Received 22 February 2024, accepted 31 March 2024, date of publication 10 April 2024, date of current version 19 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3387268

RESEARCH ARTICLE

AES Security Improvement by Utilizing New Key-Dependent XOR Tables

TRAN THI LUONG[1], **NGUYEN NGOC CUONG**[1], **AND BAY VO**[2]
[1]Academy of Cryptography Techniques, Thanh Trì, Hanoi 100000, Vietnam
[2]Faculty of Information Technology, HUTECH University, Ho Chi Minh City 700000, Vietnam
Corresponding author: Bay Vo (vd.bay@hutech.edu.vn)

ABSTRACT Increasing the security of block ciphers is a topic of great interest today, and thus there is a variety of work to enhance the strength of such ciphers. There are also many studies focusing on the Advanced Encryption Standard (AES), presenting methods of making block ciphers dynamic to improve their security. Animating methods can perform block cipher transformations such as substitution or permutation, or both. In this article, we propose an algorithm to create new, key-dependent XOR tables from an initial secret key. At the same time, we prove that in the ciphertext the new XOR operation can preserve the independent, co-probability distribution of the random key. We then apply these new XOR tables to make AES dynamic at the Addroundkey transformation. We created a considerable number of XOR tables, about $(16!)^2$ tables. With such a vast number of key-dependent dynamic XOR tables, cryptanalysts will have great difficulty finding the actual XOR table used in the modified AES block cipher. Therefore, with our new XOR tables, AES will be significantly enhanced.

INDEX TERMS New XOR table, AES, dynamic XOR table, key-dependent.

I. INTRODUCTION

According to Shannon [1], in order to design a good block cipher, the round function of the block cipher must ensure two properties, diffusion and confusion [2], [3]. S-boxes [4], [5], [6] are often used to provide confusion, and linear transformations [7], [8], [9] are often used to provide diffusion. SPNs (permutation-substitution networks) are a familiar structure of block ciphers. The round function of an SPN block cipher [10], [11], [12], [13], [14] usually consists of three transformations: key addition, permutation, and substitution. SPN block ciphers are commonly used to secure data in many applications today. In addition to block ciphers, there are a variety of fields related to information security that are also of interest today as digital signatures [15], [16], [17], [18] blockchain [16], [19], [20], big data [21].

AES [22], [23] is an SPN block cipher, and it is a block cipher standard that was developed in the US in 2001, ratified by NIST. It also includes three transformations in the round function: key addition, substitution, and linearity. While chaotic maps and DNA-based transformations are widely used today, they are not employed in AES. The reason for this is that the AES authors opted for high nonlinear substitutions by s-boxes and high diffusion through MDS matrices. These two transformations maximize the security of AES.

AES can be considered one of the strongest and most widely used encryption algorithms in the world today, extensively applied in various security applications. However, AES itself has inherent vulnerabilities that cryptanalysts could potentially exploit to attack this block cipher. Some cryptographic experts express concerns about AES's security. If attack techniques improve, AES could be compromised. Firstly, AES's mathematical structure is relatively straightforward, which attackers could leverage for future attacks. For instance, these attacks might resemble algebraic attacks [24]. Secondly, two of the most potent attacks on block ciphers, linear attacks [5], [11] and differential attacks [11], [25] still pose significant threats to AES, as cryptanalysts can accumulate a substantial amount of plaintext-ciphertext pairs. Thirdly, as supercomputers continue to advance, brute-force attacks on keys could become feasible in the future. Additionally, with the advent of quantum computers, the security

The associate editor coordinating the review of this manuscript and approving it for publication was Jiafeng Xie.

of cryptographic algorithms, including AES, is seriously jeopardized. For example, Grover's algorithm [26] would necessitate doubling the key length of block ciphers for similar levels of security.

Therefore, numerous research efforts focus on enhancing AES through animation. Some of these focus on proposing dynamic S-boxes for AES [27], [28], [29], [30], [31], [32], while others create dynamic MixColumn transformations for AES [33], [34], [35], [36]. Some studies animating both the S-boxes and the MixColumn transformation of AES are also of interest [37], [38], [39]. In addition to the above research directions, some studies work on making block ciphers dynamic based on deoxyribonucleic acid (DNA) and on chaotic maps [40], [41], [42].

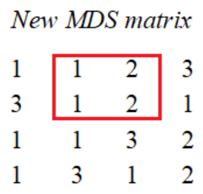
The works noted above study how to animate AES at the diffusion layer, substitution layer, or both, or how to make block ciphers dynamic based on DNA and chaotic maps. However, none of these works has worked on making AES dynamic at the Addroundkey transformation. In [43], the authors presented a dynamic method based on an individual XOR table depending on the secret parameters that are input. These secret parameters are used as the input to a 3D chaos map, and then the output of this map is used to generate that individual XOR table. In [44], the authors presented two XOR tables that depend on the initial secret parameters. These parameters are also passed through a 3D chaos map to give some output values. These output values are used to define two different XOR tables and also define a new MDS matrix. The authors use the XOR tables and the MDS matrix thus generated to apply to AES. The authors then evaluate the new AES block cipher against some statistical criteria.

Through careful analysis of the methods proposed in [43] and [44], we acknowledge that the methods are both valuable and new. However, they methods still contain many vulnerabilities and many defects, as noted below.

- In [43] and [44], three conditions of an XOR table are created including:
  + The total of each row or column equals 120.
  + Every row and column contain a series of non-repeating numbers ranging from 0 to 15.
  + The main diagonal of the XOR table exhibits symmetry.

The second property implies the first property. Moreover, the authors have failed to demonstrate that these characteristics ensure accurate decryption. Specifically, they did not indicate that if $aXORb = c$, then deducing $a = cXORb$ and $b = cXORa$ is viable, despite their created XOR tables satisfying this property. The explanation provided in Algorithm (1) of [43] is ambiguous, and it does not establish the adequacy of producing XOR tables in [43].

- Generating numeric keys from strings produced by 3D maps and Chebyshev maps of [43] and [44] is an inefficient process.
- The Z array [44] is utilized to form a key matrix, which is then used to generate a new MDS matrix based on the matrix in AES's MixColumn transformation. Nevertheless, the resulting matrix turned out not to be MDS but rather contained at least a square submatrix of size 2 that was singular (as depicted in Fig. 1).



$$\begin{array}{cccc} 1 & 1 & 2 & 3 \\ 3 & 1 & 2 & 1 \\ 1 & 1 & 3 & 2 \\ 1 & 3 & 1 & 2 \end{array}$$

FIGURE 1. The MDS matrix that is dynamically generated by [44].

Therefore, while the proposed approach of animating the block cipher in the Addroundkey transformation using XOR tables presented in [43] and [44] is an intriguing and innovative research avenue that enhances the security of the AES block cipher, the authors have yet to demonstrate the essential properties of an XOR table thoroughly. Additionally, the method for creating new MDS matrices outlined in [44] is imprecise, resulting in a dynamically generated matrix that does not meet the criteria of an MDS matrix. As a result, the authors have not proven that the new XOR can maintain the independent, co-probability distribution of the random key within the ciphertext.

In our research article, we introduce a novel approach to creating dynamic AES at the key-addition transformation using dynamic XOR tables that depend on the encryption key. We give the essential characteristics of an XOR table and we prove the correctness of the new XOR table generated by our method. Then we prove that our new XOR operation can maintain the independent, co-probability distribution of the random key within the ciphertext, which has not been previously addressed in [43] and [44]. We apply the dynamic XOR tables we created to improve AES. We then conduct a detailed security analysis on the dynamic AES block cipher using our dynamic method, which generates a higher number of dynamic XOR tables compared to [43] and [44]. With this large amount of key-dependent XOR tables, cryptanalysts will face considerable challenges in identifying the actual XOR table used in the altered AES. Furthermore, we assess the randomness of the altered AES using NIST tests and confirm that it passes these tests. As such, our proposed method has the potential to enhance the security of the AES block cipher against various formidable attacks on modern block ciphers. In this paper, we conducted experiments on AES with a 128-bit key version because it is the most familiar version. Additionally, we selected a representative version for ease of understanding in our simulations, but other versions can be readily implemented as well.

In practice, the AES block cipher is widely employed to secure information in various safety domains. However, as analyzed, AES itself harbors several inherent issues that could potentially render it vulnerable in the not-too-distant future. On the other hand, there exist many fields in reality that demand stringent data security requirements, such as in the realms of security, military, and defense. In these areas, there is a necessity for employing cryptographic systems with high and stringent security levels. Therefore, dynamic block cipher algorithms can meet these demands as they significantly enhance the security of static block ciphers. Consequently, our approach holds substantial potential for applications in sectors necessitating such high levels of security.

The structure of the paper is as follows: Section II contains information on related works, while Section III presents an algorithm for generating new XOR tables. Additionally, it provides proof that this new XOR operation maintains the independent, co-probability distribution of the random key within the ciphertext. In Section IV, a security analysis and evaluation of the altered AES block cipher using random NIST tests is provided. Lastly, Section V concludes the paper.

II. RELATED WORKS

AES is a type of block cipher known as an SPN, which comprises three distinct layers: a layer for substitution, a layer for diffusion, and a layer for adding the encryption key. Numerous research studies have been conducted to enhance the strength of the AES block cipher by making it more dynamic. Many of these studies have focused on dynamically modifying the substitution transformation of AES, as seen in [27], [28], [29], [30], [31], and [32]. In [27], the authors presented a novel algorithm for generating dynamic S-boxes by permuting the original AES S-box. The dynamic S-box is dependent on the encryption key and employs an irregular polynomial and an affine constant. With every added key bit, a new S-box with permuted values is generated, thereby increasing the algorithm's complexity. In [28], the authors proposed an algorithm for constructing key-dependent S-boxes with desirable algebraic properties, such as SAC, non-linearity, and BIC. In [29], the authors introduced an algorithm for generating a dynamic S-box for AES. Unlike other methods, this algorithm continuously varies independent of the encryption key and relies on the timestamp present in all digital systems. The crucial advantage of this approach is that it alters the ciphertext while keeping the encryption key constant, ensuring that identical data will produce different encryption results. In [30], the authors introduced a method for generating key-dependent S-boxes based on a dynamic approach. The key-dependent S-box was evaluated experimentally based on properties such as balanced output, BIC, SAC, non-linearity, and differential and linear approximation probabilities. Moreover, an alternative method for generating key-dependent S-boxes for AES was presented in [31], which involves creating a new permutation for the S-box using a pseudo-key expansion algorithm. Additionally, in [32] Murphy et al. presented a method for differential cryptanalysis of key-dependent S-boxes, outlining techniques for conducting cryptanalysis using such S-boxes.

Some works have also focused on introducing dynamism to the diffusion layer [33], [34], [35], [36] of AES. For example, one proposal in [33] involves constructing a key-dependent diffusion layer using scalar multiplication and direct exponentiation. In [34], the authors suggested a key-dependent MixColumn transformation from the AES MDS matrix, utilizing scalar multiplication of the matrix's rows and an additional m-bit key. Shamsabad and Dehnavi presented some $n \times n$ binary matrices in [35], which can be used to make dynamic AES-like matrices and recursive MDS matrices. Additionally, a direct exponentiation fast calculation algorithm was presented in [36] to enhance the execution speed of dynamic block ciphers at the diffusion layer, thereby contributing to enlarging the overall security of the block cipher.

Other research has explored the dynamic adaptation of both substitution and diffusion layers in AES, as demonstrated in [37], [38], and [39]. In [37], the authors introduced a dynamic block cipher called P-AES, where the values of AES parameters are modified for each key. Specifically, the SubBytes, ShiftRows, and MixColumns transformations are animated based on the key, resulting in different behavior with each new key. The P-AES algorithm has been proven secure against differential and linear attacks. In [38], an image encryption algorithm based on symmetric cryptography was proposed, with MixColumns, ShiftRows, and SubByte transformations that are animated based on the key. The P-AES algorithm has been proven to provide authenticity, integrity, and confidentiality. In [39], Xu et al. also presented dynamic S-boxes and new MixColumn matrices that maintain good cryptographic properties for creating dynamic AES.

In addition to the above research directions, there are studies currently underway that aim to create dynamic block ciphers based on DNA and chaotic maps, as demonstrated in [40], [41], and [42]. In [40], a 4D-hyper system using chaotic maps and DNA calculating are employed to produce a dynamic S-box. The system generates numbers to create a hexadecimal form number, which is then combined with DNA encoding and addition, subtraction, and exclusive-or operations to form the dynamic S-box. The new S-box is evaluated against balanced, bit independence, strict avalanche, linearity approximation probability, and differential approximation probability criteria. In [41], the authors introduced two efficient SPN block ciphers using a chaotic system, with dynamic S-boxes created based on keys in the chaos maps. In [42], the authors introduced an algorithm for designing key-dependent S-boxes ($n \times n$) using chaotic time series of logical mapping, using different implementation possibilities in chaotic mapping to provide low computational complexity.

III. GENERATING NEW KEY-DEPENDENT XOR TABLES BASED ON A PSEUDO-RANDOM NUMBER GENERATOR AND THE FISHER-YATES SHUFFLE ALGORITHM

In this section, we present the essential characteristics of an XOR table that are required to ensure the successful decryption and encryption of the block cipher, which was not fully demonstrated in [43] and [44]. Subsequently, we will introduce an algorithm for generating new key-dependent XOR tables based on a pseudo-random number generator and the Fisher-Yates shuffle algorithm. This algorithm makes it possible to produce a vast number of new, key-dependent XOR tables.

A. THE ESSENTIAL CHARACTERISTICS OF AN XOR TABLE

Given that the algorithm under consideration is AES, where transformations are carried out over the $GF(2^8)$ field, every element within $GF(2^8)$ is handled as a byte comprising two halves, and each half is composed of four bits. Thus, to guarantee the proper functioning of the decryption and encryption process, we give the following three properties that an XOR table performed on 4-bit symbols must satisfy, and we name them as follows.

- *XOR property 1:* The XOR table exhibits symmetry across the main diagonal, which implies that $xXORy = yXORx$.
- *XOR property 2:* Each row and column of the XOR table must include the numbers 0 to 15 exactly once without repetition (i.e., it must be a permutation of the numbers 0 to 15).
- *XOR property 3:* For any given elements x, y and z in the table such that $xXORy = z$, the table must satisfy the following conditions simultaneously: $yXORz = x$, and $xXORz = y$.

