Strengthening AES Security Through Key-Dependent ShiftRow and AddRoundKey Transformations Utilizing Permutation

Tran Thi Luong
Academy of Cryptography Techniques, No. 141 Chien Thang Road, Tan Trieu, Thanh Tri, Hanoi, Vietnam

***Abstract*—AES (Advanced Encryption Standard) is a widely applied block cipher standard in the United States, used in various security applications today. Currently, there are numerous research endeavors aimed at making AES block ciphers dynamic to improve their security against contemporary strong attacks. The most common dynamic approach involves the dynamization of AES block transformations, including SubByte, ShiftRow, AddRoundKey, and MixColumn operations. The combination of these transformations has also been explored and proposed. However, to the best of our knowledge, the dynamic combination of AddRoundKey and ShiftRow transformations remains unexplored. Therefore, in this paper we introduce algorithms for generating key-dependent AddRoundKey and ShiftRow transformations based on permutations. Subsequently, these key-dependent transformations are applied to AES to create dynamic AES block ciphers. Security analysis and evaluation of NIST's statistical criteria are performed, and the entropy of AES and dynamic AES is assessed. From our findings, it is evident that dynamic AES block ciphers can significantly enhance AES security and meet stringent randomness criteria, similar to AES.**

***Keywords*—AES; ShiftRow; AddRoundKey; dynamic AES; key-dependent**

I. Introduction

Along with the strong development of information technology, security risks and attacks are increasing in complexity. Therefore, cryptographic primitives are being widely used in various security domains nowadays [1, 2, 3]. Substitution-Permutation Network (SPN) block ciphers [4, 5, 6] represent a prevalent category of block ciphers extensively applied in contemporary cryptographic scenarios. An SPN block cipher comprises three primary components: the substitution layer, which typically employs S-boxes [7–10]; the diffusion layer, commonly utilizing MDS matrices [11–14] (matrices derived from maximum distance separable codes); and the key addition layer.

AES [15, 16] belongs to the class of SPN block ciphers and serves as a block cipher standard established by NIST in 2001, originating in the United States. The AES round function incorporates three operations, namely key addition, substitution, and linear transformations. AES, one of the world's most widely used encryption algorithms, faces potential vulnerabilities that could be exploited by cryptanalysts. The simplicity of AES's mathematical structure and the threat of attacks such as algebraic attacks [17], linear attacks [8, 18], and differential attacks [18, 19], make its security a concern. Moreover, the advent of supercomputers and quantum computing poses further risks, necessitating increased key lengths for maintaining security. Therefore, researching various approaches to enhance the security strength of AES is crucial in the current scenario.

To enhance the resilience of block ciphers against modern, potent attacks, extensive research has been conducted to animate these cryptographic algorithms. Specifically, with the AES block cipher, there is a variety of approaches to dynamize the AES block cipher to enhance its security. Some of these methods center on incorporating S-boxes in AES that depend on a secret key [20–25], while others work on creating key-dependent MixColumn transformations for AES [26–28]. Notably, there are studies exploring the dynamization of both AES's S-boxes and MixColumn [29], or the dynamization of all three transformations: S-Boxes, MixColumn, and ShiftRow, which have also received attention [30–32]. Another current research direction is to make the XOR operation dynamic in AES [33, 34].

For the dynamic S-box approach in AES, in their work [20], the authors introduced a method for generating S-boxes that depend on the encryption key and possess favorable algebraic characteristics, including non-linearity, BIC, and SAC. Furthermore, an alternative approach to produce S-boxes that rely on the encryption key in AES was introduced in [21]. This method entails establishing a novel arrangement for the S-box through the use of a simulated key expansion algorithm. In [22], the authors introduced an innovative method for creating variable S-boxes by rearranging the S-box of AES. These adaptable S-boxes rely on a secret key and utilize an affine constant and an unconventional polynomial. For each additional key bit, a fresh S-box with rearranged values is produced, thus enhancing the intricacy of the algorithm. In [23], the authors presented an approach to create S-boxes that vary with the encryption key, employing a evolving approach. The evaluative experimentation of these key-dependent S-boxes was conducted, focusing on characteristics such as achieving a SAC, BIC, balanced output, non-linearity, and probabilities related to linear and differential approximation. In [24], the authors introduced four straightforward procedures for producing key-influenced S-boxes. To assess the quality of these S-boxes, they introduced eight standardized dissimilarity measurements. The authors outlined four methods for generating key-influenced S-boxes and scrutinized eight normalized dissimilarity measurements

employed to appraise the effectiveness of these key-dependent generation techniques. Furthermore, in [25], Murphy et al. introduced an approach for the differential cryptanalysis of key-influenced S-boxes, elucidating methods for performing cryptanalysis with the utilization of these S-boxes.

For the dynamic Mixcolumn approach in AES, in [26], the authors proposed a MixColumn transformation that depends on the encryption key, derived from the AES MDS matrix, using scalar multiplication on the rows of the matrix along with an extra m-bit key. An idea presented in [27] includes the creation of a diffusion layer that relies on the encryption key, achieved through scalar multiplication and immediate exponentiation. In [28], the authors introduced a collection of $n \times n$ binary matrices that can be employed to create dynamic matrices resembling AES and recursive MDS matrices.

For the approach of making multiple transformations dynamic in AES, in [29], the authors introduced dynamic S-boxes and novel MixColumn matrices that preserve favorable cryptographic characteristics when developing dynamic AES. In [30], an image encryption method rooted in symmetric cryptography was introduced. It utilizes transformations like MixColumns, ShiftRows, and SubByte, which are dynamically influenced by the encryption key. In [31], the authors presented a dynamic block cipher based on AES, in which AES parameters vary for each unique key. More precisely, the ShiftRows, the SubBytes, and MixColumns transformations adapt according to the key, leading to distinct behavior for every key. Extensive testing has verified the security of the proposed algorithm. In [32], a fresh and efficient AES algorithm, which is dependent on the key, is introduced. The authors have put forward an innovative approach to enhance the advanced encryption standard algorithm by employing dynamic sub-byte, mix-column, and shift rows operations to ensure secure communication. This novel work exhibits superior avalanche and strict avalanche effects when compared to the conventional AES algorithm.

