A Novel Method to Increase Diffusion and Confusion in AES Algorithm

Zakria Mahrousa
Dep Computer Engineering
Faculty of Electrical and Electronic Engineering
University of Aleppo

Ahmad Bitar
Dep Computer Engineering
Faculty of Electrical and Electronic Engineering
University of Aleppo

Yahia Fareed
Communications Engineering
Faculty of Electrical and Electronic Engineering
University of Aleppo

ABSTRACT
The Advanced Encryption Standard (AES) is the most famous amongst symmetric block cipher algorithms, and the most used one in many different applications, because its encryption strength and its resistance to many attacks. In this paper, a novel approach will be introduced to remove some weakness points in AES and increase its security. Unlike most previous researches, this research will improve the most important and powerful part of AES algorithm which is the MixColumns() transformation. In the original AES, there only one function is used, which it creates a fixed array that is used in MixColumns() transformation, and this fixed array is known by attackers. Alternatively, the expanded secret key will be used to generate a different function at each round of AES. These functions will create a variable dynamic arrays at each round based on expanded secret key. The variable dynamic arrays will increase confusion amongst bits of the encrypted text. After that, the ShiftRow() transformation will be complicated from ShiftRow() with fixed pattern to ShiftRowColumn() with variable dynamic pattern according to expanded secret key. The modified ShiftRowColumn() will increase the diffusion amongst bytes of encrypted text, as will be detailed in experimental results.

General Terms
Information Security, Block cipher, symmetric, asymmetric.

Keywords
AES, NIST, encryption, decryption, Galois Filed, secret key, extended secret key, round, Mix Columns, Shift Rows Columns, function, inverse, attack, cryptanalysis.

1. INTRODUCTION
In recent years, information security has become the most important core in communication engineering, because of the continuous increasing in using of digital data which are transmitted over internet [1]. The transmission of sensitive digital data through communication channels requires speed and confidentiality for digital communication network to achieve three security requirements. The requirements are integrity, confidentiality and availability [2][3]. One of the most significant forms of information protection technology is encryption [4][5], which is a very important process to ensure the confidentiality of data that are transmitted over the internet [6] [7] [8]. Encryption algorithms consist of three types, Symmetric encryption algorithms, Asymmetric encryption algorithms and Hash functions. In Symmetric encryption, the same key is used in encryption and decryption process. It is called the secret key. That the message is encrypted by the sender and decrypted by receiver using the same secret key. Symmetric encryption includes many algorithms such as DES-3DES-RCn-Blowfish-Towfish-AES [9]. These algorithms use traditional encryption techniques that have some disadvantages in the field of information security [6]. Therefore, in this paper will be proposed an improved algorithm in the field of symmetric encryption. AES algorithm is symmetric block cipher algorithm which is the most common and used one in many different applications [10]. AES contains MixColumns() transformation that is heart and strength of AES and center of confidentiality and confusion, as MixColumns() transformation is responsible of diffusion and confusion amongst bits of encrypted text [11].

In this paper, the security of the advanced encryption algorithm AES will be Improved with keeping on low execution time. The improved algorithm is more resistant to attackers and cryptanalysis. Firstly, the static MixClomuns() array will be removed which is known by attackers and analysts by generating a different function for each round of AES based on the extended secret key. This function at a specified round generates a round array for MixColumns() transformation. As a result, the weakness point of relying on a single function and array in the encryption process will be removed. Alternatively, we have a different function and a different array for each round of AES. These functions and arrays are variable according to the extended secret key at each round. Therefore, they are unknown to attackers and cryptanalysis and the confusion in bits of encrypted text will be increased. Secondly, the ShiftRows() transformation that shifts rows only and has a fixed pattern in all rounds of AES will be complicated to become ShiftRowsColumns(i) transformation that shifts rows and columns and has a dynamic variable pattern with the expanded secret key at each round, as each round of AES is provided with different horizontal and vertical shift indexes from the other round. Consequently, will be obtained a different shift pattern at each round. In all previous modifications, this work carefully maintained on the mathematical bases that laid down by the NIST (National Institute standard and Technology) in original algorithm. this paper is organized as follows: section 2 gives an overview of related works to AES and reported enhancements in previous researches. Section 3 overviews structure of AES. Section 4 shows proposed algorithm of generating dynamic MixColumns() functions and arrays, while section 5 shows proposed ShiftRowCloumns() transformation, in section 6 simulation and result will be shown. Finally, section 7 will be for conclusion.

2. RELATED WORK
This section shows some previous researches that tried to improve the performance and security of the AES algorithm. In [12] the researchers improved the structure of MixColumns

transformation, as this transformation was redesigned by eliminating the excessive logical functions for the efficient and speed implementation of the AES algorithm on FBGA chipsets. In [13] several structures were applied to MixColumns() using finite filed techniques and selecting the best structure after applying certain tests on the different structures. In [14] [15] MixColumns() transformation was eliminated and replaced by a new technique based on chaotic system by applying a random map of type (Henon chaotic map), which provided a good diffusion and a reduced execution time. In [16] researchers replaced the Mix Column Transformation in AES by MDS Matrices which are based on default MDS Matrix of AES and mbit additional key. Then, they are implemented on DSP and FPGA where Both have their own advantages in embedded systems. In [17] for more confusion and until AES becomes more resistant to attackers, the number of rounds in AES have been increased from 10 rounds to 16 rounds. as a result, more MixCloumns and ShiftRow are applied in the improved algorithm, but this improvement increase execution time approximately twice. In [18] [19] in order to improve the performance of the algorithm, the MixColumns transformation was replaced by a permutation algorithm. The MixColumns transformation was time-consuming, so the execution time is improved but the security is decreased because the permutation algorithm didn't provide the diffusion and confusion as MixColumns() transformation. In [20] the execution time of AES was reduced by relying on new boxes and improving the key scheduling process. The improvement of execution time was 35%. In [21] the researchers used multiple Sbox in AES depending on two techniques, the first technique was Rijndael's Sbox and the second technique was constructing Sbox based on Xor operation and affine transformation, then replacing the Mixcolumns() with this Sboxes.

-There are few points overlooked by previous researches:

- They Did not maintain on the basic mathematical models that set by NIST, as the MixColumns() transformation has been replaced by other equivalents. Some of which have no clear mathematical models and no clear proofs. Moreover, they do not provide confusion and diffusion that were provided by original MixColumns() transformation.
- Some modifications improved the time and ignored the security. This did not have a balance between time and security.
- They do not find variable dynamic mechanism with the secret key for the MixColumns transformation. this paper will try to remedy these above points in the new improvements.

3. AVANCED ENCRYPTIN STANDARD

The AES encryption algorithm contains three encryption patterns based on the length of the secret key and number of Nr (Number round) as shown in Table 1:

Table 1. AES algorithm patterns

Number round	Key Size
10	128bit
12	192bit
14	256bit

The size of input block of plain text is always 128bit in the previous three patterns. This input block is stored in two dimensional array 4*4bytes. It is called State array which contains 16 bytes or four words and each word contains of 4 bytes [22].

