RESEARCH ARTICLE

Enhancing Security of AES Block Cipher Via Key-Dependent Dynamic XOR Tables

Tran Thi Luong, Nguyen Ngoc Cuong, Nguyen Thi Thu Giang, Truong Minh Phuong, Hoang Dinh Linh, Hoang Duc Tho

Luong TT, Cuong NN, Giang TTN, et al. Enhancing Security of AES Block Cipher Via Key-Dependent Dynamic XOR Tables. J Pure Appl Math. 2022; 6(5):25-31.

ABSTRACT

Block ciphers are an important area of modern cryptography and are widely used in many fields today. Studying to improve the security of block ciphers in general and for AES in particular continues to be carried out, in which there have been many works in literature on methods of making this block cipher dynamic. Dynamic methods can focus on components of the AES block cipher such as substitution layer or diffusion layer or both. In this paper, we propose a method to make the AES dynamic based on key addition transformation (Addroundkey) through key-dependent dynamic XOR tables, and concurrently prove that the new XOR operation is capable of preserving independently distributed and uniform probability properties of the random key in the ciphertext. The proposed method can help improve the security of AES against many strong attacks on the current block ciphers.

Keywords: *Block cipher; AES; Dynamic XOR table*

INTRODUCTION

Most modern block ciphers are usually designed based on iterative ciphers in which simple transformations are performed repeatedly to ensure diffusion and confusion according to the principle of C. Shannon [1]. Normally confusion is ensured by nonlinear mappings represented by S-boxes, while diffusion is ensured by a linear layer. In classical ciphers, these two properties are achieved using substitution or permutation ciphers. In modern cryptography, the use of substitution and permutation is in the form of Permutation-Substitution Networks (SPNs). SPN block ciphers consist of several rounds, each round generally consisting of three layers: substitution, permutation, and key addition, where the key addition layer adds round subkeys to the input of the substitution layer[2, 3].

The AES algorithm follows the structure of an SPN block cipher, therefore it consists of three layers: the substitution layer is performed by the substitution boxes (S boxes), the diffusion layer consists of two transformations including ShiftRow and Mix column, and the key addition layer is done through the Addroundkey transformation. To improve the security of the AES block cipher, there have been many studies towards making this block cipher dynamic. Many studies are directed towards making dynamically the substitution layer of the AES as in [1-13]. Some works focus on making the AES dynamic at the diffusion layer such as [3,4, 9, 14]. Some other works studied to make dynamically both substitution and diffusion layers of AES such as [3, 15, 16].

Recently, Salih et al. introduced a method to improve the security of AES by replacing AES's predefined XOR operation with a private XOR table for every round of AES [17]. This private XOR table depends on the 3D chaotic mapping. This 3D chaotic mapping is used as secret keys and the private XOR table is based on 4 bits. Salih et al. continued to propose a method to increase the security of AES by two dynamic double XOR tables using a 3D chaotic mapping. Dual dynamic XOR tables are based on 4 bits, one of the XOR tables is used for even rounds, and the other is used for odd rounds[18]. In addition, instead of the original MixColumn transformation in AES, the authors used the dynamic MixColumns transformation, which uses a private dynamic MDS matrix with a 3D chaotic mapping. 3D chaotic mapping is used to generate secret keys. The authors performed Diehard and NIST tests, entropy, correlation coefficient, and histogram to analyze the security of the proposed algorithms.

Through studying the proposed methods of we conclude that these are the methods to make the AES dynamic at the key addition layer, and they are very interesting, especially they can increase the security

Academy of Cryptography Techniques, No.141 Chien Thang road, Tan Trieu, Thanh Tri, Hanoi, Vietnam

Correspondence: Tran Thi Luong, Academy of Cryptography Techniques, No.141 Chien Thang road, Tan Trieu, Thanh Tri, Hanoi, Vietnam , e-mail luongtran@actvn.edu.vn

Received: Sep 29, 2022, Manuscript No. puljpam- 22-5427, Editor Assigned: Sep 30, 2022, PreQC No. puljpam- 22-5427 (PQ), Reviewed: Oct 1, 2022, QC No. puljpam-22-5427 (Q), Revised: Oct 1, 2022, Manuscript No puljpam- 22-5427 (R), Published: Oct 10, 2022, DOI:-10.37532/2752-8081.22.6(5).25-31.

of AES. However, the methods proposed still have some inaccurate and unreasonable problems, we will analyze these issues in detail in Section 2[17, 18].