In [33, 34], the authors introduced innovative techniques that employ key-dependent XOR tables utilizing 3D chaotic maps. The authors utilized XOR tables that rely on the initial confidential parameters. In [34], they established a fresh MDS matrix, however, regrettably, this matrix does not qualify as an MDS matrix. Furthermore, their approaches in [31, 32] still exhibit numerous weaknesses and shortcomings.

Based on our review of related works, to the best of our knowledge, we haven't come across any research that investigates the combination of animating both the ShiftRow and AddRoundKey transformations of AES. In this paper, we introduce algorithms for generating key-dependent AddRoundKey and ShiftRow transformations based on permutations. Subsequently, we apply these key-dependent transformations to AES to create a dynamic AES block cipher. We conduct security analysis and evaluate NIST's statistical criteria, as well as assess the entropy of AES and dynamic AES. Consequently, it becomes evident that dynamic AES block ciphers can significantly enhance the security of AES and meet rigorous randomness criteria, similar to AES.

The structure of the remaining part of the paper is as follows: Section II provides preliminaries. Section III introduces algorithms for generating key-dependent ShiftRow and AddRoundKey transformations based on permutations. Section IV adapts the AES block cipher using key-dependent ShiftRow and AddRoundKey operations. Section V is conclusion.

II. PRELIMINARIES

A. Introduction to Hadamard Matrices

A Hadamard matrix [35] of dimension d, with the initial row elements represented as $h_0, h_1, \dots, h_d$, can be designated in the following manner.

$$H = had(h_0, h_1, \dots, h_d)$$

Furthermore, a Hadamard matrix of size $2d \times 2d$ has the following form.

$$H = \begin{pmatrix} A & B \\ B & A \end{pmatrix}$$

where A and B are $d \times d$ matrices, d is even.

B. ShiffRow and AddRoundKey Transformations in AES

The ShiftRow operation in AES processes the state by left rotating the last three rows of the state with a varying number of rotation. Row 1 of the state remains unchanged, row 2 of the state left rotates by 1 byte, row 3 of the state left rotates by 2 bytes, and row 4 of the state left rotates by 3 bytes.

The AddRoundKey operation in AES performs a bitwise XOR between the state and a round key. The AddRoundKey operation is represented by a 4-bit XOR table as described in Table I.

TABLE I. THE 4-BIT XOR TABLE IN AES

	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

III. Proposing Algorithms to Generate Key-Dependent ShiftRow and AddRoundKey Transformations based on Permutation

A. Algorithm for Generating Key-Dependent ShiftRow

First, we analyze the diffusion capacity of active bytes (non-zero byte) through two rounds of AES. Fig. 1 represents the diffusion state of AES after the first round, with the initial state containing an active byte (indicated by the black cell).

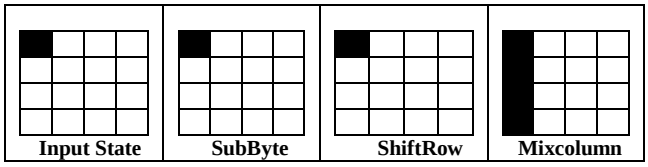

Fig. 1. Diffusion state of AES after the first round.

According to Fig. 1, after the SubByte and ShiftRow transformations in the first round, the number of active bytes in the state array remains at 1. Subsequently, during the MixColumn transformation, the number of active bytes becomes 4 due to the diffusion capabilities of the MDS matrix, and the number of active bytes is preserved when going through the AddRoundKey transformation. Thus, starting with 1 active byte, there will be 4 active bytes at the end of the first round.

Fig. 2 shows the diffusion state of AES after the second round.

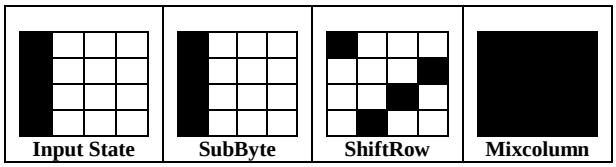

Fig. 2. Diffusion state of AES after the second round.

From Fig. 2, we can observe that the role of ShiftRow is extremely important in propagating active bytes from column 1 to all four columns of the state array. This allows, through the MixColumn transformation, the maximum number of active bytes to be achieved, which is 16 bytes in the state array.

Remark 1: The crucial point in designing the ShiftRow transformation is its capability to distribute active bytes from one column to all the remaining columns, ensuring that all four columns of the state array contain at least one active byte.

According to Fig. 2, the number of bytes rotated in each row of the state by the ShiftRow transformation in AES can be described as shown in Fig. 3.

Therefore, as long as the number of bytes rotated in each row is distinct and falls within the range [0, 3], active bytes will undoubtedly be propagated to all four columns of the state.

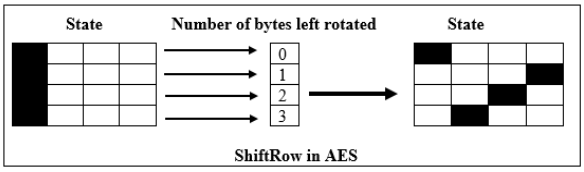

Fig. 3. The number of bytes rotated in each row of the state by the ShiftRow in AES.

Based on these observations, we propose the idea presented in Algorithm 1 to generate key-dependent ShiftRow transformations while ensuring the propagation of active bytes as described above.

