# A Novel Approach for Enhancing Advanced Encryption Standard Performance and Cryptographic Resilience

Muthu Meenakshi Ganesan, Sabeen Selvaraj
Department of Computer Science-Faculty of Science and Humanities,
SRM Institute of Science and Technology, Kattankulathur, India

***Abstract*—Advanced Encryption Standard (AES) encrypts data in blocks of sixteen bytes to secure confidential data stored in the cloud. For cloud-based systems, enhancements in existing encryption techniques are necessary as the nature of cyber threats evolves and computational speed becomes increasingly critical. This study presents an enhanced design of AES that substitutes two special operations, Byte Transformation and Bits Permuted Bytes, with the conventional S-Box operation to improve the speed and security of the encryption method. The suggested round structure in the new approach of AES, which preserves the original data block size, consists of the following operations: Byte Transformation, Shift Rows, Bits Permuted Bytes, Add Round Key, and Mix Columns. The analysis of the strict avalanche effect, correlation coefficient, entropy, execution time, and throughput outcomes confirms that the developed scheme improves the security and processing speed.**

***Keywords*—Cryptography; NIST; AES; block cipher; key expansion; symmetric encryption; galois field; statistical techniques; cryptanalysis***

## I. Introduction

Cloud platforms that utilize symmetric encryption algorithms, such as AES, ensure that data transmission, storage, and communication are secure for the government, healthcare, and financial sectors [1], [2]. AES secures vast volumes of private information from illegal access and breaches in these cloud-based systems [3]. In 2000, the National Institute of Standards and Technology (NIST) in the United States replaced the Data Encryption Standard with AES due to its superior performance in both hardware and software [4], [5]. Researchers often utilize AES to secure wireless networks, data storage, and communications [6], [7].

Several cryptanalytic studies suggest that improving diffusion and confusion mechanisms can further reinforce AES against potential future cryptanalytic attacks. When internal transformations lack sufficient randomness or complexity, the encryption process may reveal patterns that can be exploited through statistical or differential attacks. These vulnerabilities highlight the importance of strengthening AES, not only to address current vulnerabilities but also to remain resilient against evolving and more sophisticated attacks.

This study addresses these limitations by proposing a modified AES encryption algorithm that enhances both cryptographic strength and operational efficiency. The proposed approach introduces structural changes to AES transformations to improve diffusion, reduce statistical correlation, and strengthen resistance against cryptanalytic attacks.

The rest of this study is organized as follows: Section II summarizes related work; Section III recommends the enhanced AES encryption approach; Section IV outlines the statistical tests and evaluation criteria applied to assess the resilience of existing and proposed encryption algorithms. Section V presents the experimental results obtained from these evaluations, while Section VI discusses the key findings and their implications. Finally, Section VII concludes this study by highlighting potential directions for future research.

## II. Related Work

Many studies have been conducted to improve the security and efficiency of symmetric encryption algorithms. These studies suggest numerous technical variants of existing encryption methods to boost security and performance. Parthasarathy et al. [8] proposed a lightweight symmetric encryption that combined the Elliptic Curve Diffie-Hellman key exchange algorithm with a proprietary random number generator, resulting in a unique S-box and innovative folding operations for transmitting medical data to the cloud. The throughput of this method was higher than that of AES; however, it lacks the statistical evaluation of cryptographic strength.

Baladhay et al. [9] proposed an enhanced version of the AES algorithm by replacing the MixColumns function with an alternate transformation to improve the security and efficiency of encryption and decryption processes. This method is ideal for applications requiring high-speed video encryption due to the performance improvements. On the other hand, entropy and correlation analysis, which could offer additional information on cryptographic strength, are excluded from this work. Similarly, to increase security and performance for systems with low resources, Hammod et al. [10] introduced a Modified Lightweight AES (ML-AES).

Sirajuddin et al. [11] designed a modified Rijndael algorithm for resource-constrained Internet of Things devices. The shift column phase was changed, the sub-byte step was removed, and a crossover pre-processing method was added. The research did not evaluate the ciphertext's randomness or statistical independence, although these improvements were made to increase efficiency and security. Wenceslao Jr. et al. [12] developed a new version of AES encryption that retained the

original Rijndael S-Box and introduced a second S-Box using affine transformations for the Mix-Columns step. This method is faster but exhibits a slightly reduced avalanche effect, which may affect the diffusion property of the encryption process.

A DNA-based multi-round encryption approach was presented by Al-Shargabi et al. [13] to preserve confidential health data. Improving entropy levels while maintaining an avalanche effect similar to conventional AES allowed this technique to outperform AES in data security. By eliminating the Mix-Columns stage and adding a continuous fraction function to compress the ciphertext, Mohammad et al. [14] developed a lightweight AES encryption (LWC-AES) for devices with limited resources. LWC-AES reduced processing complexity and enhanced transfer rates without compromising security. Altigani et al. [15] introduced Polymorphic-AES (P-AES) for adaptive and flexible encryption using the initial key. P-AES showed exceptional resilience to attacks by modifying the Sub Bytes, Shift Rows, and Mix Columns procedures to improve security with negligible performance impact. Salih et al. [16] modified a 3D chaotic map and combined two dynamic XOR tables with a dynamic Mix-Columns transformation to strengthen the AES encryption method. While this approach improved security, additional performance evaluation would enhance its practical relevance.

Assafli et al. [17] modified AES encryption by replacing the traditional S-box with a dynamic one based on Unix epoch time. This new method solely preserved and improved the avalanche effect for single-bit input changes. Abikoye et al. [18] proposed an enhanced AES technique, with modified S-Box and Shift-Rows transformations to strengthen security. This method made the Sub Bytes operation round-key dependent and randomized the Shift Rows operation exclusively to improve the avalanche effect. Manoj Kumar et al. [19] presented a new AES key expansion technique primarily to increase the execution speed, allowing simultaneous subkey generation by isolating the operation into two concurrent blocks. This innovative approach was implemented on an FPGA (Virtex 5 XC5VLX50T) to secure ECG signal encryption and decryption in real-time communication.

In summary, many studies prioritize either efficiency or security without achieving an adequate balance. While several enhancements to AES exist, most lack rigorous statistical evaluation. The proposed system addresses these gaps by achieving a balance between security and efficiency, with its reliability confirmed through a thorough statistical analysis.