3.1 Structure of Round in AES Algorithm

The round in AES consist of four transformations as follow:

1. SubByte(): in this transformation, each byte of the State array is replaced by another byte according to a fixed matrix called "SBox []" its size is 16 * 16 bytes. This transformation provides non-linearity and confusion by inverse multiplication in finite field and Affine Transformation.
2. ShiftRows(): This transformation provides the intra-column diffusion of the State array as the last three rows are cyclically shifted.
3. MixColumn(): provides diffusion in byte of State array, as columns in the State array multiply by a fixed matrix, and the multiplication is performed over Galois Field GF (2^8) [11].
4. AddRoundKey(): This process provides confusion, as XOR is performed at every byte of the key with the corresponding byte of the State array [22][11].

Next figure 1 illustrates 128-bit encryption process, it begins with the Addroundkey(), and it is followed by ten rounds, and each round consists of the four transformations which are SubBytes(), ShiftRows(), MixColums(), Addroundkey(). These transformations will take place sequentially, noting that the last round will include only three transformations which are SubBytes(), ShiftRows(), Addroundkey(). In the decryption process, transformations SubByte(), Shift Row(), and MixColumns() will be replaced by their inverse that are InvSubBytes(), InvShiftRows(),InvMixColumns(), and they take another arrangement [23].

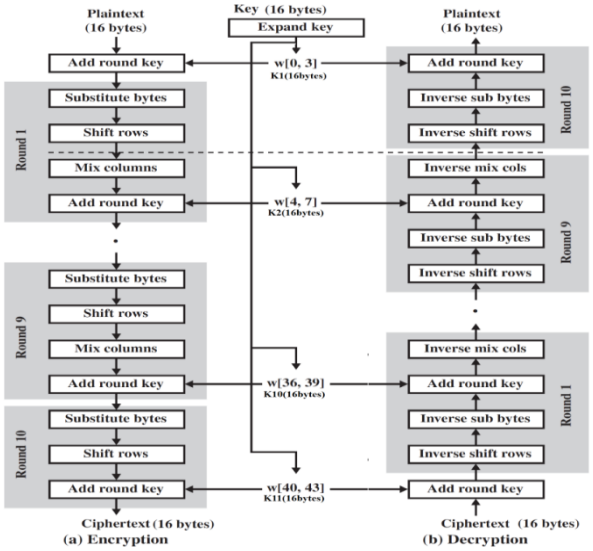

Fig 1: detailed structure of the AES algorithm [24]

4. GENERATE VARIABLE FUNCTIONS AND VARAIBLE ARRAYS

In this section, the mathematical basics of MixColumns() transformation in AES will be demonstrated, that are based on finite fields. Then, the proposed method for generating variable functions and matrices for MixColumns(i) and

InvMixColumns(i) transformation will be explained.

4.1 Multiply of Two Polynomials Their Coefficients Belong to Finite Field GF(2^8) and They are Over GF(2^4)

Let's suppose that we have a polynomial a(x), which consists of four terms, as follows:

$$a(x)=a_3 x^3+a_2 x^2+a_1 x+ a_0$$

its coefficients are [a3, a2, a1, a0] belong to Galois Field GF(2^8), so each coefficient [a3, a2, a1, a0] size is 1byte, and all mathematical operations of these coefficients perform on modulo ($x^8+x^4+x^3+x+1$) or (100011011). Let suppose a polynomial a(x) is over field GF(2^4) according to the modulo (x^4+1) and Let's have a polynomial b(x) definition as same as a(x) as follows:

$$b(x) = b_3 x^3+ b_2 x^2+ b_1 x+ b_0$$

The multiplication process of two polynomials a(x) and b(x) is accomplished in two stages. In the first stage The multiplication process is expanded mathematically c (x) = a (x) • b (x), therefore we can write the product c (x) as:

$$c(x) = c_6 x^6+ c_5 x^5+ c_4 x^4+ c_3 x^3+ c_2 x^2+ c_1 x^1+ c_0$$

where:

$c_0= a_0 \bullet b_0$; $c_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1$;

$c_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2$;

$c_3 = a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3$;

$c_4 = a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_1$;

$c_5 = a_3 \bullet b_2 \oplus a_2 \bullet b_3$;

$c_6 = a_3 \bullet b_3$;

Note that the polynomial c(x) has a sixth order (6), since the polynomials are over field GF (2^4) according to modulo (x^4+1). Therefore in the second stage, the order of C (x) is reduced according to modulo (x^4+1) to become a third order. Note the following equation (1):

$$\mathbf{x^i mod\ (x^4 + 1) = x^{imod4}} \qquad (\mathbf{1})$$

after applying equation (1) to c(x) we obtain d(x) as follows:

$$d(x) = d_3 x^3+ d_2 x^2+ d_1 x+ d_0$$ where:

$d_0=(a_0\bullet b_0)\oplus(a_3\bullet b_1) \oplus(a_2\bullet b_2)\oplus(a_1\bullet b_3)$
$d_1=(a_1\bullet b_0)\oplus(a_0\bullet b_1)\oplus(a_3\bullet b_2)\oplus(a_2\bullet b_3)$
$d_2=(a_2\bullet b_0)\oplus(a_1\bullet b_1)\oplus(a_0\bullet b_2)\oplus(a_3\bullet b_3)$
$d_3=(a_3\bullet b_0)\oplus(a_2\bullet b_1)\oplus(a_1\bullet b_2)\oplus(a_0\bullet b_3)$

therefore we can arrangement coefficients of a(x), b(x), and d(x) as follows in equation (2) [25][11]:

$$\begin{bmatrix}\mathbf{d_0}\\ \mathbf{d_1}\\ \mathbf{d_2}\\ \mathbf{d_3}\end{bmatrix} = \begin{bmatrix}\mathbf{a_0} & \mathbf{a_3} & \mathbf{a_2} & \mathbf{a_1}\\ \mathbf{a_1} & \mathbf{a_0} & \mathbf{a_3} & \mathbf{a_2}\\ \mathbf{a_2} & \mathbf{a_1} & \mathbf{a_0} & \mathbf{a_3}\\ \mathbf{a_3} & \mathbf{a_2} & \mathbf{a_1} & \mathbf{a_0}\end{bmatrix} * \begin{bmatrix}\mathbf{b_0}\\ \mathbf{b_1}\\ \mathbf{b_2}\\ \mathbf{b_3}\end{bmatrix} \qquad (\mathbf{2})$$

We conclude that when we multiplying a constant function $a(x) = a_3 x^3 + a_2 x^2 + a_1x^1 + a_0$ its coefficients are over GF(2^8) with modulo ($x^8 + x^4 + x^3 + x + 1$). It is over GF(2^4) with modulo ($x^4 + 1$) by a variable polynomial b(x). Its definition as same as a(x), the constant coefficients of a(x) can be arranged by an array as in equation (2) and the coefficients of variable polynomial b(x) is arranged by a column vector as in equation (2). The coefficients of d (x) is in a resulting column vector, noting that the multiplication process of coefficients takes place according to the field GF(2^8) and modulo ($x^8 + x^4 + x^3 + x + 1$). In MixColumns() of AES algorithm, the constant polynomial $a(x)= \{03\}x^3+ \{01\}x^2+ \{01\}x+ \{02\}$ is multiplied by each column of the State array that representing b(x) to get a new column which is the transformation array column [24][26].

