Received January 5, 2021, accepted January 10, 2021, date of publication January 13, 2021, date of current version February 4, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3051556*

# A Polymorphic Advanced Encryption Standard – A Novel Approach

**ABDELRAHMAN ALTIGANI[1], SHAFAATUNNUR HASAN[2], BAZARA BARRY[3], (Member, IEEE), SHIRAZ NASERELDEN[4], MUAWIA A. ELSADIG[5], AND HUWAIDA T. ELSHOUSH[6]**

[1]Department of Computer Science, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam 31441, Saudi Arabia
[2]School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, Johor 81310, Malaysia
[3]Macquarie Business School, Macquarie University, Sydney, NSW 2109, Australia
[4]Finance Science Department, Community College, Imam Abdulrahman Bin Faisal University, Dammam 31441, Saudi Arabia
[5]Deanship of Scientific Research, Imam Abdulrahman Bin Faisal University, Dammam 31441, Saudi Arabia
[6]Computer Science Department, Faculty of Mathematical Sciences, University of Khartoum, Khartoum 11111, Sudan

Corresponding author: Abdelrahman Altigani (a.altigani@gmail.com)

This work was supported by the Deanship of Scientific Research of Imam Abdulrahman Bin Faisal University (KSA), for the Project Grant Number 2014276.

**ABSTRACT** To secure various forms of data, a polymorphic variant of the Advanced Encryption Standard (P-AES) has been introduced. In the P-AES, the AES parameters' values will change with every new key. The exact values will be available only to legitimate communicating parties during execution time. To achieve these objectives, the basic AES transformations, SubBytes, ShiftRows, and MixColumns, have been made key-dependent in the proposed P-AES. Hence, with every new key, these transformations will operate differently. The receiver can retrieve the operations' details from the encryption key. Consequently, polymorphism has been achieved and interoperability remains intact. P-AES has been implemented seamlessly using the existing AES modules, and the performance was more or less equal to the AES performance (71 and 70 milliseconds to encrypt 500 bytes using the P-AES and the AES respectively). From a security standpoint, the proposed P-AES fully complies with Kerckhoff's principle. This means the cipher has an open design, and the security provided by the P-AES depends only on the secrecy of the encryption key. The cipher resistance to differential and linear attacks has been proved. Moreover, the resulting proposed cipher can operate in 128 different ways, which will significantly reduce the capabilities of any sophisticated attacker. Furthermore, the proposed P-AES's scores of the key avalanche and the plaintext avalanche were 0.496 and 0.504 respectively. Finally, the Statistical Test Suite (STS) recommended by the NIST has been used to ensure the randomness of the cipher output, and the cipher has passed all the STS tests.

**INDEX TERMS** Advanced encryption standard, cryptography, dynamic encryption, encryption, polymorphic cipher.

## I. INTRODUCTION

Nowadays, due to the convenience introduced by Internet technologies and computer networks, exchanging various forms of data has increased drastically. Moreover, according to [1], 60% of the web pages and 70% of the mobile phone traffic are multimedia data, which, in turn, lead to a significant increase in payload volumes. Consequently, the measures used to secure the increasing volumes of traffic must always be revised and updated to assure the security of the communicating parties' communications [1], [2].

According to the International Telecommunication Union (ITU), cybersecurity includes *confidentiality*, *integrity,* and *availability* security services [3]. These services are typically provided using a suitable mechanism or set of mechanisms. For instance, typically encryption is used to provide confidentiality security services [4]–[6].

Encryption algorithms can be classified according to different criteria. However, many research papers classified encryption algorithms to *symmetric* and *asymmetric* encryption algorithms [7]. Symmetric encryption algorithms use the same key for both encryption and decryption.

On the other hand, asymmetric encryption algorithms use two different keys per user, a public key and a private key.

This research paper focuses mainly on symmetric encryption algorithms. To be more specific, it aims at enhancing the AES cipher by introducing a polymorphic version of the AES. P-AES is a symmetric cipher that inherits the strength of

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Imran Tariq.

the AES, yet has a uniform layer of obscurity. Consequently, it does not leave enough trails of the used encryption transformations for opponents during its execution.

A high level description for the P-AES is provided in [8]. However, the contribution of this research paper includes the following:

1) Adequately exploring the literature to investigate similar attempts and highlight the key differences between the P-AES and other existing ciphers.
2) A thorough description of the proposed P-AES cipher design details is provided.
3) The P-AES has been implemented, and its performance in terms of encryption and decryption time is evaluated against the AES cipher.
4) The resistance of the cipher to linear and differential attacks has been mathematically proven.
5) The cipher plaintext and key avalanche scores are calculated.
6) The randomness of the cipher output is examined using the Statistical Test Suite (STS). The STS is the standard tool recommended by the National Institute for Standards and Technology (NIST) for evaluating the randomness of any given string.

### A. ORGANIZATION

The rest of this research paper is organized as follows: Section II is a literature review that highlights some attempts for introducing enhancements for the AES cipher and discusses some attempts of developing dynamic ciphers. Section III describes the details of the novel approach proposed by this study, where the modified transformations in P-AES are explained. Section IV presents the cipher performance results accompanied by a thorough analysis of the cipher strength, and practicality. Moreover, at the end of section IV, a brief comparison with existing work has been provided. Consequently, section V briefly concludes the paper. At the end of this manuscript, there are two appendices. The first appendix explains the steps used to calculate the avalanche scores. A complete example for the introduced cipher operation is provided in appendix II.

## II. LITERATURE REVIEW

The current trend in cryptology is to use open and standard ciphers. This preference can be attributed to three main reasons:

1) *To promote interoperability* - This is because if all systems have implemented standard ciphers interoperability can be easily granted.
2) *Testing* - Experts can focus their research on analyzing and testing few ciphers. Consequently, the user can be confident about the security of the used cipher.
3) *Compliance with Kerckhoff's principle*.

However, some experts have several concerns with using standard ciphers. For instance, C.B. Röllgen [9] stated that: ''Popular ciphers are always those that have been certified by authorities whose job mainly consists of gathering intelligence. There is a clear conflict of interest for these government organizations. Those professionals clearly know about the blatant deficiencies of the encryption algorithms that they certify''. This claim may sound extreme, but it is always better to be prepared for the worst.

Furthermore, although, using a standard cipher is indispensable for most applications, it can be argued that using a monomorphic cipher (i.e. a cipher that operates in the same way, every time) can give the potential opponent unnecessary leverage and may jeopardize the payload's confidentiality. This is due to the fact that the operation steps are the same every time. Thence, the opponent has all the time he or she needs to find any vulnerability that can be utilized to break the cipher [10]. This concern has been acknowledged implicitly by many cryptographers around the world as will be addressed in the upcoming subsection.

### A. MODIFIED VERSIONS OF THE ADVANCED ENCRYPTION STANDARD

Since Rijndael has been selected in 2001 as the AES, many research papers have been published to introduce a sort of modification or enhancement to the AES operation. For instance, in [11], it has been mentioned that a number of military or diplomatic applications apply small secret changes in the AES design. The idea is to build a new secret cipher that inherits the strength of the AES. In addition to the obvious conflict with Kerckhoff's principle [12], it has been proved that retrieving these changes is relatively easy [11].

Other researchers have suggested several radical changes in the AES design. For instance, in [13], they have suggested increasing the key length to 320 bits. Moreover, the number of rounds will be increased to 16 rounds. In addition, to increase the efficiency of the key generation stage, they have adopted the use of Polybius square to derive the encryption key from a password. There are two main issues with this model. Firstly, it is relatively easier to retrieve a password compared to retrieving an encryption key. This is because a password selected by the user will typically have minimal entropy, and can be retrieved using simple techniques such as social engineering [14]. Secondly, it is important to consider the current widespread of the AES implementations. In other words, software developers will be reluctant to apply expensive changes in the AES implementation such as the changes introduced in [13] because it may lead to rewrite the whole cipher code again, and may cause other unnecessary changes to their underlying systems. On the other hand, minimal changes that can be easily integrated with existing AES implementations will have better chances of acceptance.

Moreover, some researchers suggested an AES variant for securing a specific data type. In particular, in [15], a cipher has been designed primarily to enhance the AES performance and security for encrypting images. The design involves replacing the MixColumns transformation with chaotic mapping and Exclusive OR operation to reduce the required computation time. Furthermore, the rows of the S-Box are

circularly shifted by values extracted from the encryption key. The goal is to make the S-Box a dynamic entity and consequently equip the cipher with a layer of obscurity that can help in defeating attackers' efforts. This model has been designed primarily to enhance the AES performance on images. Needless to say, that a standard cipher needs to perform well with all potential data types.

Moreover, a considerable number of hybrid ciphers or protocols that involve the AES in a way or another have been suggested in [7], [16]–[18]. Apart from the involved complexity with implementing hybrid ciphers, using hybrids will typically incur additional computation which will degrade the encryption performance.

### B. DYNAMIC ENCRYPTION

A number of researchers investigated the possibility of designing dynamic ciphers. For instance, [19]–[21] attempted to alter the SubBytes stage in the AES operation. Thus, rather than using the same static Substitution Box (S-Box) introduced by the AES designers, they made the S-Box a dynamic entity that will be derived from the encryption key. It is claimed that all the properties of the AES S-Box such as the bit independence, avalanche criterion, and nonlinearity are met in all derived S-Boxes. Consequently, with every new key, the cipher will have a different shape. Obviously, this will hinder attackers' attempts to penetrate the cipher.

However, it can be argued that it is not judicious to manipulate the AES S-Box, because it has been selected with caution to keep the maximum values of prop-ratio and input-output correlation as minimal as possible. Consequently, the prop-ratio and input-output correlation of the AES S-Box are less than $2^{-6}$ and $2^{-3}$ respectively. These values play an essential role in determining the resistance of the cipher to differential and linear attacks. Using other S-Boxes from the space of 8-bits invertible S-Boxes will typically have the scores $2^{-5}$ to $2^{-4}$ for the maximum prop-ratio and $2^{-2}$ for the maximum input-output correlation [22]. Consequently, the overall resistance of the cipher against linear and differential attacks will decrease.

Another dynamic cipher has been suggested in [23] to build a polymorphic cipher that picks a different cryptographic suite (i.e. cipher, mode of operation, and key length) in a semi-random approach. The selected cryptographic suite will be dynamically determined according to values extracted from the encryption key. There are sixty different cryptographic suites. No one, except legitimate communicating parties, will be able to identify the currently used suite.

Nonetheless, it can be argued that this model requires sixty different cryptographic suites to be implemented. Hence, the practicality of this model might be questionable. Moreover, the overall performance of this model is slightly sluggish.

Another dynamic encryption approach is based on DES and matrices multiplication. In this model, the plaintext $x$ is initially multiplied in a binary invertible matrix $k_a$, which is generated according to the concepts of Network Coding and using an arbitrary integer positive value $D_a$. This multiplication process transforms $x$ to $z_1$. Consequently, the DES cipher is invoked to encrypt $z_1$. The outcome of this step is $z_2$. According to the authors' statement, the main reason for invoking the DES cipher is to "bring non-linearity". After that, another binary invertible matrix $k_c$ is generated and $z_2$ is multiplied in $k_c$ to get our ciphertext y. The details of generating $k_a$, $k_c$, as well as the routine used to update $k_c$, are thoroughly elaborated in [24].