B. ALGORITHM FOR GENERATING TWO NEW 4-BIT KEY-DEPENDENT XOR TABLES

In this section, we propose an algorithm that can generate two 4-bit XOR tables from an initial secret key.

Suppose that the receiver and sender need to establish a secret key a^* prior to further communication.

Remark 1 (The accuracy of the newly proposed key-dependent XOR table): The initial XOR table of AES fulfills the criteria of an XOR table (Section III-A). In addition, with Algorithm 1, all elements of the table are replaced simultaneously, resulting in two new XOR tables that retain all three required properties of an XOR table. Through experimentation, we verified that these new XOR tables also satisfy these essential characteristics, although it is important to note that the XOR operation performed in this table is not the standard bit Exclusive Or operation.

It is noteworthy that the aforementioned algorithms are capable of producing new XOR tables which rely on the initial secret key.

Now, we demonstrate that the new XOR tables generated by Algorithm 1 fully satisfy the three essential properties of an XOR table, as elucidated in Section III-A. We prove the correctness of the XOR tables obtained from Algorithm 1 through the following proposition.

Proposition 1 (Demonstrating the correctness of the new XOR tables): The XOR tables obtained from Algorithm 1 are XOR tables that satisfy all the necessary properties of an XOR table.

Algorithm 1 Generating two XOR 4-bit tables based on the secret key

INPUT: A randomly generated key K with a length of 128 bits.
OUTPUT: Two new 4-bit XOR tables.

Step 1: Select a 128-bit block consisting of 1s, indicated as $P_0 = \{11\ldots1\}$. Apply the AES algorithm to encrypt P_0 using the key K, obtaining the 128-bit sequence P_1.
Step 2: Extract the initial 64 bits from the P_1 sequence, then break down these 64 bits into 16 segments, each containing 4 bits. Let $a_i (0 \leq i \leq 15)$ represent the decimal value associated with the i-th 4-bit segment. Organize these elements into a collection of 16 elements, symbolized as $\mathcal{A} = \{a_0, a_1 \ldots, a_{15}\}$ where $0 \leq a_i \leq 15$. Initialize the corresponding index array $I_1 = \{0, 1, 2, \cdots, 15\}$.
Step 3: Perform a similar procedure as in Step 2 with the last 64 bits of the P_1 sequence. Obtain a set $\mathcal{B} = \{b_0, b_1 \ldots, b_{15}$. Initialize the corresponding index array $I_2 = \{0, 1, 2, \cdots, 15\}$.
Step 4: Sort the elements of the set $\mathcal{A} = \{a_0, a_1 \ldots, a_{15}$ in increasing order based on their values. In case of identical elements ($a_i = a_j$), prioritize the element with the lower index. The outcome is a fresh set, labeled as $\acute{\mathcal{A}}$. Applying the sorting rule to the index array I_1 yields a new index array I_1' (corresponding to the set $\acute{\mathcal{A}}$).
Step 5: Perform a similar process as in Step 4 with the set $\mathcal{B}$, resulting in a new set denoted as $\acute{\mathcal{B}}$. Applying the sorting rule to the index array I_2 yields a new index array I_2' (corresponding to the set $\acute{\mathcal{B}}$).
Step 6: By utilizing the initial XOR table (in Table 8), a new XOR table M with a dimension of 16×16 can be formed in the following manner:
+ Substitute every entry in the initial XOR table with their respective elements in the array $I_1' = [u_0, u_1 \ldots, u_{15}]$, denoting the replacement of the value $i(0 \leq i \leq 15)$ in the initial XOR table with the element r_i of the array I_1'. Executing this substitution for the entire initial XOR table results in a new XOR table M_1. Specifically, all elements in the original XOR table, including the elements of the header row and header column, will be replaced according to the following 1-1 mapping.

$$0 \rightarrow u_0; 1 \rightarrow u_1; 2 \rightarrow u_2; \ldots; 15 \rightarrow u_{15}$$

+ Rearrange the columns and rows in the new XOR table M_1 to ensure that the first column and first row form an increasing sequence ranging from 0 to 15. This action yields a different XOR table, referred to as M.
Step 7: Perform the same process as in Step 8, but replace the cells in the original XOR table with the corresponding elements from the array $I_2' = [\acute{u}_0, \acute{u}_1 \ldots, \acute{u}_{15}]$, resulting in a new XOR table denoted as N.

TABLE 1. The initial XOR table using 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

TABLE 2. The new XOR table from permutation generated in algorithm 1.

	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	u_{11}	u_{12}	u_{13}	u_{14}	u_{15}
u_0	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	u_{11}	u_{12}	u_{13}	u_{14}	u_{15}
u_1	u_1	u_0	u_3	u_2	u_5	u_4	u_7	u_6	u_9	u_8	u_{11}	u_{10}	u_{13}	u_{12}	u_{15}	u_{14}
u_2	u_2	u_3	u_0	u_1	u_6	u_7	u_4	u_5	u_{10}	u_{11}	u_8	u_9	u_{14}	u_{15}	u_{12}	u_{13}
u_3	u_3	u_2	u_1	u_0	u_7	u_6	u_5	u_4	u_{11}	u_{10}	u_9	u_8	u_{15}	u_{14}	u_{13}	u_{12}
u_4	u_4	u_5	u_6	u_7	u_0	u_1	u_2	u_3	u_{12}	u_{13}	u_{14}	u_{15}	u_8	u_9	u_{10}	u_{11}
u_5	u_5	u_4	u_7	u_6	u_1	u_0	u_3	u_2	u_{13}	u_{12}	u_{15}	u_{14}	u_9	u_8	u_{11}	u_{10}
u_6	u_6	u_7	u_4	u_5	u_2	u_3	u_0	u_1	u_{14}	u_{15}	u_{12}	u_{13}	u_{10}	u_{11}	u_8	u_9
u_7	u_7	u_6	u_5	u_4	u_3	u_2	u_1	u_0	u_{15}	u_{14}	u_{13}	u_{12}	u_{11}	u_{10}	u_9	u_8
u_8	u_8	u_9	u_{10}	u_{11}	u_{12}	u_{13}	u_{14}	u_{15}	u_0	u_1	u_2	u_3	u_4	u_5	u_6	u_7
u_9	u_9	u_8	u_{11}	u_{10}	u_{13}	u_{12}	u_{15}	u_{14}	u_1	u_0	u_3	u_2	u_5	u_4	u_7	u_6
u_{10}	u_{10}	u_{11}	u_8	u_9	u_{14}	u_{15}	u_{12}	u_{13}	u_2	u_3	u_0	u_1	u_6	u_7	u_4	u_5
u_{11}	u_{11}	u_{10}	u_9	u_8	u_{15}	u_{14}	u_{13}	u_{12}	u_3	u_2	u_1	u_0	u_7	u_6	u_5	u_4
u_{12}	u_{12}	u_{13}	u_{14}	u_{15}	u_8	u_9	u_{10}	u_{11}	u_4	u_5	u_6	u_7	u_0	u_1	u_2	u_3
u_{13}	u_{13}	u_{12}	u_{15}	u_{14}	u_9	u_8	u_{11}	u_{10}	u_5	u_4	u_7	u_6	u_1	u_0	u_3	u_2
u_{14}	u_{14}	u_{15}	u_{12}	u_{13}	u_{10}	u_{11}	u_{12}	u_{13}	u_6	u_7	u_4	u_5	u_2	u_3	u_0	u_1
u_{15}	u_{15}	u_{14}	u_{13}	u_{12}	u_{11}	u_{10}	u_9	u_8	u_7	u_6	u_5	u_4	u_3	u_2	u_1	u_0

Proof:

From array $I_1' = \{u_0, u_1 \ldots, u_{15}\}$, (where $0 \leq u_i \leq 15$) obtained in Algorithm 1, using the substitution $0 \rightarrow u_0$; $1 \rightarrow u_1$; $2 \rightarrow u_2$; ...; $15 \rightarrow u_{15}$, we obtain the firt new XOR table in the form as shown in Table 2.

Since the simultaneous replacement of elements is a 1-1 mapping, for any arbitrary values $i, j, k (0 \leq i, j, k \leq 15)$ in the original XOR table (Table 1), their corresponding values in Table 2 are u_i, u_j, u_k.

For every $0 \leq i, j, k \leq 15$, according to the original XOR table, we have:

- $iXORj = jXORi$
- Each column and row of the original XOR table consists of a permutation of numbers from 0 to 15.
- If $iXORj = k$, then $iXORk = j$ and $jXORk = i$

From the substitution $0 \rightarrow u_0$; $1 \rightarrow u_1$; $2 \rightarrow u_2$; ...; $15 \rightarrow u_{15}$, and the above properties of the original XOR table, we can deduce that Table 2 also possesses the following properties.

- $u_iXORu_j = u_jXORu_i$ (*XOR property 1*).
- Since the substitution mapping is 1-1, each column and row of Table 2 consists of a permutation of numbers from 0 to 15 (*XOR property 2*).
- If $u_iXORu_j = u_k$ then $u_iXORu_k = u_j$ and $u_jXORu_k = u_i$ (*XOR property 3*).

Next, rearranging the positions of rows and columns in the XOR table does not alter the values of the cells in the XOR table; therefore, Table 2 satisfies all three essential properties of an XOR table as outlined in Section III-A.

Based on similar reasoning applied to the second XOR table generated from array I_2', we also demonstrate that the second XOR table satisfies all three essential properties of an XOR table. ■

Remark 2: To be more specific, the array U comprises of permutations of 16 distinct elements ranging from 0 to 15, which are derived from the initial random key. Given that there exist 16! possible permutations of these 16 elements, Algorithm 1 enables us to generate up to 16! distinct key-dependent XOR tables.

Note that, the new XOR operation in the recently generated XOR table as $\overline{\oplus}$.

Suppose we have random, independent variables δ_1, δ_2, ..., δ_n and their numerical values belong to the set $\mathfrak{R} = \{0, 1, \ldots, 15\}$ with equal probability of occurrence, which is 16^{-1}. Whenever these variables ε_1, ε_2,..., ε_n take on a specific value $Key = (k_1, k_2, \ldots, k_n)$, we refer to Key as an optimal random number key of length n.

Let P_1 represent the probability distribution across the plaintext space $\mathcal{P}$, P_2 denote the probability distribution throughout the ciphertext space $\mathcal{C}$, and P_3 represent the probability distribution across the key space $\mathcal{K}$.

Suppose $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathfrak{R}$.

We have the following proposition:

Proposition 2: Let $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$ denote plaintexts and corresponding ciphertexts that assume values from set $\mathfrak{R}$, respectively. Additionally, $\delta_1, \delta_2, \ldots, \delta_n$ are independently generated random variables with a uniform distribution over set $\mathfrak{R}$, and assume values $k_1, k_2, \ldots, k_n$, which are employed in the new XOR activity to generate $y_1, y_2, \ldots, y_n$.

$$y_i = x_i \overline{\oplus} k_i, \quad \text{for} \quad i = 1, 2, \ldots, 15.$$

Then it follows that: $P_2(y_1, y_2, \ldots, y_n) = 16^{-n}$.

Proof:

Due to the independently random variables $\delta_1, \delta_2, \ldots, \delta_n$ having the same probability and taking the values $k_1, k_2, \ldots, k_n$ respectively, as well as the characteristics of the new XOR, it follows that if $y_i = x_i \overline{\oplus} k_i$, then $x_i = y_i \overline{\oplus} k_i$, and if $y_j = x_j \overline{\oplus} k_j$, then $x_j = y_j \overline{\oplus} k_j$, so:

$$\begin{aligned} P_2(y_i, y_j) &= \sum\nolimits_{k_i, k_j \in \mathcal{K}} P_3(k_i, k_j) P_1(y_i \overline{\oplus} k_i, y_j \overline{\oplus} k_j) \\ &= \sum\nolimits_{k_i, k_j \in \mathcal{K}} P_3(k_i) P_3(k_j) P_1(y_i \overline{\oplus} k_i, y_j \overline{\oplus} k_j) \\ &= \frac{1}{16^2} \sum\nolimits_{k_i, k_j \in \mathcal{K}} P_1(y_i \overline{\oplus} k_i, y_j \overline{\oplus} k_j) = \frac{1}{16^2}. \end{aligned} \quad (1)$$

Besides, for any $y \in \mathfrak{R}$, and based on the characteristics of the new XOR, $y = x\overline{\oplus}k$, then $x = y\overline{\oplus}k$. Thus, we can conclude that:

$$P_2\left(y\right) = \sum\nolimits_{k\in\mathfrak{R}} P_3(k)P_1\left(x\right) = \sum\nolimits_{k\in\mathfrak{R}} \frac{1}{16}P_1\left(y\overline{\oplus}k\right)$$
$$= \frac{1}{16}\sum\nolimits_{k\in\mathfrak{R}} P_1\left(y\overline{\oplus}k\right) = \frac{1}{16} \quad (2)$$

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

$$P_2(y_i,y_j) = P_2(y_i)\cdot P_2(y_j) = \frac{1}{16^2}. \quad (3)$$

With equivalent evidence, it follows that.

$$P_2(y_1,y_2,\ldots,y_n) = P_2(y_1)\cdot P_2(y_2)\cdot\ldots\cdot P_n(y_n) = \frac{1}{16^n}.$$

■

Remark 3: Based on Proposition 2, it is apparent that by conducting the new XOR operation between a random key and a plaintext then the resulting ciphertext will exhibit comparable characteristics to that of the random key, including uniform probability and independent distribution. This is critical since the ciphertext generated by the XOR operation may relinquish the properties of the original plaintext, while the uniform probability and independent distribution of the ciphertext make it challenging to perform cryptanalysis. Additionally, the XOR operation is executed before the AES S-box to guarantee that the properties of uniform probability and independent distribution are preserved, thereby enabling the application of differential and linear probability definitions to the S-box.

Compare our method with those in [43] and [44]

Hereinafter, we present certain contrasts between the techniques employed to produce XOR tables in [43] and [44] and our own approach.

- While [43], [44] enumerated three characteristics of the new XOR operation, they did not demonstrated that these features ensure accurate decryption. This research, however, has established the mandatory attributes of an XOR table (in section III-A) and verified the accuracy of the decryption process.
- The authors of [43] fashioned a solitary private XOR table, and [44] described the creation of two novel XOR tables; however, this our study has the capacity to generate 16! key-dependent XOR tables contingent upon an initial secret key. The availability of a significant number of key-dependent XOR tables complicates cryptanalysis substantially, thereby boosting the AES security.
- References [43] and [44] only demonstrated the procedure for generating XOR tables, but did not elaborate on the theoretical significance of the XOR operation, including the new XOR operation. This paper has addressed that gap through Proposition 1, Proposition 2, expounding upon the theoretical importance of XOR as well as our novel XOR.

