AES	AES	[16]	Optimized AES
2	24	15	14.9863	24	15	37.933
3	72	40	25.4973	72	40	56.498
4	143	125	32.1341	143	125	79.0607
5	291	240	42.7479	291	240	95.3293
6	481	425	48.8171	471	425	115.209
7	692	598	58.0047	692	598	133.92

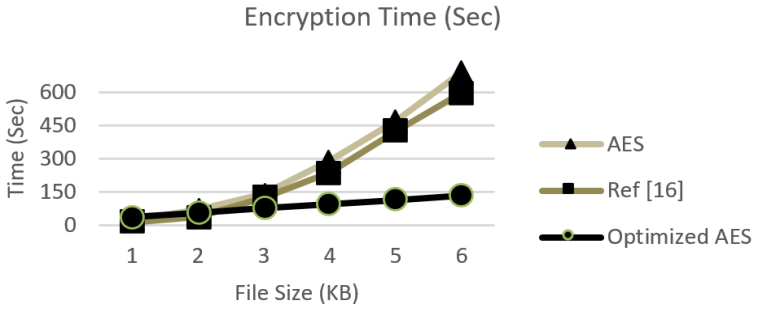


Figure 22. Graph of encryption time

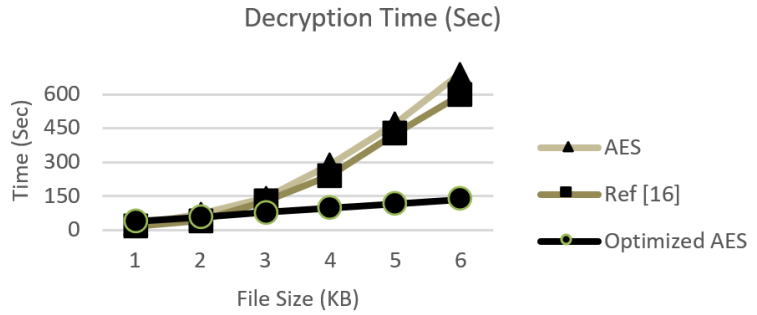


Figure 23. Graph of decryption time

6. CONCLUSION

This research introduced an execution of the optimized AES algorithm for encrypting and decrypting text. A series of tests were performed, and the results indicated that despite using the same password and the same text, different encoded texts were produced each time. This outcome indicates that the encoded text of the original text is constantly changing (in every new encryption process), which is a strong indication that the proposed algorithm is highly resistant to attacks. Statistical and differential experiments indicated that the avalanche effect reached 72.656, it means satisfies the strict avalanche criterion (SAC). The

effect of the avalanche depends on both the encryption key and the input text, random numbers, random array, and dynamic S-BOX. Moreover, the encoding and decoding time was shown to be reduced as a result of the reduction of the algorithm cycles.

REFERENCES

[1] C. R. Gopal and P. RajKumar, "Optimized approach for secure communication using DES algorithm," in *International Journal of Pure and Applied Mathematics*, 2017, vol. 116, no. 21, pp. 125–130, doi: 10.17758/eirai.f0817106.

[2] H. K. Hoomod and M. H. Zewayr, "AES modification using slantlet transform and lorenz chaotic system," *International Journal of Recent Trends in Engineering*, vol. 02, no. 06, pp. 561–578, 2016.

[3] Y. Alemami, M. A. Mohamed, and S. Atiewi, "Research on various cryptography techniques," *International Journal of Recent Technology and Engineering*, vol. 8, pp. 395–405, 2019, doi: 10.35940/ijrte.B1069.0782S319.

[4] B. Maram and J. M. Gnanasekar, "A block cipher algorithm to enhance the avalanche effect using dynamic key-dependent S-Box and genetic operations," *International Journal of Pure and Applied Mathematics*, vol. 119, no. 10, pp. 399–418, 2018.

[5] C. P. Dewangan, S. Agrawal, A. K. Mandal, and A. Tiwari, "Study of avalanche effect in AES using binary codes," in *2012 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, 2012, no. 978, pp. 183–187.