4.2 The Transformation MixColumns() and InvMixColumns()

MixColumns() transformation is the center of cryptography and confusion, and heart of AES algorithm. This transformation is the most consuming of time amongst other transformations. The input of MixColumns() is a 4 * 4bytes State array. Each column of the State array is treated separately and this column represent the polynomial b(x). The constant polynomial $a(x) = \{03\}x^3+ \{01\}x^2 + \{01\}x + \{02\}$ is multiplied by the b(x), where a(x) and b(x) over GF(2^4) according to the modulo (x^4+1), and all mathematical operations of their coefficients perform on over GF(2^8) with modulo ($x^8+x^4+x^3+x+1$). a(x) coefficients are represented by the constant matrix as in equation (3) In the encryption process [21] [13] [27] [2] [28]:

$$\begin{bmatrix}s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3}\\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3}\\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3}\\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3}\end{bmatrix} = \begin{bmatrix}\mathbf{02} & \mathbf{03} & \mathbf{01} & \mathbf{01}\\ \mathbf{01} & \mathbf{02} & \mathbf{03} & \mathbf{01}\\ \mathbf{01} & \mathbf{01} & \mathbf{02} & \mathbf{03}\\ \mathbf{03} & \mathbf{01} & \mathbf{01} & \mathbf{02}\end{bmatrix} * \begin{bmatrix}s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3}\\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3}\\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3}\\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3}\end{bmatrix} \quad (\mathbf{3})$$

The function that is used in the decryption process as follows:

$a^{-1}(x)=b(x)=\{0b\}x^3+\{0d\}x^2+\{09\}x+\{0e\}$

$a^{-1}(x)$ is the invers a(x) according to modulo $(1+x^4)$[24][11]:

$(a^{-1}(x)*a(x))mod(1+x^4)=1$

The general equation for InvMixColumns() transformation in the decryption process is as follows in equation (4):

$$\begin{bmatrix}s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3}\\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3}\\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3}\\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3}\end{bmatrix} = \begin{bmatrix}\mathbf{0E} & \mathbf{0B} & \mathbf{0D} & \mathbf{09}\\ \mathbf{09} & \mathbf{0E} & \mathbf{0B} & \mathbf{0D}\\ \mathbf{0D} & \mathbf{09} & \mathbf{0E} & \mathbf{0B}\\ \mathbf{0B} & \mathbf{0D} & \mathbf{09} & \mathbf{0E}\end{bmatrix} * \begin{bmatrix}s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3}\\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3}\\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3}\\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3}\end{bmatrix} \quad (\mathbf{4})$$

4.3 Proposed Algorithm for Generating a Variable Functions and Arrays

The AES algorithm consists of ten rounds, as MixColumns() exists in first nine rounds of encryption process, InvMixColumns() exists in the first nine rounds of decryption process, and each round has a specified extended secret key ki (16bytes) see figure(1). Next figure(2) illustrates the proposed method for generating variable functions and arrays according to the secret key for MixColumns() and InvMixColumns() transformation**.**

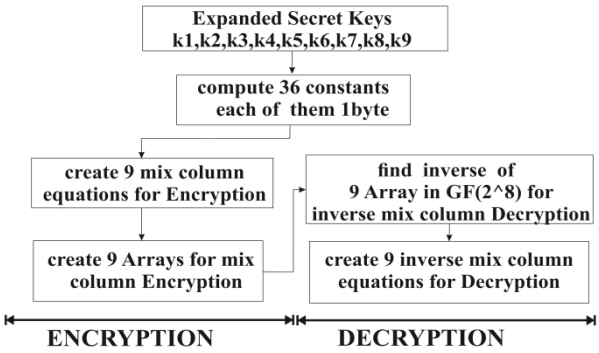

Fig 2: proposed algorithm for generating functions and arrays

After observing Figure 2, we can arrange the mechanism of the proposed algorithm as follows:

-the proposed algorithm inputs is the first nine expanded keys whish are [k1, k2, k3, k4, k5, k6, k7, k8, k9].

-step1: 36 constants are calculated [c1, c2,. ,c36],where size each of them is (1byte). The first four constants [c1, c2, c3, c4] are calculated from the first extended secret key k1 and are assigned to the first round (round (1)) in the encryption process for MixColumns(1). The second four constants [c5, c6, c7, c8] are calculated from k2 and are assigned to the second round (round2) in the encryption process for MixColumns(2), and so on until the last four constants [c33, c34, c35, c36] are calculated from k9 and are assigned to the round(9) in encryption process for MixColumns(9). The following equation(5) illustrates the calculation of four constants from expanded key Ki that are assigned to round(i) where $1 \le i \le 9$ and $\oplus$ is xor operation:

$$\left.\begin{aligned} \mathbf{c_{i*4-3}} &= \mathbf{k_i[0]} \oplus \mathbf{k_i[1]} \oplus \mathbf{k_i[2]} \oplus \mathbf{k_i[3]} \\ \mathbf{c_{i*4-2}} &= \mathbf{k_i[4]} \oplus \mathbf{k_i[5]} \oplus \mathbf{k_i[6]} \oplus \mathbf{k_i[7]} \\ \mathbf{c_{i*4-1}} &= \mathbf{k_i[8]} \oplus \mathbf{k_i[9]} \oplus \mathbf{k_i[10]} \oplus \mathbf{k_i[11]} \\ \mathbf{c_{i*4}} &= \mathbf{k_i[12]} \oplus \mathbf{k_i[13]} \oplus \mathbf{k_i[14]} \oplus \mathbf{k_i[15]} \end{aligned}\right\} \quad (5)$$

where the size of each extended key ki is (16bytes), and it is stored in linear array ki [0]….ki [15], and also calculated constants are stored in a linear array C [1 * 36bytes].

-step2: Nine functions [a1(x), a2(x), a3(x), a4(x), a5(x), a6(x), a7(x), a8(x), a9(x)] are created, where a1(x) is used for MixColumns(1) transformation in Round(1), and a2(x) is used for MixColumns(1) in Round(2), so on a9(x) is used for MixColumns(9) in Round(9). The following equation(6) illustrates formation of nine functions ai(x) from calculated constants C where $1 \le i \le 9$:

$$a_i(x) = C_{(i*4)}\,X^3 + C_{(i*4-1)}\,X^2 + C_{(i*4-2)}\,X + C_{(i*4-3)} \quad (6)$$

where coefficients of each function $a_i(x)$ over finite field $GF(2^8)$ with modulo $(x^8+x^4+x^3+x+1)$ and ai(x) over $GF(2^4)$ with modulo (x^4+1).

