WJCMS

Wasit Journal of Computer and Mathematics Science

Journal Homepage: https://wjcm.uowasit.edu.iq/index.php/WJCM

e-ISSN: 2788-5879 p-ISSN: 2788-5879

# Modifying the AES Algorithm by Improving the Add Round Key Stage

**Hasan Kadhim A. Alsuwaiedi[1],* and Abdul Monem S. Rahma[2]**

[1]Iraqi Commission for Computers & Informatics, Informatics Institute for Postgraduate Studies, Baghdad, Iraq
[2]Computer Science Department Al-Maarif University College, Anbar, Iraq

*Corresponding Author: Hasan Kadhim A. Alsuwaiedi

DOI: https://doi.org/10.31185/wjcms.162
Received: July 2023; Accepted: September 2023; Available online: September 2023

**ABSTRACT:** This study offers a new adjustment to the Advanced Encryption Standard (AES) in order to assure a high degree of security. This is achieved by replacing the binary (XOR) operation with a new (Xo) operation in each add-round-key stage. The Xo operation generated an extra six randomly selected control keys determined by six state tables (2, 4, 6, 8, 10, and 12) produced from the addition operation in the Galois Field GF ($2^2$ , $2^4$ , $2^6$ ,$2^8$ , $2^{10}$ and $2^{12}$ ) in order to boost the algorithm's unpredictability. In the suggested method, an adversary requires at least ($2^{168}$ ) 10 probabilities of keys to break the message; hence, it improves the difficulty of the original AES against brute force attacks. also enhances the performance of additional security metrics, such as NIST tests, compared to the original AES. Consequently, this replacement, including the use of six keys in both the encryption and decryption processes, offers a new level of security and a higher degree of resistance to data breaches. The novelty of the proposed (Xo) technique lies in the construction of GF tables ($2^6$ , $2^{10}$ and $2^{12}$ ) to be used in the encryption and decryption process for the first time, as well as the approach utilized to create the code for it.

**Keywords:** AES, Information security, Encryption algorithms, multi-level keys, Symmetric Block cipher

## 1. INTRODUCTION

It is vital to create a safe, protected environment for the transmission of electronic data such as texts, photos, videos, and other forms of multimedia via insecure networks [1]. Encryption is one way to protect sensitive information [2]. Specifically, "Cryptography" refers to the use of encryption methods to transform a communication into a format that cannot be read by unauthorized parties [3]. One of the most common targets of encryption is text [4], along with other types such as images, audio, video...etc. This research focuses on text encryption but can be developed for other types. The National Institute of Standards and Technology reports that AES can be used to encrypt any multimedia data, including texts, photos, and videos [5]. Rijndael is an alternate name for AES due to the fact that it was introduced by Joan Daemen and Vincent Rijmen in 1998 [6]. Since the introduction of Rijndael as an alternative to the Data Standard System (DES) algorithm [7]. AES has been widely used to protect digital data across insecure networks [8]. AES is a block cipher using symmetric keys that runs quicker in both hardware and software [9]. It supports several key lengths, including 128, 192, and 256 bits [10], and works on a plaintext block that is 128 bits long. During the key expansion step, the sub-key is obtained from the main key in order to determine the number of rounds that will be utilized for encryption and decryption. The number of rounds that are counted by the AES is dependent on the length of the keys; 10 rounds are counted for a 128-bit key; 12 rounds are counted for a 192-bit key, and 14 rounds are counted for a 256-bit key [11]. First, the round key is generated by taking the matrix that represents the plaintext and the key and combining them. After that, the output from the prior step travels through a total of ten cycles. The first nine rounds are composed of the four transitions listed below [12]: Add round Key Sub-bytes, Shift-rows and MixColumn the. Except for the MixColumn process, all of the above alterations are made in the 10th round. In this paper, the focus is on updating the add round key stage, which is a change where each round, a byte from the expanded key array is Xored to a byte from the state array [13] [14].

Binary functions having just two possible values, 0 and 1, are extensively used in current cryptographic methods for encrypting and decrypting data. As a block algorithm, AES makes use of many different operations, including the standard logical XOR operation on two binary values (0 and 1). Consequently, there are a few problems with this approach, such as the fact that attackers may easily recover the plaintext. Consequently, there are a few problems with this approach, such as the fact that attackers may easily recover the plaintext. Researchers have tried to expand the key space by replacing the two states with tables with four states (0, 1, 2, and 3), as seen in [15].

This research suggests a new change to the key space of the AES algorithm, instead of the initial XOR operation in the add-round-key stage, which only works on two states (0 and 1), with a new (Xo) operation of six state tables (2, 4, 6, 8, 10, and 12) derived from the addition operation in GF ($2^2$, $2^4$, $2^6$,$2^8$, $2^{10}$ and $2^{12}$). This is accomplished with the extra control key for these state tables. Though, this paper will make cryptography safer by maximizing the degree of difficulty that is present in each phase or round. This will ensure that information will remain private and kept confidential. This paper's remaining sections are structured as follows: sections 2 overview of relevant prior research; section 3 provide background on binary functions; sections 4 the suggested technique of AES development; section 5 offer its security metrics; followed by conclusions in section 6.

## 2. RELATED WORK

For modern encryption and decryption, most cryptographic algorithms rely on functions having two states (0, 1) [16]. As a result, most algorithms employ the traditional logical operation (XOR), which is based on two states: simply (0, 1) and has various drawbacks, including being simple since attackers can readily understand it. Therefore, researchers sought to replace the two states with four (0, 1, 2, and 3). This section provides a summary of the associated literature on possible improvements to key distribution using state tables in the AES algorithm. In 2009 [17], he and colleagues published their work, which increased the security and key space of encryption algorithms by merging curve security approaches with quantum cryptography principles. This was done in order to boost the encryption algorithm's overall strength. In this particular piece of work, the adjustment focuses on making an adaptation to the underlying protocol by employing different bit tampering techniques. This is accomplished by employing four distinct states (0, 1, 2, and 3) rather than only two states (0, 1) in order to increase the unpredictability of the polarized angles utilized in quantum description. These polarized angles are encrypted using the four state tables. The plaintext is first converted into ciphertext by manipulation ciphers, which do this by modifying the actual state pattern of each character with the use of a logical operator (#).