Algorithm 1. Key-dependent ShiftRow Transformation Generation via Permutation

Input: A secret key K consists of k ($k \geq 128$) bits; A state S of size 4×4.

Output: The new ShiftRow operation for AES depends on the key K; A new state $\acute{S}$.

Step 1: Take the first two bits of the key K and convert them into an integer, denoted as a_0. Take the next two bits of K and convert them into an integer. If this integer is different from a_0, assign it to a_1; otherwise, shift the key K to the right by one bit until you obtain $a_1 \neq a_0$. Continue this process until you have four distinct integers: a_0, a_1, a_2, a_3.

Step 2: From the permutation (a_0, a_1, a_2, a_3) obtained in step 1, left rotate the rows of the state S as follows: left rotate a_0 bytes for row 1, left rotate a_1 bytes for row 2, left rotate a_2 bytes for row 3, left rotate a_3 bytes for row 4. The resulting state is denoted as $\acute{S}$.

Step 3: The left rotation operation as in step 2 is called *KD_ShiftRow*. The *KD_ShiftRow* operation will be used to replace the ShiftRow operation in AES.

Remark 2. Because there are 4! permutations of (0, 1, 2, 3), there will be $4! = 24$ key-dependent ShiftRow operations (*KD_ShiftRow*) that can be generated by Algorithm 1.

Example 1. If step 1 of Algorithm 1 results in the permutation $(a_0, a_1, a_2, a_3) = (2, 0, 3, 1)$, then the *KD_ShiftRow* operation obtained from Algorithm 1 will function as shown in Fig. 4.

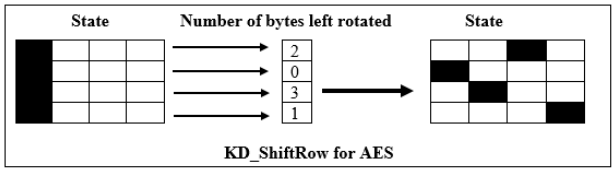

Fig. 4. An example of the KD_ShiftRow obtained from Algorithm 1.

B. Algorithm for Generating Key-Dependent AddRoundKey Transformations

From the original XOR table of AES (Table I), we have a remark regarding three essential Attributes that a XOR table must possess.

Remark 3. Three essential Attributes of an XOR table

- Attribute 1: Every row and column in the XOR table contains unique values within the range of 0 to 15.
- Attribute 2: The XOR table should exhibit symmetry along the principal diagonal, implying that $m\ XOR\ n = n\ XOR\ m$.
- Attribute 3: For any given m, n, and k elements within the XOR table where $m\ XOR\ n = k$, the following holds true: $m\ XOR\ k = n$ and $n\ XOR\ k = m$.

From the XOR table of AES, denote the 4×4 matrices as follows:

$$A_0 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 2 \\ 2 & 3 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{pmatrix}, A_1 = \begin{pmatrix} 4 & 5 & 6 & 7 \\ 5 & 4 & 7 & 6 \\ 6 & 7 & 4 & 5 \\ 7 & 6 & 5 & 4 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} 8 & 9 & 10 & 11 \\ 9 & 8 & 11 & 10 \\ 10 & 11 & 8 & 9 \\ 11 & 10 & 9 & 8 \end{pmatrix}, A_3 = \begin{pmatrix} 12 & 13 & 14 & 15 \\ 13 & 12 & 15 & 14 \\ 14 & 15 & 12 & 13 \\ 15 & 14 & 13 & 12 \end{pmatrix}.$$

$$\text{Set } \boldsymbol{A} = \begin{pmatrix} A_0 & A_1 & A_2 & A_3 \\ A_1 & A_0 & A_3 & A_2 \\ A_2 & A_3 & A_0 & A_1 \\ A_3 & A_2 & A_1 & A_0 \end{pmatrix}.$$

Remark 4. The matrices A_i $(0 \le i \le 3)$ are Hadamard matrices. And the matrix $\boldsymbol{A}$ is also a Hadamard matrix in the form $\boldsymbol{A} = had(0, 1, 2, \dots, 15)$. Therefore, the XOR table of AES is created from the Hadamard matrix $\boldsymbol{A}$, where **row 0** and **column 0** (bolded) of this XOR table are ordered according to the first row of matrix $\boldsymbol{A}$, meaning in the order 0, 1, 2, …, 15.

We can denote: $\boldsymbol{A} = had(A_0, A_1, A_2, A_3)$.

From Remark 4, it can be seen that by permuting the matrices A_i, new Hadamard matrices $\boldsymbol{A}$ can be generated. Based on this idea, we propose Algorithm 2 to generate a new XOR table and key-dependent AddRoundKey transformation by permuting the A_i matrices in matrix $\boldsymbol{A}$.

Algorithm 2. Generating a new 4-bit XOR table and key-dependent AddRoundKey transformation based on permutation.

Input: A secret key K consisting of k $(k \ge 128)$ bits, the original XOR table of AES.

Output: A new 4-bit XOR table; Key-dependent AddRoundKey operation.

Step 1: Take the first two bits of the secret key K and convert these two bits into an integer, denoted as x_0. Take the next two bits of K and convert them into an integer. If this integer is different from x_0, assign this integer to x_1; otherwise, shift the key K to the right by one bit until $x_1 \ne x_0$ is obtained. Do the same for the remaining bits until four distinct integers are obtained: x_0, x_1, x_2, x_3.

Step 2: Construct a permutation-based Hadamard matrix using the permutation (x_0, x_1, x_2, x_3) obtained in step 1, which has the form: $\hat{\boldsymbol{A}} = had(A_{x_0}, A_{x_1}, A_{x_2}, A_{x_3})$.