-step3: Nine arrays [m1, m2, m3, m4, m5, m6, m7, m8, m9] are generated from the nine preceding functions[a1(x)… a9(x)], as each ai(x) function generates a MixColumns(i) transformation array which is mi that assigns to round (i) at encryption process. Next figures(3) illustrates the method of forming the mi array from ai(x) function for $1 \le i \le 9$:

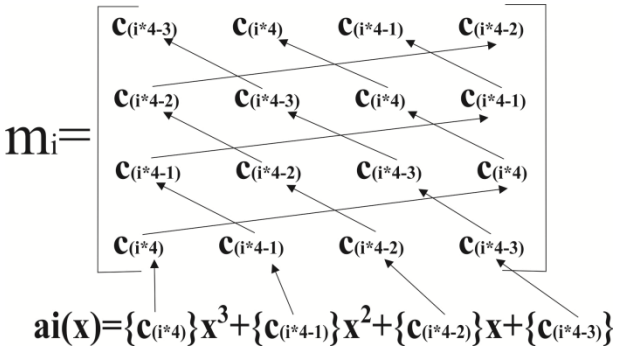

Fig 3: Formation of MixColumns(i) array mi from the function ai(x) that belong to round (i)

Notice: the order of ai(x) coefficients in the fourth row of mi array is not changed, but in the third, second, and first row, coefficients are circularly shifted to left. Next Figure 4 illustrates distribution of [m1, ……, m9] arrays and their assignment to rounds at the encryption process. The general equation for the MixColums(i) transformation in round(i) at encryption process as follows for $0 \le k \le 3$ and $1 \le i \le 9$:

$$\begin{bmatrix} s'_{0,k} \\ s'_{1,k} \\ s'_{2,k} \\ s'_{3,k} \end{bmatrix} = \begin{bmatrix} C_{i*4-3} * s_{0,k} \oplus C_{i*4} * s_{1,k} \oplus C_{i*4-1} * s_{2,k} \oplus C_{i*4-2} * s_{3,k} \\ C_{i*4-2} * s_{0,k} \oplus C_{i*4-3} * s_{1,k} \oplus C_{i*4} * s_{2,k} \oplus C_{i*4-1} * s_{3,k} \\ C_{i*4-1} * s_{0,k} \oplus C_{i*4-2} * s_{1,k} \oplus C_{i*4-3} * s_{2,k} \oplus C_{i*4} * s_{3,k} \\ C_{i*4} * s_{0,k} \oplus C_{i*4-1} * s_{1,k} \oplus C_{i*4-2} * s_{2,k} \oplus C_{i*4-3} * s_{3,k} \end{bmatrix} \quad (7)$$

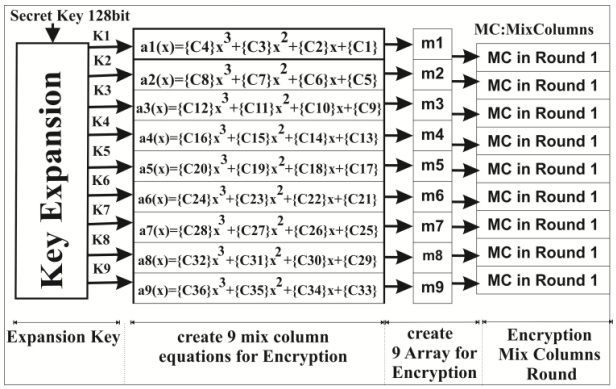

Fig 4: forming arrays and assigning them to MixColumns(i) in rounds of the encryption process

-step4: for the InvMixColumns(i) transformation in the decryption process inverse of nine arrays [m1, m2, m3, m4, m5, m6, m7, m8, m9] they are found to become $[(m1)^{-1}, (m2)^{-1}, (m3)^{-1}, (m4)^{-1}, (m5)^{-1}, (m6)^{-1}, (m7)^{-1}, (m8)^{-1}, (m9)^{-1}]$, where they are InvMixColumns(i) arrays in decryption process. The inverse of each matrix m(i) is calculated using the mathematical algorithm to find the inverse of a square matrix, noting that multiplication, division, and addition during finding the inverse of matrix are done over the finite field $GF(2^8)$ with modulo $(x^8+x^4+x^3+x+1)$. Firstly, inverse of matrix m1 is found to become $(m1)^{-1}$ for InvMixColumns(9) in round (9) of the decryption process, and so on for rest of eight arrays. Next figure (5) below shows the arrangement of inverted arrays $(mi)^{-1}$ and their distribution to InvMixColumns(i) transformation where it is existed in first nine rounds in the decryption process:

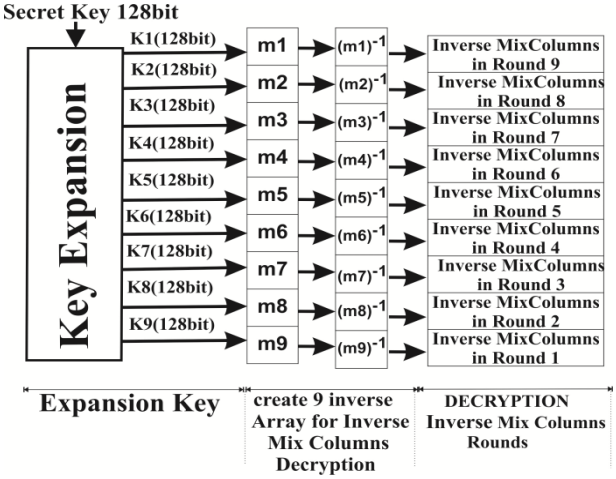

Fig 5: arrangement of inverse arrays $(mi)^{-1}$ and assigning them to InvMixColumns(i) in round(i)

-step5: $[(m1)^{-1}, (m2)^{-1}, (m3)^{-1}, (m4)^{-1}, (m5)^{-1}, (m6)^{-1}, (m7)^{-1}, (m8)^{-1}, (m9)^{-1}]$ arrays are used to conclude polynomials [b1(x), b2(x) ….., b9(x)] for InvMixColumns() in Decryption process, where (bi(x) * ai(x) mod $(x^4 + 1) = 1$) for $1 \le i \le 9$, so bi(x) is invers of ai(x) according to the modulo $(x^4 + 1)$.

$$(mi)^{-1} = \begin{bmatrix} bi3 & bi0 & bi1 & bi2 \\ bi2 & bi3 & bi0 & bi1 \\ bi1 & bi2 & bi3 & bi0 \\ bi0 & bi1 & bi2 & bi3 \end{bmatrix} \rightarrow bi(x)=\{bi0\}x3+\{bi1\}x2+\{bi2\}x+\{bi3\}$$