## III. METHODOLOGY

AES uses 128-bit data blocks and encrypts in 10, 12, or 14 rounds, based on the key length, which can be 128, 192, or 256 bits [20]. It involves three key processes: Key expansion generates round keys from the original cipher key, encryption transforms plaintext into ciphertext through various operations, and decryption reverses these operations to recover the original plaintext [21].

### A. Key Expansion in AES Algorithm

In AES-128, the key expansion process, depicted in Fig. 1, begins by dividing the 128-bit initial key into four 32-bit words (W0, W1, W2, W3), forming the initial part of the key schedule [19]. A left circular byte shift (RotateWord), a substitution using the S-Box (SubWord), and an XOR with the appropriate round constant (RCj) are used to compute the remaining 40 words [22]. The SubWord transformation uses the S-Box depicted in Fig. 2, a 16 x 16 lookup table for byte substitution. The round constants, described in Table I, are applied at specific stages during the key expansion to introduce non-linearity. These transformations are applied to designated words W3, W7, W11, and every fourth word thereafter, to generate W3', W7', W11', respectively. The subsequent words are created by XORing each new word with previously generated words, iteratively producing all 44 words. These words are divided into 11 sets, each with 4 words, forming the round keys used for encryption and decryption.

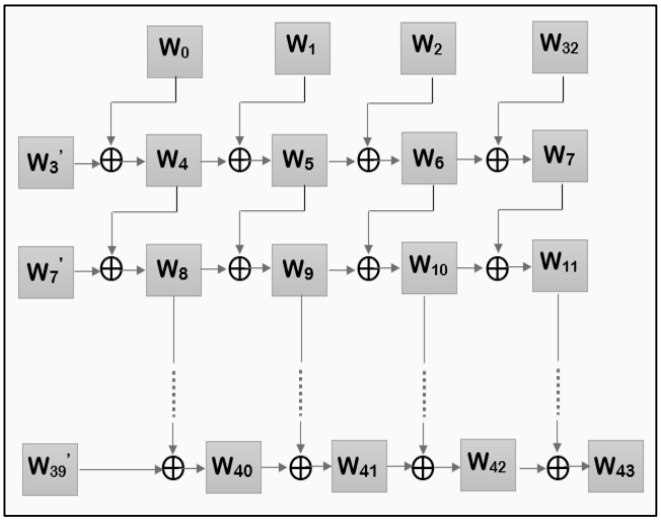


Fig. 1. AES key scheduling process.

TABLE I. CONSTANTS USED PER ROUND

| Round(j) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| RCj | 0x01 | 0x02 | 0x04 | 0x08 | 0x10 | 0x20 | 0x40 | 0x80 | 0x1b | 0x36 |

### B. Existing AES Encryption Algorithm

During encryption or decryption, a state matrix (a 4x4 grid of bytes) represents the 128-bit plaintext block. The matrix arranges each byte of the 128-bit block into a single cell in column-major order. This matrix undergoes various operations during encryption, including SubstituteBytes, ShiftRows, MixColumns, and AddRoundKey, with the final round omitting MixColumns [23]. These operations are described below:

*1) SubstituteBytes.* In this substitution phase, each byte in the state matrix is replaced with the value of the S-Box (illustrated in Fig. 2) based on its original value, introducing non-linearity into the encryption process [24]. Fig. 3 shows the inverse AES S-Box used during decryption.

*2) ShiftRows.* This step enhances byte dispersion by executing a cyclic left shift on each row of the state matrix. First row remains unaffected, while the second row is shifted one byte to the left, the third row is shifted two bytes to the left, and the fourth row is shifted three bytes to the left.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| 1 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| 2 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| 3 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| 4 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| 5 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| 6 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| 7 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| 8 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| 9 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| a | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| b | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| c | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| d | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| e | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| f | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

Fig. 2. AES S-Box.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 52 | 09 | 6a | d5 | 30 | 36 | a5 | 38 | bf | 40 | a3 | 9e | 81 | f3 | d7 | fb |
| 1 | 7c | e3 | 39 | 82 | 9b | 2f | ff | 87 | 34 | 8e | 43 | 44 | c4 | de | e9 | cb |
| 2 | 54 | 7b | 94 | 32 | a6 | c2 | 23 | 3d | ee | 4c | 95 | 0b | 42 | fa | c3 | 4e |
| 3 | 08 | 2e | a1 | 66 | 28 | d9 | 24 | b2 | 76 | 5b | a2 | 49 | 6d | 8b | d1 | 25 |
| 4 | 72 | f8 | f6 | 64 | 86 | 68 | 98 | 16 | d4 | a4 | 5c | cc | 5d | 65 | b6 | 92 |
| 5 | 6c | 70 | 48 | 50 | fd | ed | b9 | da | 5e | 15 | 46 | 57 | a7 | 8d | 9d | 84 |
| 6 | 90 | d8 | ab | 00 | 8c | bc | d3 | 0a | f7 | e4 | 58 | 05 | b8 | b3 | 45 | 06 |
| 7 | d0 | 2c | 1e | 8f | ca | 3f | 0f | 02 | c1 | af | bd | 03 | 01 | 13 | 8a | 6b |
| 8 | 3a | 91 | 11 | 41 | 4f | 67 | dc | ea | 97 | f2 | cf | ce | f0 | b4 | e6 | 73 |
| 9 | 96 | ac | 74 | 22 | e7 | ad | 35 | 85 | e2 | f9 | 37 | e8 | 1c | 75 | df | 6e |
| a | 47 | f1 | 1a | 71 | 1d | 29 | c5 | 89 | 6f | b7 | 62 | 0e | aa | 18 | be | 1b |
| b | fc | 56 | 3e | 4b | c6 | d2 | 79 | 20 | 9a | db | c0 | fe | 78 | cd | 5a | f4 |
| c | 1f | dd | a8 | 33 | 88 | 07 | c7 | 31 | b1 | 12 | 10 | 59 | 27 | 80 | ec | 5f |
| d | 60 | 51 | 7f | a9 | 19 | b5 | 4a | 0d | 2d | e5 | 7a | 9f | 93 | c9 | 9c | ef |
| e | a0 | e0 | 3b | 4d | ae | 2a | f5 | b0 | c8 | eb | bb | 3c | 83 | 53 | 99 | 61 |
| f | 17 | 2b | 04 | 7e | ba | 77 | d6 | 26 | e1 | 69 | 14 | 63 | 55 | 21 | 0c | 7d |

Fig. 3. AES inverse S-box.

