Enhancing SPN ciphers: Dynamic substitution-key addition layers via binary block circulant matrices

DOI: https://doi.org/10.54654/isj.v1i24.1100

Tran Thi Luong*, Truong Minh Phuong, Nguyen Van Long, Nguyen Nam Khanh

***Abstract*—The SPN (Substitution-Permutation Network) block cipher is one of the fundamental and important structures in the field of symmetric encryption, widely used in modern encryption algorithms such as AES. Due to its ability to efficiently diffuse and obscure data, SPN plays a key role in building secure and reliable encryption systems. However, with the development of the SPN block cipher, many studies have been conducted to identify cryptanalytic attack methods to break this cipher. To enhance the security of the SPN block cipher, recent research has focused on dynamic transformations of its components and often relies on the secret component, the key. In this study, we propose a new method that combines the dynamicization of two key components: the substitution layer and the key addition layer of the SPN block cipher. This method is based on using a binary block circular shift matrix, formed by combining a binary circular shift matrix with a binary Hadamard matrix. Our algorithm only requires the use of 26 additional key bits but can generate up to 2^{28} key-dependent S-boxes with strong cryptographic properties and 2^{33} key-dependent XOR tables. When applying these key-dependent S-boxes and XOR tables to dynamically modify the AES block cipher, it can increase the security level of the dynamic AES block cipher by 2^{33} compared to AES.**

***Tóm tắt*— Mã khối SPN (Substitution-Permutation Network) là một trong những cấu trúc cơ bản và quan trọng trong lĩnh vực mã hóa đối xứng, được sử dụng rộng rãi trong các thuật toán mã hóa hiện đại như AES. Nhờ khả năng phân tán và làm nhiễu dữ liệu hiệu quả, SPN đóng vai trò then chốt trong việc xây dựng các hệ thống mã hóa an toàn và đáng tin cậy. Tuy nhiên, cùng với sự phát triển của mã khối SPN, nhiều nghiên cứu đã được thực hiện nhằm tìm ra các phương thức tấn công thám mã để phá vỡ hệ mật này. Nhằm nâng cao tính bảo mật của mã khối SPN, các nghiên cứu gần đây tập trung vào việc động hóa các biến đổi thành phần trong cấu trúc của nó và thường có xu hướng phụ thuộc vào thành phần bí mật là khóa. Trong nghiên cứu này, nhóm tác giả đề xuất một phương pháp mới kết hợp động hóa hai thành phần quan trọng: tầng thay thế và tầng cộng khóa của mã khối SPN. Phương pháp này dựa trên việc sử dụng ma trận dịch vòng khối nhị phân, được tạo thành từ sự kết hợp giữa ma trận dịch vòng nhị phân và ma trận Hadamard nhị phân. Thuật toán của nhóm tác giả chỉ cần sử dụng 26 bít khóa nhưng có thể sinh ra tới 2^{28} Sbox phụ thuộc khóa có tính chất mật mã tốt và 2^{33} bảng XOR phụ thuộc khóa. Khi áp dụng các S-box phụ thuộc khóa và bảng XOR phụ thuộc khóa vào làm động mã khối AES có thể làm tăng thêm mức an toàn của mã khối AES động lên 2^{33} so với AES.**

Keywords*— *AES block cipher; Affine transformation; Dynamic S-box; Dynamic XOR table; Binary block circulant matrix.

Từ khóa*— *Mã khối AES; Biến đổi Affine; Hộp thế động; Bảng XOR động; Ma trận dịch vòng khối nhị phân.

I. INTRODUCTION

In the domain of symmetric key cryptography, the SPN [1-3] is a crucial concept and has wide applications in information security. By integrating both substitution and permutation operations, SPN ensures robust security by introducing complexity and unpredictability in the encryption process, making it harder to decipher and protecting the data from decryption attacks. Furthermore, key addition is another vital transformation in SPNs.

This manuscript was received on April 9, 2025. It was reviewed on May 7, 2025, revised on May 13, 2025 and accepted on May 19, 2025.

* Corresponding author

The Advanced Encryption Standard (AES) [4], a symmetric block cipher algorithm introduced by the National Institute of Standards and Technology (NIST) in 2001, was designed to replace the older DES encryption standard. AES operates on 128-bit blocks of data and offers key lengths of 128, 192, or 256 bits, providing strong security and efficient performance in safeguarding information.

In the current context, attack techniques such as differential cryptanalysis [5, 6] and linear cryptanalysis [7, 8], along with their variants, pose a significant threat to the security of block cipher algorithms. To address these challenges, many global studies have focused on developing dynamic block cipher algorithms to improve security levels compared to static block ciphers. Dynamic block ciphers have a distinct advantage in defending against statistical cryptanalysis methods such as differential cryptanalysis, linear cryptanalysis, and algebraic cryptanalysis. Expanding the key space contributes to enhancing security compared to traditional block ciphers.

Schneier [9] pointed out that relying on fixed S-boxes can expose the system to vulnerabilities if an attacker is able to exploit their weaknesses. On the other hand, using dynamic S-boxes provides a security advantage since the attacker cannot precisely identify which S-box is being used at any given moment. Incorporating dynamic S-boxes substantially improves the security of the block cipher against potential attacks that have not yet been discovered. The greater the variety of techniques used to introduce dynamism into the S-box, the more effectively the block cipher can resist powerful attack methods.

In research on dynamic block ciphers in the SPN model, the main approaches include dynamizing the diffusion layer [10-13], dynamizing the substitution layer [14-20], dynamizing the key addition layer [21-24], or combining multiple dynamic layers in one algorithm [25-27]. For approaches that dynamize the substitution layer of SPN block ciphers, many studies focus on DNA-based methods [19, 20], while others are based on chaotic mappings [14-18].

However, the dynamic S-boxes generated by these methods often fail to achieve crucial cryptographic properties comparable to those of AES S-boxes. Specifically, metrics such as nonlinearity, maximum linear probability, maximum differential probability, and algebraic degree of these dynamic S-boxes are significantly lower than those of AES S-boxes. Furthermore, these dynamic S-box generation algorithms are typically quite complex, resulting in substantial computational resource consumption and high implementation costs.

Several research efforts have focused on enhancing dynamic S-box cryptographic properties through modifications to the Affine transformation components, deriving dynamic S-boxes from the original AES S-box [15, 16]. The approach presented in [15] involves altering the binary matrix within the Affine transformation, resulting in the identification of 190 potential invertible binary matrices. Among these, 126 matrices produced balanced S-boxes while the remaining 64 failed to meet this criterion. This method faces limitations due to the presence of non-invertible matrices, potentially causing algorithm convergence issues and restricting the total number of producible dynamic S-boxes. These constraints necessitate secure storage mechanisms to prevent potential key compromise. An alternative methodology described in [16] proposes adjustments to both the additive constant and irreducible polynomial parameters. While this technique can generate dynamic S-boxes maintaining cryptographic strength equivalent to standard AES S-boxes, it encounters practical limitations. The finite availability of suitable additive constants and irreducible polynomials inherently restricts the diversity of generatable S-boxes. This limitation presents security concerns, as exposure of any single dynamic S-box could substantially increase vulnerability to key disclosure. Furthermore, polynomial modification requires recalculation of corresponding multiplicative inverses, eliminating the computational

efficiency advantage provided by AES's precomputed inverse tables.

The study in [28] introduced a novel approach for generating S-boxes through randomization techniques. This method constructs an initial S-box structure mimicking AES properties from entropy sources, then applies row and column permutations based on DES's eight predefined S-boxes, enabling the creation of 96 distinct S-box variants from each random seed. However, the research presents several critical limitations: the conversion mechanism from random sources to AES-compatible S-boxes remains unclear, and the resulting S-boxes demonstrate substantially weaker cryptographic properties compared to standard AES S-boxes. Specifically, the generated S-boxes achieve a maximum nonlinearity of just 108 (averaging 106.75), fail to meet strict avalanche criteria (with values ranging from 0.3909 to 0.6094), and require significantly more hardware resources - demanding 10 XOR gates per S-box compared to AES's optimized 4-gate implementation. These technical shortcomings not only reduce cryptographic strength but also increase implementation costs, making the approach less practical than conventional AES designs.

Several recent studies have examined dynamic approaches that combine either substitution and linear layers or linear and key addition layers in block ciphers. However, no previous work has explored the simultaneous dynamization of both substitution and key addition layers. Our research addresses this gap by introducing a novel method for generating dynamic S-boxes based on the unique properties of binary block circulant matrices, which are constructed from binary circulant matrices and binary Hadamard matrices. This approach produces new S-boxes with cryptographic properties comparable to the original AES S-box. A key advantage of our method is its efficiency - using just 26 additional key bits, we can generate 2^{28} distinct dynamic S-boxes and 2^{33} dynamic XOR tables. These key-dependent components enable the dynamization of both the substitution and key addition layers in AES, significantly enhancing security. Compared to standard AES, our dynamic version provides 2^{33} times greater security through this dual-layer dynamization approach.