Fig 6: concluding polynomial bi(x) from array (mi) $^{-1}$

5. VARIBLE SHIFTROWSCOLUMNS(i) WITH EXTENDED SECRET KEY

5.1 ShiftRow() Transformation in AES

In encryption process, the input of ShiftRow() transformation is the State array (4 * 4bytes). The last three rows of the State array are circularly shifted to left, where in the first row there is no offset, but in the second row there is a rotatory offset to left by one, and in the third row there is a rotatory offset to left by two, finally in the fourth row there is a rotatory offset to left by three [11]. The following figure(7) illustrates the ShiftRow() transformation:

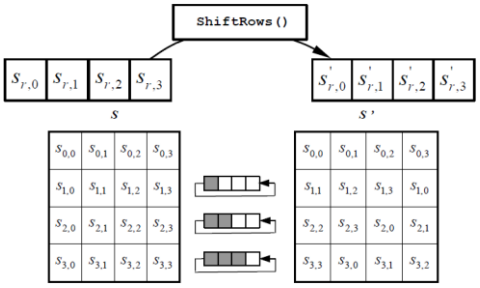

Fig 7: ShiftRow() in the original AES [11]

In the decryption process, InvShiftRow() is similar to the ShiftRow () transformation with one difference that the rotary shift is from left to right.

5.2 Proposed Method the Transformation ShiftRowColumns(i)

ShiftRow() is exists in all rounds of the encryption process, note the figure (1). it will be replaced with the proposed ShiftRowCloumns(i) transformation, that has a greater complexity (rows and columns) than ShiftRow()(rows only), and also has a dynamic variable pattern of rotatory offset according to extended secret key ki at each round(i) of AES in contrast to ShiftRow() that is static pattern in all rounds. This dynamic offset with the secret key, more complexity of shift (rows shift and columns shift) , and different pattern at each round(i) gives more confusion and diffusion to the encrypted text, a greater difficulty for cryptanalysis, and attacks and a higher security. The input of this transformation is the state array (4 * 4bytes). Initially, based on the extended secret key (ki) that dedicated to each Round (i), eight constants are calculated for each round(i). [ShiftColumn(i,1), ShiftColumn(i,2), ShiftColumn (i,3), ShiftColumn(i,4)] are four constants responsible of a rotatory offsetting of columns in State array in round(i), as the first parameter of these previous four constants indicates to the round number that belong to, and the second parameter indicates to the column number of State array that will be vertically shifted, ShiftColumn(Round Number, Column Number). While, [ShiftRow (i, 1), ShiftRow (i, 2), ShiftRow (i,3), ShiftRow (i, 4)] are the second four constants responsible for a rotatory offsetting of rows in the State matrix in round(i) ShiftRow(Round Number, Row Number), and each of the preceding eight constants take a value in rang [0 3]. We will calculate 88 constants for 10 rounds, which are the same in the encryption and decryption stages. Noting that each extended key ki (16bytes) is stored in a one-dimensional matrix Ki [1 * 16bytes], we can write the equation that represent compute of eight constants for each round(i) where $1 \leq i \leq 10$ as follows:

ShiftColumn(i,1)=(ki[0]Xorki[1])mod4
ShiftColumn(i,2)=(ki[2]Xorki[3])mod4
ShiftColumn(i,3)=(ki[4]Xorki[5])mod4
ShiftColumn(i,4)=(ki[6]Xorki[7])mod4
ShiftRow(i,1)=(ki[8]Xorki[9])mod4
ShiftRow(i,2)=(ki[10]Xorki[11])mod4
ShiftRow(i,3)=(ki[12]Xorki[13])mod4

ShiftRow(i,4)=(ki[14]Xorki[15])mod4

Figure 8: shows the distribution of the row and column shift constants in the encryption process over the AES rounds:

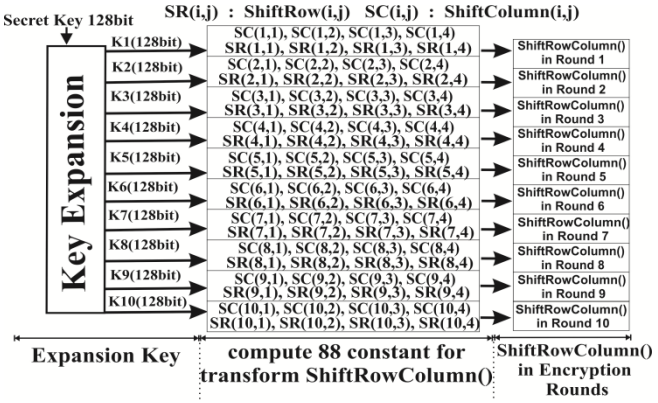

Fig 8: shift constants distribution on encryption rounds

After calculating shift constants, the ShiftRowCloumns(i) transformation in roud(i) operates as follow:

1. Firstly, it shifts State array vertically from top to down according to value of column shift constants that are assigned to round(i). The ShiftRowCloumns(i) circularly shifts the first column of State array in round(i) from top to down by ShiftCloumn(i,1) value, offsets the second column of State array in round(i) from top to down by ShiftCloumn(i,2) value, offsets the third column of State array in round(i) from top to down by ShiftCloumn (i,3) value, and offsets the fourth column of State array in round(i) from top to down by ShiftCloumn (i, 4) value.

2. Secondly, it shifts State array horizontally from right to left according to value of row shift constants that assigned to round(i). The ShiftRowCloumns(i) circularly shifts first row of State array in round(i) from right to left by ShiftRow(i,1) value, rotates second row of State array in round(i) from right to left by ShiftRow (i,2) value, rotates third row of State array in round(i) from right to left by ShiftRow(i, 3) value, and rotates fourth row of State array in round(i) from right to left by ShiftRow (i, 4) value.

Figure 9 below illustrates the method of calculating row and column shift constants in the round(i) from extended key Ki, and how ShiftRowCloumns(i) operates in encryption stage:

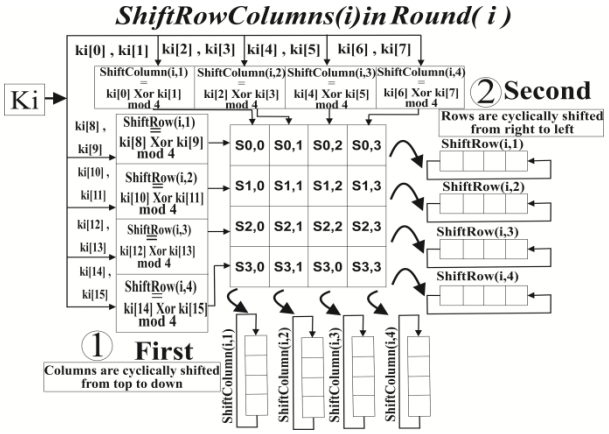

Fig 9: ShiftRowColumn() transformation in round(i)

5.3 InvShiftRowCloumns(i)

for decryption process, InvShiftRowColumn(i) works as similar as ShiftRowCloumns(i) with a difference that firstly, rows are cyclically shifted from left to right, and secondly the columns are cyclically shifted from down to top.