In addition, a great number of experts have tried to introduce these four states into the Feistel block cipher structure, which includes DES and Blowfish. In 2010 [18]**,** the researcher revealed a novel way of improving the efficiency of the DES algorithm by altering the predetermined XOR operation used during the sixteen rounds of the traditional Feistel algorithm. The (#) action is mostly dependent on the use of two keys. Each key is made up of a combination of four states (0, 1, 2, and 3) rather than the traditional two states (0 and 1). This update increases the protective strength and resilience of breaking techniques against attackers. In 2011 [19], the same method of increasing the complexity of the algorithm by employing the (#) operation with four-state tables was used in Feistel of Blowfish. This was done in order to raise the difficulty of the algorithm. Because it is not compactable to the hardware of computers that operate on 8 bits, the usage of the (#) operation in the Feistel can be considered to be its worst flaw. This operation works with just two bits at a time. In 2013 [20]**,** sub-byte transformation function encryption and decryption were performed using dual keys. The primary key uses a set of keys to create an S-box at random. The second important factor is the random distribution of the S-boxes, which leads to the generation of S-boxes with inverse relations to each other. As a result, using two keys led to an increase in complexity while simultaneously reducing the delay time inherent to the encryption and decryption procedures. In 2021 [21] **,** this work presents a new modification referred to as "extended function" (E#) by altering the classic XOR operation used during the Add round key stage in the AES algorithm with the (E#), which is based on implementing multi-state tables created depend on addition in GF($2^n$) with n-dynamic sections of (1, 2, 4, and 8-blocks), used in the AES algorithm as a symmetric cryptographic method. By enhancing its complexity and unpredictability, it offers a new level of security against breaking approaches.

## 3. BACKGROUND OF BINARY FUNCTIONS

Modern encryption is built on the finite field GF($2^n$) and the binary function XOR used in encryption and decryption, such as the Vernam Cipher in Figure 1, to produce the cipher text by mixing the key stream (ki) with the plaintext (pi), producing a ciphertext (ci), and vice versa. Inside the finite field, the arithmetic operations are performed on polynomials (addition and multiplication) using the ordinary rules of algebra based on modular polynomial arithmetic. By far, the most common algebraic structure utilized in the development of cryptographic methods is finite fields [16]. Based on this foundation, several new cryptographic systems have been developed, as described in [17].

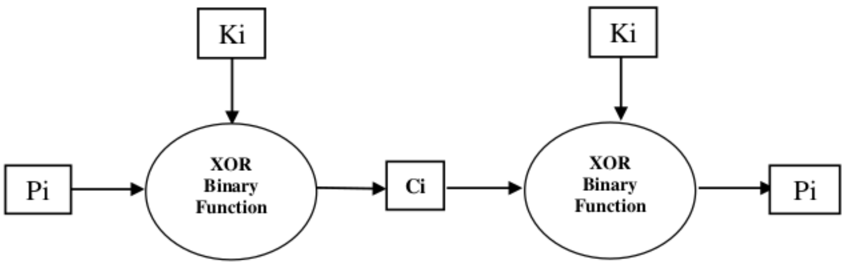


FIGURE 1. **Binary Function XOR in Vernam Cipher Invalid source**

Using state tables based on AND & XOR functions in encryption and decryption will add more security to the system due to the high randomness of the produced key or plaintext. Algorithms for cryptography depend on functions with four states [0, 1, 3, 4], as in Figure 2 [18].

| **#0** | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 1 | 0 | 3 | 2 |
| 2 | 2 | 3 | 0 | 1 |
| 3 | 3 | 2 | 1 | 0 |

| **#1** | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 1 | 1 | 0 | 3 | 2 |
| 0 | 0 | 1 | 2 | 3 |
| 3 | 3 | 2 | 1 | 0 |
| 2 | 2 | 3 | 0 | 1 |

| **#2** | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 3 | 3 | 2 | 1 | 0 |
| 0 | 0 | 1 | 2 | 3 |
| 1 | 1 | 0 | 3 | 2 |
| 2 | 2 | 3 | 0 | 1 |

| **#3** | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 |
| 3 | 3 | 2 | 1 | 0 |
| 2 | 2 | 3 | 0 | 1 |
| 1 | 1 | 0 | 3 | 2 |

FIGURE 2. **Four state tables for (#) operator**

The (#) function, which uses dynamic blocks (1, 2, 4, 8-blocks) and addition in GF($2^n$) to create multi-state tables, is added to the encryption algorithms for strengthening security, increasing the complexity of the ciphertext or key, and minimizing the encryption time [21].

## 4. THE PROPOSED AES DEVELOPMENT

It is crucial to enforcement the encryption algorithm from any form of assault. Therefore, this section presents a novel method for boosting the AES algorithm's security without increasing its computational burden. The enhanced proposal is exactly the same as the conventional AES method in the following sectors: key length, plaintext, number of rounds, operation on a byte, and number of stages in each round. However, it's different at the add round key phase, where the traditional XOR operation is swapped out by the suggested Xo operation to strengthen AES's security without adding any layers of complexity to the algorithm's computation. The new Xo function performs its work in encryption during rounds 10, 12, and 14 of the algorithm AES, taking into consideration the reverse order during decryption.

The newly introduced Xo operation results in the production of six keys rather than just one, together with a bit block of configurable size. As an element that is part of the range of the finite in GF ($2^2$, $2^4$, $2^6$,$2^8$, $2^{10}$ and $2^{12}$), the block size is mapped to the values of 2, 4, 6, 8, 10, and 12 bits, respectively. The employment of blocks of bits with varied sizes has the dual purpose of increasing the unpredictability of the algorithms and boosting their overall efficiency. A binary block of two bits is utilized on a table that comprises four distinct states (0, 1, 2, and 3), while a binary block of four bits is employed to operate on a table that encompasses 16 states ranging from 0 to 15. Similarly, an eight-bit binary block is utilized to manipulate a table that encompasses 256 states ranging from 0 to 255. A decagonal block operates on a table comprising 1024 states, ranging from 0 to 1023. In contrast, a block with a bit size of twelve operates on a set of 4096 states, ranging from 0 to 4095.