The paper is structured as follows. Section 2 presents the fundamental knowledge that serves as the basis for the research. Section 3 introduces several definitions and important mathematical results that lay the groundwork for proposing the dynamic algorithm in Section 4. Section 4 proposes an algorithm for generating a dynamic S-box and a key-dependent dynamic XOR table for AES based on binary block circulant matrices. Section 5 focuses on analyzing the security and efficiency of the proposed method. Section 6 provides the conclusion.

II. PRELIMINARIES

A. The substitution layer and the key addition layer in AES

AES operates on 128-bit data blocks and supports key lengths of 128, 192, and 256 bits, which correspond to 10, 12, and 14 encryption rounds, respectively [4]. Within the AES structure, the SubBytes step employs an 8-bit S-box that is generated using the multiplicative inverse over the finite field $\mathbb{F}_{2^8}$. This is done using the irreducible polynomial $f(x) = 1 \oplus x \oplus x^3 \oplus x^4 \oplus x^8$. Following the inversion process, an Affine transformation is applied to finalize the substitution.

$$q' = (Aq^{-1} \oplus l) mod\, f(x) \qquad (1)$$

where, A is an invertible 8×8 binary matrix, $q = (q_7, q_6, \dots, q_0) \in \mathbb{F}_{2^8}$, $q' = (q'_7, q'_6 \dots, q'_0) \in \mathbb{F}_{2^8}$, with $q_i, q'_i \in \{0,1\}$, for $i = 0,1, \dots,7$ and $l \in \mathbb{F}_{2^8}$, where $l = 0x63$. The byte $\{00\}$ is mapped to itself.

Although the substitution layer is the sole source of nonlinearity in the AES block cipher, the AddRoundKey layer is essential for enhancing security. It operates by performing a bitwise XOR between each byte of the data block and the corresponding byte of the round key. This step adds an extra layer of complexity to the encryption process, making it more resistant to statistical cryptanalysis techniques.

B. The inverse mapping in the design of the AES S-box

In this section, we analyze the inverse mapping of the form $f(x) = x^{-1}$ over $\mathbb{F}_{2^8}$, which is used in the design of the AES S-box.

Definition 1 [29]. Let $(\mathbb{F}, \cdot, +)$ be a finite field. Then the inverse mapping $f: \mathbb{F} \to \mathbb{F}$ satisfies: $f(x) = \begin{cases} x^{-1}, & if\ x \neq 0 \\ 0, & if\ x = 0 \end{cases}$

In [29], the authors pointed out several important properties of the inverse mapping, including:

- The nonlinearity of $f(x)$ is: $\mathrm{N}(f) \leq 2^{n-1} - 2^{n/2}$.

- The algebraic degree: $\deg\left(tr(\omega x^{-1})\right) = n - 1$.

In [4], the authors highlighted two key criteria when designing the AES S-box. The first criterion is that the nonlinearity should minimize the input-output correlation amplitude and the maximum differential propagation probability. The second criterion is that the S-box's algebraic expression in $\mathbb{F}_{2^8}$ should be complex. To achieve this, the authors adopted the approach outlined in [29], where the S-box is defined by the following function in $\mathbb{F}_{2^8}$:

$$\mathrm{Inv}_8: x \to y = a^{254}$$

This function is commonly described as the mapping $x \to x^{-1}$, with the exception that 0 is mapped to 0. The modulo polynomial used for the calculation of the inverse mapping here is $f(x) = 1 \oplus x \oplus x^3 \oplus x^4 \oplus x^8$.

In the design of the AES S-box, the authors also noted that the inverse mapping $x \to x^{-1}$ has a straightforward algebraic form, which could be vulnerable to exploitation in an interpolation attack. To address this, they developed the S-box by combining the inverse function Inv_8 with an invertible linear transformation Aff_8, as follows:

$$S\text{-}box = Aff_8 \circ Inv_8$$

In the study [30], the authors pointed out that the affine transformation preserves nonlinearity, correlation coefficients, and avalanche criteria, etc. In the design of the AES S-box, it is essentially an affine transformation of the inverse mapping x^{-1}. However, with appropriate parameters, the Affine transformation makes the algebraic expression for generating the S-box more complex.

Therefore, in this study, we will generate key-dependent S-boxes that create an affine equivalence layer with the mapping x^{-1} over the finite field $\mathbb{F}_{2^8}$ to preserve the important cryptographic criteria equivalent to the original AES S-box.

C. Circulant matrix, block circulant matrix, Hadamard matrix

The matrix used in the Affine transformation to compute the output bytes of the S-box in AES is the binary circulant matrix. Additionally, the block circulant matrix and Hadamard matrix are also crucial for our dynamic algorithm. Therefore, this section will reintroduce the concepts and some important properties of these types of matrices.

Definition 1 [31]. A circulant matrix is a matrix of the form:

$$B = circ(b_1, b_2, \dots, b_d) = \begin{pmatrix} b_1 & b_2 & \dots & b_d \\ b_d & b_1 & \dots & b_{d-1} \\ \vdots & \vdots & \vdots & \vdots \\ b_2 & b_3 & \dots & b_1 \end{pmatrix}$$

where $b_i \in \mathbb{F}_{2^s}$, for $1 \leq i \leq d$.

From Definition 1, we derive the general form of the left or right circulant matrix in the field $\mathbb{F}_2$ as follows:

Definition 2. Given d elements $b_1, b_2, \dots, b_d$ with $b_i \in \mathbb{F}_2, 1 \leq i \leq d$, a matrix $B = L_{\mathrm{circ}}(b_1, b_2, \dots, b_d) = [b_{(i,j)}]$ is called a left binary circulant matrix if each element in this matrix is constructed as follows:

$$b_{(i,j)} = \alpha_{((i+j) \mod d)}, \quad 0 \leq i, j \leq d - 1.$$

Definition 3. Given d elements $b_1, b_2, \dots, b_d$ with $b_i \in \mathbb{F}_2, \quad 1 \leq i \leq d$, a matrix $B = R_{\mathrm{circ}}(b_1, b_2, \dots, b_d) = [b_{(i,j)}]$ is called a right binary circulant matrix if each element in this matrix is constructed as follows:

$$b_{(i,j)} = \alpha_{((d-i+j) \mod d)}, \quad 0 \leq i, j \leq d - 1.$$

With circulant matrices, the following results are presented in [31]:

Proposition 1 [31]. Let $B = circ(b_1, b_2, \dots, b_{2^d})$ be a circulant matrix of size $2^d \times 2^d$ over the finite field $\mathbb{F}_{2^s}$, then:

$$det(B) = \sum_{i=1}^{2^d} b_i^{2^d} \text{ and } B^{2^d} = det(B) \cdot I.$$

Proposition 2 [31]. Let $B = circ(b_1, b_2, \dots, b_{2^d})$ be a $2^d \times 2^d$ circulant matrix over $\mathbb{F}_{2^s}$, then:

$$B^2 = circ(b_1^2 + b_{d+1}^2, 0, b_2^2 + b_{d+2}^2, 0, \dots, b_d^2 + b_{2d}^2, 0).$$

In [32], a definition of a block circulant matrix is provided as follows:

Definition 4 [32]. A block circulant matrix - (x, y) of size $xy \times xy$ is a matrix of the form:

$$DC = bcirc(A_1, \dots, A_x) = \begin{pmatrix} A_1 & A_2 & \dots & A_x \\ A_x & A_1 & \dots & A_{x-1} \\ \vdots & \vdots & \vdots & \vdots \\ A_2 & A_3 & \dots & A_1 \end{pmatrix}$$

where $A_1, A_2, \dots, A_x$ are square matrices of size y.

If each block A_i is a circulant matrix, then DC is a block circulant matrix $-(x, y)$ with circulant blocks.

The class of these matrices is denoted as $DC_{x,y}$.

Block circulant matrices have the following properties:

Theorem 1 [32]. Let $DC = bcirc(A_1, \dots, A_x) \in DC_{(x,y)}$ over the finite field $\mathbb{F}_{2^s}$, where $A_i = circ(a_{i,1}, a_{i,2}, \dots, a_{i,y})$, $1 \le i \le x$, $x = 2^m$, and $y = 2^n$. Then:

$$DC^{max(x,y)} = det(DC)^{\frac{1}{min(x,y)}} I_{xy},$$

where $\det(DC) = \sum_{i=1}^{x} det\,(A_i)^x$.

Definition 5 ([33], [34]). Let k elements $d_0, d_1, \dots, d_{k-1}$ form a matrix $H = had(d_0, d_1, \dots, d_{k-1}) = [d_{i,j}]$ be a Hadamard matrix if each element in this matrix is constructed as follows:

$$d_{i,j} = d_{i \oplus j}, 0 \le i, j \le k - 1,$$

where the elements of H belong to $\mathbb{F}_{2^s}$.

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

$$d_{i,j} = d_{i \oplus j}, 0 \le i, j \le k - 1,$$

where the elements of H belong to $\mathbb{F}_{2^s}$.