In this paper, we will propose a method to make AES dynamic at the key addition layer via key-dependent dynamic XOR tables. An algorithm for generating these XOR tables is introduced. We also prove that the new XOR operation is able to preserve independently distributed and uniform probability properties of the random key in the ciphertext. We also perform a security analysis of the modified AES block cipher when using this dynamic method.

The paper is organized as follows. In section 2, we give some comments on the results [17, 18]. Section 3 proposes an algorithm to generate new XOR tables, and concurrently proves that the new XOR operation is capable of preserving independently distributed and uniform probability properties of the random key in the ciphertext. Section 4 gives a security analysis of modified AES block cipher. Section 5 is the conclusion.

Remarks on the results

The authors suggest that new XOR tables are used to replace the original XOR table used in AES. However, there are still many inconsistencies in these two papers [17, 18]. We give a few general comments as follows:

The authors point out three properties of the new XOR table are:

- The sum of each row or column is 120.
- Each column or row consists of numbers (0 to 15) without repetition.
- The XOR table is symmetric through the main diagonal.

Actually, the first property is inferred from the second property. Furthermore, the authors have not shown that these properties guarantee correct decryption, i.e. if $a\ XOR\ b\ =\ c$ then the inference of $a\ =\ c\ XOR\ b$ and $b\ =\ c\ XOR\ a$ a is not pointed out, although the specific XOR tables created by the authors satisfy this property. Algorithm (1) does not make this clear, nor does it prove to be sufficient to generate XOR tables in [17].

The way to generate secret numeric keys from sequences generated from 3D Logistic and Chebyshev mappings of is not really efficient [17, 18].

The array Z is generated to create a key matrix on which to generate a dynamic MDS matrix from AES's Mixcolumn matrix. However, it created a new matrix that is not MDS. Specifically, the dynamic MDS matrix has a singular square submatrix of size 2 (Figure 1)[18].

Orignial MDS	Secret Keys	Dynamic MDS
2 3 1 1	3 2 0 1	1 1 2 3
1 2 3 1	2 3 1 0	3 1 2 1
1 1 2 3	1 0 3 2	1 1 3 2
3 1 1 2	1 0 2 3	1 3 1 2

Figure 1) *The dynamic MDS matrix generated by [18]*

Proposing an algorithm for creating key-dependent dynamic XOR tables

Necessary properties for an XOR table

Since the algorithm, we are considering is AES with transformations performed over the field $GF(2^8)$, each element over $GF(2^8)$is treated as a byte, each byte consisting of two halves, and each half consists of 4 bits. Therefore, to ensure that the encryption and decryption process is done correctly, we develop the following properties that an XOR table needs to satisfy:

Each column and each row of the XOR table contains the numbers from 0 to 15 and not repeat (i.e. a permutation of the numbers 0 to 15).

The XOR table is symmetric through the main diagonal (this property will infer $a\ XOR\ b = b\ XOR\ a$).

For every element a, b, c in the XOR table such that $a\ XOR\ b = c$, the following conditions must be satisfied simultaneously: $b\ XOR\ c = a$, $a\ XOR\ c = b$.

An algorithm for creating a new XOR table

First, we present the following algorithm for finding random permutations of n given elements.

Fisher–Yates shuffle algorithm

Assuming that, there is an input array of n given elements, the output of Fisher-Yates's shuffle algorithm is a random permutation of the n elements of the array. It is also assumed that there is a function r and () to generate a random number between 0 and $n-1$. The algorithm is implemented as follows:

- Start from the last element of the array and swap it with a randomly selected element from the entire array (including the last element).
- Now consider the array from 0 to $n-2$ (size is reduced by 1): similarly, select a random element in the array (from 0 to $n-2$) and then swap it with the $(n-2)^{th}$ element.
- Continue with the array from 0 to $n-3$ and repeat this process till we hit the first element.

This algorithm has a time complexity of $O(n)$and generate a random permutation.