An essential step suggested in the aforementioned study is to update the matrix $k_c$ before sending new messages. Consequently, even if the same message has been sent twice using this model, the outcome will be different, even without the help of the block cipher mode of operation. This change entitles the cipher to be called a dynamic cipher. The process of updating $k_c$ is called a partial key update. This is because the 64 bits DES key, $k_a$, and $k_c$ are together used as a key for this new cipher.

According to the statement of the authors, the performance of this cipher is comparable to the 3-DES cipher. Nevertheless, 3-DES performance was never considered acceptable. In fact, one of the main motivations for the AES competition is to overcome the sluggishness of the 3-DES cipher [25]. Moreover, the choice of the DES cipher in the intermediate layer is hard to understand, especially when you have other secure and efficient alternatives. Furthermore, according to the statement of this cipher authors, the security of this model needs further analysis, which has been postponed as future work. Hence, it is early to assure the strength of this cipher in terms of security.

Another example of a dynamic encryption algorithm has been introduced in [26]. It may be worth noting that this cipher also belongs to the family of lightweight ciphers. In this algorithm, the main concern is to enhance the performance of the cipher by reducing the number of rounds to only one round. This is because according to the authors' claim, several delay-sensitive applications cannot tolerate the delay introduced by typical ciphers including the AES. Furthermore, the cipher must maintain adequate security levels to resist all known attacks.

It is assumed that the communicating parties have exchanged a secret Session Key (SK) a priori of establishing their communication. Using SK, an XOR operation is carried out with 512 bits nonce. The resulting 512 bits are hashed using SHA-512. The result will also be 512 bits. These bits will change with every new nonce. Hence, it will be called the Dynamic Key (DK). DK is divided into 5 sub-keys: $\{k_{S1}, k_{S2}, k_P, k_{RK}, k_{SRK}\}$. $k_{S1}$ and $k_{S2}$ are used to construct two different key-dependent substitution tables $S_1$ and $S_2$ using the key setup algorithm of the RC4 cipher. $k_P$ is used to construct a permutation table $\pi$. $k_{RK}$ will seed a stream cipher to generate a random sequence of bits. These randomly generated bits are divided into $m$ blocks, where $m$ represents the number of blocks of the plaintext. Every block of these $m$ blocks will be used as a sub-key to be XORed with one block in the plaintext. The $k_{SRK}$ will be used to generate

a selection table that specifies which sub-key to be XORed with which plaintext block. All the cipher's building blocks are key-dependent. Hence, any change in any part of the key will lead to a major and unpredictable change in the output.

The cipher processes two blocks at a time. The first block is XORed with a sub-key selected using the selection table. Consequently, the result undergoes a byte substitution process using $S_1$. The outcome of this substitution is XORed with the second plain block and the result undergoes a byte substitution process using $S_2$. The result is the ciphered version of the first block. The second plain block undergoes a slightly different process [26].

The authors of the latter approach claim that the cipher demonstrates an adequate security level in resisting all attacks including linear and differential attacks. However, it is known that the repetition of the transformations of any cipher (i.e. rounds) is the only approach to decrease propagation ratio and correlation for linear trails. Therefore, there is no compelling argument to support the claim of resisting linear or differential analysis.

To summarize, it can be said that most of the existing research work has at least one of the following issues:

1) High implementation cost.
2) Explicit conflict with Kerckhoff's principle and interoperability issues.
3) Intolerable performance degradation.
4) A monomorphic design that repeats the same steps with every different input.
5) Questionable security.

A thorough discussion about the limitations of existing static and dynamic ciphers, and the need to devise a robust dynamic (i.e. polymorphic) cipher is available in the review paper [27].

This research aims at suggesting a cipher that can provide a practical solution to all these issues. This will be achieved by designing the P-AES. P-AES is a polymorphic variant of the AES that operates in 128 different ways. The P-AES exact operation details will be determined during execution time only for communicating parties using some of the key bits. Because the P-AES can operate in different ways, the attacker will need more time not only to retrieve the encryption key but also to determine the exact shape of the cipher. Further details will be provided in the following section.

## III. THE POLYMORPHIC ADVANCED ENCRYPTION STANDARD (P-AES)

### A. PRELIMINARIES

Firstly, in the traditional AES, the input is divided into $n$ blocks of size 16 bytes. In many cases, the size of the last block will be less than 16 bytes, therefore a padding scheme (e.g. PKCS 7) is invoked. The proposed cipher processes one block of size 16 bytes at a time. This input block is copied into the *state* matrix as follows:

$$\begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix}$$

For simplicity, the state content (e.g. $S_{0,0}$) will be represented in hexadecimal.

The block cipher mode of operation specifies the way in which all the $n$ blocks are processed by the cipher. There are five modes recommended by the NIST [28]. These modes are the Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR). The ECB mode of operation is insecure because it reveals the patterns of the plaintext [29]. Apart from the insecurity of the ECB, there is no vulnerability reported in the other recommended modes of operation. In the implementation of this study, the CBC mode of operation has been used. However, any other mode of operation can be used in the same manner.

The proposed P-AES can support keys of the lengths 16, 24 or 32 bytes. However, for consistency, it is assumed that the key length is 32 bytes in the rest of the discussion.

$$key = [byte_0|byte_1|byte_2|\ldots byte_{28}|byte_{29}|byte_{30}|byte_{31}]$$

After the key has been generated, the following values are calculated, in both the sender and receiver sides, as follows:

$$bytes\_substitution\_index = \left(< int > byte_{31}\right) mod 8 \quad (1)$$

$$row\_shifting\_index = \left(< int > byte_{30}\right) mod 4 \quad (2)$$

$$column\_mixing\_index = \left(< int > byte_{29}\right) mod 4 \quad (3)$$

Obviously,

$$byte\_substitution\_index \in \{0, 1, 2, 3, 4, 5, 6, 7\},$$
$$row\_shifting\_index \in \{0, 1, 2, 3\}, \quad and$$
$$column\_mixing\_index \in \{0, 1, 2, 3\}$$

The P-AES operation details such as the number of rounds and the key scheduling process are similar to the traditional AES.

### B. AES SUBBYTES STAGE vs. P-AES MODIFIEDSUBBYTES STAGE

In the traditional AES, all the rounds, except round$_0$, include a SubBytes stage. The traditional AES has an S-Box with good algebraic properties designed to maximize the confusion of the state. Let us assume that one of the bytes in the state array has the binary value $(10111001)_2$. The hexadecimal representation for this byte is $(B9)_{16}$. This value will be substituted with a different value from the S-Box. To perform the substitution process, the left-most 4 bits $(B)_{16}$ will be the row index, and the right-most 4 bits $(9)_{16}$ will be the column index (see figure 1). Therefore, the byte $(B9)_{16}$ will be substituted with the value located in row $(B)_{16}$ column $(9)_{16}$ in the S-Box which is $(56)_{16}$. In the decryption module, the same steps are carried out. However, instead of using the S-Box, the inverse S-Box will be used.

To understand the P-AES Modified SubBytes stage, let us assume that the value of the byte_substitution_index which has been extracted from the key according to equation (1) is 2. In the P-AES ModifiedSubBytes stage, the bits of every

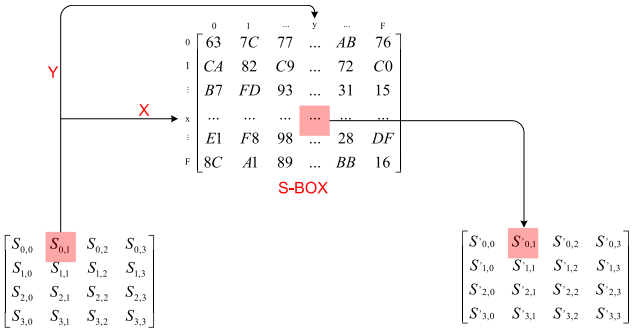


**FIGURE 1.** SubBytes Stage in the traditional AES operation [25].

byte in the state entries will be shifted circularly to the left by $7-byte\_substitution\_index$. For instance, by shifting the bits of the byte $(10111001)_2$ circularly to the left by $7-2$ (i.e. 5), it will be $(00110111)_2$. Representing it in hexadecimal will be $(37)_{16}$. After looking it up in the S-Box, the result is $(9A)_{16}$.

In the decryption process, the state entries will be looked up in the inverse S-Box. After that, a circular shift of the bits of the state entry will be done to the right by the value $7 - byte\_substitution\_index$. To clarify, let us apply inverse ModifiedSubBytes to the byte $(9A)_{16}$ given that the byte_substitution_index is 2. Firstly, we should look up $(9A)_{16}$ in the inverse S-Box. The result is $(37)_{16}$ or $(00110111)_2$. Now, we shall circularly shift the bits of this byte to the right by $7 - byte\_substitution\_index$ (i.e. 5). The result is $(10111001)_2$ or $(B9)_{16}$.

The pseudocodes depicted in algorithm 1 and algorithm 2 describe the steps of the P-AES ModifiedSubBytes stage in the encryption and decryption operations:

**Algorithm 1** P-AES ModifiedSubBytes Stage (Encryption)

**Data:** State Matrix S
**Result:** New State Matrix S'
**for** *i=0:3* **do**
  **for** *j=0:3* **do**
    $S_{i,j} \leftarrow$ CyclicallyLeftShift($S_{i,j}$,7-ByteSubstitutionIndex );
    $S_{i,j} \leftarrow$ S-Box($S_{i,j}$);
  **end for**
**end for**

*Rationale*: The AES S-Box has been selected with caution to meet various important specifications. Therefore, rather than altering the S-Box entries in the proposed P-AES, it is better to shuffle the state entries before submitting them to the substitution stage. In this way, the strength of the S-Box is preserved, and the required polymorphism is introduced to this stage.

**Algorithm 2** P-AES ModifiedSubBytes Stage (Decryption)

**Data:** State Matrix S
**Result:** New State Matrix S'
**for** *i=0:3* **do**
  **for** *j=0:3* **do**
    $S_{i,j} \leftarrow$ Inverse S-Box($S_{i,j}$);
    $S_{i,j} \leftarrow$ CyclicallyRightShift($S_{i,j}$,7-ByteSubstitutionIndex );
  **end for**
**end for**

$$\begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} \rightarrow \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,1} & S_{1,2} & S_{1,3} & S_{1,0} \\ S_{2,2} & S_{2,3} & S_{2,0} & S_{2,1} \\ S_{3,3} & S_{3,0} & S_{3,1} & S_{3,2} \end{bmatrix}$$

**FIGURE 2.** ShiftRows stage in the traditional AES operation.

$$\begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} \rightarrow \begin{bmatrix} S_{0,1} & S_{0,2} & S_{0,3} & S_{0,0} \\ S_{1,2} & S_{1,3} & S_{1,0} & S_{1,1} \\ S_{2,3} & S_{2,0} & S_{2,1} & S_{2,2} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} = \begin{bmatrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \end{bmatrix}$$

**FIGURE 3.** P-AES ModifiedShiftRows stage with row_shifting_index equals 3.

### C. AES SHIFTROWS STAGE vs. P-AES MODIFIEDSHIFTROWS STAGE

The main steps of the ShiftRows stage in the conventional AES are depicted in figure 2. At this stage, the rows of the state matrix are shifted as follows:

- No change will occur to $row_0$.
- $row_1$ will be circularly shifted to the left by one byte.
- $row_2$ will be circularly shifted to the left by two bytes.
- $row_3$ will be circularly shifted to the left by three bytes.

