Matrik: Jurnal Manajemen, Teknik Informatika, dan Rekayasa Komputer
Vol. 24, No. 3, July 2025, pp. 533∼546
ISSN: 2476-9843, accredited by Kemenristekdikti, Decree No: 10/C/C3/DT.05.00/2025
DOI: 10.30812/matrik.v24i3.5004

Square Transposition Method with Adaptive Key Flexibility and Strong Diffusion Performance

Magdalena Ariance Ineke Pakereng , Alz Danny Wowor , Yos Richard Beeh , Felix David , Erwien Christianto , Vincent Excelcio Susanto , Claudio Canavaro

Universitas Kristen Satya Wacana, Salatiga, Jawa Tengah, Indonesia

Article Info

Article history:

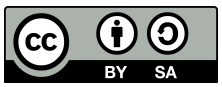
Received March 19, 2025
Revised June 23, 2025
Accepted July 10, 2025

Keywords:

Cryptography;
Data Security;
Diffusion Effect;
Key Flexibility;
Square Transposition.

ABSTRACT

The Square Transposition method has notable potential in enhancing diffusion within block encryption systems; however, its application is typically limited to perfect square key lengths. The objective of this study is to reconstruct the method to accommodate non-square key lengths by utilizing two square matrices. To assess the effectiveness of the proposed approach, the method of this study uses a comparative analysis conducted against the transposition structures found in DES and AES algorithms, both of which are cryptographic standards established by NIST. The comparison is strictly limited to the transposition component, excluding other components of the full encryption framework. The evaluation involves Monobit, Block Bit, and Run Tests, along with Pearson correlation analysis between plaintext and ciphertext. Tests are conducted on 16 input variations across three key sizes: 128-bit, 256-bit, and 512-bit. The results of this study show that the proposed method achieves lower correlation values ($r \approx 0.02$) compared to DES ($r \approx 0.07$) and AES ($r \approx 0.05$). The conclusion of this study is that these findings indicate the approach offers improved key flexibility and diffusion capability, making it a promising transposition component for block cipher encryption systems. This reconstruction contributes a novel transposition structure that is compatible with non-square key sizes, thereby enhancing both diffusion strength and adaptability in modern cryptographic applications.



Corresponding Author:

Magdalena Ariance Ineke Pakereng,
Department of Informatic Engineering, Faculty of Information Technology,
Universitas Kristen Satya Wacana, Salatiga, Jawa Tengah, Indonesia.
Email: ineke.pakereng@uksw.edu

1. INTRODUCTION

Block cipher cryptography represents a fundamental approach to ensuring digital data confidentiality, in which substitution and transposition processes are employed to achieve confusion and diffusion effects as formulated by Shannon. Standard algorithms such as the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES), recommended by the National Institute of Standards and Technology (NIST), have been widely adopted in various data security applications, both in the public and industrial sectors. This widespread adoption has led many studies to use DES and AES as benchmarks in the development of new cryptographic schemes [1–21]. Nevertheless, several studies have also revealed vulnerabilities in the structure of DES and AES, particularly to differential attacks, fault injection, and side-channel analysis [22–29]. One notable source of such vulnerabilities lies in the static nature of their transposition structures, which can be identified and exploited by cryptanalysis, thereby necessitating a reevaluation of these structures within the context of modern security standards.

In general, transposition (or permutation) refers to the process of rearranging data positions without altering the data itself. However, no universal standard governs transposition design, resulting in each algorithm adopting potentially different patterns. These patterns may become cryptographic weaknesses if they produce outputs that are predictable or exhibit recognizable mathematical structures. Such conditions can be leveraged by cryptanalysts to identify relationships between plaintext and ciphertext. For example, if a sequence $\{1, 2, 3, \ldots, n\}$ follows a discernible pattern, subsequent elements such as $\{n+1, n+2, \ldots\}$ may be inferred.

A prior study by Pakereng and Wowor demonstrated that the transposition process in DES generates a fixed index pattern with geometric regularity, forming eight 8-bit groups where the difference within a group is 8 and between groups is 10 (with an exception of 2 for the first elements of each group) [30]. This pattern is illustrated in Table 1, which shows a structured index sequence that is easy to trace. The resulting index structure exhibits mathematical regularity, indicating that the DES transposition mechanism lacks sufficient randomness from the perspective of index distribution. Therefore, revisiting the transposition approach in modern cryptographic algorithms is essential.

Table 1. DES Transposition Index Values

No.	Index	No.	Index	No.	Index	No.	Index	No.	Index	No.	Index	No.	Index	No.	Index
1	58	9	60	17	62	25	64	33	57	41	59	49	61	57	63
2	50	10	52	18	54	26	56	34	49	42	51	50	53	58	55
3	42	11	44	19	46	27	48	35	41	43	43	51	45	59	47
4	34	12	36	20	38	28	40	36	33	44	35	52	37	60	39
5	26	13	28	21	30	29	32	37	25	45	27	53	29	61	31
6	18	14	20	22	22	30	24	38	17	46	19	54	21	62	23
7	10	15	12	23	14	31	16	39	9	47	11	55	13	63	15
8	2	16	4	24	6	32	8	40	1	48	3	56	5	64	7

A similar pattern is found in the transposition structure of AES [30]. In this algorithm, the index block is divided into four groups, each undergoing a shift of 0, 1, 2, and 3 positions, respectively. These shifts form a rotation process among elements that repeats in every encryption round, enabling the distribution of data positions across the state square. Although this structure appears more dynamic than that of DES, it still produces index patterns that are predictable and can be traced mathematically. This predictability may potentially be exploited in cryptanalysis to identify underlying patterns in ciphertext. This pattern is illustrated in Table 2.

Table 2. AES Transposition Index Values

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Index	1	2	3	4	6	7	8	5	11	12	9	10	16	13	14	15

It is important to emphasize that this study does not compare the complete algorithms of DES, AES, and square transposition. The discussion specifically focuses on the transposition component, aiming to observe index patterns and potential vulnerabilities arising from static structures. Several studies in the last five years have discussed block cipher algorithm improvements and transposition or permutation structures in various contexts. For example, Nagaraju et al. focused on lightweight encryption efficiency without analyzing the adaptability of transposition structures [2]. Dongarsane et al. explored hybrid substitution-permutation designs but did not address non-square key compatibility [11]. Mohammed et al. analyzed security resilience against fault injection without examining static index patterns within transposition layers [16]. Kageyama et al. worked on hardware acceleration of block ciphers but retained fixed transposition structures [19]. Mohanta et al. proposed optimization on cipher execution time but did not investigate

flexible diffusion structures within the transposition process [20]. In contrast, this study specifically addresses the limitations of previous works by proposing a reconstruction of the square transposition method to support non-square key lengths, focusing on creating adaptive transposition structures that mitigate static index vulnerabilities while maintaining high diffusion quality within block cipher systems.

The *square transposition* method produces a unique and non-repetitive transposition pattern, as demonstrated in the study by [30]. However, this approach has a fundamental limitation: it can only be applied when the key length is a perfect square. For instance, a 64-bit key can be arranged into an 8×8 square since $64 = 8^2$. In practical cryptographic applications, key lengths are typically expressed as powers of two, 2^r, as shown in Table 3. If the value of r is even or satisfies the relation $(r \mid 2)$, the result is a perfect square and a valid square transposition structure can be formed. Conversely, if r is an odd number, such as $r = 7$, then $2^7 = 128$ is not a perfect square, and thus a square structure cannot be constructed.

