*Adv Syst Sci Appl* 2025; **3**; 74-95
Published online at https://ijassa.ipu.ru.

# A Word-Oriented Substitution–Permutation Network Cipher with Security Evaluation based on Active S-Box Bounds and a Dynamic Advanced Encryption Standard-Based Variant

Nguyen Van Long[1], Tran Thi Luong[1*],
Nguyen Bui Cuong[2], Truong Minh Phuong[2]

[1)] *Academy of Cryptography Techniques, Hanoi, Vietnam*

[2)] *Institute of Cryptographic Science and Technology, Hanoi, Vietnam*

**Abstract**: The Substitution–Permutation Network (SPN) serves as a foundational structure in the design of modern block cipher algorithms due to its effective realization of two essential properties: confusion and diffusion. Currently, the security of SPNs is actively studied, with the S-box structure and the diffusion layer being key components analyzed to enhance resistance against attacks. In this paper, we propose a generalized SPN-based cipher model inspired by the Advanced Encryption Standard (AES) structure. We then present a novel theoretical approach to evaluating the security of this SPN cipher, based on a lower bound of the number of active S-boxes. This forms the basis for identifying the roles and cryptographic properties required of the component transformations in this type of cipher. Next, we propose a dynamic block cipher algorithm based on the AES cipher, which not only ensures the required level of security but also inherits the implementation advantages of the original AES. The dynamic AES block cipher demonstrates higher security compared to the original AES, passes randomness evaluation standards, and is efficiently implementable. These results are significant in guiding the design of secure and flexible block cipher algorithms, while also providing a theoretical foundation for the selection and evaluation of secure cryptographic components in modern cryptosystems.

***Keywords***: Dynamic block cipher, active S-box, SPN block cipher, AES.

## 1. INTRODUCTION

Block ciphers serve as fundamental building blocks in numerous modern network security protocols. Typically, these ciphers are designed using an iterative round-based structure, where each round comprises two primary layers [1]: a confusion layer and a diffusion layer. Among various designs, the SPN architecture is widely adopted for symmetric encryption due to its effective combination of nonlinear substitution (via S-boxes) and permutation (P-layer) to achieve both confusion and diffusion. In most SPN-based block ciphers [2–5], the diffusion layer is implemented using a linear transformation, while the confusion layer relies on small nonlinear substitution boxes, commonly 4 or 8 bits in size. Given their small size, these S-boxes offer limited nonlinearity, which highlights the importance of the diffusion layer in propagating the S-box nonlinearity across the entire cipher block. As emphasized in Shannon's seminal work [1], strong cryptographic security depends on the joint and inseparable use of both confusion and diffusion mechanisms.

Among the class of algorithms based on the above principle, AES [2] is arguably the most prominent representative of SPN block ciphers and has attracted the most extensive research in this field. From a security perspective, the designers of AES demonstrated that the algorithm achieves at least 25 active S-boxes after four rounds of operation [2]. This issue was later

* Corresponding author: luongtranhong@gmail.com

revisited and extended in [6] to the case involving an 8×8 MDS matrix. However, the analysis in [6] is based on certain specific differential trails. As a result, the lower bounds on the number of active S-boxes for 3, 7, 11, ... rounds proposed in that study are not entirely accurate (see Table 1 and Table 3 in [6]).

Thanks to its widespread adoption and long-established security, AES has become the focus of numerous studies aimed at enhancing and developing more optimized variants. One prominent research direction involves the construction of dynamic AES versions, in which core components and transformations within the round function are designed to be key-dependent or time-varying. From a design philosophy standpoint, the concept of dynamic block cipher security was introduced by L. Knudsen in [7], and this viewpoint was later supported by the original designers of AES, also in [7].

For dynamic versions of the AES block cipher, numerous research studies have been published, each proposing different dynamicization methods. Some approaches focus on making the AES S-boxes key-dependent [8–14], while others emphasize generating key-dependent MixColumn transformations [15–18]. Notably, there are studies that explore the dynamicization of both the S-box and MixColumn components of AES [19], and even the dynamicization of all three transformations—S-box, MixColumn, and ShiftRow—has garnered attention [20–22]. Another ongoing research direction involves making the XOR operation dynamic within AES [23–27]. Additionally, the authors in [28] proposed a dynamic AES approach where the number of encryption rounds varies for each plaintext block.

In the context of dynamic S-box techniques for AES, Al-Dweik and colleagues [9] proposed a way to create key-dependent S-boxes that exhibit desirable algebraic characteristics such as nonlinearity, BIC, and SAC. However, other crucial cryptographic attributes of these S-boxes were not addressed in their study. Another approach to generating key-based S-boxes for AES was introduced in [10], which involves rearranging the S-box structure by employing a simulated key expansion algorithm. Additionally, the authors in [11] presented a technique to produce modified S-boxes by permuting the original AES S-box. These adaptable S-boxes depend on a secret key and incorporate affine constants and an unconventional polynomial. With each additional bit of the key, a newly rearranged S-box is generated, thereby enhancing the cipher's complexity. In [12], an S-box generation algorithm based on the Playfair cipher was examined. The authors evaluated and compared criteria such as balance and avalanche standards between the modified block cipher with dynamic S-boxes and the original AES. However, similar to [9], other important cryptographic properties were not addressed. In [13], the authors proposed a key-dependent dynamic S-box generation algorithm for AES, utilizing a pseudo-random number generator (PRNG) based on three linear feedback shift registers (LFSRs). In [14], a completely different approach was explored, where dynamic S-boxes are generated based on Epoch time or Unix time during each encryption cycle. However, the method proposed in [14] lacks a clear explanation of the S-box generation mechanism when using Epoch or Unix time as input, which makes it difficult for readers to understand. Notably, the cryptographic properties of the resulting dynamic S-boxes were not mentioned at all.

Returning to the problem of dynamic AES block cipher design at the MixColumns transformation layer, Murtaza et al. [15] proposed a key-dependent MixColumns transformation based on scalar multiplication, where the scalar multiplication is performed on the rows of the MDS matrix. Similar approaches were also applied in [16]. Additionally, the authors in [16] explored dynamic MixColumns transformations using exponentiation. Along the same lines, the authors in [17] utilized Self-Reciprocal Recursive MDS matrices to introduce dynamism into the MixColumns layer. Meanwhile, in [18], a collection of n×n binary matrices was presented that can be used to generate dynamic matrices resembling both the AES matrix and recursive MDS matrices.

Regarding the dynamic modification of multiple transformation components within the AES round function, a notable example is presented in [19]. The authors introduced dynamic S-boxes and new MixColumns matrices that retain favorable cryptographic properties while

 

developing a dynamic version of AES. Beyond the SubBytes and MixColumns components, the ShiftRows component was also made dynamic in the study by the authors in [20], where the dynamic approach is similarly key-dependent. The approach of constructing key-dependent, randomly generated dynamic transformation components was proposed in [21], where all three transformations—SubBytes, ShiftRows, and MixColumns—within the AES round function are made dynamic. However, no security evaluations were provided in this study. Continuing with the dynamic modification of these three cryptographic components, [22] introduced a new, efficient, key-dependent AES algorithm.

For the AddRoundKey transformation in the AES round function, the authors in [23, 24] introduced improved techniques using key-dependent XOR tables generated through 3D chaotic maps. These XOR tables are based on initial secret parameters, resulting in a dynamic AES version where the AddRoundKey layer follows the rules of the new XOR tables. This approach was further investigated and developed by T. T. Luong et al. in [25-27], employing different XOR table generation techniques.

Recently, in 2024, Adamu et al. published a study on dynamic AES [28]. In their approach, the number of encryption rounds for each data block is made dynamic and key-dependent. While this represents a novel method, we assess that it may not be fully reasonable from a security standpoint, since the number of rounds—such as 10 for AES-128—is a threshold chosen based on known cryptanalysis attacks. Randomizing the number of rounds without ensuring the necessary security bounds could compromise the entire system.

Our observations are as follows: According to the philosophy of dynamic cryptosystems proposed by Knudsen in [7], introducing dynamism is meaningful from a security perspective. However, to the best of our understanding, dynamic methods must comply with minimal security principles. For example, cryptographic components such as the MDS property of the matrix in AES's MixColumns transformation must be preserved, the cryptographic properties of the generated dynamic S-boxes should not be inferior to the original version, and the new ShiftRow operation must still ensure effective diffusion of active bytes, among others. Secondly, dynamic methods need to consider the potential significant impact on the implementation of the proposed solution. The research works we reviewed above seem to have overlooked these issues. In particular, those dynamic approaches fail to leverage AES's table lookup implementation for optimizing speed. Some studies mention recalculating these tables after each dynamic change. However, for applications requiring frequent key changes, this recalculation can be even more complex than the encryption operations themselves for a single data block. This is likely a limitation affecting the practical applicability of dynamic block ciphers.

Our contributions. In this study, we first propose a generalized AES-like SPN cipher model. Then, we introduce a novel approach to evaluate the security of these ciphers based on estimating a lower bound on the number of active S-boxes. Accordingly, we present a generalized AES-like SPN cipher model and provide a theoretical proof for the results obtained. From these results, we identify the crucial cryptographic roles and properties that each component of the cipher's round function must possess. Building on this foundation, we propose an improved dynamic AES version with dynamic components that satisfy the necessary security requirements. Notably, our dynamic method does not require recomputing lookup tables in the optimized table-based implementation. This is a significant distinction compared to previous works, as it effectively addresses the practical implementation challenges of dynamic block ciphers. The dynamic AES cipher is carefully evaluated, demonstrating higher security than the original AES, meeting randomness standards, and enabling efficient implementation.

Based on this, the remainder of the paper is organized as follows. Preliminary knowledge and notation are presented in Section 2. Section 3 introduces a generalized SPN block cipher model along with theoretical results on the lower bound of active S-boxes for four rounds of this cipher. A dynamic AES block cipher algorithm is proposed in Section 4. Section 5

 

provides a security analysis of the dynamic AES cipher. Section 6 evaluates the randomness properties of the dynamic AES cipher. Section 7 analyzes the implementation efficiency of the dynamic AES cipher on a software platform. The conclusion is in Section 8.

## 2. PRELIMINARIES

Some notations used in this paper include:
$\mathbb{F}_2$: the binary field consisting of two elements, 0 and 1.
$\mathbb{F}_{2^n}$: the Galois field containing $2^n$ elements, with addition denoted by XOR ($\oplus$) and multiplication denoted by ($\otimes$).

$\mathbb{Z}^+$: the set of positive integers.
$x \parallel y$: the concatenation of bit string $y$ to bit string $x$.
$\{x\}^d$: a bit string $x$ of length $d$ bits.
$wt(x)$: the Hamming weight of the binary vector $x$.