TABLE 3. Compare our dynamic method with other one.

Criteria	Our method	Dynamic methods in [43], [44]	Dynamic methods at the substitution layer [27], [28], [29], [30], [31], [32],	Dynamic methods at the diffusion layer [33], [34], [35], [36]	Dynamic methods at both substitution and diffusion layers [37], [38], [39]	Dynamic methods based on DNA and Chaotic maps [40], [41], [42], [43].
Dynamic components	XOR table	XOR table	S-box	MDS matrix	S-box and MDS matrix and/or Shiftrow	S-box or MDS matrix
Dynamic method depends on	Key	Key	Key or time [20]	Key	Key	Key
Some functions/transformations used	Pseudo-random numbers generator	Chaotic map	Permutation; shift and permute rows/columns; pseudo-key expansion algorithm;	Scalar multiplication; direct exponentiation; binary matrix	Use many different transformations	Chaotic maps and DNA
Advantages	Increases block cipher security by using dynamic XOR	Increases block cipher security by using dynamic XOR	Increases block cipher security by using dynamic S-box	Increases block cipher security by using dynamic MDS matrix	Increases block cipher security by using dynamic components	Increases block cipher security by using dynamic components
Disadvantages	Takes more memory space to store the new XOR table	Takes more memory space to store the new XOR table	It is necessary to reconstruct the lookup tables for the new s-box and the shared lookup table for the transformations.	It is necessary to reconstruct the lookup tables for the new MDS matrix and the shared lookup table for the transformations.	The shared lookup table for the transformations need to be reconstructed.	The shared lookup table for the transformations need to be reconstructed.

Compare our dynamic method with other ones

To have a more comprehensive view, we compare our proposed dynamic method in this paper with other ones in literature. Table 3 shows this comparison based on a few basic criteria.

In general, each dynamic method operates differently, but the commonality among these dynamic methods is their ability to "obfuscate" a certain component of the block cipher, making it significantly more challenging for cryptanalysis compared to a static block cipher.

C. UTILIZING KEY-DEPENDENT FOR TABLES TO ALTER THE AES BLOCK CIPHER

In our proposal in this paper, we only replace the XOR operation in the AddRoundKey transformation in AES with the XOR operation from a new XOR table; other transformations in AES (key schedule, MDS operation XOR, etc.) will still use the regular bitwise XOR.

By inputting a secret key K into Algorithm 1, we can generate two new XOR tables, designated as M and N, which are dependent on the above keys. These tables will be used alternatively during the creation of the altered AES algorithm, with table M utilized during even rounds and table N utilized during odd rounds.

We carry out tests utilizing K as a 128-bit authentic random key sourced from the link: https://www.random.org/bytes/. Specifically, two genuine random sequences (in hexadecimal format) are acquired in the following manner:

K = ce a2 72 31 c6 8d ba d2 32 1b fb 60 7f 6d 29 7e

Performing Algorithm 1 with this key results in two arrays as shown below.

$$I_1' = \{14, 11, 1, 3, 10, 5, 15, 13, 4, 7, 6, 8, 2, 12, 9, 0,$$
$$I_2' = \{4, 0, 7, 8, 9, 14, 13, 12, 15, 3, 1, 11, 2, 5, 6, 10$$

Two new XOR tables, M_1 and N_1, can be obtained as presented in Table 4 and Table 5, respectively.

Afterwards, reposition the columns and rows in tables M_1 and N_1 such that the initial row and column in each table form an increasing sequence from 0 to 15. This will produce two new key-dependent XOR tables, M and N, which are illustrated in Table 6 and Table 7, respectively.

Subsequently, for the purpose of encryption and decryption, we utilize XOR table M for even AES rounds and XOR table N for odd AES rounds.

In the experiment, we can use K as the encryption key of AES-128 as the input of Algorithm 1. Actually, in Algorithm 1, we do not directly use the key K to generate the XOR tables but rather a 128-bit string P_1 derived from K. This helps limit cryptanalysts from exploiting XOR tables to deduce key bits of K.

IV. RESULTS FOR EXPERIMENT AND ANALYSIS

The modified AES algorithm is implemented in C++. In this study, experiments were performed on the 128-bit key version of AES, which is the most widely recognized version.

NIST tests are performed for the modified AES algorithm on 349,632 different sequences for LW128 and HW128 data sets, and 1,048,576 different sequences for AV1 and Rot data sets, where every sequence is 128 bits. Some statistical tests for short sequences include the Test for the Longest Run of Ones, Runs Test, Approximate Entropy Test, Frequency Test, Serial Test, and CuSum Test. Our experiments are implemented on a Windows 10 64-bit computer, Intel Core i5 2430M (Bus 2500, Cache 3MB, 2 × 2.4GHz Turbo Boost 3.0GHz), RAM 2GB DDR3 1333MHz, Intel HD Graphics 3000, Chipset Intel HM67 Express, and NVidia Geforce GT 520M (1GB VRAM).

A. SECURITY ANALYSIS

Block ciphers are a popular type of cryptographic algorithm used in various fields nowadays, but they are vulnerable to attacks, with linear [5], [11], [48], [49] and differential attacks [11], [25], [49], [50], [51] being the two most potent types. Both of these attacks require cryptanalysts to gather a significant amount of plaintext/ciphertext pairs to conduct the attacks. Keliher [52] provided data complexity formulas for a linear attack (using Matsui's Algorithm 2 [49]), which

TABLE 4. The new XOR Table M_1.

M_1	14	11	1	3	10	5	15	13	4	7	6	8	2	12	9	0
14	14	11	1	3	10	5	15	13	4	7	6	8	2	12	9	0
11	11	14	3	1	5	10	13	15	7	4	8	6	12	2	0	9
1	1	3	14	11	15	13	10	5	6	8	4	7	9	0	2	12
3	3	1	11	14	13	15	5	10	8	6	7	4	0	9	12	2
10	10	5	15	13	14	11	1	3	2	12	9	0	4	7	6	8
5	5	10	13	15	11	14	3	1	12	2	0	9	7	4	8	6
15	15	13	10	5	1	3	14	11	9	0	2	12	6	8	4	7
13	13	15	5	10	3	1	11	14	0	9	12	2	8	6	7	4
4	4	7	6	8	2	12	9	0	14	11	1	3	10	5	15	13
7	7	4	8	6	12	2	0	9	11	14	3	1	5	10	13	15
6	6	8	4	7	9	0	2	12	1	3	14	11	15	13	10	5
8	8	6	7	4	0	9	12	2	3	1	11	14	13	15	5	10
2	2	12	9	0	4	7	6	8	10	5	15	13	14	11	1	3
12	12	2	0	9	7	4	8	6	5	10	13	15	11	14	3	1
9	9	0	2	12	6	8	4	7	15	13	10	5	1	3	14	11
0	0	9	12	2	8	6	7	4	13	15	5	10	3	1	11	14

TABLE 5. The new XOR Table N_1.

N_1	4	0	7	8	9	14	13	12	15	3	1	11	2	5	6	10
4	4	0	7	8	9	14	13	12	15	3	1	11	2	5	6	10
0	0	4	8	7	14	9	12	13	3	15	11	1	5	2	10	6
7	7	8	4	0	13	12	9	14	1	11	15	3	6	10	2	5
8	8	7	0	4	12	13	14	9	11	1	3	15	10	6	5	2
9	9	14	13	12	4	0	7	8	2	5	6	10	15	3	1	11
14	14	9	12	13	0	4	8	7	5	2	10	6	3	15	11	1
13	13	12	9	14	7	8	4	0	6	10	2	5	1	11	15	3
12	12	13	14	9	8	7	0	4	10	6	5	2	11	1	3	15
15	15	3	1	11	2	5	6	10	4	0	7	8	9	14	13	12
3	3	15	11	1	5	2	10	6	0	4	8	7	14	9	12	13
1	1	11	15	3	6	10	2	5	7	8	4	0	13	12	9	14
11	11	1	3	15	10	6	5	2	8	7	0	4	12	13	14	9
2	2	5	6	10	15	3	1	11	9	14	13	12	4	0	7	8
5	5	2	10	6	3	15	11	1	14	9	12	13	0	4	8	7
6	6	10	2	5	1	11	15	3	13	12	9	14	7	8	4	0
10	10	6	5	2	11	1	3	15	12	13	14	9	8	7	0	4

TABLE 6. The XOR table M used for even rounds.

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

determine the minimum amount of data required for the attack to be successful, as follows:

$$N_L = \frac{c}{EDP^{[1..T]}(a, b)} \tag{4}$$

TABLE 7. **The XOR table *N* used for odd rounds.**

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

Keliher also gave the data complexity for a differential attack, as follows:

$$N_D = \frac{1}{EDP^{[1..T]}(\Delta X, \Delta Y)} \tag{5}$$

where, $EDP^{[1..T]}(\Delta X, \Delta Y)$ is the probability of the average differential probability over $1..T$ rounds with input and output differences are ΔX and ΔY respectively; $ELP^{[1..T]}(a, b)$ is the average linear probability over $1..T$ rounds with input and output mask a and b respectively (see more in [52]).

For example, to successfully perform the linear attack on the DES block cipher, the cryptanalysts must collect about 2^{47} plaintext/ciphertext pairs [48]. As such, the data complexity to perform these attacks is very large.

To carry out linear and differential attacks on block ciphers, including the AES algorithm, cryptanalysts must possess knowledge of all the algorithm's components, such as S-box, ShiftRow, Mixcolumn matrix, and Addroundkey. Specifically for AES, the cryptanalyst needs to know these transformations precisely. However, the secret key remains unknown for encryption/decryption. Cryptanalysts should thus gather an ample amount of plaintext/ciphertext pairs to conduct these attacks and follow the typical methods described in [11], [25], [48], and [52].

However, when we animate the AES block cipher, and specifically in this paper, we animate the Addroundkey transformation. The cryptanalysts then do not know which XOR table we used for dynamic AES, so they must search to find the XOR table used for dynamic AES. After each attempt of an XOR table, the cryptanalysts must collect as many plaintext/ciphertext pairs as for attacking AES. This search will make it very difficult for cryptanalysts because of the large number of plaintext/ciphertext pairs. Algorithm 1 generates an extensive number of new XOR tables (16! to be precise), making it particularly challenging for cryptanalysts to perform an exhaustive search.

Note that when applied to AES, the total number of dynamic XOR tables that the attacker needs to iterate over is $(16!)^2$ because we use two dynamically generated XOR tables alternately in consecutive rounds. The attacker does not know which XOR table will be used in each round.

Proposition 2 demonstrates that the new XOR operation guarantees the output to exhibit the same characteristics as a random key sequence, namely uniform probability and independent distribution. This is significant because the ciphertext may lose the structure of the plaintext after the XOR operation, rendering it challenging for cryptanalysis. Additionally, the XOR operation is executed before the AES S-box, with the intention of preserving the uniform probability and independent distribution properties. Ensuring the preservation of the uniform probability and independent distribution properties is crucial for applying the definition of differential and linear probabilities to the S-box. In addition, in [1] Shannon gave a formula for evaluating the unicity distance of a cryptosystem as follows:

$$U = \frac{H(K)}{D} \tag{6}$$

The unicity distance of a block cipher is the minimum number of ciphertext characters required for cryptanalysts to uniquely deduce the secret key, given that $H(K)$ represents the entropy of the key space and D denotes the redundancy of the language. A larger unicity distance indicates a more secure block cipher.

Applying to the dynamic AES algorithm proposed in this paper, it can be seen that, since we animate the Addroundkey operation with dynamic XOR tables, in this case the key space increases, not only K but also extra space of dynamic XOR tables. As a result, the numerator of (6) experiences a significant increase, leading to a greater unicity distance and ultimately enhancing the security of the dynamic AES block cipher.

Now, to provide further persuasion, we analyze in detail how the security of the dynamic AES block cipher in this paper is enhanced compared to the AES block cipher. We focus our analysis on two of the strongest attacks on block ciphers: linear attack and differential attack.

1) ANALYZING LINEAR CYPTANALYSIS ON THE AES AND DYNAMIC AES

a: ANALYZING LINEAR CYPTANALYSIS ON THE AES

Linear cryptanalysis aims to exploit the frequent instances of linear equations involving bits of the original message, bits in the "ciphertext" (are the input data bits of the last round), and bits of the subkey. This method constitutes a known plaintext assault.

The fundamental idea of linear cryptanalysis involves representing the function of a segment within the cipher through a linear expression, with the term "linearity" signifying a bit-wise operation using mod-2 (specifically, the exclusive-OR represented by "$\oplus$"). This linear expression takes the following structure:

$$X_{i_1} \oplus X_{i_2} \oplus \ldots \oplus X_{i_a} \oplus Y_{j_1} \oplus Y_{j_2} \oplus \ldots Y_{j_b} = 0 \tag{7}$$

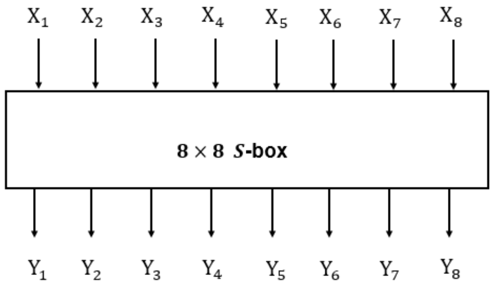


FIGURE 2. A 8 × 8 S-box mapping.

where X_i denotes the i-th bit within the input $X = [X_1, X_2, \ldots]$, and Y_j signifies the j-th bit of the output $Y = [Y_1, Y_2, \ldots]$. Please observe that X is the plaintext input of the block cipher, and Y serves as the output from the preceding round just before the final one, or alternatively expressed, it functions as the input for the last round.

Let p be the probability of the occurrence of the linear expression (7) with randomly chosen plaintexts, and ε be the linear probability bias of (7). In that case, $p - \frac{1}{2} = \varepsilon$.

The strategy in linear cryptanalysis involves identifying patterns similar to (7) that exhibit either a high or low likelihood of appearing, but what matters is that it has a high bias.