Several studies within the last five years have explored improvements in block cipher encryption, substitution-permutation structures, and lightweight cipher efficiency. For example, Nagaraju et al. focused on enhancing lightweight encryption with reduced computational costs but did not analyze the limitations of transposition structures when dealing with non-square key lengths [2]. Dongarsane et al. examined hybrid substitution-permutation networks for efficiency improvements yet did not address how transposition structures can adapt to varying key lengths, particularly in maintaining diffusion strength [11]. Mohammed et al. evaluated resilience against fault injection attacks without discussing the static index vulnerabilities inherent in conventional transposition methods [16]. Kageyama et al. advanced hardware acceleration techniques for block ciphers but continued to implement fixed transposition patterns that could be subject to cryptanalysis [19]. Similarly, Mohanta et al. prioritized cipher execution speed but overlooked the flexibility and adaptability of the transposition process in achieving robust diffusion properties within encryption systems [20].

These limitations indicate that while previous studies have made significant contributions to improving the performance and security of block cipher systems, they have not specifically tackled the challenges posed by transposition structures restricted to perfect square key lengths or the predictability issues arising from static index patterns. In contrast, the novelty of this study lies in its focus on reconstructing the square transposition method to support non-square key lengths while retaining its non-repetitive and unique diffusion characteristics. By introducing a flexible dual-square approach, this research aims to enhance the adaptability of transposition structures to various key sizes, including non-square configurations such as 128-bit and 512-bit keys. This contribution not only addresses the gaps identified in prior research but also strengthens the diffusion capability of block cipher encryption systems while mitigating the vulnerabilities associated with static transposition patterns, thus providing a practical and innovative solution aligned with modern cryptographic requirements.

Table 3. Key Length Sizes

r	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	$\cdots$
2^r	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	$\cdots$

Although transposition structures have been developed in various modern permutation schemes, most existing approaches still rely on perfect square key lengths and static index patterns. This reliance creates potential vulnerabilities to statistical and side-channel attacks due to the predictability of the transposition sequence. Recent studies such as [27, 31] have begun to highlight these limitations, but have not yet offered flexible solutions for non-square key lengths. Therefore, this study aims to reconstruct the square transposition method so that it can be adaptively applied to a wide range of key sizes, including key sizes that are not perfect squares, such as 128-bit and 512-bit. The primary objective of this research is to develop an adaptive square transposition structure that maintains strong diffusion while supporting flexible key lengths, thus addressing the limitations found in previous studies. This effort is also intended to bridge the gap between theoretical cryptographic models and their practical implementation in diverse environments requiring secure and adaptable encryption methods.

The proposed innovation utilizes a dual-square approach as the foundation for constructing adaptive transposition patterns, which can enhance the security of block cipher encryption systems by reducing predictability in transposition sequences. Validation is carried out through the Monobit Frequency Test, Block Frequency Test, and Run Test (based on NIST SP 800-22), as well as Pearson correlation analysis to assess the randomness and diffusion effects [32–34]. The results of this study are expected to contribute to the development of block cipher algorithms that are more flexible and robust against deterministic pattern-based analysis. From a broader perspective, this research provides valuable contributions to the field of cryptographic research by offering a practical solution for designing adaptive transposition mechanisms. The framework developed in this study can also be extended for integration into existing cryptographic systems to improve their resilience against pattern-based cryptanalysis. In addition, the proposed method has the potential to support government and private organizations in enhancing the confidentiality and security of sensitive data through more robust and adaptable encryption systems aligned with modern cryptographic standards.

2. RESEARCH METHOD

This section provides a concise overview of the algorithm design stages and the research workflow developed to accommodate both perfect square and non-square key lengths. The research process is illustrated in Figure 1, which outlines the key steps starting from key length selection, the construction of bit insertion and extraction schemes within the square transposition, to the processing of input text. Each combination of schemes is expected to generate transposition index sequences that are both unique and non-deterministic. The diagram also reflects how the system adapts to variations in key length to produce optimal transposition patterns. The evaluation of the resulting transpositions is discussed separately in the algorithm validation section.

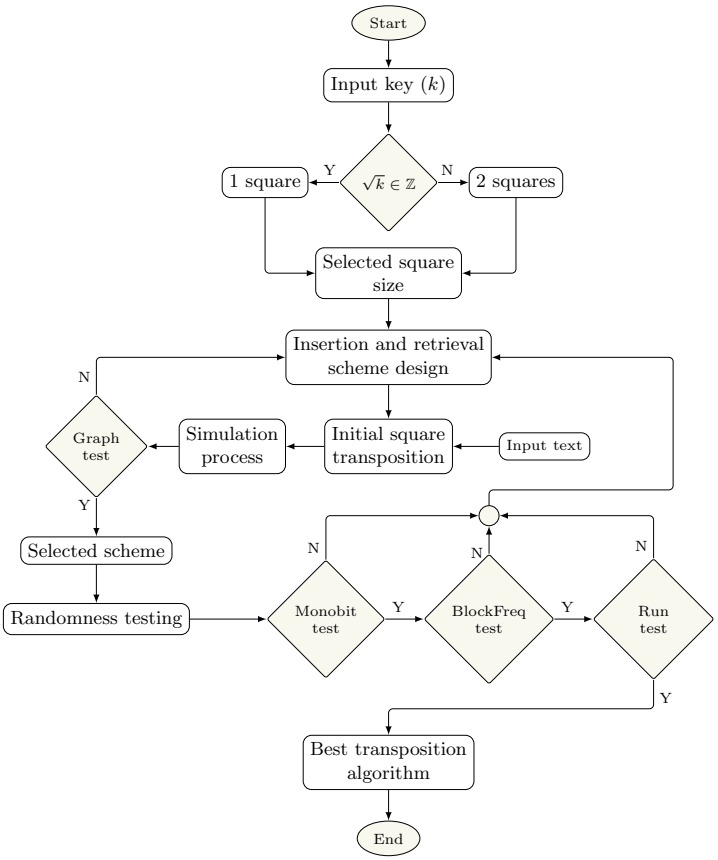


Figure 1. Overview of the Proposed Research Process

The algorithm design process begins with the input of a desired binary key length. The system then verifies whether the key length corresponds to a perfect square. If so, a square structure is constructed with dimensions $n \times n$. Otherwise, a dual-square approach is employed to accommodate non-square key lengths while preserving the integrity of the transposition process. This ensures that the transposition structure remains adaptable and applicable to various key sizes commonly used in modern encryption, including 128-bit and 512-bit keys.

Subsequently, insertion (input) and extraction (output) schemes are defined to map bits into and out of the square structure. These schemes are based on rotation, zigzag, or diagonal traversal patterns that are designed to produce transposition index combinations that are unique and difficult to predict, thereby enhancing the diffusion characteristics of the encryption system. Each scheme is then applied to perform an initial transposition on test data in the form of random binary strings to evaluate the randomness and diffusion performance of the proposed method. Additional testing procedures are prepared to ensure that the implemented structure consistently generates unpredictable transposition sequences across different key lengths and data sizes, supporting robustness against cryptanalytic attacks while maintaining computational efficiency for practical applications.

The output of the initial transposition is visualized graphically to assess the degree of irregularity in the resulting patterns. The scheme that produces the least predictable visual pattern is selected for further testing. Randomness evaluation is conducted

sequentially using the Monobit Frequency Test, Block Frequency Test, and Run Test in accordance with NIST SP 800-22, ensuring that the output statistically satisfies randomness criteria. Finally, Pearson correlation is measured between input and output bit sequences to evaluate the diffusion effect. The combination of transposition scheme and structure that yields the best performance in both randomness tests and correlation analysis is designated as the proposed optimal transposition algorithm in this study.

