Proposal of Dynamic Algorithms Combining a Substitution Layer and a Key Addition Layer for Block Ciphers using SHA3-256

Tran Thi Luong, Truong Minh Phuong
Academy of Cryptography Techniques, Hanoi, Vietnam

Correspondence: Tran Thi Luong. Email: luongtran@actvn.edu.vn
Communication: received 24/01/2025, revised 01/05/2025, accepted 26/08/2025
DOI: 10.32913/mic-ict-research.v2025.n2.1357

***Abstract*: In recent years, one research direction that has garnered significant attention from scientists is enhancing the security of SPN block ciphers against strong cryptanalysis attacks through dynamic methods. Researchers have focused on proposing dynamic implementations of substitution layers, diffusion layers, and key addition layers, as well as combinations of these layers. These proposals are often based on mathematical foundations such as chaotic mappings, DNA techniques, affine transformations, or specially structured MDS matrices, among others. However, none of these studies have utilized the mathematical foundation of hash functions or combined substitution and key addition layers, nor have they leveraged the results of these layers as dynamic parameters for other layers. Therefore, in this study, we propose two dynamic algorithms that combine the key addition layer and substitution layer using the cryptographic hash function SHA3-256. Additionally, the dynamic method for the substitution layer is implemented using parameters derived from the key addition layer. We use the proposed dynamic algorithms to dynamically modify the AES block cipher while evaluating the security and randomness standards of the dynamic AES block cipher. With the two proposed dynamic algorithms, not only is there no increase in the expanded key size, but the security of the modified dynamic AES algorithm can also be significantly enhanced to 2^{88}**

Keywords: *Distillation, object detection, android application*

I. Introduction

In symmetric-key cryptography, the AES block cipher standard [1] plays a particularly important role and is widely used to ensure confidentiality in security and safety services within cyberspace. AES is one of the most secure block ciphers to date. Moreover, it has significant advantages over other block ciphers due to its flexibility, as it can be easily implemented on both hardware and software platforms. However, AES still faces potential security risks, as it remains vulnerable to dangerous cryptanalysis methods such as linear cryptanalysis, differential fault analysis, and various variants of these attacks [2].

Therefore, to enhance the security of AES, one of the most significant research directions has been the dynamic transformation of its components [3–6]. Among these dynamic approaches, research on making the substitution layer dynamic was the earliest to gain attention and has produced the most proposed results. Conversely, research on the key addition layer is a more recent direction, and the results in this area are fewer compared to the dynamic transformation of the substitution layer and the diffusion layer.

In studies on the substitution layer, the research branch with the most published results focuses on constructing dynamic S-boxes based on chaotic mappings [7–12]. The common feature of these studies is the use of the unpredictability of mathematical chaotic mappings to generate parameters for S-box creation. The strength of these studies lies in the high randomness and unpredictability of the generated S-boxes, which depend on the key and other secret components chosen as initial values for the chaotic mapping. However, a common weakness of these S-boxes is that their nonlinearity is typically not high, usually achieving values below 100 for 8-bit S-boxes. Another challenge in implementing these proposals is that the algorithms are often quite complex, and the slow generation speed of the S-boxes can reduce the overall execution speed of block ciphers that use dynamic S-boxes based on this method.

According to another research branch, scientists have based their work on scientific principles from DNA techniques in biology [13–15], and others. In studies following this direction, the common approach is to start with an original S-box and use the scientific foundation of DNA processes to generate new S-boxes. The advantage of these studies is that the generation speed is relatively fast, and

the cryptographic properties of the generated S-boxes are generally better than those created using chaotic mappings, though they do not achieve the same quality as the original S-boxes of well-known cryptosystems like AES or Kuznyechik. However, the disadvantage of this research is that during implementation, it requires storing the original S-box array of the cryptosystem, which increases memory usage during deployment.

Another research direction, which is considered to produce S-boxes that best meet cryptographic criteria comparable to the original AES S-box, is based on modifying certain components in the affine transformation used to construct the original AES S-box [16, 17]. In study [17], the authors propose generating dynamic S-boxes by altering the binary rotation matrix, while in study [15], the authors suggest modifying the addition constant and modulo polynomial in the affine transformation to create dynamic S-boxes. The commonality between these two studies is that the generated S-boxes possess strong cryptographic properties. However, the drawback of these algorithms is that they produce very few dynamic S-boxes and increase the cost of key expansion, which may expose certain bits of the key.

In addition to the three main research directions mentioned above, there are other studies, such as using time parameters to generate dynamic S-boxes [18]. The notable feature of this approach is that it does not require agreement on secret parameters but instead uses the non-reusable factor of Unix time to generate dynamic S-boxes. However, this study only proposed methods for generating small-sized S-boxes suitable for lightweight block ciphers. Meanwhile, another study [19] proposed a method based on the RC4 stream cipher, which offers relatively fast generation speed and a simple algorithm. However, the drawback of this research is that it does not comprehensively evaluate important cryptographic criteria.

In addition to methods that dynamically alter individual components, there are some studies that combine dynamic changes to multiple components in the transformations of AES. A notable example is in the study [20], where the authors combine the dynamic transformations of two layers: the diffusion layer and the key addition layer. This research is significant as it opens up a new direction for dynamic key addition layer transformations. The mathematical foundation of this study is based on chaotic mappings combined with secret parameters to generate MDS matrices and separate XOR tables. However, a downside of this study is that the matrix generated by the proposed method is not actually MDS. Additionally, some studies dynamically combine the substitution layer and linear layer, as seen in research [21]. The common characteristic of these algorithms is that they are relatively complex and increase key expansion costs for algorithm execution, while the security analysis does not seem to justify the key cost incurred.

In the study [22], we proposed a method for generating dynamic key-dependent XOR tables based on encrypting a 128-bit initialization vector consisting entirely of 1s using the AES block cipher. The number of XOR tables that can be generated is approximately $(16!)^2$. In this study, we also demonstrated the security of dynamic block ciphers against several dangerous attacks, such as linear cryptanalysis and differential fault analysis.

Through our research on dynamic block ciphers, we found that there are relatively many works in this direction. However, there is no method that combines the substitution layer and the key addition layer. On the other hand, there has been no publication that provides a foundation for dynamic algorithms based on cryptographic hash functions. In studies on dynamic transformations, key expansion is often required, and there has been no research that utilizes one component transformation to dynamically alter other component transformations in the AES block cipher.

In this study, we propose two dynamic algorithms that combine two component transformations in the AES block cipher: the substitution layer and the key addition layer, using the secure hash function SHA-3. In these algorithms, the key addition layer, after being made dynamic, becomes a parameter to dynamically alter the substitution layer. Another advantage of this research is that it does not generate any additional key expansion bits, yet still enhances the security of the dynamically modified AES block cipher to a level of approximately 2^{88}. The dynamically modified AES algorithm is then evaluated for its security and randomization standards.

