

An improved key scheduling for advanced encryption standard with expanded round constants and non-linear property of cubic polynomials

Muthu Meenakshi Ganesan, Sabeen Selvaraj

Department of Computer Science, Faculty of Science and Humanities, SRM Institute of Science and Technology, Kattankulathur, India

Article Info

Article history:

Received May 28, 2024

Revised Oct 11, 2024

Accepted Oct 23, 2024

Keywords:

Advanced encryption standard
cloud computing

Cryptography

Cubic polynomials

Key schedule

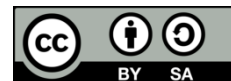
National institute of standards
and technology

Round constants

ABSTRACT

The advanced encryption standard (AES) offers strong symmetric key encryption, ensuring data security in cloud computing environments during transmission and storage. However, its key scheduling algorithm is known to have flaws, including vulnerabilities to related-key attacks, inadequate nonlinearity, less complicated key expansion, and possible side-channel attack susceptibilities. This study aims to strengthen the independence among round keys generated by the key expansion process of AES—that is, the value of one round key does not reveal anything about the value of another round key—by improving the key scheduling process. Data sets of random, low, and high-density initial secret keys were used to evaluate the strength of the improved key scheduling algorithm through the National Institute of Standards and Technology (NIST) frequency test, the avalanche effect, and the Hamming distance between two consecutive round keys. A related-key analysis was performed to assess the robustness of the proposed key scheduling algorithm, revealing improved resistance to key-related cryptanalysis.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Sabeen Selvaraj

Department of Computer Science, Faculty of Science and Humanities, SRM Institute of Science and Technology

Kattankulathur, Chengalpattu District, Tamil Nadu, 603203, India

Email: sabeens@srmist.edu.in

1. INTRODUCTION

National Institute of Standards and Technology (NIST) launched the advanced encryption standard (AES) competition to search for a better secure cryptographic algorithm. Vincent Rijmen and Joan Daemen developed the Rijndael algorithm. Rijndael evolved into AES following NIST's 2001 declaration of the winner [1], [2]. Because of its robust security, effectiveness, and adaptability in safeguarding data while it's in transit and at rest, it is extensively utilized in cloud computing [3], [4]. It is a symmetric block cipher that guarantees trustworthy and efficient information security techniques by supporting 16 bytes data block sizes and key lengths of 16, 24, and 32 bytes [5]. The three essential components of AES are key expansion, decryption, and encryption. An XOR operation is performed at each encryption round operation between the state array of data and the round key obtained during the key expansion procedure to incorporate randomness and diffusion [6]–[8].

A well-designed key scheduling algorithm (KSA) can prevent computational guessing of the plaintext or key. Despite the difficulty of executing brute force attacks with larger keys, maintaining security in the key expansion process requires sticking to the concepts of confusion and diffusion [9], [10]. Encryption garners more research focus due to its direct impact on data security, while key expansion, vital

for algorithms like AES, is considered an auxiliary component [11], [12]. Finite nonlinearity in the AES key schedule is the source of AES key expansion weaknesses, including related key attacks. Due to these flaws, there are significant security vulnerabilities, as adversaries can recover keys, exploit slow diffusion, and manipulate subkeys [13], [14]. To increase the bit transition between subkeys, KSA should generate subkeys that are independent of one another and random. Consequently, this study aims to improve the overall security of the AES encryption technique by designing a new and enhanced version of AES KSA.

Many researchers have conducted extensive research to improve the KSA's efficiency and randomness across various encryption algorithms. Hammod *et al.* [15] suggest an improved approach to the AES KSA using modified cipher feedback (MCFB) mode. It does this by implementing two processes: shift rows and substitution bytes, which reduce complexity and increase speed, efficiency, and performance for a range of key lengths. Reyes *et al.* [16] used simple operations like XOR and modulo arithmetic to modify the AES cipher round and KSA to fix low diffusion rates in early rounds. They improved the KSA by adding byte substitution and round constant addition. In rounds 1 and 3, the modified AES increased diffusion rate and improved encryption output randomness. Pehlivanoglu *et al.* [17] explore block ciphers and their key schedule algorithm, inspired by AES, with desirable properties like good avalanche effect and bit confusion. Similarly, Cao *et al.* [18] optimize the AES KSA using three improvement strategies: irreversible improvement, word shift, and random number strategy, to reduce round-key correlation, improve security, and ensure efficient operation. Kumar *et al.* [19] proposed and simulated a new subkey generation algorithm for AES on the FPGA Virtex 5 XC5VLX50T, enhancing its speed, maintaining word diffusion, and minimizing time consumption.

De Leon *et al.* [20] modified the tiny encryption algorithm (TEA), a lightweight encryption method, to improve security by rotating subkeys and shifting keys, outperforming the original TEA. By adding a salting algorithm to the subkey, Galas and Gerardo [21] enhanced the security of the corrected block tiny encryption algorithm, XXTEA, and improved its randomness and avalanche effect. This approach was more effective than the original approach, which failed the frequency test. The key expansion process of PRESENT-128 is enhanced by Imdad *et al.* [22] with improved randomness, avalanche effect, and Hamming distance between round keys through experimental tests with random, low, and high-density initial secret keys. Zakaria *et al.* [23] improved the RECTANGLE key schedule algorithm by increasing randomization and confusion properties, speed, and throughput.

This article arranges its sections as follows, section 2 provides a comprehensive description of the standard and improved AES key expansion procedures. This section also describes the statistical tests and key expansion process assessment parameters to evaluate the robustness of the standard and improved AES key expansion algorithms. Section 4 concludes with the findings and discussions from section 3.

2. METHOD

2.1. Standard AES KSA

This article considers AES-128 KSA and Figure 1 depicts its detailed key scheduling process. The round-key generation process works at the word level (32 bits). So, the procedure starts by dividing the initial secret key of length 128 bits into four words (W_0, W_1, W_2, W_3). The first four words of the key schedule are the same as the four words of the initial secret key. KSA derives the remaining 40 words iteratively through a sequence of transformations, as AES-128 encryption and decryption necessitate the generation of 10 round keys from the initial secret key. These 40 words are further divided into 10 round keys. In (1)–(10) generate the words $W_3', W_7', W_{11}', \dots, W_{39}'$ [24].