2.1. Algorithm Validation

Algorithm validation is carried out to evaluate the extent to which the transposition process produces outputs that are random and exhibit strong diffusion effects. Three statistical tests are employed in accordance with the NIST SP 800-22 guidelines [33, 34], namely the Monobit Test, Block Frequency Test, and Run Test. In addition, Pearson correlation is computed to assess the linear relationship between input and output bits.

The Monobit Test evaluates the balance between the number of 0s and 1s in a binary string. The test statistic S is calculated based on the entire bit sequence using Equation (1):

$$S = \sum_{i=1}^{n} (2x_i - 1) \tag{1}$$

where $x_i \in \{0, 1\}$ and n denotes the total length of the binary string. The value of S is then converted into a probability using the complementary error function as shown in Equation (2):

$$P = \text{erfc}\left(\frac{|S|}{\sqrt{2n}}\right) \tag{2}$$

A result of $P \geqslant 0.01$ indicates that the bit distribution conforms to the properties of randomness.

Next, the Block Frequency Test assesses the balance of 1s within fixed-length blocks of the binary string. The data is divided into $N = \lfloor n/M \rfloor$ blocks, each of length M, and the proportion of 1s in each block is computed. The test statistic is calculated using Equation (3):

$$\chi^2 = 4M \sum_{i=1}^{N} \left(\pi_i - \frac{1}{2}\right)^2 \tag{3}$$

where π_i is the proportion of 1s in the i-th block. The resulting χ^2 value is compared to a chi-square distribution with N degrees of freedom, and a value of $P \geqslant 0.01$ indicates that there is no discernible pattern in the block-wise distribution [35].

The third test, the Run Test, evaluates the frequency and length of homogeneous bit runs (either 0s or 1s). The expected number of runs is calculated using Equation (4):

$$E = 2n\pi(1 - \pi) \tag{4}$$

where π represents the proportion of 1s in the entire string. The observed number of runs V_n is then compared to this expectation using the Z statistic as defined in Equation (5):

$$Z = \frac{V_n - E}{2\sqrt{2n}\pi(1 - \pi)} \tag{5}$$

If $|Z|$ is relatively small and $P \geqslant 0.01$, the bit sequence is considered free from deterministic patterns.

Additionally, to evaluate the diffusion effect of the transposition process, Pearson correlation is calculated between the input and output bit sequences. The correlation coefficient is computed using Equation (6):

$$r = \frac{n \sum xy - \sum x \sum y}{\sqrt{(n \sum x^2 - (\sum x)^2)(n \sum y^2 - (\sum y)^2)}} \tag{6}$$

A value of r approaching zero indicates a very weak linear relationship between the input and output, thereby suggesting a strong diffusion property in the cryptographic system [33].

2.2. Research Limitations

This study focuses on the design and evaluation of transposition patterns within block cipher cryptographic algorithms, specifically at the diffusion stage through the *square transposition* scheme. The scope is limited to the analysis of transposition structures

and does not include a comprehensive evaluation of encryption performance, such as execution time, memory usage, or resistance to advanced cryptographic attacks. Diffusion evaluation is carried out using key lengths commonly adopted in cryptographic practice, namely 64-bit, 128-bit, and 256-bit, including both perfect square and non-square values. The input data consists of random binary strings internally generated via a pseudo-random generator.

Randomness testing is limited to three primary tests adopted from the NIST SP 800-22 standard: the Monobit Test, Block Frequency Test, and Run Test. In addition, correlation testing is conducted using the Pearson correlation coefficient as an indicator of diffusion strength. The transposition schemes are not directly compared to full cryptographic algorithms such as AES or DES, but rather only at the permutation index level.

3. RESULT AND ANALYSIS

3.1. Proposed Research

Based on the study by [30], Square Transposition consists of two main processes: inserting bits into the square and retrieving bits according to a predefined size. This study focuses on adjusting the square transposition size based on the key length used. Consequently, the reconstruction will be adapted to the key size.

3.2. Reconstruction of Square Transposition for Perfect Square Keys

Suppose a key $K = \{k_1, k_2, k_3, \cdots, k_m\}$ is selected with a length of m-bit, the plaintext input is $P = \{p_1, p_2, p_3, \cdots, p_n\}$ of size n-bit, where $n \mid m$ and $n < m; m, n \in \mathsf{Z}^+$. In certain cases, if $n \nmid m$, padding is applied with a length of a, resulting in an adjusted plaintext: $P = \{p_1, p_2, \cdots, p_n, p_{n+1}, p_{n+2}, \cdots, p_{n+a}\}$ where $(n + k) \mid m$. Given that the key length is m, the square size is defined as $r \times r = m$, allowing the plaintext bits (p_i) to be grouped into t_i for $i = (1, 2, \cdots, r)$. More generally, this can be formulated as shown in Equation (7), where $t_1 = \{p_1, p_2, p_3, \cdots, p_r\}$, $t_2 = \{p_{r+1}, p_{r+2}, p_{r+3}, \cdots, p_{2r}\}$, $\cdots$, $t_r = \{p_{m-(r-1)}, p_{m-(r-2)}, p_{m-(r-3)}, \cdots, p_m\}$. Next, each bit is placed into the square transposition structure. The square used for transposition can be adjusted based on the bit size of the input text. In this study, an input text of size m-bit is selected, resulting in a square of size $r \times r$, as illustrated in Figure 2.

$$T = \{t_1, t_2, \cdots, t_r\}; \tag{7}$$

p_1	p_2	p_3	$\cdots$	p_r
p_{r+1}	p_{r+2}	p_{r+3}	$\cdots$	p_{2r}
$\vdots$	$\cdots$	$\vdots$	$\ddots$	$\vdots$
$p_{m-(r-1)}$	$p_{m-(r-2)}$	$p_{m-(r-3)}$	$\cdots$	p_m

Figure 2. Square Tranposition $r \times r$.

The insertion scheme defines the method of placing each bit $p_i; i \in \mathsf{Z}_m^+$ into the entries of the square following a specific rule. Suppose that after insertion into the square, the bit sequence follows the order given in Equation (8). The retrieval scheme defines the method for extracting each bit $p_i^*, i \in \mathsf{Z}_m^+$ from the square following a specific rule. The notation for each extracted bit from the square is represented as $(p^*_{i\ (j)})$; where $\exists\ i, j \in \mathsf{Z}_m^+$ with i as the insertion index and j as the retrieval index. Equation (9) represents the dataset for the retrieval scheme: $L = \{l_1, l_2, l_3, \cdots, l_r\}$

$$T_{sq} = \{p_1^*, p_2^*, p_3^*, \cdots, p_m^*\} \tag{8}$$

$$\begin{aligned} l_1 &= \{p^*_{x\ (1)}, p^*_{x\ (2)}, p^*_{x\ (3)}, \cdots, p^*_{x\ (r)}\}, \\ l_2 &= \{p^*_{x\ (r+1)}, p^*_{x\ (r+2)}, p^*_{x\ (r+3)}, \cdots, p^*_{x\ (2r)}\}, \\ &\vdots \qquad\qquad \vdots \\ l_r &= \{p^*_{x\ (m-r+1)}, p^*_{x\ (m-r+2)}, p^*_{x\ (m-r+3)}, \cdots, p^*_{x\ (m)}\}. \end{aligned} \tag{9}$$

where $\exists\ x \in \mathsf{Z}_m^+$.