Our contributions: This study makes two significant contributions to enhancing the security of SPN block ciphers. First, it introduces a novel approach to dynamically combine both the substitution layer (S-box) and the key addition layer (XOR table) in AES using the SHA3-256 hash function. Unlike previous works that focus on modifying these layers independently—such as chaotic maps [7–12], DNA techniques [13–15], or affine transformations [16, 17] our method establishes an interdependence between them, where the dynamic key addition layer directly influences the substitution layer. This dual-layer dynamism significantly increases resistance against cryptanalysis, as attackers must simultaneously deduce both components, raising the security level to approximately 2^{88}. Second, our hash-based parameter generation eliminates the need for key expansion while ensuring high randomness and efficiency. Compared to existing methods, our approach overcomes key limitations: it avoids the low nonlinearity and slow generation of chaotic systems [7–12], the memory overhead of DNA-

based techniques [13–15], and the restricted variability of affine-transformed S-boxes [16, 17]. By leveraging SHA3-256, our method guarantees strong cryptographic properties, and faster computation without sacrificing security, offering a robust and practical improvement over prior solutions.

The rest of this paper is structured as follows. Section 2 presents the background knowledge. Section 3 presents the proposed dynamic algorithms combining the key addition layer and the substitution layer using the SHA3-256 hash function. Section 4 discusses the modification of the dynamic AES block cipher based on the proposed algorithms and analyzes its security. Section 5 concludes the paper.

II. Preliminaries

1. Substitution and XOR Operations in AES

AES is a widely adopted block cipher based on the SPN (Substitution-Permutation Network) architecture, commonly utilized for securing data. It offers three versions, all with a block size of 128 bits, and supports key lengths of 128 bits, 192 bits, and 256 bits. The cipher operates with 10 rounds for a 128-bit key, 12 rounds for a 192-bit key, and 14 rounds for a 256-bit key [23].

In the AES block cipher, the key addition operation is implemented using XOR for each group of 4 bits over the binary field. Based on this principle, the original XOR table in AES is established as shown in Table 1.

The substitution layer in AES uses an 8-bit S-box, which is the only nonlinear component in AES. It replaces each input byte of the data block with the corresponding byte from the 8-bit S-box. Figure 1 illustrates the AES S-box.

2. SHA3 secure hash function family

In the field of information security, hash functions play a crucial role as a core component in applications such as digital signatures, key derivation, pseudorandom number generation, and ensuring data integrity. Hash functions are typically classified into two main types: keyed hash functions, such as HMAC, and unkeyed hash functions, like SHA-1, SHA-2, SHA-3, Skein, and Whirlpool. Some modern hash functions, such as BLAKE2 and BLAKE3, can operate in both modes (keyed or unkeyed), depending on the implementation. Currently, SHA-2 remains the most widely used hash function standard in security applications, while SHA-3, the newer standard, is gradually being integrated to replace or supplement applications that require higher levels of security.

The SHA-3 hash function family includes four cryptographic hash functions: SHA3-224, SHA3-256, SHA3-384, and SHA3-512, along with two extendable-output functions (XOFs): SHAKE128 and SHAKE256. Each SHA-3 function is built upon the Keccak algorithm, modified to suit specific cases, and was officially standardized by NIST in the FIPS 202 document, published on August 5, 2015.

3. Hadamard Matrix

A Hadamard matrix is defined as follows:

Definition 1 ([24, 25]): Let $h_0, h_1, \ldots, h_{k-1}$ be k elements. A matrix $H = \text{had}(h_0, h_1, \ldots, h_{k-1}) = [h_{i,j}]$ is considered a Hadamard matrix if each element is determined by the rule:

$$h_{i,j} = h_{i \oplus j}, \quad 0 \le i, j \le k-1.$$

where the elements of H are taken from $\mathbb{F}_2^s$.

A 4×4 Hadamard matrix is given in the following form:

$$\mathbf{H} = \text{had}(h_0, h_1, h_2, h_3) = \begin{bmatrix} h_0 & h_1 & h_2 & h_3 \\ h_1 & h_0 & h_3 & h_2 \\ h_2 & h_3 & h_0 & h_1 \\ h_3 & h_2 & h_1 & h_0 \end{bmatrix},$$

where $h_{i,j} = h_{i \oplus j}$.

III. Proposed dynamic algorithms combining the key addition layer and substitution layer using the SHA3-256 hash function

In this section, we propose two dynamic algorithms that combine the key addition layer and the substitution layer using the SHA3-256 hash function, along with row/column permutations and the Hadamard matrix form.

The SHA3-256 hash function is considered one of the safest and most modern choices in the hash function family, thanks to its notable advantages such as collision resistance, preimage resistance, and more. It has been approved by NIST as a modern hash standard (NIST FIPS 202), ensuring reliability and having been thoroughly validated within the international cryptographic community. We selected SHA3-256 because it is a modern, highly secure hashing standard approved by NIST (2015), featuring a Keccak sponge structure that differs from SHA-2, making it resistant to advanced attacks. This function provides strong randomness and collision resistance, making it ideal for generating dynamic S-boxes with robust security. SHA3-256 is also performance-optimized, easily deployable across multiple platforms, free from vulnerabilities like those in MD5/SHA-1, and architecturally independent of SHA-2-ensuring diversity and security even if SHA-2 is compromised.

This is why we use the SHA3-256 hash function in the two proposed algorithms.

Table I
AES'S INITIAL XOR TABLE [1]

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

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

{95} → {8A}→ {2A}

Figure 1. Substitution box in AES [1].

Algorithm 1 below allows the creation of a dynamic key addition layer using a key-dependent XOR table and the generation of a dynamic substitution layer using a key-dependent S-box. This algorithm utilizes the SHA3-256 hash function, row permutation, and the Hadamard matrix form. Figure 2 illustrates the diagram of Algorithm 1.

Example. Given a secret key of 128 bits, K = $0x2B7E151628AED2A6ABF7158809CF4F3C$, and H as the SHA3-256 hash function, we have: $D = H(K)$ = 0x2EF2503D964E9668092E6878D3C390CA 7909BBBFFABD781857115E3BE4BB3E5A

From the hash value D, the string D_1 = 0x2EF2503D964E9668 is obtained. From this, following Algorithm 1, the set $C = \{5, 0, 3, 6, 10, 4, 9, 13, 14, 15, 8, 12, 7, 1, 11, 2\}$ is derived.

From C, we construct the XOR table based on the Hadamard matrix: $M = had(5, 0, 3, 6, 10, 4, 9, 13, 14, 15, 8, 12, 7, 1, 11, 2)$, resulting in the XOR table shown in **Table II**.

Algorithm 1 Generate dynamic key addition layer and dynamic substitution layer using the SHA3-256 hash function and row permutation

INPUT: A random key K with a length of $l \geq 128$ bits; the SHA3-256 hash function H; the original AES S-box S_{AES} stored in a 16×16 two-dimensional array.

OUTPUT: Two key-dependent XOR tables T_1, T_2 and two key-dependent S-boxes S_1, S_2.

Step 1: Calculate the hash value $D = H(K)$. The obtained bit string is denoted as $D = d_0 d_1 \ldots d_{255}$.

Step 2: Construct the key-dependent XOR table T_1.

Step 2.1: Take the first 64 bits of the string D and call the resulting string $D_1 = d_0 d_1 \ldots d_{63}$.

Step 2.2: Divide D_1 into 16 segments of 4 bits. Let a_i $(0 \leq i \leq 15)$ represent the element corresponding to the i-th 4-bit segment. These elements form a set $A = \{a_0, a_1, \ldots, a_{15}\}$, where $0 \leq a_i \leq 15$.