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

$$\mathrm{H} = had(d_0, d_1, d_2, d_3) = \begin{bmatrix} d_0 & d_1 & d_2 & d_3 \\ d_1 & d_0 & d_3 & d_2 \\ d_2 & d_3 & d_0 & d_1 \\ d_3 & d_2 & d_1 & d_0 \end{bmatrix}$$

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

In this paper, we propose a new definition for the matrices N_Hadamard to be used in the generation of dynamic key-dependent S-boxes.

The NXOR operation is denoted by $\odot$, which is the negation of the XOR operation. The output value is 1 if both input values are the same, and 0 if the input values are different. Based on the NXOR operation, we introduce the concept of a matrix in the form of N_Hadamard as follows:

Definition 6. Given k elements $\gamma_0, \gamma_1, \dots, \gamma_{k-1}$, a matrix $Nh = Nhad(\gamma_0, \gamma_1, \dots, \gamma_{k-1}) = [d_{i,j}]$ is called an N_Hadamard matrix if each element in this matrix is constructed as follows:

$$d_{i,j} = \gamma_{i \odot j}, 0 \le i, j \le k - 1,$$

where the elements of Nh belong to $\mathbb{F}_{2^s}$.

Note that the Hadamard matrices, N_Hadamard, circulant matrices, and block circulant matrices over $\mathbb{F}_2$ are referred to as binary Hadamard matrices, binary N_Hadamard matrices, binary circulant matrices, and binary block circulant matrices.

III. On a form of binary block circulant matriecs and some related results

In this section, we define a form of binary block circulant matrix, which is generated by

combining a binary circulant matrix and a binary Hadamard matrix. We then present some new results related to this matrix form.

Definition 7. Let $C = circ(b_1, b_2, \dots, b_{2^{n-1}})$ be a binary circulant matrix of size $2^{n-1} \times 2^{n-1}$, with $n \geq 2$, over the finite field $\mathbb{F}_2$, and $H = \text{had}(b_{2^{n-1}+1}, b_{2^{n-1}+2}, \dots, b_{2^n})$ be a binary Hadamard matrix of size $2^{n-1} \times 2^{n-1}$ over $\mathbb{F}_2$. Then, a binary block circulant matrix of size $2^n \times 2^n$ generated from these two matrices will have one of the following two forms:

$$D_{CH} = bcirc(C, H) = \begin{pmatrix} C & H \\ H & C \end{pmatrix}$$

or

$$D_{HC} = bcirc(H, C) = \begin{pmatrix} H & C \\ C & H \end{pmatrix}$$

From the definition of the binary block circulant matrix above, we present the following important results as the foundation for proposing a dynamic algorithm for the SPN block cipher in Section 4.

Lemma 1. Let the set $T = \{b_1, b_2, \dots, b_{2^n}\}$ with $b_i \in \mathbb{F}_2$, $1 \leq i \leq 2^n$, and $n \geq 1$. Let $C = circ(b_1, b_2, \dots, b_{2^n})$ be a binary circulant matrix of size $2^n \times 2^n$ and $H = had(b_1, b_2, \dots, b_{2^n})$ be a binary Hadamard matrix of size $2^n \times 2^n$. Then, $Det(C) = Det(H)$.

Proof.

First, we use the induction method to prove the determinant of the matrix H as follows:

$$det(H) = \sum_{i=1}^{2^n} b_i.$$

- Indeed, for $n = 1$, the matrix H has the following form:

$$H = had(b_1, b_2) = \begin{pmatrix} b_1 & b_2 \\ b_2 & b_1 \end{pmatrix}.$$

Thus,

$$det(H) = b_1^2 + b_2^2.$$

Since $b_i \in \mathbb{F}_{2_2}$, we have $b_i^2 = b_i$, so:

$$\det(H) = b_1 + b_2. \quad (2)$$

- For $n = 2$, the matrix H has the following form:

$$H = had(b_1, b_2, b_3, b_4) = bcirc\big(had(b_1, b_2), had(b_3, b_4)\big).$$

According to the properties of block circulant matrices (Theorem 1), we have:

$$det(H) = det\big(had(b_1, b_2)\big)^2 + det\big(had(b_3, b_4)\big)^2$$

Since $b_i \in \mathbb{F}_2$, it is to have:

$$det(H) = \sum_{i=1}^{4} b_i^4 = \sum_{i=1}^{4} b_i \quad (3)$$

- Assume the lemma is true for $n = k$, meaning that for a matrix H of size $2^k \times 2^k$, we have:

$$det(H) = \sum_{i=1}^{2^k} b_i$$

Now, consider the Hadamard matrix H of size $2^{k+1} \times 2^{k+1}$, which can be represented as follows:

$$H = had\big(b_1, b_2, \dots, b_{2^{k+1}-1}, b_{2^{k+1}}\big) = bcirc(had\big(b_1, \dots b_{2^k}\big), had\big(b_{2^k+1}, \dots, b_{2^{k+1}}\big))$$

According to Theorem 1, we have:

$$det(H) = det\left(had\big(b_1, \dots b_{2^k}\big)\right)^2 + det\left(had\big(b_{2^k+1}, \dots, b_{2^{k+1}}\big)\right)^2 \quad (4)$$

By the induction hypothesis, we have:

$$det\left(had\big(b_1, \dots b_{2^k}\big)\right) = \sum_{i=1}^{2^k} b_i,$$

$$det\left(had\big(b_{2^k+1}, \dots, b_{2^{k+1}}\big)\right) = \sum_{i=2^k+1}^{2^{k+1}} b_i,$$

Substituting these values into (4), we get:

$$det(H) = \sum_{i=1}^{2^k} b_i^2 = \sum_{i=1}^{2^k} b_i \quad (5)$$

From (2), (3), and (5), we can prove that:

$$det(H) = \sum_{i=1}^{2^n} b_i \quad (6)$$

Now, let's consider the matrix $C = circ(b_1, b_2, \dots, b_{2^n})$.

According to Proposition 1, we have: $det(C) = \sum_{i=1}^{2^n} b_i^{2^n}$

As $b_i \in \mathbb{F}_2$, we have:

$$det(C) = \sum_{i=1}^{2^n} b_i \quad (7)$$

From (6) and (7), we conclude that:

$$det(H) = det(C).$$

Next, we present the following two propositions.

Proposition 3. Let the set $T = \{b_1, b_2, \dots, b_{2^n}\}$ where $b_i \in \mathbb{F}_2$ and $1 \le i \le 2^n, n \ge 2$, contain an odd number of elements equal to 1. Let $C = circ(b_1, b_2, \dots, b_{2^{n-1}})$ be the binary circulant matrix of size $2^{n-1} \times 2^{n-1}$ and $H = had(b_{2^{n-1}+1}, \dots, b_{2^n})$ be the binary Hadamard matrix of size $2^{n-1} \times 2^{n-1}$. Then, the two binary block circulant matrices $D_{CH} = bcirc(C, H)$ and $D_{HC} = bcirc(H, C)$ are invertible.

Proof.

According to the proof of Proposition 1, we have:

$det(C) = \sum_{i=1}^{2^{n-1}} b_i$ and $det(H) = \sum_{i=2^{n-1}+1}^{2^n} b_i$

Therefore, according to Theorem 1, we have:

$$det(D_{CH}) = det(D_{HC}) = det(C)^2 + det(H)^2$$
$$= \sum_{i=1}^{2^{n-1}} b_i^2 + \sum_{i=2^{n-1}+1}^{2^n} b_i^2$$
$$= \sum_{i=1}^{2^n} b_i^2$$
$$= \sum_{i=1}^{2^n} b_i$$

According to the assumption, the set $T = \{b_1, b_2, \dots, b_{2^n}\}$ contains an odd number of 1's, so $\sum_{i=1}^{2^n} b_i = 1$. Therefore, $det(D_{CH}) = det(D_{HC}) = 1$.

Proposition 4. Let the set $T = \{b_1, b_2, \dots, b_{2^n}\}$ where $b_i \in \mathbb{F}_2$ and $1 \le i \le 2^n$, $n \ge 2$, contain an odd number of 1's. Let $C = circ(b_1, b_2, \dots, b_{2^{n-1}})$ be a binary circulant matrix of size $2^{n-1} \times 2^{n-1}$, and $H = had(b_{2^{n-1}+1}, \dots, b_{2^n})$ be a binary Hadamard matrix of size $2^{n-1} \times 2^{n-1}$. Then, the two binary block circulant matrices $D_{CH} = bcirc(C, H)$ and $D_{HC} = bcirc(H, C)$ satisfy:

$$(D_{CH})^{2^{n-1}} = (D_{HC})^{2^{n-1}} = I_{2^n}$$

where I_{2^n} is the identity matrix of size $2^n \times 2^n$.

Proof.

Based on the assumption that $n \ge 2$, we have $2^{n-1} \ge 2$. Therefore, $max(2, 2^{n-1}) = 2^{n-1}$ and $min(2, 2^{n-1}) = 2$.