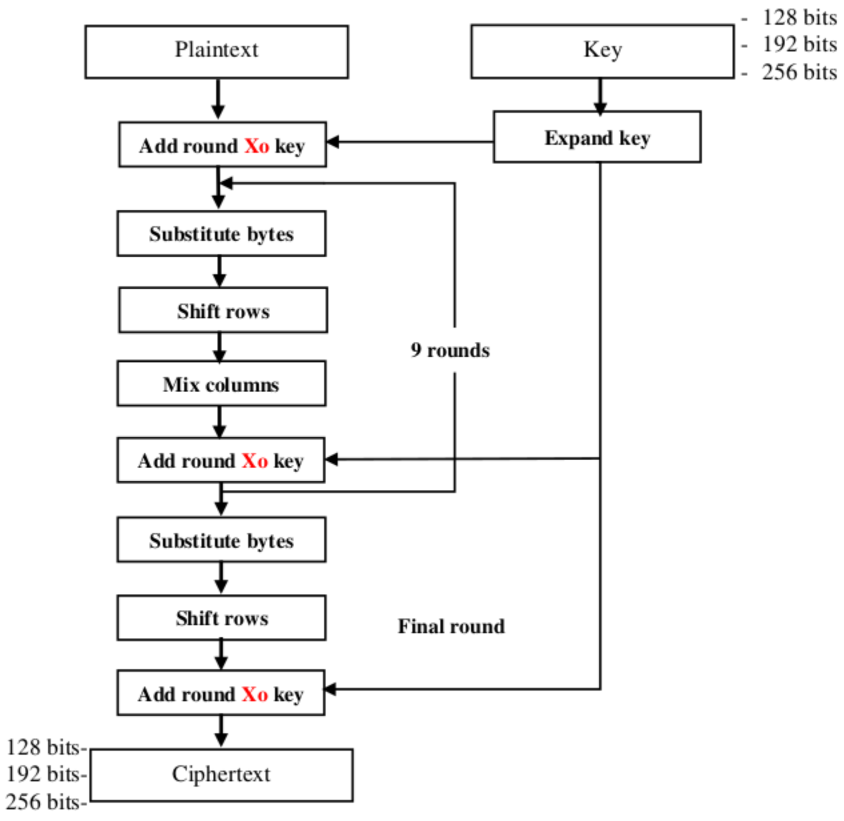


FIGURE 3. **The Modified AES using (Xo) operation**

In Figure 3, notice the general structure of the AES algorithm and the location of the proposed function (Xo), where it is inside the add-round key stage and wherever it is repeated within the algorithm.

In order to regulate the quantity of bits and state tables, the novel (Xo) operation produced two supplementary keys (k1, k2) with a variable size (for instance, consider a size of 16 bits), divided into three parts with different sizes, generated in a dependent manner. The first key, k1, is extracted from plaintext. The second key, k2, is extracted from the original key, as depicted in Figure 4. Inside the function Xo, the key k1 is segmented into three parts as (**k1_1, k1_2, k1_3**). Also, the key k2 is segmented into three parts as (**k2_1, k2_2, k2_3**). The whole parts are considered inputs to the function Xo and are described as follows:

- The first segment in both keys (**k1_1 and k2_1**) represents the first main state table, where both have the same bit size but different decimal values.
- The second segment in both keys (**k1_2 and k2_2**) represents the second main state table, where both have the same bit size but different decimal values.
- The third segment in both keys (**k1_3 and k2_3**) represents the third main state table, where both have the same bit size but different decimal values.

Figure 4 depicts the proposed Xo functionality, whereas the result is based on the intersection between row (k1_1, k1_2, k1_3) and column (k2_1, k2_2, k2_3) in the specified substate tables. Additionally, the same procedure used for encryption is repeated for decryption.

The variety of tables containing more states is employed to increase the unpredictability of the process. The new suggested approach employs the function (Xo), which works on six variable block bit sizes: 2, 4, 6, 8, 10, and 12 based on the constructed state tables. The sizes of the keys (k1_1, k1_2, k1_3) and (k2_1, k2_2, k2_3) can take one of the values 2, 4, 6, 8, 10, or 12, which represent the six state tables used in this research. Also, it is significant to notice that only

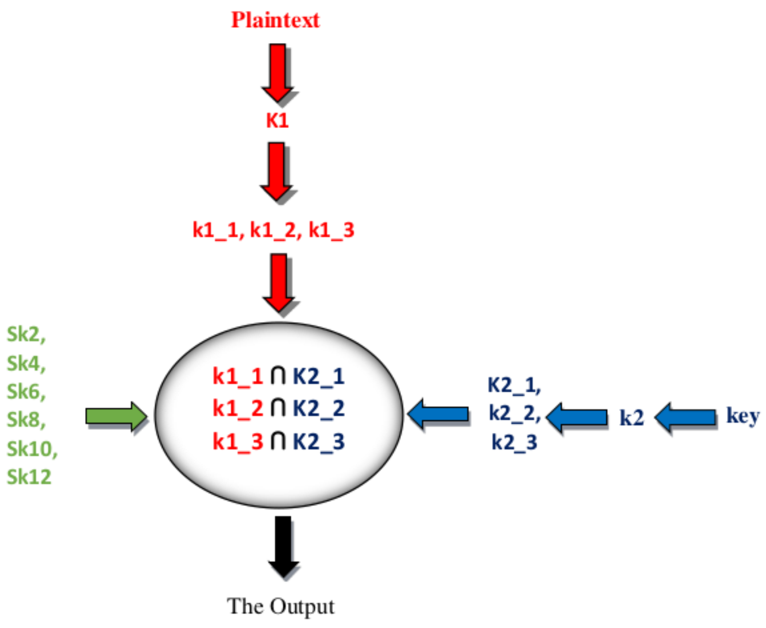


**FIGURE 4.** **The proposed Xo function specification**

seven probabilities were employed in the division of the keys **k1, k2,** as samples in this research due to the enormous possibilities. An illustration of the key's specification of the Xo function is shown in Table 1.