Given a vector $x$ represented as $x = x_1 \parallel x_2 \parallel \cdots \parallel x_m$ with $x_i \in \{0,1\}^n$, the quantity $wt_n(x)$ is called the bundle weight of the vector $x$ and is defined as follows:

$$wt_n(x) = \#\{x_i | x_i \neq 0\}.$$

Let $A$ be a linear transformation: $\mathbb{F}_{2^n}^m \longrightarrow \mathbb{F}_{2^n}^m$, we define the branch number of $A$, denoted by $Br_n(A)$, as follows:

$$Br_n(A) = \min_{x \in \mathbb{F}_{2^n}^m, x \neq 0} (wt_n(x) + wt_n(A(x)))$$

Here, $\lceil x \rceil$ denotes the smallest integer $q$ such that $q \geq x$.

## 3. PROPOSE A GENERALIZED SPN CIPHER MODEL AND EVALUATE A LOWER BOUND ON THE NUMBER OF ACTIVE S-BOXES.

In this section, we first introduce a generalized word-oriented SPN cipher model. We then present some theoretical results on the lower bound of the number of active S-boxes for this type of cipher. The security of this block cipher structure against linear and differential cryptanalysis is typically based on the lower bound of the number of active S-boxes.

### *3.1. Propose a Generalized SPN Cipher Model Inspired by the AES Structure*

In this section, we present a word-oriented SPN block cipher model based on three transformations that play roles similar to the SubBytes, ShiftRows, and MixColumns operations in AES.

First, we define an *m-diffuse linear transformation* as follows:

***Definition 1.*** *Let $\pi$ be a word-oriented linear transformation from $\{0,1\}^{m \times t}$ to $\{0,1\}^{m \times t}$, where $m, t \in Z^+$, and $t = m \times l' \times n$. Then, $\pi$ is called an* ***m-diffuse*** *transformation if each input and output state (belonging to $\{0,1\}^{m \times t}$) is partitioned into $m$ disjoint consecutive subsets of $\{0,1\}^t$, satisfying the following property: in each output subset, there exist $l'$ distinct $w$-bit words originating from all the input subsets of the transformation.*

For each string $x \in \{0,1\}^{m \times t}$ with $m, t \in Z^+$, and $t = m \times l' \times n$, besides the representation as $n$-bit words (in which case we can compute the bundle weight $wt_n(x)$), $x$ can also be represented as $t$-bit words, where $t = m \times l' \times n$. Let $x = x_1 \parallel \cdots \parallel x_m$, with $x_i \in \{0,1\}^t$, $1 \leq i \leq m$. In this representation, we can also compute the bundle weight $wt_t(x)$. In this form, each $x_i$ is called a substate, and the transformation $\pi$ can be illustrated in Figure 3.1.

 

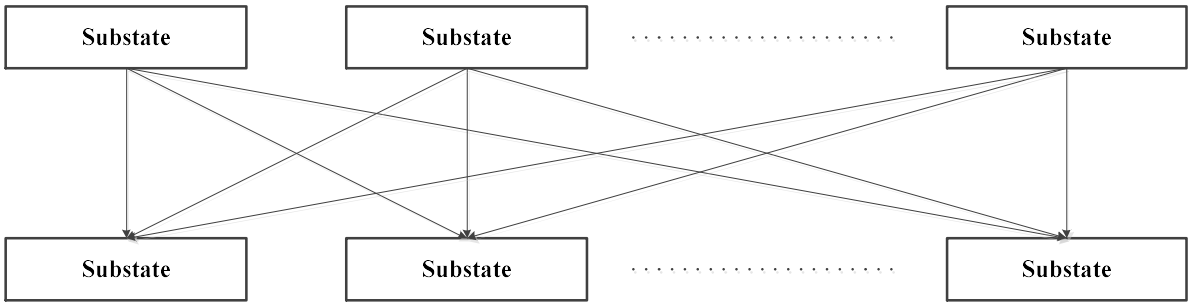


**Fig. 3.1.** Illustration of the transformation π, where the black arrows represent the transfer of l′ n-bit words from each input state to the output substates

We present some properties related to the $m$-diffuse transformation as follows:

***Lemma 3.1.*** *Let $\pi$ be an $m$-diffuse transformation from the set $\{0,1\}^{m\times t}$ to $\{0,1\}^{m\times t}$, where $m, k \in Z^+$, and $t = m \times l' \times n$. Then:*

1. *$\pi^{-1}$ is also m-diffuse.*
2. *$wt_t(\pi(x^0)) \times l' \geq wt_n(x^{0,j})$ for all $j = 1, \dots, m$,*

*where $x^0 = x^{0,1} \parallel x^{0,2} \parallel \cdots \parallel x^{0,m}$, and $x^{0,i} \in \{0,1\}^t$.*

**Proof.**

It is straightforward to verify this from the definition of an $m$-diffuse permutation.

Consider the weight of $x^0$ as follows:

- When $wt(x^0) = 0$, we have $wt_t(x^0) = \min\limits_{j=1,\dots,m} wt_n(x^{0,j}) = 0$.
- When $wt(x^0) > 0$, there exists some $j$ such that $x^{0,j} \neq 0 \in \{0,1\}^t$. Suppose $wt_n(x^{0,j}) = d$, where $1 \leq d \leq t$. Then, $\left\lceil \frac{d}{l'} \right\rceil$ $n$-bit blocks in $x^{(0,j)}$ have nonzero bundle weight. Since the permutation $\pi$ has the $m$-diffuse property, these $\left\lceil \frac{d}{l'} \right\rceil$ $n$-bit blocks in $x^{(0,j)}$ will be diffused into $\left\lceil \frac{d}{l'} \right\rceil$ $t$-bit substate blocks in $\pi(x^0)$. Thus,

$$wt_t(\pi(x^0)) \geq \left\lceil \frac{d}{l'} \right\rceil = \left\lceil \frac{wt_n(x^{0,j})}{l'} \right\rceil$$

Therefore, $wt_t(\pi(x^0)) \times l' \geq wt_n(x^{0,j})$ for all $j = 1, \dots, m$. ■

Next, we consider the local diffusion transformation $\theta: \{0,1\}^{t\times m} \to \{0,1\}^{t\times m}$, constructed from smaller-dimensional diffusion transformations $\theta_i: \{0,1\}^t \to \{0,1\}^t$ for $1 \leq i \leq m$, defined as follows:

$$\theta(x) = \theta_1(x_1) \parallel \theta_2(x_2) \parallel \cdots \parallel \theta_m(x_m) \quad (1)$$

where $x = x_1 \parallel x_2 \parallel \cdots \parallel x_m, x_i \in \{0,1\}^t$.

In this case, we obtain the following result about a transformation composed of $\pi$ and $\theta$ to achieve an optimally diffusive transformation with a higher dimension.

***Theorem 3.1.*** *Let $\pi, \pi': \{0,1\}^{t\times m} \to \{0,1\}^{t\times m}$ be two $m$-diffuse permutations, and let $\theta: \{0,1\}^{t\times m} \to \{0,1\}^{t\times m}$ be a transformation defined by (1) based on $m$ n-bit-oriented transformations $\theta_i: \{0,1\}^t \to \{0,1\}^t$ satisfying $Br(\theta_i) = t_i$. Then, the transformation $\sigma = \pi \circ \theta \circ \pi'$ satisfies:*

$$min_{x\in\{0,1\}^{t\times m}, x\neq \mathbf{0}}\{wt_t(x) + wt_t(\sigma(x))\} \times l' \geq min_{i\in\{1,..,m\}}\{t_i\}$$

**Proof.**

Consider the input to the transformation $\sigma$ of the form $x^0 = (x_1^0, .., x_{t\times m}^0)$, where $x_j^0 \in \{0,1\}$ for $j = 1, \dots, t \times m$. Let $x^1, x^2, y \in \{0,1\}^{t\times m}$ satisfy $x^1 = \pi(x^0), x^2 = \theta(x^1)$, and $x^3 = \pi'(x^2)$. Then, we have:

$$min_{x^0\in\{0,1\}^{t\times m},x^0\neq\mathbf{0}}\{wt_t(x^0)+wt_t(\sigma(x^0))\}=min_{x\in\{0,1\}^{t\times m},x\neq\mathbf{0}}\{wt_t(x^0)+wt_t(x^3)\}$$
$$=min_{x^1\in\{0,1\}^{t\times m},x\neq\mathbf{0}}\left\{wt_t(\pi^{-1}(x^1))+wt_t\left(\pi(\theta(x^1))\right)\right\}.$$

Since $x^1\neq\mathbf{0}$ (where $\mathbf{0}$ is the all-zero vector), there exists a subset in the partition of $x^1$ into $m$ disjoint consecutive subsets of the form $x^{1,j}=(x^1_{(j-1)\times t+1},\dots,x^1_{j\times t})$ satisfying $wt_n(x^{1,j})=d$ with $1\leq d\leq l'\times m$, $1\leq j\leq m$. Then, since $Br(\theta_j)=t$, we have:

$wt_n(x^{1,j})+wt_n\left(\theta_j(x^{1,j})\right)\geq t_j$.

From Lemma 3.1 and the fact that $\pi$ and $\pi^{-1}$ are $m$-diffuse transformations, we have:

$wt_t(\pi^{-1}(x^1))\times l'\geq wt_n(x^{1,j})$ và $wt_t(\pi(\theta_j(x^{1,j})))\times l'\geq wt_n\left(\theta_j(x^{1,j})\right)$

It follows that:

$$\left(wt_t(\pi^{-1}(x^1))+wt_t\left(\pi(\theta(x^1))\right)\right)\times l'\geq t_j,\ \forall x^1\in\{0,1\}^{t\times m}\backslash\{0\},j=1,\dots,m.$$
$$min_{x\in\{0,1\}^{t\times m},x\neq\mathbf{0}}\{wt_t(x)+wt_t(\sigma(x))\}\times l'\geq\ min_{i\in\{1,..,m\}}\{t_i\}.\ \blacksquare$$

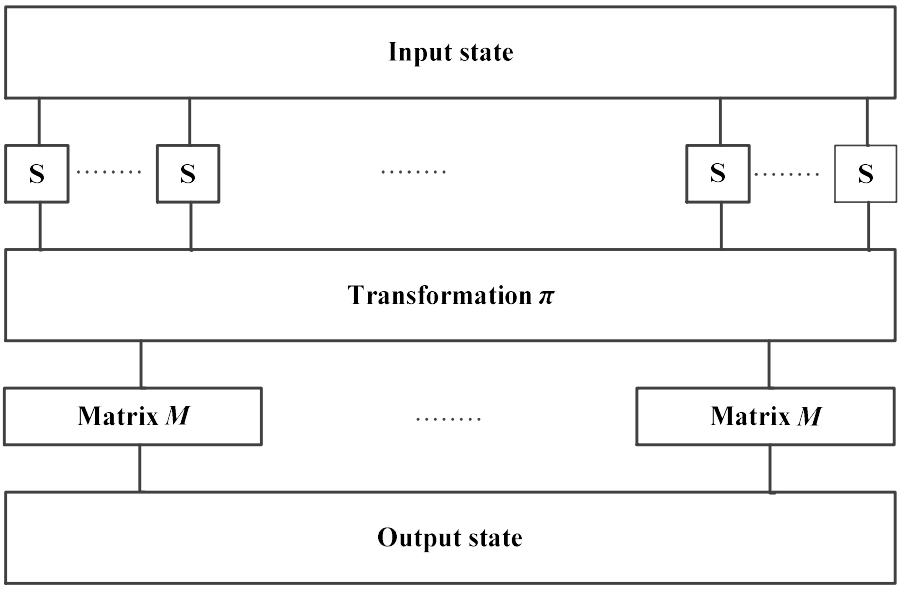


**Fig. 3.2.** Description of the round function of the proposed generalized SPN cipher

From the two diffusion transformations above, we propose a specific iterative block cipher with an SPN structure and an $l$-bit word-oriented round function, where the block size is $n$ bits, with $l=m^2\times l'\times w$. The input and output states of the round function of the block cipher are strings represented as states consisting of $m^2\times l'$ blocks of $n$ bits, denoted by $x=(x_1,x_2,\dots,x_{m^2\times l'})$, where each $x_i$ is an $n$-bit block. These are transformed through the following four basic operations:

*The round key addition:* $K_i$ with the input state $x$. However, this addition does not affect the number of active S-boxes for different fault patterns, so we can omit it from consideration.