Matsui [48] presented the following Piling-Up lemma:

For a set of n binary variables that are independent and random, $X_1, X_2, \ldots, X_n$,

$$\Pr(X_1 \oplus X_2 \oplus \ldots \oplus X_n = 0) = \frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \varepsilon_i$$

where $\varepsilon_{1,2,\ldots,n} = 2^{n-1} \prod_{i=1}^{n} \varepsilon_i$ is the bias of $X_1 \oplus X_2 \oplus \ldots \oplus X_n = 0$.

To execute a linear attack on a block cipher, the attacker must necessarily find a high probability linear expression for the entire block cipher, with a significant probability bias. In conducting linear cryptanalysis, the attacker will run a direct search algorithm to discover the linear expression or the best linear approximation with the highest probability.

To find a linear expression for the entire n-round block cipher in the form of (7), we present Table 8 describing the general steps to be carried out. These steps follow the Algorithm 2 outlined by Matsui [48], [49].

Figure 2 illustrates the input and output of a 8 × 8 S-box.

As we know, the AES block cipher has n rounds ($n = 10$, 12, or 14), with a 128-bit input block and employs 8 × 8 S-boxes.

To align with the above analysis regarding the structure of a general SPN block cipher, here, we examine each round of AES, which consists of four transformations (from round 1 to round $n - 1$): AddRoundKey, SubByte, ShiftRow, and MixColumn. The last round (round n) of AES comprises Four transformations: AddRoundKey, SubByte, ShiftRow, and AddRoundKey.

TABLE 8. Steps to construct a linear expression for the SPN block cipher.

Construct a linear approximation for $n-1$ rounds and a corresponding linear expression for the n-round SPN block cipher.
Step 1: Construct the linear approximation table for the S-box. - For a static SPN block cipher, the S-box component is public, so the linear approximation table can be constructed with all possible input masks and output masks. - From this table, the probability bias of all possible linear approximations of the S-box can be calculated.
Step 2: Select specific S-boxes at each round (these S-boxes must be active ones, meaning they have non-zero input masks and output masks), and choose specific linear approximations from the linear approximation table in Step 1 to associate with the selected S-boxes. - The selected S-boxes range from round 1 to round n. - Note that the S-boxes are not chosen randomly but depend on the permutation layer of the SPN block cipher, meaning the output bits of the selected S-boxes in round i must be the input bits of one of the selected S-boxes in round $i + 1$. - Each of these linear approximations have corresponding probability and probability bias (calculated from the linear approximation table in Step 1).
Step 3: Find the linear expression for the entire SPN block cipher and calculate its probability bias. - Combine the corresponding linear approximations to the selected S-boxes at each round from Step 2, along with their respective probabilities and probability biases, we can obtain a linear approximation for the $n-1$ rounds and a corresponding linear expression for the n-round SPN block cipher. From this, probabilities and biases of the linear expression can be calculated. - Note that the linear approximations of the S-boxes are assumed to be independent. Consequently, the linear probability and probability bias of the linear expression for entire block cipher can be calculated based on the Piling-Up Lemma.

Note that in the final round, the ShiftRow operation takes place. This operation on a 4 × 4 byte state is as follows: the first row of the state remains unchanged, the second row left-rotates by one byte, the third row left-rotates by two bytes, and the fourth row left-rotates by three bytes.

Thus, the inverse operation of the ShiftRow, denoted as $ShiftRow^{-1}$ will perform the reverse. The first row of the state remains unchanged, the second row right-rotates by one byte, the third row right-rotates by two bytes, and the fourth row right-rotates by three bytes.

Therefore, the ShiftRow operation can be simulated as a byte permutation as follows.

X	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$ShiftRow(X)$	1	2	3	4	8	5	6	7	11	12	9	10	14	15	16	13

The $ShiftRow^{-1}$ operation can also be represented as a byte permutation as follows.

X	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$ShiftRow^{-1}(X)$	1	2	3	4	6	7	8	5	11	12	9	10	16	13	14	15

Assume K is the secret key of AES, and the round keys derived from K are denoted as $K_1, K_2, \ldots, K_n, K_{n+1}$.

Let P and C be the corresponding plaintext and ciphertext of the AES block cipher.

Let U_i, V_i represent the inputs and outputs of the S-boxes at round i. Denote $U_{i,j}$ as the j-th bit of U_i, where $1 \le i \le n$, $1 \le j \le 128$.

Note that the linear expression of AES obtained only relates to the plaintext input, the input bits of the final round (round n), and the key bits of the subkeys $K_1, K_2, \ldots, K_n$.

After obtaining a linear approximation for the $n-1$ rounds and the corresponding linear expression with a sufficiently large probability bias for the entire AES, the attacker proceeds to extract the key bits of the last subkey in the final round of AES (key K_{n+1}). Naturally, for this cryptanalysis to succeed, a considerable number of plaintext/ciphertext pairs must be collected.

Assuming the attacker collected 100,000 plaintext/cipher text pairs and found both a linear approximation for $n-1$ rounds and a corresponding linear expression with a linear probability p and a sufficiently large probability bias ε. Suppose in this linear expression, it includes the inputs of two S-boxes in the final round $S_{n,2}$ and $S_{n,4}$ (the 2nd S-box and 4th S-box in round n). Corresponding to these two S-boxes are two related key bytes from the subkey K_{n+1}.

The notation for these two subkey bytes consists of a 16-bit key $[K_{n+1,9}K_{n+1,10}\ldots K_{n+1,16}, K_{n+1,25}K_{n+1,26}\ldots K_{n+1,32}]$. There are a total of 2^{16} possible combinations for these 16 key bits. For each set of 16 key bits, there will be a corresponding count value $count_i(1 \leq i \leq 2^{16})$, initialized to 0.

From the obtained linear expression, temporarily excluding the key bits (as the sum of key bits is fixed and can only be 0 or 1), the linear expression becomes the XOR sum of the plaintext bits and the input data bits for round n. The linear expression has the following form.

$$U_{n,i} \oplus U_{n,j} \oplus \ldots \oplus U_{n,t} \oplus (P_l \oplus P_h \oplus \ldots \oplus P_k) = 0 \quad (8)$$

For each plaintext/ciphertext pair, the cryptanalyst will try all 2^{16} possibilities of the 16 key bits $[K_{n+1,9}K_{n+1,10}\ldots K_{n+1,16}, K_{n+1,25}K_{n+1,26}\ldots K_{n+1,32}]$. In the final round we have the following formula:

$$ShiftRow(V_n) \oplus K_{n+1} = C \quad (9)$$

where C is the known ciphertext corresponding to the known plaintext P.

Therefore, for each 16-bit candidate $[K_{n+1,9}K_{n+1,10}\ldots K_{n+1,16}, K_{n+1,25}K_{n+1,26}\ldots K_{n+1,32}]$, the cryptanalyst can find the relevant bits of V_n from (9) as follows.

$$V_n = ShiftRow^{-1}(C \oplus K_{n+1})$$

We have $U_n \rightarrow S_n \rightarrow V_n$, so from V_n, it can be found by working backward to obtain U_n. Replace the corresponding bits of U_n into (8) to check if (8) is satisfied or not.

Try all 2^{16} possibilities for the 16-bit key above. If any candidate 16-bit key $[K_{n+1,9}K_{n+1,10}\ldots K_{n+1,16}, K_{n+1,25}K_{n+1,26}\ldots K_{n+1,32}]$ makes the linear expression (8) true, increase the count value corresponding to that 16-bit key candidate by 1. Repeat this process for the 100,000 pairs of plaintext/ciphertext that the cryptanalyst collected. Then perform the calculation:

$$|bias_i| = |count_i - 50000| / 100000 for 1 \leq i \leq 2^{16}.$$

The value of $count_i$ that differs the most from 50,000, or in other words, the value of $|bias_i|$ that is the largest, corresponds to the correct 16-bit key. Therefore, the 16-bit key found is the correct key bits for the second and fourth bytes of the subkey K_{n+1}.

The permutation operation of the AES block is denoted as T_P (comprising two transformations: ShiftRow and MixColumn).

To proceed in finding the key bits of the K_n key, the cryptanalyst relies on the following formula:

$$U_n = T_P(V_{n-1}) \oplus K_n \quad (10)$$

Because some bits of U_n have already been determined, and the SPN block cipher is static, meaning that the permutation operation T_P is known, the cryptanalyst continues to guess the key bits of the K_n subkey (the key bits related to the active S-boxes in round n). From (10), the cryptanalyst can calculate the bits of V_{n-1} as follows:

$$V_{n-1} = T_P^{-1}(U_n \oplus K_n) \quad (11)$$

On the other hand, in round $n-1$, we also obtain a linear approximation formula in the following form.

$$V_{n-1,i} \oplus V_{n-1,j} \oplus \ldots \oplus V_{n-1,t} \oplus (P_l \oplus P_h \oplus \ldots \oplus P_k) \oplus \left(K_{1,g} \oplus \ldots K_{1,e} \oplus K_{2,f} \oplus \ldots . \oplus K_{3,u} \oplus \ldots \oplus K_{n-1,v}\right) = 0$$

Temporarily excluding these key bits, since their XOR sum can only be 0 or 1, we obtain:

$$V_{n-1,i} \oplus V_{n-1,j} \oplus \ldots \oplus V_{n-1,t} \oplus (P_l \oplus P_h \oplus \ldots \oplus P_k) = 0 \quad (12)$$

Since the necessary bits of V_{n-1} have been found, substituting them into (12) will verify whether this expression is true or false. Thus, for a candidate set of key bits for K_n making (12) true, we increment count value for that set by 1. Similar to the case of the K_{n+1} key, we will identify some correct key bits for K_n (the key bits of K_n related to the S-boxes operating at round n).

Similarly to the process above, for the remaining rounds, we can accurately determine the key bits of the subkeys K_i (the key bits of K_i related to the S-boxes chosen for the linear approximation at round i).

Above are the detailed analyses with a general perspective of our linear attack on AES block cipher based on Matsui's Algorithm 2 [48], [49]. These analyses are designed for linear cryptanalysis on original AES block ciphers, meaning that the components of the block cipher such as S-boxes, AddRoundKey, ShiftRow, Mixcolumns are all public.

b: ANALYZING LINEAR CRYPTANALYSIS ON THE DYNAMIC AES BLOCK CIPHER AT THE ADDROUNDKEY OPERATION

When the SPN block cipher is made dynamic at the AddRoundKey operation, the XOR operation in the AddRoundKey at the rounds of the block cipher will be replaced by a dynamically key-dependent XOR operation.

With U_i and V_i being the inputs and outputs of the S-boxes at round i, then:

$$U_1 = P \overline{\oplus} K_1$$

where P is the 128-bit input plaintext block, K_1 is the round 1 subkey, and $\overline{\oplus}$ represents the key-dependent XOR operation.

Suppose the cryptanalyst continues to follow the steps outlined in Table 8 to construct a linear approximation for $n-1$ rounds and a corresponding linear expression for the dynamic AES block cipher. Assume that the cryptanalyst can perform Steps 1 and 2 as in the case of the static block cipher, meaning that a linear approximation table for the S-box has been constructed, specific S-boxes have been chosen, and their particular linear approximations for each round have been selected (using regular bit XOR operations).

When performing Step 3 of Table 8, the cryptanalyst still obtain a linear expression for the entire dynamic AES block cipher involving plaintext bits (P), input data bits for round n (bits of U_n), and key bits of the subkeys in n rounds (subkey bits of $K_1, K_2, \ldots, K_n$). Suppose this linear expression takes the form:

$$\begin{aligned} &U_{n,i} \oplus U_{n,j} \oplus \ldots \oplus U_{n,t} \oplus \left(\left(P_l \overline{\oplus} P_h \overline{\oplus} \ldots \overline{\oplus} P_k\right)\right. \\ &\left.\overline{\oplus}\left(K_{1,g} \overline{\oplus} \ldots K_{1,e} \overline{\oplus} K_{2,f} \overline{\oplus} \ldots . \overline{\oplus} K_{3,u} \overline{\oplus} \ldots \overline{\oplus} K_{n,v}\right)\right)=0 \end{aligned} \tag{13}$$

Thus, the linear expression of dynamic AES includes both regular bitwise XOR and XOR operations dependent on the key. Temporarily omitting the key bits, the obtained expression is as follows:

$$U_{n,i} \oplus U_{n,j} \oplus \ldots \oplus U_{n,t} \oplus \left(P_l \overline{\oplus} P_h \overline{\oplus} \ldots \overline{\oplus} P_k\right)=0 \tag{14}$$

In this case, even if the attacker attempts to try the key bits of K_{n+1} in the final round to recover $U_{n,i} \oplus U_{n,j} \oplus \ldots \oplus U_{n,t}$ (through the ShiftRow operation and the selected S-boxes in round n), the attacker still doesn't know whether the linear expression (14) is correct with those key bits. Because the $\overline{\oplus}$ operation is key-dependent, the attacker doesn't know which XOR operation it corresponds to. Therefore, the attacker cannot find the key bits of K_{n+1} even with any number of collected plaintext/ciphertext pairs.

Therefore, to launch a linear attack on the case of dynamic AES block cipher with the XOR operation in the AddRoundKey transformation, the attacker needs to identify the dynamic XOR operation $\overline{\oplus}$ that is used. Only after that, the attacker can proceed with the linear attack on the dynamic AES as in the static case. Thus, it is evident that this dynamic aspect adds considerable complexity to the attacker's task, as they must first determine the dynamic XOR operation or the dynamic XOR table used in the dynamic AES before conducting the attack.

2) ANALYZING DIFFERENTIAL CYPTANALYSIS ON THE AES AND DYNAMIC AES

a: ANALYZING DIFFERENTIAL CYPTANALYSIS ON THE AES

Differential cryptanalysis leverages the elevated likelihood of certain instances of differences in plaintexts and input differences occurring in the final round of a block cipher.

Consider an n-round SPN block cipher with a block size of m bits. The input to the block cipher is a bit vector denoted as $X=[X_1X_2 \ldots X_m]$, and the output of the block cipher is denoted as $Y=[Y_1Y_2 \ldots Y_m]$.

Assuming X' and X'' are two inputs of the SPN block cipher with corresponding outputs Y' and Y''.

Notation: $\Delta X=X' \oplus X''=[\Delta X_1 \Delta X_2 \ldots \Delta X_m]$ represents the input difference of the SPN block cipher, and $\Delta Y=Y' \oplus Y''=[\Delta Y_1 \Delta Y_2 \ldots \Delta Y_m]$ represents the output difference of the block cipher, where $\oplus$ is the bitwise XOR operation.