**Table 1.** **The keys specification of Xo function**

| | **First main state table = k1_1 or k2_1**/bits | **Second main state table = k1_2 or k2_2** /bits | **Third main state table = k1_3 or k2_3** /bits | **k1 = k1_1 + k1_2 + k1_3 k2 = k2_1 + k2_2 + k2_3** |
|---|---|---|---|---|
| 1 | 8 | 8 | 0 | 16 bits |
| 2 | 8 | 4 | 4 | = |
| 3 | 4 | 4 | 8 | = |
| 4 | 6 | 10 | 0 | = |
| 5 | 2 | 4 | 10 | = |
| 6 | 12 | 4 | 0 | = |
| 7 | 2 | 2 | 12 | = |
| : | : | : | : | : |

Moreover, work was done in this study to create extra **substate tables** from the three **main state tables** (k1_1, k1_2, k1_3) and (k2_1, k2_2, k2_3) based on the six state tables (2, 4, 6, 8, 10, or 12). Therefore, there are:

- 4 state tables (0, 1, 2, and 3) for the main state table 2 or GF ($2^2$).
- 16 state tables (0, 1, . . . , 15) for the main state table 4 or GF ($2^4$).
- 64 state tables (0, 1, . . . , 63) for the main state table 6 or GF ($2^6$).
- 256 state tables (0, 1, . . . , 255) for the main state table 8 or GF ($2^8$).
- 1024 state tables (0, 1, . . . , 1023) for the main state table 10 or GF ($2^{10}$).
- 4096 state tables (0, 1, . . . , 4095) for the main state table 12 or GF ($2^{12}$).

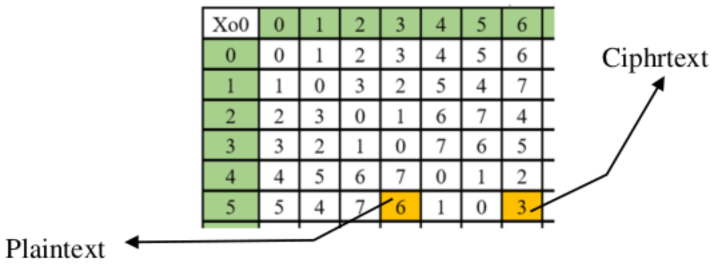
A number of instances of these tables are presented in Tables 2-5 below. After specifying the three main state tables for both (k1_1, k1_2, k1_3) and (k2_1, k2_2, k2_3) inside the Xo function, new processing values are generated in a sequence base referred to as (**St2, St4, St6, St8, St10, St12**) to specify the needed sequence of **substate tables** as mentioned in the previous figure 3. The following equation explains the input values of the Xo function: the intersection of the plaintext with the key inside the substate table, where the intersection between row and column produces the ciphertext value.

$$\text{Ciphertext}=\text{Xo}\ ((\text{St4}),\ k1\_1\ \cap\ k2\_1) \tag{1}$$

where **St4**=Substate table, k1_1= Row in decimal, k2_1= Column in decimal.

**Example 1**

In this example, a partial drawing was observed from Table 2 below, in which **St4**= Xo0, **Row**=k1_1=5, **Column** = k2_1=6 then the intersection value is:



| Xo0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 1 | 0 | 3 | 2 | 5 | 4 | 7 |
| 2 | 2 | 3 | 0 | 1 | 6 | 7 | 4 |
| 3 | 3 | 2 | 1 | 0 | 7 | 6 | 5 |
| 4 | 4 | 5 | 6 | 7 | 0 | 1 | 2 |
| 5 | 5 | 4 | 7 | 6 | 1 | 0 | 3 |

Ciphrtext=Xo ((Xo0), 5 ∩ 6) =3
Plaintext=Xo ((Xo0), 5 ∩ 3) =6

## 4.1 THE MODIFIED ADD-ROUND-KEY STAGE OF AES

Algorithm 1 explains the Xo operation inside the add-round key phase inside the AES algorithm.

**Algorithm 1: The Work of (Xo) Operation**

```
Input: Block clear message (or ciphertext of next rounds) = k1, and block key= k2
Output: Block of Ciphertext.
Begin
1.  Initialize counters as Sk2=1, Sk4=1, Sk6=1, Sk8=1, Sk10=1, Sk12=1 to represents the submain table
    index. Where SK2(take range 1-4), KS4(take range 1-16), KS6(1- 64), ……etc.
   While not end of the target state size (clear or cipher text)
    2.  k1 divided into three pieces of bits (k1_1, k1_2, k1_3) randomly within range 2, 4, 6, 8, 10 or 12
    3.  k2 divided into three pieces of bits (k2_1, k2_2, k2_3) randomly within range 2, 4, 6, 8, 10 or 12
    4.  Recall main states tables using addition operation on GF ($2^n$). based on the keys in 1 and 2.
    5.  Change the n-block keys chosen from 1 and 2 into a decimal number.
    6.  Recall substates tables based on Sk2, Sk4, Sk6, Sk8, Sk10, Sk12 depending on 4 of selected sizes
    7.  The intersection of the row and column in the substate table index, to get the result as ciphertext or
        plaintext
        Calculate based on three inputs: (Skj= submain table index, row=k1_i, column=k2_i).  where i=1, 2
        or 3: j=2, 4, 6, 8, 10 or 12
          Ciphrtext= Ciphrtext + (Skj, k1_i ∩ k2_i)
          Plaintext= Plaintext + (Skj, k1_i ∩ k2_i)
    8.  Increment the counters Sk2, Sk4, Sk6, Sk8, Sk10, Sk12 depending on 1 and 2 selected sizes.
   End While
9.  Store final result of the encryption and decryption process in new state
End
```

## 4.2 THE SUGGESTED AES

The suggested AES algorithm is laid out in detail in Algorithm 2 below. Changed procedures are marked in blue, replacing the red one:

**Algorithm 2: famous AES algorithm**

```
Input: -Original text message(pltx)128bits and key (k) 128bits
Output: -Ciphertext 128bit
Begin
1.  The cipher key is used to generate the set of round keys ($k_n$)
2.  Use the plaintext (pltx) for filling the state array (state)
3.  *Generate six keys with n bits size each.
4.  State = Addroundkey (pltx $\oplus$ $k_0$ )
5.  State = Addroundkey (pltx Xo $k_0$ )
6.  For round i = 1 to 9
        Begin
        SubBytes(state)
        ShiftRows(state)
        MixColomns(state)
        Addroundkey(state $\oplus$ $K_i$)
        Addroundkey(state Xo $K_i$)
        end
7.  Compute final round as:
        SubBytes(state)
        ShiftRows(state)
        Addroundkey(state $\oplus$ $K_{10}$)
        AddroundKey(state Xo $K_{10}$)
8.  Output = state
9.  End
* Calculation of Xo operation in Algorithm 1
```

## 5. THE BUILD STATUS TABLES

This section contains examples of the built tables using the mathematical addition operation in Galois Field GF($2^n$). The built state-tables were GF ($2^2$), GF ($2^4$), GF ($2^6$), GF ($2^8$), GF ($2^{10}$) and GF ($2^{12}$). As an example, tables below 2, 3, 4, and 5 represent samples of the addition tables in GF ($2^{10}$) and GF ($2^{12}$) consecutively.

Table 2. **State (Xo0) addition in GF ($2^{10}$ )**

| Xo0 | 0 | 1 | 2 | 3 | … | … | 60 | 61 | 62 | 63 | … | … | 885 | 886 | 887 | … | 1021 | 1022 | 1023 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | … | … | 60 | 61 | 62 | 60 | … | … | 885 | 886 | 887 | … | 1021 | 1022 | 1023 |
| 1 | 1 | 0 | 3 | 2 | … | … | 61 | 60 | 63 | 61 | … | … | 884 | 887 | 886 | … | 1020 | 1023 | 1022 |
| 2 | 2 | 3 | 0 | 1 | … | … | 62 | 63 | 60 | 62 | … | … | 887 | 884 | 885 | … | 1023 | 1020 | 1021 |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| 1021 | 1021 | 1020 | 1023 | 1021 | … | … | 961 | 960 | 963 | 961 | … | … | 170 | 165 | 164 | … | 1 | 2 | 3 |
| 1022 | 1022 | 1023 | 1020 | 1022 | … | … | 962 | 963 | 960 | 962 | … | … | 169 | 166 | 167 | … | 0 | 3 | 2 |
| 1023 | 1023 | 1022 | 1021 | 1023 | … | … | 963 | 962 | 961 | 963 | … | … | 168 | 167 | 166 | … | 3 | 0 | 1 |

**Table 3. State (Xo1023) addition in GF ($2^{10}$ )**

| Xo1023 | 0 | 1 | 2 | 3 | … | … | 60 | 61 | 62 | 63 | … | … | 885 | 886 | 887 | … | 1021 | 1022 | 1023 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 197 | 196 | 199 | 198 | … | … | 249 | 248 | 251 | 250 | … | … | 954 | 837 | 836 | … | 824 | 827 | 826 |
| 2 | 102 | 103 | 100 | 101 | … | … | 90 | 91 | 88 | 89 | … | … | 793 | 998 | 999 | … | 923 | 920 | 921 |
| 3 | 89 | 88 | 91 | 90 | … | … | 101 | 100 | 103 | 102 | … | … | 806 | 985 | 984 | … | 932 | 935 | 934 |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| 1021 | 149 | 148 | 151 | 150 | … | … | 169 | 168 | 171 | 170 | … | … | 1002 | 789 | 788 | … | 872 | 875 | 874 |
| 1022 | 166 | 167 | 164 | 165 | … | … | 154 | 155 | 152 | 153 | … | … | 985 | 806 | 807 | … | 859 | 856 | 857 |
| 1023 | 1023 | 1022 | 1021 | 1020 | … | … | 963 | 962 | 961 | 960 | … | … | 128 | 127 | 126 | … | 2 | 1 | 0 |

**Table 4. State (Xo7) addition in GF ($2^{12}$ )**

| Xo7 | 0 | 1 | 2 | 3 | … | … | 2572 | 2573 | … | … | 3006 | 3007 | 3008 | … | 4093 | 4094 | 4095 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | … | … | 2572 | 2573 | … | … | 3006 | 3007 | 3008 | … | 4093 | 4094 | 4095 |
| 1 | 5 | 4 | 7 | 6 | … | … | 2569 | 2568 | … | … | 3003 | 3002 | 3013 | … | 4088 | 4091 | 4090 |
| 2 | 3 | 2 | 1 | 0 | … | … | 2575 | 2574 | … | … | 3001 | 3000 | 3015 | … | 4094 | 4093 | 4092 |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| 4093 | 4093 | 4092 | 4095 | 4093 | … | … | 1521 | 1520 | … | … | 1091 | 1090 | 1085 | … | 0 | 3 | 2 |
| 4094 | 4094 | 4095 | 4092 | 4094 | … | … | 1522 | 1523 | … | … | 1088 | 1089 | 1086 | … | 3 | 0 | 1 |
| 4095 | 4095 | 4094 | 4093 | 4095 | … | … | 1523 | 1522 | … | … | 1089 | 1088 | 1087 | … | 2 | 1 | 0 |

**Table 5. State (Xo4096) addition in GF ($2^{12}$ )**

| Xo4096 | 0 | 1 | 2 | 3 | … | … | 2017 | 2018 | … | … | 3974 | 3975 | 3976 | … | 4093 | 4094 | 4095 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 19 | 18 | 17 | 16 | … | … | 2034 | 2033 | … | … | 3989 | 3988 | 3995 | … | 4078 | 4077 | 4076 |
| 1 | 6 | 7 | 4 | 5 | … | … | 2023 | 2020 | … | … | 3968 | 3969 | 3982 | … | 4091 | 4088 | 4089 |
| 2 | 3 | 2 | 1 | 0 | … | … | 2017 | 2018 | … | … | 3973 | 3972 | 3979 | … | 4094 | 4093 | 4092 |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … | … |
| 4093 | 12 | 13 | 14 | 15 | … | … | 2030 | 2029 | … | … | 3978 | 3979 | 3972 | … | 4081 | 4082 | 4083 |
| 4094 | 17 | 16 | 19 | 18 | … | … | 2029 | 2030 | … | … | 3991 | 3990 | 3993 | … | 4076 | 4079 | 4078 |
| 4095 | 14 | 15 | 12 | 13 | … | … | 2032 | 2035 | … | … | 3976 | 3977 | 3974 | … | 4083 | 4080 | 4081 |

## 6. RESULTS DISCUSSION

In the present section, the proposed AES algorithm is compared to the original AES algorithm for encrypting texts using three indicators (**complexity analysis, encryption time, throughout and NIST tests**), as shown below. The algorithm is simulated and evaluated using Microsoft Visual Studio C++ 2019 on an Intel Core i9-12900H 2.90 GHz processor and 32.0 GB of RAM.