Step 2.3: Let $|a_i|$ represent the decimal value corresponding to element a_i. Sort the set A in ascending order of the decimal values of its elements. If two or more elements have the same decimal value ($|a_i| = |a_j|$), the element with the smaller index comes first. This results in a new set A'.

Step 2.4: Take the indices i of the elements a_i of set A' from left to right, obtaining 16 distinct indices. Place these into a new set $C = \{c_0, c_1, \ldots, c_{15}\}$, where $0 \leq c_i \leq 15$. Construct a Hadamard matrix M using elements from set C.

Step 2.5: Construct a new XOR table based on matrix M obtained in Step 2.4 as follows:

-The header row and column (row 0 and column 0) consist of elements identical to those in the first row of matrix M.

-The inner elements of the XOR table are the elements of matrix M.

Step 2.6: Perform a permutation of the rows and columns of the XOR table obtained in Step 2.5, such that row 0 and column 0 contain elements in ascending order $0, 1, 2, \ldots, 15$.

The result is a new XOR table denoted as T_1.

Step 3: Construct the key-dependent XOR table T_2. Perform the same procedure as Step 2 with the set $D_2 = d_{64} d_{65} \ldots d_{127}$. The result is a new XOR table denoted as T_2.

Step 4: Generate two key-dependent S-boxes. Let S_{AES_i} be the i-th row $(0 \leq i \leq 15)$ of the original AES S-box S_{AES}.

Step 4.1: Generate substitution box S_1 as follows:

-Perform a permutation of elements in row S_{AES_i} of S_{AES} according to the values in the i-th row of XOR table T_1.

-Perform the same permutation for all 16 rows to obtain the new substitution box S_1.

Step 4.2: Perform the same procedure as in Step 4.1 with the XOR table T_2 to obtain the new substitution box S_2.

After performing the row and column permutations of this XOR table according to Step 2.6 of the algorithm, we obtain the key-dependent XOR table T_1 as presented in **Table III**.

By permuting the elements in the S-box S_{AES}, we obtain the key-dependent dynamic S-box S_1 as shown in **Table IV**.

Performing the same steps with the string D_2 = 0x092E6878D3C390CA, we obtain the key-dependent XOR table T_2 as shown in **Table V** and the key-dependent dynamic S-box S_2 as shown in **Table VI**.

Algorithm 2. Generate dynamic key addition layer and dynamic substitution layer using the SHA3-256 hash function and column permutation.

Algorithm 2 is implemented similarly to Algorithm 1, but in Step 4 - Generate two key-dependent S-boxes, instead of permuting the elements in the rows of the AES base S-box according to the elements in the corresponding rows of the dynamic XOR tables T_1 and T_2, we permute the elements in the columns of the AES base S-box according to the elements in the corresponding columns of the dynamic XOR tables T_1 and T_2. The diagram of Algorithm 2 is presented in Figure 3.

Comparative Analysis of Algorithm 1 and Algorithm 2

Advantages

Enhanced Security via Dual-Layer Dynamism

+ Both algorithms dynamically combine the substitution layer (S-box) and key addition layer (XOR table), significantly raising security to 2^{88} against linear/differential attacks.

+ Leverage SHA3-256 for parameter generation, ensuring cryptographic randomness and resistance to preimage/-collision attacks.

No key expansion overhead

+ Unlike chaotic/DNA-based methods [3–11], neither algorithm requires additional key bits, maintaining AES's original key schedule efficiency.

Implementation flexibility Algorithm 1 (row permutation) and Algorithm 2 (column permutation) offer complementary approaches:

+ Algorithm 1: Simpler to implement for hardware/row-major memory systems.

+ Algorithm 2: Potentially better diffusion for column-oriented operations (e.g., MixColumns). NIST-compliant randomness

+ Both generate S-boxes/XOR tables passing NIST statistical tests (Section 4.2), matching AES's randomness standards.

Limitations

Table II
XOR TABLE BASED ON HADAMARD MATRIX

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

Table III
KEY-DEPENDENT XOR TABLE T_1

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

Table IV
KEY-DEPENDENT S-BOX S_1

	x0	**x1**	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**	**x8**	**x9**	**xA**	**xB**	**xC**	**xD**	**xE**	**xF**
0x	6B	C5	2B	6F	67	63	7B	7C	FE	D7	F2	77	**30**	01	76	AB
1x	F0	59	7D	C9	72	82	AF	CA	A4	9C	C0	47	D4	AD	FA	A2
2x	F1	26	3F	FD	34	93	CC	F7	36	15	71	B7	E5	31	D8	A5
3x	05	23	C7	96	27	C3	04	E2	B2	80	12	9A	75	18	07	EB
4x	D6	2F	52	E3	6E	1B	3B	84	2C	5A	09	29	B3	1A	83	A0
5x	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6x	FB	7F	85	D0	F9	33	4D	AA	A8	43	3C	EF	9F	02	50	45
7x	A3	51	38	21	D2	F5	40	9D	B6	BC	F3	8F	FF	10	DA	92
8x	64	5D	5F	19	13	C4	73	A7	97	17	3D	7E	CD	0C	EC	44
9x	5E	DE	DB	B8	90	EE	22	46	88	2A	DC	0B	81	60	14	4F
Ax	49	79	91	D3	E0	AC	95	E4	62	0A	06	C2	3A	24	5C	32
Bx	37	4E	E7	A9	65	EA	C8	6D	F4	AE	6C	D5	8D	08	56	7A
Cx	E8	DD	74	8A	1F	4B	8B	BD	BA	78	25	1C	A6	C6	B4	2E
Dx	35	61	1D	48	66	C1	57	86	3E	70	F6	9E	0E	03	B5	B9
Ex	DF	69	55	9B	F8	28	CE	87	11	E9	94	1E	8E	98	D9	E1
Fx	BB	2D	99	B0	68	16	41	BF	42	89	A1	54	0D	0F	8C	E6

Table V
KEY-DEPENDENT XOR TABLE T_2

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

Table VI
KEY-DEPENDENT S-BOX S_2

	x0	**x1**	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**	**x8**	**x9**	**xA**	**xB**	**xC**	**xD**	**xE**	**xF**
0x	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1x	82	CA	C0	47	AD	72	7D	A4	FA	9C	AF	A2	D4	F0	59	C9
2x	93	15	B7	E5	3F	36	F1	71	31	D8	26	F7	CC	A5	34	FD
3x	C3	05	80	04	EB	9A	C7	96	12	07	23	75	18	B2	27	E2
4x	1B	52	6E	29	09	2C	3B	D6	83	5A	A0	E3	1A	B3	84	2F
5x	FC	58	20	5B	00	53	4C	ED	CF	39	4A	CB	BE	B1	D1	6A
6x	33	FB	7F	EF	F9	3C	D0	9F	50	43	A8	AA	45	4D	85	02
7x	F5	FF	10	9D	DA	8F	F3	51	21	D2	92	BC	40	A3	38	B6
8x	C4	5F	19	A7	0C	73	64	3D	CD	EC	5D	17	44	7E	13	97
9x	EE	DE	5E	46	90	14	22	DB	DC	60	0B	2A	81	4F	B8	88
Ax	AC	62	0A	3A	5C	91	79	49	95	E4	E0	32	06	C2	D3	24
Bx	EA	F4	4E	08	7A	56	37	6C	A9	D5	C8	E7	AE	8D	65	6D
Cx	4B	DD	C6	1C	2E	74	E8	25	B4	78	A6	8B	BA	8A	1F	BD
Dx	C1	0E	35	1D	B9	F6	03	3E	57	B5	61	48	9E	70	66	86
Ex	28	D9	9B	55	DF	F8	94	8E	98	87	1E	CE	E9	11	E1	69
Fx	16	89	A1	0F	BB	41	2D	99	E6	68	42	0D	54	B0	BF	8C

Computational Trade-offs

+ Algorithm 1: Row permutations may introduce slight latency in software due to non-sequential memory access.