In that case, the pair $(\Delta X, \Delta Y)$ is referred to as a *differential*.

To execute a differential attack, it is necessary to construct a high-probability differential $(\Delta X, \Delta Y)$ related to the plaintext bits and the input bits of the final round of the block cipher. To achieve this, one needs to explore high-probability *differential characteristics*. From these characteristics, the corresponding differential can be identified for the block cipher. A differential characteristic entails a succession of differences in input and output throughout multiple rounds, with the resultant output difference from one round aligning with the input difference for the subsequent round. Then, utilize this high-probability differential to exploit information in the final round and deduce the subkey bits for the final round.

Now, suppose the input to the S-box is the vector $X=[X_1X_2 \ldots X_m]$, and the output of the S-box is $Y=[Y_1Y_2 \ldots Y_m]$. Assume X' and X'' are two inputs to the S-box, and their respective outputs are Y' and Y''.

Note that in the SPN block cipher, the input X to the S-box, before passing through the S-box, is XORed with the round key (AddRoundKey operation). Therefore, the actual input difference to the S-box should be:

$$\Delta W=W' \oplus W''=[\Delta W_1 \Delta W_2 \ldots \Delta W_n]$$

where,

$$\begin{aligned} \Delta W_i &= W_i' \oplus W_i''=\left(X_i' \oplus K_i\right) \oplus\left(X_i'' \oplus K_i\right) \\ &= X_i' \oplus X_i''=\Delta X_i \end{aligned} \tag{15}$$

Note that the XOR operation ($\oplus$) here is a bitwise XOR.

Figure 4 is an example of an 8×8 S-box with key XOR operation.

Thus, for regular SPN ciphers, the input differences of the S-boxes do not depend on the key bits.

Based on the above analysis, we provide Table 9, illustrating the general steps needed to find a differential characteristic for $n-1$ rounds and a corresponding differential for the entire n-round block cipher. These steps follow the differential cryptanalysis method introduced by Biham and Shamir [49], [50], [51].

Note that, the AES block cipher has n rounds, takes a 128-bit input block, and employs 8×8 S-boxes. Let K be the secret key of AES, and the round keys derived from K are denoted as: $K_1, K_2, \ldots, K_n, K_{n+1}$.

Let P and C be the plaintext and corresponding ciphertext of AES. Let ΔP and ΔC be the differences in input plaintext and output ciphertext, respectively, of AES.

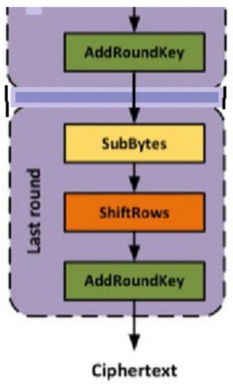


FIGURE 3. The final round of AES.

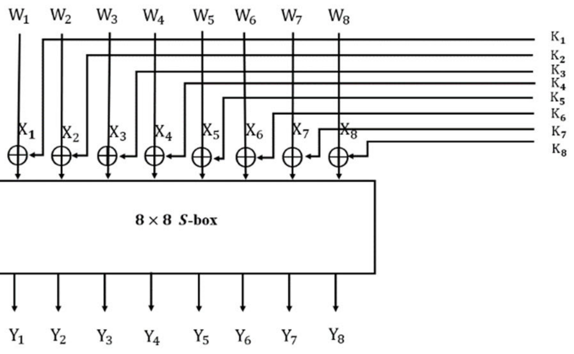


FIGURE 4. A 8 × 8 S-box with key XOR operation.

TABLE 9. Steps to construct a differential for an SPN block cipher.

Constructing a differential characteristic for $n-1$ rounds and a corresponding differential for an n-round SPN block cipher.
Step 1: Construct the difference distribution table for the S-box. - For a static SPN block cipher, where the S-box components are public, this table can be built with all its possible input differences (ΔX) and output differences (ΔY). - From this difference distribution table, the probabilities of various differential $(\Delta X, \Delta Y)$ for the S-box can be computed.
Step 2: Select specific S-boxes in each round (these S-boxes must be active, meaning they exhibit a non-zero differential), and choose specific differentials from the difference distribution table in Step 1 to associate with the selected S-boxes. - The selected S-boxes range from round 1 to round n. - Note that the S-boxes are not randomly chosen but depend on the permutation layer of the SPN block cipher, meaning the output bits of the selected S-boxes in round i must be the input bits of one of the selected S-boxes in round $i+1$. - Each of these chosen characteristics will have a corresponding probability (calculated from the distribution table in Step 1).
Step 3: Find a differential characteristic for the entire block cipher and calculate its probability. - Concatenate the pairs of differences corresponding to the selected S-boxes from Step 2, along with their respective probabilities, to create a differential characteristic for $n-1$ rounds. From this, we also obtain a corresponding differential for the entire SPN block cipher. Based on the assumption that the pairs of differences for the S-boxes are independent across all rounds, we can calculate the probability of this differential.

Let U_i and V_i represent the input and output of the S-boxes at round i. The notation $U_{i,j}$ represents the jth bit of U_i, where $1 \le i \le n$, $1 \le j \le 128$.

Let $\Delta U_i = U_i \oplus U_i'$ and $\Delta V_i = V_i \oplus V_i'$ be the differences in input and output of the S-boxes at round i. The notation $\Delta U_{i,j}$ represents the jth bit of ΔU_i, where $1 \le i \le n$, $1 \le j \le 128$.

From the obtained differential (Table 9) related to the plaintext input bits and the input bits of the S-boxes in the last round (round n), the cryptanalyst can attack the SPN block cipher by recovering a subset of the subkey bits following the last round (bits of the subkey K_{n+1}). Naturally, the cryptanalyst needs to collect a substantial number of plaintext/ciphertext pairs to determine these key bits.

As indicated in (16) (where $\Delta W = \Delta X$), thus the plaintext input difference of AES is equivalent to the input difference of the S-box in the first round, meaning:

$$\Delta U_1 = \Delta P$$

Similarly, the input difference of the SPN block cipher in round i is equivalent to the input difference of the S-box in round i.

We have the differential characteristic $n-1$ rounds in the following form.

$$((\Delta U_1, \Delta V_1, p_1), (\Delta U_2, \Delta V_2, p_2), \ldots, (\Delta U_{n-1}, \Delta V_{n-1}, p_{n-1})) \tag{16}$$

And the differential of the entire SPN block cipher associated with the above differential characteristic will have the following form.

$$(\Delta U_1, \Delta U_n, p) hay (\Delta P, \Delta U_n, p) \tag{17}$$

In which p, and $p_i (1 \le i \le n-1)$ are the corresponding probabilities of the above differentials. The probability p will be calculated from the probabilities p_i with the assumption that the differential pairs of the S-boxes are independent across all rounds.

Now, we will seek ways to find the key bits of the last subkey in the last round of AES from (18).

Suppose in the obtained differential characteristic, the last round (round n) involves the input of two active S-boxes, namely $S_{n,2}$ and $S_{n,5}$ (the 2nd and 5th S-boxes in round n). Corresponding to these two S-boxes, there will be a relationship with two bytes of the last subkey in the last round K_{n+1}.

The notation for these two subkey bytes consists of 16 key bits $[K_{n+1,9} K_{n+1,10} \ldots K_{n+1,16}, K_{n+1,33} K_{n+1,34} \ldots K_{n+1,40}]$. There are a total of 2^{16} possibilities for these 16 key bits. For each set of these 16 key bits, there will be a corresponding count value $count_i (1 \le i \le 2^{16})$, initialized to 0.

From (18), we obtain the value of $\Delta U_n = (\Delta U_{n,1}, \Delta U_{n,2}, \ldots \Delta U_{n,128})$ with probability p.

We proceed to partially decrypt at the last round with the aforementioned two subkey bytes of K_{n+1}. According to (9), we have $ShiftRow(V_n) \oplus K_{n+1} = C$, which implies:

$$V_n = ShiftRow^{-1}(C \oplus K_{n+1}) \tag{18}$$

Assume that the cryptanalyst collected a large number of plaintext/ciphertext pairs, and after some filtering techniques, selected 10,000 pairs of plaintext in the form (P, P') such that

$P \oplus P' = \Delta P$. The corresponding ciphertext pairs are denoted as (C, C').

For each plaintext pair and the corresponding ciphertext pair, attempt all 2^{16} possibilities of the two key bytes $[K_{n+1,9}K_{n+1,10}\ldots K_{n+1,16}, K_{n+1,33}K_{n+1,34}\ldots K_{n+1,40}]$.

For each candidate $[K_{n+1,9}K_{n+1,10}\ldots K_{n+1,16}, K_{n+1,33}K_{n+1,34}\ldots K_{n+1,40}]$, the cryptanalyst decrypts at the last round (2 bytes of the ciphertext corresponding to the two key bytes) as follows:

$$V_n = ShiftRow^{-1}(C \oplus K_{n+1}) \text{ and}$$
$$V'_n = ShiftRow^{-1}(C' \oplus K_{n+1})$$

Because V_n, V'_n are the outputs of the S-box at round n with the corresponding inputs U_n, U'_n, we can calculate by traversing back through the S-box to obtain U_n, U'_n. Calculate:

$$U_n \oplus U'_n = \Delta U_n \tag{19}$$

Check if the two bytes (corresponding to the two key bytes above in K_{n+1}) of ΔU_n obtained from (19) match those two bytes of ΔU_n in the differential (18) or not. If they match, increase the count value corresponding to the candidate $[K_{n+1,9}K_{n+1,10}\ldots K_{n+1,16}, K_{n+1,33}K_{n+1,34}\ldots K_{n+1,40}]$ by one.

Repeat this process with 10,000 plaintext pairs and the corresponding ciphertext pairs. The candidate $[K_{n+1,9}K_{n+1,10}\ldots K_{n+1,16}, K_{n+1,33}K_{n+1,34}\ldots K_{n+1,40}]$ with the highest count value will be the correct two key bytes of K_{n+1}.

To proceed with finding the key bits of K_n, the cryptanalyst utilizes the formula (10) ($U_n = T_P(V_{n-1}) \oplus K_n$), where T_P is denoted as the permutation operation of the AES block cipher.

Because some bits of U_n, U'_n have been found, and the AES block cipher has public components, meaning the permutation operation T_P is known, the cryptanalyst continues to guess the key bits of the subkey K_n (key bits related to the active S-boxes at round n). Then, from (10), the cryptanalyst can compute the related bits of V_{n-1} (which is $V_{n-1} = T_P^{-1}(U_n \oplus K_n)$).

Similarly, the cryptanalyst also computes the related bits of V'_{n-1}. Next, calculate:

$$V_{n-1} \oplus V'_{n-1} = \Delta V_{n-1} \tag{20}$$

On the other hand, at round $n-1$, we already know the value of ΔV_{n-1} in the obtained differential characteristic. Compare the bits of ΔV_{n-1} in the differential characteristic with the corresponding bits of ΔV_{n-1} obtained in (20). If they match, the guessed key bits may be correct as subkey bits of K_n, and increase the count value of these candidate key bits. Similarly to the process above with the key K_{n+1}, we can accurately find some related key bits of the subkey K_n.

Above is our detailed and general analysis for the differential attack on AES block cipher based on the method of Biham and Shamir [49], [50], [51]. These analyses are dedicated to the differential cryptanalysis for the original AES block cipher, meaning that the components of AES such as S-boxes, AddRoundKey, ShiftRows, and MixColumns are all public.

b: ANALYZING DIFFERENTIAL CRYPTANALYSIS ON THE DYNAMIC AES BLOCK CIPHER AT THE ADDROUNDKEY OPERATION

When the AES block cipher is made dynamic in the AddRoundKey operation, the XOR operation in the AddRoundKey at the rounds of AES will be replaced by a dynamically key-dependent XOR operation.

With U_i and V_i being the inputs and outputs of the S-boxes at the i-th round, we have:

$$U_1 = P \overline{\oplus} K_1, \; U'_1 = P' \overline{\oplus} K_1$$

where, $\overline{\oplus}$ is denoted as the dynamic key-dependent XOR operation applied to the AddRoundKey transformation.

Therefore,

$$\Delta U_1 = (P \overline{\oplus} K_1) \oplus (P' \overline{\oplus} K_1) \tag{21}$$

So, the first challenge in this dynamic case is that the input difference of the S-box in round 1 depends on the unknown key K_1, and the cryptanalyst does not know the dynamic XOR operation $\overline{\oplus}$. As a result, they cannot determine the value of ΔU_1.

Now, assuming the cryptanalyst is still able to construct a differential characteristic for $n-1$ rounds using the steps outlined in Table 9, obtaining the corresponding differential characteristic in (17) and the corresponding differential in (18).

Next, the cryptanalyst proceeds with decryption in the final round using a given plaintext pair and a corresponding ciphertext pair and the key bytes of K_{n+1} related to the active S-boxes operating in round n.

$$V_n = ShiftRow^{-1}(C \overline{\oplus} K_{n+1}) \text{ and}$$
$$V'_n = ShiftRow^{-1}(C' \overline{\oplus} K_{n+1}) \tag{22}$$

At this point, the cryptanalyst encounters the second challenge as they cannot decrypt V_n and V'_n due to the unknown nature of the $\overline{\oplus}$ operation, even if they attempt to guess the relevant key bits of K_{n+1}. Consequently, the cryptanalyst is unable to determine the correct key bits for K_{n+1}.

Therefore, for the case of dynamic AES block cipher at the AddRoundKey operation, if cryptanalysts want to perform a differential attack, they must first determine the dynamic XOR operation $\overline{\oplus}$ used. Even if the cryptanalyst finds the dynamic XOR operation $\overline{\oplus}$, they still face additional difficulties because $\Delta U_1 = (P \overline{\oplus} K_1) \oplus (P' \overline{\oplus} K_1)$ depends on the key K_1. Thus, they continue to try the relevant key bits of K_1, and then they have to choose suitable plaintext pairs so that the value of ΔU_1 matches the value of ΔU_1 in the constructed differential characteristic of $n-1$ rounds. After that, they proceed with the differential cryptanalysis as usual, but the probability of selecting correct plaintext pairs/ciphertext pairs becomes smaller compared to the case of the original AES block cipher.