### 6.1 SECURITY COMPLEXITY ANALYSIS

By calculating the number of possible keys an attacker would need to use the original AES key with the extra six keys of the proposed AES, with six blocks of either (2, 4, 6, 8, 10, or 12) bit size and different state tables to decipher the encrypted message with 64 bits, the paper can determine how hard the proposed technique is. The number of potential keys utilized for encryption/decryption is estimated by first determining the complexity of the famous AES algorithm using the binary operation XOR (0, 1) in the add-round-key stage, which is computed as stated in Equation (2) [21]:

$$\text{Complexity} = \text{Plaintext} \times \text{Key} \tag{2}$$

$One\ round\ complexity = 2^{24} \times 2^{40} \times 2^{16} \times 2^{46} = 2^{126}$

$Ten\ round\ complexity = (2^{24} \times 2^{40} \times 2^{16} \times 2^{46})^{10} = (2^{126})^{10}$

Consequently, the complexity of our suggested approach is calculated by using six keys, which utilized in the add-round-key stage with (2, 4, 6, 8, 10, 12) distinct state tables. The total complexity of each cycle of our proposed method is computed using Equation (3).

$$\text{Complexity} = \text{Plaintext} \times \text{Key} \times \text{six state tables} \tag{3}$$

$One\ round\ complexity = 2^{24} \times 2^{40} \times 2^{16} \times 2^{46} \times (2^{2} \times 2^{4} \times 2^{6} \times 2^{8} \times 2^{10} \times 2^{12}) = 2^{168}$

$Ten\ round\ complexity = (2^{24} \times 2^{40} \times 2^{16} \times 2^{46} \times \left(2^{2} \times 2^{4} \times 2^{6} \times 2^{8} \times 2^{10} \times 2^{12}\right))^{10} = (2^{168})^{10}$

Table 6 and Figure 5 demonstrate the results of comparing the suggested AES algorithm with the famous AES method in terms of computational complexity, showing how much more difficult the suggested AES algorithm is in one round and ten rounds than the standard one.

**Table 6. Results from analysis of complexity**

| Algorithm | One round | Ten rounds |
|---|---|---|
| Famous AES | $2^{126}$ | $(2^{126})^{10}$ |
| Proposed AES | $2^{168}$ | $(2^{168})^{10}$ |

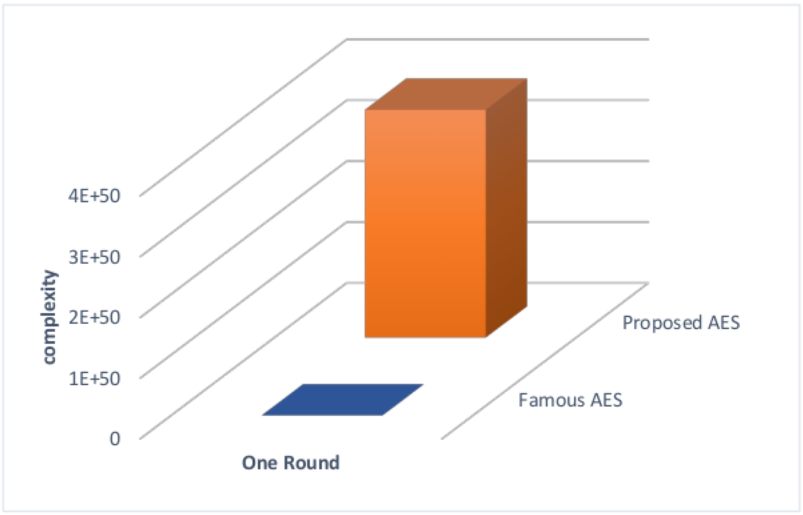


**FIGURE 5. One round complexity**

## 6.2 ENCRYPTION TIME AND THROUGHPUT

The encryption time is calculated using the amount of time needed to transform the plaintext into an unknown (ciphertext), which is another metric for evaluating the algorithm's performance [22]. In this situation, the throughput metric is calculated as follows:

$$\text{Throughput}=\text{plaintext size (in kilobyte)}/\text{total encryption time (ms)} \quad (4)$$

According to Table 7, Table 8, and Figure 6, Figure 7, in this test using different text file sizes from 15 kb to 1200 kb, the computation times for the original and the suggested AES algorithms are identical in encryption time for some file sizes and higher for others in the decryption process. However, the results of the suggested method are more effective in terms of complexity evaluation against attacks, making it more challenging for an attacker to recover the clear-message from the proposed AES algorithm.

**Table 7. The Encryption time for 10 rounds (in milliseconds)**

| File sizes(kb) | **famous AES algorithm** | **The proposed AES algorithm** |
|---|---|---|
| 15 | 56 | 43 |
| 50 | 81 | 78 |
| 200 | 286 | 298 |
| 500 | 653 | 650 |
| 600 | 775 | 770 |
| 1200 | 1584 | 1591 |
| Average time | 572.5 | 41.5 |
| throughput | 0.746 | 3.674 |

**Table 8. The Decryption time for 10 rounds (in milliseconds)**

| File sizes(kb) | **famous AES algorithm** | **The proposed AES algorithm** |
|---|---|---|
| 15 | 134 | 72 |
| 50 | 180 | 177 |
| 200 | 830 | 863 |
| 500 | 1964 | 2148 |
| 600 | 2349 | 2388 |
| 1200 | 4610 | 4105 |
| Average time | 1677.8 | 1625.5 |
| throughput | 0.254 | 0.262 |