Next, we propose Algorithm 1 to generate a new XOR table depending on a secret key which uses the above Fisher-Yates algorithm. The key is pre-shared by the sender and receiver.

Assume that the sender and receiver must first share a secret key X^*.

Algorithm 1 (Generate a new XOR table)

Input: A secret key X^*, the original XOR Table used in AES (Table 1); A pseudo-random number generator G; an identical permutation $P = \{0,1,\ldots,15\}$

Output: A new XOR table A.

Step 1: Use the pseudo-random number generator G with the key seed X^* to generate an output that is a pseudo-random bit sequence. Trimming each of the 4 bits of this output sequence and converting them to decimal form, obtains the sequence $\mathcal{E}$ consisting of the elements in [15]. Denote the sequence $\mathcal{E}$ as follows:

$$\mathcal{E} = \{\mathcal{E}_o, \mathcal{E}_1 \ldots, \mathcal{E}_\eta, \ldots\},$$

Step 2: Taking the sequence $\mathcal{E}$ obtained from Step 1 as input to the Fisher-Yates shuffle algorithm with the permutation P, we obtain a

random permutation consisting of 16 distinct elements belonging to [15]. We put this permutation into an array x, denote as follows:

$$x = [\chi_{O,}\chi_{1,}...,\chi_{15}]$$

Step 3: From the original AES XOR table (Table 1), we can create a new 16×16 XOR table A as follows:

- Replace all entries in the original XOR table with the corresponding elements in the array x, that is, replace the value i (0 ≤ I ≤ 15) in the original XOR table with the element x_i of the array x. Doing this replacement for the entire original XOR table, we get a new XOR table.
- Move the positions of rows and columns in the new XOR table so that the first row and the first column in this table is an ascending sequences from 0 to 15. Then, another new XOR table is obtained, denoted by A.

Note

- We should choose the pseudorandom number generator G as a cryptographically secure pseudo-random number generator [15, 16].
- When using the sequence E for the Fisher-Yates algorithm, the elements ε_i are considered as array indices in the Fisher-Yates algorithm, in the jth step needs a random number in the array from position 0 to $n-j$, if the element ε_i is greater than the array indices corresponding to the elements are already selected in the right of the array, ignore it and consider $\varepsilon_{i+1,}$ ...

Remark 1 (On the correctness of the new XOR table)

It can be seen that the original AES XOR table satisfies the properties of an XOR table proposed in Section 3.1. On the other hand, Algorithm 1 simply performs the replacement of the elements, and also performs the simultaneous replacement of all the elements of the table, so the new XOR table can preserve all three required properties of an XOR table. However, it should be noted that the XOR operation in the new XOR table will be the new XOR operation, unlike the regular bit Exclusive Or.

Interestingly, the above algorithms can generate new XOR tables that are dynamic XOR tables that depend on the original secret key.

Remark 2

In fact, X_1 consists of elements that are permutations of 16 elements from 0 to 15. And these permutations are generated from the original random key. Because there are 16! permutations of 16 elements from 0 to 15. Thus we can get 16! possible dynamic XOR tables through Algorithm 1.

Now, we denote the new XOR operation in the new XOR table as $\oplus$

Assume that $\varepsilon_{1,}\varepsilon_2,....,\varepsilon_\eta$ are independent, random variables that take the numeric values in the set $B=\{0,1,...,15\}$ with the same probability 16^{-1}. When $\varepsilon_{1,}\varepsilon_2,....,\varepsilon_\eta$ take a particular value $K=(k_1,k_2,...,k_\eta)$ then K is said to be an ideal random number key of length n.

Denote P_1 is the probability distribution over the plaintext space P, P_2 is the probability distribution over the ciphertext space C, and P_3 is the probability distribution over the key space K.