*The nonlinear transformation* $\gamma$*:* uses a $w$-bit S-box $s$, which is applied in parallel across the state as follows:

$$\gamma(x)=\gamma(x_1,x_2,\dots,x_{l'\times m^2})=s(x_1)\parallel s(x_2)\parallel\cdots\parallel s(x_{l'\times m^2-1})\parallel s(x_{l'\times m^2})$$

*The permutation* $\pi$: provides full diffusion, permuting the $n$-bit words within each state as follows:

$\pi(x_1,\dots,x_{t\times m})=x_{\rho(1)}\parallel x_{\rho(2)}\parallel\cdots\parallel x_{\rho(t\times m)}$, where $\rho$ is a permutation of the set $\{1,\dots,l'\times m^2\}$ such that $\pi$ s an $m$-diffuse from $\{0,1\}^{t\times m}\to\{0,1\}^{t\times m}$, with $t=l'\times m\times n$.

 

*The mixing step* $\theta$: is applied in parallel to substates of size $(m \cdot l') \times (m \cdot l')$ over the field $GF(2^n)$, based on a matrix $M$ with branch number $Br(M)$. The state is divided into $m$ substates, each of which is transformed through this mixing step.

We refer to the SPN cipher with this proposed structure as $\gamma\pi\theta\_SPN$. This structure is illustrated in Figure 3.2.

Based on the proposed generalized SPN cipher ($\gamma\pi\theta\_SPN$), we describe several existing SPN block ciphers in terms of the transformations and parameters of the proposed generalized SPN structure.

Table 3.1 lists several word-oriented block ciphers that use this structure with a block size of 128 bits and their corresponding transformations. For the case of 128-bit byte-oriented block ciphers, Table 3.1 includes all existing SPN ciphers whose transformations satisfy the properties and conditions we analyzed above. Note that for $m = 8$, there is no permutation $\pi$ that satisfies the $m$-diffuse property.

**Table 3.1.** Some specific instances of the proposed generalized SPN structure

| Cipher | $n$ | $m$ | $l'$ | $\gamma$ | $\pi$ | $\theta$ | Source |
|---|---|---|---|---|---|---|---|
| **AES** | 8 | 4 | 1 | SubBytes | ShiftRows | MixColumns ($4 \times 4$ MDS matrix over $\mathbb{F}_{2^8}$) | [2] |
| **Kalyna128** | 8 | 2 | 4 | $\pi_l'$ | $\tau_l$ | $\psi_l$ ($8 \times 8$ MDS matrix over $\mathbb{F}_{2^8}$) | [4] |
| **Kuznyechik** | 8 | 1 | 16 | $\pi'$ | Identity mapping | $L$ ($16 \times 16$ MDS matrix over $\mathbb{F}_{2^8}$) | [5] |

### *3.2. Lower Bound Evaluation of the Number of Active S-Boxes in the Proposed Generalized SPN Cipher Model*

In this section, we present the lower bound on the number of active S-boxes over four rounds of the proposed generalized SPN cipher $\gamma\pi\theta\_SPN$.

***Theorem 3.2.*** *Four consecutive rounds of a block cipher using a round function structured in the form of* $\gamma\pi\theta\_SPN$ *will have a number of active S-boxes no less than* $\left\lceil \frac{Br(M)}{l'} \right\rceil \times Br(M)$.

**Proof.**

For the convenience of presentation, we denote the input difference of a transformation—such as the transformation $\theta$—at round $i$ as $\Delta in^{\theta,i} = (\Delta in_1^{\theta,i}, \dots, \Delta in_l^{\theta,i})$, where $\Delta in_j^{\theta,i} \in \{0,1\}, 1 \le j \le l$. Then, the total number of active S-boxes over rounds 1, 2, 3, and 4 is:

$$n_1 + n_2 + n_3 + n_4 = wt_n(\Delta in^{\gamma,1}) + wt_n(\Delta in^{\gamma,2}) + wt_n(\Delta in^{\gamma,3}) + +wt_n(\Delta in^{\gamma,3}).$$

Since the $n$-bit S-boxes only shuffle within $n$-bits, and the transformations $\pi$ and $\pi^{-1}$ are word-oriented $m$-diffuse transformations over $w$-bit words, it is easy to prove that:

$$wt_n(\Delta in^{\gamma,r}) = wt_n(\Delta out^{\gamma,r}), \forall r \in \{1, \dots, 4\}$$

$$wt_t(x) = wt_t\left(\pi\left(\theta\left(\pi^{-1}(x)\right)\right)\right), \forall x \in \{0,1\}^{m \times t}.$$

To facilitate the calculation of the number of active S-boxes, we add two transformations, $\pi$ and $\pi^{-1}$, to the output of the $\theta$ transformation in the second round. Then, by applying the result of Theorem 3.1 to the inputs of the three transformations—$\pi$ (of the second round), $\theta$ (of the second round), and the added $\pi$ transformation—we obtain:

$$\left(wt_t(\Delta in^{\pi,2}) + wt_t\left(\pi\left(\theta\left(\pi(\Delta in^{\pi,2})\right)\right)\right)\right) \times l' \ge Br(M)$$

Since: $\pi\left(\theta\left(\pi(\Delta in^{\pi,2})\right)\right) = \pi\left(\Delta in^{\theta,3}\right)$, therefore, we obtain:

$$\left(wt_t(\Delta in^{\pi,2}) + wt_t\left(\pi\left(\Delta in^{\theta,3}\right)\right)\right) \times l' \geq Br(M)$$

Thus, the input differences of the substitution-diffusion-substitution structures in rounds (1, 2) and (3, 4) will activate a number of S-boxes equal to $\left\lceil \frac{Br(M)}{l'} \right\rceil$ (see illustration in Figure 3). Since in each of these patterns, the number of active S-boxes is no less than $Br(M)$, the total number of active S-boxes for any input difference will be no less than $\left\lceil \frac{Br(M)}{l'} \right\rceil \times Br(M)$.■

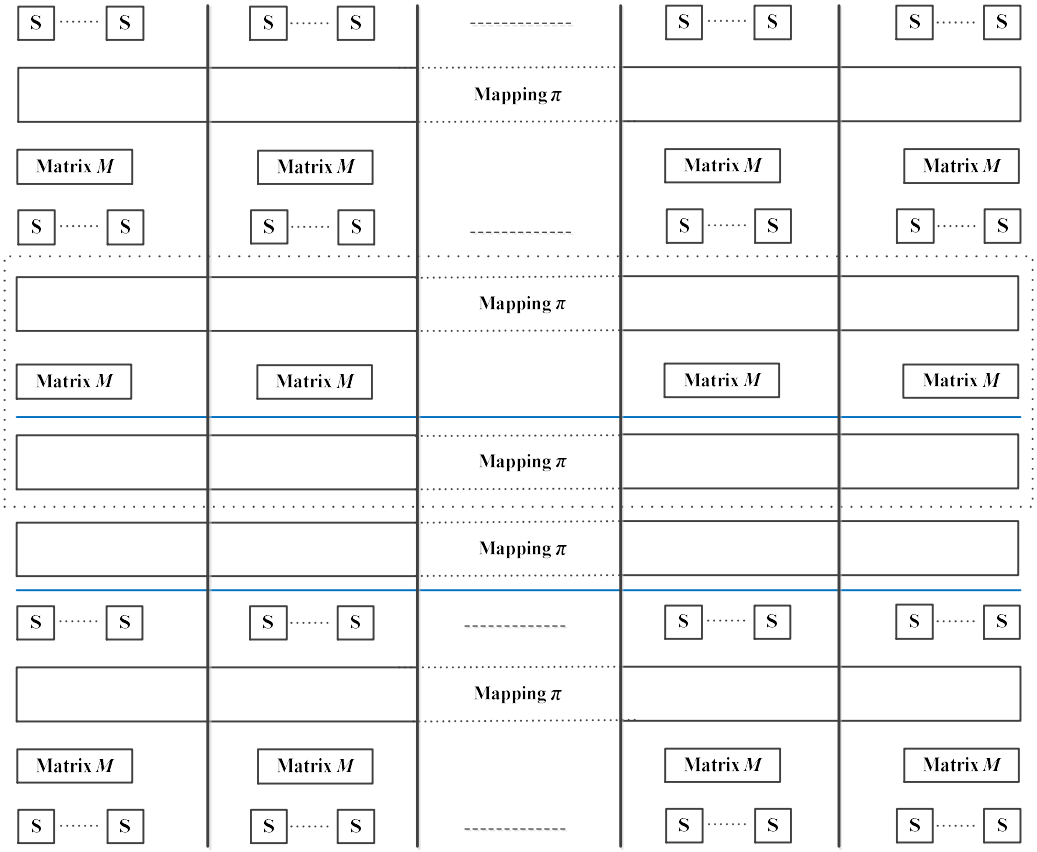


**Fig. 3.3.** Illustration of difference propagation through four rounds of the block cipher γπθ_SPN

From Theorem 3.2, we derive the following corollary.

***Corollary 3.1.*** *Four consecutive rounds of iterative block ciphers using a round function structured like* $\gamma\pi\theta\_SPN$ *, where the transformation* $\theta$ *employs MDS matrices of size* $(m \cdot l') \times (m \cdot l')$ *over the field* $\mathbb{F}_{2^n}$*, will have a number of active S-boxes no less than* $(m + 1) \times (m \cdot l' + 1)$.

Indeed, this follows directly from the fact that $Br(M) = (m \cdot l' + 1)$.

By applying Corollary 3.1, we can recover known results for practical SPN block ciphers (see Table 2.2). This further confirms the validity of our Theorem 3.2and Corollary 3.1.