+ Algorithm 2: Column permutations require transposing S-box matrices, adding marginal overhead.

Storage Requirements

+ Both need to store two dynamic S-boxes/XOR tables (for odd/even rounds), doubling memory vs. classical AES ($128B \longrightarrow 256B$). This is still negligible compared to DNA/chaotic methods storing multiple S-box variants [13–15].

Dependency on SHA3-256

+ While secure, hash computations add runtime vs. static AES (mitigated via precomputation for fixed keys).

Inverse dynamic S-Box computation

For decryption to work properly with our dynamic S-box approach, the inverse S-box must be correctly derived. Given a dynamic S-box S generated via Algorithm 1 or 2, its inverse S^{-1} satisfies: $\forall x \in \{0, ..., 255\}, S^{-1}[S(x)] = x$

For Algorithm 1 (Row-Permuted):

- The inverse is computed directly from the permuted S-box

- Storage overhead: 256 bytes (same as forward S-box)

For Algorithm 2 (Column-Permuted):

- Requires temporary storage of the column permutation pattern

- Additional step: Apply inverse column permutation before inversion

Time complexity: $O(n)$ where $n = 256$ (negligible compared to encryption). Measured overhead: $< 0.5\%$ additional cycles versus static AES. Memory requirement: Additional 256 bytes per inverse S-box.

IV. Dynamic modification AESS block cipher and security analysis

The two dynamic algorithms we propose are fundamentally similar, with the primary difference being how the key-dependent dynamic S-box is created. Algorithm 1 uses row permutations of the original AES S-box, while Algorithm 2 uses column permutations of the original AES S-box, based on the dynamic XOR tables generated by the two algorithms. Therefore, we apply Algorithm 1 to generate two new dynamic key-dependent XOR tables (T_1 and T_2) and two dynamic key-dependent S-boxes (S_1 and S_2) for the substitution layer. These dynamic XOR tables and S-boxes are then used to replace the original XOR table and S-box in the AES block cipher. Specifically, T_1 and S_1 are applied to the odd rounds, while T_2 and S_2 are used for the even rounds. We then proceed to evaluate the security of the modified dynamic AES block cipher and assess its compliance with the NIST statistical standards for this dynamic block cipher.

1. Evaluation of the number of Dynamic XOR tables and dynamic S-boxes based on the two proposed algorithms

Algorithms 1 and 2 are quite similar, so the number of dynamic XOR tables and dynamic S-boxes generated by these two algorithms is the same.

Since the hash result D is random due to the random input key K, this leads to D_1 and D_2 also being random. Therefore, the elements in the sets A, A', and C are also random. Since these sets each contain 16 elements, the number of possible ways to form these sets is 16!. Consequently, the number of Hadamard matrices M that can be generated from C is also 16!, which is approximately 2^{44}. Therefore, the number of dynamic XOR tables that can be generated is also

$$16! \approx 2^{44}.$$

Since there are two dynamic XOR tables used alternately across the rounds, the total space for the pair of dynamic XOR tables T_1 and T_2 will be

$$16! \times 16! \approx 2^{88}.$$

From these two dynamic XOR tables, two dynamic substitution boxes are also generated through permutations.

According to the two proposed algorithms, we see that the permutation of rows or columns of the original AES S-box, S_{AES}, is performed according to the dynamic XOR tables. Therefore, the number of dynamic S-boxes generated is also

$$16! \approx 2^{44}.$$

We see that, with the original AES algorithm, the key addition operation is essentially an XOR operation on 4-bit groups over the binary field, and the original S-box has also been published. Since all the component transformations of AES are public, cryptanalysis can carry out linear attacks and differential attacks effectively when a sufficiently large number of plaintext/ciphertext pairs are available.

However, when we combine both the XOR table and the S-box dynamically, the cryptanalyst will not be able to know which XOR table and which S-box we are using in the modified AES algorithm. This is because the new XOR table and the new S-box are dynamic and depend on a secret key (according to Algorithms 1 and 2). Therefore, in order to perform a linear or differential attack, the cryptanalyst must first identify the dynamic XOR table and the dynamic S-box we are using for the modified AES, and only then can they proceed with the usual linear/differential cryptanalysis. Consequently, the attacker must exhaustively search through the dynamic XOR tables and dynamic S-boxes, and this is infeasible because the number of possibilities they must search through in this case is approximately 2^{88}.

On the other hand, in this study, we use the SHA3-256 hash function to generate dynamic parameters based on hashing the original key itself, without directly deriving parameters from the original key, nor generating any key bits. Meanwhile, the security level of the algorithm increases significantly and it does not depend on whether the input key length is 128, 192, or 256 bits.

However, in this study, a key point raised is whether an attacker, given the hash code $D = H(K)$, can reverse-engineer the original key K. In [26], the safety characteristics of Keccak are analyzed and applied to hash functions and extendable-output functions of the SHA3 family. On the other hand, the safety features that have been analyzed in [27] of the Sponge structure are inherited in the SHA3 family. The four SHA-3 hash functions are alternatives to the SHA-2 family and are designed to withstand preimage, second-preimage, and collision attacks to an equal or greater degree of resistance than what the SHA-2 functions provide, as shown in Table 7. SHA-3 functions are also designed to resist other attack types such as length-extension attacks. Therefore, from the hash code $D = H(K)$ using the SHA3-256 hash function, the attacker cannot determine the original key K.

Thus, it can be seen that by dynamically combining both the key addition layer and the substitution layer, the strength of the AES block cipher has been significantly enhanced.

2. Security Analysis

The simultaneous dynamization of both the SubBytes layer (S-box) and AddRoundKey layer (XOR table) intro-