$$W_3' = \text{SubWord}(\text{RotateWord}(W_3)) \oplus \text{Round_constants} \quad (1)$$

$$W_7' = \text{SubWord}(\text{RotateWord}(W_7)) \oplus \text{Round_constants} \quad (2)$$

$$W_{11}' = \text{SubWord}(\text{RotateWord}(W_{11})) \oplus \text{Round_constants} \quad (3)$$

$$\dots \dots \dots \dots \dots \dots \dots \quad (4)-(9)$$

$$W_{39}' = \text{SubWord}(\text{RotateWord}(W_{39})) \oplus \text{Round_constants} \quad (10)$$

where *RotateWord* is the circular left shift of one byte, *SubWord* is the substitution method using a built-in $16 \times 16 S - \text{Box}$ and \oplus is the XOR operation and *Round_constants* are in the form of $(RC_j, 00, 00, 00)$ as shown in the Table 1. The subsequent words are generated by simple XOR operation, as follows:

$W_4=W_3' \oplus W_0$, $W_5=W_4 \oplus W_1$, $W_6=W_5 \oplus W_2$, $W_7=W_6 \oplus W_3$, $W_8=W_7' \oplus W_4$, $W_9=W_8 \oplus W_5$,
 $W_{10}=W_9 \oplus W_6$, $W_{11}=W_{10} \oplus W_7$, $W_{12}=W_{11}' \oplus W_8$, $W_{13}=W_{12} \oplus W_9$, $W_{14}=W_{13} \oplus W_{10}$, $W_{15}=W_{14}$
 $\oplus W_{11}$, $W_{16}=W_{15}' \oplus W_{12}$, $W_{17}=W_{16} \oplus W_{13}$, $W_{18}=W_{17} \oplus W_{14}$, $W_{19}=W_{18} \oplus W_{15}$, $W_{20}=W_{19}' \oplus$
 W_{16} , $W_{21}=W_{20} \oplus W_{17}$, $W_{22}=W_{21} \oplus W_{18}$, $W_{23}=W_{22} \oplus W_{19}$, $W_{24}=W_{23}' \oplus W_{20}$, $W_{25}=W_{24} \oplus W_{21}$,
 $W_{26}=W_{25} \oplus W_{22}$, $W_{27}=W_{26} \oplus W_{23}$, $W_{28}=W_{27}' \oplus W_{24}$, $W_{29}=W_{28} \oplus W_{25}$, $W_{30}=W_{29} \oplus W_{26}$,
 $W_{31}=W_{30} \oplus W_{27}$, $W_{32}=W_{31}' \oplus W_{28}$, $W_{33}=W_{32} \oplus W_{29}$, $W_{34}=W_{33} \oplus W_{30}$, $W_{35}=W_{34} \oplus W_{31}$,
 $W_{36}=W_{35}' \oplus W_{32}$, $W_{37}=W_{36} \oplus W_{33}$, $W_{38}=W_{37} \oplus W_{34}$, $W_{39}=W_{38} \oplus W_{35}$, $W_{40}=W_{39}' \oplus W_{36}$,
 $W_{41}=W_{40} \oplus W_{37}$, $W_{42}=W_{41} \oplus W_{38}$, $W_{43}=W_{42} \oplus W_{39}$.

Both the encryption and decryption processes will use these 44 words from the AES 128 key schedule.

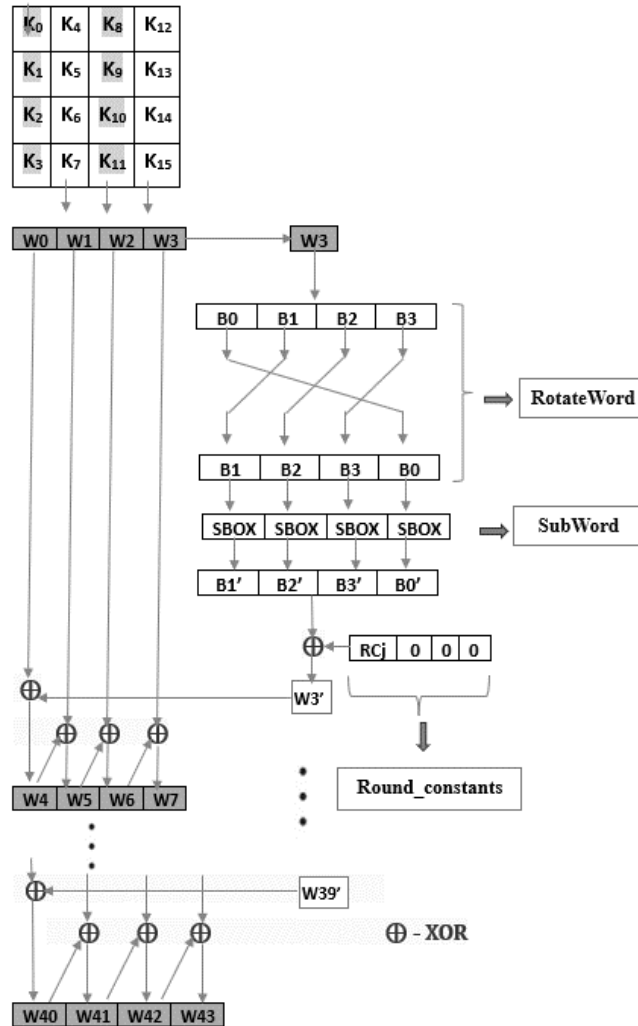


Figure 1. AES key expansion process

Table 1. Round constants in standard AES KSA

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

Pseudocode of standard AES KSA

```

ASE128KeyExpansion (byte initial_secretkey [16], word w [44])
{
    word tmp_word;
    Round_constants=[0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x04, 0x80, 0x1b, 0x36]
    for (x=0; x<4; x++)
        w[x]=(initial_secretkey[4*x], initial_secretkey [4*x+1], initial_secretkey [4*x+2],
            initial_secretkey [4*x+3]);
    for (x=4; x<44; x++)
    {
        tmp_word=w[x-1];
        if (x mod 4=0)
            tmp_word=SubWord (RotateWord(tmp_word)) ⊕ Round_constants [x/4];
        w[x]=tmp_word ⊕ w[x-4];
    }
}
    
```

2.2. Enhanced AES KSA

2.2.1. Enhanced key expansion using S-Box based expanded round constants

The AES key scheduling technique uses an implementation of cyclic rotation, S-box, and XOR with round constants to find the temporary words ($W_3', W_7', W_{11}', \dots, W_{39}'$). All the remaining round keys can be produced from the original key using these temporary variables. However, the round constants ($RC_j, 0, 0, 0$) in AES leave three bytes as zeros, as shown in Figure 1, which lessens the amount of confusion and diffusion in the round key generation process. Attackers can exploit chosen, known, and related key assaults because of this flaw. To put it another way, XORing with zero doesn't create more confusion, which makes it easier for adversaries to deduce parts of the key. As shown in Figure 2, the stretched round constants ($RC_j, S\text{-Box}[RC_j], S\text{-Box}[S\text{-Box}[RC_j]], S\text{-Box}[S\text{-Box}[S\text{-Box}[RC_j]]]$) are used in place of the round constants ($RC_j, 0, 0, 0$) in the proposed AES KSA. The expanded round constants using S-Box are given in Table 2. These expanded round constants are generated by applying S-Box on the round constants RC_j iteratively.