The purpose of this stage in the traditional AES is to provide the diffusion property by assuring that the bytes of each column in the state will be distributed among all the state's columns.

To understand the P-AES Modified ShiftRows, let us assume that the value of the row_shifting_index, which has been extracted from the key using equation (2), is 3. In this case, the ModifiedShiftRows stage will operate as follows:

- No change will occur to $row_3$.
- $row_0$ will be circularly shifted to the left by one byte.
- $row_1$ will be circularly shifted to the left by two bytes.
- $row_2$ will be circularly shifted to the left by three bytes.

All of the above steps are depicted in figure 3 and algorithm 3.

In other words, the value of the row_shifting_index will determine the first row that will not be shifted. Next rows (annularly) will be circularly shifted to the left by the

**Algorithm 3** P-AES ModifiedShiftRows Stage (Encryption)

**Data:** State Matrix S
**Result:** New State Matrix S'
1 **for** *i=RowShiftingIndex:(RowShiftingIndex+3)mod4* **do**
2 | $row_i \leftarrow$ ShiftBytesToTheLeftCyclically($row_i$,*i*);
3 **end for**

offset 1, 2, or 3 respectively. Similarly, in the decryption module, the value of the row_shifting_index will be retrieved. This value specifies the state row that will not be shifted. Subsequent rows (circularly) will be shifted annularly to the right by 1, 2, or 3 bytes respectively.

**Algorithm 4** P-AES ModifiedShiftRows Stage (Decryption)

**Data:** State Matrix S
**Result:** New State Matrix S'
1 **for** *i=RowShiftingIndex:(RowShiftingIndex+3)mod4* **do**
2 | $row_i \leftarrow$ ShiftBytesToTheRightCyclically($row_i$,*i*);
3 **end for**

The pseudocodes in algorithms 3 and 4 depict the steps of the P-AES ModifiedShiftRows Stage in the encryption and decryption operations respectively.

*Rationale:* The ModifiedShiftRows stage will preserve the strength of the traditional AES ShiftRows because the columns' bytes in the state matrix will be scattered equivalently across all other columns in the state. However, instead of shifting the rows in the same way with every different key, the exact ModifiedShiftRows methodology of the proposed P-AES will be determined in execution time with every new key.

### D. AES MIXCOLUMNS STAGE vs. P-AES MODIFIEDMIXCOLUMNS STAGE

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} = \begin{bmatrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \end{bmatrix}$$

**FIGURE 4.** MixColumns stage in the traditional AES operation.

At this stage in the conventional AES, the state matrix is multiplied in the MixColumns matrix as depicted in figure 4. For decryption, the inverse matrix is used in the multiplication as depicted in figure 5. It is worth mentioning that the multiplication process is carried under Galois Field $(2^8)$. Obviously, this procedure has a strong diffusion effect because, for every column, all its bytes will contribute unequally to determine the new value of the bytes of that column.

The key difference between the traditional AES MixColumns and the proposed P-AES ModifiedMixColumns

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \times \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{bmatrix} = \begin{bmatrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \end{bmatrix}$$

**FIGURE 5.** Inverse MixColumns stage in the traditional AES operation.

is that the order of the ModifiedMixColumns matrix rows will be determined during execution time depending on the value of the column_mixing_index. For instance, assume the value of the column_mixing_index that has been calculated using equation (3) equals 3. In this case, $row_3$ in the default MixColumns matrix will be $row_0$ in the ModifiedMixColumns matrix. Similarly, $row_0$,$row_1$ and $row_2$ in the traditional AES MixColumns matrix will be $row_1$, $row_2$ and $row_3$ in the proposed P-AES ModifiedMixColumns matrix respectively. Hence, the ModifiedMixColumns matrix will look as follows:

$$\begin{bmatrix} 03 & 01 & 01 & 02 \\ 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \end{bmatrix}$$

For decryption, the inverse ModifiedMixColumns matrix is reordered in the same manner. To elucidate, if the Column_Mixing_Index equals 3, the reverse ModifiedMixColumns matrix will be:

$$\begin{bmatrix} 0B & 0D & 09 & 0E \\ 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \end{bmatrix}$$

The pseudocodes depicted in algorithms 5 and 6 describe the steps of the P-AES ModifiedMixColumns Stage in the encryption and decryption operations respectively.

**Algorithm 5** P-AES ModifiedMixColumns Stage (Encryption)

**Data:** State Matrix S
**Result:** New State Matrix S'
1 **foreach** $row_i \in MixColumnsMatrix$ **do**
2 | $row_i \leftarrow row_{(i+ColumnMixingIndex)mod4}$;
3 **end foreach**
4 *Apply the AES MixColumns using the updated MixColumnsMatrix*

*Rationale:* The desired effect of the MixColumns stage in the conventional AES operation has been preserved in the proposed P-AES ModifiedMixColumns transformation. Nevertheless, instead of using the same static matrix with every different input, the order of the rows is changed unpredictably. This is an additional obstacle to hinder potential attacks.

**Algorithm 6** P-AES ModifiedMixColumns Stage (Decryption)

**Data:** State Matrix S
**Result:** New State Matrix S'
**foreach** $row_i \in InverseMixColumnsMatrix$ **do**
  $row_i \leftarrow row_{(i+ColumnMixingIndex)mod4}$;
**end foreach**
*Apply the AES InverseMixColumns using the updated InverseMixColumnsMatrix*

### E. THE P-AES FULL OPERATION

Apart from the differences highlighted in previous subsections, the overall structure of the P-AES cipher is quite similar to the AES structure. In fact, one of the cardinal goals during the process of designing this polymorphic cipher is to inherit the strength of the AES. Figures 6 and 7 depict the overall operation of the encryption and decryption operations. The reader should note that we assumed that the key length is 32 bytes. Consequently, we shall have 14 rounds. Just like the AES operation, the number of rounds may change with the length of the used key.

## IV. EXPERIMENTAL RESULTS AND ANALYSIS

### A. PRELIMINARIES

The specifications of the testing machine are as follows:

- Intel Core i7-6500 CPU @ 2.50GHz (4 CPUs), ~2.6GHz.
- Memory: 16 GB.
- Storage: 2 TB.

The algorithm has been implemented as a console application using the C++ compiler deployed in Visual Studio 2010. Some of the utilities provided by the CryptoPP library such as the HexEncoder and the AutoSeededRandomPool have been utilized. However, codes for traditional AES and the proposed P-AES implementations were written to accurately benchmark the performance of the proposed P-AES compared to the traditional AES. The functions QueryPerformanceCounter() and QueryPerformanceFrequency() are both used to measure the time elapsed for encryption and decryption.

### B. TIME EXECUTION PERFORMANCE

Performance is a critical factor that can determine the success or failure of any cipher. Even the strongest cipher will be deemed impractical if it has poor performance. To illustrate, it is worth mentioning that during the process of selecting the traditional AES, although other ciphers, such as Serpent, were more secure compared to Rijndael, the judges voted for the latter mainly due to the high performance it provides [30]. Hence, although a number of cardinal changes have been applied to the operation of the traditional AES, preserving the high-performance nature of the AES was one of the pivotal considerations in the design of the proposed P-AES.

Figure 8 depicts the time required to encrypt and decrypt the payload using the traditional AES versus the proposed

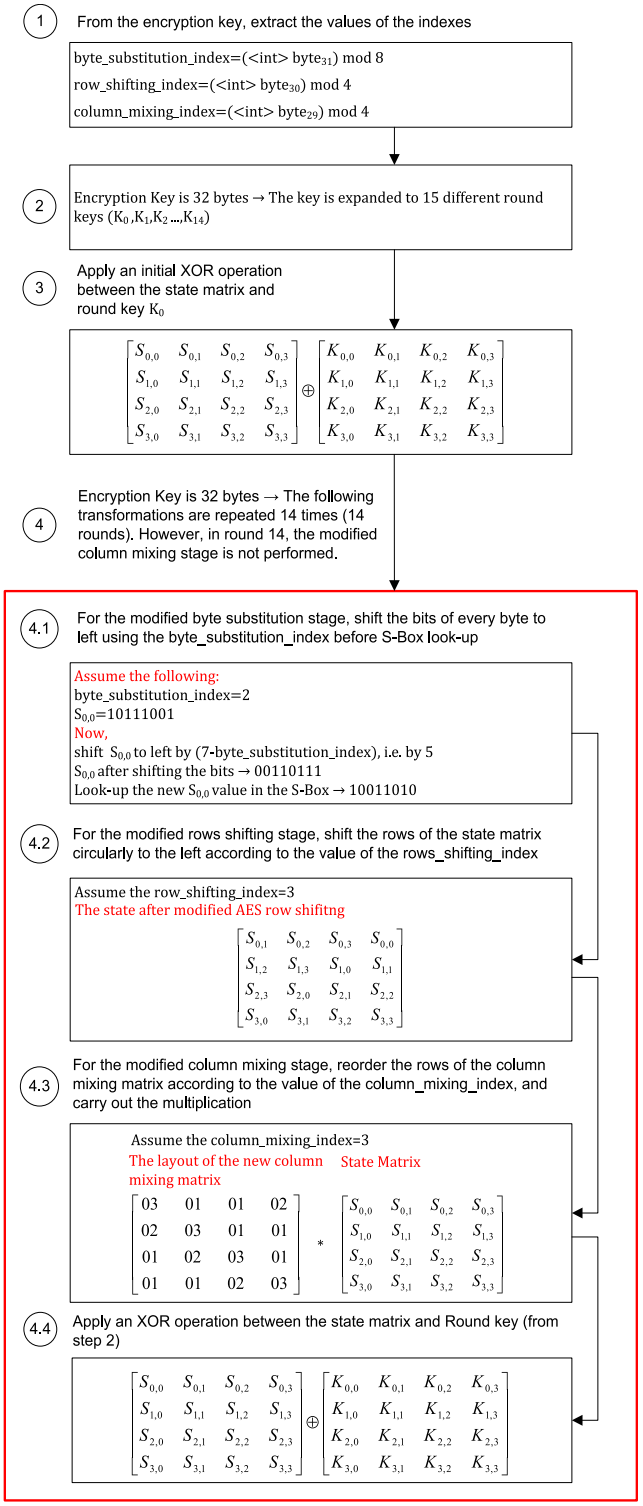


**FIGURE 6.** P-AES encryption operation.

P-AES both in the CBC mode of operation. Three different dummy inputs were tested with the sizes 500, 1000, and 2000 bytes. To make sure that the results were consistent, the implementation had been executed 1000 times with both ciphers (i.e. AES-CBC and P-AES-CBC), and every input