[6] A. M. Babu, K. J. Singh, "Performance evaluation of chaotic encryption technique," *American Journal of Applied Sciences*, vol. 10, no. 1, pp. 35–41, Jan. 2013, doi: 10.3844/ajassp.2013.35.41.

[7] S. D. Rihan, A. Khalid, and S. E. F. Osman, "A performance comparison of encryption algorithms AES and DES," *International Journal of Engineering Research and Technology*, vol. 4, no. 12, pp. 151–154, 2015.

[8] A. Singh, "Comparatively analysis of AES, DES and SDES algorithms," *International Journal of Computer Science And Technology*, vol. 8491, no. 3, pp. 200–202, 2014.

[9] K. P. Karule and N. V. Nagrale, "Comparative analysis of encryption algorithms for various types of data files for data security," *International Journal of Scientific Engineering and Applied Science (IJSEAS)*, vol. 2, no. 2, pp. 495–498, 2016.

[10] M. M. Bachtiar, T. H. Ditanaya, S. Wasista, and R. R. K. Perdana, "Security enhancement of AES based encryption using dynamic salt algorithm," in *Proceedings of the 2018 International Conference on Applied Engineering, ICAE 2018*, 2018, pp. 1–6, doi: 10.1109/INCAE.2018.8579381.

[11] M. I. Reddy, "A modified advanced encryption standard algorithm," *Journal of Mechanics of Continua and Mathematical Sciences*, no. 1, pp. 112–117, 2020, doi: 10.26782/jmcms.spl.5/2020.01.00027.

[12] A. Hafsa, A. Sghaier, M. Zeghid, J. Malek, and M. Machhout, "An improved co-designed AES-ECC cryptosystem for secure data transmission," in *International Journal of Information and Computer Security*, 2020, vol. 13, no. 1, pp. 118–140, doi: 10.1504/IJICS.2020.108145.

[13] J. H. Anajemba, C. Iwendi, M. Mittal, and T. Yue, "Improved advance encryption standard with a privacy database structure for IoT nodes," in *Proceedings-2020 IEEE 9th International Conference on Communication Systems and Network Technologies, CSNT 2020*, 2020, pp. 201–206, doi: 10.1109/CSNT48778.2020.9115741.

[14] K. Mohamed, F. H. H. M. Ali, S. Ariffin, N. H. Zakaria, and M. N. M. Pauzi, "An Improved AES S-box based on Fibonacci numbers and prime factor," *International Journal of Network Security*, vol. 20, no. 6, pp. 1206–1214, 2018.

[15] B. N. Rao, D. Tejaswi, K. A. Varshini, K. P. Shankar, and B. Prasanth, "Design of modified AES algorithm for data security," *International Journal for Technological Research in Engineering*, vol. 4, no. 8, pp. 1289–1292, 2017.

[16] N. A. A. Mohd and A. Y. A. Ashawesh, "Enhanced AES algorithm based on 14 rounds in securing data and minimizing processing time," *Journal of Physics: Conference Series*, vol. 1793, no. 1, Feb. 2021, doi: 10.1088/1742-6596/1793/1/012066.

[17] R. Riyaldhi and A. Kurniawan, "Improvement of advanced encryption standard algorithm with shift row and S-Box modification mapping in mix column," *Procedia Computer Science*, vol. 116, pp. 401–407, 2017, doi: 10.1016/j.procs.2017.10.079.

[18] F. V. Wenceslao, "Enhancing the performance of the advanced encryption standard (AES) algorithm using multiple substitution boxes," *International Journal of Communication Networks and Information Security*, vol. 10, no. 3, pp. 496–501, 2018, doi: 10.17762/ijcnis.v10i3.3589.

[19] V. L. Hallappanavar, B. P. Halagali, and V. V Desai, "Efficient implementation of AES by modifying S-Box," *IOSR Journal of Computer Science (IOSR-JCE)*, pp. 35–39, 2014.

[20] H. V. Gamido, "Implementation of a bit permutation-based advanced encryption standard for securing text and image files," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 19, no. 3, pp. 1596–1601, Sep. 2020, doi: 10.11591/ijeecs.v19.i3.pp1596-1601.

[21] Z. Lu and H. Mohamed, "A complex encryption system design implemented by AES," *Journal of Information Security*, vol. 12, no. 2, pp. 177–187, 2021, doi: 10.4236/jis.2021.122009.