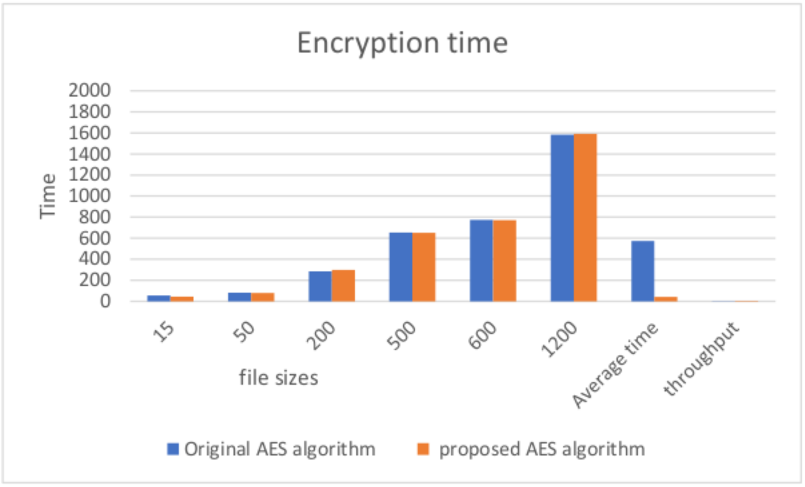


**FIGURE 6. The encryption time and throughout**

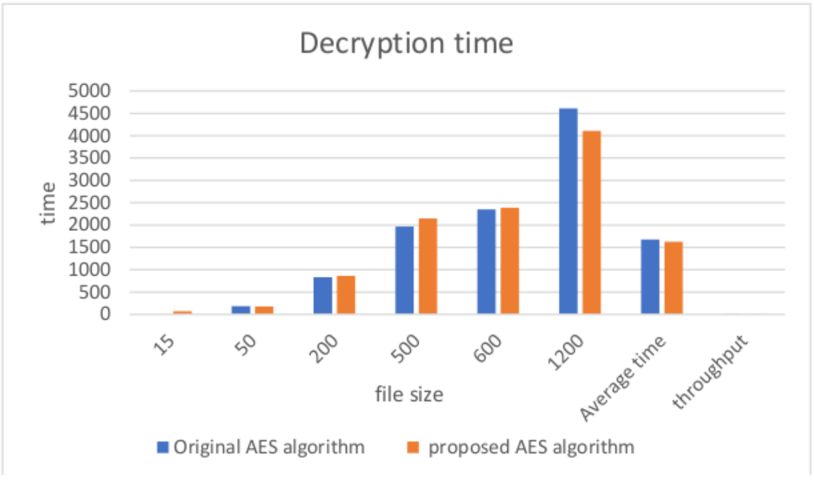


FIGURE 7. **The decryption time and throughout**

## 6.3 NIST TESTS ANALYSIS

The goal of an encryption algorithm is to make the binary output more unpredictable and random [23]. Numerous encryption schemes, such as the NIST (National Institute of Standards and Technology), Diehard tests, and TestU01, can compute randomness. In this study, 15 statistical NIST tests are used to assess the unpredictability of binary sequences. Three encrypted files are evaluated using the original and new AES algorithms. In evaluating these encrypted files, the original AES failed to pass the p-values that were set to 0.01 in order to check that the output is random. While the suggested AES encryption algorithm passed all tests for these encrypted data, the values of the tests that failed the p-value threshold will be discarded. The average p-value for these encrypted files is then determined and presented in Table 9. If the test results provide a p-value that approaches 1 asymptotically, the output should be fully random. A p-value of 0 indicates that the output was not generated at random [24] [25]. The pass status shows that the p-value for these tests exceeds 0.001, indicating that the result is acceptable. According to Table 9 and Figure 8, the new technique passes these statistical tests, but the original AES fails the same tests. Therefore, the suggested AES is superior to the original AES in the majority of tests.

Table 9. **Nist Tests computation between the standard AES and suggested AES**

| Name of Statistical Test | **Standard AES** | | **Suggested AES** | |
|---|---|---|---|---|
| | **P-Value** | **Status** | **P-Value** | **Status** |
| Approximate Entropy | 0.107645 | pass | 0.536802 | pass |
| Block Frequency | 0.350485 | pass | 0.002043 | pass |
| Cumulative Sums | 0.213309 | pass | 0.911413 | pass |
| FFT | 0.534146 | pass | 0.911413 | pass |
| Frequency | 0.035174 | pass | 0.122325 | pass |
| Linear complexity | 0.008879 | pass | 0.911413 | pass |
| Longest Run | 0.213309 | pass | 0.534146 | pass |
| Non-Overlapping Template | 0.991468 | pass | 0.911413 | pass |
| Overlapping Template | 0.350485 | pass | 0.066882 | pass |
| Random Excursions | 0.790776 | pass | 0.847120 | pass |
| Random Excursions Variant | 0.881847 | pass | 0.913203 | pass |
| Rank | 0.017912 | pass | 0.017912 | pass |
| Runs | 0.739918 | pass | 0.066882 | pass |
| Serial | 0.623656 | pass | 0.527541 | pass |
| Universal | 0.534146 | pass | 0.213309 | pass |