6. SIMULATION AND RESULTS

this section shows arrays that are generated by proposed algorithm with specified prime secret key. For a prime secret key 597c70a424a6e4ce12ae8496550a6e2b. Table (2) below illustrates generated arrays for MixColumn() and InvMixColumn() at each round(i) of AES in hexadecimal presentation:

Table 2. Generated Arrays at specified Secret Key

Prime Secret Key = 597c70a424a6e4ce12ae8496550a6e2b			
	Extended key	**MixColumn Array**	**InvMixColumn Array**
	k1= 597c70a424a6 e4ce12ae8496 550a6e2b	$\begin{bmatrix} f1 & 1a & ae & a8 \\ a8 & f1 & 1a & ae \\ ae & a8 & f1 & 1a \\ 1a & ae & a8 & f1 \end{bmatrix}$	$\begin{bmatrix} 4f & 12 & 89 & 84 \\ 84 & 4f & 12 & 89 \\ 89 & 84 & 4f & 12 \\ 12 & 89 & 84 & 4f \end{bmatrix}$
	k2= 3fe381581b45 659609ebe100 5ce18f2b	$\begin{bmatrix} 05 & 19 & 03 & ad \\ ad & 05 & 19 & 03 \\ 03 & ad & 05 & 19 \\ 19 & 03 & ad & 05 \end{bmatrix}$	$\begin{bmatrix} 6f & e8 & d0 & 48 \\ 48 & 6f & e8 & d0 \\ d0 & 48 & 6f & e8 \\ e8 & d0 & 48 & 6f \end{bmatrix}$
	k3= c5907012ded5 1584d73ef484 8bdf7baf	$\begin{bmatrix} 37 & 80 & 99 & 9a \\ 9a & 37 & 80 & 99 \\ 99 & 9a & 37 & 80 \\ 80 & 99 & 9a & 37 \end{bmatrix}$	$\begin{bmatrix} e4 & c5 & aa & 9a \\ 9a & e4 & c5 & aa \\ aa & 9a & e4 & c5 \\ c5 & aa & 9a & e4 \end{bmatrix}$
	k4= 5fb1092f8164 1cab565ae82f dd859380	$\begin{bmatrix} c8 & 4b & cb & 52 \\ 52 & c8 & 4b & cb \\ cb & 52 & c8 & 4b \\ 4b & cb & 52 & c8 \end{bmatrix}$	$\begin{bmatrix} 19 & 22 & 20 & e6 \\ e6 & 19 & 22 & 20 \\ 20 & e6 & 19 & 22 \\ 22 & 20 & e6 & 19 \end{bmatrix}$
	k5= c06dc4ee4109 d8451753306 acad6a3ea	$\begin{bmatrix} 87 & 55 & 1e & d5 \\ d5 & 87 & 55 & 1e \\ 1e & d5 & 87 & 55 \\ 55 & 1e & d5 & 87 \end{bmatrix}$	$\begin{bmatrix} 6e & 42 & e1 & f2 \\ f2 & 6e & 42 & e1 \\ e1 & f2 & 6e & 42 \\ 42 & e1 & f2 & 6e \end{bmatrix}$
	k6= 2667439a676e 9bdf703dabb5 baeb085f	$\begin{bmatrix} 98 & 06 & 53 & 4d \\ 4d & 98 & 06 & 53 \\ 53 & 4d & 98 & 06 \\ 06 & 53 & 4d & 98 \end{bmatrix}$	$\begin{bmatrix} f9 & cb & 98 & 29 \\ 29 & f9 & cb & 98 \\ 98 & 29 & f9 & cb \\ cb & 98 & 29 & f9 \end{bmatrix}$
	k7= ef578c6e8839 17b1f804bc04 42efb45b	$\begin{bmatrix} 5a & 42 & 44 & 17 \\ 17 & 5a & 42 & 44 \\ 44 & 17 & 5a & 42 \\ 42 & 44 & 17 & 5a \end{bmatrix}$	$\begin{bmatrix} 43 & 0c & 0c & 50 \\ 50 & 43 & 0c & 0c \\ 0c & 50 & 43 & 0c \\ 0c & 0c & 50 & 43 \end{bmatrix}$
	k8= 70dab542f8e3 a2f300e71ef7 4208aaac	$\begin{bmatrix} 5d & 4c & 0e & 4a \\ 4a & 5d & 4c & 0e \\ 0e & 4a & 5d & 4c \\ 4c & 0e & 4a & 5d \end{bmatrix}$	$\begin{bmatrix} f1 & 50 & c6 & 43 \\ 43 & f1 & 50 & c6 \\ c6 & 43 & f1 & 50 \\ 50 & c6 & 43 & f1 \end{bmatrix}$
	k9 = c076246e3895 869d3872986 a7a7a32c6	$\begin{bmatrix} fc & f4 & b8 & b6 \\ b6 & fc & f4 & b8 \\ b8 & b6 & fc & f4 \\ f4 & b8 & b6 & fc \end{bmatrix}$	$\begin{bmatrix} 97 & 2e & 92 & 50 \\ 50 & 97 & 2e & 92 \\ 92 & 50 & 97 & 2e \\ 2e & 92 & 50 & 97 \end{bmatrix}$
	k10 = 015590b439c 0162901b28e 437bc8bc85	$\begin{bmatrix} 70 & 8a & 7e & c6 \\ c6 & 70 & 8a & 7e \\ 7e & c6 & 70 & 8a \\ 8a & 7e & c6 & 70 \end{bmatrix}$	$\begin{bmatrix} 44 & 9a & 2e & c7 \\ c7 & 44 & 9a & 2e \\ 2e & c7 & 44 & 9a \\ 9a & 2e & c7 & 44 \end{bmatrix}$

Note that multiply MixColumn array by InvMixcolumn array over GF(2^8) results identity matrix() (the n × n square matrix with ones on the main diagonal and zeros elsewhere). Next table 3 demonstrates the output of ShiftRowColumn(1) in round(1) and shift constants (rows and columns) that are assigned to round(1) in encryption stage, and for expanded round key k1= c076246e3895869d3872986a7a7a32c5 :

Table 3. output ShiftRowColumn()

Round 1	**Input array**	**Extended K1**
proposed State Array	$\begin{bmatrix} 0e & 7c & 33 & 02 \\ f3 & de & 6b & a6 \\ 90 & 21 & c0 & 76 \\ 67 & 03 & 14 & 55 \end{bmatrix}$	k1= c076246e3895869d3872986a7a7a32c5
After Columns Shift ShiftColumn(1,1)=02 ShiftColumn(1,2)=02 ShiftColumn(1,3)=01 ShiftColumn(1,4)=03	$\begin{bmatrix} 90 & 21 & 14 & a6 \\ 67 & 03 & 33 & 76 \\ 0e & 7c & 6b & 55 \\ f3 & de & c0 & 02 \end{bmatrix}$	
After Rows Shift ShiftRow(1,1)=02 ShiftRow(1,1)=02 ShiftRow(1,1)=00 ShiftRow(1,1)=03	$\begin{bmatrix} 14 & a6 & 90 & 21 \\ 33 & 76 & 67 & 03 \\ 0e & 7c & 6b & 55 \\ 02 & f3 & de & c0 \end{bmatrix}$	

6.1 Basic Statistical Tests

After modifying the original AES, the statistical characteristics of the cipher text will be tested which is output of enhancement AES, through using the four basic statistical tests that developed by the National Institute of Standards and Technology (NIST) which are Frequency Test**,** Run Test, Serial Test, and Frequency Test within a Block. Cipher text passes the test if p-value is greater than 0.01 [29].