Suppose $\mathrm{P}=C=\mathrm{K}=B$

We have the following proposition:

Proposition 1

Suppose $x_1,x_2,...,x_{n,}y_{1,}y_2,...,y_n$ are plaintexts and ciphertexts that take values on the set B, respectively, and independent random variables $\varepsilon_{1,}\varepsilon_2,....,\varepsilon_\eta$ having the same uniform distribution over the set B, take the corresponding values $k_1,k_2,...,k_\eta$, where $y_{1,}y_2,...,y_n$ are the results of the new XOR operation:

$$y_i = x_i \oplus k_i, i=1,2,...,15.$$

Then it is to have: $P_2(y_{1,}y_2,...,y_n)=16^{-n}$

Proof

Indeed, because of the independently random variables with the same probability $\varepsilon_{1,}\varepsilon_2,....,\varepsilon_\eta$ take the corresponding values $k_1,k_2,...,k_\eta$, and because of property of the new XOR, it is to have: if $y_i = x_i \oplus k$ then $x_i = y_i \oplus k_i$, and if $y_j = x_j \oplus k_j$ then $x_j = y_j \hat{\oplus} k_j$, so:

$$P_2(y_{i,}y_j) = \sum_{k_i,k_j\in K} P_3(k_i,k_j)\, P_1(y_i \hat{\oplus} k_i, y_j \hat{\oplus} k_j)$$

$$= \sum_{k_i,k_j\in K} P_3(k_i)P_3(k_j)\, P_1(y_i \hat{\oplus} k_i, y_j \hat{\oplus} k_j)$$

$$= \tfrac{1}{16^2}\sum_{k_i,k_j\in K} P_1(y_i \hat{\oplus} k_i, y_j \hat{\oplus} k_j) = 16^{-2}. \quad (1)$$

On the other hand, for $y = x \hat{\oplus} kf$, and since by the property of the new XOR, if $y = x \hat{\oplus} k$ then $x = y \hat{\oplus} k$, so it is to have:

$$P_2(y) = \sum_{k\in B} P_3(k)P_1(x) = \sum_{k\in B} \frac{1}{16} P_1\left(y \hat{\oplus} k\right)$$

$$= \tfrac{1}{16}\sum_{k\in B} P_1\left(y \hat{\oplus} k\right) = \tfrac{1}{16} \quad (2)$$

By (1), (2), it is inferred that:

$$P_2(y_{i,}y_j) = P_2(y_i)\cdot P_2(y_j) = 16^{-2}. \quad (3)$$

With similar proof, it is to have:

$$P_2(y_1.y_2,..., y_n) = P_2(y_1) \cdot P_2(y_2) \cdot ... \cdot P_n(y_n) = 16^{-n}. \blacksquare$$

Remark 3

From Proposition 1, it can be seen that, when performing a new XOR operation between the plaintext and the random key, we obtain the ciphertext which is a new sequence that has the same properties as the random key (that is, with independent distribution and uniform probability). This is important, because the ciphertext after the XOR operation can lose the rules of the plaintext, the ciphertext has an independent distribution and uniform probability, making it difficult for cryptanalysis. On the other hand, this XOR operation is performed before the S box of AES, so the ability to preserve the properties of independent distribution and uniform probability is also intended to ensure that the definitions of linear and differential probabilities can be applied to the Sbox (Table 1).

TABLE 1
The original XOR table in AES

XOR	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	3	2	5	4	7	6	9	8	11	10	13	12	15	14
2	2	3	0	1	6	7	4	5	10	11	8	9	14	15	12	13
3	3	2	1	0	7	6	5	4	11	10	9	8	15	14	13	12
4	4	5	6	7	0	1	2	3	12	13	14	15	8	9	10	11
5	5	4	7	6	1	0	3	2	13	12	15	14	9	8	11	10
6	6	7	4	5	2	3	0	1	14	15	12	13	10	11	8	9
7	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
8	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
9	9	8	11	10	13	12	15	14	1	0	3	2	5	4	7	6
10	10	11	8	9	14	15	12	13	2	3	0	1	6	7	4	5
11	11	10	9	8	15	14	13	12	3	2	1	0	7	6	5	4
12	12	13	14	15	8	9	10	11	4	5	6	7	0	1	2	3
13	13	12	15	14	9	8	11	10	5	4	7	6	1	0	3	2
14	14	15	12	13	10	11	8	9	6	7	4	5	2	3	0	1
15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Compare our method

In the following, we will show some comparisons of methods creating dynamic XOR tables in with our method [17, 18].