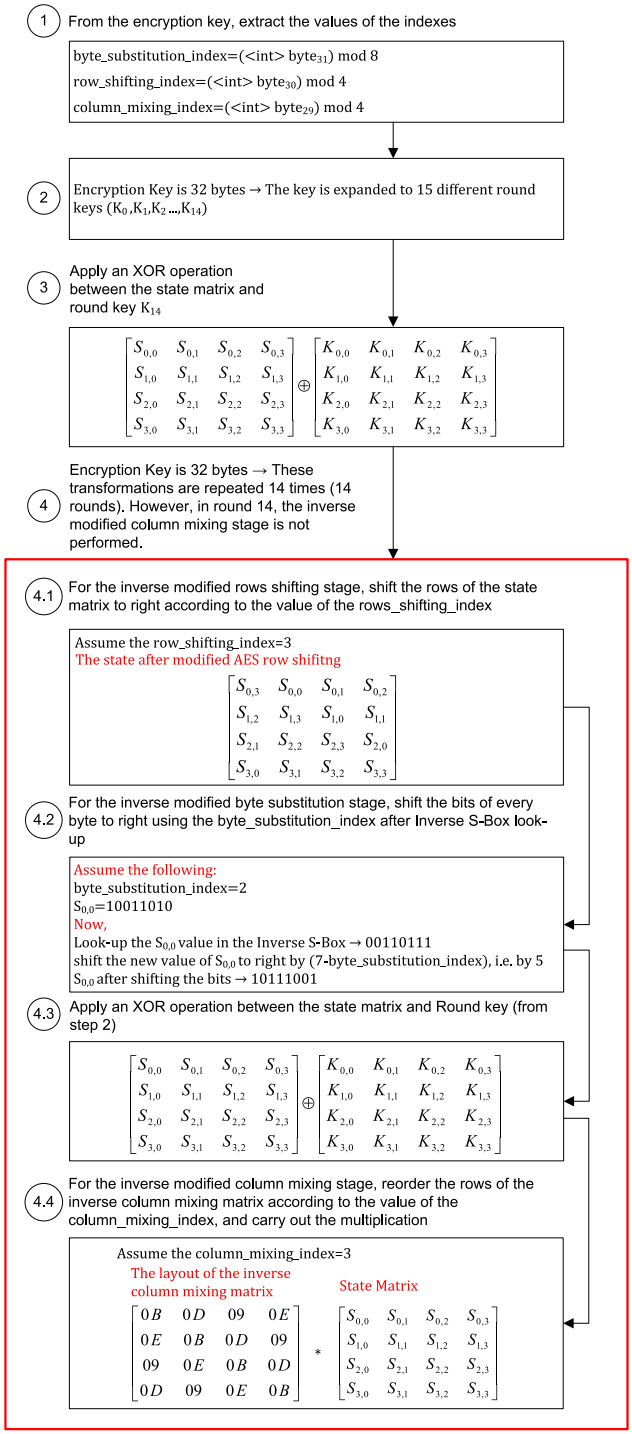


**FIGURE 7.** P-AES decryption operation.

size (500, 1000, and 2000 bytes). After that, the average had been calculated.

As can be seen, the time required to encrypt 500 bytes using the traditional AES-CBC is 70 microseconds, while it takes 71 microseconds to encrypt the same payload using the proposed P-AES-CBC. Other readings can be interpreted similarly. Needless to say, that other AES implementations can provide different performance readings.

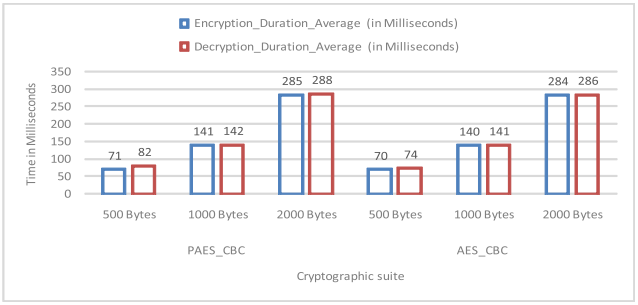


**FIGURE 8.** Comparing the performance of the traditional AES and the Introduced Cipher.

According to the depicted readings, the performance of the proposed P-AES is slightly less than the conventional AES performance. This delay can be attributed to the additional required processing that accompanies the modified proposed cipher. Nevertheless, the introduced overhead is insignificant and tolerable for most applications and platforms.

### C. RESISTANCE TO LINEAR AND DIFFERENTIAL ATTACKS

The notations and definitions used in this subsection is similar to the notations and definitions used to prove the AES resistance to linear and differential attacks. For a better understanding of the rest of this subsection, the reader is advised to read section 8.2 in [22] before proceeding.

The Propagation Ratio (prop ratio) for any differential trails can be approximated by the product of the prop ratios of its active S-Boxes [22], [31].

The correlation for the linear trail can be approximated by the product of the input-output correlation of its active S-Boxes [22], [31].

The proposed P-AES uses the same traditional AES S-Box. This S-Box has been selected with caution to provide the minimal prop ratio and input-output correlation. These values are $2^{-6}$ and $2^{-3}$ respectively.

In the rest of this subsection, we shall prove that the minimum number of active S-Boxes in the proposed P-AES, up to its fourth round is 25 which gives a prop ratio and a correlation of $2^{-6\times25}$ and $2^{-3\times25}$ respectively. That is $2^{-150}$ and $2^{-75}$. This is significantly sufficient to resist the differential and linear attacks [22], [31].

*Definition:* The branch number for the linear transformation $F$ is:

$$min_{a\neq0}\left(W\left(a\right)+W\left(F\left(a\right)\right)\right)$$

where:

$F$: is a linear transformation.

$a$: is a byte vector.

$W(a)$: the weight of the vector $a$. This is measured by the number of non-zero bytes in a pattern.

*Lemma 1:* The branch number of the ModifiedMixColumns is lower bounded by 5.

*Proof:* Assume we have a pattern $a$ with only one active byte in the initial state (that is $W\left(a\right) = 1$). Then, $W\left(ModifiedMixColumns\left(a\right)\right) = 4$. This is because

the P-AES ModifiedMixColumns will diffuse this active byte to 4 different bytes in the following state. According to the definition of the branch number provided in definition 1, the branch number of the ModifiedMixColumns transformation is lower bounded by 5 ■

Let $a$ be a pattern that represents the *difference activity pattern* or the *correlation activity pattern*.

The column weight ($W_c(a)$) is the number of active columns in $a$. An active column is a column with at least one active byte (non-zero byte in the pattern $a$).

The byte weight of column $j$, ($W(a)|_j$) is the number of active bytes in the column $j$.

Note that the ModifiedSubBytes and the AddRoundKey stages do not affect the propagation of the active bytes. Hence, we shall focus solely on the ModifiedShiftRows stage followed by the ModifiedMixColumns stage.

From lemma 1, the ModifiedMixColumns branch number is lower bounded by 5. The ModifiedShiftRows diffuse the bytes of every column to all the states 4 columns. Hence:

$$\forall j \in \{0,1,2,3\} : W_c\left(\acute{a}\right) \geq max\left(W(a)|_j\right) \quad (4)$$

$$\forall j \in \{0,1,2,3\} : (W_c(a)) \geq max\left(W(\acute{a})|_j\right) \quad (5)$$

where $a$ and $\acute{a}$ are the same pattern before and after the ModifiedShiftRows transformation respectively.

Now, we shall use the following notation. The initial round is round 1. The pattern in the round $i$ is denoted by $a_{i-1}$. After the ModifiedShiftRows the pattern will be called $b_{i-1}$. In the following round, the pattern $b_{i-1}$ after ModifiedMixColumns is $a_i$.

The weight of the $m$ round trail is given by the sum of weights $a_0$ to $a_{m-1}$.

*Theorem 1:* The weight of 2 round trail with $C$ active columns at the beginning of the second round is lower bounded by $5C$.

*Proof:* The ModifiedMixColumns has branch number 5. Consequently, $(W(b_0)+W(a_1)) \geq 5$. Therefore, if the column weight of $a_1$ is $C$, this implies $(W(b_0) + W(a_1)) \geq 5C$. Because the ModifiedShiftRows will not impact the byte weight, therefore $W(b_0) = W(a_0)$. Consequently, $(W(a_0) + W(a_1)) \geq 5C$ ■

*Lemma 2:* The sum of the active columns in the input and output of a 2-round trail is lower bounded by 5.

*Proof:* From inequalities (4) and (5):

$$\forall j \in \{0,1,2,3\} : W_c(a_i) \geq W(b_i)|_j.$$
$$\forall j \in \{0,1,2,3\} : W_c(b_i) \geq W(a_i)|_j.$$

At least 1 column in $a_1$ (similarly $b_0$) is active. Let us assume that this active column is column $g$. As proven in Lemma 1, the ModifiedMixColumns has branch number 5.

So,
$\left(W(b_0)|_g + W(a_1)|_g\right) \geq 5.$
But, $W_c(a_0) \geq W(b_0)|_g$ and $W_c(b_1) \geq W(a_1)|_g$.
So, $W_c(a_0) + W_c(b_1) \geq 5$
Since, $W_c(b_1) = W_c(a_2)$
Consequently $W_c(a_0) + W_c(a_2) \geq 5$ ■

*Theorem 2:* The number of active bytes in a trial of 4 or more rounds is lower bounded by 25

*Proof:* Applying theorem 1 on (round 1, round 2) and (round 3, round 4) implies:

The byte weight of a 4 rounds trial $\geq 5 \times (W_c(a_1) + W_c(a_3))$.

From lemma 2, $(W_c(a_1) + W_c(a_3) \geq 5$.

So, the byte weight of a 4 rounds trial $\geq 25$ ■

### D. AVALANCHE CRITERION

One of the most important criteria in assessing the strength of any new cipher is to measure how much the cipher output will change if a small change is introduced to the input. This property is called the avalanche criterion. It is highly desirable in any cipher to have an avalanche score in the range $0.5 \mp \varepsilon$ [32]. This means, if only one bit is changed in the cipher input, then every bit in the cipher output has a probability of $0.5 \mp \varepsilon$ to change its value. Obviously, the cipher has two inputs: the *plaintext* and the *encryption key*. Therefore, the key and plaintext avalanche scores should be inspected for the proposed P-AES cipher when one bit is changed in the key (Key Avalanche) and in the plaintext (Plaintext Avalanche). This property has been investigated in the P-AES, and the scores were 0.496 and 0.504 for the key avalanche and the plaintext avalanche respectively. The measures used to calculate these two scores are elaborated in appendix I.