Table 4. Statistical tests for the improved algorithm

Plain Text	Secret Key	output of improved AES	Test name	p-value	
111111 111111 111111 111111 111111 11	1bb3a4 0129df 93c5dc 0a1c89 860cec a8	e16316 a1940b f7d371 bff25f 0c0528 e9	Frequency Test	0.86	pass
			Run Test	0.38	pass
			Serial Test	0.03	pass
			Frequency Test within a Block	0.22	pass
1095cb c4d956 8b2cd1 faf9f2 9bb79e aa	8bc157 783196 9a89dd 39e168 c8518d 38	355a9c 37c2a9 78f443 44f65e 6a8499 30	Frequency Test	0.59	pass
			Run Test	0.98	pass
			Serial Test	0.03	pass
			Frequency Test within a Block	0.10	pass
b1481e ea2918 b4891a 76d42a 9ddf4f 54	597c70 a424a6 e4ce12 ae8496 550a6e 2b	c44622 d4bdca 952100 3031f3 13098a d7	Frequency Test	0.05	pass
			Run Test	0.98	pass
			Serial Test	0.03	pass
			Frequency Test within a Block	0.31	pass
7d3bfb 0d6106 cf94dd faf9db d991e0 c3	9c75d2 084382 26acfb 585051 48fbe7 51	f12eb8 a6a146 78c5b3 7d8e35 ee1c98 13	Frequency Test	0.86	pass
			Run Test	0.48	pass
			Serial Test	0.03	pass
			Frequency Test within a Block	0.53	pass

after noting previous table 4, we conclude that the cipher text (out of enhancement AES) passed all test for four random different of plain text and secret key.

6.2 Avalanche Test [30]

Avalanche effect is very important in cryptographic algorithms. This means a single bit change in the secret key or in plain text it should give a significant change in cipher text at least half of the output. This criterion has the following formula:

$$\text{Avalanche Effect} = \frac{\text{number of flibed text in cipher text}}{\text{number of bits in cipher text}}$$

Table 5. Avalanche effect comparison between original AES and improved AES after changing in secret key

Plain text	Secret key	avalanche effect	
		Improved AES	Original AES
7d3bfb0d6106 cf94ddfaf9db d991e0c3	649c68ed14fb5dbb e**a**37ce114993996f	**0.63**	0.54
	649c68ed14fb5dbb e**b**37ce114993996f		
3b641a6b2f83 5db6913a0af1 2a47424e	**5**97c70a424a6e4ce 12ae8496550a6e2b	0.589	0.539
	797c70a424a6e4ce 12ae8496550a6e2b		
25eb122e82e6 dc4de0779f8d d5ee6e21	**C**d6f6e4ae3531e7f c0aa4fc33b356674	0.563	0.476
	dd6f6e4ae3531e7f c0aa4fc33b356674		
565556789876 78909878d5f5 5fda4567	Cd6f6e4ae3531e7fc0 aa4fc33b356674	0.54	0.45
	dd6f6e4ae3531e7fc0 aa4fc33b356674		
264d23be0b98 f55ec414deee a436b76b	5628d8c8d8c8a88f77 e7fa5f3687caf2	0.52	0.4453
	5628d8c8d8c8a88f77 f7fa5f3687caf2		
649a82dc93ca 06827d4e93ea cd9361fa	649c68ed14fb5dbb e**a**37ce114993996f	0.5	0.5
	649c68ed14fb5dbb e**b**37ce114993996f		
649a82dc93ca 06827d4e93ea cd9361fa	c4442cc89162b813 bdd98c245ae9e17**e**	**0.60**	0.51
	c4442cc89162b813 bdd98c245ae9e17**f**		
565556789876 78909878d5f5 5fda4567	1bb3a40129df93c5 dc0a1c89860ceca**8**	0.524	0.50
	1bb3a40129df93c5 dc0a1c89860ceca**9**		
3b641a6b2f83 5db6913a0af1 2a47424e	c4442cc89162b813 bdd98c245ae9e17**e**	0.555	0.476
	c4442cc89162b813 bdd98c245ae9e17**f**		
565556789876	8b**c**1577831969a89	0.532	0.523

78909878d5f5 5fda4567	dd39e168c8518d38		
	8b**d**1577831969a89 dd39e168c8518d38		
565556789876 78909878d5f5 5fda4567	1bb3a40129df93c5 dc0a1c89860ceca**8**	0.524	0.50
	1bb3a40129df93c5 dc0a1c89860ceca**9**		

Previous table 5 shows a comparison between AES and Improved AES after changing a single bit of secret key, and next finger10 demonstrate a comparison in table5:

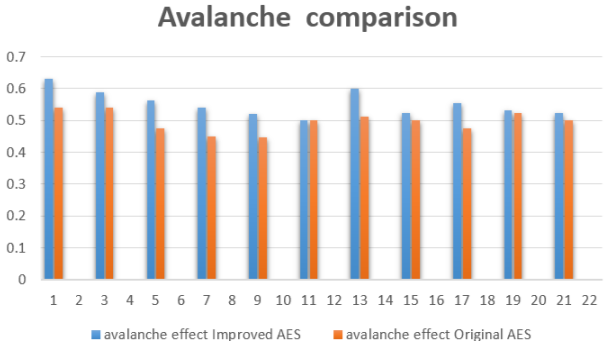

Fig 10: Avalanche effect comparison

6.3 Cryptanalytic Attack

This attack depends on nature and behavior of the encryption algorithm, and attempts to use characteristics of encryption algorithm to extract the plain text or the secret key from the cipher text [19]. this paper focused on developing the most important transformation in the AES algorithm which is MixColumn() transformation. This transformation is heart of the AES algorithm and confusion and diffusion center. After previous improvements, the static matrix of MixColumn() was removed which is known by attackers and analysts, and replaced by variable dynamic matrices with the secret key that are unknown to attackers and analysts. Also, more complexity to ShiftRows() transformation was added to become ShiftRowColumn(), that increased the diffusion on the bytes of cipher text. Moreover, the fixed pattern of cyclical shift was eliminated which is known to attackers, and replaced by variable dynamic pattern with secret key that is unknown to attackers and analysts. Therefore after these improvements, this work increased the diffusion inside the byte of the cipher text, the diffusion between the bytes of the cipher text, and the confusion by suggesting the proposed dynamic pattern with the secret key.