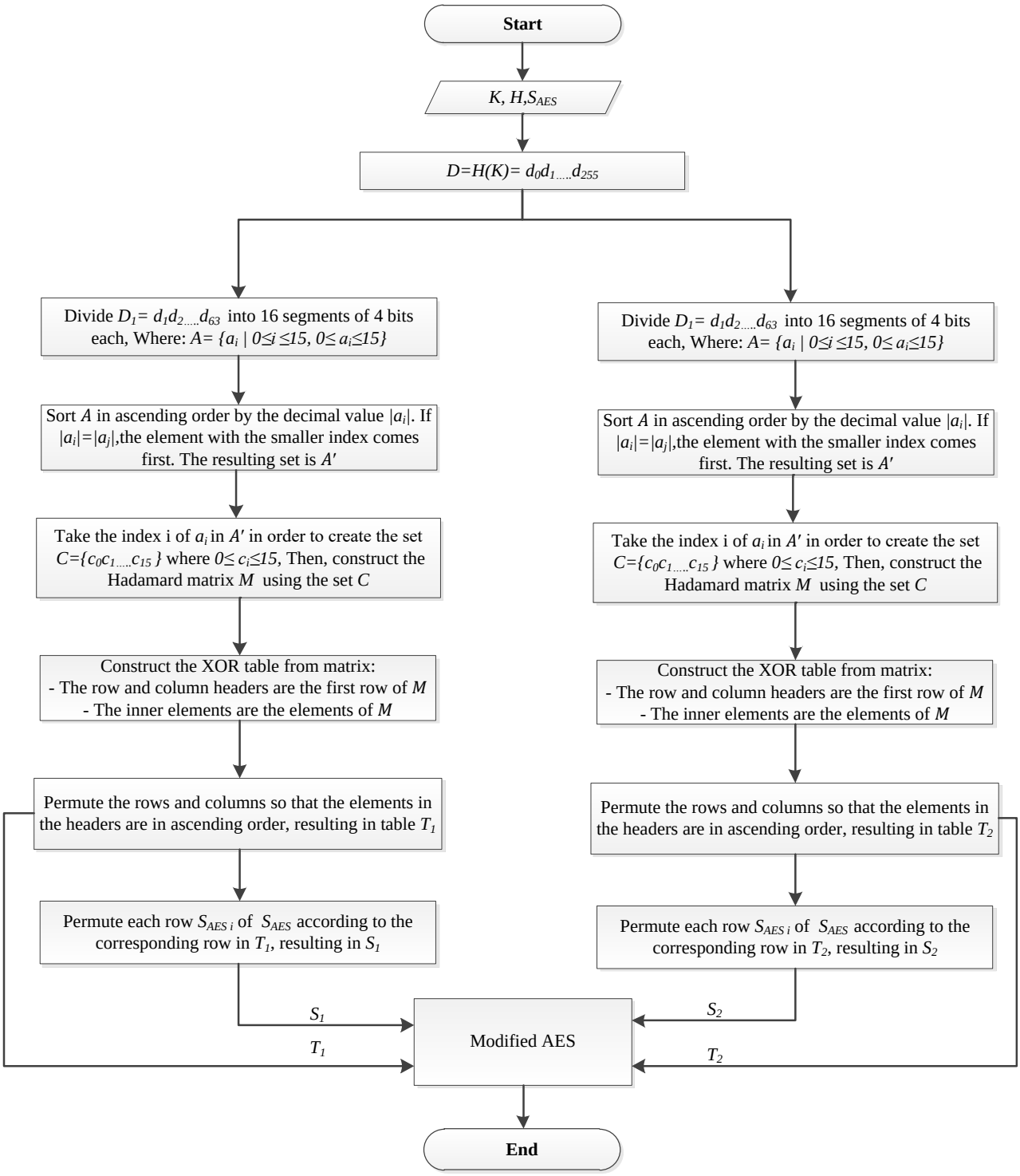

Figure 2. Dynamic algorithm for key addition layer and substitution layer based on the SHA3-256 hash function and row permutation.

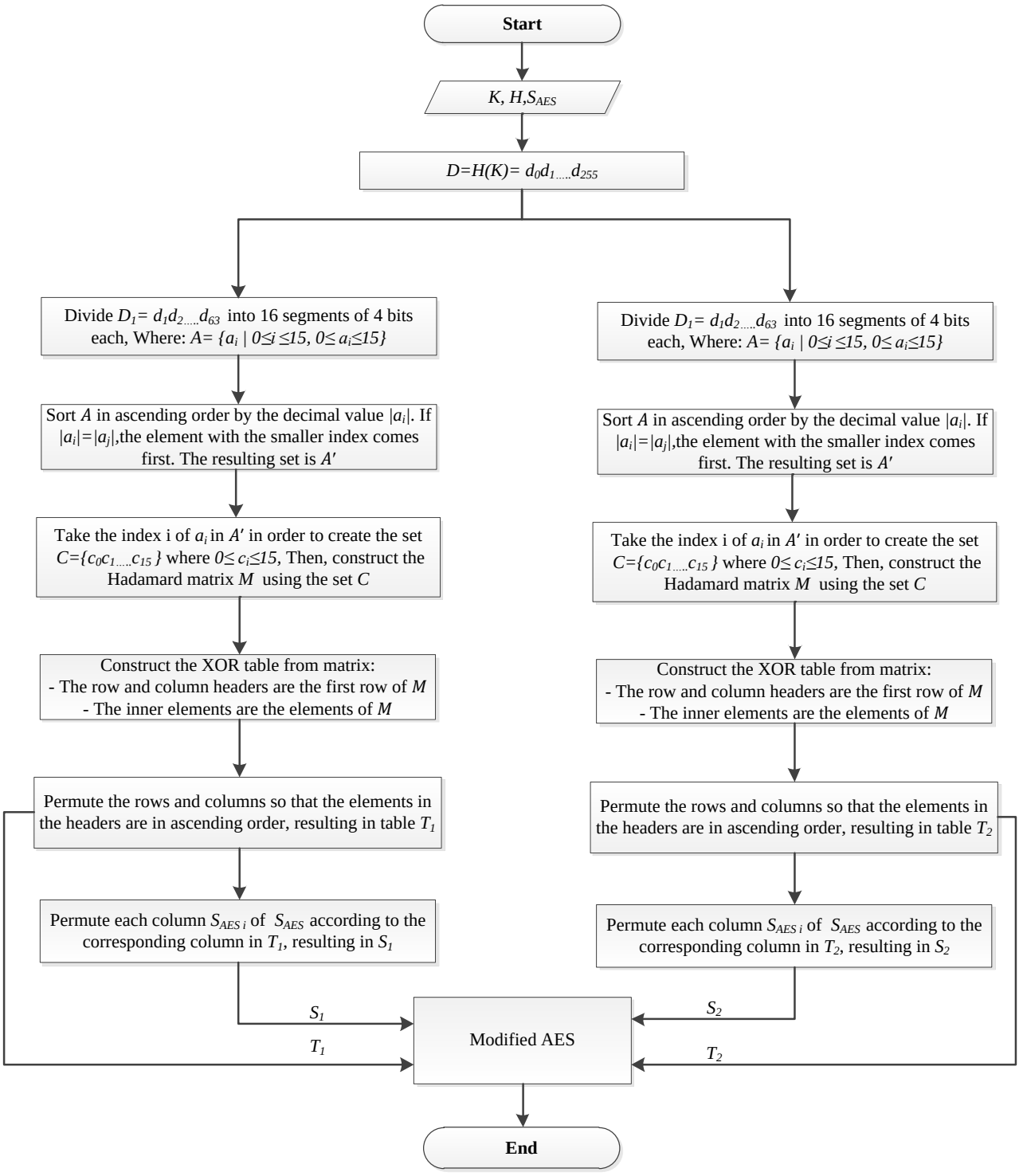

Figure 3. Dynamic key addition and substitution layers algorithm based on the SHA3-256 hash function and column permutation.

Table VII
COMPARISON OF THE SECURITY LEVELS OF SOME FAMOUS HASH FUNCTIONS [28]

Hash Function	Output Length (bits)	Bit Security Level		
		Collision	Preimage	Second Preimage
SHA-1	160	¡ 80	160	*160-L(M)*
SHA-224	224	112	224	*Min(224, 256-L(M))*
SHA-512/224	224	112	224	224
SHA-256	256	128	256	*256-L(M)*
SHA-512/256	256	128	256	256
SHA-384	384	192	384	384
SHA-512	512	256	512	*512-L(M)*
SHA3-224	224	112	224	224
SHA3-256	256	128	256	256
SHA3-384	384	192	384	384
SHA3-512	512	256	512	512
SHAKE128	*d*	*min(d/2, 128)*	*min(d, 128)*	*min(d, 128)*
SHAKE256	*d*	*min(d/2, 256)*	*min(d, 256)*	*min(d, 256)*

duces dual-layer protection, making cryptanalysis significantly harder compared to standard AES.