Step 3: Construct a new 4-bit XOR table based on the Hadamard matrix $\hat{\boldsymbol{A}}$ such that the elements in the first row and the first column of this new XOR table follow the order of elements in the first row of matrix $\hat{\boldsymbol{A}}$.

Step 4: Reorder the rows and columns of the new XOR table so that both row 0 and column 0 of the new XOR table follow an increasing order from 0 to 15.

Step 5: Replace the regular bitwise XOR operation in the AddRoundKey transformation of AES with the new XOR operation determined by the new XOR table created in step 4. The result is the *KD_AddRoundKey* operation, which is used in place of the AddRoundKey operation in AES.

Remark 5. Since there are 4! permutations of (0, 1, 2, 3), there will be a total of $4! = 24$ new XOR tables generated by Algorithm 2, corresponding to the number of *KD_AddRoundKey* operations obtained.

Example 2. Suppose that step 1 of Algorithm 2 yields the permutation $(x_0, x_1, x_2, x_3) = (3, 2, 0, 1)$. The resulting Hadamard matrix is as follows:

$$\hat{\boldsymbol{A}} = had(A_3, A_2, A_0, A_1) = \begin{pmatrix} A_3 & A_2 & A_0 & A_1 \\ A_2 & A_3 & A_1 & A_0 \\ A_0 & A_1 & A_3 & A_2 \\ A_1 & A_0 & A_2 & A_3 \end{pmatrix}$$

In that case, the resulting new XOR table after Algorithm 2 is presented in Table II.

TABLE II. THE NEW XOR TABLE GENERATED FROM ALGORITHM 1

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

From the three attributes of an XOR table as mentioned in Remark 3, we prove the accuracy of the new XOR table generated by Algorithm 1 with the following proposition.

Proposition 1. The new XOR table generated by Algorithm 1 satisfies the three necessary attributes of an XOR table.

Proof.

Since the Hadamard matrix $\hat{\boldsymbol{A}}$ is essentially a permutation of elements within a row of the original XOR table of AES, Atrribute 1 of the new XOR table is satisfied.

Matrix

$$\acute{\boldsymbol{A}} = had\left(A_{x_0}, A_{x_1}, A_{x_2}, A_{x_3}\right) = \begin{pmatrix} A_{x_0} & A_{x_1} & A_{x_2} & A_{x_3} \\ A_{x_1} & A_{x_0} & A_{x_3} & A_{x_2} \\ A_{x_2} & A_{x_3} & A_{x_0} & A_{x_1} \\ A_{x_3} & A_{x_2} & A_{x_1} & A_{x_0} \end{pmatrix}$$

is a Hadamard one, and the matrices $A_{x_i}(0 \le i \le 3)$ are also Hadamard matrices, so the matrix $\acute{\boldsymbol{A}}$ is symmetric across the main diagonal. Thus, Attribute 2 of the new XOR table is satisfied.

The matrix $\acute{\boldsymbol{A}}$ is denoted as follows: $\acute{\boldsymbol{A}} = had(a_0, a_1, \dots, a_{15})$.

Comparing with the original XOR table of AES, it can be observed that the elements of the new XOR table have been replaced by a one-to-one mapping as follows: $0 \to a_0$, $1 \to a_1$, …, $15 \to a_{15}$. This substitution is applied to all elements of the original XOR table, including both row 0 and column 0.

According to Attribute 3 of the original XOR table: if $i\ XOR\ l = j$, it always holds that $i\ XOR\ j = l$ and $l\ XOR\ j = i$.

From the replacement operation, we also have the corresponding relationship: if $a_i\ XOR\ a_l = a_j$, it always holds that $a_i\ XOR\ a_j = a_l$ and $a_l\ XOR\ a_j = a_i$. Therefore, Attribute 3 of the new XOR table is satisfied.

On the other hand, rearranging the order of rows and columns of the new XOR table so that row 0 and column 0 of that XOR table are in increasing order from 0 to 15 will not affect the values in the XOR table. Therefore, all three properties are satisfied for the new XOR table.

C. Proposing the Combined Algorithm

In this section, we will propose a combined algorithm to generate both the key-dependent ShiftRow and AddRoundKey transformations for the AES block cipher.

Algorithm 3. Generating key-dependent ShiftRow and AddRoundKey transformations.

Input: A secret key K consisting of k ($k \ge 128$) bits; The original XOR table of AES.

Output: A new 4-bit XOR table; New key-dependent ShiftRow and AddRoundKey transformations for AES.

Step 1: Take the first two bits of key K and convert these two bits into integers, denoted as a_0. Take the next two bits of K and convert them into integers. If this integer is different from a_0, assign it to a_1; otherwise, shift key K to the right by one bit until you obtain $a_1 \neq a_0$.. Repeat this process until you have four distinct integers: a_0, a_1, a_2, a_3.

Step 2: From the permutation (a_0, a_1, a_2, a_3) obtained in step 1, for any arbitrary state S, perform a left rotation on the rows of the state S as follows: rotate row 1 to the left by a_0 bytes, rotate row 2 to the left by a_1 bytes, rotate row 3 to the left by a_2 bytes, rotate row 3 to the left by a_2 bytes. This left rotating operation is named *KD_ShiftRow* and will be used to replace the ShiftRow operation in AES.

Step 3: Following the same procedure as in step 1, with the next bits of the secret key K after step 1, we obtain a permutation (x_0, x_1, x_2, x_3).

Step 4: Construct a Hadamard matrix based on the permutation (x_0, x_1, x_2, x_3) obtained in step 3, in the form of: $\acute{\boldsymbol{A}} = had(A_{x_0}, A_{x_1}, A_{x_2}, A_{x_3})$.

Step 5: Construct a new 4-bit XOR table based on the Hadamard matrix $\acute{\boldsymbol{A}}$, with the first row and column of the new XOR table containing elements in the same order as those in the first row of matrix $\acute{\boldsymbol{A}}$. Rearrange the rows and columns of the new XOR table so that the first row and column follow an increasing order from 0 to 15.