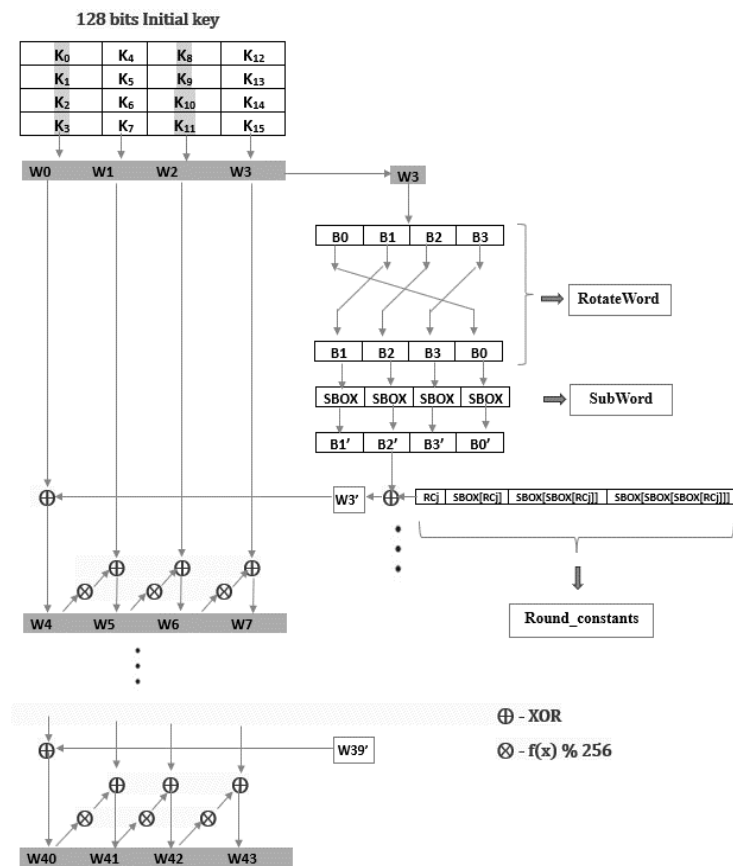


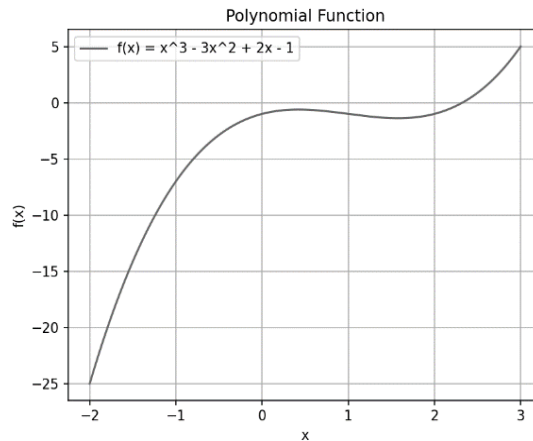
Figure 2. Enhanced AES key expansion process

Table 2. Expanded round constants using S-Box

Round(j)	1	2	3	4	5
Round_constants	0x017c10ca	0x0277f5e6	0x04f289a7	0x083004f2	0x10ca7492
Round(j)	6	7	8	9	10
Round_constants	0x20b7a9d3	0x4009017c	0x80cdbd7a	0x1baf79b6	0x36056b7f

2.2.2. Enhanced key expansion using cubic polynomial function

AES KSA uses circular left shift of one byte, S-box substitutions, and XOR operation with round constants. However, round keys still show some level of correlation, because each round key is generated sequentially, with each subsequent round key being derived from the previous one. This operation is performed word-wise, meaning that each corresponding word from the previously generated round key is XORed with the current word to generate the next word. For instance, W5 (6th word) is generated from W4 and W1, W6 (7th word) from W5 and W2, and W7 (8th word) from W6 and W3 and so on. So, AES needs improvements to reduce correlations between round keys, which makes the key schedule stronger. Such improvements can consist of refining the derivation techniques, which include more nonlinear processes. To accomplish this, a cubic polynomial function is used in the proposed KSA to introduce chaos between round keys. The generic form of this type of polynomial is $f(x) = mx^3 + nx^2 + ox + p$, where m does not equal zero. The behavior of these curves is determined by the values of the real coefficients (m , n , o , and p). A cubic polynomial function, for example, has coefficients $m=1$, $n=-3$, $o=2$, and $p=-1$ as shown in Figure 3. The enhanced AES KSA uses these coefficient values because it improves the complexity of the key expansion process. These functions can be employed as a powerful substitution technique to improve the diffusion and confusion properties of the round key generation mechanisms used in block ciphers.

Figure 3. Graphical representation of $f(x)$

In AES KSA, for each word generation, each corresponding word from the previously generated round key is XORed with the current word to generate the next word. The proposed KSA applies the $f(x)$ mod 256 (\otimes) operation on each current word which directly participates in the XOR operation with the corresponding word from the previous key. For instance, the word W5 is generated equal to $(f(W4) \text{ mod } 256 \oplus W1)$ instead of $W4 \oplus W1$. In the proposed KSA, the key schedule is generated as follows:

$W4 = W3' \oplus W0$, $W5 = f(W4) \text{ mod } 256 \oplus W1$, $W6 = f(W5) \text{ mod } 256 \oplus W2$, $W7 = f(W6) \text{ mod } 256 \oplus W3$,
 $W8 = W7' \oplus W4$, $W9 = f(W8) \text{ mod } 256 \oplus W5$, $W10 = f(W9) \text{ mod } 256 \oplus W6$, $W11 = f(W10) \text{ mod } 256 \oplus W7$,
 $W12 = W11' \oplus W8$, $W13 = f(W12) \text{ mod } 256 \oplus W9$, $W14 = f(W13) \text{ mod } 256 \oplus W10$, $W15 = f(W14) \text{ mod } 256$
 $\oplus W11$, $W16 = W15' \oplus W12$, $W17 = f(W16) \text{ mod } 256 \oplus W13$, $W18 = f(W17) \text{ mod } 256 \oplus W14$, $W19 = f(W18)$
 $\text{mod } 256 \oplus W15$, $W20 = W19' \oplus W16$, $W21 = f(W20) \text{ mod } 256 \oplus W17$, $W22 = f(W21) \text{ mod } 256 \oplus W18$,
 $W23 = f(W22) \text{ mod } 256 \oplus W19$, $W24 = W23' \oplus W20$, $W25 = f(W24) \text{ mod } 256 \oplus W21$, $W26 = f(W25) \text{ mod } 256$
 $\oplus W22$, $W27 = f(W26) \text{ mod } 256 \oplus W23$, $W28 = W27' \oplus W24$, $W29 = f(W28) \text{ mod } 256 \oplus W25$, $W30 = f(W29)$
 $\text{mod } 256 \oplus W26$, $W31 = f(W30) \text{ mod } 256 \oplus W27$, $W32 = W31' \oplus W28$, $W33 = f(W32) \text{ mod } 256 \oplus W29$,
 $W34 = f(W33) \text{ mod } 256 \oplus W30$, $W35 = f(W34) \text{ mod } 256 \oplus W31$, $W36 = W35' \oplus W32$, $W37 = f(W36) \text{ mod } 256$
 $\oplus W33$, $W38 = f(W37) \text{ mod } 256 \oplus W34$, $W39 = f(W38) \text{ mod } 256 \oplus W35$, $W40 = W39' \oplus W36$, $W41 = f(W40)$
 $\text{mod } 256 \oplus W37$, $W42 = W41 \oplus W38$, $W43 = f(W42) \text{ mod } 256 \oplus W39$.