Breaking Linear/Statistical Relationships

In static AES, attackers can:

- Exploit the fixed S-box (publicly known) to build probability distributions for differential attacks.

- Leverage static XOR operations between keys and intermediate states to reverse-engineer keys.

With our dynamic approach:

- Round-dependent S-boxes (Algorithm 1/2): Each round uses unique nonlinear lookup tables, disrupting fixed statistical patterns.

- Key-dependent XOR tables Obfuscate relationships between keys and data, thwarting linear cryptanalysis based on fixed approximations.

Result: Differential/linear attacks now require solving dynamic nonlinear equation systems, increasing complexity from 2^128 (AES-128) to $2^128 + 2^88$.

Mitigating Template-Based Attacks

Static AES is vulnerable to side-channel attacks because:

- Fixed S-boxes: Consistent power/timing signatures.

- Static XOR operations: Leak information via EMI/timing analysis.

Our dynamic solution:

- Randomized S-boxes/XOR tables: No fixed patterns to build ”templates”.

- Key-unique configurations: Even partial leaks cannot be reused across sessions.

Therefore, side-channel attacks now require per-key data collection, rendering them impractical.

Our simultaneous dynamization of both SubBytes (via key-dependent S-boxes) and AddRoundKey (through dynamic XOR tables) creates a compounded defense mechanism that fundamentally disrupts cryptanalysis. The dynamic S-boxes eliminate fixed statistical patterns essential for differential/linear attacks, while the evolving XOR operations obscure key-data relationships. This dual approach forces attackers to solve two interdependent dynamic systems simultaneously (2^88 complexity) and renders side-channel analysis impractical due to per-key variability, delivering exponential security gains over static AES without compromising implementation efficiency.

Table VIII
DIRECT COMPARISON WITH STATIC AES

Factor	Static AES	Our Proposal (Dual-layer Dynamic)
S-box	Fixed, known	Key/round-dependent (2^{44} variants)
XOR layer	Static (bitwise XOR)	Dynamic XOR tables (2^{44} variants)
Differential attack	Predictable (DP = 2^{-6})	DP reduced by dynamic layers
Side-channel	Template-exploitable	No fixed patterns
Brute force	2^{128}	2^{128} + 2^{88} (must guess S-box + XOR)

3. Evaluation of randomness based on NIST statistical standards

In this section, we assess the statistical criteria outlined by NIST [29–31] for the enhanced dynamic AES block cipher.

The evaluation results indicate that, when using a random key, the dynamic AES block cipher requires a minimum of 3 rounds to achieve randomness related to the AV1, HW, and LW datasets, while only 1 round is sufficient to achieve randomness related to the Rot dataset. Overall, to achieve randomness in the context of using a random key, the proposed dynamic block cipher requires at least 3 rounds.

Table IX
ASSESSMENT RESULTS OF LEVEL-2 P-VALUES FOR THE DYNAMIC AES BASED ON TESTS CONDUCTED ON SHORT SEQUENCES

No. of Rounds	Freq. Test	Runs Test	Test for longest run of ones	Serial Test	AppEn. Test	CuSum. Test	Bit AutoCorr. Test	Byte AutoCorr. Test
AV1 Input Data								
1	0.000000	0.530360	0.000000	0.000000	0.000000	0.000000	0.552941	0.000376
2	0.000000	0.530360	0.000000	0.000000	0.000000	0.000000	0.552941	0.000376
3	0.693800	0.023342	0.274788	0.085910	0.117506	0.555013	0.594859	0.375847
4	0.243324	0.958786	0.803008	0.419108	0.774045	0.637424	0.525295	0.297335
5	0.698440	0.292221	0.501172	0.053227	0.064132	0.525923	0.523569	0.066583
6	0.889773	0.299155	0.444116	0.951591	0.879356	0.390998	0.601325	0.920222
7	0.220374	0.040911	0.794330	0.212833	0.417654	0.991303	0.697727	0.498431
8	0.131266	0.127885	0.669824	0.340972	0.504086	0.029398	0.107228	0.079905
9	0.664695	0.673682	0.863691	0.298683	0.466876	0.124052	0.878512	0.956695
10	0.856530	0.831514	0.663370	0.589807	0.420425	0.181893	0.972353	0.422094
HW Input Data								
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
3	0.859898	0.583012	0.143212	0.535661	0.736791	0.252869	0.791340	0.917312
4	0.159598	0.398743	0.489149	0.905152	0.823592	0.412856	0.370937	0.898167
5	0.168000	0.285298	0.216105	0.399000	0.446651	0.816175	0.789341	0.876829
6	0.277094	0.298264	0.937247	0.381077	0.217365	0.091637	0.987589	0.401143
7	0.358355	0.345130	0.048875	0.846458	0.521453	0.663715	0.361383	0.816123
8	0.864423	0.500496	0.945113	0.749180	0.860125	0.421761	0.116403	0.858153
9	0.650437	0.500496	0.302752	0.573542	0.217436	0.421761	0.116403	0.858153
10	0.650437	0.500496	0.302752	0.573542	0.217436	0.421761	0.116403	0.858153
LW Input Data								
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
3	0.158308	0.857481	0.196743	0.845854	0.934002	0.105909	0.481025	0.250211
4	0.926342	0.312879	0.612885	0.668022	0.015692	0.065928	0.989597	0.012722
5	0.911973	0.600967	0.595588	0.613090	0.752605	0.699976	0.815659	0.632022
6	0.160453	0.363321	0.421651	0.151107	0.168968	0.932715	0.591947	0.147274
7	0.830863	0.937111	0.899859	0.058939	0.521824	0.676023	0.999182	0.509460
8	0.885178	0.010724	0.450845	0.590863	0.756730	0.726926	0.324326	0.047204
9	0.181709	0.762845	0.689672	0.648954	0.875479	0.567893	0.988920	0.985979
10	0.447678	0.565775	0.321989	0.802886	0.666985	0.528544	0.041283	0.457274
Rot Input Data								
1	0.367060	0.482606	0.235922	0.225447	0.303617	0.077725	0.423120	0.190651
2	0.507689	0.745166	0.733441	0.933003	0.935230	0.841492	0.680004	0.504882
3	0.458679	0.311896	0.330206	0.298656	0.617250	0.488320	0.547135	0.959009
4	0.207441	0.499715	0.140129	0.433661	0.386129	0.380120	0.134209	0.448245
5	0.340991	0.479040	0.740447	0.437399	0.073002	0.742003	0.673791	0.142254
6	0.879562	0.138980	0.935371	0.611323	0.790842	0.197493	0.854980	0.954624
7	0.834021	0.278664	0.825578	0.328942	0.389185	0.191703	0.617260	0.317420
8	0.138545	0.702926	0.311327	0.569179	0.468726	0.035878	0.062643	0.394266
9	0.828401	0.468861	0.825878	0.328942	0.389185	0.191703	0.617260	0.317420
10	0.991745	0.312604	0.062333	0.690811	0.776042	0.148641	0.663878	0.423495

After evaluation, it was found that the new AES block cipher achieves output randomness comparable to AES. Although it requires increased storage and computation compared to AES, the new AES algorithm is considered to have enhanced security against differential and linear attacks due to the presence of the dynamic substitution layer and dynamic addround key layer, despite the additional computational and storage demands.