*3) MixColumns.* This step applies a linear transformation to each column of the 4 x 4 state matrix. It mixes the bytes within each column by multiplying the MixColumns matrix over a Galois Field (GF($2^8$)). This step enhances diffusion by ensuring that each input byte influences multiple output bytes, as shown in Eq. (1):

$$\text{MixColumns Matrix} = \begin{matrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{matrix} \quad (1)$$

During decryption, the inverse transformation is applied using the Inverse MixColumn matrix, which is shown in Eq. (2):

$$\text{Inverse MixColumns Matrix} = \begin{matrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{matrix} \quad (2)$$

*4) AddRoundKey.* This step combines the key with the data to strengthen the encryption by XORing, the round key with the state matrix.

During decryption, the process reverses encryption by applying inverse operations in reverse order. It begins with InverseShiftRows to restore the original row positions, followed by InverseSubstituteBytes using the inverse S-box, and AddRoundKey XORs the state matrix with the corresponding round key. Finally, InverseMixColumns (except in the final round) to unmix the columns to recover the plaintext from the ciphertext.

### C. Proposed Enhanced AES Encryption Architecture

In the new design of AES encryption, the input block size remains 128 bits; however, the Sub Bytes operation has been eliminated. Instead, two new operations, Byte Transformation and Bits Permuted Bytes, have been introduced. In each round of the proposed approach, Bytes Transformation, Shift Rows, Mix Columns, Bits Permuted Bytes, and Add Round Key transform the state matrix. These transformations aim to improve the encryption process's confusion and diffusion properties. The newly introduced operations are described below:

*1) Byte transformation operation.* Let S represent the 4x4 state matrix, where each element ($S_{ij}$) is an 8-bit byte. AES transforms this state matrix through various stages as part of its encryption and decryption mechanisms. In the proposed method, the Byte Transformation function is applied to each byte ($S_{ij}$). This operation introduces non-linearity and enhances the diffusion properties of the encryption process. This transformation is defined as follows:

Let $S_{ij} = b_7b_6b_5b_4b_3b_2b_1b_0$; here $b_7$ and $b_0$ are the Most Significant Bit (MSB) and the Least Significant Bit (LSB), respectively. The transformation for each bit ( $t_i$ ) of the transformed byte ($T_{ij}$) is defined as follows:

$t_7 = b_7$ (MSB remains unchanged), $t_6 = b_6 \oplus b_7$ , $t_5 = b_5 \oplus b_6$ , $t_4 = b_4 \oplus b_5$ , $t_3 = b_3 \oplus b_4$ , $t_2 = b_2 \oplus b_3$ , $t_1 = b_1 \oplus b_2$, and $t_0 = b_0 \oplus b_1$. Here $t_7t_6t_5t_4t_3t_2t_1t_0$ ($T_{ij}$) is the transformed byte and $\oplus$ denotes the XOR operation. For example, consider a byte. $S_{ij}$ , where $b_7b_6b_5b_4b_3b_2b_1b_0$ = 11011010. $t_7 = b_7 = 1$, $t_6 = b_6 \oplus b_7 = 1 \oplus 1 = 0$ , $t_5 = b_5 \oplus b_6 = 0 \oplus 1 = 1$ , $t_4 = b_4 \oplus b_5 = 1 \oplus 0 = 1$ , $t_3 = b_3 \oplus b_4 = 1 \oplus 1 = 0$ , $t_2 = b_2 \oplus b_3 = 0 \oplus 1 = 1$ , $t_1 = b_1 \oplus b_2 = 1 \oplus 0 = 1$ , and $t_0 = b_0 \oplus b_1 = 0 \oplus 1 = 1$. The transformed byte $T_{ij}$becomes 10110111. This transformation is visually represented in Fig. 4.

*2) Inverse byte transformation operation.* The Inverse Byte Transformation operation undoes the transformations applied during the encryption phase. Given the transformed byte, $S_{ij}$ can be recovered as follows:

$b_7 = t_7$ (MSB remains unchanged), $b_6 = t_6 \oplus b_7$ , $b_5 = t_5 \oplus b_6$ , $b_4 = t_4 \oplus b_5$ , $b_3 = t_3 \oplus b_4$ , $b_2 = t_2 \oplus b_3$ , $b_1 = t_1 \oplus b_2$, and $b_0 = t_0 \oplus b_1$

For example, consider a byte, $T_{ij} = t_7t_6t_5t_4t_3t_2t_1t_0 =$ 10110111. $b_7 = t_7 = 1$ (MSB remains unchanged), $b_6 = t_6 \oplus b_7 = 0 \oplus 1 = 1$ , $b_5 = t_5 \oplus b_6 = 1 \oplus 1 = 0$ , $b_4 = t_4 \oplus b_5 = 1 \oplus 0 = 1$ , $b_3 = t_3 \oplus b_4 = 0 \oplus 1 = 1$ , $b_2 = t_2 \oplus b_3 = 1 \oplus 1 = 0$ , $b_1 = t_1 \oplus b_2 = 1 \oplus 0 = 1$ , and $b_0 = t_0 \oplus b_1 = 1 \oplus 1 = 0$ . Now, combine all the bits. $b_7b_6b_5b_4b_3b_2b_1b_0$ to form the original byte ($S_{ij}$) as 11011010. In the encryption process, the Bytes Transformation procedure modifies each byte, and the Inverse Bytes Transformation procedure restores each byte in decryption. The step-by-step process of this inverse transformation is shown in Fig. 5.

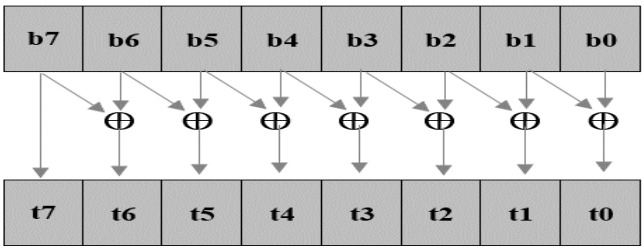


Fig. 4. Byte transformation operation.