[22] A. A. Thinn and M. M. S. Thwin, "Modification of AES algorithm by using second key and modified subbytes operation for text encryption," in *Lecture Notes in Electrical Engineering*, 2019, vol. 481, pp. 435–444, doi: 10.1007/978-981-13-2622-6_42.

[23] N. Siddaiah, A. Moduli, K. Bhargavram, P. M. Krishna, B. Rakesh, and G. V Ganesh, "A novel approch to modified advanced encryption standard algorithm," *Journal of Critical Reviews*, vol. 7, no. 2. pp. 629–633, 2020, doi: 10.31838/jcr.07.02.115.

[24] H. M. Azzawi, "Enhancing the encryption process of advanced encryption standard (AES) by using proposed algorithm to generate S-Box," *Journal of Engineering and Development*, vol. 18, no. 2, pp. 89–105, 2014.

[25] G. Nissar, D. K. Garg, and B. U. I. Khan, "Implementation of security enhancement in AES by inducting dynamicity in AES S-Box," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 11, pp. 2364–2373, Sep. 2019, doi: 10.35940/ijitee.J9311.0981119.

[26] Y. Bhavani, S. S. Puppala, B. J. Krishna, and S. Madarapu, "Modified AES using dynamic S-Box and DNA cryptography," in *Proceedings of the 3rd International Conference on I-SMAC IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2019*, 2019, pp. 164–168, doi: 10.1109/I-SMAC47947.2019.9032516.

[27] A. Shakiba, "Generating dynamical S-boxes using 1D Chebyshev chaotic maps," *Journal of Computing and Security*, vol. 7, no. 1, pp. 1–17, 2020, doi: 10.22108/jcs.2020.116547.1023.

[28] M. Education, "Avalanche analysis of variant polynomials for AES," *Turkish Journal of Computer and Mathematics Education*, vol. 12, no. 14, pp. 2696–2703, 2021.

[29] Y. Alemami, M. A. Mohamed, S. Atiewi, and M. Mamat, "Speech encryption by multiple chaotic map with fast Fourier transform," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 6, pp. 5658–5664, Dec. 2020, doi: 10.11591/ijece.v10i6.pp5658-5664.

[30] A. Kumar, "Effective implementation and avalanche effect of AES," *International Journal of Security, Privacy and Trust Management*, vol. 1, no. 3, pp. 31–35, Aug. 2012, doi: 10.5121/ijsptm.2012.1303.
[31] M. Faheem, S. Jamel, A. Hassan, Z. A. Pindar, N. Shafinaz, and M. Mat, "A survey on the cryptographic encryption algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 11, pp. 333–344, 2017, doi: 10.14569/ijacsa.2017.081141.

BIOGRAPHIES OF AUTHORS

Yahia Alemami received the B.Sc. degree in Computer Science from AL Hussein Bin Talal University, Maan, Jordan, in 2004, and the master's degree in computer engineering from Anadolu University, Turkey, in 2012. He worked at Al-Hussein Bin Talal University in Jordan as a lecturer since 2013 until now. He can be contacted at email: Yehea_m@ahu.edu.jo.

Mohamad Afendee Mohamed received the Ph.D. degree in mathematical cryptography from Universiti Sultan Zainal Abidin, in 2012. He currently serves as an Associate Professor for Universiti Sultan Zainal Abidin. His research interests include both theoretical and application issues in the domain of data security, and mobile and wireless networking. He can be contacted at email: mafendee@unisza.edu.my.

Saleh Atiewi received the B.Sc. degree in Computer Science from Al-Isra University, Amman, Jordan, in 1999, the Master degree in Internet Technology from the Wollongong University, Wollongong Australia, 2004, and the Ph.D. degree in Computer Science from Tenaga Nasional University, Putrajaya, Malaysia, in 2017 Since 2004, he has been with Al Hussein Bin Talal University, Maan, Jordan, where he is currently an Assistant Professor with the department of Computer Science, He is currently head of Computer Science department, His research interests are in the areas of Network Security, Cloud computing and Internet of things. He can be contacted at email: saleh@ahu.edu.jo.