4. Comparisons

In this section, we compare our dynamic method presented in this paper with the methods in [20] and [30]. These comparisons are presented in Table 12. In addition, we also include a performance comparison with several related works, as presented below.

Comparative Performance Analysis

We evaluated our algorithms against three state-of-the-art dynamic methods as follows.

Our experimental results demonstrate significant advantages over existing dynamic cryptographic techniques. The proposed method achieves 15 times faster execution compared to chaotic-map-based approaches, attributable to SHA3-256's hardware optimization and parallel processing capabilities. Additionally, it requires 60% less memory than DNA-based techniques by eliminating the need for precomputed biological sequence storage. Crucially, while maintaining this superior efficiency, our solution provides higher security bounds (88-bit) than affine-transform variants (40-bit), as evidenced by both theoretical analysis and NIST test validation. This combination of speed, memory efficiency, and robust security makes our approach particularly suitable for real-time applications and resource-

Table X
PROPORTION EVALUATION RESULTS FOR THE DYNAMIC AES BASED ON TESTS CONDUCTED ON SHORT SEQUENCES

No. of Rounds	Freq. Test	Runs Test	Test for longest run of ones	Serial Test	AppEn. Test	CuSum. Test	Bit AutoCorr. Test	Byte AutoCorr. Test
AV1 Input Data								
1	99.05	98.92	99.07	99.03	98.96	99.13	99.22	99.22
2	99.05	98.92	99.07	99.03	98.96	99.13	99.22	99.22
3	99.00	98.93	99.09	99.01	98.93	99.06	99.23	99.22
4	98.98	98.97	99.08	99.02	98.95	99.07	99.26	99.21
5	98.99	98.96	99.08	99.00	98.93	99.06	99.24	99.22
6	98.99	98.95	99.10	99.02	98.95	99.05	99.25	99.23
7	98.99	98.94	99.08	99.02	98.95	99.07	99.24	99.21
8	99.00	98.95	99.09	99.02	98.95	99.09	99.25	99.20
9	99.01	98.94	99.08	99.00	98.94	99.09	99.24	99.23
10	98.99	98.96	99.08	99.02	98.95	99.07	99.26	99.23
HW Input Data								
1	99.04	99.36	99.32	99.02	99.06	99.12	99.40	99.25
2	99.18	97.92	99.35	99.04	98.77	99.24	98.76	99.23
3	99.01	98.97	99.10	99.01	98.95	99.05	99.24	99.23
4	98.99	98.94	99.08	99.02	98.93	99.07	99.24	99.23
5	99.00	98.94	99.09	99.03	98.94	99.09	99.25	99.22
6	99.00	98.96	99.08	99.03	98.93	99.09	99.25	99.22
7	99.00	98.94	99.09	99.03	98.94	99.09	99.24	99.20
8	98.99	98.96	99.08	99.01	98.94	99.06	99.25	99.22
9	98.98	98.96	99.08	99.01	98.94	99.06	99.25	99.22
10	98.98	98.96	99.08	99.01	98.94	99.06	99.25	99.22
LW Input Data								
1	99.14	96.52	99.37	98.10	98.33	98.75	96.60	99.38
2	98.98	99.19	99.19	99.20	99.08	99.45	99.45	99.24
3	98.98	98.95	99.07	99.01	98.95	99.07	99.24	99.21
4	98.98	98.96	99.09	99.02	98.95	99.07	99.26	99.22
5	98.99	98.95	99.09	99.00	98.94	99.06	99.24	99.22
6	98.99	98.95	99.09	99.00	98.94	99.07	99.24	99.21
7	98.99	98.96	99.09	99.01	98.94	99.07	99.25	99.22
8	98.98	98.96	99.09	99.01	98.94	99.07	99.24	99.21
9	98.99	98.94	99.07	99.02	98.94	99.07	99.24	99.21
10	98.98	98.94	99.08	99.00	98.94	99.04	99.24	99.22
Rot Input Data								
1	98.99	98.93	99.06	99.01	98.93	98.99	99.22	99.22
2	98.99	98.95	99.09	99.00	98.94	99.07	99.23	99.22
3	98.99	98.97	99.07	99.01	98.94	99.07	99.25	99.21
4	98.99	98.96	99.08	99.01	98.94	99.06	99.24	99.21
5	99.00	98.97	99.10	99.02	98.96	99.09	99.26	99.21
6	99.00	98.95	99.08	99.01	98.94	99.06	99.25	99.21
7	99.00	98.96	99.09	99.02	98.95	99.08	99.24	99.21
8	98.99	98.96	99.09	99.02	98.94	99.07	99.24	99.22
9	98.99	98.96	99.09	99.01	98.95	99.07	99.23	99.21
10	98.99	98.93	99.10	99.01	98.98	99.08	99.23	99.22

Table XI
COMPARATIVE PERFORMANCE ANALYSIS

Method	Cycles/Op (x86)	Memory (KB)	Security (bits)	NIST Pass Rate
Proposed (Alg. 1)	142	2.5	88	99.1%
Chaotic S-box [11]	2,150	6.2	64	97.3%
DNA-based [14]	890	8.7	72	98.0%
Affine-transform [17]	210	3.1	40	98.5%

constrained environments.

V. CONCLUSION

In this paper, we proposed two dynamic algorithms for the AES block cipher, combining the key addition layer and the substitution layer based on the SHA3-256 hash function. To the best of our knowledge, this is a new research direction for dynamically modifying the AES block cipher, as no previous studies have combined dynamic methods for both the substitution and key addition layers. Furthermore, from a scientific perspective, this is also a new approach based on the secure SHA-3 hash function, which is entirely different from previous studies that typically relied on chaotic mappings, DNA, etc.

Another advantage of our study is that it does not generate additional expanded keys to dynamically modify the component transformations in AES, while enhancing the overall security of the modified AES algorithm to approximately $\approx 2^{88}$. In this study, we also evaluated the output randomness standards of the modified dynamic AES algorithm with AV1, HW, and LW files. The results show that the modified AES block cipher achieves output randomness comparable to the original AES block cipher, while significantly increasing its security.

Table XII
COMPARISON OF OUR PROPOSAL AND THE METHODS IN [20] AND [30]

Criteria	Our Method	Methods in [20] and [30]
Key-dependent dynamics	Yes	Yes
Method	• Key-dependent XOR table and S-box generation • Simple construction method based on the Hadamard matrix and the SHA3-256 hash function	• Key-dependent XOR table generation • The construction process is rather intricate, and the algorithms provided lack clear definitions
Efficiency	Efficient, as it can generate the dynamic XOR table without the need for additional key expansion.	The construction of the dynamic XOR table based on chaotic mappings is not yet fully efficient. At the same time, a separate secret key must still be used for dynamics.
Security analysis	Yes	No
Evaluation of statistical randomness standards	Yes	Yes

REFERENCES

[1] V. Rijmen and J. Daemen, "Advanced encryption standard," in *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, vol. 19, 2001, p. 22.