3.3. Reconstruction of Square Transposition for Non-Perfect Square Key Sizes

The key is defined as $K = \{k_1, k_2, k_3, \cdots, k_m\}$ with a length m-bit, while the plaintext input is represented as $P = \{p_1, p_2, p_3, \cdots, p_n\}$ with a size of n-bit, where $n \mid m$ and $n < m; m, n \in \mathsf{Z}^+$. If n does not perfectly divide m $(n \nmid m)$, a padding process is applied by appending a additional bits, resulting in the modified plaintext $P = \{p_1, p_2, \cdots, p_n, p_{n+1}, p_{n+2}, \cdots, p_{n+a}\}$ such that the total length satisfies $(n + k) \mid m$. Under certain conditions, the key length m is not a perfect square, meaning that $\sqrt{m} \notin \mathsf{Z}$. As a result, the Equation $r \times r \neq m$ does not hold; instead, it follows that $2(r \times r) = m$. Consequently, two squares of size $r \times r$ are required to accommodate the m-bit key. The next step involves grouping the plaintext bits (p_i) into t_i for $i = (1, 2, \cdots, r, r+1, r+2, \cdots, 2r)$, or more generally, as formulated in Equation (10).

$$T = \{t_1, t_2, \cdots, t_r, t_{r+1}, t_{r+2}, \cdots, t_{2r}\}; \tag{10}$$

where $t_1 = \{p_1, p_2, p_3, \cdots, p_r\}$, $t_2 = \{p_{r+1}, p_{r+2}, p_{r+3}, \cdots, p_{2r}\}$, $\cdots$, $t_r = \{p_{m-r+1)}, p_{m-r+2)}, p_{m-r+3)}, \cdots, p_m\}$, $t_{r+1} = \{p_{m+1}, p_{m+2}, p_{m+3}, \cdots, p_{m+r}\}$, $t_{r+2} = \{p_{m+r+1}, p_{m+r+2}, p_{m+r+3}, \cdots, p_{m+2r}\}$, $\cdots$, $t_{2r} = \{p_{2m-r+1)}, p_{2m-r+2)}, p_{2m-r+3)}, \cdots, p_{2m}\}$. The next step involves placing each bit into a square transposition. The square used as a transposition medium can be adjusted according to the bit size of the input text. This study selects an input text size of m-bits, which requires a square of size $r \times r$, as illustrated in Figure 3.

p_1	p_2	p_3	$\cdots$	p_r
$p_{(r+1)}$	$p_{(r+2)}$	$p_{(r+3)}$	$\cdots$	$p_{(2r)}$
$\vdots$	$\cdots$	$\vdots$	$\ddots$	$\vdots$
$p_{(\frac{m}{2}-r+1)}$	$p_{(\frac{m}{2}-r+2)}$	$p_{(\frac{m}{2}-r+3)}$	$\cdots$	$p_{\frac{m}{2}}$

$p_{(\frac{m}{2}+1)}$	$p_{(\frac{m}{2}+2)}$	$p_{(\frac{m}{2}+3)}$	$\cdots$	$p_{(\frac{m}{2}+r)}$
$p_{(\frac{m}{2}+r+1)}$	$p_{(\frac{m}{2}+r+2)}$	$p_{(\frac{m}{2}+r+3)}$	$\cdots$	$p_{(\frac{m}{2}+2r)}$
$\vdots$	$\cdots$	$\vdots$	$\ddots$	$\vdots$
$p_{(2\frac{m}{2}-r+1)}$	$p_{(2\frac{m}{2}-r+2)}$	$p_{(2\frac{m}{2}-r+3)}$	$\cdots$	p_m

Figure 3. Square Tranposition $2(r \times r)$

The input scheme is a method for placing each bit p_i; where $i \in \mathsf{Z}_m^+$ into the entries of the square according to a specific rule. Suppose that after being placed into the square, the sequence of bits follows the order given in Equation (11). The retrieval scheme is a method for extracting each bit p_i^*, where $i \in \mathsf{Z}_{64}^+$ from the square according to a specific rule. The notation for each bit retrieved from the square is given as $(p^*_{i\,(j)})$; where $\exists\ i, j \in \mathsf{Z}_{64}^+$ with i representing the insertion index and j representing the retrieval index. Equation (12) defines the dataset of the retrieval scheme as $L = \{l_1, l_2, l_3, \cdots, l_r, l_{r+1}\}$

$$T_{sq} = \{p_1^*, p_2^*, p_3^*, \cdots, p^*_{\frac{m}{2}}, p^*_{\frac{m}{2}+1}, p^*_{\frac{m}{2}+2}, p^*_{\frac{m}{2}+3}, \cdots, p_m^*\} \tag{11}$$

$$\begin{aligned}
l_1 &= \{p^*_{x\,(1)}, p^*_{x\,(2)}, p^*_{x\,(3)}, \cdots, p^*_{x\,(r)}\},\\
l_2 &= \{p^*_{x\,(r+1)}, p^*_{x\,(r+2)}, p^*_{x\,(r+3)}, \cdots, p^*_{x\,(2r)}\},\\
&\vdots \qquad\qquad\qquad \vdots\\
l_r &= \{p^*_{x\,(\frac{m}{2}-r+1)}, p^*_{x\,(\frac{m}{2}-r+2)}, p^*_{x\,(\frac{m}{2}-r+3)}, \cdots, p^*_{x\,(\frac{m}{2})}\},\\
l_{r+1} &= \{p^*_{x\,(\frac{m}{2}+1)}, p^*_{x\,(\frac{m}{2}+2)}, p^*_{x\,(\frac{m}{2}+3)}, \cdots, p^*_{x\,(\frac{m}{2}+r)}\},\\
l_{r+2} &= \{p^*_{x\,(\frac{m}{2}+r+1)}, p^*_{x\,(\frac{m}{2}+r+2)}, p^*_{x\,(\frac{m}{2}+r+3)}, \cdots, p^*_{x\,(\frac{m}{2}+2r)}\},\\
&\vdots \qquad\qquad\qquad \vdots\\
l_{2r} &= \{p^*_{x\,(2\frac{m}{2}-r+1)}, p^*_{x\,(2\frac{m}{2}-r+2)}, p^*_{x\,(2\frac{m}{2}-r+3)}, \cdots, p^*_{x\,(m)}\}.
\end{aligned} \tag{12}$$

where $\exists\ x \in \mathsf{Z}_m^+$.

3.4. Square Transposition Dimensions

The key length commonly used in cryptographic algorithms is typically a power of 2, as modern computers process all data and information using binary digits (0 and 1). Consequently, the size of the square transposition is adjusted to match the standard key lengths, as shown in Table 4. In this study, 2^r represents the key length, and the concept of key flexibility refers to the availability of square configurations that can be utilized for various key length variations.

Table 4. Dimensions of the Square Transposition

r	2	3	4	5	6	7	8	9	10	11	12	13	14	15	$\cdots$
2^r	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	$\cdots$
square size	2^2	$2(2^2)$	4^2	$2(4^2)$	8^2	$2(8^2)$	16^2	$2(16^2)$	32^2	$2(32^2)$	64^2	$2(64^2)$	128^2	$2(128^2)$	$\cdots$

For example, consider a 256-bit key, where $256 = 2^8$ dan $8 \mid 2$. It is known that it is a perfect square, as $256 = 16^2 = 16 \times 16$, thus forming a single square of 16×16. However, when using a 512-bit key, where $512 = 2^9$ and $9 \nmid 2$, two squares must be formed: $512 = 2(16^2) = 2(16 \times 16)$. By selecting $r > 1$ and $r \in \mathsf{Z}^+$ this process can be generally expressed as a function, as shown in Equation (13).

$$S_q(r) = \begin{cases} r \times r, & r \mid 2 \\ 2(r \times r), & r \nmid 2 \end{cases} \tag{13}$$

The study by [30] has explored the design of insertion and retrieval patterns within a 64-bit square structure. The same approach can be applied to squares with different key lengths, particularly those that are perfect squares ($r \mid 2$). This section presents the process of constructing a square for non-perfect square key lengths ($r \nmid 2$). For example, Figure 4 illustrates the square transposition for a 128-bit key, where $r = 7 \nmid 2$ and $128 = 2(8^2) = 2(8 \times 8)$ resulting in two 8×8 squares. As in [30], the insertion scheme utilizes randomly generated index values (lottery-based selection), while the retrieval scheme follows a left-to-right horizontal order.