According to Theorem 1, we have:

$$(D_{CH})^{2^{n-1}} = det(D_{CH})^{1/2} I_{2^n}$$

From the result of Proposition 3, we know:

$$det(D_{CH}) = 1$$

Thus, we conclude:

$$(D_{CH})^{2^{n-1}} = I_{2^n}$$

Similarly, we also have:

$$(D_{HC})^{2^{n-1}} = I_{2^n}.$$

The meaning of Proposition 3 is to prove that matrices of the form $D_{CH} = bcirc(C, H)$ and $D_{HC} = bcirc(H, C)$, constructed from the set T, are always invertible. This will serve as the foundation for applying changes to the matrix in the Affine transformation to generate new S-boxes, ensuring that these new S-boxes always have an inverse S-box for the decryption process.

With Proposition 4, when we raise the matrices $D_{CH} = bcirc(C, H)$ and $D_{HC} = bcirc(H, C)$ to the power of 2^{n-1}, we obtain the identity matrix of size $2^n \times 2^n$. By applying this when generating $D_{CH} = bcirc(C, H)$ and $D_{HC} = bcirc(H, C)$ matrices of size 8×8, we can only raise them to the power of $3 = 2^{n-1} - 1$, with $n = 3$, to create additional matrix spaces, as $D_{CH}^4 = I$.

From the results obtained in this section, we will use them as a foundation to construct a dynamic algorithm for the SPN block cipher in Section 4.

IV. PROPOSED ALGORITHM FOR GENERATING KEY-DEPENDENT S-BOX AND KEY-DEPENDENT DYNAMIC XOR TABLE BASED ON BINARY BLOCK CIRCULANT MATRICES

In this section, we propose an algorithm for generating both key-dependent S-boxes and key-dependent XOR tables based on the results obtained in Section 3, using the binary block circulant matrix. The algorithm is designed to generate 8-bit key-dependent S-boxes and 4-bit key-dependent XOR tables, which can be applied to improve the dynamic operation of the AES block cipher.

Algorithm 1. Generation of key-dependent S-box and key-dependent XOR table based on binary block circulant matrix

INPUT:

- The secret key sK has a bit length of m (where $m \geq 128$).

- The array R consists of the inverses of elements from the field $\mathbb{F}_{2^8}$ based on the generator polynomial $f(x) = 1 \oplus x \oplus x^3 \oplus x^4 \oplus x^8$.

OUTPUT:

- An 8-bit key-dependent S-box S_{kd} is represented as a 256-byte array.

- A key-dependent XOR table X_{kd}.

Step 1. Generate the key-dependent S-box S_{kd}.

Step 1.1. Extract the initial 16 bits from the secret key sK and split them into two separate segments.

$$sK_1 = (s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7),\ sK_2 = (s_8 s_9 s_{10} s_{11} s_{12} s_{13} s_{14} s_{15})$$

Step 1.2. In case the number of 1s in sK_2 is odd, move on to Step 1.4. If not, apply bitwise negation to sK_2: $s_{15} = s_{15} \oplus 1$.

Step 1.3. Construct the following matrices:

+ Two left circulant matrices:

$$C_{L1} = Lcirc(s_8, s_9, s_{10}, s_{11}), \quad C_{L2} = Lcirc(s_{12}, s_{13}, s_{14}, s_{15})$$

+ Two right circulant matrices:

$$C_{R1} = Rcirc(s_8, s_9, s_{10}, s_{11}), \quad C_{R2} = Rcirc(s_{12}, s_{13}, s_{14}, s_{15})$$

+ Two Hadamard matrices:

$$H_1 = had(s_8, s_9, s_{10}, s_{11}), \quad H_2 = had(s_{12}, s_{13}, s_{14}, s_{15})$$

+ Two N_Hadamard matrices:

$$N_{h1} = Nhad(s_8, s_9, s_{10}, s_{11}), \quad N_{h2} = Nhad(s_{12}, s_{13}, s_{14}, s_{15})$$

Step 1.4. Build the binary matrix A used in the Affine transformation:

Extract the subsequent 4 bits of the secret key, specifically $(s_{16}s_{17}s_{18}s_{19})$, and utilize the matrices generated in Step 1.3 to identify the invertible binary matrix A_1 using the selection function f_1 described in Table 1. $A_1 = f_1(s_{16}s_{17}s_{18}s_{19}, C_{L1}, C_{L2}, C_{R1}, C_{R2}, H_1, H_2, N_{h1}, N_{h2})$

(Note that Table 1 includes various binary block circulant matrices.)

Then, extract the next 2 bits of the key (s_{20}, s_{21}) from sK and use the selection function f_2 to determine the binary matrix A, as shown in Table 2.

$$A = f_2(s_{20}s_{21}, A_1)$$

Step 1.5. Generate the key-dependent S-box S_{kd}

For each element $q \in \mathbb{F}_{2^8}$, the output of the key-dependent S-box S_{kd} is computed as:

$$S_{kd}[q] = A \cdot R[i_q] \oplus sK_1$$

where i_q represents the decimal value of q.

Step 2. Build the key-dependent XOR table X_{kd}

Step 2.1.

- Let i be the decimal value corresponding to the 4 bits of the key $s_{22}s_{23}s_{24}s_{25}$.

Step 2.2.

- If $s_{22} = 0$, select column i of S_{kd}. If $s_{22} = 1$, select row i of S_{kd}.

- Take the 16 values from the i-th row/column of S_{kd} and store them in the set $T = \{t_0, t_1, \dots, t_{15}\}$.

Step 2.3.

Sort the set T based on the decimal values of its elements in ascending order, resulting in a new set T'.

Step 2.4. Extract the indices i of the elements t_i from T', from left to right, obtaining 16 unique indices. These indices are then placed in a set $L = \{l_0, l_1, \dots, l_{15}\}$.

Create a Hadamard matrix D using the elements from the set V.

$$D = had\{l_0, l_1, \ldots, l_{15}\}$$

Step 2.5. Construct the dynamic XOR table
Generate the XOR table with the first row and first column taken from the initial row of matrix D. The elements within the XOR table come from D.

Step 2.6. Rearrange rows and columns
Reorganize the first row and first column of the XOR table in ascending order from 0 to 15. The final result is the dynamic XOR table X_{kd}.

Algorithm 1 is described in Figure 1.

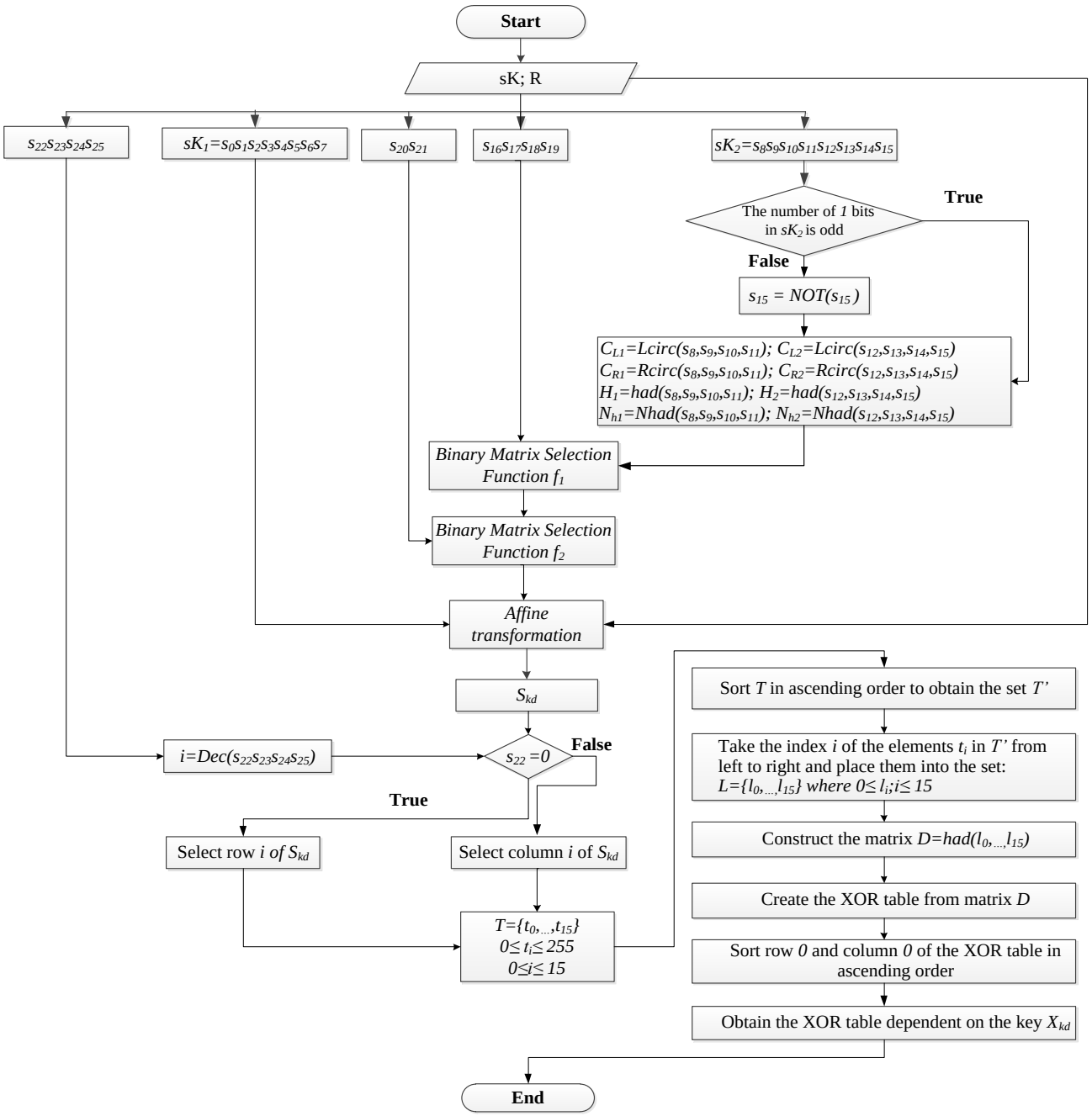

Figure 1. Algorithm 1 diagram

TABLE 1. DESCRIBING THE SELECTION FUNCTION FOR THE BINARY BLOCK CIRCULANT MATRIX $\boldsymbol{f_1}$.