The output of the word to be XORed with corresponding words from the previous key is determined by the cubic polynomial function's modulus of 256. Once it has generated all the words or round keys, it completes the key scheduling process. As shown in Figure 2, the function $(f(x) \bmod 256)$ is denoted by the symbol \otimes . The relationship between the input (x) of the operation (\otimes) and its output is shown in Figure 4. A complex relationship can be seen by analyzing the backtracking of the input (x) value from the $f(x) \bmod 256$ (\otimes) operation output, which has several bends and oscillations as illustrated in Figure 4. There are irregular shifts in the outputs due to the nonlinear relationship between inputs (x) and the outputs of \otimes ; this reduces the correlation between words and round keys in the key schedule. Due to this non-linear behavior, attackers often find it challenging to infer the input key from the round keys, as the same output may not always correspond to the same input in the operation. It depends on the real coefficient values used in the cubic polynomial function. The backtracking process is more intricate and unpredictable, which makes the function more resilient to cryptographic attacks.

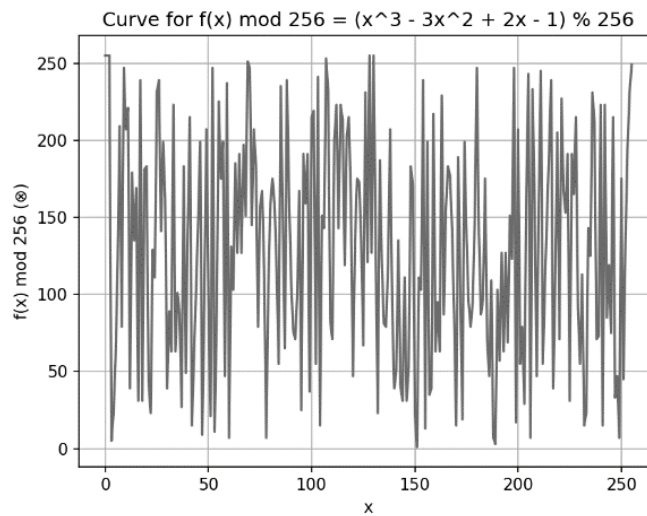


Figure 4. Relationship between input (x) and \otimes operation

Pseudocode of enhanced AES KSA

```
EnhancedAES128KeyExpansion (byte initial_secretkey [16], word w [44], int m, int n, int o,
int p)
{
    word tmp_word;
    Round_constants=(0x017c10ca, 0x0277f5e6, 0x04f289a7, 0x083004f2, 0x10ca7492, 0x20b7a9d3,
0x4009017c, 0x80cdbc7a, 0x1baf79b6, 0x36056b7f);
    for (x=0; x<4; x++)
    w[x]=(initial_secretkey [4*x], initial_secretkey [4*x+1], initial_secretkey [4*x+2],
initial_secretkey [4*x+3]);
    for (x=4; x<44; x++)
    {
        tmp_word=w[x-1];
        if (x mod 4=0)
        {
            tmp_word =SubWord (RotateWord(tmp_word))  $\oplus$  Round_constants[x/4];
            w[x]=tmp_word  $\oplus$  w[x-4];
        }
        else
        {
            for (y=0; y<4; y++)
            {
                tmp_word[y]=(m*tmp_word[y]^3+n*tmp_word[y]^2+o*tmp_word[y]+p) % 256
            }
            tmp_word=[tmp_word [0], tmp_word [1], tmp_word [2], tmp_word [3]];
            w[k]=tmp_word  $\oplus$  w[k-4];
        }
    }
}
```

2.3. Evaluation

This section discusses the evaluation method to measure the strength of the enhanced AES KSA. High density, low density, and random density input key sets were used for the evaluation process. The high-density key (HDK), with its large number of '1' bits and a small number of '0' bits (at most two), poses a challenge to algorithms with extreme biases towards the '1' bits. On the other hand, the low-density key (LDK), with its large number of '0' bits and a small number of '1' bits (at most two), poses a challenge to algorithms with extreme biases towards the '0' bits. Random density keys (RDK) consist of random sequences of '0' and '1' bits, serving as a baseline for more typical key distributions. The following tests were used to evaluate the proposed and standard AES KSAs: frequency test, avalanche effect, bit difference between successive subkeys, and related key analysis.

2.3.1. Frequency test

The indeterminacy of round keys generated from KSA in block ciphers can be evaluated using the frequency test, which is intended to evaluate the randomness of random number generators (RNGs). By assessing the distribution of '0' and '1' bits in the binary sequence of round keys, the test determines if it displays the essential randomness for secure encryption operations [25]. Passing the test indicates an equal distribution of 0s and 1s, revealing the strength of the KSA; failure implies potential bias or non-randomness. To conduct the frequency test, the following steps a through step d are followed:

- a. Converting the binary sequence pattern (ϵ) into ± 1 : This procedure converts the sequence into values -1 and +1. The formula $X_i = 2\epsilon_i - 1$ represents a conversion of this type. That is, if $\epsilon_i = 0$, then $X_i = -1$, and if $\epsilon_i = 1$, then $X_i = 1$.
- b. The overall computation of Sum (S_n):

$$X_1 + X_2 + \dots + X_n = S_n$$

where n is the total number of bits.