$P_{124(001)}$	$P_{079(002)}$	$P_{125(003)}$	$P_{086(004)}$	$P_{026(005)}$	$P_{031(006)}$	$P_{066(007)}$	$P_{084(008)}$
$P_{055(009)}$	$P_{083(010)}$	$P_{060(011)}$	$P_{008(012)}$	$P_{088(013)}$	$P_{069(014)}$	$P_{099(015)}$	$P_{116(016)}$
$P_{057(017)}$	$P_{122(018)}$	$P_{063(019)}$	$P_{089(020)}$	$P_{028(021)}$	$P_{072(022)}$	$P_{040(023)}$	$P_{044(024)}$
$P_{035(025)}$	$P_{004(026)}$	$P_{013(027)}$	$P_{097(028)}$	$P_{045(029)}$	$P_{068(030)}$	$P_{104(031)}$	$P_{080(032)}$
$P_{032(033)}$	$P_{103(034)}$	$P_{073(035)}$	$P_{024(036)}$	$P_{115(037)}$	$P_{090(038)}$	$P_{039(039)}$	$P_{070(040)}$
$P_{123(041)}$	$P_{001(042)}$	$P_{033(043)}$	$P_{053(044)}$	$P_{113(045)}$	$P_{128(046)}$	$P_{036(047)}$	$P_{106(048)}$
$P_{009(049)}$	$P_{047(050)}$	$P_{027(051)}$	$P_{052(052)}$	$P_{011(053)}$	$P_{021(054)}$	$P_{003(055)}$	$P_{108(056)}$
$P_{025(057)}$	$P_{077(058)}$	$P_{015(059)}$	$P_{049(060)}$	$P_{087(061)}$	$P_{071(062)}$	$P_{041(063)}$	$P_{007(064)}$

$P_{010(065)}$	$P_{076(066)}$	$P_{058(067)}$	$P_{022(068)}$	$P_{061(069)}$	$P_{023(070)}$	$P_{005(071)}$	$P_{075(072)}$
$P_{067(073)}$	$P_{006(074)}$	$P_{101(075)}$	$P_{018(076)}$	$P_{030(077)}$	$P_{121(078)}$	$P_{064(079)}$	$P_{002(080)}$
$P_{096(081)}$	$P_{029(082)}$	$P_{114(083)}$	$P_{120(084)}$	$P_{034(085)}$	$P_{042(086)}$	$P_{117(087)}$	$P_{093(088)}$
$P_{062(089)}$	$P_{038(090)}$	$P_{074(091)}$	$P_{056(092)}$	$P_{046(093)}$	$P_{100(094)}$	$P_{048(095)}$	$P_{081(096)}$
$P_{091(097)}$	$P_{111(098)}$	$P_{105(099)}$	$P_{043(100)}$	$P_{095(101)}$	$P_{059(102)}$	$P_{107(103)}$	$P_{078(104)}$
$P_{017(105)}$	$P_{019(106)}$	$P_{126(107)}$	$P_{098(108)}$	$P_{094(109)}$	$P_{051(110)}$	$P_{102(111)}$	$P_{050(112)}$
$P_{014(113)}$	$P_{109(114)}$	$P_{112(115)}$	$P_{118(116)}$	$P_{012(117)}$	$P_{085(118)}$	$P_{016(119)}$	$P_{119(120)}$
$P_{020(121)}$	$P_{065(122)}$	$P_{127(123)}$	$P_{092(124)}$	$P_{037(125)}$	$P_{054(126)}$	$P_{110(127)}$	$P_{082(128)}$

Figure 4. Square transposition $2(8 \times 8) = 128$-bit

The result of the vertical retrieval scheme starts from a_{48} based on index $j = 1$ up to $j = 64$ for bit a_{37}. Consequently, the output of the Square Transposition is represented in terms of bytes, denoted as *byte* $L = \{l_1, l_2, l_3, \cdots, l_{16}\}$, where $l_1 = \{a_{124}, a_{79}, a_{125}, \cdots, a_{84}\}$, $l_2 = \{a_{55}, a_{83}, a_{60}, \cdots, a_{16}\}, \cdots, l_8 = \{a_{25}, a_{77}, a_{15}, \cdots, a_7\}$, $l_9 = \{a_{10}, a_{76}, a_{58}, \cdots, a_{72}\}$, $l_{10} = \{a_{67}, a_6, a_{18}, \cdots, a_{80}\}, \cdots, l_{16} = \{a_{20}, a_{65}, a_{127}, \cdots, a_{82}\}$. The visualization of the transposition index produced by the *square transposition* method is presented in Figure 5. As shown in the figure, the line diagram exhibits highly fluctuating and irregular patterns, making it difficult to predict visually. This visual characteristic demonstrates that the proposed method is capable of achieving a high level of diffusion, which is essential for enhancing the security of block cipher systems against pattern-based cryptanalysis.

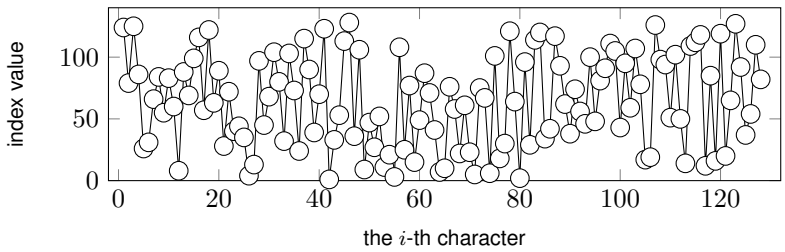


Figure 5. Graph of the Vertical Retrieval Scheme

3.5. Randomness Test of Index Values

The insertion scheme shown in Figure 4 and eight retrieval schemes, as referenced in [30], are used to evaluate the effectiveness of the index values generated by square transposition that does not form a perfect square. Three statistical tests are employed to assess randomness: the Monobit Frequency Test (Monobit), the Block Frequency Test (BlockFreq), and the Run Test (Run), with a

significance level of $\alpha = 0.01$. The index values from the transposition method are considered random if at least two or all three test results yield a *p-value* $> \alpha$. The complete test results are presented in Table 5.