Therefore, it is evident that this dynamic AddRoundKey operation poses significant challenges for cryptanalysts. The dynamic key-dependent XOR operations $\overline{\oplus}$ add an extra difficulty for cryptanalysts compared to the original AES case. At the very least, the cryptanalyst needs to invest effort in determining the dynamic XOR operation or dynamic XOR table used in dynamic AES among the $(16!)^2$ possible dynamic XOR tables.

Based on the aforementioned analysis, it is evident that the dynamic AES utilizing key-dependent XOR tables can substantially enhance the security of the AES block cipher.

3) COMPLEXITY ANALYSIS OF LINEAR AND DIFFERENTIAL ATTACKS ON AES AND DYNAMIC AES

For linear cryptanalysis, the related S-boxes in linear approximations are active S-boxes. The probability of a linear expression being correct is related to the linear probability bias in the active S-boxes and the number of active S-boxes. In general, the larger the bias in the S-boxes, the larger the bias of the overall linear expression. Also, the fewer active S-boxes, the larger the bias of the overall linear expression.

With ε denoting the bias from 1/2 of the probability for the linear expression to hold true for the entire block cipher, Matsui [48] showed that the quantity of known plaintexts needed for a linear attack scale proportionally with ε^{-2}. Denote by N_L the quantity of plaintexts needed for the attack, meaning that when there are N_L known plaintexts, the linear attack can be carried out. In that case,

$$N_L \approx \frac{1}{\varepsilon^2} \tag{23}$$

As the bias is determined through the utilization of the Piling-Up Lemma, wherein product refers to a linear approximation of an S-box, it's clear that the bias relies on the biases of the linear approximations of the S-boxes and the quantity of engaged active S-boxes. Common approaches to enhancing resilience against linear cryptanalysis have centered on improving the S-boxes (i.e., reducing the highest bias) and discovering configurations to maximize the quantity of active S-boxes. The design methodology employed in Rijndael stands as a noteworthy illustration of this strategy.

For differential cryptanalysis, active S-boxes within a differential characteristic are S-boxes with a non-zero input difference (and thus a non-zero output difference). Typically, the greater the differential probabilities associated with active S-boxes, the higher the probability of the overall characteristic across the cipher. Similarly, fewer active S-boxes lead to a higher probability of the characteristic. Analogous to linear cryptanalysis, we refer to the required data for conducting a differential attack when assessing the complexity of the cryptanalysis. This means that if we have N_D pairs of plaintexts, we can carry out a differential attack.

Typically, precisely ascertaining the precise count of selected plaintext pairs needed to carry out a successful differential attack poses significant complexity. Nonetheless, it can be demonstrated [49], [50], [51] that there exists a straightforward and efficient guideline for estimating the requisite quantity of chosen plaintext pairs N_D as follow.

$$N_D \approx \frac{c}{p_D} \tag{24}$$

Given that $\mathrm{p_D}$ represents the probability of the differential characteristic for the $n-1$ rounds of the n-round cipher and c is a minor constant, if we suppose that the appearances of difference pairs in every active S-box are unrelated, then the differential characteristic probability can be expressed as:

$$p_D = \prod\nolimits_{i=1}^{\tau} pd_i \tag{25}$$

where τ stands for the count of active S-boxes, and the likelihood of the specific difference pair appearing in the i-th active S-box of the characteristic is denoted by $\mathrm{pd_i}$.

Efforts to resist against differential cryptanalysis have concentrated on optimizing the characteristics of S-boxes (such as reducing the probability of difference pairs in an S-box) and identifying arrangements to increase the quantity of active S-boxes. Rijndael serves as a notable instance of a cipher engineered to offer robust protection against differential cryptanalysis.

Thus, the prerequisite task is to identify specific differential characteristics/linear expressions with high probabilities before conducting a differential/linear attack. To impart practical relevance to cryptographic constructions, Knudsen [57] introduced the concept of practical resilience against differential and linear cryptanalysis by showing the non-existence of a differential characteristic or a linear expression with a probability high enough to enable a successful attack. In essence, we ascertain a cipher's practical security when the minimum complexity bound of the characteristics/expressions can be suitably low. The prevalent method involves tallying the least number of active S-boxes (differential and linear) throughout the block cipher's rounds, as employed in the wide trail strategy of AES ([23]).

In [37], the authors proposed a dynamic modified AES algorithm and also assess its security using an approach that involves identifying active S-boxes; however, their evaluation is quite rudimentary. In [58], the authors evaluate linear cryptanalysis on one and two rounds of AES and assess the success rates on single and multi-systems. In [59], the authors analyzed the potential of linear cryptanalysis against the Simplified AES Algorithm (SAES). Based on the findings of this linear cryptanalysis, it was demonstrated that linear calculations can compromise the security of both the first and second rounds of SAES.

Daemen and Rijmen introduced the wide trail strategy in [22] and [23] as a design approach aimed at maximizing the active S-boxes by selecting appropriate linear transformations. This strategy focuses on employing concise and efficient transformations to ensure sufficient active S-boxes throughout the encryption process. It is utilized in crafting SPN-based encryption schemes like AES, LED, Kalyna, among others. In their work [23], the AES authors demonstrated that the minimum number of active S-boxes in AES within any 4-round differential characteristic or

TABLE 10. **Number of active S-boxes across the encryption rounds of AES and dynamic AES.**

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Count of active S-boxes per round	1	4	16	4	1	4	16	4	1	4	16	4	1	4
Cumulative count of active S-boxes throughout the rounds	1	5	21	**25**	26	30	46	**50**	51	55	71	**75**	76	80

linear approximation is 25 (see more in Theorem 9.5.1, page 142 in [23])

Since the dynamic AES algorithm we propose in this paper only involves dynamic XOR operations in the AddRoundKey transformation, the substitution boxes and diffusion layer of dynamic AES remain the same as AES. Therefore, the wide trail strategy of AES is preserved in dynamic AES. Consequently, the total number of active S-boxes across rounds can be summarized, and the total number of active S-boxes across rounds for both AES and dynamic AES is presented in Table 10.

For both AES and dynamic AES, the S-box used in the SubByte transformation has the following parameters:

+ The maximum differential probability is $Max\left(pd_i\right) = 2^{-6}$.

+ The maximum linear probability bias is: $Max\left(\varepsilon\right) = 2^{-3}$.

In this case, the formulas for calculating the data complexity against linear cryptanalysis and differential cryptanalysis are as follows.

Let $S(t)$ denote the number of active S-boxes after t rounds of the AES block cipher and dynamic AES. $S(t)$ can be obtained from Table 10.

Then, from equations (24) and (26), the data complexity of the AES block cipher and dynamic AES against differential cryptanalysis at the t-th round can be calculated using the following formula:

$$N_D \approx \frac{1}{Max\left(pd_i\right)^{S(t)}} \tag{26}$$

From (23) and the Piling-Up Lemma, the data complexity of the AES block cipher and dynamic AES against linear cryptanalysis at the t-th round can be calculated using the following formula:

$$N_L \approx \frac{1}{Max\left(\varepsilon\right)^{2S(t)}} \tag{27}$$

Therefore, from Table 10 and formulas (26) and (27), the lower bounds of data complexity for linear cryptanalysis and differential cryptanalysis against AES and dynamic AES with a 128-bit key version are determined in Table 11.

From Table 11, it can be observed that with a 128-bit secret key, after 4 rounds both AES and dynamic AES algorithms are secure against linear and differential cryptanalysis. It is noteworthy that the data complexity of dynamic AES here is when we assume the attacker is guessing two specific dynamic XOR tables used in dynamic AES.

TABLE 11. **Data complexity of linear cryptanalysis and differential cryptanalysis against AES and dynamic AES.**

Round	Differential cryptanalysis		Linear cryptanalysis	
	Upper bound of the probability of differential probability	Data complexity (Number of chosen plaintext pairs)	Upper bound of the probability bias	Data complexity (Number of known plaintexts)
1	2^{-6}	2^{6}	2^{-3}	2^{6}
2	2^{-30}	2^{30}	2^{-15}	2^{30}
3	2^{-126}	2^{126}	2^{-63}	2^{126}
4	2^{-150}	2^{150}	2^{-75}	2^{150}
5	2^{-156}	2^{156}	2^{-78}	2^{156}
6	2^{-180}	2^{180}	2^{-90}	2^{180}
7	2^{-276}	2^{276}	2^{-138}	2^{276}
8	2^{-300}	2^{300}	2^{-150}	2^{300}
9	2^{-306}	2^{306}	2^{-153}	2^{306}
10	2^{-330}	2^{330}	2^{-165}	2^{330}

On the other hand, as we have analyzed in detail above, a prerequisite for both differential and linear cryptanalysis is to find differential characteristics or linear expressions with high probabilities. However, when using dynamic AES with key-dependent XOR tables in the AddRoundKey operation, cryptanalysis becomes extremely difficult because without knowing which dynamic XOR tables are used, attackers face significant obstacles in finding such characteristics and expressions. In this paper, we employ $(16!)^2$ key-dependent XOR tables for dynamic AES. Therefore, cryptanalysts must first identify the exact two XOR tables we use alternately for rounds before they can proceed with their cryptanalysis.

Furthermore, with differential cryptanalysis, as analyzed above, even if cryptanalysts manage to identify the dynamic XOR operations as $\overline{\oplus}$, they still encounter additional difficulties because $\Delta U_1 = (P\overline{\oplus}K_1) \oplus (P'\overline{\oplus}K_1)$ depends on the key K_1. Therefore, they continue to need to test related key bits of K_1, then they have to select appropriate plaintext pairs to match the value of ΔU_1 with the value of ΔU_1 in the previously constructed $n-1$ round differential characteristic. Only then can they proceed with differential cryptanalysis as usual; however, at that point, the probability of selecting correct plaintext/ciphertext pairs will be even smaller compared to the case of the original AES block cipher.

4) SOME LIMITATIONS OF OUR METHOD

Our approach can significantly enhance the security of the AES block cipher; however, it also has some limitations and vulnerabilities, such as:

- In the implementation, additional memory space must be allocated for the new XOR table. Certainly, we do not pre-generate all the XOR tables but generate a new XOR table for each session corresponding to a secret key, then store this new XOR table throughout that session. When transitioning to another session with other key, a new XOR table is generated once again. Therefore, this XOR table needs to be safeguarded until the end of that session.
- Generating key-dependent XOR tables will impact the encryption/decryption time. Because for each session using a secret key, both the sender and receiver must execute Algorithm 1 to generate a new XOR table before using it for AES. Therefore, it will take some time at the beginning of a session for both parties to generate a new XOR table.
- Algorithm 1 utilizes a pseudo-random number generator, hence the security of this algorithm is also influenced by the security of the random number generator it employs. Therefore, it is essential to select a cryptographically secure random number generator.

Our proposed method in this paper can enhance the security of the AES block cipher against various strong attacks, such as linear and differential attacks. However, it is important to note that side-channel attacks can be used to extract encryption key information by analyzing the physical characteristics of the system. Therefore, both static and dynamic block ciphers can be vulnerable to this type of attack. Hence, regardless of the cryptographic system in use, users should ensure that appropriate countermeasures are in place to prevent side-channel attacks, such as using secure hardware or implementing software-based countermeasures.

Our proposed dynamic AES algorithm can be applied to systems demanding very high levels of security and can be used to protect highly sensitive data. The dynamic AES algorithm operates similarly to AES, making it integratable into existing standards, protocols, and systems. However, the challenge arises when the secret key changes, meaning a switch to a different session, dynamic AES needs to be reconfigured as it relies on that specific secret key. Hence, the dynamic AES algorithm is best suited for systems using session keys for relatively long periods, which can reduce administrative effort. On the other hand, hardening the dynamic AES algorithm can be problematic since the algorithm changes with the session key. Therefore, depending on the real-world environment and application, users can apply the proposed dynamic AES algorithm with the primary goal of enhancing the security of their data

B. ASSESSMENT OF RANDOMNESS VIA THE NIST TESTS

It is imperative for any cryptographic module to ensure that the redundancies at the input do not compromise the output. During the selection process of AES finalists, NIST utilized a method of concatenating the output strings of cryptographic primitives, followed by an evaluation of the randomness through NIST SP 800-22 test suite [53]. However, it is crucial to treat the cryptographic primitives as a PRNG while making such evaluations, and the results of the randomness assessment are applicable to the PRNG rather than the cryptographic primitive being evaluated.

Several tests have been proposed in the literature in order to evaluate the randomness of the altered AES block cipher. In [54], the authors suggested using Collision Test, Linear Span Test, SAC Test, and Coverage Test to evaluate cryptographic criteria for hash functions and block ciphers.

Block ciphers in particular, and symmetric-key primitives in general, need to satisfy a fundamental requirement: randomness. Currently, there are two main strategies for assessing the randomness of these primitives, as follows.

1) TESTING RANDOMNESS FOR CRYPTOGRAPHIC FUNCTIONS AS WITH A PRNG

In this approach, a cryptographic function is treated as a pseudo-random number generator (PRNG). In this case, the randomness of the output can be assessed using statistical tests designed for pseudo-random number generators. Commonly used statistical test suites include Diehard, Dieharder, ENT, TestU01, and the NIST SP 800-22 test suite.

Although the title of the NIST SP 800-22 test suite [60] states that it is for ‘‘cryptography applications,’’ in reality, it is a general-purpose statistical test suite designed to evaluate the randomness of binary sequences-unlike statistical tests for sequences of evenly distributed integers or real numbers, as commonly seen in other test suites. The NIST test suite does not concern itself with the origin of the binary sequences, whether they come from a cryptographic function or another source.

In practice, there arises an issue when using the NIST SP 800-22 test suite (or any statistical test suite) to assess a cryptographic function that produces fixed-length output due to the fact that the cryptographic function is not a PRNG; it does not generate an arbitrarily long binary sequence. To apply the NIST test suite to a cryptographic function, the function must be adjusted to operate as a PRNG and produce binary sequences of sufficient length. The NIST SP 800-22 specification document does not specify how this should be done. NIST has published a document [61] describing how NIST implements candidate block ciphers in the AES competition to generate long binary sequences. The technique used is encrypting a long sequence of 0 bits using block cipher in Cipher Block Chaining (CBC) mode with randomly chosen different keys and applying the test suite to the resulting ciphertext sequence. Another similar technique involves encrypting a sequence of all 1 bits instead of a long sequence of all 0 bits using block cipher in ECB mode.