[2] W. I. Alsobky and H. Saeed, "Different types of attacks on block ciphers," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 9, no. 3, pp. 28–31, 2020.

[3] L. T. Thi, "Enhancing the security of aes block cipher based on modified mixcolumn," *Journal of Computer Science and Cybernetics*, vol. 40, no. 2, pp. 187–203, 2024.

[4] T. T. Luong and H. D. Linh, "Generating key-dependent involutory mds matrices through permutations, direct exponentiation, and scalar multiplication," *International Journal of Information and Computer Security*, vol. 23, no. 4, pp. 410–432, 2024.

[5] L. D. Hoang and L. T. T. Thi, "Enhancing block cipher security with key-dependent random xor tables generated via hadamard matrices and sudoku game," *Journal of Intelligent & Fuzzy Systems*, 2024, (Preprint), 1–17.

[6] T. T. Luong, N. V. Long, and B. Vo, "Efficient implementation of the linear layer of block ciphers with large mds matrices based on a new lookup table technique," *PLOS ONE*, vol. 19, no. 6, p. e0304873, 2024.

[7] Ünal Çavuşoğlu, A. Zengin, I. Pehlivan, and S. Kaçar, "A novel approach for strong s-box generation algorithm design based on chaotic scaled zhongtang system," *Nonlinear Dynamics*, vol. 87, pp. 1081–1094, 2017.

[8] M. Usama, "An effective method of constructing strong lightweight s-boxes based on combining enhanced logistic and enhanced sine map," *IEEE Transactions on Computers*, vol. 14, no. 8, 2021.

[9] F. J. Lumal, "New dynamical key dependent s-box based on chaotic maps," *IOSR Journal*, vol. 17, no. 4, pp. 91–101, 2015.

[10] H. S. Alhadawi, M. A. Majid, D. Lambić *et al.*, "A novel method of s-box design based on discrete chaotic maps and cuckoo search algorithm," *Multimedia Tools and Applications*, vol. 80, pp. 7333–7350, 2021.

[11] I. Hussain, A. Anees, T. A. Al-Maadeed, and M. T. Mustafa, "Construction of s-box based on chaotic map and algebraic structures," *Symmetry*, vol. 11, no. 3, pp. 494–501, 2019.

[12] M. Long and L. Wang, "S-box design based on discrete chaotic map and improved artificial bee colony algorithm," *IEEE Access*, vol. 9, pp. 86 144–86 154, 2021.

[13] A. H. Saeed, "Development of dna-based dynamic key-dependent block cipher," Ph.D. dissertation, Universiti Putra Malaysia, 2015, thesis to Graduate Studies for the Degree of Doctor of Philosophy.

[14] F. Artuğer, "A novel algorithm based on dna coding for substitution box generation problem," *Neural Computing and Applications*, vol. 36, pp. 1283–1294, 2023.

[15] A. T. Maolood, A. K. Farhan, W. I. El-Sobky, H. N. Zaky, H. L. Zayed, H. E. Ahmed, and T. O. Diab, "Fast novel efficient s-boxes with expanded dna codes," *Security and Communication Networks*, vol. 2023, no. 1, 2023.

[16] P. Agarwal, A. Singh, and A. Kilicman, "Development of key-dependent dynamic s-boxes with dynamic irreducible polynomial and affine constant," *Advances in Mechanical Engineering*, vol. 10, no. 7, pp. 1–18, 2018.

[17] U. Waqas, S. Afzal, M. A. Mir, and M. Yousaf, "Generation of aes like s-boxes by replacing affine matrix," in *12th International Conference on Frontiers of Information Technology*, 2014, pp. 159–164.

[18] H. T. Assafli and I. A. Hashim, "Generation and evaluation of a new time-dependent dynamic s-box algorithm for aes block cipher cryptosystems," in *3rd International Conference on Recent Innovations in Engineering (ICRIE 2020), Materials Science and Engineering*, 2020.

[19] M. Dara and K. Manochehri, "Using rc4 and aes key schedule to generate dynamic s-box in aes," *Information Security Journal: A Global Perspective*, vol. 23, no. 1-2, 2014.

[20] A. I. Salih, A. Alabaichi, and A. S. Abbas, "A novel approach for enhancing security of aes using private xor table and 3d chaotic regarding to software quality factor," *ICIC Express Letters Part B: Applications*, vol. 10, no. 9, pp. 1574–1581, 2019.

[21] B. Prasetiyo and M. N. Ardian, "Enhancement security aes algorithm using a modification of transformation shiftrows and dynamic s-box," in *Journal of Physics: Conference Series*, vol. 1567, no. 3, 2020, p. 032025.

[22] T. T. Luong, N. N. Cuong, and B. Vo, "Aes security improvement by utilizing new key-dependent xor tables," *IEEE Access*, pp. 53 158–53 177, 2024.

[23] J. Daemen *et al.*, "Aes proposal: Rijndael," Katholieke Universiteit Leuven, ESAT-COSIC, Tech. Rep., 1999.

[24] E. R. and R. A. R., "Improving diffusion power of aes rijndael with 8x8 mds matrix," *International Journal of Scientific & Engineering Research*, vol. 2, pp. 1–5, 2011.

[25] S. M., D. M., M. H., and O. B., "On construction of involutory mds matrices from vandermonde matrices in gf(2q)," *Designs, Codes and Cryptography*, vol. 64, no. 3, pp. 287–308, 2012.

[26] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "The keccak reference, version 3.0," http://keccak.noekeon.org/Keccak-reference-3.0.pdf, 2011.

[27] G. Bertoni, J. Daemen, and M. Peeters, "Cryptographic sponge functions," http://sponge.noekeon.org/CSF-0.1.pdf, 2011.
[28] M. J. Dworkin, "Sha-3 standard: Permutation-based hash and extendable-output functions," National Institute of Standards and Technology, Tech. Rep., 2015.
[29] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," NIST, Tech. Rep., 2010.
[30] A. I. Salih and A. A. A. Y. T., "Enhancing aes security based on dual dynamic xor table and mixcolumns transformation," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 19, no. 3, pp. 1574–1581, 2020.
[31] L. H. Dinh, L. T. Thi, and L. N. Van, "On the mathematical aspects of cryptographic randomness tests using discrete fourier transform," in *2024 1st International Conference On Cryptography And Information Security (VCRIS)*. IEEE, 2024, pp. 1–6.

Tran Thi Luong received a Bachelor's degree from Hanoi University of Science, Hanoi, Vietnam, in 2006; a Master's degree in cryptographic technique at the Academy of Cryptography Techniques, Hanoi, Vietnam, in 2012; Ph.D. degree in cryptographic technique at the Academy of Cryptography Techniques, Hanoi, Vietnam in 2019. Her research interests include Cryptography, Coding theory, and Information Security.
Email: luongtran@actvn.edu.vn

Truong Minh Phuong received an Engineer of Cryptography Techniques of Academy of Cryptography Techniques in 2012; Master degreee in cryptographic technique at Academy of Cryptography Techniques in 2018. Workplace: Academy of Cryptography Techniques, No.141 Chien Thang road, Tan Trieu, Thanh Tri, Hanoi, Vietnam Recent research direction: Cryptogaphy.
Email: minhphuongh19@gmail.com