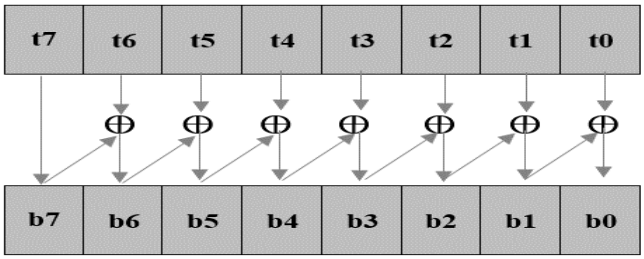


Fig. 5. Inverse byte transformation operation.

*3) Bits permuted bytes operation.* In the Bits Permuted Bytes operation, each byte in the state matrix $S_{ij}$ ( $b_7b_6b_5b_4b_3b_2b_1b_0$) undergoes a bit permutation based on the involutory permutation table 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 during the encryption phase. The involutory permutation table is self-inverse. This feature simplifies both encryption and decryption. It removes the need for a separate inverse table. This permutation rearranges the bits. The bit at index 3 ($b_4$) moves to index 0, the bit at index 2 ($b_5$) moves to index 1, the bit at index 1 ($b_6$) moves to index 2, the bit at index 0 ($b_7$) moves to index 3, the bit at index 7 ($b_0$) moves to index 4, the bit at index 6 ($b_1$) moves to index 5, the bit at index 5 ($b_2$) moves to index 6, and the bit at index 4 ( $b_3$ ) moves to index 7. In this transformation, the original bit order of each byte in the state matrix is rearranged to $T_{ij} = b_4b_5b_6b_7b_0b_1b_2b_3$. For example, consider a byte $S_{ij}$= 11011010 in the state matrix. This byte is transformed by rearranging its bits to obtain $T_{ij}$ =10110101. This bit permutation is applied to each byte in the state matrix, converting the original byte into a new form.

*4) Inverse bits permuted bytes operation.* The Inverse Bits Permuted Bytes operation reverses the bit permutation applied during encryption. The encryption phase rearranges the bits of each byte in the state matrix according to the involutory permutation table 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4, whereas the inverse operation restores the original bit order. For example, a byte $T_{ij}$ =10110101, when passed through the inverse transformation, it reverts to its original form $S_{ij}$= 11011010. Fig. 6 and Fig. 7 show the Bits Permuted Bytes and Inverse Bits Permuted Bytes operations, which rearrange and restore the bit order within each byte during encryption and decryption, respectively, in the developed scheme.

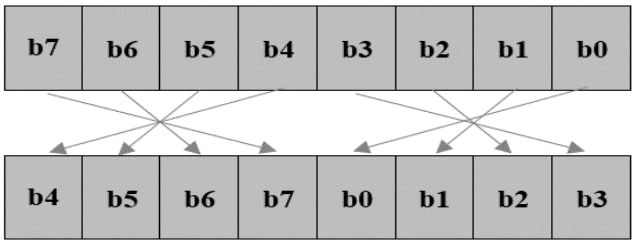


Fig. 6. Bits permuted bytes operation.

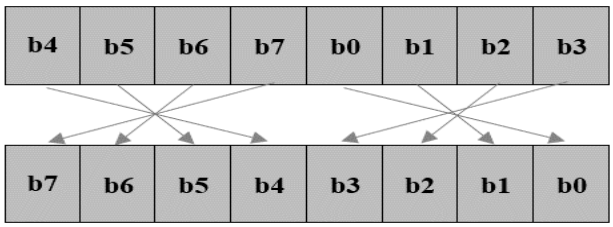


Fig. 7. Inverse bits permuted bytes operation.

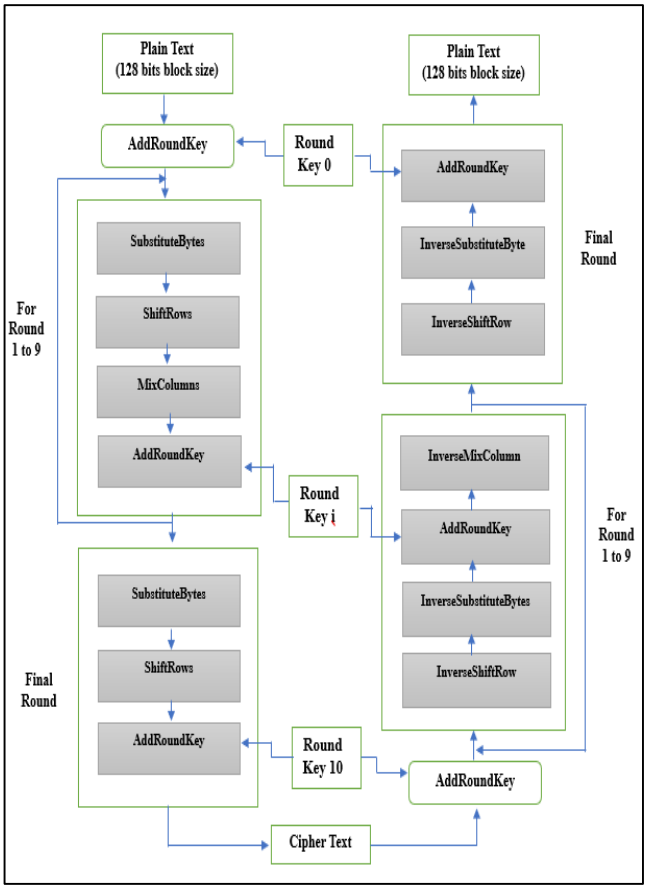


Fig. 8. Standard AES algorithm.

To enhance efficiency, the final round of the proposed encryption approach omits the Bits Permuted Bytes operation and Mix columns, thereby reducing complexity while maintaining security strength. Furthermore, the number of rounds has been decreased to accelerate execution speed. The standard AES functions with ten rounds for a 128-bit key size, as shown in Fig. 8, while the developed encryption strategy executes only five rounds with the same key size, as illustrated in Fig. 9.