**Table 3.2.** Lower bounds on the number of active S-boxes over 4 rounds for several SPN ciphers

| Cipher | $n$ | $m$ | $l'$ | Lower bound on the number of active S-boxes |
|---|---|---|---|---|
| AES | 8 | 4 | 1 | 25 |
| Kalyna | 8 | 2 | 4 | 27 |
| Kuneztik | 8 | 1 | 16 | 34 |

***Remark 3.1.*** A natural question arises: if we remove the $m$-diffuse condition from the ShiftRows-type transformation such as $\pi$ in $\gamma\pi\theta - SPN$, will the lower bound on the number of active S-boxes over four rounds be affected? The answer is yes. We examined a specific

 

case with AES, where its ShiftRows transformation does not satisfy the $m$-diffuse property, and the result shows that for its four-round transformation, the number of active S-boxes is always less than 25.

Thus, the role of the transformation $\pi$ in the generalized SPN cipher model is truly significant. Its design not only directly impacts the security level but also affects the implementation efficiency of the block cipher algorithm. Based on the theoretical foundation presented in this section, in the next section, we propose a modification to the ShiftRows layer in AES to obtain a key-dependent dynamic version of AES, which maintains the required level of security without significantly compromising the high-speed implementation capability of the original AES algorithm.

## 4. PROPOSAL OF A SECURE AND EFFICIENT KEY-DEPENDENT DYNAMIC AES ALGORITHM

The original AES algorithm is not only secure but also achieves high execution speed on many platforms. The implementation method using lookup tables for this type of block cipher has become popular and familiar within the cryptographic community. It is regarded as a "must-have property" of a byte-oriented SPN block cipher. Many dynamic AES algorithm approaches have been proposed, but it seems that they do not maintain the flexible implementation using the lookup tables of the original AES. This may be a limitation in the applicability of the dynamic algorithm. To overcome this issue, we propose a modification in the ShiftRows transformation of the original AES to obtain a secure and efficient key-dependent dynamic AES variant.

As we know, the AES algorithm processes a 128-bit data block divided into 16 bytes arranged into a $4 \times 4$ two-dimensional state matrix.

We name the proposed dynamic AES algorithm **AES_DST**, which includes the following basic transformations:

- The AddRoundKeys operation is the same as in the original AES
- The SubBytes operation is the same as in the original AES
- The MixColumns operation is the same as in the original AES
- The key schedule is the same as in the original AES

The byte permutation consists of two key-dependent transformations: ShiftRows and TranBytes. Among them, the ShiftRows operation is the same as in the original AES, while TranBytes is a transpose of the $4 \times 4$ data state matrix. These two transformations are selected based on a secret key.

The steps performed in the dynamic block cipher algorithm AES_DST are described as in Algorithm 4.1.

**Algorithm 4.1. The dynamic block cipher algorithm AES_DST**

**INPUT:**

- A 128-bit input data block (*in*) is organized into a $4 \times 4$ state matrix *state*.
- The master key (*masterKey*) has a length of $k$ bits ( $k = 128, 192, 256$ bits corresponding to the dynamic AES-128, AES-192, and AES-256 versions with $n_r = 10, 12, 14$ rounds, respectively).
- The dynamic key $kd$ has a length of $n_r$ bits.

**OUTPUT:** 128-bit ciphertext block.

**Step 1.** From the master key (*masterKey*) compute the array of round keys ($w$) through the key schedule.

**Step 2.** Encryption process – Procedure $Cipher(in, dKey, n_r, w)$:

1. $state \leftarrow in$
2. $state \leftarrow AddRoundKey(state, w[0..3])$

 

3. **for** *round* **from** 1 **to** $n_r - 1$ **do**
4. $state \leftarrow SubBytes(state)$
5. **if** $kd[round - 1] = 1$ **then**
6. $state \leftarrow ShiftRows(state)$
7. **else**
8. $state \leftarrow TranBytes(state)$
9. **end if**
10. $state \leftarrow MixColumns(state)$
11. $state \leftarrow AddRoundKeys(state, w[4 \cdot round..4 \cdot round + 3])$
12. **end for**
13. $state \leftarrow SubBytes(state)$
14. **if** $kd[n_r] = 1$ **then**
15. $state \leftarrow ShiftRows(state)$
16. **else**
17. $state \leftarrow TranBytes(state)$
18. **end if**
19. $state \leftarrow AddRoundKeys(state, w[4 \cdot r_r..4 \cdot r_r + 3])$
20. **return** $state$.

Here, we only describe the procedure for the encryption process; the decryption process can be performed similarly but in reverse order.

## 5. SECURITY EVALUATION OF THE DYNAMIC BLOCK CIPHER AES_DST

First, we can observe that the *TranBytes* operation is a transposition of the state matrix. It can be easily verified that it satisfies Definition 1, or in other words, it possesses the $m$-diffusion property. This means that four encryption rounds of our dynamic AES variant still conform to the results stated in Corollary 3.1.

Figure 4 illustrates the encryption/decryption diagram of the dynamic block cipher AES_DST.

In the field of block cipher analysis, several attack techniques have been developed, including differential cryptanalysis [29, 30, 31], linear cryptanalysis [32, 33], and algebraic attacks [31, 34], among others. This section concentrates on two of the most influential and widely studied methods: linear and differential cryptanalysis. These techniques play a key role in assessing the robustness of block cipher designs. Linear cryptanalysis seeks to uncover approximate linear correlations between input bits, output bits, and key bits, which can then be exploited to infer parts of the secret key. In contrast, differential cryptanalysis examines how specific differences in input values affect the output, allowing attackers to detect high probability patterns that can aid in key recovery. Together, these methods serve as critical benchmarks for measuring the cryptographic strength of modern block ciphers.

### *5.1. Preservation of the Wide Trail Strategy in AES_DST*

By incorporating key-dependent dynamic byte permutation transformations (including ShiftRows and TranBytes) based on the bits of the key $kd$, we obtain the dynamic block cipher AES_DST without compromising the wide trail strategy used in the design of the original AES [2]. To clarify this point, we analyze the diffusion level of active bytes as they pass through two consecutive rounds in the AES_DST algorithm.

 

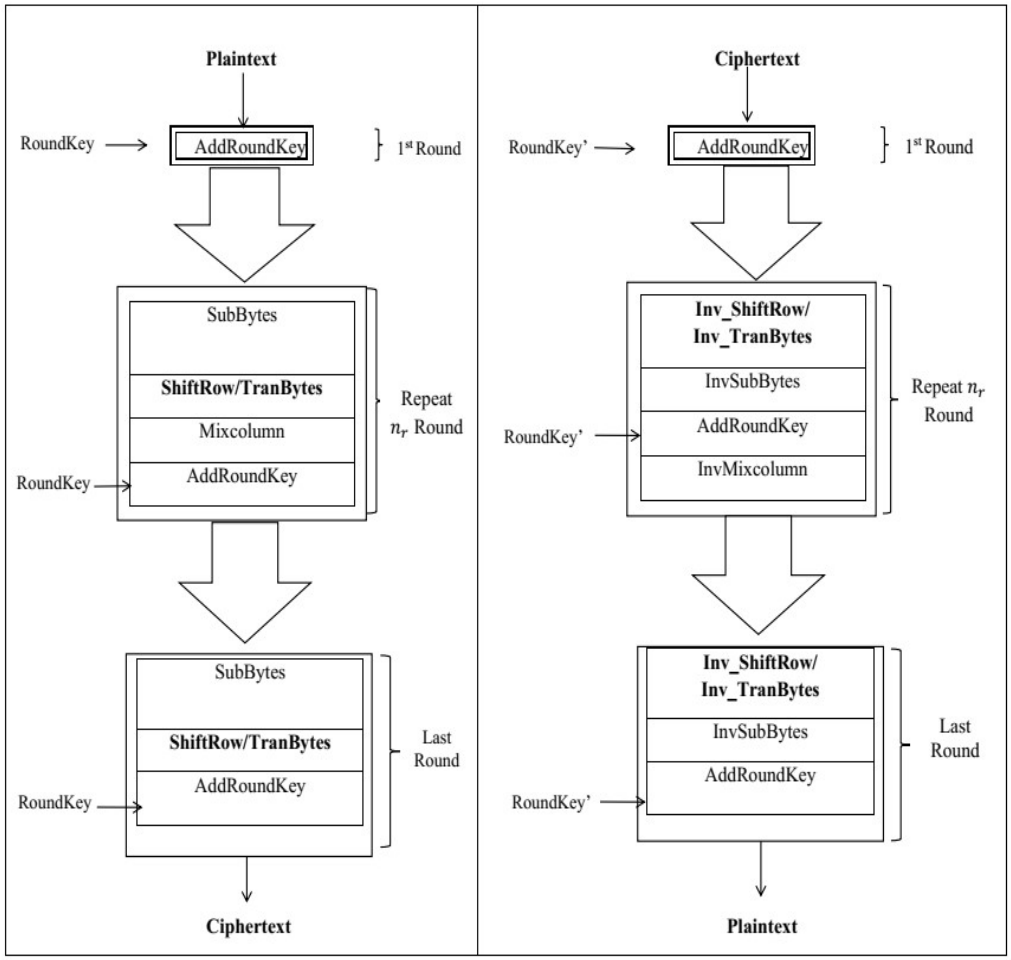


**Fig. 5.1.** Round Structure Diagram of the AES_DST Encryption/Decryption Process

Figures 5.1 and Figures 5.2 illustrate the role of the *ShiftRows* transformation in propagating active bytes.

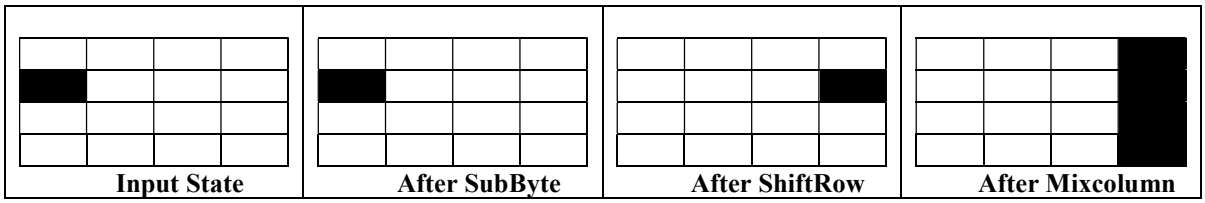


**Fig. 5.2.** Distribution of active bytes during the first round of the AES_DST dynamic block cipher using ShiftRows

Figure 5.1 demonstrates the diffusion behavior during the first round of AES, beginning with a state matrix that contains only one active byte. After the SubBytes and ShiftRows operations, the state still has a single active byte; however, the MixColumns transformation, thanks to the diffusion characteristics of the MDS matrix, expands this to four active bytes. This number remains unchanged after the AddRoundKey step. Therefore, a single active byte at the start of the round propagates to four active bytes by the end of the first round. Active bytes are highlighted as black squares.