No.	$\boldsymbol{s_{16}s_{17}s_{18}s_{19}}$	$\boldsymbol{A_1 = f_1(s_{16}s_{17}s_{18}s_{19}, C_{L1}, C_{L2}, C_{R1}, C_{R2}, H_1, H_2, N_{h1}, N_{h2})}$
1	**0000**	$bcirc(C_{L1}, H_2)$
2	**0001**	$bcirc(C_{L2}, H_1)$
3	**0010**	$bcirc(C_{L1}, Nh_2)$
4	**0011**	$bcirc(C_{L2}, Nh_1)$
5	**0100**	$bcirc(C_{R1}, H_2)$
6	**0101**	$bcirc(C_{R2}, H_1)$
7	**0110**	$bcirc(C_{R1}, Nh_2)$
8	**0111**	$bcirc(C_{R2}, Nh_1)$
9	**1000**	$bcirc(H_2, C_{L1})$
10	**1001**	$bcirc(H_1, C_{L2})$
11	**1010**	$bcirc(Nh_2, C_{L1})$
12	**1011**	$bcirc(Nh_1, C_{L2})$
13	**1100**	$bcirc(H_2, C_{R1})$
14	**1101**	$bcirc(H_1, C_{R2})$
15	**1110**	$bcirc(Nh_2, C_{R1})$
16	**1111**	$bcirc(Nh_1, C_{R2})$

TABLE 2. DESCRIPTION OF THE SELECTION FUNCTION $\boldsymbol{f_2}$.

No.	$\boldsymbol{s_{20}s_{21}}$	$\boldsymbol{A = f_2(s_{20}s_{21}, A_1)}$
1	**00** or **01**	A_1
2	**10**	A_1^2
3	**11**	A_1^3

Example. Given a secret key $sK = 0x2B7E16F3D6CB81628AED2A6ABF7158809CF4F3C$.

We have: $sK_1 = 0x2B = 00101011$; $sK_2 = 0x7E = 01111110$.

Since the number of 1 bits in sK_2 is even, we perform $s_{15} = s_{15} \oplus 1$, resulting in $sK_2 = 01111111$. We have $s_{16}s_{17}s_{18}s_{19} = 0001$, so from Table 1, we get $A_1 = bcirc(C_{L2}, H_1)$.

Since $s_{20}s_{21} = 01$, according to Table 2, the output matrix of the function f_2 is:

$$A = A_1 = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

The dynamic key-dependent S-box is then generated using the Affine transformation with the binary matrix A and the constant $l = sK_1 = 0x2B$, according to equation (1):

$$q' = (Aq^{-1} \oplus l) mod\ f(x),$$

where $f(x) = 1 \oplus x \oplus x^3 \oplus x^4 \oplus x^8$.

We obtain the dynamic key-dependent S-box, which is presented in Table 3.

TABLE 3. A DYNAMIC KEY-DEPENDENT S-BOX GENERATED FROM ALGORITHM 1

S_{kd}	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0x2B	0x54	0x9A	0x44	0x07	0x9E	0xF5	0xA0	0x3C	0x26	0x40	0x28	0xD9	0xAC	0x73	0x37
1	0x05	0x06	0x7E	0xF9	0xB2	0xFF	0x2D	0xD1	0xCE	0xD7	0x6B	0x18	0xD4	0xD6	0x5C	0x66
2	0x77	0xA2	0x71	0x5B	0x81	0x99	0xC1	0xB8	0x57	0x7B	0xCD	0x7C	0x27	0x09	0x91	0x97
3	0xE0	0x76	0x9D	0x1D	0xE4	0xB7	0x4D	0x69	0x9B	0x87	0xD0	0xDD	0xC5	0xF6	0xB1	0x4C
4	0x93	0xAB	0x38	0x32	0x9F	0x58	0x84	0xBD	0x31	0xF2	0x01	0x1C	0xFE	0x70	0x43	0xBB
5	0x9C	0x11	0x8B	0x78	0xE6	0x0F	0xCA	0x29	0x33	0xA8	0x6F	0x24	0x5E	0xE3	0x52	0xC3
6	0xBC	0xAE	0xDE	0x1F	0x46	0x4E	0x47	0x16	0xFB	0xC9	0x5D	0x2F	0xE7	0xE2	0xAF	0x08
7	0x4A	0xC6	0x3D	0x75	0xDC	0x79	0x89	0x17	0xB9	0xDA	0xDF	0x4B	0x51	0x74	0xAA	0x6A
8	0x15	0x55	0x2A	0xD5	0x42	0x1A	0x56	0x41	0x0D	0xAD	0x82	0x4F	0x3B	0x94	0x49	0xF1
9	0x50	0x7D	0x98	0x62	0x30	0x85	0x0A	0x65	0x80	0x03	0xD2	0x3A	0xB4	0xEF	0x45	0x72
A	0x0B	0xEA	0x5F	0xE8	0x25	0x36	0x8D	0x39	0xB0	0xC7	0x63	0x86	0xA7	0xCC	0x60	0x92
B	0x1B	0xD3	0x2C	0x23	0xA3	0x7A	0x35	0xA5	0x8E	0x5A	0xBA	0x68	0x96	0xE9	0xB5	0x3E
C	0x04	0x3F	0x20	0xEE	0x8F	0x00	0x0C	0xDB	0xBE	0xCF	0xE1	0xF4	0xF3	0x14	0x1E	0xB3
D	0x8A	0x34	0xA1	0xED	0x88	0xF0	0x6C	0x83	0xFD	0xFA	0xF7	0x7F	0x6D	0xCB	0x22	0x02
E	0xA6	0x64	0xBF	0xFC	0x48	0xA4	0xD8	0x61	0xC4	0x59	0xC0	0x13	0x6E	0x21	0x53	0x19
F	0x0E	0x10	0xC8	0xF8	0xC2	0xB6	0xEB	0xE5	0x90	0x12	0x95	0x2E	0x67	0x8C	0xA9	0xEC

Corresponding to the dynamic S-box obtained in Table 3, we also obtain a dynamic XOR table in Table 4.

TABLE 4. A KEY-DEPENDENT XOR TABLE X_{kd} FROM ALGORITHM 1

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

V. IMPROVEMENTS TO THE AES BLOCK CIPHER, SECURITY EVALUATION, COMPARION

We use Algorithm 1 to introduce dynamic changes to the AES block cipher at both the substitution and key addition layers. For each secret key used in a communication session, a dynamic key-dependent S-box and XOR table are generated, replacing the original S-box and XOR table of AES. Afterward, we evaluate the security of the dynamic AES block cipher and examine its compliance with the NIST statistical randomness standards for its output.

According to Algorithm 1, the number of key-dependent S-boxes generated depends on the number of invertible binary matrices that can be created and the number of constant values used.

In Algorithm 1, the number of 1-bits in sK_2 must be odd. Since sK_2 consists of 8 bits, there are 128 possibilities for sK_2 to contain an odd number of 1-bits. Additionally, 8 bits of the key are used to generate two left circulant matrices, two right circulant matrices, one Hadamard matrix, and one N_hadamard matrix. Therefore, the possible combinations of these matrices is 2^8. The number of constant values depends on sK_1, so there are 2^8 possibilities. Combining with the two selection functions f_1 and f_2 in Table 1 and Table 2, the number of key-dependent S-boxes that can be generated according to Algorithm 1 is:$\approx 2^7 \times 2^8 \times 2^8 \times 16 \times 3 \approx 2^{28}$. Since there are 2^4 possibilities for the bits $s_{22}s_{23}s_{24}s_{25}$, and two options for choosing a row or column, for each new S-box, we have 2^5 possibilities for generating the corresponding XOR tables.

Thus, with Algorithm 1, the total number of key bits required is 26, but it is possible to generate approximately 2^{28} key-dependent S-boxes and 2^{33} key-dependent XOR tables.