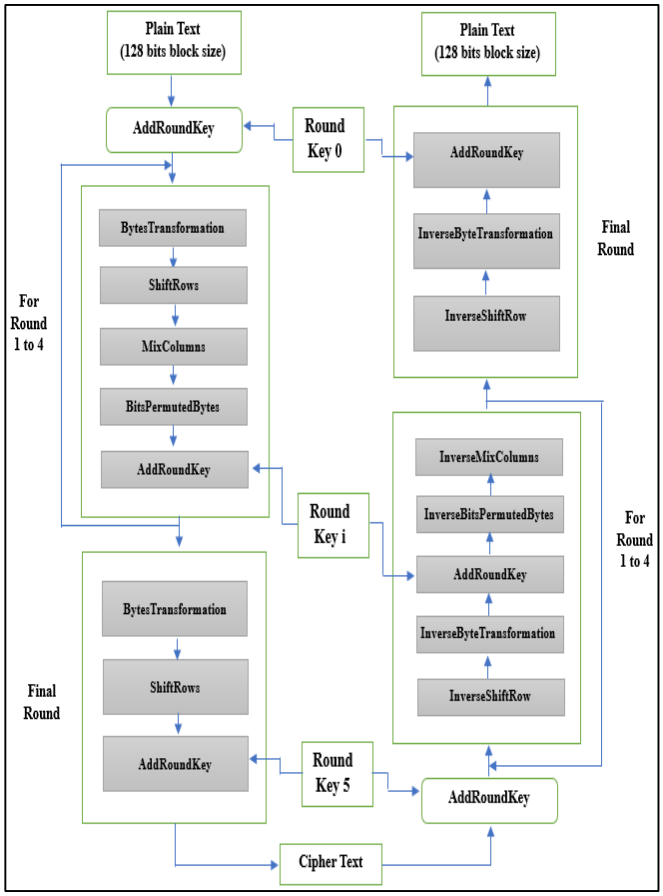


Fig. 9. Enhanced AES algorithm.

## IV. EVALUATION

### A. Strict Avalanche Effect

The Avalanche Effect is determined by flipping each bit in the n-bit plaintext and observing the percentage of bits that differ in the resulting ciphertext. Averaging the percentages across all n-bit flips gives the Strict Avalanche Effect (SAE). This metric ensures the algorithm's robustness and resistance to cryptanalysis. The formula used to calculate the SAE [25] is shown in Eq. (3):

$$SAE = \frac{1}{n}\sum_{i=1}^{n}\frac{HammingWeight(C \oplus C'_i)X100}{n} \quad (3)$$

Here, C denotes the ciphertext obtained by encrypting the original plaintext, $C_i'$ denotes the ciphertext produced by encrypting the plaintext with the $i^{th}$ bit flipped, and n represents the total number of bits in the ciphertext. One hundred random 128-bit plaintexts and keys were used to evaluate SAE.

### B. Correlation Coefficient

This measure implies the level of a linear association between the plaintext and encrypted output. For an encryption algorithm to be considered secure, the ciphertext must have a minimal linear dependence on the plaintext, as indicated by a correlation coefficient close to zero [26].

The following steps are used to determine the correlation coefficient between plaintext and ciphertext:

*1) Plaintext and ciphertext* are converted into binary sequences of 0s and 1s.

*2) Mean calculation.* The mean value of the plaintext bits ($\mu_X$) was computed using Eq. (4), and the mean value of the ciphertext bits ($\mu_Y$) was determined using Eq. (5):

$$\mu_X = \frac{1}{N}\sum_{i=1}^{N} X_i \quad (4)$$

$$\mu_Y = \frac{1}{N}\sum_{i=1}^{N} Y_i \quad (5)$$

Here, N is the number of bits in the plaintext or ciphertext, and $X_i$ and $Y_i$ are the individual bits of the plaintext and ciphertext, respectively.

*3) Covariance Calculation ( $Cov(X,Y)$) .* Covariance measures how much the plaintext and ciphertext vary together: The covariance between the plaintext bits and ciphertext bits was calculated using Eq. (6):

$$\mathrm{Cov(X,Y)} = \frac{1}{N}\sum_{i=1}^{N}(X_i - \mu_X)\ \mathrm{X}\ (Y_i - \mu_Y) \quad (6)$$

Here, $\mu_X$ is the mean of plaintext bits, $\mu_Y$ is the mean of ciphertext bits, $X_i - \mu_X$ is the deviation of the $i^{th}$ plaintext bit from the mean, and $Y_i - \mu_Y$ is the deviation of the $i^{th}$ ciphertext bit from its mean. The deviations measure how each bit in the plaintext and ciphertext differs from their respective mean.

*4) Standard deviation.* The standard deviation of the plaintext bits ($\sigma_X$) and the ciphertext bits ($\sigma_Y$) was calculated using Eq. (7) and Eq. (8), respectively.

$$\sigma_X = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(X_i - \mu_X)^2} \quad (7)$$

$$\sigma_Y = \frac{1}{N}\sqrt{\sum_{i=1}^{N}(Y_i - \mu_Y)^2} \quad (8)$$

*5) Correlation coefficient.* The value of the correlation coefficient (r) was calculated between the plaintext and ciphertext bits using Eq. (9):

$$r = \frac{Cov(X,Y)}{\sigma_X\ \mathrm{X}\ \sigma_Y} \quad (9)$$

One thousand random 128-bit plaintexts and keys were used to evaluate this metric.

### C. Entropy

The entropy test assesses the degree of unpredictability and randomness in the ciphertext. A higher entropy indicates greater randomness in the ciphertext, as it is more difficult to crack the encryption. To determine the average entropy of encrypted ciphertexts in their binary representation, convert each

ciphertext $C_i$ to $B_i$(binary form). The entropy of the ciphertext $H(X)$ can be expressed for Shannon entropy [27], [28] using Eq. (10):

$$H(X) = -\sum_{i=1}^{N} P(x_i) log_2 P(x_i) \quad (10)$$

In this context, $x_i$ represents the possible outcomes, and N is the number of unique outcomes. The entropy for each binary string (B) was calculated using Eq. (11):

$$H(B) = -[P(0) log_2 P(0) + P(1) log_2 P(1)] \quad (11)$$

Here, $P(0)$ and $P(1)$ are the probabilities of the binary outcomes 0s and 1s, respectively, in the binary ciphertext ($B$).

The average entropy $\bar{H}$ across all N ciphertexts was calculated using Eq. (12):

$$\bar{H} = \frac{1}{N} \sum_{i=1}^{N} H(B_i) \quad (12)$$

This method interprets every bit as a random variable with two possible outcomes and obtains the average entropy of the binary representations, where N is the total number of ciphertexts. One thousand random 128-bit plaintexts and keys were used to evaluate this randomness property of the produced ciphertexts.