 

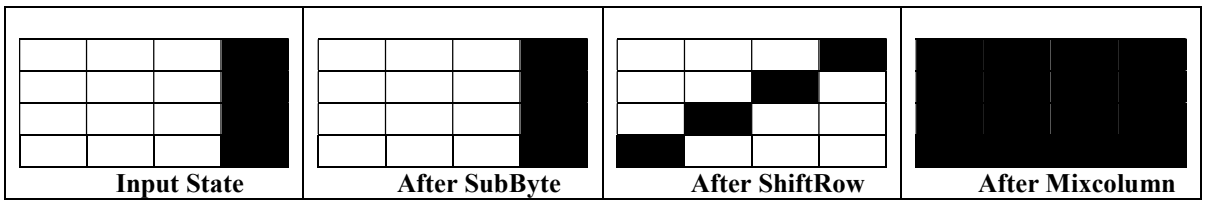


**Fig. 5.2.** Spread of active bytes during the second round of the AES_DST dynamic block cipher using ShiftRows

Figure 5.2 illustrates the key role of ShiftRows in distributing active bytes from one column across all four columns of the state matrix. Consequently, following the MixColumns operation, the active bytes spread to cover all 16 bytes in the state matrix.

Figure 5.3 and Figure 5.4 demonstrate how the *TranBytes* transformation contributes to the diffusion of active bytes.

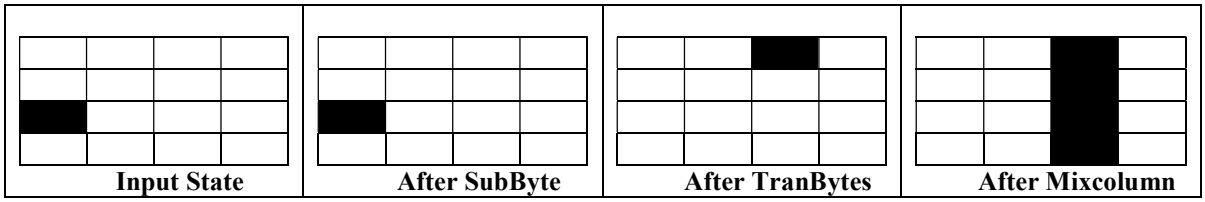


**Fig.5.3.** Spread of active bytes during the first round of the AES_DST dynamic block cipher using TranBytes

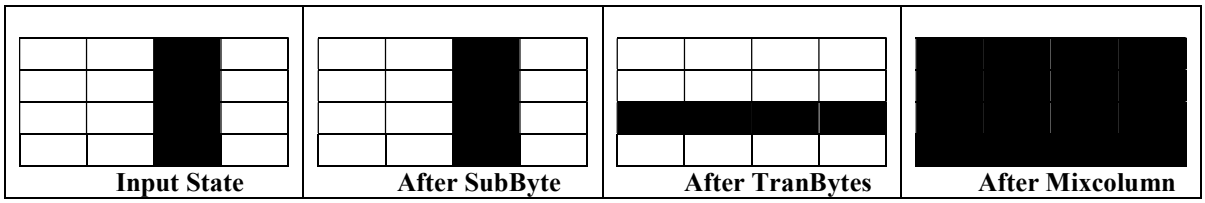


**Fig. 5.4.** Distribution of active bytes during the second round of the AES_DST dynamic block cipher using TranBytes

Similar to the ShiftRows transformation, thanks to the TranBytes transformation, starting from a single active byte, the number of active bytes expands to the maximum of 16 bytes across the entire state matrix after two rounds.

From the above illustrations, it is evident that both ShiftRows and TranBytes transformations enable the dynamic block cipher AES_DST to adhere to the wide trail strategy used in the design of the original AES.

According to the philosophy of dynamic block ciphers proposed by Knudsen in 2015 [7], our dynamic AES version not only inherits the security properties of the original AES but, in some respects, also increases the complexity of cryptanalysis. In differential cryptanalysis and its variants, when differential trails depend on the key, the complexity of attacks rises. This is because the attacker must consider all possible cases—on the order of $2^{n_r}$, where $n_r$ is the number of encryption rounds—to determine which differential path the data follows. Furthermore, the security of the dynamic block cipher is also supported by Vincent Rijmen [8], one of the two principal designers of the AES block cipher.

We will provide a more detailed analysis of this in the following sections.

### *5.2. Analysis of Linear Cryptanalysis on the Dynamic Block Cipher AES_DST*

To perform linear cryptanalysis on the block cipher, one needs to build a linear approximation covering $n_r - 1$ rounds, along with a corresponding linear expression for the full $n_r$-round cipher. The procedure involves the following steps:

***Develop a linear approximation spanning $\mathbf{n_r - 1}$ rounds along with an associated linear equation for the $\mathbf{n_r}$-round block cipher.***

**Step 1:** Generate the linear approximation matrix for the S-box.

 

**Step 2:** Determine which S-boxes are active (with non-zero input and output masks) in the rounds and assign suitable linear approximations from the matrix created in Step 1 to these S-boxes.

**Step 3:** Develop the overall linear equation for the block cipher and assess its deviation from uniform probability.

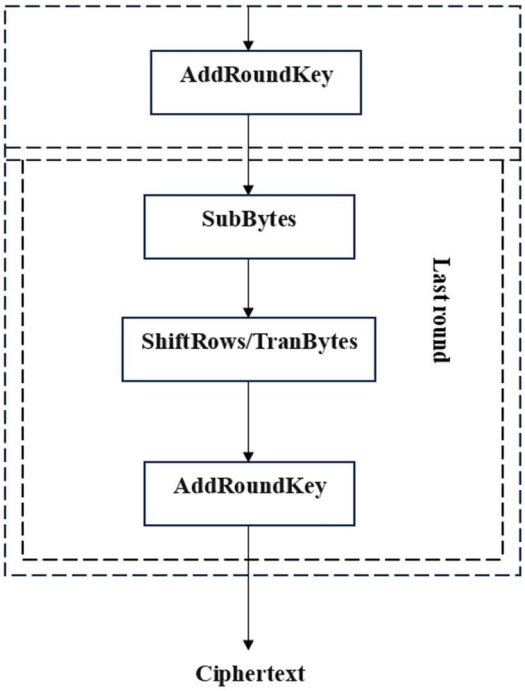


**Fig. 5.5.** The final round of the AES_DST dynamic block cipher

AES_DST, the dynamic block cipher, performs $n_r$ rounds on a 128-bit input block and employs $8 \times 8$ substitution boxes. Let $K$ represent the secret key for AES_DST, from which the round keys $K_1, K_2, \ldots, K_{n_r}, K_{n_r+1}$ are generated.

Denote by $P$ the plaintext and by $C$ the ciphertext produced by the AES_DST dynamic block cipher.

For the $i$-th round, let $A_i$ and $B_i$ be the inputs and outputs of the S-boxes, respectively. The bit $A_{i,j}$ refers to the $j$-th bit of $A_i$, where $i$ ranges from 1 to $n_r$ and $j$ ranges from 1 to 128.

In AES_DST, the permutation is represented by $D_T$.

It is important to note that the linear approximation of AES depends solely on the plaintext, the input bits to the last round ($n_r$-th round), and the bits of the subkeys $K_1, K_2, \ldots, K_{n_r}$.

In AES_DST, the permutation $D_T$—encompassing both the ShiftRows/TranBytes and MixColumns steps—is transformed from a fixed operation into a dynamic one. Specifically, unlike the original static ShiftRows, this version employs a key-dependent dynamic ShiftRows/TranBytes.

The ShiftRow/TranBytes operation works on a $4 \times 4$ byte state array. Let $S$ denote the input state array for ShiftRow/TranBytes, with its bytes numbered from 1 to 16 as follows:

$$S = \begin{pmatrix} s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \\ s_4 & s_8 & s_{12} & s_{16} \end{pmatrix}$$

The ShiftRow transformation works by keeping the first row of the state array intact, while the second row is shifted left by one byte, the third row by two bytes, and the fourth row by three bytes.

 

Hence, ShiftRow can be described as a specific permutation of bytes as follows:

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

The inverse of the ShiftRow transformation, denoted as $ShiftRow^{-1}$, can likewise be expressed through a byte permutation as shown below:

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

TranBytes functions by rearranging the state matrix, and as such, it can be represented as a byte-level permutation as follows:

| $S$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $TranBytes(S)$ | 1 | 5 | 9 | 13 | 2 | 6 | 10 | 14 | 3 | 7 | 11 | 15 | 4 | 8 | 12 | 16 |

The inverse TranBytes operation, $TranBytes^{-1}$, can similarly be expressed as a byte permutation as follows:

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

In the AES_DST dynamic block cipher, a major obstacle for attackers lies in forming a linear approximation over $n_r - 1$ rounds and deriving a linear expression for the full cipher. While the initial step of cryptanalysis is carried out normally, difficulties arise in the next step when choosing active S-boxes in each round, due to the unknown nature of the permutation. This uncertainty complicates correlating S-boxes between rounds, forcing attackers to select them randomly. Even if by chance an attacker succeeds in constructing a linear approximation spanning $n_r - 1$ rounds and a corresponding expression for the entire AES_DST, the process remains highly challenging.

Assume that this linear expression takes the following form:

$$A_{n_r,i} \oplus A_{n_r,j} \oplus \ldots \oplus A_{n_r,t} \oplus ((P_l \oplus P_h \oplus \ldots \oplus P_k) \oplus \big(K_{1,g} \oplus \ldots K_{1,e} \oplus K_{2,f} \oplus \ldots . \oplus K_{3,u} \oplus \ldots \oplus K_{n_r,v}\big)) = 0 \quad (2)$$

Referring to the last round (see Fig. 9), let's assume the attacker can extract certain bits of the subkey $K_{n_r+1}$, which correspond to the active S-boxes identified in round $n_r$. Additionally, the attacker can derive some essential bits of $A_{n_r}$.

$$A_{n_r} = D_T\big(B_{n_r-1}\big) \oplus K_{n_r} \quad (3)$$

Using equation (3) as a starting point, the attacker continues by making guesses on the associated key bits of $K_{n_r}$, subsequently trying to derive the required bits of $B_{n_r-1}$ following equation (4).

$$B_{n_r-1} = D_T^{-1}\big(A_{n_r} \oplus K_{n_r}\big) \quad (4)$$

In dynamic AES block ciphers, the diffusion-layer permutation $D_T$ varies depending on the key. This means the attacker cannot identify the exact $D_T$ used in the cipher, preventing them from calculating the required bits of $B_{n_r-1}$ based on equation (4). As a result, the attacker is unable to extract the corresponding key bits of $K_{n_r}$, and faces similar obstacles with the other subkeys as well.

Therefore, under these circumstances, the attacker is limited to obtaining some bits of the subkey $K_{n_r+1}$ and is unable to deduce any further key bits from the remaining subkeys.

To perform standard linear cryptanalysis, the attacker needs to determine the specific permutation $D_T$ applied within the dynamic AES block cipher before they can continue with the attack.

Initially, it is assumed that during the process of forming a linear approximation for the dynamic AES block cipher, the attacker encounters more difficulties in Step 2 than with the original AES. Because the permutation used in AES_DST varies dynamically and is unknown, the attacker is forced to pick S-boxes randomly across rounds. These selections may lack correlation, which hampers the construction of $n_r - 1$-round linear approximations and the overall linear expression for the dynamic AES_DST cipher.