Step 6: Replace the usual bitwise XOR operation in the AddRoundKey transformation of AES with the new XOR operation defined by the XOR table generated in step 5. The result is the *KD_AddRoundKey* operation, which is used in place of AddRoundKey in AES.

IV. Adapt the AES Block Cipher by Incorporating the Key-Dependent ShiftRow and AddRoundKey Transformations

A. Implementation of Experiments

Execute the combined algorithm in Algorithm 3 to obtain two key-dependent transformations: *KD_ShiftRow* and *KD_AddRoundKey*. Then, use these two transformations to replace the original ShiftRow and AddRoundKey in AES. The resulting dynamic AES algorithm is denoted as *ShiftAES*. Fig. 5 illustrates the diagram of encryption/decryption rounds in the *ShiftAES* algorithm.

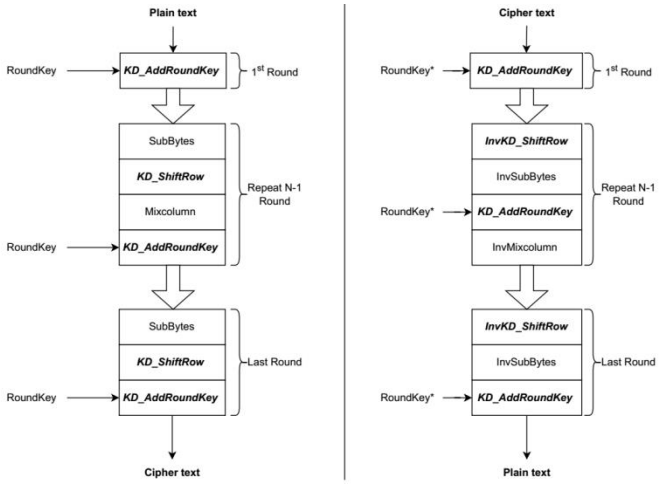

Fig. 5. Encryption / decryption round diagram of the ShiftAES algorithm.

For experiment, we select the AES-128 block cipher with a XOR table of 4-bit. This led to the development of a key-dependent dynamic block cipher algorithm, which we named ShiftAES-128 based on Algorithm 3. We implement these algorithms using C++ on an Asus K43SJ Laptop (Core i5-2430M, 500GB, HDD 6GB RAM, Nvidia Geforce GT 520M).

B. Security Analysis

In the realm of block ciphers, encompassing AES, the most potent threats manifest in the form of differential attacks [18, 19], linear attacks [18, 36], or their derivatives. In the case of differential attacks, attackers must rely on predefined differential patterns to execute their strategies. Our ShiftAES block cipher ensures the continuous alteration of differential patterns [17], contingent upon the encryption key. This

dynamic behavior significantly enhances the security of the ShiftAES block cipher, as it becomes exceedingly arduous for potential attackers to compute the differentials required for launching attacks.

The use of the key-dependent ShiftRow operation makes it considerably more challenging for attackers as it conceals the exact rotation values applied in AES's *KD_ShiftRow*, rendering decryption significantly more difficult. Moreover, the utilization of key-dependent AddRoundKey operations will heighten the intricacy of key-related attacks [37]. It disrupts the conventional property of XOR tables, where different inputs with the same difference yield an output difference of 0. The dynamic nature of the relationship between input differences, driven by the variable key, adds an additional layer of security to block ciphers that incorporate such elements.

Combining both KD_ShiftRow and KD_AddRoundKey makes the attacker's task significantly more challenging. In the case of AES, attackers know how ShiftRow and AddRoundKey work, they might collect a large number of plaintext/ciphertext pairs for differential or linear cryptanalysis attacks. This number of pairs could be very large, let's assume it's 2^{80}. However, with the *ShiftAES* algorithm, there can be 24 possibilities for *KD_ShiftRow* and 24 possibilities for dynamic XOR tables, which results in $24 \times 24 = 576$ possible combinations for ShiftRow and AddRoundKey used in *ShiftAES*. In this scenario, the number of plaintext/ciphertext pairs required for an attack is not just 2^{80} but 576×2^{80}. This number, at first glance, may not seem significantly larger, but in reality, it's extremely huge and has practical implications. So, for the dynamic AES block cipher, an attacker needs to collect plaintext/ciphertext pairs that are 576 times more than in the case of regular AES. This is not an easy task in practice. Furthermore, when the encryption/decryption switches to a different secret key, meaning different *KD_ShiftRow* and *KD_AddRoundKey* are used, the attacker might not have enough time to gather a sufficient number of plaintext/ciphertext pairs to carry out the attack.

Therefore, it can be seen that the proposed dynamic method can significantly increase the security of AES. Furthermore, as mentioned in the introduction, our proposed method is a novel approach, which may be highly beneficial for cryptography designers in designing secure dynamic SPN block ciphers.

C. Evaluating the Random Statistical Standards

In this section, we assess the random statistical standards according to NIST SP 800-22 [38] and Shannon Entropy criteria [39] of the AES and *ShiftAES* block cipher.

NIST SP 800-22 [38] has been developed to serve as the primary and extensively utilized means for evaluating the statistical randomness of random or pseudorandom number generators in the field of cryptography. NIST's set of randomness assessments encompasses a total of 15 tests, which are outlined below.

Random Excursions Test; Frequency Test within a Block; Test for the Longest Run of Ones in a Block; Approximate Entropy Test; Non-overlapping Template Matching Test; Binary Matrix Rank Test; Random Excursions Variant Test; Cumulative Sums (Cusum) Test; Frequency (Monobit) Test; Linear Complexity Test; Runs Test; Serial Test; Overlapping Template Matching Test; Discrete Fourier Transform (Spectral) Test; Maurer's "Universal Statistical" Test.