However, when a block cipher is transformed into a PRNG and subjected to the NIST test suite (or another test suite), it is testing the randomness of the PRNG, not the inherent randomness of the block cipher itself. If a non-random bias is detected in the output, it is unclear whether that bias stems from the block cipher or is introduced by the mode of operation used.

2) TESTING THE RANDOMNESS OF CRYPTOGRAPHIC FUNCTIONS DIRECTLY AS A RANDOM MAPPING

In fact, there are some studies analyzing the randomness of the mapping of cryptographic functions directly, without considering the function as a PRNG. Specifically, Filiol [62] defined a statistical test based on comparing the algebraic normal form of the cryptographic function with the algebraic normal form of a random Bool function, and applied the test to DES and AES block ciphers as well as some stream ciphers and hash functions. Katos [63] defined a statistical test to measure the diffusion of the block cipher mapping but did not apply the test to any actual block cipher. Doğanaksoy et al. [54] defined four statistical tests based on block cipher mappings - strict avalanche criterion (SAC), linear span test, collision test, and coverage test - and applied these tests to candidates in the final round of AES selection. These authors also applied the methodology of [54] and [56] to second-round SHA-3 candidate hash functions [64]. In addition to frequency-based approaches like those above, in [65], the authors used Bayesian approaches to assess the randomness of block ciphers and hash functions. In the literature, there are several approaches to assessing randomness, as outlined in [66]. In this document, the authors presented the outcomes of NIST's statistical analyses conducted on diverse datasets derived from the output of every feasible truncated iteration of the finalists from the NIST Lightweight standardization procedure.

However, in this paper, we rely on the approach outlined in [55], which we find most reasonable for evaluating the randomness of proposed block ciphers, particularly by employing the NIST SP 800-22 test suite with adjustments for short sequences to suit block cipher evaluations. The input to the block cipher consists of keys and plaintexts, where we use non-random plaintext datasets and two scenarios of completely zero keys and random keys to assess the randomness of the block cipher. When the input data is non-random and the evaluation result is random, then the randomness must stem from the block cipher's structure. In practice, keys used are generated from random number generators that satisfy stringent standards such as NIST SP 800-90 or AIS20/AIS31, so we do not consider randomness testing standards for keys. Furthermore, the scope of the paper necessitates evaluating the randomness of the block cipher rather than assessing the keys.

In [55], the author proposed a technique for assessing the randomness of hash functions and block ciphers by generating input data sets with redundancy and calculating their corresponding outputs that use the cryptographic primitives. The randomness of the output data sets is then checked using statistical tests. If the data exhibits randomness, it is evident that the cryptographic primitive is the source of the randomness; otherwise, the cryptographic primitive cannot be attributed with any randomness. Our aim is to assess the output randomness of the altered AES block cipher using these methods, assuming that the algorithm being tested is a mapping $F_2^m \times F_2^n \rightarrow F_2^n$, where n is the block size and m is the key size of the altered block cipher.

Evaluation Process: The process employed to assess the randomness of the output from the altered AES block cipher is as follows.

1. As outlined in [55], there are four types of datasets used as input to be considered for the block cipher: High Weight (HW) Plaintext, Plaintext rotation (Rot), 1-bit Plaintext avalanche (Av1), Low Weight (LW) Plaintext data sets, all of which are non-random in nature.

2. Calculate the output dataset of the cipher, corresponding to the data generated as the first step of the process, and the discretionary key, for each truncated version of the cipher. In the case of, for example, the Low Weight 128-bit (LW128) input dataset, calculate 10 output sequences for each altered cipher with round numbers ranging between 1 and 10, each output sequence comprising 349,632 output sets of input strings of128-bit from the Low Weight 128-bit input data.

3. Derive the p-values for each of output strings by utilizing the two-tiered testing methodology (from NIST) with some adjustment made for statistical testing of short strings.

4. One can conclude that a T-round block cipher can only be considered random if all its corresponding output data sets satisfy the below statistical tests, as evidenced by the results and consequences.

Note: In this context, the variable T refers to the round number performed by the cipher.

Statistical Tests: Within this manuscript, we adopt a dual-tiered testing methodology as prescribed in NIST SP 800-22 [46]. The distribution of p-values resulting from the tests, when applied to brief sequences, has been analyzed and demonstrated in [56].

Our assessment encompassed both AES and the altered AES, and we examined two scenarios regarding the choice of the master key: a random key and a zero key. The findings indicated that the altered version achieves output randomness akin to AES. Table 12 displays the outcome of a specific scenario in which input data set AV1 was used in conjunction with a random key.

We present the results of the randomness evaluation for AES and the modified AES from Table 12 using Figure 5 and Figure 6.

We can observe that the original AES algorithm achieves randomness after three rounds with the input data AV1 and a random key, whereas the modified AES algorithm achieves randomness after only two rounds with similar data and a random key.

The summary results are presented in Table 13.

To sum up, we can see that both the original and modified AES algorithms achieve randomness after three rounds with a

TABLE 12. The results of the randomness evaluation for the AV1 data set using a random key of AES-128 and the altered AES-128.

No. of Rounds	Freq. Test	Runs Test	Test for longest run of ones	Serial Test	AppEn. Test	CuSum. Test	Bit AutoCorr. Test	Byte Autocor. Test
Original AES-128								
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0.000022	0
3	**0.598894**	**0.018305**	**0.339122**	**0.114644**	**0.017900**	**0.401902**	**0.068564**	**0.783297**
4	0.543253	0.601551	0.597370	0.474953	0.471361	0.511647	0.962485	0.907979
5	0.704804	0.098806	0.752760	0.019669	0.009143	0.770070	0.349626	0.745146
6	0.590701	0.129585	0.685053	0.286265	0.313612	0.489214	0.106276	0.727202
7	0.936945	0.023026	0.582644	0.416645	0.402667	0.176081	0.005053	0.302354
8	0.128106	0.413249	0.187094	0.450533	0.667270	0.617485	0.098959	0.065653
9	0.752848	0.202312	0.891466	0.449087	0.144161	0.583684	0.141038	0.861803
10	0.528404	0.357128	0.749070	0.221138	0.430105	0.752828	0.624921	0.712713
Modified AES-128								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0.291414	0	0	0	0	0	0	0.000036
2	**0.097656**	**0.962868**	**0.225424**	**0.961320**	**0.986249**	**0.308869**	**0.401396**	**0.574189**
3	0.756877	0.584815	0.283272	0.806989	0.932261	0.995404	0.409544	0.007807
4	0.013000	0.375452	0.021257	0.520289	0.861955	0.660107	0.001587	0.878102
5	0.000871	0.657434	0.925328	0.002519	0.002378	0.972854	0.381931	0.727420
6	0.201483	0.226633	0.016339	0.000127	0.005079	0.067994	0.148640	0.496364
7	0.035630	0.702862	0.487486	0.000436	0.002082	0.004356	0.015346	0.750629
8	0.119203	0.445908	0.154656	0.833488	0.195910	0.474348	0.609771	0.774577
9	0.892658	0.990783	0.966658	0.739805	0.716736	0.195910	0.474348	0.609771
10	0.857050	0.144153	0.037715	0.197433	0.445099	0.700560	0.596591	0.484707

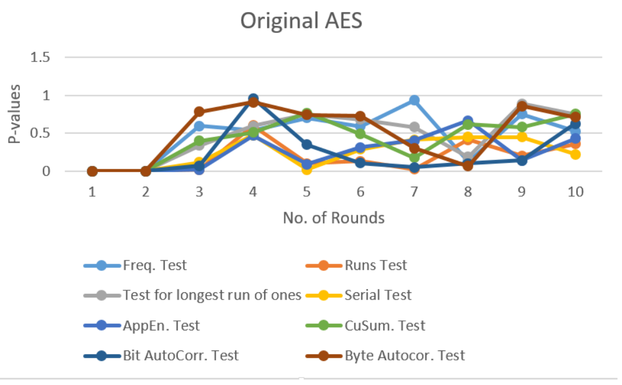


FIGURE 5. Randomness evaluation results of AES.

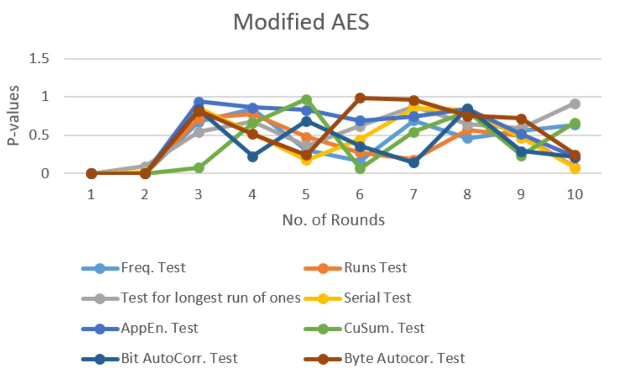


FIGURE 6. Randomness evaluation results of the altered AES.

random key and input data sets LW, HW, Av1, Rot. Also, both the AES and altered AES algorithms achieve randomness

TABLE 13. The findings of the randomness assessment for the AES-128 and the altered AES.

The algorithm	Number of rounds	Number of rounds random	Input data sets [56]
The AES-128 utilizing a zero key.	10	≥ 4	Rot, HW, LW, Av1
The altered AES-128 utilizing a zero key	10	≥ 4	Rot, HW, LW, Av1
The AES-128 utilizing a random key	10	≥ 3	Rot, HW, LW, Av1
The altered AES-128 utilizing a random key	10	≥ 2	Rot, HW, LW, Av1

after four rounds with input data sets LW, HW, Av1, Rot, and zero key.

V. CONCLUSION

In an effort to enhance the strength of SPN block ciphers, including the AES block cipher, dynamic approaches may target individual parts of the ciphers, such as the diffusion layer transformation, substitution transformation, or both. This manuscript introduces a technique for rendering AES dynamic at the key addition transformation with key-dependent XOR tables. Within this study, we outline the requisite characteristics of an XOR table and we prove the correctness of the new XOR table generated by our method. Notably, the revised XOR operation can maintain the uniform probability and independent distribution of the random key in the ciphertext. We analyze the security of the altered AES block cipher, and the number of key-dependent XOR tables that can be created through our approach is equivalent to $(16!)^2$. With such an extensive array of new XOR tables, cryptanalysts will face significant challenges in ascertaining the specific XOR table employed in the altered AES block cipher. Lastly, both the AES block cipher and dynamic AES block cipher were assessed for randomness utilizing NIST statistical criteria. The results of the experiment indicate that both the altered AES and AES algorithms exhibit randomness after three rounds with a random key and input data sets Rot, HW, LW, Av1, and after four rounds with a zero key and input data sets Rot, HW, LW, Av1, both exhibit randomness. Therefore, the method proposed has the ability to generate a dynamic AES block cipher that strengthens the strength of AES against various powerful attacks targeting block ciphers. In our upcoming research, we plan to integrate dynamic techniques into the substitution, key addition, and diffusion layers to enhance the security of SPN block ciphers even further. Simultaneously making multiple components of the block cipher dynamic will make it more challenging for attackers to carry out attacks. We will also invest more time and effort in researching specific attacks, especially conducting linear and differential attacks against the proposed dynamic block ciphers.

REFERENCES

[1] C. E. Shannon, "Communication theory of secrecy systems," *Bell Syst. Tech. J.*, vol. 28, no. 4, pp. 656–715, Oct. 1949.

[2] P. L. Andono and D. R. I. M. Setiadi, "Improved pixel and bit confusion-diffusion based on mixed chaos and hash operation for image encryption," *IEEE Access*, vol. 10, pp. 115143–115156, 2022, doi: 10.1109/ACCESS.2022.3218886.

[3] E. Winarno, K. Nugroho, P. W. Adi, and D. R. I. M. Setiadi, "Combined interleaved pattern to improve confusion-diffusion image encryption based on hyperchaotic system," *IEEE Access*, vol. 11, pp. 69005–69021, 2023, doi: 10.1109/ACCESS.2023.3285481.

[4] 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 Appl.*, vol. 79, nos. 17–18, pp. 11667–11684, May 2020, doi: 10.1007/s11042-019-08464-6.

[5] S. Mister and C. Adams, "Practical S-box design," in *Proc. Workshop Sel. Areas Cryptography (SAC)*, vol. 96, Aug. 1996, pp. 61–76.

[6] A. M. Youssef and S. E. Tavares, "Resistance of balanced S-boxes to linear and differential cryptanalysis," *Inf. Process. Lett.*, vol. 56, no. 5, pp. 249–252, Dec. 1995, doi: 10.1016/0020-0190(95)00156-6.

[7] 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," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, Germany. Berlin, Springer, 2006, pp. 51–64.

[8] M. Kumar, P. Yadav, S. Pal, and A. Panigrahi, "Secure and efficient diffusion layers for block ciphers," *J. Appl. Comput. Sci. Math.*, vol. 11, no. 2, pp. 15–20, 2017.

[9] H. N. Noura and A. Chehab, "Efficient binary diffusion matrix structures for dynamic key-dependent cryptographic algorithms," *J. Inf. Secur. Appl.*, vol. 68, Aug. 2022, Art. no. 103264.

[10] L. Grassi, R. Lüftenegger, C. Rechberger, D. Rotaru, and M. Schofnegger, "On a generalization of substitution-permutation networks: The HADES design strategy," in *Proc. 39th Annu. Int. Conf. Theory Appl. Cryptograph. Techn. (EUROCRYPT)*, Zagreb, Croatia. Springer, 2020, pp. 674–704. [Online]. Available: https://eprint.iacr.org/2019/1107, doi: 10.1007/978-3-030-45724-2_23.

[11] H. M. Heys and S. E. Tavares, "Substitution-permutation networks resistant to differential and linear cryptanalysis," *J. Cryptol.*, vol. 9, no. 1, pp. 1–19, 1996.

[12] H. M. Heys and S. E. Tavares, "Avalanche characteristics of substitution-permutation encryption networks," *IEEE Trans. Comput.*, vol. 44, no. 9, pp. 1131–1139, Sep. 1995.

[13] J. B. Kam and G. I. Davida, "Structured design of substitution-permutation encryption networks," *IEEE Trans. Comput.*, vol. C-28, no. 10, pp. 747–753, Oct. 1979.

[14] R. Li, B. Sun, and C. Li, "Impossible differential cryptanalysis of SPN ciphers," *IET Inf. Secur.*, vol. 5, no. 2, pp. 111–120, 2011, doi: 10.1049/iet-ifs.2010.0174.