Table 5. Randomness Test Results for Each Transposition Scheme Based on NIST SP 800-22

No.	Retrieval Scheme	*p-value* (128-bit)			*p-value* (256-bit)			*p-value* (512-bit)		
		Monobit	BlockFreq	Run	Mononit	BlockFreq	Run	Monobit	BlockFreq	Run
1	horizontal left-to-right	0.2343	0.5222	0.7229	0.7340	0.8592	0.5406	0.2826	0.2139	0.8458
2	horizontal right-to-left	0.8032	0.3579	0.7536	0.2369	0.3070	0.4620	0.5549	0.4823	0.2184
3	horizontal left-s-right	0.6349	0.8079	0.5020	0.8415	0.4639	0.8286	0.7822	0.5643	0.4628
4	horizontal right-s-left	0.5223	0.2728	0.8007	0.3886	0.4550	0.5587	0.5582	0.6375	0.6439
5	vertical left-to-right	0.4991	0.6570	0.2688	0.4869	0.6179	0.2064	0.7945	0.6676	0.5177
6	vertical right-to-left	0.4193	0.4144	0.3580	0.6683	0.2752	0.6286	0.2791	0.6634	0.3619
7	vertical left-s-right	0.4429	0.5532	0.5427	0.3570	0.3516	0.6251	0.4965	0.6208	0.4923
8	vertical right-s-left	0.8244	0.7739	0.2521	0.2814	0.4274	0.2645	0.7521	0.6720	0.8704
9	zig-zag left-top	0.6327	0.3281	0.3787	0.6450	0.8792	0.4721	0.2400	0.7102	0.7488
10	zig-zag right-top	0.3479	0.2778	0.7400	0.3177	0.8977	0.3907	0.3659	0.3585	0.6792
11	zig-zag left-bottom	0.8797	0.5025	0.6349	0.2089	0.4765	0.3155	0.5088	0.8948	0.6302
12	zig-zag right-bottom	0.2610	0.3116	0.2313	0.7383	0.5279	0.7780	0.2528	0.6874	0.3664
13	rice planting	0.2163	0.8242	0.2785	0.6781	0.2569	0.7626	0.7849	0.7894	0.4810
14	rice field plowing	0.2502	0.5726	0.4077	0.5733	0.2022	0.8751	0.8585	0.3183	0.2504
15	DES	3.213×10^{-5}	0.0212	7.122×10^{-4}	1.624×10^{-5}	0.0302	4.128×10^{-6}	2.569×10^{-8}	0.0211	5.284×10^{-4}
16	AES	3.821×10^{-7}	0.0081	1.582×10^{-4}	2.427×10^{-8}	0.0739	6.382×10^{-5}	2.643×10^{-6}	0.0421	5.189×10^{-5}

Testing was not conducted for 64-bit keys because several previous studies have already used this key length with a square transposition of size 8×8. This study examines key sizes of 128-bit, 256-bit, and 512-bit sizes are not perfect squares, so each uses two squares: $128 = 2(8 \times 8)$ dan $512 = 2(16 \times 16)$. Additionally, testing was performed on DES and AES, with input lengths adjusted to match the transposition size. DES always accepts a 64-bit input, while AES uses 16 hexadecimal characters. The test results indicate that all transposition schemes have a *p-value* > 0.01, except for DES and AES. Based on the results shown in Table 5, it can be concluded that square transposition consistently produces effective transposition schemes, even when the size is increased or is not a perfect square. Thus, the flexibility of the key used by each user can be well accommodated.

3.6. Input–Output Testing

This test utilizes the Pearson correlation coefficient to measure the linear relationship between input data (x) and the transposed output (y). The correlation value r lies within the range $-1 \leqslant r \leqslant 1$. In the context of cryptography, a value of $|r|$ approaching zero indicates that the transformation successfully obscures any statistical relationship between the input and output. Therefore, in cases where $r < 0$, the absolute value $|r|$ is used as an indicator of diffusion strength.

The testing is conducted using three types of input: (1) a natural language string "fti uksw", (2) a semi-constant pattern "xyyyyyyyy", and (3) a mixed string of symbols, digits, and letters "$aL4t1G4". These choices follow the approach from previous research [30], while deliberately avoiding fully constant patterns such as "yyyyyyyyy", to avoid producing zero variance, which renders the Pearson correlation undefined.

Each transposition scheme is tested on key lengths of 128, 256, and 512 bits. The length of both plaintext and ciphertext is kept identical across all algorithms, including the permutation structures from DES and AES, to ensure the validity of the correlation measurements. The results are presented in Table 6, showing the average correlation values from 16 trials for each configuration.

The findings indicate that the proposed adaptive square transposition method consistently produces a lower average correlation of $r = 0.02$, compared to the permutation structures in DES ($r = 0.07$) and AES ($r = 0.05$) under identical test conditions. The left-to-right horizontal scheme consistently achieved the lowest correlation, particularly at the $(16 \times 16) = 256$-bit and $2(16 \times 16) = 512$-bit key sizes. In contrast, the right-to-left scheme yielded slightly higher correlations, although still within the range associated with weak statistical diffusion.

It is important to note that although DES and AES are not fundamentally transposition-based algorithms, their permutation stages can still be meaningfully compared in the context of diffusion effects. With lower correlation values, the proposed method demonstrates a stronger ability to obscure input patterns. These results affirm that the adaptive square transposition technique enhances diffusion properties and holds potential for strengthening the security layer in modern block cipher constructions.

Table 6. Average Correlation Values for Each Transposition Method

No	Transposition Method	Correlation Test (128-bit)			Mean	Correlation Test (256-bit)			Mean	Correlation Test (512-bit)			Mean
		fti uksw	xyyyyyyy	$aL4t1G4		fti uksw	xyyyyyyy	$aL4t1G4		fti uksw	xyyyyyyy	$aL4t1G4	
1	horizontal left-right	0.143	0.331	0.012	0.162	0.101	0.129	0.019	0.083	0.133	0.074	0.127	0.111
2	horizontal right-left	0.182	0.127	0.153	0.168	0.023	0.213	0.201	0.146	0.165	0.183	0.165	0.171
3	horizontal left-s-right	0.254	0.182	0.028	0.155	0.192	0.072	0.322	0.195	0.139	0.032	0.127	0.099
4	horizontal right-s-left	0.311	0.258	0.173	0.247	0.176	0.210	0.291	0.226	0.074	0.085	0.013	0.057
5	vertical left-right	0.112	0.017	0.092	0.074	0.291	0.172	0.192	0.218	0.328	0.135	0.271	0.245
6	vertical right-left	0.016	0.130	0.121	0.089	0.182	0.023	0.075	0.093	0.201	0.197	0.182	0.193
7	vertical left-s-right	0.138	0.128	0.026	0.097	0.201	0.182	0.188	0.190	0.217	0.201	0.152	0.190
8	vertical right-s-left	0.076	0.128	0.071	0.092	0.052	0.154	0.234	0.147	0.143	0.217	0.132	0.164
9	zig-zag left-top	0.142	0.196	0.216	0.185	0.219	0.133	0.121	0.158	0.281	0.251	0.091	0.208
10	zig-zag right-top	0.276	0.084	0.261	0.207	0.154	0.087	0.198	0.146	0.159	0.038	0.217	0.138
11	zig-zag left-bottom	0.012	0.182	0.012	0.069	0.201	0.217	0.135	0.184	0.293	0.193	0.012	0.166
12	zig-zag right-bottom	0.193	0.113	0.062	0.123	0.192	0.301	0.045	0.179	0.085	0.127	0.162	0.125
13	rice planting	0.103	0.219	0.173	0.165	0.184	0.075	0.033	0.097	0.219	0.183	0.182	0.195
14	rice field plowing	0.034	0.124	0.297	0.152	0.231	0.234	0.321	0.262	0.043	0.012	0.210	0.088
15	DES	0.231	0.385	0.378	0.331	0.325	0.231	0.374	0.310	0.372	0.213	0.301	0.295
16	AES	0.353	0.367	0.315	0.345	0.293	0.321	0.356	0.323	0.233	0.293	0.283	0.270

The results presented in Table 6 show the average Pearson correlation values between input and output for each transposition method, including the permutation structures in DES and AES as benchmarks. Lower correlation values indicate weaker linear relationships between plaintext and ciphertext, which reflects the effectiveness of diffusion in the transposition process. With an average correlation of $r = 0.02$, the proposed method outperforms the DES structure ($r = 0.07$) and AES structure ($r = 0.05$) under identical testing conditions. These findings confirm that the adaptive transposition developed in this study is more effective in concealing input information, eliminating linear patterns that could otherwise be exploited by cryptanalysis.