 

By introducing key-dependent ShiftRow/TranBytes operations that make the permutation layer dynamic in the AES_DST block cipher, the difficulty of carrying out linear cryptanalysis rises considerably compared to the conventional static AES cipher.

### *5.3. Analysis of Differential Cryptanalysis on the Dynamic Block Cipher AES_DST*

To conduct differential cryptanalysis on the block cipher, it is necessary to develop a differential characteristic spanning $n_r - 1$ rounds, as well as the associated differential for the full $n_r$-round cipher. The process for building these is outlined below:

***Develop a differential characteristic covering $\mathbf{n_r - 1}$ rounds along with the matching differential applicable to the entire $\mathbf{n_r}$-round block cipher.***

**Step 1*:** Create the differential distribution table for the S-box.

**Step 2*:** Identify specific active S-boxes in each round (i.e., those exhibiting non-zero differentials) and assign corresponding differentials from the table developed in Step 1*.

**Step 3*:** Calculate the overall differential characteristic of the block cipher and evaluate its probability.

It is important to highlight that the block cipher's differential is linked to a differential characteristic spanning $n_r - 1$ rounds. This differential is denoted by $(\Delta A, \Delta B)$, with $\Delta A$ representing the input difference at the plaintext stage, and $\Delta B$ indicating the input difference at the final round. The overall differential characteristic encompasses the individual differentials of the chosen S-boxes across the rounds.

AES_DST, the dynamic block cipher, performs $n_r$ rounds on a 128-bit input block and employs $8 \times 8$ substitution boxes. Let $K$ represent the secret key for AES_DST, from which the round keys $K_1, K_2, \dots, K_{n_r}, K_{n_r+1}$ are generated.

Denote by $P$ the plaintext and by $C$ the ciphertext produced by the AES_DST dynamic block cipher.

For the $i$-th round, let $A_i$ and $B_i$ be the inputs and outputs of the S-boxes, respectively. The bit $A_{i,j}$ refers to the $j$-th bit of $A_i$, where $i$ ranges from 1 to $n_r$ and $j$ ranges from 1 to 128.

Let $\Delta A_i = A_i \oplus A_{i'}$ and $\Delta B_i = B_i \oplus B_{i'}$ represent the input and output differences of the S-boxes at round $i$. Here, $\Delta A_{i,j}$ denotes the $j$-th bit of $\Delta A_i$, with $1 \le i \le n_r$ and $1 \le j \le 128$.

The permutation operation in AES_DST is still represented by $D_T$.

For the AES_DST dynamic cipher, a significant obstacle encountered by attackers lies in forming an $n_r - 1$-round differential characteristic along with the matching differential for the complete cipher. Although Step 1* proceeds as usual, Step 2*, which involves choosing active S-boxes in various rounds, proves problematic since the permutation is unknown. This uncertainty complicates the task of correlating S-boxes across rounds, forcing the attacker to select these S-boxes randomly. Assuming the attacker is fortunate enough, they might still construct a valid differential characteristic for the entire AES_DST cipher.

Assuming the attacker manages to recover specific bits of the subkey $K_{n_r+1}$, particularly those associated with the selected active substitution boxes in the final round, they may also be capable of deriving some necessary bits from both inputs $A_{n_r}$ and $A'_{n_r}$ of that round.

Using equation (3), the cryptanalyst can infer $B_{n_r-1}$ based on equation (4) ($B_{n_r-1} = D_T^{-1}(A_{n_r} \oplus K_{n_r})$).

Following this, the attacker attempts to estimate the key bits of the subkey $K_{n_r}$ that influence the output of the round. With these guesses, they reconstruct $B_{n_r-1}$, and do the same to get $B'_{n_r-1}$. The bitwise difference is then calculated as:

$$\Delta B_{n_r-1} = B_{n_r-1} \oplus B_{n_r-1'}$$

Then, if the derived bits of $\Delta B_{n_r-1}$ match those in the differential characteristic previously constructed, the guessed key bits are likely correct for $K_{n_r}$.

 

In contrast to static designs, the AES_DST cipher employs a key-dependent, dynamically changing permutation $D_T$. This uncertainty prevents the attacker from identifying the exact permutation applied during encryption. Without knowing $D_T$, they are unable to accurately compute the critical bits of $B_{n_r-1}$, making it impossible to verify whether the guessed subkey bits for $K_{n_r}$ — or any other round keys — are valid.

In this case, the attacker can only retrieve a limited number of key bits from the subkey $K_{n_r+1}$, with no feasible way to uncover additional key bits from the remaining round keys.

Moreover, as mentioned earlier, developing a differential characteristic for AES_DST proves more challenging than for the standard AES, particularly in Step 2*. The core issue stems from the fact that the internal permutation used in AES_DST varies with the key and is not known to the attacker. This forces the attacker to select active S-boxes at random across rounds, reducing the likelihood that those S-boxes are meaningfully connected. As a result, constructing a coherent $n_r - 1$-round differential characteristic and a complete differential trail becomes nearly impossible in the context of AES_DST.

To apply traditional differential cryptanalysis to the AES_DST cipher, an attacker must first identify the specific permutation $D_T$ employed within the cipher. Without this information, the attack cannot effectively proceed.

By introducing dynamic, key-dependent transformations in the diffusion stage—specifically through the ShiftRow and TranBytes operations—the AES_DST cipher substantially raises the difficulty level for differential attacks. This added complexity marks a significant departure from the more predictable structure of standard, static AES, thereby enhancing resistance against such cryptanalytic techniques.

## 6. EVALUATION OF THE RANDOMNESS OF AES_DST

This section presents a randomness analysis of the AES_DST dynamic block cipher, based on statistical tests recommended by the NIST framework [35].

Our evaluation integrates the randomness testing strategy outlined in NIST SP 800-22 [35] with the structured plaintext generation approach introduced by Sulak [36]. This allows us to investigate how AES_DST behaves under controlled, non-random plaintext conditions across different encryption rounds. Specifically, we construct several deterministic input sets—such as Low Weight (LW), High Weight (HW), 1-bit difference (AV1), and rotated plaintexts (Rot)—as test vectors. We then apply the two-tiered testing strategy from NIST SP 800-22 to assess the pseudorandomness of the resulting ciphertexts. Notably, the evaluation is limited to test cases optimized for short data sequences, by the guidance from [37].

Based on the analysis (Table 6.1 and Table 6.2), it is evident that employing random keys necessitates at least three rounds of encryption in the dynamic block cipher AES_DST to generate output data exhibiting adequate randomness for the HW, AV1, and LW datasets. This implies that for the AES_DST system to maintain essential randomness in its encrypted output, a minimum of three rounds must be executed when random keys are used. Additionally, the findings reveal that the randomness level produced by AES_DST matches that of the conventional AES cipher.

**Table 6.1.** Outcomes of second-level p-value assessments for AES_DST based on statistical tests applied to short data sequences

| No. of rounds | Freq. Test | Runs Test | Test for longest run of ones | Serial Test | AppEn. Test | CuSum. Test | Bit AutoCorr. Test | Byte Autocor. Test |
|---|---|---|---|---|---|---|---|---|
| **AV1 Input Data** | | | | | | | | |
| 1 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 2 | 0.004821 | 0.000000 | 0.000000 | 0.000001 | 0.000000 | 0.000001 | 0.000000 | 0.023187 |
| **3** | 0.388140 | 0.690279 | 0.296774 | 0.360755 | 0.113957 | 0.492432 | 0.560673 | 0.009957 |
| 4 | 0.925600 | 0.526663 | 0.773360 | 0.996072 | 0.965937 | 0.265690 | 0.917013 | 0.909591 |
| 5 | 0.969725 | 0.359059 | 0.562529 | 0.691677 | 0.590030 | 0.337997 | 0.255215 | 0.770595 |
| 6 | 0.001659 | 0.357359 | 0.867525 | 0.277019 | 0.118567 | 0.250200 | 0.739263 | 0.051301 |

 

| No. of rounds | Freq. Test | Runs Test | Test for longest run of ones | Serial Test | AppEn. Test | CuSum. Test | Bit AutoCorr. Test | Byte Autocor. Test |
|---|---|---|---|---|---|---|---|---|
| 7 | 0.715970 | 0.965837 | 0.099518 | 0.856579 | 0.752517 | 0.815556 | 0.335218 | 0.858150 |
| 8 | 0.915977 | 0.620556 | 0.255037 | 0.102592 | 0.255590 | 0.781520 | 0.087702 | 0.853995 |
| 9 | 0.677561 | 0.058897 | 0.926591 | 0.799319 | 0.733049 | 0.206479 | 0.023996 | 0.299263 |
| 10 | 0.190943 | 0.907337 | 0.907299 | 0.506631 | 0.375749 | 0.641292 | 0.409264 | 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 | 0.790742 | 0.956944 | 0.507715 | 0.767315 | 0.557299 | 0.063931 | 0.932396 |
| 4 | 0.971935 | 0.713995 | 0.569726 | 0.130933 | 0.555359 | 0.712793 | 0.715195 | 0.717960 |
| 5 | 0.550656 | 0.016966 | 0.930257 | 0.000669 | 0.000229 | 0.193905 | 0.009111 | 0.157356 |
| 6 | 0.712010 | 0.125157 | 0.269899 | 0.169880 | 0.103291 | 0.611226 | 0.226911 | 0.551235 |
| 7 | 0.738873 | 0.075857 | 0.790729 | 0.239091 | 0.108675 | 0.953965 | 0.978852 | 0.955501 |
| 8 | 0.785150 | 0.693697 | 0.305570 | 0.595033 | 0.762953 | 0.777388 | 0.288855 | 0.572716 |
| 9 | 0.597013 | 0.525833 | 0.287850 | 0.365215 | 0.378768 | 0.837909 | 0.252725 | 0.007599 |
| 10 | 0.162613 | 0.932569 | 0.722379 | 0.636367 | 0.716321 | 0.316012 | 0.670659 | 0.005735 |
| **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** | 0.815197 | 0.936996 | 0.896293 | 0.907959 | 0.950627 | 0.915199 | 0.591385 | 0.256157 |
| 4 | 0.059672 | 0.760510 | 0.263700 | 0.597756 | 0.657905 | 0.999708 | 0.352198 | 0.628523 |
| 5 | 0.721655 | 0.928550 | 0.502976 | 0.997965 | 0.995961 | 0.530596 | 0.275560 | 0.769510 |
| 6 | 0.360959 | 0.275531 | 0.770137 | 0.099165 | 0.095005 | 0.999728 | 0.389153 | 0.370810 |
| 7 | 0.821301 | 0.850561 | 0.223176 | 0.729395 | 0.728277 | 0.529035 | 0.703023 | 0.110923 |
| 8 | 0.906001 | 0.922559 | 0.926138 | 0.963057 | 0.931601 | 0.875985 | 0.571562 | 0.035327 |
| 9 | 0.959171 | 0.162170 | 0.737297 | 0.155859 | 0.356525 | 0.599980 | 0.323053 | 0.565623 |
| 10 | 0.060375 | 0.297092 | 0.636063 | 0.621519 | 0.537975 | 0.505325 | 0.657715 | 0.266918 |
| **Rot Input Data** | | | | | | | | |
| 1 | 0.357559 | 0.526258 | 0.200096 | 0.750682 | 0.925820 | 0.029858 | 0.526090 | 0.565327 |
| **2** | 0.595618 | 0.667512 | 0.615079 | 0.635559 | 0.350689 | 0.535297 | 0.951519 | 0.115702 |
| 3 | 0.779571 | 0.528596 | 0.569855 | 0.506580 | 0.112316 | 0.089115 | 0.093269 | 0.369525 |
| 4 | 0.690950 | 0.144490 | 0.229079 | 0.297999 | 0.236025 | 0.194579 | 0.215946 | 0.203690 |
| 5 | 0.199359 | 0.901740 | 0.279104 | 0.979264 | 0.975541 | 0.951572 | 0.769115 | 0.743401 |
| 6 | 0.355609 | 0.419073 | 0.377150 | 0.647660 | 0.523603 | 0.963475 | 0.929479 | 0.315552 |
| 7 | 0.057450 | 0.965519 | 0.491526 | 0.749116 | 0.957566 | 0.229471 | 0.961226 | 0.410902 |
| 9 | 0.298895 | 0.059195 | 0.771945 | 0.507281 | 0.758229 | 0.148971 | 0.509822 | 0.511561 |
| 9 | 0.751565 | 0.815415 | 0.560448 | 0.877156 | 0.869469 | 0.405955 | 0.789552 | 0.552502 |
| 10 | 0.207595 | 0.158116 | 0.659577 | 0.752650 | 0.480489 | 0.850594 | 0.124907 | 0.122742 |