- c. Determine the test statistic estimator (S_{obs}):

$$S_{obs} = \frac{|S_n|}{\sqrt{n}}$$

- d. P-value assessment: The p-value is computed and investigated using the complementary error function (*erfc*) as given below.

$$p - value = \text{erfc}\left(\frac{S_{obs}}{\sqrt{2}}\right) \quad (11)$$

To evaluate the strength of both existing and proposed KSA, 10,000 initial random secret keys were generated and stored in a single file. Using these initial secret keys 10 round keys were generated and stored in 10 different files for existing and proposed KSAs. Every file is examined separately. The outcomes of the proposed and existing ones were compared. The frequency test evaluates the randomness of a bit sequence by calculating the p-value using (11) and analyzing the outcomes. If the p-value is more than or equal to 0.01, a sequence is considered pseudo-random; if not, it is considered non-random [25].

2.3.2. Avalanche effect of round keys

In cryptography, the phenomenon known as the "avalanche effect (AE)" occurs when an alteration of a single bit in the input (plaintext in encryption or secret key in KSA) causes a noticeably different output (ciphertext in encryption or round keys in KSA). The AE test for the KSA can be carried out by comparing two key schedules generated before and after complementing one bit of the original secret key [26]–[28]. To do this, it is necessary to find the Hamming distance (number of bits flipped) between the two Key schedules, generated before flipping any bits of the original secret key and after flipping a single bit of the original secret key—the next bit of the original key changes with each iteration to determine the AE. The computation of the average AE for the existing and proposed KSA is given below.

$$\text{Avalanche Effect} = \frac{\text{Number of bits flipped} \times 100}{\text{Total number of bits}} \quad (12)$$

2.3.3. Bit difference between round keys

The Hamming distance between two successive subkeys was calculated using the XOR function in this test to assess their correlation. The statistical relationship between the round keys becomes extremely

complicated when the reliable KSA offers a bit difference of more than 50% [22], [27]. To evaluate this, 100 RDK, 100 LDK, and 100 HDK were used. Since the AES KSA algorithm generates 10 round keys from an initial secret key, in each round, the keys generated from the existing and proposed algorithms are also written to separate files, and these 10 round keys, including the initial secret key, *i.e.*, 11 separate files for both algorithms, were taken for testing. The Hamming distance between two successive file contents was determined iteratively. This test was performed for both existing and proposed KSA with all three types of keys, and average Hamming distances between round keys were used to measure the correlation between round keys.

2.3.4. Related key analysis

The related key analysis identifies key schedule vulnerabilities and evaluates cryptographic protection. It prevents key recovery by attackers and enhances cryptographic design by exposing weaknesses. To perform related key analysis, given an original key K and plaintext X with a predetermined difference Δ , the related key K' is generated as $K' = K \oplus \Delta$ and the related plaintext X' is generated as $X' = X \oplus \Delta$, where \oplus denotes the bitwise XOR operation. To find the ciphertext C , encrypt the plaintext X with the key K using $C = E_K(X)$. For the related ciphertext C' , encrypt the related plaintext X' with the related key K' using $C' = E_{K'}(X')$. If the output difference distribution ($C \oplus C'$) is not uniform, an attacker could use this irregularity to deduce the key for encrypting messages with the input difference between input differences ($X \oplus X'$). The Hamming distance d_H measures the bit difference between C and C' [29], [30]. The mean Hamming distance (μ_d) between ciphertexts is calculated as (13):

$$\mu_d = \frac{1}{N} \sum_{i=1}^N d_H(C_i, C'_i) \quad (13)$$

where N is the number of bytes in $C \oplus C'$. The variance of the Hamming distance (σ_d^2) is calculated as the average of the squared differences between each Hamming distance, $d_H(C_i, C'_i)$ and the mean Hamming distance μ_d , given by (14):

$$\sigma_d^2 = \frac{1}{N} \sum_{i=1}^N (d_H(C_i, C'_i) - \mu_d)^2 \quad (14)$$

The variance of the Hamming distance is used to assess the uniformity in the distribution of the output differences $C \oplus C'$. Low variance indicates consistent and uniform behavior, reducing the likelihood of exploitable patterns in related key analysis. To find the mean Hamming distance between ciphertexts and related ciphertexts, 1000 random keys and plaintexts were used. The mean Hamming distance was calculated by averaging the Hamming distances between N bytes of ciphertexts C_i and their corresponding related ciphertexts C'_i using (13) and its variance was calculated using (14). The overall mean Hamming distance and the variance were calculated by taking the average of individual Hamming distances and their variances respectively from 1000 different trials.

2.3.5. Execution time

Execution time is a crucial factor in evaluating the efficiency of an algorithm, as faster execution leads to improved user experience and more efficient resource utilization. To get the execution time, one must record the start and completion times of the algorithm and subtract the start from the completion time [31]. The execution time for both KSAs was calculated and compared in this way.

3. RESULTS AND DISCUSSION

3.1. Frequency test

Table 3 displays the frequency test p-values for 10 round keys generated by the standard AES KSA and the enhanced AES KSA. The standard version of AES KSA has an average p-value of 0.510392174, but the improved AES KSA has an average p-value of 0.529139941. The average p-values are greater than 0.01 for the round key sequences generated by the standard AES KSA and Enhanced AES KSA passes the frequency test. So, the test results reveal that the subkey sequences generated by both methods have nearly identical ratios of 1 to 0 s.

3.2. Avalanche effect of round keys

Three secret keys—one for each type—RDK, LDK, and HDK—were used to test the AE of both methods. These results were obtained by applying formula (12). The subsequent bit of the initial key is complemented with each iteration to produce a new AE value. In this way, the AE for each round key is

calculated and the average values of test results are given in Table 4. As per the results of this study shown in Figure 5, for the enhanced AES KSA, the average values of AE of three types of initial secret keys are RDK-36.74%, LDK-35.81%, and HDK-32.09%. Meanwhile, the standard AES KSA has these results: RDK-34.23%, LDK-33.16%, and HDK-29.14%. In all rounds, these test results indicate that the enhanced AES KSA can provide better diffusion (avalanche effect) than the standard AES KSA. As a result, the improved KSA is better at spreading information, with average avalanche effects of 35%, whereas the standard AES KSA has 32%. The improved AES KSA does not affect the encryption method's avalanche effect as shown in Table 5, which remains almost constant at 49.90% compared to the original's 49.55%.