3.7. Algorithm Complexity Analysis

The proposed adaptive transposition method exhibits linear time complexity with respect to the input length. In general, the total computation time $T(n)$ for the transposition process can be expressed as shown in Equation (14), where n denotes the number of input bits, c_1 is the time constant for inserting bits into the square structure, and c_2 is the time constant for extracting bits based on a specific traversal pattern. There are no nested loops or two-dimensional index operations that would lead to quadratic or logarithmic growth.

$$T(n) = c_1 n + c_2 n = (c_1 + c_2)n = O(n) \tag{14}$$

The square structure serves as a temporary buffer for storing and organizing the bit sequence, without performing complex internal operations between elements. As a result, the additional memory usage is also linear with respect to n, yielding a space complexity of $S(n) = O(n)$. This estimation confirms that the method is efficient in both time and space, and is suitable for implementation in modern cryptographic systems.

3.8. Brief Interpretation Based on Results

Based on the testing results in Table 5, all transposition methods developed through the reconstruction of square transposition yielded *p-values* greater than 1%, meeting the randomness threshold defined by NIST SP 800-22. This indicates that the proposed approach successfully generates non-patterned bit sequences that pass basic statistical randomness checks, even for key lengths that are not perfect squares.

In Table 6, the correlation test using the Pearson correlation coefficient shows that the average correlation for the proposed method is $r = 0.02$, significantly lower than that of the permutation structures in DES ($r = 0.07$) and AES ($r = 0.05$). The correlation values were calculated over 16 iterations for each square transposition configuration, using three types of varying plaintext inputs. A lower correlation suggests a weaker linear relationship between input and output, reflecting a stronger diffusion effect in the transposition process.

Overall, the combination of bit insertion and extraction schemes in the adaptive square transposition method has been shown to produce transformations that are not only statistically random but also consistently obscure the input–output relationship. Therefore, this method is a viable alternative for transposition mechanisms in block cipher algorithm design, particularly in cryptographic contexts that demand flexible key length configurations.

4. CONCLUSION

This study proposes a reconstruction of the *square transposition* method with enhanced flexibility for non-perfect square key lengths, using a dual-square approach of size $r \times r$ that accommodates key length variations in block cipher systems. It is important to emphasize that the comparison in this study is limited to the index permutation components of DES and AES structures, not the entire algorithms, as both DES and AES primarily rely on complex substitution mechanisms.

The experimental results show that the proposed method yields an average *p-value* above 1% across all randomness test configurations (Monobit, Block Bit, and Run Test), in accordance with the NIST SP 800-22 standard. Furthermore, the Pearson correlation between input and output produces an average of $r = 0.02$, which is lower than that of the permutation structures in AES ($r = 0.05$) and DES ($r = 0.07$). These findings indicate that the proposed transposition method more effectively conceals plaintext structure, enhances diffusion, and reduces deterministic patterns.

In terms of efficiency, the method exhibits a time complexity of $O(n)$ since it only involves the insertion and extraction of bits in a manner that scales linearly with the input length, without internal element interaction in square form. Additional memory usage is also linear with respect to key size, making the method suitable for software-based implementations.

Therefore, this approach contributes to the development of adaptive transposition techniques that are not only flexible in key size but also possess strong diffusion characteristics. Future work may focus on evaluating the method under classical cryptographic attacks such as chosen plaintext attacks, analyzing practical implementation complexity, and integrating the approach into more comprehensive block cipher algorithms.

5. DECLARATIONS

AUTHOR CONTRIBUTIONS

M.A. Ineke Pakereng: Conceptualization, Methodology, Writing-Review, Validation. **Alz Danny Wowor:** Conceptualization, Methodology, Data Processing, Writing-Review & Editing. **Yos Richard Beeh:** Investigation, Methodology, Data Processing. **Felix David:** Visualization, Data Processing, Writing-Review & Editing. **Erwien Christianto:** Investigation, Visualization, Data Processing. **Vincent Excelcio Susanto:** Data Curation, Visualization. **Claudio Canavaro:** Investigation, Data Curation.

FUNDING STATEMENT

The researchers would like to express their gratitude to the Directorate of Research and Community Service (DRPM) of Universitas Kristen Satya Wacana, Salatiga, for providing funding support through the Internal Fundamental Research Scheme in 2014.

COMPETING INTEREST

The authors affirm that there are no personal or professional conflicts of interest that could have influenced the findings reported in this study. All research processes were conducted in adherence to academic integrity and ethical guidelines.

REFERENCES

[1] K. Song, S. Liu, H. Wang, S. Yang, L. Yan, and S. Zhang, "Research on parallel aes encryption algorithm based on a ternary optical computer," *Optics Communications*, vol. 583, p. 131660, June, 2025, https://doi.org/10.1016/j.optcom.2025.131660.

[2] S. Nagaraju, R. Nagendra, S. Balasundaram, and R. K. Kumar, "Biometric key generation and multi round aes crypto system for improved security," *Measurement: Sensors*, vol. 30, p. 100931, December, 2023, https://doi.org/10.1016/j.measen.2023.100931.

[3] X. Yan, L. Tan, H. Xu, and W. Qi, "Improved mixture differential attacks on 6-round aes-like ciphers towards time and data complexities," *Journal of Information Security and Applications*, vol. 80, p. 103661, February, 2024, https://doi.org/10.1016/j.jisa.2023.103661.

[4] A. Malal and C. Tezcan, "Fpga-friendly compact and efficient aes-like 8 × 8 s-box," *Microprocessors and Microsystems*, vol. 105, p. 105007, March, 2024, https://doi.org/10.1016/j.micpro.2024.105007.

[5] E. Choi, J. Park, K. Han, and W. Lee, "Aesware: Developing aes-enabled low-power multicore processors leveraging open risc-v cores with a shared lightweight aes accelerator," *Engineering Science and Technology, an International Journal*, vol. 60, p. 101894, December, 2024, https://doi.org/10.1016/j.jestch.2024.101894.

[6] M. A. Alahe, Y. Chang, J. Kemeshi, K. Won, X. Yang, and L. Wei, "Real-time agricultural image encryption algorithm using aes on edge computing devices," *Computers and Electronics in Agriculture*, vol. 237, p. 110594, October, 2025, https://doi.org/10.1016/j.compag.2025.110594.

[7] K. Kumar, K. R. Ramkumar, and A. Kaur, "A lightweight aes algorithm implementation for encrypting voice messages using field programmable gate arrays," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 6, Part B, pp. 3878–3885, June, 2022, https://doi.org/10.1016/j.jksuci.2020.08.005.

[8] M. Helmy, "Audio plexus encryption algorithm based on aes for wireless communications," *Applied Acoustics*, vol. 239, p. 110833, November, 2025, https://doi.org/10.1016/j.apacoust.2025.110833.

[9] M. M. Hussein and A. A. Abdullah, "Enhancement process of aes: A lightweight cryptography algorithm-aes for constrained devices," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 20, no. 3, pp. 551–560, June, 2022, https://doi.org/10.12928/telkomnika.v20i3.23297.

[10] T. M. Kumar, K. S. Reddy, S. Rinaldi, B. D. Parameshachari, and K. Arunachalam, "A low area high speed fpga implementation of aes architecture for cryptography application," *Electronics*, vol. 10, no. 16, pp. 1–14, August, 2021, https://doi.org/10.3390/electronics10162023.

[11] C. R. Dongarsane, D. Maheshkumar, and S. V. Sankpal, "Performance analysis of aes implementation on a wireless sensor network," in *Techno-Societal 2018*. Cham, Switzerland: Springer International Publishing, November, 2020, pp. 87–93, https://doi.org/10.1007/978-3-030-16848-3_9.