For a given input sequence, every test computes a respective p-value, which is then compared to the significance level $\alpha = 0.01$. Should $p_i \geq \alpha$, it is inferred that the sequence exhibits randomness, and conversely.

Entropy, also known as information entropy, is described as the degree of unpredictability concerning an individual's knowledge or the result of an experiment before it is observed, as well as the connected deterministic characteristics for forecasting its value. Higher entropy indicates increased uncertainty when forecasting an observation's value. Shannon entropy represents one category of information entropy developed by Shannon in [39].

1) Evaluate the shannon entropy change using the ENT tool: Shannon entropy is a vital indicator of randomness. In this section, we assess the alteration in Shannon entropy for the four datasets, namely LW, HW, AV1, and Rot, across each round of the DAES block cipher. This analysis aims to provide a comprehensive view of the randomness at each round. We utilized the ENT tool [40] to perform the evaluations and acquired the subsequent outcomes.

The results of the entropy evaluation are presented in Table III.

TABLE III. EVALUATION RESULTS OF ENTROPY FOR AES AND SHIFTAES

Rounds	AES	ShiftAES
1	4.994138	5.014244
2	7.077322	7.040298
3	8	7.999285
4	7.999989	7.999987
5	7.999990	7.999990
6	7.999989	7.999989
7	7.999990	7.999989
8	7.999990	7.999988
9	7.999989	7.999988
10	7.999989	7.999988

The entropy evaluation results show that after three rounds, the entropy of data encrypted by both original AES and ShiftAES is approximately 8 bits/byte. This implies that both block ciphers achieve randomness properties with three or more rounds.

2) The evaluation results according to NIST SP 800-22: We conducted an evaluation of the statistical tests for rounds 1, 2, ..., 10 of both AES and ShiftAES. Tables IV to VIII display the results of the randomness evaluation of AES and ShiftAES across 1, 2, 3, 4, and 10 rounds.

TABLE IV. RANDOMNESS EVALUATION OF AES AND SHIFTAES OVER 1 ROUND

Test	AES-1R	ShiftAES-1R
AppEnt	0	0
BlocFreq	0	0
Cusum 1	0	0
Cusum 2	0	0
FFT	0	0
Freq	0	0
Linear Complexity	0.631288	0.752788
LongRun	0	0
NonOverLap	148 p-values = 0	148 p-values = 0
OverLap	0	0
RanEx	8 p-values = 0	8 p-values = 0
RanEx Var	18 p-values = 0	18 p-values = 0
Rank	0	0
Run	0	0
Serial 1	0	0
Serial 2	0	0
Universal	0	0

TABLE V. RANDOMNESS EVALUATION OF AES AND SHIFTAES OVER 2 ROUNDS

Test	AES-2R	ShiftAES-2R
AppEnt	0	0
BlocFreq	0	0
Cusum 1	0	0
Cusum 2	0	0
FFT	0	0
Freq	0	0
Linear Complexity	0.204973	0.051540
LongRun	0	0
NonOverLap	148 p-values = 0	148 p-values = 0
OverLap	0	0
RanEx	8 p-values = 0	3 p-values >= 0.01, 5 p-values < 0.01
RanEx Var	18 p-values = 0	18 p-values >= 0.01
Rank	0	0
Run	0	0
Serial 1	0	0
Serial 2	0	0
Universal	0	0

TABLE VI. RANDOMNESS EVALUATION OF AES AND SHIFTAES OVER 3 ROUNDS

Test	AES-3R	ShiftAES-3R
AppEnt	0.925352	0
BlocFreq	0.831754	0.999999
Cusum 1	0	0
Cusum 2	0	0
FFT	0	0
Freq	0.993653	0
Linear Complexity	0.978496	0.876043
LongRun	0.041736	0
NonOverLap	148 p-values >= 0.01	148 p-values <= 0.01
OverLap	0	0
RanEx	3 p-values >= 0.01, 5 p-values < 0.01	8 p-values = 0
RanEx Var	2 p-values >= 0.01, 16 < 0.01	18 p-values = 0
Rank	0.611499	0.662867
Run	0.340678	0
Serial 1	0.437274	0.029134
Serial 2	0.113612	0.577590
Universal	0.000003	0

TABLE VII. RANDOMNESS EVALUATION OF AES AND SHIFTAES OVER 4 ROUNDS

Test	AES-4R	ShiftAES-4R
AppEnt	0.898152	0.826485
BlocFreq	0.970639	0.862578
Cusum 1	0.254482	0.046664
Cusum 2	0.230798	0.010182
FFT	0.339743	0.896743
Freq	0.363824	0.034995
Linear Complexity	0.540695	0.080683
LongRun	0.162664	0.675598
NonOverLap	148 p-values >= 0.01	148 p-values >= 0.01
OverLap	0.220930	0.628086
RanEx	8 p-values >= 0.01	8 p-values >= 0.01
RanEx Var	18 p-values >= 0.01	18 p-values >= 0.01
Rank	0.615078	0.995817
Run	0.225007	0.415387
Serial 1	0.987050	0.564141
Serial 2	0.996948	0.403646
Universal	0.571751	0.494948

TABLE VIII. RANDOMNESS EVALUATION OF AES AND SHIFTAES OVER 10 ROUNDS

Test	AES-10R	ShiftAES-10R
AppEnt	0.863909	0.002023
BlocFreq	0.415025	0.510530
Cusum 1	0.537509	0.143294
Cusum 2	0.927344	0.306088
FFT	0.307167	0.782720
Freq	0.604613	0.234997
Linear Complexity	0.002433	0.057640
LongRun	0.748478	0.465346
NonOverLap	148 p-values >= 0.01	148 p-values >= 0.01
OverLap	0.000320	0.133895
RanEx	8 p-values >= 0.01	8 p-values >= 0.01
RanEx Var	18 p-values >= 0.01	18 p-values >= 0.01
Rank	0.908691	0.312501
Run	0.369725	0.584987
Serial 1	0.759251	0.618985
Serial 2	0.412087	0.396930
Universal	0.674145	0.165542

The results show that both the original AES and *ShiftAES* achieve randomness when they pass all the tests. For rounds fewer than four, both AES and *ShiftAES* still fail in some tests. From round 5 to round 10, both AES and *ShiftAES* block ciphers pass all tests. It means that the p-values of those tests are all greater than or equal to 0.01 from round 5 to round 10. Therefore, both block ciphers exhibit randomness when using four rounds or more.