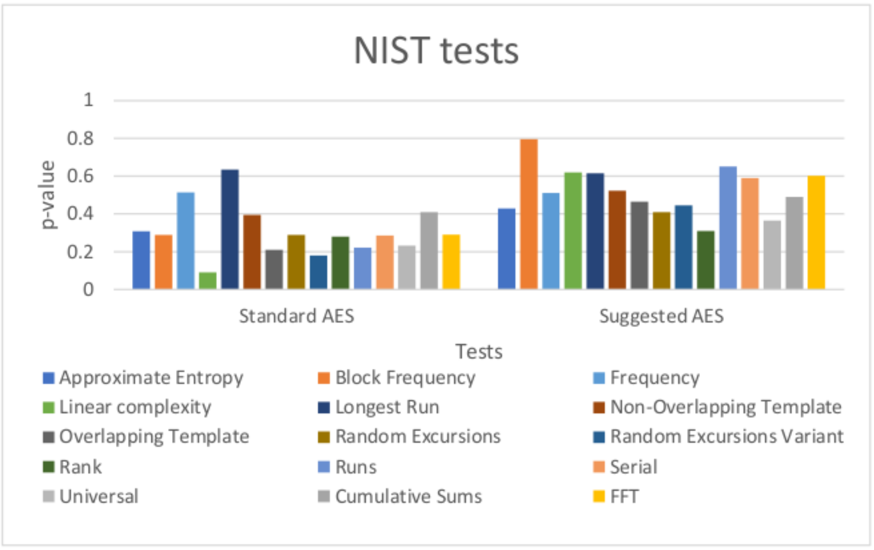


**FIGURE 8.** **The NIST tests**

## 7. CONCLUSION

This article describes an improvement to the widespread AES algorithm. The improved AES algorithm provides enhanced protection against brute-force attacks. This improvement is accomplished by replacing the preset XOR operation found in the add round key stage of AES, which is dependent on two binary states (0, 1), with the new (**Xo**) operation consisting of six state tables (2, 4, 6, 8, 10, and 12) derived from the addition operation in Galois Field GF ($2^2$, $2^4$, $2^6$,$2^8$, $2^{10}$ and $2^{12}$). The following conclusions may be drawn from the data: Using six keys boosts both the key scheduling's dependability and the encryption's efficiency. Additionally, the modified AES demonstrates that it is more complicated than the standard AES. Although this adjustment adds complexity, it decreases the time required for the encryption and decryption of certain file sizes while increasing it a little for others. According to NIST testing, the suggested technique also yields robust results when encrypting and decrypting samples of text files. The suggested algorithm's primary benefit is that it can build a huge number of state tables.

## FUNDING

None

## ACKNOWLEDGEMENT

None

## CONFLICTS OF INTEREST

The author declares no conflict of interest.

## REFERENCES

[1] O. M. A and A. A. Abu-Ein, "Chaotic based multimedia encryption: a survey for network and internet security," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 4, pp. 2151–2159, 2022.
[2] S. P. J. Meghana and N, "Multimedia Encryption For Enhancing Data Security Using AES and Logistic Mapping," *International Journal Of Creative Research Thoughts*, vol. 10, no. 2, pp. 2320–2882, 2022.
[3] M. D and A. H. Q. Z. Abdulla, "Robust Password Encryption Technique with an Extra Security Layer," *Iraqi Journal of Science*, vol. 64, no. 3, pp. 1477–1486, 2023.
[4] L. M. M. Lawnik *Chaos-Based Cryptography: Text Encryption Using Image Algorithms*.
[5] M. E. Smid, "Development of the Advanced Encryption Standard," *Journal of Research of the National Institute of Standards and Technology*, vol. 126, 2021.
[6] V. R. J. Daemen, *The Design of Rijndael The Advanced Encryption Standard (AES)*. GmbH Germany: Springer, 2020.
[7] Y. Wang, "Application of AES and DES Algorithms in File Management," *Journal of Physics: Conference Series*, 2021.
[8] N. Mouha, "Review of the Advanced Encryption Standard," *Technology Interagency or Internal Report*, vol. 8319, pp. 2021–2021.

[9] R. P. S. H. Vignesh, “An Efficient K-N Secret Sharing Image and AES Encryption Algorithm in Visual Cryptography,” *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 7, no. 2, 2018.

[10] A. B. Jang, “Quantum Analysis of AES Lowering Limit of Quantum Attack Complexity,” *IACR Cryptol. ePrint Arch*, vol. 2022, pp. 683–683, 2022.

[11] M. Masoumi, “A highly efficient and secure hardware implementation of the advanced encryption standard,” *Journal of Information Security and Applications*, 2019.

[12] H. L and G. M. Li, “A reconfigurable and compact subpipelined architecture for AES encryption and decryption,” *EURASIP Journal on Advances in Signal Processing*, 2023.

[13] A. P. K. J. M. Farissi, “Securing Messages Using AES Algorithm and Blockchain Technology on Mobile Devices,” *Sinkron: Jurnal dan Penelitian Teknik Informatika*, vol. 8, 2023.

[14] J and F. V. Wenceslao, “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, 2018.

[15] B. Z and A. M. S. Rahma *An Improved Algorithm for Partial Cryptography of Digital Video*, 2012.

[16] W. Stallings *Cryptography And Network Security Principles And Practice*, 2023.

[17] B. A. A. M. S. R. Hala, “Proposed new quantum cryptography system using quantum description techniques for generated curves,” in *The 2009 International conference on security and management*, 2009.

[18] R. F. Hassan, “New Approach for Modifying DES Algorithm Using 4-States Multi keys,” *Eng. & Tech. Journal*, vol. 28, no. 20, 2010.

[19] R. F. H. Afaf, M, and A. Al-Neaimi, “New Approach for Modifying Blowfish Algorithm by Using Multiple Keys multiple Keys,” *IJCSNS International Journal of Computer Science and Network Security*, vol. 11, no. 3, 2011.

[20] H. M. Nada, “Encryption using dual key transformation based on creation of multi S-boxes in AES algorithm,” *International Journal of Computer Applications*, vol. 83, no. 10, 2013.

[21] S. M. K and A. M. S. Rahma, “New method for improving add round key in the advanced encryption standard algorithm,” *Information Security Journal: A Global Perspective*, 2021.

[22] H. M. A. K. M. M. H. D. S. Abdul and Elminaam, “Performance Evaluation of Symmetric Encryption Algorithms,” *Communications of the IBIMA*, vol. 8, pp. 58–64, 2010.

[23] J. S. T. M. Alawida, “A chaos-based block cipher based on an enhanced logistic map and simultaneous confusion-diffusion operations,” *Journal of King Saud University Computer and Information Sciences*, vol. 34, no. 10, pp. 8136–8151, 2022.

[24] S. S and S. Velampalli, “Performance Evaluation for DES and AES Algorithms- An Comprehensive Overview,” in *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT-2018)*, 2018.

[25] W. Zhang, “p-value based statistical significance tests: Concepts, misuses, critiques, solutions and beyond,” *Computational Ecology and Software*, vol. 12, no. 3, pp. 80–122, 2022.