Table 3. P-value of frequency test

Round No.	p-value of Standard AES KSA	p-value of improved AES KSA
1	0.699960939	0.085747474
2	0.270757927	0.873590856
3	0.386377309	0.908520539
4	0.569206199	0.780009098
5	0.195663552	0.735631866
6	0.170129707	0.702581781
7	0.611909882	0.007640417
8	0.59221195	0.816866208
9	0.711782429	0.330036882
10	0.895921851	0.05077429
Average p-value	0.510392174	0.529139941

Table 4. Average avalanche effect of round keys

Round No.	RDK		LDK		HDK	
	Standard AES KSA	Enhanced AES KSA	Standard AES KSA	Enhanced AES KSA	Standard AES KSA	Enhanced AES KSA
1	10.30273438	10.58959961	5.859375	7.574462891	4.680461712	4.983108108
2	20.82519531	19.81201172	18.33496094	16.00952148	15.72353604	14.78744369
3	30.28564453	29.06494141	29.16259766	28.49121094	24.23986486	21.80461712
4	37.72583008	37.24365234	38.03100586	36.18774414	32.43947072	33.28406532
5	41.07666016	43.27392578	39.87426758	43.64624023	37.15512387	37.76745495
6	40.34423828	44.39697266	40.46630859	45.03173828	36.00788288	40.5053491
7	39.16015625	45.73364258	40.30151367	45.13549805	36.00788288	40.625
8	41.20483398	45.75805664	40.00244141	45.703125	36.04307432	42.07488739
9	40.27099609	45.75195313	39.58129883	45.35522461	35.11402027	42.0678491
10	41.12548828	45.77026367	39.9597168	44.95239258	34.03716216	43.00394144
Average	34.23217773	36.73950195	33.15734863	35.80871582	29.14484797	32.09037162

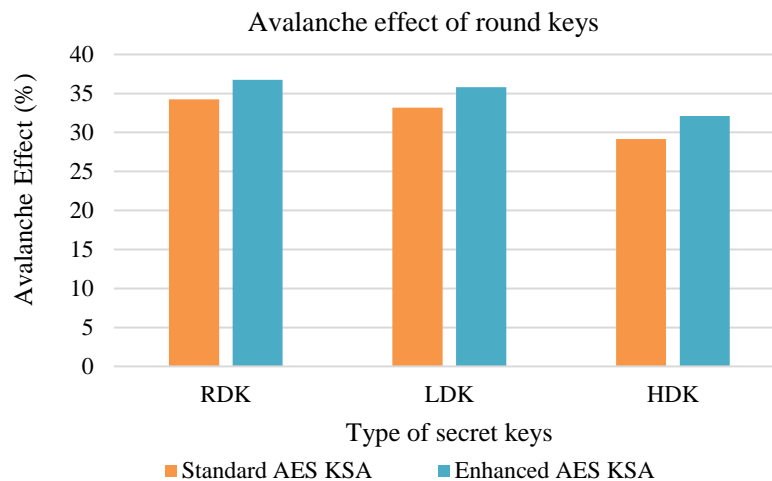


Figure 5. Avalanche effect of round keys

Table 5. Average avalanche effect of cipher text

Standard AES KSA	Enhanced AES KSA
49.55%	49.90%

3.3. Bit difference between round keys

Tables 6 and 7 provide the Hamming distance between two successive round keys and the percentage of the Hamming distances between two subkeys for standard AES KSA and enhanced AES KSA, respectively. Based on these test results as shown in Figure 6, the overall average values of bit difference between subkeys for enhanced AES KSA are RDK-59.28%, LDK-59.87%, and HDK-58.60%. Meanwhile, the standard AES KSA has these results: RDK-49.91%, LDK-50.16%, and HDK-50.60%. Results show that the enhanced version of AES KSA consistently yields higher bit differences between two successive subkeys, suggesting potentially enhanced security through greater variability in generated subkeys.

Table 6. Bit difference between round keys of standard AES KSA

Round No.(i) k(i) ⊕ k(i+1)	RDK		LDK		HDK	
	Hamming distance	% of Bit diff.	Hamming distance	% of Bit diff.	Hamming distance	% of Bit diff.
k1⊕k2	6303	49.2421875	6055	47.3046875	7184	56.125
k2⊕k3	6408	50.0625	7417	57.9453125	6405	50.0390625
k3⊕k4	6447	50.3671875	6493	50.7265625	6264	48.9375
k4⊕k5	6436	50.28125	6549	51.1640625	6394	49.953125
k5⊕k6	6393	49.9453125	6365	49.7265625	6457	50.4453125
k6⊕k7	6331	49.4609375	6208	48.5	6057	47.3203125
k7⊕k8	6348	49.59375	6083	47.5234375	6486	50.671875
k8⊕k9	6412	50.09375	6115	47.7734375	6519	50.9296875
k9⊕k10	6420	50.15625	6750	52.734375	6988	54.59375
k10⊕k11	6381	49.8515625	6175	48.2421875	6010	46.953125
Average	6387.9	49.90546875	6421	50.1640625	6476.4	50.596875

Table 7. Bit difference between round keys of enhanced AES KSA

Round No.(i) k(i) ⊕ k(i+1)	RDK		LDK		HDK	
	Hamming Distance	% of Bit Diff.	Hamming Distance	% of Bit Diff.	Hamming Distance	% of Bit Diff.
k1⊕k2	7599	59.3671875	7599	59.3671875	7496	58.5625
k2⊕k3	7651	59.7734375	7961	62.1953125	6810	53.203125
k3⊕k4	7602	59.390625	7349	57.4140625	7717	60.2890625
k4⊕k5	7598	59.359375	8164	63.78125	7249	56.6328125
k5⊕k6	7639	59.6796875	7272	56.8125	8097	63.2578125
k6⊕k7	7665	59.8828125	8030	62.734375	7124	55.65625
k7⊕k8	7506	58.640625	7099	55.4609375	8028	62.71875
k8⊕k9	7532	58.84375	7931	61.9609375	7274	56.828125
k9⊕k10	7535	58.8671875	7245	56.6015625	8082	63.140625
k10⊕k11	7556	59.03125	7988	62.40625	7135	55.7421875
Average	7588.3	59.28359375	7663.8	59.8734375	7501.2	58.603125