6.4 Execution Time

After adding previous enhancements to the AES 128-bit algorithm, we attempted to reduce the time of the optimized algorithm as much as possible by calculating MixColumn() and InvMixColumn() matrices, and shift constants in the initialization stage and isolating this stage from the functions that are used in the encryption and decryption process. The following table 6 show a comparison between the execution time of the original 128-bit algorithm and the improved 128-bit algorithm, as the execution time is calculated at specified data volume.

Table 6. Execution time comparison between original AES and improved AES

Data size	Original AES	Improved AES
1Kbyte	3.5ms	3.72ms
2Kbyte	7.16ms	7.72ms
3Kbyte	11ms	11.86ms
5Kbyte	17ms	20.59ms
25Kbyte	75ms	102.86ms
50Kbyte	148ms	205ms
100Kbyte	298ms	434ms
200kbyte	594ms	860ms
300kbyte	940ms	1270ms

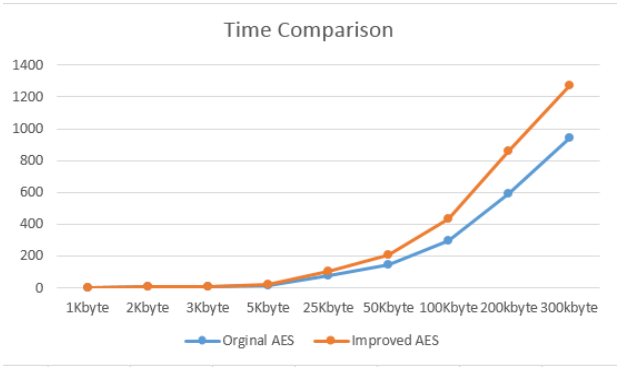

Fig 11: execution time comparison

7. CONCLUSION

This paper assumes a novel approach to improve the security of AES by increasing confusion and diffusion within encrypted text bytes (bits) and amongst encrypted text bytes of AES algorithm. It will be done by modifying the most important transformation in the AES algorithm which is MixColumn(), and increasing the complexity of ShitRow transformation. This work also increased the confusion process by suggesting an unknown dynamic pattern of the MixColumn(i) and ShiftRowColumn(i) transformations, this pattern changes with the secret key. Then, the basic statistical tests were applied on output of improved algorithm (cipher text), and tested the avalanche property, and then compared the additional time cost of this improvement with the original AES algorithm.

8. REFERENCES

[1] Hameed, E.M., Ibrahim, M. M., and Manap, A.N., 2018. Review on Improvement of Advanced Encryption Standard (AES) Algorithm based on Time Execution, Differential Cryptanalysis and Level of Security.

[2] Dewangan, P.C., Agrawal, S.,2012. A Novel Approach to Improve Avalanche Effect of AES Algorithm.

[3] Karthigaikumar, P., and Rasheed, S., 2011. Simulation of Image Encryption using AES Algorithm.

[4] Maolood, T.A.,and Yasser A. Y., 2017. Modifying Advanced Encryption Standard (AES) Algorithm.

[5] Jothy, A.K., Sivakumar, K., and Delsey, J. M., 2017. Efficient Cloud Computing with Secure Data Storage

Using AES and PGP Algorithm.

[6] Choudhury, P. K., and Kakoty S.,2017. Comparative Analysis of Different Modified Advanced Encryption Standard Algorithms over Conventional Advanced Encryption Standard Algorithm.

[7] Shekhar, S., Singh, P.,and Jaiswal,M., 2016. An Enhanced AES Algorithm Based on Variable Sbox and 200 Bit Data Block.

[8] Pahal, R., and kumar, V., 2013. Efficient Implementation of AES.

[9] Gul, F., Amin, A., and Ashraf, S., 2017. Enhancement of Cloud Computing Security with Secure Data Storage using AES.

[10] Yan, J.,and Chen,F., 2016.An Improved AES Key Expansion Algorithm". International Conference on Electrical.

[11] Federal Information Processing Standards Publication 197,2001. Announcing the ADVANCED ENCRYPTION STANDARD.

[12] Vaidehi, M., and Rabi, J. B., 2015. Enhanced MixColumn Design for AES Encryption.

[13] Arrag, S., Hamdoun, A., Tragha, A., and Khamlich, E.S.,2013. Design and Implementation a Different Architectures of Mixcolumn in FPGA.

[14] Abdulgader, A., Ismail, M., Zainal, N., and Idbeaa, T., 2015. Enhancement of AES algorithm based on chaotic maps and shift operation for image encryption.

[15] Abaas, and A. S.,and Shibeeb. K. A., 2015. A New Approach for Video Encryption Based on Modified AES Algorithm.

[16] Murtaza, G., Khan, A. A., Alam, W. S., and Farooqi, A.,2010. Fortification of AES with Dynamic Mix-Column Transformation.

[17] Kumar, P., and Rana, B. S., 2016. Development of modified AES algorithm for data security.

[18] Hameed, S., Riaz, F., Moghal, R., Akhtar, G., Ahmed, A., and Dar G. A., 2011. Modified Advanced Encryption Standard for Text and Images.

[19] Kawle, P., Hiwase, A., Bagde, G., Tekam, E., and Kalbande, R., 2014. Modified Advanced Encryption Standard.

[20] Wadi, M. S.,and Zainal, N., 2014. High Definition Image Encryption Algorithm Based on AES Modification.

[21] Wenceslao, V. F, 2018.Enhancing the Performance of the Advanced Encryption (AES) Algorithm Using Multiple Substitution Boxes. Standard.

[22] Hernandez, J. O., etc. al, 2008. A Low Cost Advanced Encryption Standard (AES) Co-Processor Implementation.

[23] Kohli, R., etc. al, 2012. S-Box Design Analysis and Parameter Variation in AES Algorithm.

[24] Stallings, W.,2011 FIFTH EDITION, CRYPTOGRAPHY AND NETWORK SECURITY.

[25] Benvenuto, J. C., 2012.Galois Field in Cryptography.

[26] Leith, V., 2010.The Rijndael Block Cipher.

[27] Anand, K., Sekar, A. C., Nagappan, G.,2017. Enhanced AES Algorithm using 512 Bit Key Implementation.

[28] Kak A., 2019. Finite Fields of the Form GF(2^n) Theoretical Underpinnings of Modern Cryptography Lecture Notes on "Computer and Network Security".

[29] Rukhin, A., Soto, J., Nechvatal, J., Smid, M., and Barker, E., 2012." A Statistical Test Suite for Random andPseudorandom Number Generators for Cryptographic Applications".

[30] Mandal, K. A., and Tiwari, A., 2012. Analysis of Avalanche Effect in Plaintext of DES using Binary Codes.