In cryptography, the security level of a cryptosystem is typically determined by the length of the secret key. Specifically, if the secret key is s bits long, the cryptosystem must offer a security level of at least 2^s, meaning no attack should be more efficient than a brute-force search. In our algorithm for generating dynamic key-dependent S-boxes and XOR tables, assuming the encryption key is s bits long and 26 bits are used to generate the dynamic key-dependent S-box and XOR table, the overall security of the algorithm can be roughly estimated as $2^{(s+26)}$. Additionally, the proposed method, utilizing only 26 bits of the key, can generate approximately 2^{28} key-dependent S-boxes and 2^{33} key-dependent XOR tables, thereby significantly enhancing the security of the AES block cipher.

The implementation of dynamic S-boxes and dynamic XOR tables in block ciphers such as AES provides a substantial boost in security against cryptanalysis attacks. In the standard AES, the S-box is static and publicly available, enabling attackers to gather and study statistical patterns in an attempt to deduce the secret key using methods like differential or statistical analysis. By making the S-box dynamic with the key, these predictable patterns are disrupted, making such cryptanalysis methods less effective [4].

In addition, linear cryptanalysis, which rely on finding high-probability linear paths, are significantly hindered when the S-box changes with the key, making it difficult for attackers to predict stable linear characteristics [7]. Moreover, the dynamic XOR table enhances the randomness of the key mixing process by altering the way the subkeys are XORed into the state. As this XOR table changes with the secret key, fixed XOR patterns are disrupted, complicating related-key attacks or techniques that attempt to derive subkeys through pattern analysis [35].

Furthermore, the standard AES is vulnerable to side-channel attacks, including methods such as timing analysis, electromagnetic leakage, or power consumption analysis (DPA/SPA). When the S-box and XOR table change dynamically with each encryption, it becomes significantly harder to collect stable features from leakage information, thus reducing the risk of exploitation through side-channel attacks [36].

Therefore, the use of dynamic S-boxes and dynamic XOR tables not only helps AES resist differential and linear cryptanalysis attacks but also improves its resistance to side-channel attacks, significantly enhancing the security of the algorithm.

Our proposed algorithm requires only a small key cost of 26 bits but is capable of generating a large number of S-boxes. Furthermore, from the dynamically generated S-boxes, we leverage them to produce the corresponding dynamic XOR tables. By demonstrating two mathematical propositions (Proposition 3 and Proposition 4), we have laid a strong scientific foundation for Algorithm 1, ensuring its convergence through the generation of invertible binary matrices. This guarantees that the dynamically generated S-boxes adhere to solid cryptographic standards based on the Affine transformation and possess key cryptographic properties that are comparable

to the original AES S-box. A comparison of the cryptographic properties of the dynamically generated S-boxes from our algorithm with the AES S-box is shown in Table 5.

In addition, the proposed algorithm streamlines the process of calculating the inverse of the Affine matrix, which is crucial for generating the inverse S-box for decryption. Notably, the use of the binary block circulant matrix we introduced enhances the algorithm, significantly lowering memory consumption and execution time, which in turn boosts the practical performance of the algorithm.

To make a comparison, we can examine the approach taken by the authors in [15]. Although this method utilizes a 16-bit key, it is unable to generate all 2^{16} dynamic S-boxes. As a result, when applied in dynamic algorithms, the overall security level does not reach 2^{n+16}. Consequently, the security level in the study [15] is considerably lower than in our research. This demonstrates that making one or more components of a block cipher dynamic, based on a secret key, does not reduce the complexity of brute-force attacks. This principle applies not only to the encryption key but also to the key used in the dynamic modification of cryptographic elements.

To evaluate efficiency, we compare our approach with the one in [16]. In this study, the authors proposed altering the generator polynomial of the finite field and the constant l in the Affine transformation (1) based on the key. However, modifying the generator polynomial changes the entire computational basis, making the inverse calculation with the new polynomial relatively complex, which increases both resource costs and execution time. Furthermore, this alteration could impact other transformations, such as Mix-Columns. With the original polynomial, it may be an MDS matrix, but this has not been verified with alternative polynomials. This could potentially disrupt the diffusion strategy used in the AES design, thereby reducing AES's overall security.

In comparison with approaches that utilize chaotic maps [14, 17, 18] or DNA-based techniques [19, 20], these methods generally rely on intricate mathematical frameworks, which can complicate their practical implementation and demand a significant number of key bits. Additionally, the S-boxes produced by such methods often lack the robust cryptographic properties found in the original AES S-box, potentially compromising their effectiveness in secure encryption schemes.

TABLE 5. COMPARISON OF KEY CRYPTOGRAPHIC PROPERTIES BETWEEN A KEY-DEPENDENT S-BOX GENERATED BY ALGORITHM 1 AND THE AES S-BOX

No.	**Criterion**		**S-box**	
			Standard AES S-box	***Algorithm 1-based dynamic S-box***
1	Balance property		Yes	Yes
2	Nonlinearity		112	112
3	Algebraic degree		7	7
4	Overall avalanche criterion	AC_max	32	32
		Sum of squares index	133120	133120
5	Resilience degree		No	No
6	Maximum linear probability		0.0156	0.0156
7	Maximum differential probability		0.0156	0.0156
8	AI		4	4
9	Transparency degree		7.458333	7.414461
10	Resistance to interpolation attack		Yes	Yes

Testing the degree of randomness according to NIST statistical methods:

This section presents an evaluation of the modified dynamic AES block cipher using the statistical tests recommended by NIST [37], with the dynamic S-box and XOR table produced through Algorithm 1.

TABLE 6. PERFORMANCE REVIEW OF DYNAMIC AES THROUGH LEVEL-2 STATISTICAL METRICS APPLIED TO BRIEF SEQUENCES

No. of Rouns	Freq. Test	Runs Test	Test for longest run of ones	Serial Test	AppEn. Test	CuSum. Test	Bit AutoCorr. Test	Byte Autocor. Test
AV1 Input Data								
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.004611	0.000000	0.000000	0.000001	0.000000	0.000001	0.000000	0.024087
3	0.399140	0.680278	0.296774	0.360755	0.113857	0.482432	0.460673	0.008957
4	0.925600	0.526663	0.773360	0.986072	0.964837	0.265680	0.917013	0.808591
5	0.868725	0.348059	0.562529	0.681677	0.590030	0.337897	0.254214	0.770585
6	0.001659	0.357358	0.867524	0.277019	0.118467	0.240200	0.739263	0.051301
7	0.715970	0.965837	0.099418	0.856479	0.742517	0.815446	0.334218	0.848140
8	0.914977	0.620546	0.245037	0.102492	0.254490	0.781520	0.087702	0.843995
9	0.677461	0.048887	0.826481	0.789318	0.733048	0.206479	0.023896	0.288263
10	0.190943	0.807337	0.807289	0.506631	0.375749	0.641292	0.408264	0.060240
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	0.672962	0.780742	0.956844	0.507715	0.767314	0.447289	0.063831	0.832386
4	0.971935	0.713985	0.569726	0.130933	0.445359	0.712783	0.714184	0.717960
5	0.440646	0.016866	0.830247	0.000669	0.000229	0.193904	0.008111	0.147356
6	0.712010	0.124147	0.269899	0.169880	0.103291	0.611226	0.226911	0.551235
7	0.738873	0.074857	0.790728	0.238081	0.108675	0.953965	0.978842	0.944501
8	0.785150	0.693697	0.305470	0.494033	0.762943	0.777388	0.288845	0.472716
9	0.497013	0.524833	0.287850	0.365214	0.378768	0.837909	0.252724	0.007599
10	0.162613	0.932469	0.722379	0.636367	0.716321	0.316012	0.670649	0.005734
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	0.814197	0.936996	0.896293	0.907959	0.940627	0.915199	0.591385	0.256157
4	0.058672	0.760510	0.263700	0.597756	0.657805	0.899708	0.342198	0.628523
5	0.721644	0.928540	0.402876	0.887964	0.894961	0.530496	0.275460	0.769510
6	0.360958	0.274531	0.770137	0.098165	0.095004	0.899728	0.389153	0.370810
7	0.821301	0.850561	0.223176	0.729395	0.728277	0.529035	0.703023	0.110923
8	0.906001	0.922548	0.926138	0.963057	0.931601	0.875985	0.571562	0.035327
9	0.949171	0.162170	0.737297	0.144849	0.356525	0.588980	0.323043	0.564623
10	0.060375	0.297092	0.636063	0.621419	0.437975	0.404324	0.647714	0.266918
Rot Input Data								
1	0.347449	0.426258	0.200096	0.750682	0.824820	0.029848	0.526090	0.564327
2	0.594618	0.667512	0.614079	0.634559	0.340689	0.534297	0.941419	0.115702
3	0.779471	0.528496	0.569845	0.406480	0.112316	0.089114	0.093269	0.368424
4	0.690940	0.144480	0.229078	0.287988	0.236025	0.194579	0.215946	0.203690
5	0.188338	0.901740	0.278104	0.978264	0.873541	0.831572	0.769113	0.743401
6	0.355608	0.418073	0.377130	0.647660	0.523603	0.863475	0.829478	0.315532
7	0.037430	0.865518	0.481526	0.749116	0.857566	0.229471	0.861226	0.410902
8	0.298893	0.059195	0.771945	0.307281	0.758229	0.148971	0.509822	0.311561
9	0.751365	0.815413	0.360448	0.877136	0.869469	0.403933	0.789332	0.552502
10	0.207395	0.158116	0.659377	0.752650	0.480489	0.850594	0.124907	0.122742