### E. OBSCURITY LAYER

In the proposed P-AES design, the main contribution is hiding the exact parameters' values from everyone except the sender and the legitimate receiver. The proposed P-AES cipher can operate in 128 different ways. This is because, as has been described in section III, subsection A, the byte_substitution_index can have 8 different values, the row_shifting_index and the column_mixing_index parameters can each have 4 different values. The product of the possible values for these parameters is 128, which represents the number of different shapes of the proposed P-AES cipher. As a result, the attacker cannot assert which version of the AES he or she is trying to attack.

As proved in section IV subsection C, each of these 128 versions is as strong as the AES. However, let us assume that one of the 128 versions of the P-AES has been somehow broken. The probability that a given message has been encrypted using the vulnerable P-AES version is $1/128$ which is 0.0078125. Given that no one except legitimate communicating parties is aware of which version of the P-AES has been used, the attacker has nothing to do except blindly launching his attack hoping that the current message is encrypted using this vulnerable version of P-AES. However, the probability of failure due to missing the vulnerable version of P-AES is 0.9921875. Hence, if the cost of this hypothetical attack is high, the attacker will be less urged to launch it due to the high probability of failure.

### F. RESISTANCE TO SIDE CHANNELS ATTACKS

It has been realized during the New European Schemes for Signatures, Integrity, and Encryption (NESSIE) and the AES processes that even the strongest cipher will be deemed vulnerable unless it has been implemented in a secure fashion that preserves its theoretical robustness [33]. Nevertheless, the proposed P-AES cipher have an innate resistance to several implementation attacks. To clarify, the reader is reminded that any monomorphic cipher repeats the same steps with every input regardless of the value of the key. Hence, disclosing the exact operation details of a cipher combined with a weak implementation for the cipher may enable the attacker to extract some information about the encryption key. However, the proposed P-AES is a polymorphic cipher that operates differently with almost every new key. Consequently, instead of generously sharing the exact operation details with potential attackers, they will be forced to work in identifying which version of the AES they are dealing with.

### G. THE RANDOMNESS OF THE CIPHER OUTPUT

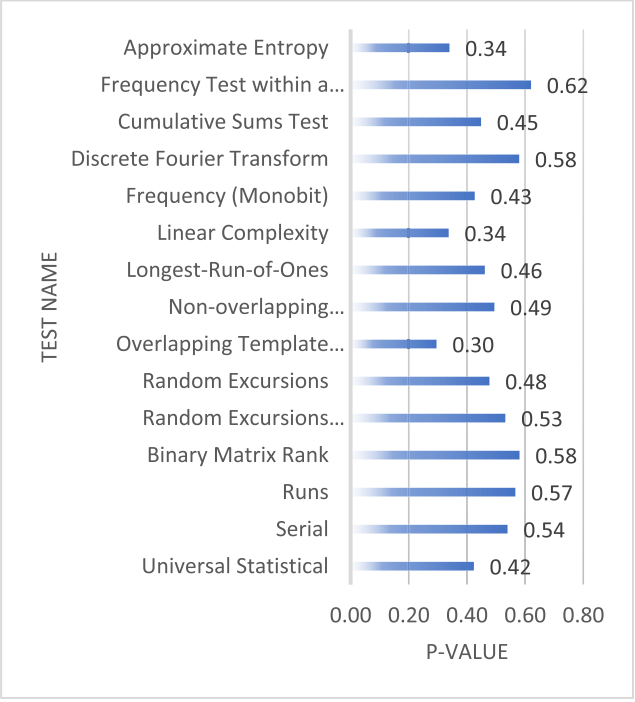


**FIGURE 9.** Examining the randomness of the P-AES Cipher output when all inputs are 0's using the NIST statistical test suite.

During the AES competition, one of the criteria that has been used to evaluate candidate ciphers is the cipher's capability to work as a Pseudo-Random Number Generator (PRGN) [34]. This property will not only ensure that the cipher can act as a PRNG, but it will also indicate that the cipher output is statistically indistinguishable from the

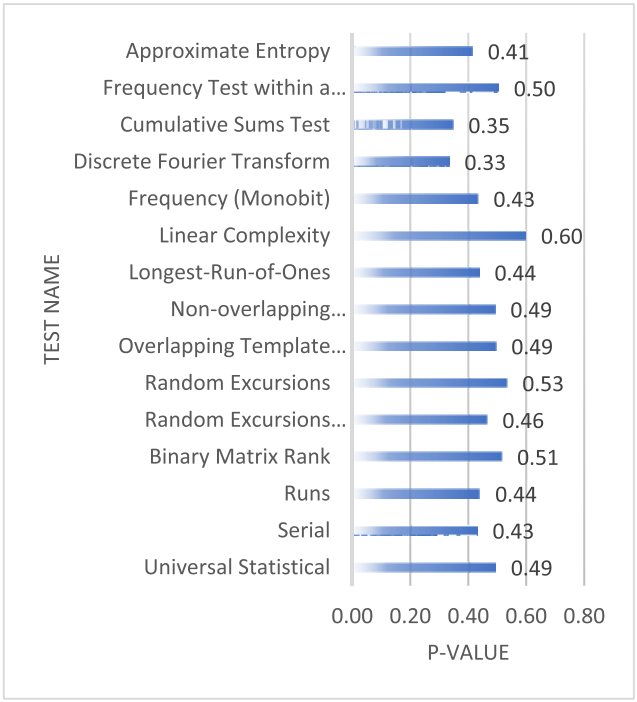


**FIGURE 10.** Examining the randomness of the P-AES Cipher output when all inputs are 1's using the NIST statistical test suite.

random output. To test this property, the Statistical Test Suite (STS) recommended by the NIST has been used [35].

This suite has 15 different tests. For every test, a calculated probability (P-Value) is calculated. The cipher passes any particular test if the P-Value is in the range $0.01 \leq P \leq 1$ [35]. As per the guidelines of the NIST, the null hypothesis is that the sequence being tested is random [35]. All these tests have been carried on a significance level of 0.01. This means the probability of rejecting the null hypothesis while it is true is 0.01.

In the used STS implementation, the default length of the input stream is 131,072 bytes. However, to get accurate results, it is recommended to pass larger input, and the STS will process it in chunks of 131,072 bytes. Therefore, in this research, the chosen input length was 10,048,576 bits (1,310,720 bytes). With this input, the STS will iterate (i.e. repeat the experiment) ten times for each block of 131,072 bytes. The results depicted in figures 9 and 10 are the average of the iterations' results for each test.

The experiment has been carried twice with two different inputs. In the first attempt, the STS input stream is generated by encrypting 1,310,720 bytes using the P-AES. The key and IV lengths are 32 and 16 bytes respectively. All the bits of the plaintext, key, and IV are zeros. The second attempt is identical to the first one. However, the plaintext bits, key bits, and IV bits were all ones.

After encrypting both plaintexts using the P-AES, the randomness of the ciphertext is tested using the 15 different tests

of the STS. An elaborative description and detailed examples for these tests are available in [35].

All these tests are implemented in software. In fact, there are several implementations for this suite. However, in this research paper, an optimized Linux compatible implementation is used [36]. The user should submit a file of the cipher's output, which can be represented as ASCII binary (i.e. zeros and ones) or raw binary, which will render the cipher output as unintelligible gibberish. Because the latter format is the default choice, the cipher output has been submitted in the raw binary format.

As illustrated in figure 9 and figure 10, the proposed P-AES cipher has passed all the tests. It is worth mentioning that even random number generators have a high probability of failing some of these tests [37].

### H. PRACTICALITY

It is judicious to consider the widespread usage of the traditional AES before suggesting a new cipher. Needless to say, that most developers will be reluctant to adopt a new cipher if that will require them to implement the whole encryption algorithm from scratch. However, the proposed P-AES cipher can be integrated seamlessly with several existing traditional AES software implementations. To elucidate, in the implementation developed for this study, the same transformations' functions defined in the traditional AES cipher have been used. However, the state matrix is manipulated to accommodate the required effect.

Moreover, the interoperability will remain intact, because, although, the exact operation details will be obscured from outsiders during execution time, communicating parties can accurately retrieve the exact parameters' values from the encryption key bits, as has been described in Section III, Subsection A.

That being said, the P-AES implementation might not be feasible in some circumstances. For instance, the traditional AES implementation is currently integrated with many processors in the form of an instruction set such as the Advanced Encryption Standard New Instructions (AES-NI) in Intel and AMD processors. The applications can use the traditional AES by invoking the appropriate instruction and the processor will carry the needful encryption or decryption operations. Therefore, for instance, if a given application uses the AES-NI, then the developers of that application must write the whole P-AES code if they decided to use the P-AES.

Moreover, in some circumstances, the AES implementation might be written in a compact fashion that cannot be easily adjusted to the P-AES.

To conclude, the potential of reusing the AES components to develop the P-AES in a given platform depends on the technique used for providing the encryption service.

### I. P-AES vs. AES

The AES is a static cipher. According to [38]–[40], static ciphers might be vulnerable to future attacks that target the cipher static nature. The P-AES dynamic or polymorphic structure overcomes this limitation. This is because as described in section IV, subsection E, the P-AES can operate in 128 different ways. The attacker will not be able to know which version is used to secure the current message.

In terms of security, as proven in section IV, subsections C, D and G, each of these 128 versions will have the same AES strength in resisting linear and differential attacks. In addition, the P-AES satisfies the other security requirements such as the avalanche criterion and the output randomness. Moreover, the P-AES dynamic structure makes it more capable of resisting implementation attacks.