- Gave three properties of the new XOR operation, but the authors have not shown that these properties guarantee correct decryption. In this paper, we have shown the necessary properties for an XOR table (section 3.1) and ensure that the decryption process is correct [17, 18].

- The authors create only one private XOR table, and in [18] the authors create two new XOR tables, but in this paper we can create 16! dynamic XOR tables depending on an original secret key. The large number of dynamic XOR tables makes it much more difficult for cryptanalysis, which in turn increases the security of the AES block cipher [17].

- The authors only showed how to create new XOR tables, but did not show theoretically the importance of the XOR as well as the new XOR. In this article, we have made that clear through Proposition 1 [17, 18].

Application of dynamic XOR tables to modify AES block cipher

Executing Algorithm 1 twice with two different secret keys x^* and y^* as inputs, we obtain two new key-dependent XOR tables, denoted A and B. We will use these tables alternately in rounds, specifically: XOR table A is used for even rounds of AES, XOR table B is used for odd rounds.

We conduct experiments with x^* and y^* as two 128-bit true random keys derived from the source of true random numbers (refer to the link: https://www.random.org/bytes/). In particular, two true random strings (in hexadecimal) are obtained as follows:

$$x^* = f2\ 65\ 9f\ \ 55\ 65\ 5f\ \ 5f\ \ 0e\ 86\ 84\ 33\ bd\ d5\ 06\ 6c\ e3$$

$$y^* = ee349c5ee7093e34f6612ba82f3c146d$$

Then, perform Algorithm 1 twice with these two keys, we get two arrays X_1 and Y_1 as follows:

$$X_1 = [3, 8, 0, 5, 11, 9, 13, 10, 2, 4, 1, 6, 12, 7, 15, 14]$$

$$Y_1 = [5, 11, 7, 13, 2, 10, 15, 1, 12, 6, 4, 9, 0, 3, 14, 8]$$

And obtain two new XOR tables A_1 and B_1 as shown in Tables 2 and 3.

TABLE 2
XOR table A_1

XOR	3	8	0	5	11	9	13	10	2	4	1	6	12	7	15	14
3	3	8	0	5	11	9	13	10	2	4	1	6	12	7	15	14
8	8	3	5	0	9	11	10	13	4	2	6	1	7	12	14	15
0	0	5	3	8	13	10	11	9	1	6	2	4	15	14	12	7
5	5	0	8	3	10	13	9	11	6	1	4	2	14	15	7	12
11	11	9	13	10	3	8	0	5	12	7	15	14	2	4	1	6
9	9	11	10	13	8	3	5	0	7	12	14	15	4	2	6	1
13	13	10	11	9	0	5	3	8	15	14	12	7	1	6	2	4
10	10	13	9	11	5	0	8	3	14	15	7	12	6	1	4	2
2	2	4	1	6	12	7	15	14	3	8	0	5	11	9	13	10
4	4	2	6	1	7	12	14	15	8	3	5	0	9	11	10	13
1	1	6	2	4	15	14	12	7	0	5	3	8	13	10	11	9
6	6	1	4	2	14	15	7	12	5	0	8	3	10	13	9	11
12	12	7	15	14	2	4	1	6	11	9	13	10	3	8	0	5
7	7	12	14	15	4	2	6	1	9	11	10	13	8	3	5	0
15	15	14	12	7	1	6	2	4	13	10	11	9	0	5	3	8

14	14	15	7	12	6	1	4	2	10	13	9	11	5	0	8	3

TABLE 3
XOR Table B_1

XOR	**5**	**11**	**7**	**13**	**2**	**10**	**15**	**1**	**12**	**6**	**4**	**9**	**0**	**3**	**14**	**8**
5	5	11	7	13	2	10	15	1	12	6	4	9	0	3	14	8
11	11	5	13	7	10	2	1	15	6	12	9	4	3	0	8	14
7	7	13	5	11	15	1	2	10	4	9	12	6	14	8	0	3
13	13	7	11	5	1	15	10	2	9	4	6	12	8	14	3	0
2	2	10	15	1	5	11	7	13	0	3	14	8	12	6	4	9
10	10	2	1	15	11	5	13	7	3	0	8	14	6	12	9	4
15	15	1	2	10	7	13	5	11	14	8	0	3	4	9	12	6
1	1	15	10	2	13	7	11	5	8	14	3	0	9	4	6	12
12	12	6	4	9	0	3	14	8	5	11	7	13	2	10	15	1
6	6	12	9	4	3	0	8	14	11	5	13	7	10	2	1	15
4	4	9	12	6	14	8	0	3	7	13	5	11	15	1	2	10
9	9	4	6	12	8	14	3	0	13	7	11	5	1	15	10	2
0	0	3	14	8	12	6	4	9	2	10	15	1	5	11	7	13
3	3	0	8	14	6	12	9	4	10	2	1	15	11	5	13	7
14	14	8	0	3	4	9	12	6	15	1	2	10	7	13	5	11
8	8	14	3	0	9	4	6	12	1	15	10	2	13	7	11	5