**Table 6.2.** Outcomes of proportion tests for AES DST conducted on short sequence datasets

| 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 | 98.95 | 99.09 | 99.29 | 99.16 |
| 2 | 99.92 | 99.95 | 99.07 | 98.95 | 98.91 | 98.98 | 99.24 | 99.22 |
| 3 | 99.00 | 99.96 | 99.09 | 99.04 | 99.95 | 99.09 | 99.25 | 99.22 |
| 4 | 99.99 | 99.96 | 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 | 99.24 |
| 6 | 99.99 | 99.94 | 99.09 | 99.01 | 98.94 | 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 | 99.95 | 99.08 | 99.01 | 98.94 | 99.06 | 99.25 | 99.22 |
| 9 | 98.99 | 98.95 | 99.09 | 99.04 | 98.95 | 99.07 | 99.25 | 99.22 |
| 10 | 98.99 | 98.95 | 99.08 | 99.00 | 98.94 | 99.07 | 99.25 | 99.21 |
| **HW Input Data** | | | | | | | | |
| 1 | 98.74 | 98.96 | 99.24 | 98.65 | 99.67 | 98.82 | 99.01 | 99.44 |
| 2 | 98.91 | 99.11 | 99.29 | 99.01 | 99.05 | 99.04 | 99.41 | 99.20 |
| 3 | 98.99 | 99.94 | 99.06 | 99.00 | 99.94 | 99.07 | 99.24 | 99.22 |
| 4 | 98.98 | 98.95 | 99.10 | 99.02 | 98.94 | 99.06 | 99.25 | 99.20 |
| 5 | 99.00 | 98.94 | 99.10 | 99.01 | 98.95 | 99.08 | 99.26 | 99.21 |
| 6 | 98.96 | 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 | 99.04 | 98.96 | 99.06 | 99.25 | 99.21 |
| 9 | 99.00 | 98.96 | 99.08 | 99.02 | 98.95 | 99.06 | 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 | 99.40 | 99.12 | 99.04 | 99.15 | 99.46 | 99.46 |
| 2 | 99.14 | 99.04 | 99.18 | 99.11 | 99.12 | 99.06 | 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 | 98.94 | 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 | 98.96 | 99.06 | 99.26 | 99.22 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 7 | 98.99 | 98.95 | 99.08 | 99.01 | 98.94 | 99.06 | 99.25 | 99.22 |
| 8 | 98.99 | 98.96 | 99.09 | 99.04 | 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 | 99.04 | 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 | 99.04 | 98.96 | 99.08 | 99.26 | 99.24 |
| 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 | 98.94 | 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 | 98.94 | 99.06 | 99.24 | 99.24 |
| 10 | 98.99 | 98.94 | 99.07 | 99.02 | 98.94 | 99.06 | 99.25 | 99.24 |

## 7. ANALYSIS OF THE IMPLEMENTATION EFFICIENCY OF THE DYNAMIC BLOCK CIPHER AES_DST ON A SOFTWARE PLATFORM

Unlike previously published versions of dynamic AES block ciphers, our dynamic AES block cipher maintains implementation efficiency on 32-bit software platforms. Before analyzing the approach to handling the dynamic byte permutation layer, we briefly review the lookup table implementation technique of the original AES [2].

Let $e$ denote the data state after the *MixColumns* transformation, $a$ be the input data state to the *SubBytes* transformation, $b$ be the input data state to the *ShiftRows* transformation, and $c$ be the input data state to the *MixColumns* transformation. Then, we have:

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \otimes \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix},$$

with

$$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j} \\ b_{1,j-1} \\ b_{2,j-2} \\ b_{3,j-3} \end{bmatrix} = \begin{bmatrix} S[a_{0,j}] \\ S[a_{1,j-1}] \\ S[a_{2,j-2}] \\ S[a_{3,j-3}] \end{bmatrix}, \tag{5}$$

where $S$ is the S-box of the original AES, the operation " $-$ " is defined modulo 4, and $j = 0,1,2,3$. Thus, we have

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = S[a_{0,j}] \otimes \begin{bmatrix} 2 \\ 1 \\ 1 \\ 3 \end{bmatrix} \oplus S[a_{1,j-1}] \otimes \begin{bmatrix} 3 \\ 2 \\ 1 \\ 1 \end{bmatrix} \oplus S[a_{2,j-2}] \otimes \begin{bmatrix} 1 \\ 3 \\ 2 \\ 1 \end{bmatrix} \oplus S[a_{3,j-3}] \otimes \begin{bmatrix} 1 \\ 1 \\ 3 \\ 2 \end{bmatrix} \tag{6}$$

Then, if we compute the following four tables:

$$T_0[a] = \begin{bmatrix} S[a] \otimes 2 \\ S[a] \\ S[a] \\ S[a] \otimes 3 \end{bmatrix}, T_1[a] = \begin{bmatrix} S[a] \otimes 3 \\ S[a] \otimes 2 \\ S[a] \\ S[a] \end{bmatrix},$$

$$T_2[a] = \begin{bmatrix} S[a] \\ S[a] \otimes 3 \\ S[a] \otimes 2 \\ S[a] \end{bmatrix}, T_3[a] = \begin{bmatrix} S[a] \\ S[a] \\ S[a] \otimes 3 \\ S[a] \otimes 2 \end{bmatrix},$$

where $a \in \mathbb{F}_{2^8}$, and the operation " $\otimes$ " represents multiplication in $\mathbb{F}_{2^8}$. Each table consists of 256 entries, each of which is 32 bits long. Thus, the $j$-th column of the state $e$ is calculated as follows:

$$e_j = T_0[a_{0,j}] \oplus T_1[a_{1,j-1}] \oplus T_2[a_{2,j-2}] \oplus T_3[a_{3,j-3}] \tag{7}$$

 

Now, if we replace *ShiftRows* with *TranBytes*, then equation (5) becomes:

$$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{j,0} \\ b_{j,1} \\ b_{j,2} \\ b_{j,3} \end{bmatrix} = \begin{bmatrix} S[a_{j,0}] \\ S[a_{j,1}] \\ S[a_{j,2}] \\ S[a_{j,3}] \end{bmatrix}, \tag{8}$$

Equation (6) becomes:

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = S[a_{j,0}] \otimes \begin{bmatrix} 2 \\ 1 \\ 1 \\ 3 \end{bmatrix} \oplus S[a_{j,1}] \otimes \begin{bmatrix} 3 \\ 2 \\ 1 \\ 1 \end{bmatrix} \oplus S[a_{j,2}] \otimes \begin{bmatrix} 1 \\ 3 \\ 2 \\ 1 \end{bmatrix} \oplus S[a_{j,3}] \otimes \begin{bmatrix} 1 \\ 1 \\ 3 \\ 2 \end{bmatrix}, \tag{9}$$

And equation (7) becomes:

$$e_j = T_0[a_{j,0}] \oplus T_1[a_{j,1}] \oplus T_2[a_{j,2}] \oplus T_3[a_{j,3}] \tag{10}$$

From (7) and (10), we see that it is sufficient to use only 4 tables $T_j, j = 0,1,2,3$ to compute the output after the MixColumns transformation. To eliminate the "if-else" branching in the encryption procedure described in the dynamic AES_DST block cipher algorithm in section 4.1, we calculate the $j$-th column in state $e$ at round $r$ based on the dynamic key bit $kd_r$ as follows:

$$\begin{aligned} e_j = T_0[kd_r \cdot a_{0,j} \oplus \overline{kd_r} \cdot a_{j,0}] \oplus T_1[kd_r \cdot a_{1,j-1} \oplus \overline{kd_r} \cdot a_{j,1}] \oplus \\ T_2[kd_r \cdot a_{2,j-2} \oplus \overline{kd_r} \cdot a_{j,2}] \oplus T_3[kd_r \cdot a_{3,j-3} \oplus \overline{kd_r} \cdot a_{j,3}], \end{aligned} \tag{11}$$

where the symbol " $\cdot$ " denotes multiplication in the decimal system, and $\overline{kd_r}$ represents the negation of the bit $kd_r$.

In this way, we have eliminated branching in the encryption procedure. Moreover, the implementation using precomputed lookup tables can be fully applied to our proposed dynamic AES_DST block cipher. The only difference lies in how the addresses are obtained to access the tables $T_j, j = 0,1,2,3$.

To evaluate the execution speed, we implemented the proposed dynamic AES_DST block cipher in C++ using the precomputed lookup table method. Below are some statistical results on execution performance along with comparisons to other commonly used block ciphers (compiled and run on the same platform). All tests were conducted on a single-core machine equipped with an Intel® Core™ i5-7200U CPU @ 2.50GHz (2.471GHz effective), 12GB RAM, running Windows 10 64-bit, using Visual Studio 2019 compiler in Release mode (x64). The block cipher implementations did not use any assembly instructions and operated in ECB mode. The performance evaluation and speed comparison results are presented in Table 7.1.