**TABLE 1. Comparing the P-AES with existing dynamic ciphers.**

| Existing Dynamic Cipher | Shortcomings | Resolution in the P-AES |
|---|---|---|
| AES with a dynamic S-Box [19-21] | The AES S-Box has been chosen to provide the least possible prop-ratio and input-output correlation. Altering the AES-S-Box will increase these values which in turn will decrease the cipher resistance to linear and differential attacks. | - Instead of altering the S-Box, cyclically shift the state byte by an obscured value extracted from the key. Consequently, the same S-Box introduced in the traditional AES is used.<br>- Introduce polymorphism not only to the byte substitution phase but also to the other transformations (ShiftRows and MixColumns). |
| Multi-shape cipher [23] | - To use this cipher, it is needed to implement sixty different cryptographic suites.<br>- The performance of this model is sluggish compared to the AES. | - The existing implementation of the AES transformations can be utilized to implement the P-AES cipher. The performance of the P-AES is very close to the AES performance. |
| A network coding and DES-based dynamic encryption scheme [24] | - Poor performance compared to the AES.<br>- According to the authors' statement, further security analysis is required to assure the security of the cipher. | - The performance of the P-AES is very close to traditional AES performance.<br>- Apart from the obscurity it provides, the P-AES minimum security will be equal to the traditional AES security. |
| Dynamic key-dependent cipher [26] | - The cipher has only one round. It is known that the repetitions of the encryption transformations (rounds) are a reliable approach to decrease the prop-ratio and input-output correlation. Hence, the resistance of the cipher against linear and differential attacks might be questionable. | - The P-AES has 10, 12, or 14 rounds. Similar to the AES design, these rounds will significantly decrease the prop-ratio and input-output correlation to ensure sufficient resistance against linear and differential attacks. |

**TABLE 2.** Steps for calculating the key avalanche.

| Step Description | Value | Number of Changed Bits |
|---|---|---|
| Master Encryption Key ($K_0$) | 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 0000000000000 | N.A. |
| Initialization Vector | 00000000000000000100000010000 00011000001000000010100000 1 10000001110000100000000 10010 00010100000101100001100000 0 11010000111000001111 | N.A. |
| Plaintext: | Lorem ipsum dol | N.A. |
| Plaintext (Binary): | 01001100011011110111001011 1 00101011011010010000001 1010 01011100000111001101110 1010 11011010010000001100100011 0 111101101100 | N.A. |
| Plaintext after padding: | 01001100011011110111001011 1 00101011011010010000001 1010 01011100000111001101110 1010 11011010010000001100100011 0 11110110110000000001 | N.A. |
| Encrypt Plaintext using P-AES and $K_0$ | 11101100101111011011000 1100 10100000000101100011101 0011 01010000000110101000001 0100 01011010000000100101101011 10110000101010100000 | N.A. |
| $K_1$ | 0000000*1*00000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 0000000000000 | N.A. |
| Encrypt Plaintext using P-AES and $K_1$ | 00110100010101101100001 1001 10001011100000100101110 1010 10000011110001011101001 0011 01111011111011100101001 1110 11010100011010000111 | 72 |
| $K_2$ | 00000000000000*1*00000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 0000000000000 | N.A. |
| Encrypt Plaintext using P-AES and $K_2$ | 01110111101010001101110001 0 10110110101011011001001 0001 01111000010010010101010 1000 00010000110111111010000 0101 00000011100100101001 | 67 |
| $K_3$ | 00000000000000000000000*1*000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 | N.A. |

**TABLE 2.** *(Continued.)* Steps for calculating the key avalanche.

| | | |
|---|---|---|
| | 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 0000000000000 | |
| Encrypt Plaintext using P-AES and $K_3$ | 01100011100011010100010 0101 00000110001100111000011 1101 11001001110001100000110 0000 01100011101111010001100 1101 01001001101000110100 | 66 |
| $K_4$ | 00000000000000000000000000000000 0000*1*0000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 0000000000000 | N.A. |
| Encrypt Plaintext using P-AES and $K_4$ | 10110010111110101011001 0101 10101011001001001010110 1000 01011011010011100111000 0110 01010000110101111101011 0110 01001010011000101 | 66 |
| $K_5$ | 00000000000000000000000000000000 0000000000001*0*0000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 0000000000000 | N.A. |
| Encrypt Plaintext using P-AES and $K_5$ | 11010010101100010010010101 0 10100111010010010011101000 0000 11011001100000000101111 0000 11100111101000101001011 1010000101100001010 0 | 57 |
| $K_6$ | 00000000000000000000000000000000 0000000000000000000*1*000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 0000000000000 | N.A. |
| Encrypt Plaintext using P-AES and $K_6$ | 00100000110110000001111101 0 10011010111000011000000000 1111100010011111111000100001 1111110101000000101101101 11 00101100010100111001 | 70 |
| $K_7$ | 00000000000000000000000000000000 00000000000000000000000000000000 0*1*0000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 0000000000000 | N.A. |
| Encrypt Plaintext using P-AES and $K_7$ | 01001111100000001010110101 0 11010011111001111001001011 1 11110000010100111011000110 1001111011100010101110011 10 10010001101000110101 | 57 |

**TABLE 2.** *(Continued.)* **Steps for calculating the key avalanche.**

| | | |
|---|---|---|
| $K_8$ | 00000000000000000000000000<br>00000000000000000000000000<br>00000000*1*0000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>0000000000000 | N.A. |
| Encrypt Plaintext using P-AES and $K_8$ | 11000001110000111101011100<br>11001110000111111010001011<br>10001011101111010100000011<br>01110111010100100100100110<br>00110110100111010000 | 63 |
| $K_9$ | 00000000000000000000000000<br>00000000000000000000000000<br>00000000000000001*0*000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>0000000000000 | N.A. |
| Encrypt Plaintext using P-AES and $K_9$ | 10101000110110011101101010<br>10111001110011110001001010<br>11001100101011100111010101010<br>11111100001001110010111100 0<br>10101011110110001010 | 61 |
| $K_{10}$ | 00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000*1*0<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>0000000000000 | N.A. |
| Encrypt Plaintext using P-AES and $K_{10}$ | 11011111010000111010001 1011<br>00100000010001011011 0100<br>01011101000101111 0001000100<br>00010111000001100010000001 0<br>01111100011110011000 | 56 |

The P-AES performance in terms of encryption and decryption time is almost identical to the AES performance.

### J. COMPARISON WITH THE RELATED WORK

Table 1 summarizes the shortcomings of the existing dynamic ciphers highlighted in the literature review section and how these limitations have been resolved in the P-AES:

## V. CONCLUSION

This research paper proposed a polymorphic version of the Advanced Encryption Standard, namely P-AES. The changes involve manipulating the SubBytes, ShiftRows, and MixColumns transformations and make them key-dependent. The purpose is to change the static nature of the AES and replace it with a dynamic (i.e. polymorphic) cipher that inherits the

**TABLE 3.** **Steps for calculating the Plaintext avalanche.**

| Step Description | Value | Number of Changed Bits |
|---|---|---|
| Master Encryption Key ($K_0$) | 00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>00000000000000000000000000<br>0000000000000 | |
| Initialization Vector | 00000000000000000100000010000<br>00011000001000000010100000 1<br>10000001110000100000001 0010<br>00010100000101100001100000 0<br>11010000111000001111 | |
| $Plaintext_0$: | Lorem ipsum dol | |
| $Plaintext_0$ (Binary): | 01001100011011110111001001 1<br>00101011011010010000001 1010<br>01011100000111001101110101 0<br>11011010010000001100100011 0<br>11110110011 00 | |
| $Plaintext_0$ after padding: | 01001100011011110111001001 1<br>00101011011010010000001 1010<br>01011100000111001101110101 0<br>11011010010000001100100011 0<br>11110110110000000001 | |
| Encrypt $Plaintext_0$ using P-AES and $K_0$ | 11101100101111011011000 1100<br>10100000000101100011101001 1<br>01010000000110101000001010 0<br>01011010000000100101101011 1<br>10110000101010100000 | |
| $Plaintext_1$ | 0100110*1*01101111011100100 11<br>00101011011010010000001 1010<br>01011100000111001101110101 0<br>11011010010000001100100011 0<br>11110110011 00 | |
| Encrypt $Plaintext_1$ using P-AES and $K_0$ | 00100101001000000111011111 0<br>11011101000101101000100000 0<br>01110101000000101101011011 0<br>10001000101100001000100111 0<br>11100110110001111101 | 60 |
| $Plaintext_2$ | 0100110001101111*0*01110010011<br>00101011011010010000001 1010<br>01011100000111001101110101 0<br>11011010010000001100100011 0<br>11110110011 00 | |
| Encrypt $Plaintext_2$ using P-AES and $K_0$ | 00101101101010110011011011 1<br>10101101111111001110001 1001<br>10001001001010110110100 1100<br>10101001111010101000010000 0<br>11000000000111111110 | 69 |
| $Plaintext_3$ | 0100110001101111011100 1*1*011<br>00101011011010010000001 1010<br>01011100000111001101110101 0<br>11011010010000001100100011 0<br>11110110110 00 | |
| Encrypt $Plaintext_3$ using P-AES and $K_0$ | 11101101111000000111001101 0<br>00011000000101100010100 1101<br>11011001100100010111110100 1<br>10001010100110000111001111 1<br>01100010111001011111 | 62 |
| $Plaintext_4$ | 01001100011011110111001001 1<br>0010*0*11011010010000001 1010<br>01011100000111001101110101 0 | |

**TABLE 3.** *(Continued.)* **Steps for calculating the Plaintext avalanche.**

| | | |
|---|---|---|
| | 11011010010000001100100011 0 111101101100 | |
| Encrypt Plaintext$_4$ using P-AES and $K_0$ | 00010011110000110111101001 1 01111100110010000110001111 1 00110011001100010101011110 0 00011111010011001001000111 1 01100010001000110010 | 73 |
| Plaintext$_5$ | 01001100011011110111001100 1 00101011011 0 00100000011010 0 01011100000111001101110101 0 11011010010000001100100011 0 111101101100 | |
| Encrypt Plaintext$_5$ using P-AES and $K_0$ | 00010000011011000010111010 1 11100011010100101001011100 1 01110000010000010101111010 0 01110110101101101101010101 1 00000101011101010001 | 66 |
| Plaintext$_6$ | 01001100011011110111001100 1 00101011011010010000 1 011010 0 01011100000111001101110101 0 11011010010000001100100011 0 111101101100 | |
| Encrypt Plaintext$_6$ using P-AES and $K_0$ | 11100110000101011011001000 0 00001101000011101110011010 0 00110001001000110011011001 0 00101100011101100110000101 0 10000000101101001110 | 60 |
| Plaintext$_7$ | 01001100011011110111001100 1 00101011011010010000001101 0 0 0 01110000011100110111010 1 0 11011010010000001100100011 0 111101101100 | |
| Encrypt Plaintext$_7$ using P-AES and $K_0$ | 10100110010111011100000111 0 00101110110011001110000011 1 10111000110011011010110000 0 10110101100010101100111101 0 00001101111010000100 | 68 |
| Plaintext$_8$ | 01001100011011110111001100 1 00101011011010010000001101 0 0 1 01110000 1 011100110111010 1 0 11011010010000001100100011 0 111101101100 | |
| Encrypt Plaintext$_8$ using P-AES and $K_0$ | 11110101100011011011001001 1 11100011110101000100111111 1 00110001111110100001110000 0 10000011011010000111010010 1 00001111111011111001 | 58 |
| Plaintext$_9$ | 01001100011011110111001100 1 00101011011010010000001101 0 0 1 01110000011100 1 0 011101010 1 0 11011010010000001100100011 0 111101101100 | |
| Encrypt Plaintext$_9$ using P-AES and $K_0$ | 10100111101011010011001111 0 10000001001000001000010010 0 00001010111100011000010011 0 01000011000100001011110000 0 11111101101010111001 | 58 |
| Plaintext$_{10}$ | 01001100011011110111001100 1 00101011011010010000001101 0 0 1 01110000011100110111010 1 0 0 11011010010000001100100011 0 111101101100 | |
| Encrypt Plaintext$_{10}$ using P-AES and $K_0$ | 10010110110010010010111110 1 01010011100011101110111100 1 10100100011101100010100011 1 11110000110011101001110011 1 01110100101000011100 | 71 |