Next, move the positions of the rows and columns in the two tables A_1 and B_1 so that the first row and column in these two tables is an ascending sequence from 0 to 15, yielding two dynamic XOR tables A and B, respectively, as shown in Tables 4 and 5.

TABLE 4
XOR table *A* for even rounds

XOR	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	3	2	1	0	6	8	4	14	5	10	9	13	15	11	7	12
1	2	3	0	1	5	4	8	10	6	14	7	15	13	12	9	11
2	1	0	3	2	8	6	5	9	4	7	14	12	11	15	10	13
3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	6	5	8	4	3	1	0	11	2	12	15	7	9	14	13	10
5	8	4	6	5	1	3	2	15	0	13	11	10	14	9	12	7
6	4	8	5	6	0	2	3	13	1	15	12	14	10	7	11	9
7	14	10	9	7	11	15	13	3	12	2	1	4	8	6	0	5
8	5	6	4	8	2	0	1	12	3	11	13	9	7	10	15	14
9	10	14	7	9	12	13	15	2	11	3	0	8	4	5	1	6
10	9	7	14	10	15	11	12	1	13	0	3	5	6	8	2	4
11	13	15	12	11	7	10	14	4	9	8	5	3	2	0	6	1
12	15	13	11	12	9	14	10	8	7	4	6	2	3	1	5	0
13	11	12	15	13	14	9	7	6	10	5	8	0	1	3	4	2
14	7	9	10	14	13	12	11	0	15	1	2	6	5	4	3	8
15	12	11	13	15	10	7	9	5	14	6	4	1	0	2	8	3

TABLE 5
XOR table *B* for odd rounds

XOR	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	5	9	12	11	15	0	10	14	13	1	6	3	2	8	7	4
1	9	5	13	4	3	1	14	10	12	0	7	15	8	2	6	11
2	12	13	5	6	14	2	3	15	9	8	11	10	0	1	4	7
3	11	4	6	5	1	3	2	8	7	15	12	0	10	14	13	9
4	15	3	14	1	5	4	13	12	10	11	8	9	7	6	2	0
5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	10	14	3	2	13	6	5	9	15	7	0	12	11	4	1	8
7	14	10	15	8	12	7	9	5	3	6	1	13	4	11	0	2
8	13	12	9	7	10	8	15	3	5	2	4	14	1	0	11	6
9	1	0	8	15	11	9	7	6	2	5	14	4	13	12	10	3
10	6	7	11	12	8	10	0	1	4	14	5	2	3	15	9	13
11	3	15	10	0	9	11	12	13	14	4	2	5	6	7	8	1
12	2	8	0	10	7	12	11	4	1	13	3	6	5	9	15	14
13	8	2	1	14	6	13	4	11	0	12	15	7	9	5	3	10
14	7	6	4	13	2	14	1	0	11	10	9	8	15	3	5	12
15	4	11	7	9	0	15	8	2	6	3	13	1	14	10	12	5

Then, we use XOR table A for even AES rounds, XOR table B for odd rounds of AES to perform encryption and decryption [19-21]

Security analysis of the modified AES algorithm

Security analysis

Instead of using an original AES XOR table, in the modified AES algorithm we use two dynamic XOR tables generated from the secret keys and the original XOR table. In practice, the cryptanalyst may know the algorithm for generating the dynamic XOR tables (Algorithm 1), but not the secret keys. Furthermore, the modified AES algorithm uses two dynamic XOR tables alternately between rounds. This makes cryptanalysis even more difficult.