### D. Execution Time

Execution time impacts efficiency, performance, and resource usage of cryptographic systems. To calculate the execution time of the modified and standard AES algorithms implemented in Python, the start and end times were recorded, and the difference value was calculated. The evaluations were performed on a 12th Gen Intel Core i7-1255U CPU running at 1.70 GHz. The dataset consists of a 1025 KB text file, a 1028 KB PDF document, a 1126 KB JPEG image, a 1026 KB MP3 audio file, and a 1031 KB MP4 video file used for the evaluation. The encryption and decryption times were calculated over 10 runs for each algorithm using all file types and averaged to obtain a reliable result.

### E. Throughput

The encryption and decryption throughputs were calculated separately using Eq. (13) [29]:

$$\text{Throughput (KB/sec)} = \frac{\text{Total File Size (KB)}}{\text{Total Time (sec)}} \quad (13)$$

The Total File Size is the sum of all the file sizes used, which includes text, PDF, JPEG, MP3, and MP4. The amount of time needed to complete encrypting or decrypting all files is referred to as Total Time. This method enables a reliable evaluation of the algorithm's performance on various file types and sizes.

## V. Results

### A. Strict Avalanche Effect

The test results in Table II and Fig. 10 clearly show that the suggested new encryption process achieves a higher SAE of 53.46%, compared to 49.55% for the standard AES encryption. These results highlight that the modified encryption design achieves improved diffusion and higher security in fewer rounds.

TABLE II. Measured SAE in Standard and Enhanced AES

| Algorithm | SAE |
|---|---|
| Standard AES | 49.55 |
| Enhanced AES | 53.46 |

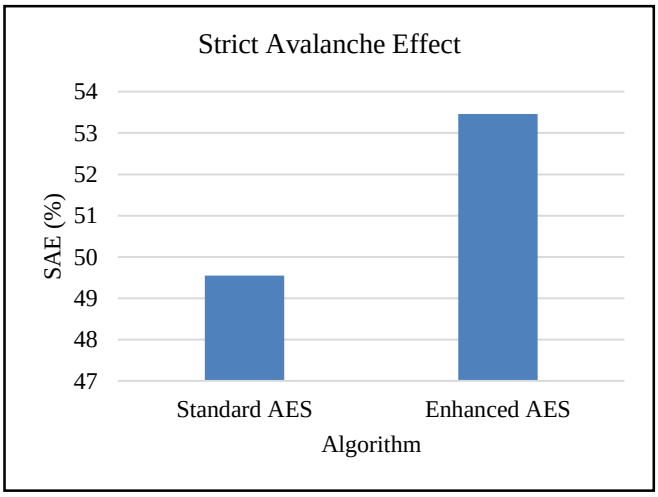


Fig. 10. SAE Analysis of standard versus enhanced AES.

### B. Correlation Coefficient

Table III and Fig. 11, present the correlation analysis between plaintext and ciphertext. When compared to the standard AES, which has a correlation coefficient of 0.004409, the enhanced encryption approach demonstrates a lower correlation coefficient of -0.000293. The observed negative and nearly zero value confirms that the proposed technique effectively diminishes the relationship between plaintext and ciphertext, thereby enhancing cryptographic strength.

TABLE III. Statistical Correlation Assessment Between Plaintext and Ciphertext

| Algorithm | Correlation Coefficient |
|---|---|
| Standard AES | 0.004409 |
| Enhanced AES | -0.000293 |

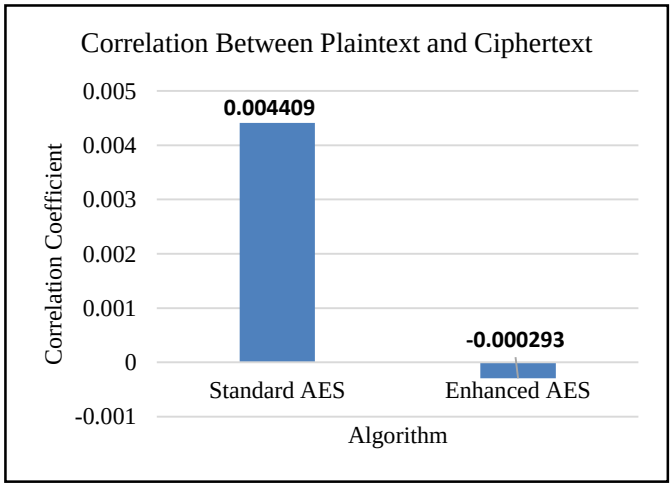


Fig. 11. Comparison of correlation coefficient between plaintext and ciphertext.

### C. Entropy

The results indicate that the existing and the redesigned AES encryption achieved high entropy levels in their ciphertexts, as shown in Table IV. The enhanced encryption process offers a slightly higher average entropy of 0.994619 bits than the standard AES's 0.994327 bits. These values, close to the ideal maximum of one bit per bit, show that both algorithms generate random ciphertext sequences.

TABLE IV. ENTROPY VALUES OF CIPHERTEXT

| Algorithm | Average Entropy |
|---|---|
| Standard AES | 0.994327 |
| Enhanced AES | 0.994619 |

### D. Execution Time

The method proposed in this study reduces encryption and decryption times for various file types, as illustrated in Table V and Fig. 12. It minimizes the encryption time by approximately 8.17% and the decryption time by 13.30% for a text file. A notable 32.96% reduction in encryption time and an 8.51% reduction in decryption time are observed for a PDF file. There is a 19.65% decrease in encryption time and a 16.17% decrease in decryption time for a JPEG file. An MP3 file shows a 20.44% reduction in encrypting time and an 11.95% reduction in decryption time. There is a 27.75% decrease in encryption time and a 20.16% decrease in decryption time for an MP4 file.

### E. Throughput

The suggested encryption method achieves higher throughput than standard AES. Its encryption throughput is 389.16 KB/sec, approximately 28.9% higher than the standard AES throughput of 301.85 KB/sec. Similarly, it improves the decryption throughput by approximately 18%, achieving 311.30 KB/sec, whereas the standard AES achieves 263.79 KB/sec. These results highlight the efficiency gains provided by the proposed approach, as shown in Table VI and Fig. 13.