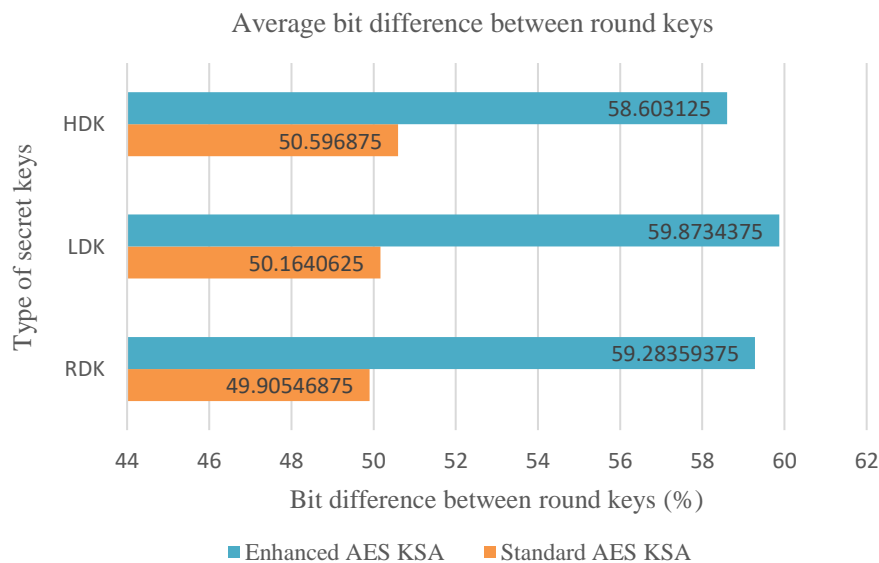


Figure 6. Average bit difference between round keys

3.4. Related key analysis

The related key analysis, shown in Table 8 and illustrated in Figure 7, reveals that the enhanced AES key scheduling provides stronger security than the standard version. The enhanced key schedule achieves a higher byte-level mean Hamming distance of 4.1837, indicating greater separation between related ciphertexts, compared to 3.9198 for the standard AES. Also, the enhanced AES exhibits a lower mean Hamming distance variance of 1.686 versus 2.0865 for the standard key schedule, reflecting more uniform behavior. The reduced variance ensures consistent output differences, minimizing exploitable patterns. In conclusion, the enhanced AES key scheduling boosts security and resists related key attacks.

Table 8. Related key analysis based on μ_d and σ_d^2

Related key analysis measures	AES with enhanced key scheduling	AES with standard key scheduling
Byte-level mean Hamming distance	4.1837	3.9198
Variance of mean Hamming distance	1.686	2.0865

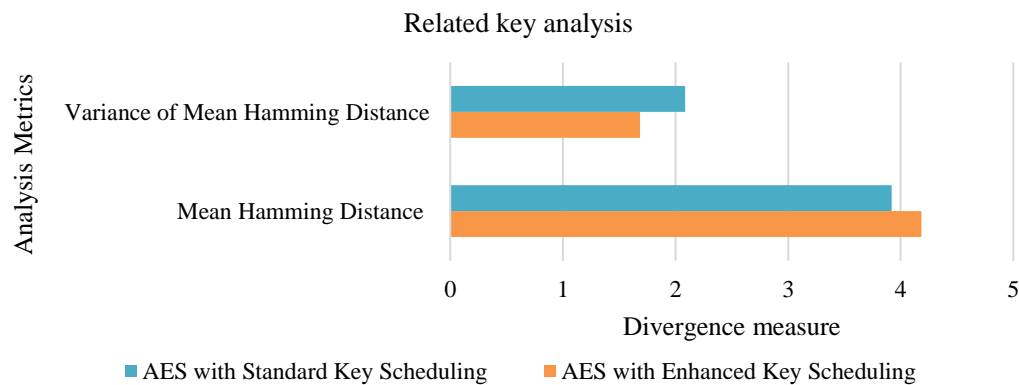


Figure 7. Related key analysis

3.5. Execution time

Both the standard and improved AES KSAs were implemented in Python and executed on an Intel (R) Core™ i7-11255U, 4.7 GHz CPU, to measure the execution time for comparison. Both algorithms were executed 150 times with different keys, and the execution times were measured for each run and then averaged. As a result, the improved AES KSA requires 0.0008364 s to generate 10 round keys whereas the standard AES KSA requires 0.0007690 s. The proposed KSA was improved using powerful nonlinear substitution techniques without significantly increasing the execution time. The enhanced KSA's security improvements for the avalanche effect and non-correlated round key generation negate the insignificant time difference of 0.0000674 s.

4. CONCLUSION





The round keys generated by the modified KSA are more random and better uncorrelated due to expanded round constants and robust nonlinear cubic polynomials with modulus operations. Both standard and improved KSA show similar NIST frequency test p-values, indicating nearly identical 1 to 0 s ratios in their subkey sequences. The improved KSA shows better performance with an average avalanche effect of 35% (vs. 32% for standard AES KSA) and an average bit difference between round keys of 59% (vs. 50% for standard AES KSA), enhancing round-key independence. Additionally, the related key analysis, indicates that AES with enhanced key scheduling has a Byte-level mean Hamming distance (4.1837 vs. 3.9198) and its lower variance (1.686 vs. 2.0865), demonstrating improved resistance and consistency against related key attacks. However, as a security cost trade-off, creating round keys adds 0.0000674s to the overall processing time. Future work will improve security and speed efficiency in cloud data storage by optimizing round functions and enhancing the AES encryption method's avalanche effect.

REFERENCES





- [1] R. Riyaldhi, Rojali, and A. Kurniawan, "Improvement of advanced encryption standard algorithm with shift row and S.Box modification mapping in mix column," *Procedia Computer Science*, vol. 116, pp. 401–407, 2017, doi: 10.1016/j.procs.2017.10.079.