On the other hand, as the proof of Proposition 1, the new XOR operation makes the output guaranteed to have the same properties as a random key sequence (i.e. independent distribution, uniform probability). This is important because the ciphertext after the XOR operation can lose the rules of the plaintext, making it difficult for cryptanalysis. Moreover, this XOR operation is performed before the Sbox of AES, so the ability to preserve the properties of independent distribution and uniform probability is also intended to ensure that the definition of linear and differential probabilities can be applied to the S box.

Another aspect is that the number of dynamic XOR tables generated by Algorithm 1 is very large [16]. Therefore if the cryptanalyst

performs the exhaustive searching method, it is very difficult to do.

Evaluation of randomness through the NIST tests

Any cryptographic module must have the property that the redundancies at the input should not leak to the output. In the selection of the AES finalists, NIST used a method that concatenates the output strings of cryptographic primitives and then used the NIST SP 800-22 test suite to evaluate the randomness [22]. However, when making such an assessment it is necessary to treat the cryptographic primitives as a PRNG, and the results of the randomness assessment are then for the PRNG and not the cryptographic primitive in question itself.

In proposed to use a number of tests including SAC Test, Linear Span Test, Collision Test, and Coverage Test to evaluate some cryptographic criteria for block ciphers and hash function [23]. In Sulak proposed a method to evaluate the randomness of block ciphers and hash functions [24]. The main idea is to construct non-random input data sets (with redundancy) and compute the corresponding outputs using the cryptographic primitives to be evaluated for randomness. Then, the randomness of the output data sets is checked by statistical tests. If the data is random then obviously the randomness comes from the cryptographic primitive, otherwise, the cryptographic primitive has no randomness. Our goal is to evaluate the output randomness of the modified AES block cipher. In all of the tests, it is assumed that the algorithm subject to the test is a mapping $\mathbb{F}_2^m \times \mathbb{F}_2^n \to \mathbb{F}_2^n$, where m is the key size and n is the block size of the modified block cipher.

Statistical Tests: In this paper, we use two levels test approach in the NIST SP 800-22 test suite [15]. The p-value distribution of these tests for sequences of short length has been studied and shown in [25].

We have applied the evaluation to AES and the modified AES. We also consider two cases for selection the master key: zero key and random key. The obtained results show that the modified one achieve the output randomness equivalent to AES (Table 6).

TABLE 6.
Randomness evaluation results for AES-128 and modified AES block cipher

Algorithm	# rounds	#rounds random	Input data sets [24]
AES-128 with zero key	10	≥ 4 ≥ 4	LW, HW, Av1, Rot
Modified AES-128 with zero key	10	≥ 3	LW, HW, Av1, Rot
AES-128 with random key	10	≥ 3	LW, HW, Av1, Rot
Modified AES-128 with random key	10		LW, HW, Av1, Rot

CONCLUSION

To improve the security of SPN block ciphers in general and AES block cipher in particular, dynamic methods can focus on components of the ciphers such as the substitution layer or diffusion layer, or both. In this paper, we propose a method to make AES dynamic at the key addition layer through key-dependent dynamic XOR tables. Interestingly, the new XOR operation has been shown to be able to preserve the independent distribution and uniform probability of the random key in the ciphertext. Modified AES block ciphers are analyzed for security and evaluated for randomness using NIST statistical criterion. The proposed method can create a dynamic AES block cipher that enhances the security of AES against many strong attacks on the current block ciphers.

STATEMENTS AND DECLARATIONS

FUNDING: No funding was received for conducting this study.

FINANCIAL INTERESTS: The authors declare they have no financial interests.