[15] H. Q. Le, B. Vo, D. H. Duong, W. Susilo, N. T. Le, K. Fukushima, and S. Kiyomoto, "Identity-based linkable ring signatures from lattices," *IEEE Access*, vol. 9, pp. 84739–84755, 2021, doi: 10.1109/ACCESS.2021.3087808.

[16] R. Guo, H. Shi, Q. Zhao, and D. Zheng, "Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems," *IEEE Access*, vol. 6, pp. 11676–11686, 2018, doi: 10.1109/ACCESS.2018.2801266.

[17] A. Waheed, A. I. Umar, N. Din, N. U. Amin, S. Abdullah, and P. Kumam, "Cryptanalysis of an authentication scheme using an identity based generalized signcryption," *Mathematics*, vol. 7, no. 9, p. 782, Aug. 2019, doi: 10.3390/math7090782.

[18] N. Din, A. Waheed, M. Zareei, and F. Alanazi, "An improved identity-based generalized signcryption scheme for secure multi-access edge computing empowered flying ad hoc networks," *IEEE Access*, vol. 9, pp. 120704–120714, 2021, doi: 10.1109/ACCESS.2021.3108130.

[19] X. Yang, Y. Zhang, S. Wang, B. Yu, F. Li, Y. Li, and W. Yan, "LedgerDB: A centralized ledger database for universal audit and verification," *Proc. VLDB Endowment*, vol. 13, no. 12, pp. 3138–3151, Aug. 2020.

[20] D. L. Fekete and A. Kiss, "A survey of ledger technology-based databases," *Future Internet*, vol. 13, no. 8, p. 197, Jul. 2021.

[21] Z.-K. Gao, A.-A. Liu, Y. Wang, M. Small, X. Chang, and J. Kurths, "IEEE access special section editorial: Big data learning and discovery," *IEEE Access*, vol. 9, pp. 158064–158073, 2021.

[22] J. Daemen and V. Rijmen, "AES proposal: Rijndael," in *Proc. 1st Adv. Encryption Conf.*, 1998, pp. 1–45.

[23] J. Daemen and V. Rijmen, *The Design of Rijndael*, vol. 2. New York, NY, USA: Springer-Verlag, 2002.

[24] L. R. Knudsen and C. V. Miolane, "Counting equations in algebraic attacks on block ciphers," *Int. J. Inf. Secur.*, vol. 9, no. 2, pp. 127–135, Apr. 2010, doi: 10.1007/s10207-009-0099-9.

[25] X. Lai, J. L. Massey, and S. Murphy, "Markov ciphers and differential cryptanalysis," in *Proc. Adv. Cryptology-EUROCRYPT Workshop Theory Appl. Cryptograph. Techn.*, Brighton, U.K. Berlin, Germany: Springer, Apr. 1991, pp. 17–38.

[26] M. Rahman and G. Paul, "Grover on KATAN: Quantum resource estimation," *IEEE Trans. Quantum Eng.*, vol. 3, pp. 1–9, 2022, doi: 10.1109/TQE.2022.3140376.

[27] P. Agarwal, A. Singh, and A. Kilicman, "Development of key-dependent dynamic S-boxes with dynamic irreducible polynomial and affine constant," *Adv. Mech. Eng.*, vol. 10, no. 7, Jul. 2018, Art. no. 168781401878163, doi: 10.1177/1687814018781638.

[28] 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," *J. Inf. Secur. Appl.*, vol. 64, Feb. 2022, Art. no. 103065, doi: 10.1016/j.jisa.2021.103065.

[29] H. T. Assafli and I. A. Hashim, "Generation and evaluation of a new time-dependent dynamic S-box algorithm for AES block cipher cryptosystems," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 978, no. 1, 2020, Art. no. 012042, doi: 10.1088/1757-899X/978/1/012042.

[30] A. Ejaz, I. A. Shoukat, U. Iqbal, A. Rauf, and A. Kanwal, "A secure key dependent dynamic substitution method for symmetric cryptosystems," *PeerJ Comput. Sci.*, vol. 7, p. e587, Jul. 2021.

[31] K. Kazlauskas and J. Kazlauskas, "Key-dependent S-box generation in AES block cipher system," *Informatica*, vol. 20, no. 1, pp. 23–34, Jan. 2009.

[32] S. Murphy and M. J. B. Robshaw, "Key-dependent S-boxes and differential cryptanalysis," *Des., Codes Cryptogr.*, vol. 27, no. 3, pp. 229–255, 2002.

[33] T. T. Luong, "Building the dynamic diffusion layer for SPN block ciphers based on direct exponent and scalar multiplication," *J. Sci. Technol. Inf. Secur.*, vol. 1, no. 15, pp. 38–45, Jun. 2022.

[34] G. Murtaza, A. A. Khan, S. W. Alam, and A. Farooqi, "Fortification of AES with dynamic mix-column transformation," Cryptol. ePrint Arch., Tech. Paper 2011/184, pp. 1–13, Jan. 2011.

[35] M. R. M. Shamsabad and S. M. Dehnavi, "Dynamic MDS diffusion layers with efficient software implementation," *Int. J. Appl. Cryptography*, vol. 4, no. 1, pp. 36–44, 2020, doi: 10.1504/ijact.2020.107164.

[36] L. Tran Thi, "Fast computation of direct exponentiation to speed up implementation of dynamic block ciphers," *J. Comput. Sci. Cybern.*, vol. 38, no. 4, pp. 365–375, Dec. 2022, doi: 10.15625/1813-9663/38/4/17226.

[37] 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.

[38] V. Sawant, A. Solkar, and R. Mangrulkar, "Modified symmetric image encryption approach based on mixed column and substitution box," *J. Appl. Secur. Res.*, vol. 19, no. 2, pp. 196–229, Apr. 2024, doi: 10.1080/19361610.2022.2150498.

[39] T. Xu, F. Liu, and C. Wu, "A white-box AES-like implementation based on key-dependent substitution-linear transformations," *Multimedia Tools Appl.*, vol. 77, no. 14, pp. 18117–18137, Jul. 2018, doi: 10.1007/s11042-017-4562-8.

[40] A. Belazi, A. A. Abd El-Latif, A.-V. Diaconu, R. Rhouma, and S. Belghith, "Chaos-based partial image encryption scheme based on linear fractional and lifting wavelet transforms," *Opt. Lasers Eng.*, vol. 88, pp. 37–50, Jan. 2017, doi: 10.1016/j.optlaseng.2016.07.010.

[41] B. B. Cassal-Quiroga and E. Campos-Cantón, "Generation of dynamical S-boxes for block ciphers via extended logistic map," *Math. Problems Eng.*, vol. 2020, pp. 1–12, Mar. 2020, doi: 10.1155/2020/2702653.

[42] H. R. Shakir, S. A. A. Mehdi, and A. A. Hattab, "A dynamic S-box generation based on a hybrid method of new chaotic system and DNA computing," *TELKOMNIKA Telecommunication Comput. Electron. Control*, vol. 20, no. 6, pp. 1230–1238, Dec. 2022, doi: 10.12928/telkomnika.v20i6.23449.

[43] 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 Exp. Lett. B, Appl., Int. J. Res. Surv.*, vol. 10, no. 9, pp. 823–832, 2019, doi: 10.24507/icicelb.10.09.823.

[44] A. I. Salih, A. M. Alabaichi, and A. Y. Tuama, ''Enhancing advance encryption standard security based on dual dynamic XOR table and Mix-Columns transformation,'' *Indonesian J. Electr. Eng. Comput. Sci.*, vol. 19, no. 3, p. 1574, Sep. 2020, doi: 10.11591/ijeecs.v19.i3.pp1574-1581.
[45] R. A. Fisher and F. Yates, *Statistical Tables for Biological, Agricultural and Medical Research*, Ra Fisher and F. Yates, Eds. Edinburgh, Scotland: Oliver and Boyd, 1963.
[46] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, ''A statistical test suite for random and pseudorandom number generators for cryptographic applications,'' Booz-Allen Hamilton Inc., McLean, VA, USA, Tech. Rep., 2001, vol. 22.
[47] M. O. Saarinen, ''SP 800–22 and GM/T 0005–2012 tests: Clearly obsolete, possibly harmful,'' in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops (EuroS&PW)*, Jun. 2022, pp. 31–37, doi: 10.1109/EuroSPW55150.2022.00011.
[48] M. Matsui, ''Linear cryptanalysis method for DES cipher,'' in *Proc. Workshop Theory Appl. Cryptograph. Techn. (EUROCRYPT)*, Lofthus, Norway. Berlin, Germany: Springer, 1994, pp. 386–397.
[49] H. M. Heys, ''A tutorial on linear and differential cryptanalysis,'' *Cryptologia*, vol. 26, no. 3, pp. 189–221, Jul. 2002.
[50] E. Biham and A. Shamir, ''Differential cryptanalysis of DES-like cryptosystems,'' *J. Cryptol.*, vol. 4, no. 1, pp. 3–72, Jan. 1991.
[51] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4613-9314-6
[52] L. T. Keliher, ''Linear cryptanalysis of substitution-permutation networks,'' Ph.D. thesis, Queen's Univ., Canada, 2003. [Online]. Available: https://dl.acm.org/doi/10.5555/1037576
[53] J. Soto, ''Randomness testing of the advanced encryption standard candidate algorithms,'' U.S. Dept. Commerce, Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. NISTIR 6390, pp. 1-9, Sep. 1999.
[54] A. Doğanaksoy, B. Ege, O. Koçak, and F. Sulak, ''Cryptographic randomness testing of block ciphers and hash functions,'' Cryptol. ePrint Arch., Tech. Paper 2010/564, pp. 1–12, Jan. 2010.
[55] F. Sulak, ''Statistical analysis of block ciphers and hash functions,'' Ph.D. thesis, Middle East Technical Univ., 2011. [Online]. Available: http://etd.lib.metu.edu.tr/upload/12613045/index.pdf
[56] F. Sulak, A. Doğanaksoy, B. Ege, and O. Koçak, ''Evaluation of randomness test results for short sequences,'' in *Proc. Int. Conf. Sequences Their Appl.* Berlin, Germany: Springer, 2010, pp. 309–319, doi: 10.1007/978-3-642-15874-2_27.
[57] L. R. Knudsen, ''Practically secure Feistel ciphers,'' in *Proc. Int. Workshop Fast Softw. Encryption*. Berlin, Germany: Springer, 1993, pp. 211–221.
[58] K. Sakamura, W. X. Dong, and H. Ishikawa, ''A study on the linear cryptanalysis of AES cipher,'' Okayama Univ. Fac. Environ. Sci. Technol. Res., Okayama, Japan, Tech. Rep. 9, 2004, pp. 19–26. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-36159-6_29
[59] S. D. Mansoori and H. K. Bizaki, ''On the vulnerability of simplified AES algorithm against linear cryptanalysis,'' *Int. J. Comput. Sci. Netw. Secur.*, vol. 7, no. 7, pp. 257–263, 2007.
[60] L. E. Bassham III, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, and S. Vo, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, document NIST SP 800-22 Rev. 1, Nat. Inst. Standards Technol., Gaithersburg, MD, USA, 2010.
[61] J. Soto and L. E. Bassham, ''Randomness testing of the advanced encryption standard finalist candidates,'' U.S. Dept. Commerce, Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. ADA393385, 2000, pp. 1–4.
[62] E. Filiol, ''A new statistical testing for symmetric ciphers and hash functions,'' in *Proc. 4th Int. Conf. Inf. Commun. Security (ICICS)*, Singapore. Berlin, Germany: Springer, Dec. 2002, pp. 342–353. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-36159-6_29
[63] V. Katos, ''A randomness test for block ciphers,'' *Appl. Math. Comput.*, vol. 162, no. 1, pp. 29–35, Mar. 2005.
[64] A. Doğanaksoy, B. Ege, O. Koçak, and F. Sulak, ''Statistical analysis of reduced round compression functions of SHA-3 second round candidates,'' Cryptol. ePrint Arch., Tech. Paper 2010/611, pp. 1–10, Jan. 2010.
[65] A. Kaminsky, ''Testing the randomness of cryptographic function mappings,'' Cryptol. ePrint Arch., Tech. Paper 2019/078, pp. 1–25, Jan. 2019.
[66] E. Bellini and Y. J. Huang, ''Randomness testing of the NIST light weight cipher finalist candidates,'' in *Proc. NIST Lightweight Cryptogr. Workshop*, 2022, pp. 1–26.

TRAN THI LUONG received the bachelor's degree from Hanoi University of Science, Hanoi, Vietnam, in 2006, and the master's and Ph.D. degrees in cryptographic technique from the Academy of Cryptography Techniques, Hanoi, in 2012 and 2019, respectively.

She was the Chief Investigator of the Project ''Constructions of MDS and Dynamic MDS Matrices for the Dynamic Diffusion Layer of Block Ciphers'' (2014–2015) and an Investigator of the Project ''Construction of Cryptographic Primitives for Digital Signature Schemes and Key Exchange Protocols using Public Key Cryptography'' (2018–2020) and the Project ''Research on Constructing a Secure and Efficient Dynamic SPN Block Cipher Algorithm'' (2022–2023). Her research interests include cryptography, coding theory, and information security. She served as the Co-Chair for special session of KSE 2023 and the *PeerJ Computer Science* Reviewer.

NGUYEN NGOC CUONG received the degree and Ph.D. degrees from Hanoi National University, Hanoi, Vietnam, in 1972 and 1984, respectively. He is currently pursuing the Ph.D. degree in mathematics. His research interests include computing mathematics and cryptographic science.

BAY VO received the Ph.D. degree in computer science from the University of Science, Vietnam National University, Ho Chi Minh City, Vietnam, in 2011. He is currently an Associate Professor with Ho Chi Minh City University of Technology, Vietnam. His research interests include association rules, classification, mining in the incremental database, distributed databases and privacy-preserving in data mining, and soft computing. He also served as the Co-Chair for several special sessions, such as ICCCI, ACIIDS, KSE 2013 and 2014, and SMC 2015; and a Reviewer for many international journals, such as IEEE Transactions on Knowledge and Data Engineering, *Knowledge and Information Systems*, *Expert Systems with Applications*, IEEE Transactions on Systems, Man, and Cybernetics, *Information Sciences*, *Knowledge-Based Systems*, *Soft Computing*, *PLOS One*, and IEEE Access. He serves as an Associate Editor for the *Journal of Intelligent and Fuzzy Systems* and the Managing Editor for *Vietnam Journal of Computer Science*.