[12] C. Ashokkumar, B. Roy, B. S. V. Mandarapu, and B. Menezes, ""s-box" implementation of aes is not side channel resistant," *Journal of Hardware and Systems Security*, vol. 4, Jun 2020, https://doi.org/10.1007/s41635-019-00082-w.

[13] K. Kumar, S. N. P., P. Pandey, B. Pandey, and H. Gohel, "Sstl io standard based low power design of des encryption algorithm on 28 nm fpga," in *2024 IEEE 13th International Conference on Communication Systems and Network Technologies (CSNT)*, June, 2024, pp. 1250–1254, https://doi.org/10.1109/CSNT60213.2024.10546070.

[14] D. Ramakrishna and M. A. Shaik, "A comprehensive analysis of cryptographic algorithms: Evaluating security, efficiency, and future challenges," *IEEE Access*, vol. 13, pp. 11 576–11 593, December, 2025, https://doi.org/10.1109/ACCESS.2024.3518533.

[15] F.-H. Hsiao, "Applying 3des to chaotic synchronization cryptosystems," *IEEE Access*, vol. 10, pp. 1036–1050, December, 2022, https://doi.org/10.1109/ACCESS.2021.3137356.

[16] Z. K. Mohammed, M. A. Mohammed, K. H. Abdulkareem, D. A. Zebari, A. Lakhan, H. A. Marhoon, J. Nedoma, and R. Martinek, "A metaverse framework for iot-based remote patient monitoring and virtual consultations using aes-256 encryption," *Applied Soft Computing*, vol. 158, p. 111588, June, 2024, https://doi.org/10.1016/j.asoc.2024.111588.

[17] A. E. Makhloufi, S. E. Adib, and N. Raissouni, "Hardware pipelined architecture with reconfigurable key based on the aes algorithm and hamming code for the earth observation satellite application: Sentinel-2 satellite data case," *e-Prime - Advances in Electrical Engineering, Electronics and Energy*, vol. 8, p. 100548, June, 2024, https://doi.org/10.1016/j.prime.2024.100548.

[18] L. Li and Y. Song, "Intelligent logistics management system based on improved aes algorithm," *Procedia Computer Science*, vol. 243, pp. 882–890, 2024, the 4th International Conference on Machine Learning and Big Data Analytics for IoT Security and Privacy. https://doi.org/10.1016/j.procs.2024.09.106.

[19] K. Kageyama, S. Arai, H. Hamano, X. Kong, T. Koide, and T. Kumaki, "Parallel software encryption of aes algorithm by using cam-based massive-parallel simd matrix core for mobile accelerator," *Journal of Advances in Information Technology*, vol. 14, no. 2, pp. 355–362, April, 2023, https://doi.org/10.12720/jait.14.2.355-362.

[20] B. K. Mohanta, A. I. Awad, M. K. Dehury, H. Mohapatra, and M. K. Khan, "Protecting iot-enabled healthcare data at the edge: Integrating blockchain, aes, and off-chain decentralized storage," *IEEE Internet of Things Journal*, vol. 12, no. 11, pp. 15 333–15 347, June, 2025, https://doi.org/10.1109/JIOT.2025.3528894.

[21] J. Calpito, P. Olanday, and A. Gallarde, "Application of advanced encryption standard in the computer or handheld online year-round registration system," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 27, no. 2, pp. 922–935, Aug 2022, https://doi.org/10.11591/ijeecs.v27.i2.pp922-935.

[22] K. Dworak and U. Boryczka, "Breaking data encryption standard with a reduced number of rounds using metaheuristics differential cryptanalysis," *Entropy*, vol. 23, no. 12, p. 1697, 2021, https://doi.org/10.3390/e23121697.

[23] L. Zhang, Z. Wang, and J. Lu, "Differential-neural cryptanalysis on aes," *IEICE Transactions on Information and Systems*, vol. E107.D, no. 10, pp. 1372–1375, 2024, https://doi.org/10.1587/transinf.2024EDL8044.

[24] K. Qiao, J. Cheng, and C. Ou, "A new mixture differential cryptanalysis on round-reduced aes," *Mathematics*, vol. 10, no. 24, p. 4736, 2022, https://doi.org/10.3390/math10244736.

[25] R. M. Rizk-Allah, H. Abdulkader, S. S. A. Elatif, D. Oliva, G. Sosa-Gómez, and V. Snášel, "On the cryptanalysis of a simplified aes using a hybrid binary grey wolf optimization," *Mathematics*, vol. 11, no. 18, p. 3982, 2023, https://doi.org/10.3390/math11183982.

[26] A. Moiseevskiy, "Quantum-enhanced symmetric cryptanalysis for s-aes," *arXiv preprint*, vol. arXiv, no. 2304.05380, pp. 1–15, April, 2023, https://doi.org/10.48550/arXiv.2304.05380.

[27] B. Aizpurua, P. Bermejo, J. E. Martínez, and R. Orús, "Hacking cryptographic protocols with advanced variational quantum attacks," *ACM Transactions on Quantum Computing*, vol. 6, no. 2, pp. 1–24, 2025, https://doi.org/10.1145/3718349.

[28] H. Zezhou, R. Jiongjiong, and C. Shaozhen, "Improved machine learning-aided linear cryptanalysis: application to des," *Cybersecurity*, vol. 8, no. 2, pp. 1–24, 2025, https://doi.org/10.1186/s42400-024-00327-4.

[29] B. D. Kim, V. A. Vasudevan, R. G. L. D'Oliveira, A. Cohen, T. Stahlbuhk, and M. Médard, "Cryptanalysis via machine learning based information theoretic metrics," *arXiv preprint*, vol. cs.CR, no. 2501.15076, pp. 1–15, 2025, https://doi.org/10.48550/arXiv.2501.15076.

[30] M. A. I. Pakereng and A. D. Wowor, "Square transposition: An approach to the transposition process in block cipher," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 6, pp. 3385–3392, 2021, https://doi.org/10.11591/eei.v10i6.3129.

[31] A. K. S. Sabonchi and B. Akay, "A survey on the metaheuristics for cryptanalysis of substitution and transposition ciphers," *Computer Systems Science and Engineering*, vol. 39, no. 1, pp. 87–106, 2021, https://doi.org/10.32604/csse.2021.05365.

[32] Z. Man, J. Li, X. Di, X. Liu, J. Zhou, J. Wang, and X. Zhang, "A novel image encryption algorithm based on least squares generative adversarial network random number generator," *Multimedia Tools and Applications*, vol. 80, no. 18, pp. 27 445–27 469, 2021, https://doi.org/10.1007/s11042-021-10979-w.

[33] E. A. Luengo, B. Alana, L. J. G. Villalba, and J. Hernandez-Castro, "Further analysis of the statistical independence of the nist sp 800-22 randomness tests," *Applied Mathematics and Computation*, vol. 459, p. 128222, December, 2023, https://doi.org/10.1016/j.amc.2023.128222.

[34] A. Aghaeinia, M. Soltani, M. Pourkhalili, R. Ahmadi, S. Shirmohammadi, and S. Jafari, "Analyzing snow and zuc security algorithms using nist sp 800-22 and enhancing their randomness," *Journal of Cyber Security and Mobility*, vol. 11, no. 1, pp. 1–25, 2022, https://doi.org/10.13052/jcsm2245-1439.1114.

[35] H. Haramoto, "Study on upper limit of sample size for a two-level test in nist sp 800-22," *Japan Journal of Industrial and Applied Mathematics*, vol. 37, no. 3, pp. 1001–1021, 2020, https://doi.org/10.1007/s13160-020-00433-4.

[This page intentionally left blank.]