**TABLE 4.** **Example for encryption and decryption using the P-AES.**

| Ser. | Value | Number of Change Bits |
|---|---|---|
| 1 | Generate the Encryption Key | 2815C2510C18312DF278C50B23389 5FBF250F57DE7DEB7C8358C526C 5401EBD0 |
| 2 | Generate the Initialization Vector (IV) | 2D2F782F03BAD230F639810159A6 B2AD |
| 3 | Share the key | - |
| 4 | Get the plaintext for encryption | 4C6F72656D20697073756D20646F6 C6F722073697420616D65742C2063 6F6E73656374657475657220616469 70697363696E6720656C69742E2041 656E65616E20636F6D6D6F646F206 C6967756C61206567657420646F6C 6F722E2041656E65616E206D61737 3612E2043756D20736F63696973206 E61746F7175652070656E617469627 573206574206D61676E69732064697 32070617274757269656E74206D6F6 E7465732C206E61736365747572207 269646963756C7573206D75732E20 446F6E6563207175616D2066656C6 9732C20756C74726963696573206E6 5632C2070656C6C656E7465737175 652065752C207072657469756D2071 7569732C2073656D2E204E756C6C6 120636F6E736571756174206D61737 3612071756973206 56E696D2E20446 F6E65632070656465206A7573746F2 C206672696E67696C6C612076656C 2C20616C6971756574206E65632C2 076756C707574617465 |
| 5 | Padding (if necessary) | 4C6F72656D20697073756D20646F6 C6F722073697420616D65742C2063 6F6E73656374657475657220616469 70697363696E6720656C69742E2041 656E65616E20636F6D6D6F646F206 C6967756C61206567657420646F6C 6F722E2041656E65616E206D61737 3612E2043756D20736F63696973206 E61746F7175652070656E617469627 573206574206D61676E69732064697 32070617274757269656E74206D6F6 E7465732C206E61736365747572207 269646963756C7573206D75732E20 446F6E6563207175616D2066656C6 9732C20756C74726963696573206E6 5632C2070656C6C656E7465737175 652065752C207072657469756D2071 7569732C2073656D2E204E756C6C6 120636F6E736571756174206D61737 3612071756973206 56E696D2E20446 F6E65632070656465206A7573746F2 C206672696E67696C6C612076656C 2C20616C6971756574206E65632C2 076756C707574617465030303 |
| 6 | Calculate the Values: byte_substitution_index Column_mixing_index Row_shifting_index | 0 3 1 |
| 7 | Carry out the encryption | 95B034BEE6F4365B819971AE7582 2C6100E078838B764C3336B86DFB D9DE931449C5B54F77594A362FA2 9DC571939762CE5DC0E6E9A4965 CD03934183AB2B166224552370D0 |

**TABLE 4.** ***(Continued.)*** **Example for encryption and decryption using the P-AES.**

| | | |
|---|---|---|
| | | 99F18709A79ADDAA01F758CE697<br>50EFAD7A77FEA7B896FD2A6A26<br>78F1951E4CDC3154C5A39591F178<br>02B31E07AC0E387406E9056C60A1<br>B608F0C522D9E484B0DA0C07FBE<br>9310D0B4E46391C4089A5D4E38C4<br>C1397D327D02517596510EC55FF9<br>68FD1698FD833B4EFBABD320CF<br>E32CDDD3060F47712268EBB6FE9<br>DD7C94DB018D02D3D0F40F5C9E<br>9D8ED48AB454E9FDE5958C54B81<br>415CB4412C038947DF49ABB432F<br>DECB513853D7AC22862C15E0675<br>E20575256F38A7D91AF87538BE7D<br>A411A474B9996F6F88C974F26A45<br>C93C1F15C55DE17198F3FAC27CD<br>D9E125CE0844E0390E69CEAB70A<br>7D50AB055F1572BCA955E1D0D2<br>A764F08AA10268DB1EA109D76E<br>A98FC062971AF464C8F9EFA55F26<br>BB21243DC3F797DD6BB7E98 |
| 8 | After receiving the plaintext, key and IV, find the values, byte_substitution_index<br>Column_mixing_index<br>Row_shifting_index<br>And carry the decryption | 4C6F72656D20697073756D20646F6<br>C6F722073697420616D65742C2063<br>6F6E736563746574757657220616469<br>70697363696E6720656C69742E2041<br>656E65616E20636F6D6D6F646F206<br>C6967756C61206567657420646F6C<br>6F722E2041656E65616E206D61737<br>3612E2043756D20736F63696973206<br>E61746F7175652070656E617469627<br>573206574206D61676E69732064697<br>32070617274757269656E74206D6F6<br>E7465732C206E61736365747572207<br>269646963756C7573206D75732E20<br>446F6E6563207175616D2066656C6<br>9732C20756C74726963696573206E6<br>5632C2070656C6C656E7465737175<br>652065752C207072657469756D2071<br>7569732C2073656D2E204E756C6C6<br>120636F6E736571756174206D61737<br>361207175697320656E696D2E20446<br>F6E65632070656465206A7573746F2<br>C206672696E67696C6C612076656C<br>2C20616C6971756574206E65632C2<br>076756C707574617465030303 |
| 9 | Remove the padding | 4C6F72656D20697073756D20646F6<br>C6F722073697420616D65742C2063<br>6F6E736563746574757657220616469<br>70697363696E6720656C69742E2041<br>656E65616E20636F6D6D6F646F206<br>C6967756C61206567657420646F6C<br>6F722E2041656E65616E206D61737<br>3612E2043756D20736F63696973206<br>E61746F7175652070656E617469627<br>573206574206D61676E69732064697<br>32070617274757269656E74206D6F6<br>E7465732C206E61736365747572207<br>269646963756C7573206D75732E20<br>446F6E6563207175616D2066656C6<br>9732C20756C74726963696573206E6<br>5632C2070656C6C656E7465737175<br>652065752C207072657469756D2071<br>7569732C2073656D2E204E756C6C6<br>120636F6E736571756174206D61737<br>361207175697320656E696D2E20446<br>F6E65632070656465206A7573746F2<br>C206672696E67696C6C612076656C<br>2C20616C6971756574206E65632C2<br>076756C707574617465 |

AES strengths. As proven, the theoretical strength for every version of the 128 different versions of the P-AES is at least equal to the strength of the conventional AES. However, in the P-AES cipher, a uniform layer of obscurity has been included to hinder any potential attacker from identifying the cipher's exact operation details. The avalanche criterion is investigated and the P-AES scores for the key avalanche and plaintext avalanche were 0.495 and 0.504 respectively. Moreover, the P-AES passed all STS tests. All these changes will not affect the interoperability. The performance assessment shows an insignificant delay in the performance of the proposed P-AES compared to the traditional AES (approximately 1 millisecond per message). In future work, similar manipulations can be suggested to attain better performance and security without increasing the implementation cost.

## APPENDIX I
## EXPERIMENTING THE AVALANCHE CRITERION

This part of the research paper demonstrates the steps taken to calculate the avalanche scores for the P-AES.

To calculate the score of the key avalanche, we shall test the dummy plaintext "Lorem ipsum dol" with the following:

- key (represented in hexadecimal): "$0 \times 00, 0 \times 00, 0 \times 00, 0 \times 00, 0 \times 00, 0 \times 00, 0 \times 00, 0 \times 00, 0 \times 00, 0 \times 00, 0 \times 00, 0 \times 00, 0 \times 00, 0 \times 00, 0 \times 00, 0 \times 00$"
- the Initialization Vector (represented in hexadecimal): "$0 \times 00, 0 \times 01, 0 \times 02, 0 \times 03, 0 \times 04, 0 \times 05, 0 \times 05, 0 \times 07, 0 \times 08, 0 \times 09, 0 \times 0A, 0 \times 0B, 0 \times 0C, 0 \times 0D, 0 \times 0E, 0 \times 0F$".

In the start, the plaintext is encrypted using the P-AES and the initial key. After that, a value of only one bit in the key is changed, and the updated key along with the P-AES is used to encrypt the plaintext. When our new ciphertext is ready, the number of bits which has been changed (compared with the initial ciphered text) is counted. These steps are repeated for times, to calculate the key avalanche score. The steps are depicted in Table 2.

Given that the count of bits of the ciphertext is always 128, and the number of iterations is 10, the avalanche score is calculated as follows:

$$\frac{\sum_{i=1}^{10} number\ of\ changedbitsiniteration\,(i)}{10 \times 128} \approx 0.496$$

The same steps are carried out to calculate the plaintext avalanche. However, instead of changing one bit in the key, one bit in the plaintext is changed. These steps are depicted below in Table 3:

Given that the count of bits of the ciphertext is always 128, and the number of iterations is 10, the plaintext avalanche score is calculated as follows:

$$\frac{\sum_{i=1}^{10} number\ of\ changedbitsiniteration\,(i)}{10 \times 128} \approx 0.504$$

## APPENDIX II
## EXAMPLE FOR THE CIPHER OPERATION

Assume we have the following plaintext "*Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate*, " and we want to encrypt and decrypt it using the P-AES. The steps presented in table 4 are carried out to perform these operations.