**Table 7.1.** Performance evaluation and comparison of the dynamic AES_DST block cipher execution speed

| No. | Block Cipher | Block/Key Size | Number of Rounds | Lookup Table Size (Encryption + Decryption) KBytes | Encryption/ Decryption Speed (Mb/s) | Implementation Source |
|---|---|---|---|---|---|---|
| 1 | Kalyna | 128/128 | 10 | 128 | 1598 | Oliynykov† |
| | | 128/256 | 14 | 128 | 1186 | |
| 2 | Kuznyechik | 128/256 | 10 | 128 | 640 | |
| 3 | AES | 128/128 | 10 | 8 | 1696 | Gladman‡ (on a 32-bit platform) |
| | | 128/192 | 12 | 8 | 1510 | |
| | | 128/256 | 14 | 8 | 1302 | |
| **4** | **AES_DST** | **128/128** | **10** | **8** | **1402** | **Ours** |
| | | **128/192** | **12** | **8** | **1210** | |

† https://github.com/Roman-Oliynykov/ciphers-speed

‡ http://brg.a2hosted.com//oldsite/cryptography_technology/rijndael/index.php

 

| | | | | | | |
|---|---|---|---|---|---|---|
| | | **128/256** | **14** | **8** | **1087** | |

From Table 7.1, it can be observed that with our dynamic approach, the encryption speed of the AES_DST dynamic block cipher decreases only slightly compared to the original AES version (about a 20% reduction). Compared to several other block ciphers, the dynamic AES version still maintains a comparable encryption speed. This means that the dynamic AES version we propose can meet the demands of building high-speed cryptographic applications while also enhancing security compared to the original AES block cipher.

## 8. CONCLUSION

The paper provides a different perspective on the security of byte-oriented AES-style block ciphers. To do this, we propose a general SPN cipher model. Then, we prove using number theory the minimal number of active S-boxes after 4 rounds of encryption in this cipher. Our method is generalized and can be applied to evaluate AES-like ciphers such as Kalyna, LED, Kuznyechik, and block ciphers used in hash functions like GOST R 34.11-2012, Whirlpool, and others. From this analysis, we observe that the role of the byte permutation layer—such as ShiftRows in AES—is especially critical. This layer must ensure the $m$-diffusion property to preserve the block cipher's design based on the wide trail strategy. Based on these findings, we introduce a key-dependent dynamic variant of the AES block cipher (AES_DST). Unlike earlier methods, our approach leverages table lookup implementations without the need for pre-calculated lookup tables. Practical tests demonstrate that the dynamic AES cipher's performance is comparable to well-established block ciphers globally. This dynamic version offers enhanced security over the standard AES while maintaining efficient software implementation. The insights gained here are valuable for the development of secure and adaptable block ciphers, as well as for providing a theoretical basis to assess cryptographic components in current encryption systems. Moving forward, we plan to analyze the dynamic AES cipher's resilience against different types of attacks and estimate its quantum security, particularly considering attacks using Grover's algorithm.

ACKNOWLEDGEMENTS

**Funding**: No funding was received for conducting this study.
**Financial interests:** We declare that we have no financial interests.
**Data availability:** Data sharing does not apply to this article as no datasets were used, generated, or analyzed during this study
**Declarations Competing interests :** We declare that we have no competing interests.

REFERENCES

[1] Shannon, C.E. (1949) Communication theory of secrecy systems, *The Bell System Technical Journal*, **28**(4), 656–715.

[2] Daemen, J., Rijmen, V. (2002) *The design of Rijndael, Vol. 2*. New York, NY: Springer-Verlag.

[3] Guo, J., Peyrin, T., Poschmann, A. & Robshaw, M. (2011) The LED block cipher, *Proc. of 13th International Workshop* (Nara, Japan), pp. 326–341.

[4] Oliynykov, R., Gorbenko, I., Kazymyrov, O., Ruzhentsev, V., Kuznetsov, O., (2015) A new encryption standard of Ukraine: The Kalyna block cipher, *Cryptology ePrint Archive*, [Online]. Available: https://eprint.iacr.org/2015/650

[5] Dolmatov, V. (2016) GOST R 34.12-2015: Block Cipher "Kuznyechik", [Online]. Available: https://www.rfc-editor.org/rfc/rfc7801

 

[6] Phuong, T.M. & Luong, T.T. (2023) Evaluating the number of active S-boxes in dynamic AES block ciphers using MDS matrices of size 4×4 and 8×8, *TNU Journal of Science and Technology*, **228**(15), 190–199, doi: 10.34238/tnu-jst.9053

[7] Knudsen, L.R. (2014) Dynamic encryption, *Journal of Cyber Security and Mobility*, **3**, 357–370.

[8] Rijmen, V. (2017) *Opinion on dynamic encryption*, [Online]. Available: https://www.dencrypt.dk/wp-content/uploads/2017/05/Dencrypt-Vincent-Rijmen-opinion-on-Dynamic-Encryption.pdf (2017)

[9] Al-Dweik, A.Y., Hussain, I., Saleh, M. & Mustafa, M.T. (2022) A novel method to generate key-dependent S-boxes with identical algebraic properties, *Journal of Information Security and Applications*, **64**, 103065.

[10] Kazlauskas, K. & Kazlauskas, J. (2009) Key-dependent S-box generation in AES block cipher system, *Informatica*, **20**(1), 23–34.

[11] Agarwal, P., Singh, A. & Kilicman, A. (2018). Development of key-dependent dynamic S-boxes with dynamic irreducible polynomial and affine constant, *Advances in Mechanical Engineering*, **10**(7), doi: 10.1177/1687814018781638.

[12] Abdulrazaq, N.N. (2024). Generating of a dynamic and secure S-box for AES block cipher system based on modified hexadecimal Playfair cipher, *Zanco Journal of Pure and Applied Sciences*, **36**(5), 82–94.

[13] Mahmoud, E.M., Hafez, A.A., Elgarf, T.A. & Zekry, A.H. (2013) Dynamic AES-128 with key-dependent S-box, *International Journal of Engineering Research and Applications*, **3**(1), 1662–1670.

[14] Assafli, H.T. & Hashim, I.A. (2020) Generation and evaluation of a new time-dependent dynamic S-box algorithm for AES block cipher cryptosystems, *IOP Conference Series: Materials Science and Engineering*, **978**(1), 012042.

[15] Murtaza, G., Khan, A.A., Alam, S.W. & Farooqi, A. (2011) Fortification of AES with dynamic mix-column transformation, *Cryptology ePrint Archive* [Online]. Available: https://eprint.iacr.org/2011/184

[16] Luong, T.T. (2022) Building the dynamic diffusion layer for SPN block ciphers based on direct exponent and scalar multiplication, *Journal of Science and Technology on Information Security*, **1**(15), 38–45.

[17] Luong, T.T. (2023) A dynamic algorithm for the linear layer of SPN block ciphers based on self-reciprocal recursive MDS matrices, *Proc. of 2023 15th International Conference on Knowledge and Systems Engineering* (Hanoi, Vietnam), pp. 1–6.

[18] Shamsabad, M.R.M. & Dehnavi, S.M. (2020) Dynamic MDS diffusion layers with efficient software implementation, *International Journal of Applied Cryptography*, **4**(1), 36–44

[19] Xu, T., Liu, F. & Wu, C. (2018) A white-box AES-like implementation based on key-dependent substitution-linear transformations, *Multimedia Tools and Applications*, **77**, 18117–18137.

[20] Altigani, A., Hasan, S., Barry, B., Naserelden, S., Elsadig, M.A., et al. (2021). A polymorphic advanced encryption standard–a novel approach, *IEEE Access*, **9**, 20191–20207.

[21] Freyre, P., Cuellar, O., Díaz, N. & Alfonso, A. (2020) From AES to dynamic AES, *Journal of Science and Technology on Information Security*, **1**(11), 11–22.

[22] Manoj Kumar, T. & Karthigaikumar, P. (2020) A novel method of improvement in advanced encryption standard algorithm with dynamic shift rows, sub byte and mixcolumn

 

operations for the secure communication, *International Journal of Information Technology*, **12**(3), 825–830.
[23] Salih, A.I., Alabaichi, A. & Abbas, A.S. (2019) A novel approach for enhancing security of advance encryption standard using private XOR table and 3D chaotic regarding to software quality factor, *An International Journal of Research and Surveys*, **10**(9), 823–832.
[24] Salih, A.I., Alabaichi, A.M. & Tuama, A.Y. (2020) Enhancing advance encryption standard security based on dual dynamic XOR table and mixcolumns transformation, *Indonesian Journal of Electrical Engineering and Computer Science*, **19**(3), 1574–1581.
[25] Luong, T.T. (2023) Strengthening AES security through key-dependent ShiftRow and AddRoundKey transformations utilizing permutation, *International Journal of Advanced Computer Science & Applications*, **14**(11).
[26] Luong, T.T., Cuong, N.N. & Vo, B. (2024) AES security improvement by utilizing new key-dependent XOR tables, *IEEE Access*, **PP**(99), 1–1.
[27] Luong, T.T. (2025) Improving block cipher resilience with dynamically generated XOR matrices using the Fisher-Yates shuffle method, *Cryptogr. Commun.,* **17**(4), 1051–1074, doi: 10.1007/s12095-025-00802-w
[28] Adamu, M., Oyefolahan, O. I., Ojerinde, O. A. & Abdulmalik, M. D. (2024). Dynamic randomized advanced encryption standard (DR-AES): a solution to enhance the security of mobile learning system, *ATBU Journal of Science, Technology and Education*, **12**(2), 191–202.
[29] Kang, J.S., Hong, S., Lee, S., Yi, O., Park, C., et al. (2001) Practical and provable security against differential and linear cryptanalysis for substitution-permutation networks, *ETRI Journal*, **23**(4), 158–167.
[30] Lai, X., Massey, J.L. & Murphy, S. (1991) Markov ciphers and differential cryptanalysis, *Proc. of the Workshop on the Theory and Application of Cryptographic Techniques* (Brighton, UK), pp. 17–38.
[31] Dobraunig, C., Rotella, Y. & Schoone, J. (2020) Algebraic and higher-order differential cryptanalysis of Pyjamask-96, *IACR Transactions on Symmetric Cryptology*, **2020**(1), 289–312.
[32] Keliher, L. T. (2003) Linear cryptanalysis of substitution-permutation networks. *PhD thesis*, Queen's University.
[33] Matsui, M. (1993) Linear cryptanalysis method for DES cipher, *Proc. of Workshop on the Theory and Application of Cryptographic Techniques* (Lofthus, Norway), pp. 386–397.
[34] Jakobsen, T. & Knudsen, L. R. (2001) Attacks on block ciphers of low algebraic degree, *Journal of Cryptology*, **14**, 197–210.
[35] Bassham, L. E., Rukhin, A. L., Soto, J., Nechvatal, J. R., Smid, M. E., et al. (2010) A statistical test suite for random and pseudorandom number generators for cryptographic applications, *NIST Special Publication No. 800-22*.
[36] Sulak, F. (2011). Statistical analysis of block ciphers and hash functions. *PhD thesis*, Middle East Technical University.
[37] Sulak, F., Doğanaksoy, A., Ege, B. & Koçak, O. (2010). Evaluation of randomness test results for short sequences, *Proc. of Sequences and Their Applications–SETA 2010: 6th International Conference* (Paris, France), pp. 309–319.

 