Combining the evaluation results based on entropy and statistical tests from NIST SP 800-22, we conclude that the *ShiftAES* algorithm achieves randomness with a number of rounds greater than or equal to four and is equivalent to the original AES algorithm.

V. Conclusion

In this paper, we propose algorithms for generating key-dependent AddRoundKey and ShiftRow transformations based on permutations. Subsequently, we apply these key-dependent transformations to AES to create a dynamic AES block cipher. We conduct a security analysis and evaluate the statistical standards of NIST, assess the entropy of both AES and the dynamic AES (ShiftAES). Consequently, it is evident that the dynamic AES block cipher can significantly strengthen the security of AES and meets statistical randomness criteria similar to AES. This result is significant both in theory and practice, providing cryptographic researchers with a new method to improve block cipher security. Our future research direction involves further development of other dynamic algorithms to strengthen the resilience of SPN block ciphers against cryptanalysis.

Statements and Declarations

Funding: No funding was received for conducting this study.

Financial interests: The author declare they have no financial interests.

Competing interests: The author 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] R. Bharathi, R. and N. Parvatham, "LEA-Siot: Hardware architecture of lightweight encryption algorithm for secure IoT on FPGA platform," International Journal of Advanced Computer Science and Applications, vol. 11, 2020.

[2] A. Maetouq, S. M. Daud, N. A. Ahmad, N. Maarop, N. N. A. Sjarif and H. Abas, "Comparison of hash function algorithms against attacks: A review," International Journal of Advanced Computer Science and Applications, vol. 9, 2018.

[3] H. Abroshan, "A hybrid encryption solution to improve cloud computing security using symmetric and asymmetric cryptography algorithms," International Journal of Advanced Computer Science and Applications, vol. 12, pp. 31–37, 2021.

[4] L. T. Keliher, "Linear cryptanalysis of substitution-permutation networks," *Queen's University, Kingston, Ontario, Canada*, 2003.

[5] A. M. Youssef, S. E. Tavares and H. M. Heys, "A new class of substitution permutation networks," in Proceedings of Third Annual Workshop on Selected Areas in Cryptography (SAC 96), Queens University, Kingston, Canada, vol. 96, pp. 132–147, 1996.

[6] A. S. Alanazi, N. Munir, M. Khan and I. Hussain, "A novel design of audio signals encryption with substitution permutation network based on the Genesio-Tesi chaotic system," Multimedia Tools and Applications, pp. 1–17, 2023.

[7] S. Beg, N. Ahmad, A. Anjum, M. Ahmad, A. Khan, F. Baig and A. Khan, "S-box design based on optimize LFT parameter selection: a practical approach in recommendation system domain," Multimedia Tools and Applications, vol. 79, pp. 11667–11684, 2020. https://doi.org/10.1007/s11042-019-08464-6.

[8] S. Mister and C. Adams, "Practical S-box design," Workshop on Selected Areas in Cryptography, SAC, vol. 96, pp. 61–76, August 1996.

[9] A. M. Youssef and S. E. Tavares, "Resistance of balanced s-boxes to linear and differential cryptanalysis," Information Processing Letters, vol. 56, pp. 249–252, 1995. https://doi.org/10.1016/0020-0190(95)00156-6.

[10] S. Farwa, T. Shah, N. Muhammad, N. Bibi, A. Jahangir and S. Arshad, "An image encryption technique based on chaotic S-box and Arnold transform," International Journal of Advanced Computer Science and Applications, vol. 8, 2017.

[11] B. W. Koo, H. S. Jang and J. H. Song, "On constructing of a 32×32 binary matrix as a diffusion layer for a 256-bit block cipher," International Conference on Information Security and Cryptology, Springer, Berlin, Heidelberg, pp. 51–64, December 2006.

[12] M. KUMAR, P. YADAV, S. K. Pal and A. PANIGRAHI, "Secure and Efficient Diffusion Layers for Block Ciphers," Journal of Applied Computer Science & Mathematics, vol. 11, pp. 24, 2017.

[13] H. N. Noura and A. Chehab, "Efficient binary diffusion matrix structures for dynamic key-dependent cryptographic algorithms," Journal of Information Security and Applications, vol. 68, pp. 103264, 2022.

[14] G. Tuncay, F. B. Sakallı, M. K. Pehlivanoğlu, G. G. Yılmazgüç, S. Akleylek and M. T. Sakallı, M. T, "A new hybrid method combining search and direct based construction ideas to generate all 4× 4 involutory maximum distance separable (MDS) matrices over binary field extensions," PeerJ Computer Science, vol. 9, e1577, 2023.

[15] J. Daemen and V. Rijmen, "AES proposal: Rijndael (version 2)," NIST AES website, 1999.

[16] J. Daemen and V. Rijmen, "The design of Rijndael," New York: Springer-verlag, vol. 2, 2002.