COMPETING INTERESTS: The authors have no competing interests to declare that are relevant to the content of this article**.**
Research Data Policy and Data Availability Statements: Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

REFERENCES

1. Rohiem A, Diaa A, Mohammed F, et al. Generation of aes key dependent s-boxes using rc4 algorithm. In Int Conf Aerosp Sci Aviat Technol. 2009;13:1-09.
2. Agarwal P, Singh A, Kilicman A. Development of key-dependentdynam ic s-boxes with dynamic irreducible polynomial and affine constant. Av mech eng. 2018;10(7):1-18.
3. Ahmed F, Elkamchouchi D. Strongest aes with s-boxes bank and dynamic key mds matrix (sdk-aes). Int J Comput Commun Eng. 2013;2(4):530.
4. Al-Wattar A H, Mahmod R, Zukarnain Z A, et al. A new DNA based approach of generating key dependent Mixcolumns transformation. Int J Comput Netw Commun. 2018;7(2):93-102.
5. Arrag S, Hamdoun A, Tragha A, et al. Implementation of stronger AES by using dynamic s-box dependent of masterkey. J Theor Appl Inf Technol. 2013 ;53(2):1.
6. Assafli H T, Hashim I A. Generation and Evaluation of a New Time-Dependent Dynamic s-box Algorithm for AES Block Cipher Cryptosystems. 3rd Int Conf Recent Innov Eng Mater Sci Eng.2020;978(1):012042.
7. Daemen J, Rijmen V. The design of Rijndael: AES-the advanced encryption standard. N Y: Springer-verl.2002;2(1):1.
8. Hosseinkhani R, Javadi HHS. Using cipher key to generate dynamic s-box in aes cipher system. Int J Comput Sci Secur .2012;6(1):19-28.
9. Ismil I A, Galal – Edeen G H, Khattab S, et al. Performance examination of AES encrytion algorithm with constant and dynamic rotation. Int J Rev Comput. 2012; (12):1.
10. Juremi J, Mahmod R, Zukarnain Z A, et al. Modified AES s-box Based on Determinant Matrix Algorithm. Int J Adv Res Comput Sci Softw Eng. 2017;7(1):110-16.
11. Kazlauskas K, Kazlauskas J. Key-dependent s-box generation in aes block cipher system. INFORMATICA. 2009 ;20(1) :23-34.
12. Keliher L. Linear cryptanalysis of substitution-permutation networks. Queen's University, Kingston, Ontario, Canada. 2003.

13. Mahmoud E M, Abd El Hafez A, Elgarf T A, et al. Dynamic aes-128 with key-dependent s-box. Int J Eng Res Appl. 2013 ;3(1) :1162-70.

14. Murtaza G, Khan A A, Alam S W, et al. Fortification of aes with dynamic mix-column transformation. IACR Cryptol ePrint Arch. 2011;184(1):1-13.

15. Rukhin A, Soto J, Nechvatal J, et al. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. NIST Spec Publ.2010;800-22.

16. Saarinen Markku-Juhani O. SP 800-22 and GM/T 0005-2012 Tests: Clearly Obsolete, Possibly Harmful. Announc Propos Revise Spec Publ.2022;169 :1.

17. Salih A I, Alabaichi A, Abbas A S, et al. A novel approach for enhancing security of advance encryption standard using private Xor table and 3d chaotic regarding to software quality factor. ICIC Express Lett B: Appl.2019;10(9):823-32.

18. Salih A I, Alabaichi A, Tuama A Y. Enhancing advance encryption standard security based on dual dynamic XOR table and mixcolumns transformatio. Indones J Electr Eng Comput Sci.2020 ;19(3) :1574-81.

19. Shannon C E. Communication theory of secrecy systems. Bell Syst Tech J.1949 ;28(4) :656-715.

20. Yousif I A. Proposed A Permutation and Substitution Methods of Serpent Block Cipher. Ibn Al Haitham J Pure Appl Sci.2019 ;32(2) :131-44.

21. Youssef AM, Tavares S E, Heys H M. A new class of substitution permutation networks. In Proc Third Annu Workshop Sel Areas Cryptogr. 1996;132-147.

22. Soto J, Soto J. Randomness testing of the advanced encryption standard candidate algorithms . US Dep Commer Technol Adm Natl Inst Stand Technol.1999 :9

23. Doanaksoy A, Ege B, Koçak O, et al. Cryptographic randomness testing of block ciphers and hash functions. Cryptol ePrint Arch.2010 .

24. Sulak F. Statistical analysis of block ciphers and hash functions. 2011.

25. Fatih Sulak. Evaluation of randomness test results for short sequences. In International Conference on Sequences and Their Applications.2010 ;10(1) :309-19.