TABLE 7. ASSESSMENT OF THE PROPORTION METRICS FOR THE DYNAMIC AES BASED ON TESTS WITH SHORT DATA SEQUENCES

No. of Rouns	Freq. Test	Runs Test	Test for longest run of ones	Serial Test	AppEn. Test	CuSum. Test	Bit AutoCorr. Test	Byte Autocor. Test
AV1 Input Data								
1	98.98	98.92	99.08	98.99	98.85	99.08	99.28	99.16
2	98.92	98.95	99.07	98.95	98.91	98.98	99.24	99.22
3	99.00	98.96	99.09	99.03	98.95	99.08	99.25	99.22
4	98.99	98.96	99.08	99.01	98.95	99.08	99.25	99.21
5	99.00	98.95	99.09	99.01	98.94	99.08	99.26	99.23
6	98.99	98.94	99.09	99.01	98.93	99.06	99.24	99.21
7	98.98	98.97	99.07	99.01	98.95	99.06	99.25	99.21
8	98.98	98.95	99.08	99.01	98.94	99.06	99.25	99.22
9	98.99	98.95	99.09	99.03	98.95	99.07	99.25	99.22
10	98.99	98.95	99.08	99.00	98.93	99.07	99.25	99.21
HW Input Data								
1	98.74	98.96	99.23	98.65	98.67	98.82	99.01	99.43
2	98.91	99.11	99.29	99.01	99.05	99.04	99.31	99.20
3	98.98	98.94	99.06	99.00	98.93	99.07	99.24	99.22
4	98.98	98.95	99.10	99.02	98.94	99.07	99.25	99.20
5	99.00	98.94	99.10	99.01	98.95	99.08	99.26	99.21
6	98.97	98.95	99.09	98.99	98.91	99.05	99.22	99.21
7	98.99	98.94	99.10	99.02	98.94	99.06	99.25	99.21
8	99.00	98.95	99.09	99.03	98.96	99.07	99.25	99.21
9	99.00	98.97	99.08	99.02	98.95	99.07	99.26	99.21
10	99.00	98.95	99.09	99.02	98.95	99.06	99.25	99.21
LW Input Data								
1	99.14	99.16	99.30	99.12	99.04	99.15	99.37	99.47
2	99.14	99.04	99.18	99.11	99.12	99.07	99.25	99.25
3	99.01	98.94	99.09	99.02	98.95	99.09	99.25	99.21
4	99.00	98.94	99.08	99.01	98.93	99.08	99.24	99.22
5	98.99	98.96	99.08	99.02	98.94	99.06	99.25	99.20
6	99.00	98.96	99.08	99.05	98.97	99.07	99.26	99.22
7	98.99	98.95	99.08	99.01	98.93	99.07	99.25	99.22
8	98.99	98.96	99.09	99.03	98.95	99.08	99.26	99.20
9	98.99	98.95	99.08	99.01	98.94	99.07	99.25	99.20
10	98.99	98.95	99.07	99.03	98.95	99.07	99.25	99.22
Rot Input Data								
1	98.98	98.95	99.08	99.02	98.94	99.07	99.25	99.21
2	98.99	98.95	99.09	99.02	98.94	99.07	99.25	99.21
3	98.97	98.97	99.09	99.02	98.94	99.06	99.26	99.20
4	98.99	98.96	99.10	99.01	98.94	99.07	99.25	99.22
5	99.00	98.96	99.09	99.03	98.96	99.08	99.26	99.23
6	98.98	98.94	99.09	99.02	98.94	99.06	99.24	99.22
7	98.99	98.95	99.07	99.01	98.93	99.07	99.24	99.21
8	98.99	98.95	99.08	99.01	98.94	99.06	99.25	99.21
9	99.00	98.94	99.09	99.02	98.93	99.06	99.24	99.23
10	98.99	98.94	99.07	99.02	98.94	99.06	99.25	99.23

According to the results obtained, applying random keys in dynamic AES encryption requires at least 3 rounds to ensure randomness in the AV1, HW, and LW data sets. Therefore, with random key usage, the dynamic AES system must perform a minimum of 3 rounds to guarantee the randomness of the output data. Furthermore, the study indicates that the enhanced AES version provides similar randomness performance when compared to the traditional AES. While there is an additional cost in key generation and computation, the new algorithm is considered

more robust against differential and linear attacks due to its dynamic substitution and flexible key addition mechanisms.

Some limitations of our proposed method:

- When modifying AES by dynamically changing the SubBytes and AddRoundKey layers as proposed, additional key bits are required to support the dynamic generation, which increases the key overhead.

- Since both the SubBytes and AddRoundKey layers are made dynamic, it is no longer possible to implement the algorithm using static lookup tables. Instead, the S-box and dynamic XOR tables must be computed before each encryption session, which significantly reduces execution speed compared to the original AES.

- Furthermore, making the substitution and key addition layers dynamic also increases memory usage compared to the original algorithm, resulting in higher hardware resource consumption.

VI. CONCLUSION

In this paper, we proposed algorithms for generating dynamic key-dependent S-boxes and XOR tables, utilizing the unique properties of a matrix form we introduced: a binary block circulant matrix combined with a binary circulant matrix and a binary Hadamard matrix. The dynamic S-boxes generated exhibit strong cryptographic qualities, ensuring their security in use. The validity of the proposed algorithm is demonstrated through mathematical propositions outlined in this study. One of the main advantages of our algorithm is its simplicity compared to other methods for generating key-dependent S-boxes, requiring only 26 bits of extended key while being able to generate a large number of dynamic S-boxes and key-dependent XOR tables, approximately $\cong 2^{28}$ and $\cong 2^{33}$, respectively, thereby increasing the overall security level to $\cong 2^{33}$. Importantly, the generated 8-bit dynamic S-boxes maintain cryptographic properties similar to the original AES S-box. Additionally, we evaluated the random output standards of the modified dynamic AES block cipher's substitution and key addition layers, with results generated by Algorithm 1 using AV1, HW, LW, and ROT datasets. The results show that the dynamic modified AES block cipher achieves output randomness comparable to the original AES while significantly enhancing security.

REFERENCES

[1] Youssef, A.M., Tavares, S.E. và Heys, H.M., "A new class of substitution-permutation networks", *In: Proceedings of the Workshop on Selected Areas in Cryptography (SAC)*, vol. 96, pp. 132–147, 1996.

[2] Dodis, Y., Katz, J., Steinberger, J., Thiruvengadam, A. và Zhang, Z., "Provable security of substitution-permutation networks", *Cryptology ePrint Archive*, 2017.

[3] Sajjad, M., Shah, T., Hamza, R., Almutairi, B. và Serna, R.J., "Multiple color images security by SPN over the residue classes of Gaussian integer", *Scientific Reports*, vol. 15, no. 1, pp. 6425, 2025.

[4] Daemen, J. và Rijmen, V., "AES Proposal: Rijndael (Version 2)", *NIST AES Website*, 1999.

[5] Biham, E. và Shamir, A., "Differential cryptanalysis of DES-like cryptosystems", *Journal of Cryptology*, vol. 4, pp. 3–72, 1991.

[6] Gilbert, H. và Jean, J.E., "Differential Cryptanalysis", *Symmetric Cryptography, Volume 2: Cryptanalysis and Future Directions*, pp. 1, 2024.

[7] Matsui, M., "Linear cryptanalysis method for DES cipher", *In: Advances in Cryptology—EUROCRYPT'93*, vol. 12, pp. 386–397, 1993.

[8] Das, A., "Bit-Based MILP Modelling of Non-Bit-Permutation Linear Layers for Linear Cryptanalysis", *In: Proceedings of the 2024 19th Asia Joint Conference on Information Security (AsiaJCIS)*, pp. 1–8, IEEE, 2024.

[9] Schneier, B., "The Twofish encryption algorithm", *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 23, no. 12, pp. 30–34, 1998.

[10] Luong, T.T. và Linh, H.D., "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.

[11] Noura, H.N., Salman, O. và Chehab, A., "Conception of efficient key-dependent binary diffusion matrix structures for dynamic cryptographic algorithms", *Journal of Information Security and Applications*, vol. 76, pp. 103514, 2023.

[12] T. T. Luong, "Building the dynamic diffusion layer for SPN block ciphers based on direct exponent and scalar multiplication," *Journal of Science and Technology on Information Security*, vol. 1, no. 15, pp. 38–45, 2022.

[13] T. T. Luong and T. M. Phuong, "Generating efficient circulant-like MDS matrices for implementation," *Journal of Science and Technology on Information Security*, vol. 2, no. 22, pp. 58–68, 2024.