[17] L. R. Knudsen and C. V. Miolane, "Counting equations in algebraic attacks on block ciphers," International Journal of Information Security, vol. 9, pp. 127–135, 2010. https://doi.org/10.1007/s10207-009-0099-9.

[18] H. M. Heys and S. E. Tavares, "The design of product ciphers resistant to differential and linear cryptanalysis," Journal of Cryptology, vol. 9, pp. 1–19, 1996.

[19] X. Lai, J. L. Massey and S. Murphy, "Markov ciphers and differential cryptanalysis," Advances in Cryptology—EUROCRYPT'91: Workshop on the Theory and Application of Cryptographic Techniques Brighton, UK, April 8–11, 1991 Proceedings 10, Springer Berlin Heidelberg, pp. 17–38, April 1991.

[20] A. Y. Al-Dweik, I. Hussain, M. Saleh and M. T. Mustafa, "A novel method to generate key-dependent s-boxes with identical algebraic properties," Journal of Information Security and Applications, vol. 64, p. 103065, 2022. https://doi.org/10.1016/j.jisa.2021.103065.

[21] K. Kazlauskas and J. Kazlauskas, "Key-dependent S-box generation in AES block cipher system," Informatica, vol. 20, pp. 23–34, 2009.

[22] P. Agarwal, A. Singh and A. Kilicman, "Development of key-dependent dynamic S-boxes with dynamic irreducible polynomial and affine constant," *Advances in Mechanical Engineering*, vol. 10, 2018. https://doi.org/10.1177/1687814018781638.

[23] A. Ejaz, I. A. Shoukat, U. Iqbal, A. Rauf and A. Kanwal, "A secure key dependent dynamic substitution method for symmetric cryptosystems," PeerJ Computer Science, vol. **7**, e587, 2021.

[24] G. Vaicekauskas, K. Kazlauskas and R. Smaliukas, "A novel method to design S-boxes based on key-depdendent permutation schemes and its quality analysis," International Journal of Advanced Computer Science and Applications, vol. 7, pp. 93–99, 2016.

[25] S. Murphy and M. J. B. Robshaw, "Key-dependent S-boxes and differential cryptanalysis," Designs, Codes and Cryptography, vol. 27, pp. 229–255, 2002.

[26] G. Murtaza, A. A. Khan, S. W. Alam and A. Farooqi, "Fortification of AES with dynamic mix-column transformation," Cryptology ePrint Archive, 2011.

[27] T. T. Luong, "Building the dynamic diffusion layer for SPN block ciphers based on direct exponent and scalar multiplication," Journal of Science and Technology on Information security, vol. 1, pp. 38–45, 2022.

[28] M. R. M. Shamsabad and S. M. Dehnavi, "Dynamic MDS diffusion layers with efficient software implementation," International Journal of Applied Cryptography, vol. 4, pp. 36–44, 2020. https://doi.org/10.1504/IJACT.2020.107164.

[29] T. Xu, F. Liu and C. Wu, "A white-box AES-like implementation based on key-dependent substitution-linear transformations," Multimedia Tools and Applications, vol. 77, pp. 18117–18137, 2018. https://doi.org/10.1007/s11042-017-4562-8.

[30] V. Sawant, A. Solkar and R. Mangrulkar, "Modified Symmetric Image Encryption Approach Based on Mixed Column and Substitution Box," Journal of Applied Security Research, pp. 1–34, 2022. https://doi.org/10.1080/19361610.2022.2150498.

[31] A. Altigani, S. Hasan, B. Barry, S. Naserelden, M. A. Elsadig and H. T. Elshoush, "A polymorphic advanced encryption standard–a novel approach," IEEE Access, vol. 9, pp. 20191–20207, 2021. DOI: 10.1109/ACCESS.2021.3051556.

[32] T. Manoj Kumar and P. Karthigaikumar, "A novel method of improvement in advanced encryption standard algorithm with dynamic shift rows, sub byte and mixcolumn operations for the secure communication," International Journal of Information Technology, vol. 12, pp. 825–830, 2020.

[33] A. I. Salih, A. Alabaichi and A. S. Abbas, "A novel approach for enhancing security of advance encryption standard using private XOR table and 3D chaotic regarding to software quality factor," ICIC Express Letters Part B: Applications, An International Journal of Research and Surveys, vol. 10, pp. 823–832, 2019. DOI: 10.24507/icicelb.10.09.823.

[34] A. I. Salih, A. M. Alabaichi and A. Y. Tuama, "Enhancing advance encryption standard security based on dual dynamic XOR table and mixcolumns transformation," Indonesian Journal of Electrical Engineering and Computer Science, vol. 19, pp. 1574–1581, 2020. DOI: 10.11591/ijeecs.v19.i3.

[35] M. Sajadieh, M. Dakhilalian, H. Mala and B. Omoomi, "On construction of involutory MDS matrices from Vandermonde Matrices in GF (2^q)," Designs, Codes and Cryptography, vol. 64, pp. 287–308, 2012.

[36] M. Matsui, "Linear cryptanalysis method for DES cipher," Advances in Cryptology—EUROCRYPT'93: Workshop on the Theory and Application of Cryptographic Techniques Lofthus, Norway, Proceedings 12, Springer Berlin Heidelberg, pp. 386–397, May 1993.

[37] J. Guo, L. Song and H. Wang, "Key structures: Improved related-key boomerang attack against the full AES-256," In Australasian Conference on Information Security and Privacy, Cham: Springer International Publishing, pp. 3–23, November 2022.

[38] L. E. Bassham III, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, and et al, "Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications," National Institute of Standards & Technology, 2010.

[39] C. E. Shannon, "A mathematical theory of communication, Bell System Technical Journal 27," pp. 379–423 and 623–656, 1948.

[40] W. John, "ENT A Pseudorandom Number Sequence Test Program", 2008. Retrieved September, 3, 2022.