TABLE V. EXECUTION TIME FOR ENCRYPTION AND DECRYPTION OF MULTIPLE FILE TYPES

| File Type & Size | Standard AES Encryption Time (sec) | Enhanced AES Encryption Time(sec) | Standard AES Decryption Time(sec) | Enhanced AES Decryption Time(sec) |
|---|---|---|---|---|
| Text File(1025KB) | 2.010364 | 1.846177 | 2.539962 | 2.202060 |
| PDF File(1028KB) | 1.482714 | 0.994051 | 2.207906 | 2.019965 |
| JPEG File(1126KB) | 3.441201 | 2.764880 | 3.466463 | 2.905801 |
| MP3 File(1026KB) | 4.473375 | 3.559055 | 4.905859 | 4.319511 |
| MP4 File(1031KB) | 5.938684 | 4.290501 | 6.728915 | 5.372506 |


Fig. 12. Comparison of execution time across multiple file formats.

TABLE VI. THROUGHPUT METRICS FOR ENCRYPTION AND DECRYPTION PROCESSES

| Algorithm | Encryption Throughput (KB/Sec) | Decryption Throughput (KB/Sec) |
|---|---|---|
| **Standard AES** | 301.85 | 263.79 |
| **Enhanced AES** | 389.16 | 311.30 |

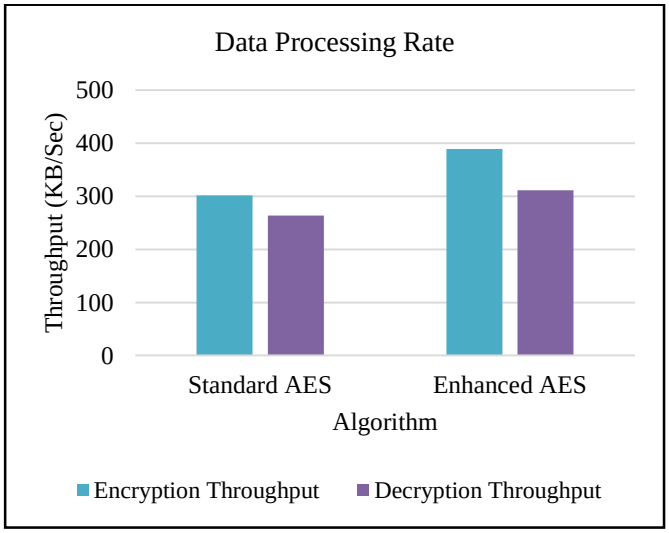


Fig. 13. Efficiency assessment based on throughput.

## VI. Discussion

The experimental results confirm that the proposed encryption approach has notable improvements over the standard AES across several metrics. The enhanced proposed scheme offers improved diffusion and security, achieving a higher SAE of 53.46% compared to 49.55% for the standard AES. Its correlation coefficient is lower at -0.000293, indicating reduced predictability between plaintexts and ciphertexts, while the standard AES records a value of 0.004409. Additionally, its average entropy increases slightly from 0.994327 to 0.994619, signifying improved randomness in the ciphertext. It outperforms standard AES, boosting encryption throughput from 301.85 KB/s to 389.16 KB/s and decryption from 263.79 KB/s to 311.30 KB/s. These improvements make the proposed encryption method more suitable for applications that require stronger security and fast processing.

## VII. Conclusion

The analysis indicates that the suggested encryption method provides substantial enhancements compared to the conventional AES. The proposed encryption method promotes the diffusion property and minimizes the predictable nature between plaintext and ciphertext, resulting in increased security. It also generates more random ciphertext and delivers better performance in terms of processing time and throughput. The enhanced design provides a well-balanced increase in efficiency and security.

Future work will focus on implementing a more secure hybrid encryption technique within cloud-based blockchain systems to enhance data protection, prevent tampering, ensure secure key management, and strengthen user authentication.

## References

[1] Z. A. Mohammed, H. Q. Gheni, Z. J. Hussein, and A. K. M. Al-Qurabat, "Advancing cloud image security via AES algorithm enhancement techniques," *Engineering Technology & Applied Science Research*, vol. 14, no. 1, pp. 12694–12701, Feb. 2024, doi: 10.48084/etasr.6601.

[2] S. Kumar and D. Kumar, "Securing of cloud storage data using hybrid AES-ECC cryptographic approach," Journal of Mobile Multimedia, Nov. 2022, doi: 10.13052/jmm1550-4646.1921.

[3] Z. A. Mohammed, H. Q. Gheni, Z. J. Hussein, and A. K. M. Al-Qurabat, "Advancing Cloud Image Security via AES Algorithm Enhancement Techniques," *Engineering Technology & Applied Science Research*, vol. 14, no. 1, pp. 12694–12701, Feb. 2024, doi: 10.48084/etasr.6601.

[4] M. Bedoui, H. Mestiri, B. Bouallegue, B. Hamdi, and M. Machhout, "An improvement of both security and reliability for AES implementations," Journal of King Saud University - Computer and Information Sciences, vol. 34, no. 10, pp. 9844–9851, Jan. 2022, doi: 10.1016/j.jksuci.2021.12.012.

[5] M. Y. Shakor, M. I. Khaleel, M. Safran, S. Alfarhood, and M. Zhu, "Dynamic AES Encryption and Blockchain Key Management: a novel solution for cloud data security," IEEE Access, vol. 12, pp. 26334–26343, Jan. 2024, doi: 10.1109/access.2024.3351119.

[6] N. Pronika and S. S. Tyagi, "Secure Data Storage in Cloud using Encryption Algorithm," 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), pp. 136–141, Feb. 2021, doi: 10.1109/icicv50876.2021.9388388.

[7] S. Urooj, S. Lata, S. Ahmad, S. Mehfuz, and S. Kalathil, "Cryptographic Data Security for Reliable Wireless Sensor Network," Alexandria Engineering Journal, vol. 72, pp. 37–50, Apr. 2023, doi: 10.1016/j.aej.2023.03.061.

[8] V. D. Parthasarathy and K. Viswalingam, "Healthcare Data Security in Cloud Storage Using Lightweight Symmetric Key Algorithm," *The International Arab Journal of Information Technology*, vol. 21, no. 1, Jan. 2024, doi: 10.34028/iajit/21/1/5.