[14] Zhu, H., Tong, X., Wang, Z. và Ma, J., "A novel method of dynamic S-box design based on combined chaotic map and fitness function", *Multimedia Tools and Applications*, vol. 79, pp. 12329–12347, 2020.

[15] Waqas, U., Afzal, S., Mir, M.A. và Yousaf, M., "Generation of AES-like S-boxes by replacing affine matrix", *In: Proceedings of the 2014 12th International Conference on Frontiers of Information Technology*, pp. 159–164, IEEE, 2014.

[16] Agarwal, P., Singh, A. và Kilicman, A., "Development of key-dependent dynamic S-boxes with dynamic irreducible polynomial and affine constant", *Advances in Mechanical Engineering*, vol. 10, no. 7, pp. 1687814018781638, 2018.

[17] Alhadawi, H.S., Majid, M.A., Lambić, D. và Ahmad, M., "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.

[18] Hussain, I., Anees, A., Al-Maadeed, T.A. và Mustafa, M.T., "Construction of S-box based on chaotic map and algebraic structures", *Symmetry*, vol. 11, no. 3, pp. 351, 2019.

[19] Artuğer, F., "A novel algorithm based on DNA coding for substitution box generation problem", *Neural Computing and Applications*, vol. 36, no. 3, pp. 1283–1294, 2024.

[20] Maolood, A.T. et al., "Fast Novel Efficient S-Boxes with Expanded DNA Codes", *Security and Communication Networks*, vol. 2023, no. 1, pp. 5767102, 2023.

[21] Salih, A.I., Alabaichi, A.M. và Tuama, A.Y., "Enhancing advance encryption standard 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.

[22] Luong, T.T., Cuong, N.N. và Vo, B., "AES Security Improvement by Utilizing New Key-Dependent XOR Tables", *IEEE Access*, 2024.

[23] Luong, T.T. và Linh, H.D., "On generating new key dependent XOR tables to improve AES security and evaluating the randomness of the output of block ciphers", *International Journal of Information and Computer Security*, vol. 23, no. 1, pp. 16–39, 2024.

[24] Hoang, D.L. và Luong, T.T., "Enhancing block cipher security with key-dependent random XOR tables generated via hadamard matrices and Sudoku game", *Journal of Intelligent & Fuzzy Systems*, vol. 46, no. 4, pp. 7805–7821, 2024.

[25] Kim, S.H. và Han, G.T., "Enhanced hybrid encryption method using the half-key exchange and the dynamic S-box and shift-row in AES", *Information*, vol. 19, no. 2, pp. 683, 2016.

[26] Prasetiyo, B. và Ardian, M.N., "Enhancement security AES algorithm using a modification of transformation ShiftRows and dynamic S-box", *In: Journal of Physics: Conference Series*, vol. 1567, no. 3, pp. 032025, IOP Publishing, 2020.

[27] Navneet, J.R. et al., "AES algorithm with dynamic shift rows and bit permuted mix column", *In: Proceedings of the 2023 International Conference on Next Generation Electronics (NEleX)*, pp. 1–6, IEEE, 2023.

[28] Artuğer, F. và Özkaynak, F., "A method for generation of substitution box based on random selection", *Egyptian Informatics Journal*, vol. 23, no. 1, pp. 127–135, 2022.

[29] Nyberg, K., "Perfect nonlinear S-boxes", *In: Workshop on the Theory and Application of Cryptographic Techniques*, pp. 378–386, Springer, 1991.

[30] Burnett, L.D., "Heuristic optimization of Boolean functions and substitution boxes for cryptography", *PhD Thesis*, Queensland University of Technology, 2005.

[31] Gupta, K.C. và Ray, I.G., "Cryptographically significant MDS matrices based on circulant and circulant-like matrices for lightweight applications", *Cryptography and Communications*, vol. 7, pp. 257–287, 2015.

[32] Han, H., Tang, C., Lou, Y. và Xu, M., "Construction of efficient MDS matrices based on block circulant matrices for lightweight application", *Fundamenta Informaticae*, vol. 145, no. 2, pp. 111–124, 2016.

[33] Elumalai, R. và Reddy, A.R., "Improving diffusion power of AES Rijndael with 8x8 MDS matrix", *International Journal of Scientific & Engineering Research*, vol. 2, no. 3, 2011.

[34] Sajadieh, M. et al., "On construction of involutory MDS matrices from Vandermonde Matrices in $GF(2^q)$", *Designs, Codes and Cryptography*, vol. 64, pp. 287–308, 2012.

[35] Knudsen, L.R., "Truncated and higher order differentials", *In: Fast Software Encryption: Second International Workshop*, pp. 196–211, Springer, 1995.

[36] Kocher, P., Jaffe, J. và Jun, B., "Differential power analysis", *In: Advances in Cryptology - CRYPTO'99*, vol. 19, pp. 388–397, Springer, 1999.

[37] Bassham III, L.E. et al., "SP 800-22 Rev. 1a. A statistical test suite for random and pseudorandom number generators for cryptographic applications", *NIST Special Publication*, 2010.

ABOUT THE AUTHOR

Tran Thi Luong

Workplace: Academy of Cryptography Techniques, Vietnam

Email: luongtran@actvn.edu.vn

Education: Her 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. 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), an investigator of the project "Construction of cryptographic primitives for digital signature schemes and key exchange protocols using public key cryptography" (2018-2020), and an investigator of the project "Research on constructing a secure and efficient dynamic SPN block cipher algorithm" (2022-2023). She also served as a co-chair of special session of KSE 2023; VICRIS 2024.

Recent research direction: Cryptography, Coding theory, and Information Security.

Tên tác giả: **Trần Thị Lượng**

Cơ quan công tác: Học viện Kỹ thuật mật mã

Email: luongtran@actvn.edu.vn

Quá trình đào tạo: Nhận bằng Cử nhân tại Đại học Khoa học Tự nhiên Hà Nội, Việt Nam năm 2006; bằng Thạc sĩ về Kỹ thuật mật mã tại Viện Kỹ thuật Mật mã, Hà Nội, Việt Nam, năm 2012; bằng Tiến sĩ về Kỹ thuật mật mã tại Viện Kỹ thuật Mật mã, Hà Nội, Việt Nam, năm 2019. Tác giả là nghiên cứu viên chính của dự án "Xây dựng ma trận MDS và MDS động cho lớp khuếch tán động của mật mã khối" (2014-2015), là nghiên cứu viên của dự án "Xây dựng nguyên hàm mật mã cho các lược đồ chữ ký số và giao thức trao đổi khóa sử dụng mật mã khóa công khai" (2018-2020) và là nghiên cứu viên của dự án "Nghiên cứu xây dựng thuật toán mã hóa khối SPN động an toàn và hiệu quả" (2022-2023). Tác giả cũng là đồng chủ tịch phiên họp đặc biệt của KSE 2023; VICRIS 2024.

Hướng nghiên cứu hiện nay: Mật mã, lý thuyết mã, an toàn thông tin.

Truong Minh Phuong

Workplace: Academy of Cryptography Techniques, Vietnam

Email:

minhphuongh19@gmail.com

Education: He 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.

Recent research direction: Cryptogaphy.

Tên tác giả: **Trương Minh Phương**

Cơ quan công tác: Học viện Kỹ thuật mật mã.

Email: minhphuongh19@gmail.com

Quá trình đào tạo: Ông nhận bằng Kỹ sư Kỹ thuật Mật mã của Học viện Kỹ thuật mật mã năm 2012; bằng Thạc sĩ Kỹ thuật Mật mã tại Học viện Kỹ thuật mật mã năm 2018.

Hướng nghiên cứu hiện nay: Mật mã học

Nguyen Van Long

Workplace: Academy of Cryptography Techniques,Vietnam

Email: nvlong.bcy@gmail.com

Education: He received an Engineer's degree in Information Security of Telecommunication Systems at FSO Academy, Russian Federation in 2008; Ph.D degree at FSO Academy, Russian Federation in 2015.

Recent research direction: Cryptography, Coding theory and Information Security.

Tên tác giả: **Nguyễn Văn Long**

Cơ quan công tác: Học viện Kỹ thuật mật mã.

Email: nvlong.bcy@gmail.com

Quá trình đào tạo: Nhận bằng Kỹ sư Kỹ thuật Mật mã của Học viện Kỹ thuật mật mã năm 2012; bằng Thạc sĩ Kỹ thuật Mật mã tại Học viện Kỹ thuật mật mã năm 2018.

Hướng nghiên cứu hiện nay: Mật mã học, lý thuyết mã hóa và an toàn thông tin

Nguyen Nam Khanh

Workplace: Academy of Cryptography Techniques,Vietnam

Email: khanh6ts@gmail.com

Education: Cryptographic Engineering major student at the Academy of Cryptography Techniques.

Recent research direction: Cryptography

Tên tác giả: **Nguyễn Nam Khánh**

Cơ quan công tác: Học viện Kỹ thuật mật mã.

Email: khanh6ts@gmail.com

Quá trình đào tạo: Sinh viên chuyên ngành Kỹ thuật mật mã tại Học viện Kỹ thuật mật mã.

Hướng nghiên cứu hiện nay: Mật mã học