## REFERENCES

[1] S. Aljawarneh and M. B. Yassein, "A multithreaded programming approach for multimedia big data: Encryption system," *Multimedia Tools Appl.*, vol. 77, no. 9, pp. 10997–11016, 2018.
[2] C. Qin, Z. Qian, J. Wang, and X. Zhang, *New Advances of Privacy Protection and Multimedia Content Security for Big Data and Cloud Computing*. Bangalore, India: Taylor & Francis, 2018.
[3] R. von Solms and J. van Niekerk, "From information security to cyber security," *Comput. Secur.*, vol. 38, pp. 97–102, Oct. 2013.
[4] I. Ali, S. Sabir, and Z. Ullah, "Internet of Things security, device authentication and access control: A review," 2019, *arXiv:1901.07309*. [Online]. Available: http://arxiv.org/abs/1901.07309
[5] P. P. Tayade and P. Vijayakumar, "Enhancement of security and confidentiality for D2D communication in LTE-advanced network using optimised protocol," in *Wireless Communication Networks and Internet Things*. Singapore: Springer, 2019, pp. 131–139.
[6] A. M. Ahmadian and M. Amirmazlaghani, "A novel secret image sharing with steganography scheme utilizing optimal asymmetric encryption padding and information dispersal algorithms," *Signal Process., Image Commun.*, vol. 74, pp. 78–88, May 2019.
[7] A. Altigani and B. Barry, "A hybrid approach to secure transmitted messages using advanced encryption standard (AES) and word shift coding protocol," in *Proc. Int. Conf. Comput., Electr. Electron. Eng. (ICCEEE)*, Aug. 2013, pp. 134–139.
[8] A. Altigani, S. Hasan, B. Barry, and S. M. Shamsuddin, "Key-dependent advanced encryption standard," in *Proc. Int. Conf. Comput., Control, Electr., Electron. Eng. (ICCCEEE)*, Aug. 2018, pp. 1–5.
[9] C. K. B. Röllgen, *Block Cipher*. Menlo Park, CA, USA: Google, 2010.
[10] A. Altigani and S. M. Shamsuddin, "Highlighting issues relevant to encryption algorithms and security schemes," *Indian J. Sci. Technol.*, vol. 10, p. 39, Oct. 2017.
[11] C. Clavier, Q. Isorez, D. Marion, and A. Wurcker, "Complete reverse-engineering of AES-like block ciphers by SCARE and FIRE attacks," *Cryptography Commun.*, vol. 7, no. 1, pp. 121–162, Mar. 2015.
[12] M. S. Taha, M. S. M. Rahim, S. A. Lafta, M. M. Hashim, and H. M. Alzuabidi, "Combination of steganography and cryptography: A short survey," in *Proc. IOP Conf. Ser., Mater. Sci. Eng.*, vol. 518, no. 5. Bristol, U.K.: IOP Publishing, 2019, p. 052003.
[13] P. Kumar and S. B. Rana, "Development of modified AES algorithm for data security," *Optik*, vol. 127, no. 4, pp. 2341–2345, Feb. 2016.
[14] J. Saleem and M. Hammoudeh, "Defense methods against social engineering attacks," in *Computer and Network Security Essentials*. Cham, Switzerland: Springer, 2018, pp. 603–618.
[15] A. Abdulgader, M. Ismail, N. Zainal, and T. Idbeaa, "Enhancement of AES algorithm based on chaotic maps and shift operation for image encryption," *J. Theor. Appl. Inf. Technol.*, vol. 71, no. 1, pp. 2005–2015, 2015.
[16] P. Singhai and A. Shrivastava, "An efficient image security mechanism based on advanced encryption standard," *Int. J. Adv. Technol. Eng. Explor.*, vol. 2, no. 13, p. 175, 2015.
[17] N. Nisha and N. Singh, "A hybrid approach of AES and file encryption to enhance the cloud security," *Int. J. Comput. Technol.*, vol. 14, no. 11, pp. 6250–6257, Aug. 2015.
[18] S. Suri and R. Vijay, "An AES–CHAOS-based hybrid approach to encrypt multiple images," in *Recent Developments in Intelligent Computing, Communication and Devices*. Singapore: Springer, 2017, pp. 37–43.
[19] V. Ramaswamy, "Making AES stronger: AES with key dependent S-box," *Int. J. Comput. Sci. Netw. Secur.*, vol. 8, no. 9, pp. 388–398, 2008.
[20] R. Hosseinkhani and H. H. S. Javadi, "Using cipher key to generate dynamic S-box in AES cipher system," *Int. J. Comput. Sci. Secur.*, vol. 6, no. 1, pp. 19–28, 2012.
[21] K. Kazlauskas and J. Kazlauskas, "Key-dependent S-box generation in AES block cipher system," *Informatica*, vol. 20, no. 1, pp. 23–34, Jan. 2009.
[22] J. Daemen and V. Rijmen, "AES proposal: Rijndael," in *Proc. 1st Adv. Encryption Conf.*, CA, USA, 1998, pp. 1–45.
[23] A. Altigani, S. Hasan, S. M. Shamsuddin, and B. Barry, "A multi-shape hybrid symmetric encryption algorithm to thwart attacks based on the knowledge of the used cryptographic suite," *J. Inf. Secur. Appl.*, vol. 46, pp. 210–221, Jun. 2019.
[24] H. Tang, Q. T. Sun, X. Yang, and K. Long, "A network coding and DES based dynamic encryption scheme for moving target defense," *IEEE Access*, vol. 6, pp. 26059–26068, 2018.
[25] W. Stallings, *Cryptography and Network Security: Principles and Practice*. London, U.K.: Pearson, 2016.
[26] H. N. Noura, A. Chehab, and R. Couturier, "Efficient & secure cipher scheme with dynamic key-dependent mode of operation," *Signal Process., Image Commun.*, vol. 78, pp. 448–464, Dec. 2019.
[27] A. ALTIGANI, S. HASAN, and B. BARRY, "The need for polymorphic encryption algorithms: A review paper," *J. Theor. Appl. Inf. Technol., Rev. Paper*, vol. 98, no. 3, p. 18, Feb. 2020.
[28] M. Dworkin, *Recommendation for Block Cipher Modes of Operation Methods and Techniques*. Gaithersburg, MD, USA: National Inst of Standards and Technology, 2001.
[29] A. Altigani, M. Abdelmagid, and B. Barry, "Analyzing the performance of the advanced encryption standard block cipher modes of operation: Highlighting the national institute of standards and technology recommendations," *Indian J. Sci. Technol.*, vol. 9, no. 28, Jul. 2016.
[30] W. E. Burr, "Selecting the advanced encryption standard," *IEEE Secur. Privacy*, vol. 99, no. 2, pp. 43–52, Mar. 2003.
[31] J. Daemen, "Cipher and hash function design strategies based on linear and differential cryptanalysis," Ph.D. dissertation, Dept. Comput. Secur. Ind. Cryptogr., KU Leuven, Leuven, Belgium, Mar. 1995.
[32] M. Khan and Z. Asghar, "A novel construction of substitution box for image encryption applications with gingerbreadman chaotic map and s8 permutation," *Neural Comput. Appl.*, vol. 29, no. 4, pp. 993–999, Feb. 2018.
[33] Y. Zhou and D. Feng, "Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing," in *Proc. IACR*, Sep. 2005, p. 388.
[34] J. Soto and J. Soto, *Randomness Testing of the Advanced Encryption Standard Candidate Algorithms*. Gaithersburg, MD, USA: National Institute of Technology Administration, 1999.
[35] L. Bassham *et al.*, "SP 800-22 Rev. 1a. A statistical test suite for random and pseudorandom number generators for cryptographic applications," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep., 2010.
[36] M. Sýs, "Faster randomness testing with the NIST statistical test suite," in *Proc. Int. Conf. Secur., Privacy, Appl. Cryptogr. Eng.* Springer, 2014, pp. 272–284.
[37] M. Sys at al, "On the interpretation of results from the NIST statistical test suite," *Romanian J. Inf. Sci. Technol.*, vol. 18, no. 1, pp. 18–32, 2015.
[38] H. N. Noura, A. Chehab, and R. Couturier, "Overview of efficient symmetric cryptography: Dynamic vs static approaches," in *Proc. 8th Int. Symp. Digit. Forensics Secur. (ISDFS)*, Jun. 2020, pp. 1–6.
[39] L. Chen and R. Zhang, "A key-dependent cipher DSDP," in *Proc. Int. Symp. Electron. Commerce Secur.* Cham, Switzerland: Springer, 2008, pp. 310–313. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-12060-7_18
[40] R. Zhang and L. Chen, "A block cipher using key-dependent S-box and P-boxes," in *Proc. IEEE Int. Symp. Ind. Electron.*, Jun. 2008, pp. 1463–1468.

**ABDELRAHMAN ALTIGANI** received the bachelor's degree in mathematics and the master's degree in computer science from the Faculty of Mathematical Sciences, University of Khartoum, Sudan. He is currently pursuing the Ph.D. degree. He has 11 publications in international refereed journals and conferences. He received the award of the best research article in the computing track at the ICCEEE2013 conference. During his studies, he received the International Ph.D. Fellowship scholarship thrice, in recognition of his outstanding academic performance. He participated in reviewing a number of research articles for many journals. His area of interest includes symmetric encryption.

**SHAFAATUNNUR HASAN** received the bachelor's degree in computer science (artificial intelligence) from Universiti Malaya, and the master's degree in computer science and the Ph.D. degree in GPU-based machine learning from Universiti Teknologi Malaysia. She used to work at Shimano Components Malaysia for five years as a Planning and Analytics Officer dealing with fishing tackles components. She is currently doing her research on GPU-based Hadoop technology for big data and a Pelangi Deep Learning Analytics for Big Data Research and Development. She has developed GPUMLiB: Machine Learning Library for Big Data Processing, an open source library using cuda language to deal with Nth dimensional space under GPU platform (http://gpumlib.sourceforge.net/), with Professor Noel Lopes from Portugal. Her experiences in solving industries problems ranging from fundamental issues to practical applications, such as Polymer Degradation Analytics, Retail Sport Analytics, Desk Help Service Analytics, Malware Analytics, and others. She has published many journals and conference papers. Her research interests include big data computing, machine learning, soft computing, cuda programming, hadoop engineering, and big data platform. She is also the Managing Editor of *Scopus-Index Journal*, *International Journal of Advances in Soft Computing and Its Applications* (http://www.home.ijasca.com), and reviewer of many international reputable journals related to her fields.

**BAZARA BARRY** (Member, IEEE) was born in Sudan. He received the B.Sc. (Hons.) and M.Sc. degrees in computer science from the University of Khartoum, and the Ph.D. degree in cyber security from the University of Cape Town in South Africa, in 2009.

He has been active in both academia and industry leading research, supervising students, and advising businesses. He is currently a Principal Consultant with Barnardos Australia and a Senior Lecturer with Macquarie University. He has published more than 30 journal articles, book chapters, and conference papers.

Dr. Barry has received several honours and awards for best research and industry contributions.

**SHIRAZ NASERELDEN** received the bachelor's and master's degrees in mathematics from the Faculty of Mathematical Sciences, University of Khartoum, Sudan. She is currently pursuing the Ph.D. degree from the Universiti Teknologi Malaysia. She is a Teaching Staff with Imam Abdulrahman Bin Faisal University. Her research interests include cryptology and fuzzy mathematics. She has a number of publications in international refereed journals.

**MUAWIA A. ELSADIG** received the bachelor's degree in computer engineering, the M.Sc. degree in computer network, and the Ph.D. degree in computer science. His research interests include in the area of information security, network security, cybersecurity, wireless sensor networks, bioinformatics, and information extraction; ranging from theory to design to implementation. He worked for different accredited international universities and has many publications at recognized international journals and conferences.

**HUWAIDA T. ELSHOUSH** received the bachelor's degree in computer science (division 1), the master's degree in computer science, and the Ph.D. degree in information security from the Faculty of Mathematical Sciences, University of Khartoum, Sudan, in 1994, 2001, and 2012, respectively.

Her M.Sc. dissertation dealt with Frame Relay Security. She is currently an Associate Professor with the Computer Science Department, Faculty of Mathematical Sciences, University of Khartoum, where she also acting as the Head of Research office. She has more than 20 publications and some of her publications appeared in *Applied Soft Computing Journal* (Elsevier), *PLOS One Journal*, and Springer book chapters. Her research interests include the information security, cryptography, steganography, and intrusion detection systems. She is a Reviewer of many international reputable journals related to her fields, including *Applied Soft Computing Journal* (Elsevier).

Dr. Elshoush's awards and honors include the second-place prize in the ACM Student Research Competition SRC - SAC in 2013 in Coimbra, Portugal. Her article An Improved Framework for Intrusion Alert Correlation has been awarded the Best Student Paper Award of the 2012 International Conference of Information Security and Internet Engineering (ICISIE) in WCE 2012. Other prizes were the best student during the five years of undergraduate study.