- [2] Y. Zhang, A. Chen, and B. Chen, "A unified improvement of the AES algorithm," *Multimedia Tools and Applications*, vol. 81, no. 13, pp. 18875–18895, May 2022, doi: 10.1007/s11042-022-12742-1.
- [3] Y. Alemami, A. M. Al-Ghonmein, K. G. Al-Moghrabi, and M. A. Mohamed, "Cloud data security and various cryptographic algorithms," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 2, pp. 1867–1879, Apr. 2023, doi: 10.11591/ijece.v13i2.pp1867-1879.
- [4] I. A. Awan, M. Shiraz, M. U. Hashmi, Q. Shaheen, R. Akhtar, and A. Ditta, "Secure framework enhancing AES algorithm in cloud computing," *Security and Communication Networks*, vol. 2020, pp. 1–16, Sep. 2020, doi: 10.1155/2020/8863345.
- [5] 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, pp. 1641–1652, Mar. 2024, doi: 10.11591/ijeecs.v33.i3.pp1641-1652.
- [6] 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, pp. 551–560, Jun. 2022, doi: 10.12928/TELKOMNIKA.v20i3.23297.
- [7] M. Sawka and M. Niemiec, "A sponge-based key expansion scheme for modern block ciphers," *Energies*, vol. 15, no. 19, pp. 1–18, Sep. 2022, doi: 10.3390/en15196864.
- [8] I. Sultan, M. Y. Lone, M. Nazish, and M. T. Banday, "A secure key expansion algorithm for present," *IEEE Sensors Journal*, vol. 23, no. 20, pp. 25367–25376, Oct. 2023, doi: 10.1109/JSEN.2023.3267386.
- [9] P. Kulkarni, R. Khanai, D. Torse, N. Iyer, and G. Bindagi, "Neural crypto-coding based approach to enhance the security of images over the untrusted cloud environment," *Cryptography*, vol. 7, no. 2, pp. 1–17, May 2023, doi: 10.3390/cryptography7020023.
- [10] Y. Harmouch and R. El Kouch, "The benefit of using chaos in key schedule algorithm," *Journal of Information Security and Applications*, vol. 45, pp. 143–155, Apr. 2019, doi: 10.1016/j.jisa.2019.02.001.
- [11] A. Dmukh, D. Trifonov, and A. Chookhno, "Modification of the key schedule of the 2-GOST block cipher and its implementation on FPGA," *Journal of Computer Virology and Hacking Techniques*, vol. 18, no. 1, pp. 49–59, Mar. 2022, doi: 10.1007/s11416-021-00406-x.
- [12] Y. Wei, T. Ye, W. Wu, and E. Pasalic, "Generalized nonlinear invariant attack and a new design criterion for round constants," *IACR Transactions on Symmetric Cryptology*, vol. 2018, no. 4, pp. 62–79, 2018, doi: 10.13154/tosc.v2018.i4.62-79.
- [13] R. Saha, G. Geetha, G. Kumar, and T. H. Kim, "RK-AES: an improved version of AES using a new key generation process with random keys," *Security and Communication Networks*, vol. 2018, pp. 1–11, Nov. 2018, doi: 10.1155/2018/9802475.
- [14] H. M. Hussien, Z. Muda, and S. M. Yasin, "New key expansion function of Rijndael 128-bit resistance to the related-key attacks," *Journal of Information and Communication Technology*, vol. 17, no. 3, pp. 409–434, 2018, doi: 10.32890/jict2018.17.3.2802.
- [15] D. N. Hammod, M. H. Al-Rawi, and H. S. Abdulah, "An enhancement method based on modifying CFB mode for key generation in AES algorithm," *Engineering and Technology Journal*, vol. 34, no. 6B, pp. 759–768, Jun. 2016, doi: 10.30684/etj.34.6b.5.
- [16] E. M. De Los Reyes, A. M. Sison, and R. P. Medina, "Modified AES cipher round and key schedule," in *2018 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, Oct. 2018, pp. 146–146, doi: 10.1109/iciibms.2018.8549995.
- [17] M. K. Pehlivanoglu, M. T. Sakalli, N. Duru, and F. B. Sakalli, "The new approach of AES key schedule for lightweight block ciphers," *IOSR Journal of Computer Engineering*, vol. 19, no. 03, pp. 21–26, May 2017, doi: 10.9790/0661-1903042126.
- [18] Z. Cao, G. Yi, B. Wu, J. Li, and D. Xiao, "Analysis and improvement of AES key expansion algorithm," in *2022 International Conference on Artificial Intelligence and Computer Information Technology, AICIT 2022*, Sep. 2022, pp. 1–5, doi: 10.1109/AICIT55386.2022.9930239.
- [19] T. Manoj 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, Jun. 2018, doi: 10.1007/s10617-017-9189-5.
- [20] R. M. de Leon, A. M. Sison, and R. P. Medina, "A modified tiny encryption algorithm using key rotation to enhance data security for internet of things," in *2019 International Conference on Information and Communications Technology, ICOIACT 2019*, Jul. 2019, pp. 56–60, doi: 10.1109/ICOIACT46704.2019.8938456.
- [21] E. M. Galas and B. D. Gerardo, "Implementing randomized salt on round key for corrected block tiny encryption algorithm (XXTEA)," in *2019 IEEE 11th International Conference on Communication Software and Networks, ICCSN 2019*, Jun. 2019, pp. 795–799, doi: 10.1109/ICCSN.2019.8905270.
- [22] M. Imdad, S. N. Ramli, and H. Mahdin, "An enhanced key schedule algorithm of PRESENT-128 block cipher for random and non-random secret keys," *Symmetry*, vol. 14, no. 3, pp. 1–22, Mar. 2022, doi: 10.3390/sym14030604.
- [23] A. A. Zakaria, A. H. Azni, F. Ridzuan, N. H. Zakaria, and M. Daud, "Modifications of key schedule algorithm on RECTANGLE block cipher," in *Communications in Computer and Information Science*, vol. 1347, 2021, pp. 194–206, doi: 10.1007/978-981-33-6835-4_13.
- [24] 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, pp. 1–17, Dec. 2021, doi: 10.3390/s21248347.
- [25] A. Rukhin, J. Soto, and J. Nechvatal, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Gaithersburg, MD, 2010, doi: 10.6028/NIST.SP.800-22r1a.
- [26] 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.
- [27] N. Kapalova, K. Algazy, A. Haumen, and K. Sakan, "Statistical analysis of the key scheduling of the new lightweight block cipher," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 6, pp. 6817–6826, Dec. 2023, doi: 10.11591/ijece.v13i6.pp6817-6826.
- [28] G. Yi and Z. Cao, "An algorithm of image encryption based on AES and rossler hyperchaotic modeling," *Mobile Networks and Applications*, Sep. 2023, doi: 10.1007/s11036-023-02216-5.
- [29] D. Gerault, P. Lafourcade, M. Minier, and C. Solnon, "Computing AES related-key differential characteristics with constraint programming," *Artificial Intelligence*, vol. 278, p. 103183, Jan. 2020, doi: 10.1016/j.artint.2019.103183.
- [30] C. Boura, P. Derbez, and M. Funk, "Related-key differential analysis of the AES," *IACR Transactions on Symmetric Cryptology*, vol. 2023, no. 4, pp. 215–243, Dec. 2023, doi: 10.46586/tosc.v2023.i4.215-243.
- [31] 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.gltip.2021.01.014.

BIOGRAPHIES OF AUTHORS

Muthu Meenakshi Ganesan     received her B.Sc. degree in computer science from Madurai Kamaraj University in 2004 and her MCA degree from Alagappa University in 2007. She earned her M.Phil. degree in computer science from Mother Teresa Women's University in 2012. From 2012 to 2022, she worked as an assistant professor at various arts and science colleges. She is pursuing her Ph.D. in computer science at the SRM Institute of Science and Technology in Kattankulathur, Tamil Nadu, India. Her research interests include cloud computing and cryptography. She can be contacted at mg0480@srmist.edu.in.



Sabeen Selvaraj     works in the Department of Computer Science, Faculty of Science and Humanities, SRM Institute of Science and Technology, Kattankulathur. He received his Ph.D. in computer science from Anna University in 2012, his MCA in 2002, and his M.Tech. in computer science and engineering in 2015. With 22 years of teaching experience, he has worked at several engineering colleges, including Mohamed Sathak Engineering College, Noorul Islam College of Engineering, Jaya Engineering College, and Sidharth Institute of Engineering and Technology. He is a life member of ISTE, and his research interests include data mining, IoT, machine learning, cloud computing, and cryptography. He has published numerous papers in international journals and has guided over 100 postgraduate projects. He can be contacted at sabeens@srmist.edu.in.