[9] J. S. Baladhay and E. M. De Los Reyes, "AES-128 reduced-round permutation by replacing the MixColumns function," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 33, no. 3, p. 1641, Mar. 2024, doi: 10.11591/ijeecs.v33.i3.pp1641-1652.

[10] D. N. Hammod, "Modified Lightweight AES based on Replacement Table and Chaotic System," *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, Ankara, Turkey, 2022, pp. 1-5, doi: 10.1109/HORA55278.2022.9799804.

[11] M. Sirajuddin and B. S. Kumar, "Modified Rijndael Algorithm for Resource-Constrained IoT-Based Wireless Sensor Networks," *ICST Transactions on Scalable Information Systems*, Aug. 2023, doi: 10.4108/eetsis.2748.

[12] F. V. Wenceslao Jr, "Enhancing the Performance of the Advanced Encryption Standard (AES) Algorithm Using Multiple Substitution Boxes," *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 10, no. 3, Apr. 2022, doi: 10.17762/ijcnis.v10i3.3589.

[13] B. Al-Shargabi and M. A. F. Al-Husainy, "Multi-round encryption for COVID-19 data using the DNA key," *International Journal of Electrical and Computer Engineering*, vol. 12, no. 1, p. 478, Feb. 2022, doi: 10.11591/ijece.v12i1.pp478-488.

[14] H. M. Mohammad and A. A. Abdullah, "Enhancement process of AES: a lightweight cryptography algorithm-AES for constrained devices," TELKOMNIKA (Telecommunication Computing Electronics and Control), vol. 20, no. 3, p. 551, Jun. 2022, doi: 10.12928/telkomnika.v20i3.23297.

[15] A. Altigani, S. Hasan, B. Barry, S. Naserelden, M. A. Elsadig and H. T. Elshoush, "A Polymorphic Advanced Encryption Standard – A Novel Approach," in *IEEE Access*, vol. 9, pp. 20191-20207, 2021, doi: 10.1109/ACCESS.2021.3051556.

[16] A. I. Salih, A. M. Alabaichi, and A. Y. Tuama, "Enhancing advance encryption standard security based on dual dynamic XOR table and MixColumns transformation," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 19, no. 3, p. 1574, Sep. 2020, doi: 10.11591/ijeecs.v19.i3.pp1574-1581.

[17] H. T. Assafli and I. A. Hashim, "Generation and Evaluation of a New Time-Dependent Dynamic S-Box Algorithm for AES Block Cipher Cryptosystems," *IOP Conference Series Materials Science and Engineering*, vol. 978, p. 012042, Dec. 2020, doi: 10.1088/1757-899x/978/1/012042.

[18] O. C. Abikoye, A. D. Haruna, A. Abubakar, N. O. Akande, and E. O. Asani, "Modified Advanced Encryption Standard Algorithm for

Information Security," *Symmetry*, vol. 11, no. 12, p. 1484, Dec. 2019, doi: 10.3390/sym11121484.

[19] T. M. Kumar and P. Karthigaikumar, "FPGA implementation of an optimized key expansion module of AES algorithm for secure transmission of personal ECG signals," Design Automation for Embedded Systems, vol. 22, no. 1–2, pp. 13–24, Oct. 2017, doi: 10.1007/s10617-017-9189-5.

[20] M. Al-Mashhadani and M. Shujaa, "IoT Security Using AES Encryption Technology based ESP32 Platform," The International Arab Journal of Information Technology, vol. 19, no. 2, Jan. 2022, doi: 10.34028/iajit/19/2/8.

[21] M. Sawka and M. Niemiec, "A Sponge-Based Key Expansion Scheme for Modern Block Ciphers," *Energies*, vol. 15, no. 19, p. 6864, Sep. 2022, doi: 10.3390/en15196864.

[22] T. M. Kumar, K. R. Balmuri, A. Marchewka, P. B. Divakarachari, and S. Konda, "Implementation of Speed-Efficient Key-Scheduling Process of AES for Secure Storage and Transmission of Data," *Sensors*, vol. 21, no. 24, p. 8347, Dec. 2021, doi: 10.3390/s21248347.

[23] Y. S. Vaz, J. C. B. Mattos and R. I. Soares, "Improving an Ultra Lightweight AES for IoT Applications," *2023 IEEE 9th World Forum on Internet of Things (WF-IoT)*, Aveiro, Portugal, 2023, pp. 01-06, doi: 10.1109/WF-IoT58464.2023.10539597.

[24] A. J. Hintaw, S. Manickam, S. Karuppayah, M. A. Aladaileh, M. F. Aboalmaaly and S. U. A. Laghari, "A Robust Security Scheme Based on Enhanced Symmetric Algorithm for MQTT in the Internet of Things," in *IEEE Access*, vol. 11, pp. 43019-43040, 2023, doi: 10.1109/ACCESS.2023.3267718.

[25] S. Afzal, M. Yousaf, H. Afzal, N. Alharbe, and M. R. Mufti, "Cryptographic strength evaluation of key schedule algorithms," Security and Communication Networks, vol. 2020, pp. 1–9, May 2020, doi: 10.1155/2020/3189601.

[26] V. M. Silva-García, R. Flores-Carapia, and M. A. Cardona-López, "A Hybrid Cryptosystem Incorporating a New Algorithm for Improved Entropy," *Entropy*, vol. 26, no. 2, p. 154, Feb. 2024, doi: 10.3390/e26020154.

[27] G. Yi and Z. Cao, "An Algorithm of Image Encryption based on AES & Rossler Hyperchaotic Modeling," Mobile Networks and Applications, Sep. 2023, doi: 10.1007/s11036-023-02216-5.

[28] F. Thabit, S. Alhomdy, and S. Jagtap, "Security analysis and performance evaluation of a new lightweight cryptographic algorithm for cloud computing," Global Transitions Proceedings, vol. 2, no. 1, pp. 100–110, Jun. 2021, doi: 10.1016/j.gltp.2021.01.014.

[29] K. Assa-Agyei and F. Olajide, "A Comparative Study of Twofish, Blowfish, and Advanced Encryption Standard for Secured Data Transmission," International Journal of Advanced Computer Science and Applications, vol. 14, no. 3, pp. 393–398, 2023. doi: 10.14569/ijacsa.2023.0140344